ГЕННАДИЙ САМКОВ

bhv®

jQuery

Сборник рецептов

2-е издание

МЕТОДЫ
И ВСПОМОГАТЕЛЬНЫЕ
ФУНКЦИИ БИБЛИОТЕКИ jQuery

ВЗАИМОДЕЙСТВИЕ jQuery
И AJAX

НАДСТРОЙКА UI jQuery

САМЫЕ ПОПУЛЯРНЫЕ
ПЛАГИНЫ ДЛЯ jQuery

PRO
ПРОФЕССИОНАЛЬНОЕ
ПРОГРАММИРОВАНИЕ

+

Геннадий Самков

jQuery
Сборник рецептов

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.068+800.92 jQuery
ББК 32.973.26-018.1
С17

Самков Г. А.

С17 jQuery. Сборник рецептов. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. — 416 с.: ил. + CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0732-5

Книга является сборником решений наиболее часто встречающихся задач при веб-программировании пользовательских интерфейсов с использованием библиотеки jQuery. Рассмотрены практически все методы и вспомогательные функции jQuery, в том числе обеспечивающие взаимодействие jQuery и AJAX. Подробно рассказано о надстройке UI jQuery. Приведено большое количество примеров использования плагинов для jQuery — создание графиков и диаграмм, фотогалерей, меню, работа с таймерами и cookies, обработка табличных данных и др. Во втором издании в примерах используется библиотека jQuery версий 1.4.4 и 1.5.2, а также надстройка UI jQuery 1.8.9. Компакт-диск содержит примеры из книги, файлы библиотеки jQuery 1.4.4 и 1.5.2, файлы надстройки UI jQuery 1.8.9, а также файлы расширений сторонних разработчиков.

Для веб-программистов

УДК 681.3.068+800.92 jQuery
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.06.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 33,54.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0732-5

Оглавление

Введение

Предлагаемая книга представляет собой сборник примеров, поясняющих возможности большинства методов, предоставляемых API популярной JavaScript-библиотеки jQuery. Много внимания уделено UI jQuery — надстройке над библиотекой, применяемой при проектировании пользовательских интерфейсов. Также в книгу включены подробные описания и рекомендации по использованию наиболее востребованных плагинов, которые пригодятся при решении задач, часто встречающихся в программировании.

Предполагается, что читатель знаком с CSS, HTML и основами JavaScript. Кроме того, потребуются некоторые начальные знания PHP — языка программирования серверных сценариев, который необходим в некоторых примерах, посвященных организации взаимодействия "клиент — сервер" с применением технологии AJAX.

Книга может служить не только учебником, но и справочником по библиотеке jQuery и надстройке UI jQuery.

Структура книги

Книга содержит две части и приложение.

В *части I* решения задач представлены так, чтобы помочь читателю на простых примерах освоить подавляющее большинство методов библиотеки jQuery, имеющихся в распоряжении разработчика. Подробно освещены такие вопросы, как:

- выбор элементов — базовые правила, иерархия, фильтры, селекторы атрибутов, фильтры элементов форм;
- работа с атрибутами и содержимым элементов, работа с данными в элементах форм;
- создание визуальных эффектов, в том числе с помощью надстройки UI jQuery;
- работа с CSS-свойствами элементов;
- работа с данными в jQuery, в том числе с использованием HTML 5;

- манипуляции элементами — изменение содержимого, вставка, замена, удаление и копирование элементов DOM;
- перемещение по элементам DOM — поиск и фильтрация элементов, управление цепочками вызовов функций;
- работа с событиями — назначение, удаление и вызов событий для мыши, клавиатуры, браузера и форм;
- взаимодействие jQuery и AJAX;
- некоторые полезные вспомогательные функции и методы jQuery.

В *части II* приведены решения на основе наиболее популярных расширений для библиотеки jQuery, в том числе рассмотрен официальный пакет расширений UI jQuery. Подробно рассматриваются:

- вертикальное и горизонтальное многоуровневое меню на основе плагина jQuery Superfish;
- организация работы с данными, представленными в табличной форме, рассматривается на примере плагина jqGrid;
- возможности реализации графиков и диаграмм на страницах веб-сайта демонстрируются на примере плагина jqPlot;
- работа с AJAX-формами — плагины jQuery Form (построение AJAX-формы), jQuery Validate (проверка данных в AJAX-форме) и FileUpload (загрузка файлов на сервер);
- фотогалереи для веб-сайтов рассматриваются на примере плагина FancyBox;
- некоторые полезные плагины — плагин jQuery Cookie (установка и считывание cookie), плагин jQuery ClueTip (всплывающие подсказки), плагин jQuery Timers (управление таймерами);
- виджеты надстройки UI jQuery — Accordion (раскрывающееся меню), Datepicker (выбор даты), Dialog — (диалоговое окно), ProgressBar (шкала загрузки), Slider (шкала с бегунком), Tabs (организация переключения вкладок), Button (стилизация и управление поведением кнопок и некоторых элементов форм), Autocomplete (список подсказок);
- надстройка UI jQuery — взаимодействие с элементами страницы: Draggable (перемещение элементов), Droppable ("сброс" элементов), Resizable (изменение размеров элементов), Selectable (выбор элементов), Sortable (сортировка элементов).

В *приложении* описан компакт-диск, прилагаемый к книге.

Как работать с книгой

Книга в основном ориентирована на разработчика, работающего с операционной системой Windows, но пользователь UNIX также сможет выполнить на своем компьютере все примеры.

В ходе чтения следует выполнять на своем компьютере все примеры, описываемые в книге. Рекомендуем читателю самостоятельно изменять и переделывать каждый пример, чтобы лучше понять, как он работает.

Автор приложил все усилия, чтобы изложить материал с наибольшей точностью, но не исключает возможности ошибок и опечаток. Автор также не несет ответственности за последствия использования сведений, изложенных в книге.

Источники информации

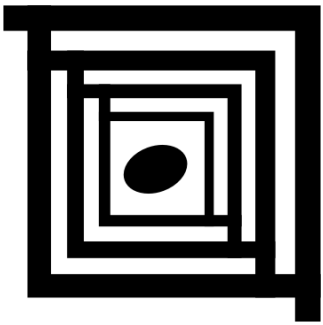
В книге невозможно охватить все вопросы, и читателю наверняка потребуются дополнительные сведения, например, из Интернета. Кроме того, могут изменяться версии программного обеспечения, рассматриваемого в книге. Вот адреса, которыми вы можете воспользоваться:

- **http://jquery.com/** — официальный сайт библиотеки jQuery (англ.);
- **http://api.jquery.com/** — оригинальная документация по библиотеке jQuery (англ.);
- **http://jqueryui.com/** — официальный сайт UI jQuery (англ.);
- **http://www.linkexchanger.su** — блог автора книги, содержит много статей с примерами использования библиотеки jQuery (рус.);
- **http://www.linkexchanger.su/forum/** — форум, посвященный вопросам разработки с применением библиотеки jQuery, UI jQuery и плагинов для библиотеки (рус.);
- **http://slyweb.ru/jquerymain/** — перевод документации jQuery (рус.);
- **http://plugins.jquery.com/** — на сайте представлено большое количество разнообразных плагинов для библиотеки jQuery (англ.);
- **http://users.tpg.com.au/j_birch/plugins/superfish/** — страница плагина jQuery Superfish (англ.);
- **http://trirand.com/blog/jqgrid/jqgrid.html** — демо-галерея плагина jqGrid (англ.);
- **http://www.trirand.com/jqgridwiki/doku.php?id=wiki:jqgriddocs** — официальная документация на плагин jqGrid (англ.);
- **http://www.jqplot.com** — сайт плагина jqPlot (англ.);
- **http://malsup.com/jquery/form/** — страница плагина jQuery Form (англ.);
- **http://bassistance.de/jquery-plugins/jquery-plugin-validation/** — страница плагина jQuery Validation (англ);
- **http://www.uploadify.com** — сайт плагина jQuery Uploadify (англ.);
- **http://fancybox.net** — сайт плагина jQuery FancyBox (англ.);
- **http://plugins.learningjquery.com/cluetip/** — страница плагина jQuery ClueTip (англ.);

- **http://www.stilbuero.de/2006/09/17/cookie-plugin-for-jquery/** — страница плагина jQuery Cookie (англ.);
- **http://jquery.offput.ca/timers/** — страница плагина jQuery Timers (англ.);
- **http://firebug.ru** — сайт, посвященный Firebug, замечательному средству отладки JavaScript-кода (рус.).

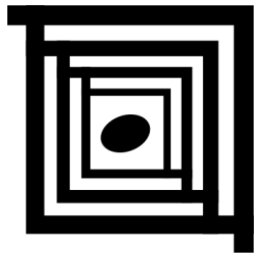
Благодарности

Автор приносит искренние благодарности Владимиру Кондратьеву за его готовность поделиться своим богатым практическим опытом работы с плагинами, Игорю Пирогову и Андрею Зайцеву — за их поддержку и квалифицированные технические консультации, Эдуарду Аметову — за консультации по вопросам работы с изображениями.



ЧАСТЬ I

Методы библиотеки jQuery



ГЛАВА 1

Выбор элементов

А для чего вообще нужно выбирать элементы? Ответ простой — для того, чтобы как-то на них воздействовать. Например, можно изменить атрибуты или CSS-свойства элементов, меняя, таким образом, их визуальное представление, изменять содержимое этих элементов, связывать с ними определенные события и т. д.

Можно выбрать как один элемент, так и множество элементов. Независимо от того, сколько именно элементов выбрано, мы будем рассматривать это как набор элементов, называя его объектом jQuery.

1.1. Базовые правила

ЗАДАЧА

Необходимо отыскать абсолютно все элементы веб-страницы.

Решение

Используем селектор * для решения этой задачи (листинг 1.1.1).

Листинг 1.1.1. Использование селектора *

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  alert($("*").length);
});
```

```
</script>
</head>
<body>
<ul>
  <li></li>
  <li></li>
</ul>
<p></p>
<div><span></span></div>
</body>
</html>
```

Обсуждение

Чтобы рассмотренный пример не выглядел совсем скучно, и можно было понять, что он действительно работает, мы не только выбрали в объект jQuery все элементы веб-страницы, но также подсчитали их число и вывели его в окне предупреждения. Поскольку контекстом в приведенном примере является объект `document`, то в набор попадут элементы не только из `body`, но и из `head`. В наборе также окажутся элементы `script` и т. п.

ЗАДАЧА

Необходимо отыскать все элементы веб-страницы, но только в контексте `body`, исключив, таким образом, все прочие элементы.

Решение

Для решения этой задачи также воспользуемся селектором `*`, но в качестве второго аргумента явно передадим контекст (листинг 1.1.2).

Листинг 1.1.2. Использование селектора *

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  alert($("*",document.body).length);
});
</script>
</head>
```

```
<body>
<ul>
  <li></li>
  <li></li>
</ul>
<p></p>
<div><span></span></div>
</body>
</html>
```

Обсуждение

Мы точно так же вывели в окно предупреждения число выбранных элементов. Заметили разницу? 6 против 13 в примере из листинга 1.1.1. В набор попали только элементы, находящиеся внутри `body`. Сам элемент `body` в набор не попал.

ЗАДАЧА

Необходимо отыскать элемент по известному значению его атрибута `id`.

Решение

Для поиска элемента по значению его атрибута `id` воспользуемся селектором идентификатора (листинг 1.1.3).

Листинг 1.1.3. Использование селектора `#id`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#myDiv").css("border","1px solid #f00");
});
</script>
<style type="text/css">
div {
  float:left;
  width:150px;
  height:150px;
  border:1px solid #00f;
  margin:2px;
}
```

```
</style>
</head>
<body>
<div class="myDiv"></div>
<div></div>
<div id="myDiv"></div>
<div><span></span></div>
<div id="otherDiv"></div>
</body>
</html>
```

Обсуждение

Отыскав элемент, который имеет значение идентификатора `myId`, мы применили к нему метод `css()`, добавив выбранному элементу красную рамку шириной в 1 px, чтобы убедиться в том, что селектор действительно обнаружил нужный элемент.

> ***Примечание***
>
> Не забывайте важный момент — в пределах документа значение атрибута `id` обязательно должно быть уникальным.

ЗАДАЧА

Необходимо отыскать элемент по значению атрибута `id`, в который входят специфические символы, такие как точка или квадратные скобки. Проблема состоит в том, что эти символы имеют специальное значение в CSS.

Решение

Снова воспользуемся селектором идентификатора, но перед специальными символами поставим два обратных слэша подряд (листинг 1.1.4).

Листинг 1.1.4. Использование селектора `#id`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#my\\.Div").css("border","1px solid #f00");
  $("#my\\[Div\\]").css("border","1px solid #0f0");
});
```

```
</script>
<style type="text/css">
div {
  float:left;
  width:150px;
  height:150px;
  border:1px solid #00f;
  margin:2px;
}
</style>
</head>
<body>
<div></div>
<div id="my.Div"></div>
<div><span></span></div>
<div id="my[Div]"></div>
</body>
</html>
```

Обсуждение

Чтобы убедиться, что этот прием работает корректно, мы с помощью метода `css()` устанавливаем для найденных элементов различный цвет рамок. Элементу с идентификатором `my.Div` мы установили красную рамку, а элементу с идентификатором `my[Div]` — зеленую.

ЗАДАЧА

Необходимо отыскать на веб-странице все элементы определенного типа, например все `div`.

Решение

Для решения задачи достаточно воспользоваться селектором `element` (листинг 1.1.5).

Листинг 1.1.5. Использование селектора `element`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
```

```
$(function(){
  $("div").css("border","2px dotted #f00");
});
</script>
<style type="text/css">
div, p { width:150px; height:150px; border:1px solid #00f; margin:2px; }
</style>
</head>
<body>
<div></div>
<p></p>
<div></div>
<p></p>
<div></div>
</body>
</html>
```

Обсуждение

Обратите внимание — с помощью нескольких CSS-правил мы задали совершенно одинаковое оформление как для элементов `p`, так и для `div`. Но мы добавили и JavaScript-код и с помощью селектора `element` нам удалось выбрать все элементы `div`, имеющиеся на веб-странице. Используя метод `css()`, мы установили для выбранных элементов рамки из точек красного цвета шириной 2 px.

ЗАДАЧА

Необходимо отыскать элемент (или элементы) по имени класса.

Решение

Для решения задачи применяется селектор, который в точности повторяет синтаксис `css` (листинг 1.1.6).

Листинг 1.1.6. Использование селектора `.class`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $(".test").css("border","1px solid #f00");
});
</script>
```

```
<style type="text/css">
div, p, ul { border:1px solid #00f; margin:2px; }
</style>
</head>
<body>
<div>div</div>
<p class="test">p class="test"</p>
<ul class="test">
  <li>li списка ul class="test"</li>
  <li>li списка ul class="test"</li>
</ul>
<p>p</p>
<div class="test">div class="test"</div>
</body>
</html>
```

Обсуждение

С помощью селектора `.class` мы выбрали все элементы, которые имеют значение `test` атрибута `class`. Для наглядности вновь применяем метод `css()`, чтобы установить для выбранных элементов рамки красного цвета шириной 1 px.

ЗАДАЧА

Используя только один селектор, необходимо выбрать разные элементы по имени класса, значению идентификатора и названию элемента.

Решение

Для решения задачи применяется составной селектор (листинг 1.1.7).

Листинг 1.1.7. Использование составного селектора

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-1-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#item2, p.test, div").css("background-color","#ccc");
});
</script>
<style type="text/css">
```

```
div,p,li { border:1px solid #00f; margin:2px; }
</style>
</head>
<body>
<div>Элемент div</div>
<p>Элемент p</p>
<div>Элемент div</div>
<p class="test">Элемент p class="test"</p>
<p>Элемент p</p>
<ul id="test">
  <li id="item1">Первый пункт списка id="item1"</li>
  <li id="item2">Второй пункт списка id="item2"</li>
</ul>
<p class="test">Элемент p class="test"</p>
<div>Элемент div</div>
</body>
</html>
```

Обсуждение

С помощью составного селектора мы смогли сразу выбрать все элементы `div`, элементы `p` с классом `test` и элемент `li` по его идентификатору `item2`. Для всех элементов, попавших в этот набор, устанавливаем серый цвет фона методом `css()`.

1.2. Выбор элементов с учетом иерархии

ЗАДАЧА

Необходимо отыскать элементы, являющиеся потомками какого-либо элемента.

Решение

Селектор `ancestor descendant` поможет решить эту задачу (листинг 1.2.1).

Листинг 1.2.1. Использование селектора `ancestor descendant`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
```

```
form {
  border:2px green solid;
  padding:2px;
  margin:0;
  background:#efe;
}
div {
  color:red;
}
fieldset {
  margin:1px;
  padding:3px;
}
</style>
<script type="text/javascript">
$(function(){
  $("form input").css("border", "2px dotted brown");
});
</script>
</head>
<body>
<form>
  <div>Форма заключена в зеленую рамку</div>
  <label>Ребенок:</label>
  <input type="text" name="name" />
  <fieldset>
    <label>Внук:</label>
    <input type="text" name="newsletter" />
  </fieldset>
</form>
Сестринский элемент по отношению к форме: <input type="text" name="none" />
</body>
</html>
```

Обсуждение

В HTML-коде, приведенном в листинге 1.2.1, присутствуют три элемента `input`. Наша задача — отыскать только те из них, которые являются наследниками элемента `form`. Указав в селекторе выражение `form input`, мы легко находим только нужные нам элементы и отмечаем их коричневой рамкой. Элемент `input`, расположенный вне пределов тега `<form>`, такой рамкой выделен не будет.

ЗАДАЧА

Необходимо отыскать элементы, являющиеся прямыми потомками какого-либо элемента.

Решение

Используем селектор `parent > child` для решения этой задачи (листинг 1.2.2).

Листинг 1.2.2. Использование селектора `parent > child`

```
<script type="text/javascript">
$(function(){
  $("form > input").css("border", "2px dotted brown");
});
</script>
```

Обсуждение

В листинге 1.2.2 приведен только JavaScript-код, т. к. все остальное осталось без изменений. Поскольку необходимо отыскать только элементы `input`, являющиеся прямыми наследниками `form`, мы указываем в селекторе выражение `form > input`. Коричневой рамкой в итоге будет отмечен только первый элемент `input`.

ЗАДАЧА

Необходимо отыскать элементы, следующие непосредственно за известным элементом.

Решение

Решить эту задачу позволит селектор `prev + next` (листинг 1.2.3).

Листинг 1.2.3. Использование селектора `prev + next`

```
<script type="text/javascript">
$(function(){
  $("label + input").css("border", "2px dotted brown");
});
</script>
```

Обсуждение

В листинге 1.2.3 приведен только JavaScript-код. Мы указали в селекторе выражение `prev + next` и, таким образом, нашли элементы `input`, которые следуют непосредственно за элементами `label`. Коричневая рамка досталась лишь двум элементам `input` внутри формы, поскольку только перед ними есть элемент `label`. Элемент `input`, находящийся вне тега `form`, не имеет перед собой элемента `label`, поэтому и не был выбран.

ЗАДАЧА

Необходимо отыскать все элементы, располагающиеся на одном уровне с некоторым известным элементом, иными словами — сестринские элементы.

Решение

Используем селектор `prev ~ siblings` для решения этой задачи (листинг 1.2.4).

Листинг 1.2.4. Использование селектора `prev ~ siblings`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-2-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
form {
  border:2px green solid;
  padding:2px;
  margin:0;
  background:#efe;
}
div {
  color:red;
}
fieldset {
  margin:1px;
  padding:3px;
}
</style>
<script type="text/javascript">
$(function(){
  $("label ~ fieldset").css("border", "2px dotted brown");
});
</script>
</head>
<body>
<form>
  <div>Форма заключена в зеленую рамку</div>
  <label>Ребенок:</label>
  <input type="text" name="name" />
  <fieldset>
    <label>Внук:</label>
```

```
    <input type="text" name="newsletter" />
  </fieldset>
</form>
Сестринский элемент по отношению к форме: <input type="text" name="none" />
<fieldset>
  <input type="text" name="email" />
</fieldset>
</body>
</html>
```

Обсуждение

В листинге 1.2.4 мы немного изменили HTML-код, добавив еще один элемент `fieldset`, внутри которого находится элемент `input`. Указываем в селекторе выражение `label ~ fieldset` и видим, что коричневой рамкой отмечен только тот элемент `fieldset`, который находится внутри формы, поскольку только он, в отличие от `fieldset`, находящегося вне `form`, является сестринским элементом по отношению к `label`.

1.3. Основные фильтры

ЗАДАЧА

Необходимо установить серый цвет фона только для первой строки в таблице.

Решение

Решить задачу поможет фильтр `:first` (листинг 1.3.1).

Листинг 1.3.1. Использование фильтра `:first`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
table {
  width:400px;
}
</style>
<script type="text/javascript">
```

```
$(function(){
  $("tr:first").css("background-color","#ccc");
});
</script>
</head>
<body>
<table>
  <tr>
    <td>1-1</td><td>1-2</td><td>1-3</td><td>1-4</td>
  </tr>
  <tr>
    <td>2-1</td><td>2-2</td><td>2-3</td><td>2-4</td>
  </tr>
  <tr>
    <td>3-1</td><td>3-2</td><td>3-3</td><td>3-4</td>
  </tr>
  <tr>
    <td>4-1</td><td>4-2</td><td>4-3</td><td>4-4</td>
  </tr>
</table>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 1.3.1, не представляет собой ничего интересного — обычная таблица. Посмотрим на JavaScript-код. Указав в селекторе выражение `tr:first`, мы смогли установить серый цвет фона только для первой строки таблицы.

ЗАДАЧА

Необходимо установить серый цвет фона только для последней строки в таблице.

Решение

Для решения задачи применим фильтр `:last` (листинг 1.3.2).

Листинг 1.3.2. Использование фильтра `:last`

```
<script type="text/javascript">
$(function(){
  $("tr:last").css("background-color","#ccc");
});
</script>
```

Обсуждение

Рассмотрим только JavaScript-код из листинга 1.3.2, поскольку HTML-код остался без изменений. На этот раз мы указали в селекторе выражение `tr:last` и установили серый цвет фона уже для последней строки в таблице.

ЗАДАЧА

Необходимо установить серый цвет фона только для четных строк в таблице.

Решение

Для решения задачи воспользуемся фильтром `:even` (листинг 1.3.3).

Листинг 1.3.3. Использование фильтра `:even`

```
<script type="text/javascript">
$(function(){
  $("tr:even").css("background-color","#ccc");
});
</script>
```

Обсуждение

Нам стоит только указать выражение `tr:even` в коде, который приведен в листинге 1.3.3, и все четные строки таблицы (отсчет будет идти от нуля) станут серыми.

ЗАДАЧА

Необходимо установить серый цвет фона только для нечетных строк в таблице.

Решение

Для решения задачи используем фильтр `:odd` (листинг 1.3.4).

Листинг 1.3.4. Использование фильтра `:odd`

```
<script type="text/javascript">
$(function(){
  $("tr:odd").css("background-color","#ccc");
});
</script>
```

Обсуждение

Так же просто обстоит дело с отысканием всех нечетных строк. Мы найдем их, всего лишь записав выражение `tr:odd` в коде, который приведен в листинге 1.3.4.

ЗАДАЧА

Необходимо отыскать шестую по счету ячейку таблицы.

Решение

Для решения задачи воспользуемся фильтром `:eq(index)` (листинг 1.3.5).

Листинг 1.3.5. Использование фильтра `:eq(index)`

```
<script type="text/javascript">
$(function(){
  $("td:eq(5)").css("background-color","#ccc");
});
</script>
```

Обсуждение

В примере из листинга 1.3.5 мы нашли шестую ячейку таблицы, указав в селекторе выражение `td:eq(5)`, потому что отсчет идет, начиная от нуля.

ЗАДАЧА

Необходимо отыскать все ячейки таблицы, которые следуют после шестой по счету (иначе говоря — с индексом, более пяти).

Решение

Решить задачу поможет фильтр `:gt(index)` (листинг 1.3.6).

Листинг 1.3.6. Использование фильтра `:gt(index)`

```
<script type="text/javascript">
$(function(){
  $("td:gt(5)").css("background-color","#ccc");
});
</script>
```

Обсуждение

Все очень просто, как код в листинге 1.3.6. Указываем в селекторе выражение `td:gt(5)`, и все ячейки с индексом более пяти стали серого цвета.

ЗАДАЧА

Необходимо отыскать все ячейки таблицы, которые следуют перед шестой по счету (иначе говоря — с индексом, менее пяти).

Решение

Для решения задачи применим фильтр `:lt(index)` (листинг 1.3.7).

Листинг 1.3.7. Использование фильтра `:lt(index)`

```
<script type="text/javascript">
$(function(){
  $("td:lt(5)").css("background-color","#ccc");
});
</script>
```

Обсуждение

Тоже проще простого. Указываем в селекторе выражение `td:lt(5)`, и все ячейки перед шестой стали серыми.

ЗАДАЧА

Необходимо отыскать все ячейки таблицы, кроме одной, индекс которой равен пяти.

Решение

Для решения задачи применим фильтр `:not()` (листинг 1.3.8).

Листинг 1.3.8. Использование фильтра `:not()`

```
<script type="text/javascript">
$(function(){
  $("td:not(:eq(5))").css("background-color","#ccc");
});
</script>
```

Обсуждение

Внутри фильтра `:not()` мы указали другой фильтр — `:eq(5)`, получив таким образом выражение "все, кроме". В результате все ячейки таблицы кроме одной стали серыми.

ЗАДАЧА

Необходимо отыскать на веб-странице все элементы, являющиеся заголовками, например, `h1`, `h2`, `h3` и т. д.

Решение

Решить задачу позволяет фильтр `:header` (листинг 1.3.9).

Листинг 1.3.9. Использование фильтра `:header`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-3-8</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
table {
  width:400px;
}
</style>
<script type="text/javascript">
$(function(){
  $(":header").css("background-color","#ccc");
});
</script>
</head>
<body>
<h1>Заголовок h1</h1>
<p>Некоторый текст в элементе p</p>
<h3>Заголовок h3</h3>
<ul>
  <li>Список: пункт 1</li>
  <li>Список: пункт 2</li>
  <li>Список: пункт 3</li>
</ul>
</body>
</html>
```

Обсуждение

В HTML-коде, который приведен в листинге 1.3.9, присутствуют элемент параграф `p`, список `ul` и два элемента, являющиеся заголовками, — `h1` и `h3`. Указывая в селекторе фильтр `:header`, мы имеем возможность легко отыскать эти заголовки и установить для них серый цвет фона.

ЗАДАЧА

Необходимо выбрать все элементы, которые к моменту выбора будут находиться в процессе анимации.

Решение

Для решения задачи применим фильтр `:animated()` (листинг 1.3.10).

Листинг 1.3.10. Использование фильтра `:animated()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-3-10</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { background-color:#ff0; border:1px solid #aaa; width:80px; height:80px;
margin:0 5px; float:left; }
div.colored { background:#008000; }
</style>
<script type="text/javascript">
$(function(){
  $("#run").click(function(){
    $("div:animated").toggleClass("colored");
  });
  function animateIt() {
    $("#mover").slideToggle("slow", animateIt);
  }
  animateIt();
});
</script>
</head>
<body>
<div></div>
<div id="mover"></div>
<div></div>
<button id="run">Run</button>
</body>
</html>
```

Обсуждение

Код, приведенный в листинге 1.3.10, несколько сложнее предыдущих примеров. Не будем на этом этапе вдаваться в подробности. Отметим лишь, что при щелчке левой кнопкой мыши на кнопке с идентификатором `run` будет выполнена некоторая функция. И первое, что она делает, — обнаруживает один из трех элементов `div`, а именно тот элемент, который находится в процессе анимации.

1.4. Фильтрация по содержимому

ЗАДАЧА

Необходимо отыскать все элементы, внутри которых находится текст, содержащий подстроку 'John'.

Решение

Для решения задачи используем фильтр `:contains` (листинг 1.4.1).

Листинг 1.4.1. Использование фильтра `:contains`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-4-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("div:contains('John')").css("text-decoration", "underline");
});
</script>
</head>
<body>
<div>John Resig</div>
<div>George Martin</div>
<div>Malcom John Sinclair</div>
<div>J. Ohn</div>
<div>b byJohns a</div>
</body>
</html>
```

Обсуждение

Рассмотрим пример кода, приведенный в листинге 1.4.1. Имеются пять элементов `div`, внутри которых находится некоторый текст. Чтобы отыскать только те элементы `div`, текст внутри которых содержит подстроку `'John'`, указываем в селекторе выражение `div:contains('John')`, выбирая нужные нам элементы. Для наглядности делаем текст внутри этих элементов подчеркнутым с помощью изменения CSS-свойства `text-decoration`.

ЗАДАЧА

В таблице необходимо отыскать только те ячейки, которые не содержат элементов-потомков, включая и текст.

Решение

Для решения задачи воспользуемся фильтром `:empty` (листинг 1.4.2).

Листинг 1.4.2. Использование фильтра `:empty`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-4-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("td:empty").css("background-color","rgb(255,204,153)")
});
</script>
</head>
<body>
<table style="width:200px;border:1px solid #000;">
  <tr><td>1-1</td><td><span></span></td></tr>
  <tr><td>2-1</td><td></td></tr>
  <tr><td></td><td>3-2</td></tr>
  <tr><td></td><td>4-2</td></tr>
</table>
</body>
</html>
```

Обсуждение

В примере из листинга 1.4.2 мы применили фильтр `:empty` к элементам `td`, чтобы отыскать пустые ячейки, т. е. ячейки, не имеющие элементов-потомков, в том числе и текстовых узлов. Для найденных ячеек установили цвет фона, используя CSS-свойство `background-color`. Точно так же фильтр `:empty` можно применить и к другим элементам.

ЗАДАЧА

На веб-странице необходимо отыскать все элементы `div`, внутри которых находится как минимум один элемент `p`.

Решение

Задача решается с помощью фильтра `:has()`(листинг 1.4.3).

Листинг 1.4.3. Использование фильтра `:has()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-1-4-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
  .test{ border: 3px inset #f00; }
</style>
<script type="text/javascript">
$(function(){
  $("div:has(p)").addClass("test");
});
</script>
</head>
<body>
<div><p>Текст внутри p, который внутри div</p></div>
<div>Это просто текст внутри div</div>
</body>
</html>
```

Обсуждение

В примере из листинга 1.4.3 к элементам `div` мы применили фильтр `:has(p)`. Выбранным оказался элемент `div`, внутри которого находится элемент `p`. Найденный элемент отметили красной рамкой, добавив ему класс с именем `test`.

ЗАДАЧА

В таблице необходимо отыскать только те ячейки, которые содержат элементы-потомки, включая и текст.

Решение

Для решения задачи воспользуемся фильтром `:parent` (листинг 1.4.4).

Листинг 1.4.4. Использование фильтра `:parent`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-4-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
  td { background-color:#393; }
</style>
<script type="text/javascript">
```

```
$(function(){
  $("td:parent").fadeTo(3000, 0.3);
});
</script>
</head>
<body>
<table style="width:200px;border:1px solid #000;">
  <tr><td>1-1</td><td><span></span></td></tr>
  <tr><td>2-1</td><td></td></tr>
  <tr><td></td><td>3-2</td></tr>
  <tr><td></td><td>4-2</td></tr>
</table>
</body>
</html>
```

Обсуждение

В примере из листинга 1.4.4 мы применили фильтр `:parent` к элементам `td`, чтобы отыскать непустые ячейки, т. е. ячейки, имеющие элементы-потомки, в том числе и текстовые узлы. К найденным ячейкам таблицы применили метод `fadeTo()`, плавно изменяя значение CSS-свойства `opacity` от 1 до 0,3 в течение 3000 мс. Точно так же фильтр `:parent` можно применить и к другим элементам.

1.5. Фильтры видимых и невидимых элементов

ЗАДАЧА

На веб-странице необходимо подсчитать число видимых и невидимых элементов и вывести эти значения в окне предупреждения.

Решение

Решим задачу с помощью двух фильтров — `:visible` и `:hidden` (листинг 1.5.1).

Листинг 1.5.1. Использование фильтров `:visible` и `:hidden`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-5-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { width:50px; height:50px; margin:5px; border:4px outset #00f; float:left; }
```

```
.starthidden { display:none; }
</style>
<script type="text/javascript">
$(function(){
  alert("Видимых элементов найдено ... " +
  $(":visible", document.body).length +
  "\nНевидимых элементов найдено ... " +
  $(":hidden", document.body).length);
});
</script>
</head>
<body>
<div></div>
<div class="starthidden"></div>
<div></div>
<div></div>
<div style="display:none;"></div>
<form>
  <input type="hidden" />
  <input type="text" />
  <input type="hidden" />
</form>
</body>
</html>
```

Обсуждение

Код листинга 1.5.1 очень прост. Имеются пять элементов `div`, два из которых скрыты с помощью CSS-свойства `display`, и элемент `form` с тремя элементами `input`, два из которых имеют значение `hidden` в атрибуте `type`, т. е. являются скрытыми полями формы.

В JavaScript-коде мы выводим в окне предупреждения число найденных видимых и невидимых элементов, сопровождая это поясняющим текстом. Интересующие элементы мы ищем в контексте `document.body` из-за того, что фильтр `:hidden` в некоторых браузерах подсчитывает элементы `head`, `title`, `script` и т. д.

В ряде случаев следует учитывать и другие особенности подсчета элементов фильтром `:hidden` — для него считаются невидимыми те элементы (или их родители), которые не имеют размеров в документе. И конечно учитываются элементы, скрытые с помощью CSS.

Итак, в результате выполнения кода из листинга 1.5.1 мы получим сообщение о том, что в теле документа удалось обнаружить пять видимых элементов и четыре невидимых.

1.6. Селекторы атрибутов

ЗАДАЧА

На веб-странице необходимо отыскать элементы `div`, у которых присутствует атрибут `id`. Значение атрибута для нас пока совершенно неважно.

Решение

Для решения задачи воспользуемся селектором `selector[name]` (листинг 1.6.1).

Листинг 1.6.1. Использование селектора `selector[name]`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-6-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { width:100px; height:100px; margin:5px; padding:5px; border:1px solid
#00f; float:left; }
</style>
<script type="text/javascript">
$(function(){
  $("div[id]").css({ "border-color": "#f00", "color": "#f00" });
});
</script>
</head>
<body>
<div>Элемент div без атрибута id, но с class="example"</div>
<div id="test">Элемент div с атрибутом id="test"</div>
<div id="pretest" class="example">Элемент div с атрибутами id="pretest" и
class="example"</div>
<div>Элемент div без атрибута id</div>
<div id="testend">Элемент div с атрибутом id="testend"</div>
<div id="sometestvalue">Элемент div с атрибутом id="sometestvalue"</div>
<div>Элемент div без атрибута id</div>
<div id="somevalue">Элемент div с атрибутом id="somevalue"</div>
</body>
</html>
```

Обсуждение

В примере из листинга 1.6.1 HTML-код описывает восемь элементов `div`. Одни из них имеют атрибуты `id` или `class`, или даже оба атрибута вместе, а другие не имеют

атрибутов вовсе. Этот HTML-код потребуется нам для изучения всех примеров в разделе, посвященном селекторам атрибутов.

Для того чтобы отыскать все элементы `div`, имеющие атрибут `id`, в селекторе необходимо записать выражение `div[id]`, и задача будет решена. А для наглядности мы установим значения `#f00` для CSS-свойств `border-color` и `color` найденных элементов `div`, окрасив рамку элементов и текст внутри них в красный цвет. "Красными" окажутся второй, третий, пятый, шестой и восьмой элементы `div`.

ЗАДАЧА

На веб-странице необходимо отыскать элемент `div`, у которого атрибут `id` имеет определенное значение.

Решение

Для решения задачи воспользуемся селектором `selector[name="value"]` (листинг 1.6.2).

Листинг 1.6.2. Использование селектора `selector[name="value"]`

```
<script type="text/javascript">
$(function(){
  $("div[id='somevalue']").css({
    "border-color": "#f00", "color": "#f00"
  });
});
</script>
```

Обсуждение

В примере из листинга 1.6.2 в селекторе мы записываем выражение `div[id='somevalue']`, обнаруживая, таким образом, единственный элемент, у которого атрибут `id` имеет значение `somevalue`. Им оказывается последний, восьмой по счету элемент `div`.

> ***Примечание***
>
> Селектор из листинга 1.6.2 приведен исключительно в демонстрационных целях. В реальном программировании, для отыскания элемента по значению его идентификатора, гораздо уместнее использовать соответствующий селектор — `#somevalue`.

ЗАДАЧА

На веб-странице необходимо отыскать элементы `div`, у которых атрибут `id` не соответствует определенному значению.

Решение

Для решения задачи применим селектор `selector[name!="value"]` (листинг 1.6.3).

Листинг 1.6.3. Использование селектора `selector[name!="value"]`

```
<script type="text/javascript">
$(function(){
  $("div[id!='somevalue']").css({
    "border-color": "#f00", "color": "#f00"
  });
});
</script>
```

Обсуждение

В примере из листинга 1.6.3 мы записали в селекторе выражение `div[id!='somevalue']` и нашли все элементы `div`, у которых значение атрибута `id` отличается от `somevalue`. Обратите внимание, что выбранными оказались также элементы `div`, у которых атрибут `id` и вовсе отсутствует. Таким образом, у нас "покраснели" все элементы `div`, за исключением восьмого.

ЗАДАЧА

На веб-странице необходимо отыскать элементы `div`, у которых значение атрибута `id` начинается с определенной последовательности символов.

Решение

Для решения задачи воспользуемся селектором `selector[name^="value"]` (листинг 1.6.4).

Листинг 1.6.4. Использование селектора `selector[name^="value"]`

```
<script type="text/javascript">
$(function(){
  $("div[id^='test']").css({
    "border-color": "#f00", "color": "#f00"
  });
});
</script>
```

Обсуждение

В примере из листинга 1.6.4, чтобы отыскать все элементы `div`, значение атрибута `id` которых начинается с последовательности символов `test`, мы записали в селек-

торе выражение `div[id^='somevalue']`. "Красными" стали второй и пятый элементы `div`.

ЗАДАЧА

На веб-странице необходимо отыскать элементы `div`, у которых значение атрибута `id` заканчивается определенной последовательностью символов.

Решение

Для решения задачи воспользуемся селектором `selector[name$="value"]` (листинг 1.6.5).

Листинг 1.6.5. Использование селектора `selector[name$="value"]`

```
<script type="text/javascript">
$(function(){
  $("div[id$='test']").css({
    "border-color": "#f00", "color": "#f00"
  });
});
</script>
```

Обсуждение

В примере из листинга 1.6.5, чтобы отыскать все элементы `div`, значение атрибута `id` которых заканчивается последовательностью символов `test`, мы записали в селекторе выражение `div[id$='somevalue']`. "Красными" стали второй и третий элементы `div`.

ЗАДАЧА

На веб-странице необходимо отыскать элементы `div`, у которых значение атрибута `id` содержит определенную последовательность символов.

Решение

Решить задачу поможет селектор `selector[name*="value"]` (листинг 1.6.6).

Листинг 1.6.6. Использование селектора `selector[name*="value"]`

```
<script type="text/javascript">
$(function(){
  $("div[id*='test']").css({
    "border-color": "#f00", "color": "#f00"
  });
});
</script>
```

Обсуждение

В примере из листинга 1.6.6, чтобы отыскать все элементы `div`, значение атрибута `id` которых содержит последовательность символов `test`, мы записали в селекторе выражение `div[id*='somevalue']`. "Красными" стали второй, третий, пятый и шестой элементы `div`. Обратите внимание, что при использовании этого селектора совершенно неважно, где находится искомая последовательность символов — в начале, в конце или в середине значения атрибута.

ЗАДАЧА

На веб-странице необходимо отыскать элементы `div`, у которых значение атрибута `id` содержит определенную последовательность символов и к тому же присутствует атрибут `class` с заданным значением.

Решение

Решим задачу с помощью составного селектора `selector[name="value"][name2="value2"]` (листинг 1.6.7).

Листинг 1.6.7. Использование составного селектора `selector[name="value"][name2="value2"]`

```
<script type="text/javascript">
$(function(){
  $("div[id*='test'][class='example']").css({
    "border-color": "#f00", "color": "#f00"
  });
});
</script>
```

Обсуждение

Чтобы отыскать все элементы `div`, значение атрибута `id` которых содержит последовательность символов `test` и одновременно атрибут `class` которых имеет значение `example`, в примере из листинга 1.6.7 мы записали в селекторе выражение `div[id*='test'][class='example']`. Такому комбинированному условию удовлетворяет только третий элемент `div`.

1.7. Фильтры элементов форм

ЗАДАЧА

На веб-странице необходимо найти все элементы `input` независимо от их типа.

Решение

Для решения задачи воспользуемся фильтром `:input` (листинг 1.7.1).

Листинг 1.7.1. Использование фильтра `:input`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-7-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
form { width:450px; }
input { margin-left:75px; }
label { float:right; }
</style>
<script type="text/javascript">
$(function(){
  $(":input").prev("label").css("color","#c33");
});
</script>
</head>
<body>
<form>
  <label>элемент type="checkbox"</label> <input type="checkbox" /><br />
  <label>элемент type="file"</label> <input type="file" /><br />
  <label>элемент type="image"</label> <input type="image" /><br />
  <label>элемент type="password"</label> <input type="password" /><br />
  <label>элемент type="radio"</label> <input type="radio" /><br />
  <label>элемент type="reset"</label> <input type="reset" /><br />
  <label>элемент type="submit"</label> <input type="submit" /><br />
  <label>элемент type="text"</label> <input type="text" /><br />
  <label>элемент type="button"</label> <input type="button" value="Button" />
</form>
</body>
</html>
```

Обсуждение

Приведенный в листинге 1.7.1 HTML-код, описывающий форму с элементами `input` практически всех возможных типов, мы будем использовать при знакомстве с фильтрами элементов форм. Указав в селекторе выражение `:input`, мы выберем все девять элементов. Затем с помощью метода `prev()` находим предыдущий элемент `label` и делаем текст внутри него красноватого цвета, чтобы как-то отметить найденные элементы.

ЗАДАЧА

На веб-странице необходимо отыскать все элементы `input`, атрибут `type` которых содержит значение `text`.

Решение

Для решения задачи воспользуемся фильтром `:text` (листинг 1.7.2).

Листинг 1.7.2. Использование фильтра `:text`

```
<script type="text/javascript">
$(function(){
  $(":text").prev("label").css("color","#c33");
});
</script>
```

Обсуждение

В листинге 1.7.2 приведен только JavaScript-код, поскольку HTML-разметка не изменилась. Мы также экспериментируем с формой, состоящей из элементов `input` всех возможных типов. Чтобы обнаружить элемент `input` типа `text`, необходимо просто записать в селекторе выражение `:text` и текст в соответствующем элементе `label` станет красного цвета.

ЗАДАЧА

На веб-странице необходимо найти все элементы `input`, атрибут `type` которых содержит значение `password`.

Решение

Для решения задачи применим фильтр `:password` (листинг 1.7.3).

Листинг 1.7.3. Использование фильтра `:password`

```
<script type="text/javascript">
$(function(){
  $(":password").prev("label").css("color","#c33");
});
</script>
```

Обсуждение

В листинге 1.7.3 записываем выражение `:password` в селекторе, и элемент нужного типа обнаружен.

Использование остальных типов фильтров элементов форм абсолютно ничем не отличается. Для того чтобы суметь их применить, их следует просто знать. Итак, наряду с фильтрами :input, :text и :password, существуют еще :radio, :checkbox, :submit, :image, :reset, :button и :file.

Кроме этих фильтров существуют также фильтры, с помощью которых можно отслеживать состояние элементов форм.

1.8. Фильтры состояния элементов форм

ЗАДАЧА

На веб-странице существует форма, в которой нужно отыскать все поля, доступные для заполнения, и отметить их. То же самое необходимо сделать с теми полями, которые для заполнения недоступны.

Решение

Решить задачу помогут фильтры :enabled и :disabled (листинг 1.8.1).

Листинг 1.8.1. Использование фильтров :enabled и :disabled

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-8-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
form { width:300px; }
input { float:right; margin-bottom:2px; }
br { clear:right; }
</style>
<script type="text/javascript">
$(function(){
  $("input:disabled").val("DISABLED");
  $("input:enabled").val("ENABLED");
});
</script>
</head>
<body>
<form>
  <input type="text" /><label>Имя</label><br />
  <input type="text" /><label>Фамилия</label><br />
  <input type="text" disabled="disabled" /><label>Серия паспорта</label><br />
  <input type="text" disabled="disabled" /><label>Номер паспорта</label><br />
  <input type="text" /><label>E-mail адрес</label><br />
```

```
</form>
</body>
</html>
```

Обсуждение

В HTML-коде листинга 1.8.1 описана форма, состоящая из пяти полей ввода, два из которых имеют значение `disabled` атрибута `disabled`. Остальные поля такого атрибута не имеют и, следовательно, доступны для заполнения.

Посмотрим на JavaScript-код. Указывая в селекторе выражение `input:disabled`, мы выбираем два поля, которые запрещены для заполнения. С помощью метода `val()` мы, тем не менее, можем вставить какое-либо значение в эти поля. Пользуемся этим и вставляем значение `DISABLED`. Аналогично поступаем с доступными полями — записывая в селекторе выражение `input:enabled`, выбираем их, и вставляем значение `ENABLED` с помощью того же метода `val()`.

ЗАДАЧА

На веб-странице необходимо отслеживать состояние элементов `checkbox`.

Решение

Для решения задачи используем фильтр `:checked` (листинг 1.8.2).

Листинг 1.8.2. Использование фильтра `:checked`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-8-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
form { width:100px; }
input { float:right; margin-bottom:2px; }
br { clear:right; }
</style>
<script type="text/javascript">
$(function(){
  function countChecked() {
    var n = $("input:checked").length;
    $("div").text("Отмечено — " + n);
  }
  countChecked();
  $(":checkbox").click(countChecked);
});
```

```
</script>
</head>
<body>
<form>
  <div></div><hr />
  <input type="checkbox" /><label>Секунда</label><br />
  <input type="checkbox" /><label>Минута</label><br />
  <input type="checkbox" /><label>Час</label><br />
  <input type="checkbox" /><label>День</label><br />
  <input type="checkbox" /><label>Неделя</label><br />
  <input type="checkbox" checked="checked" /><label>Месяц</label><br />
  <input type="checkbox" checked="checked" /><label>Год</label><br />
</form>
</body>
</html>
```

Обсуждение

Сначала разберем HTML-разметку из листинга 1.8.2. Форма содержит семь элементов `checkbox`, два из которых отмечены по умолчанию. Есть элемент `div` — сюда мы будем вставлять сообщения о количестве отмеченных элементов каждый раз, когда в форме произойдет какое-либо изменение.

Теперь обратим внимание на JavaScript-код. Сначала мы определили простую функцию для подсчета отмеченных элементов `checkbox`. В первой строке кода функции мы отыскиваем все отмеченные элементы `checkbox`, подсчитываем их число с помощью метода `length()` и запоминаем в переменной `n`. Во второй строке находим элемент `div` и вставляем в него текст, где присутствует и переменная `n` — число отмеченных `checkbox`. Осталось только вызвать эту функцию при загрузке страницы и, конечно, вызывать каждый раз при наступлении события `click` на каком-либо из элементов `checkbox`.

ЗАДАЧА

На веб-странице находится выпадающий список с возможностью одновременного выбора нескольких опций. Необходимо отслеживать, какие именно опции были выбраны.

Решение

Для решения задачи применим фильтр `:selected` (листинг 1.8.3).

Листинг 1.8.3. Использование фильтра `:selected`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-1-8-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("select").change(function() {
    var str = "";
    $("select option:selected").each(function() {
      str += $(this).text() + " ";
    });
    $("div").text(str);
  }).change();
});
</script>
</head>
<body>
<select name="colors" multiple="multiple">
  <option>Красный</option>
  <option selected="selected">Оранжевый</option>
  <option>Желтый</option>
  <option selected="selected">Зеленый</option>
  <option>Голубой</option>
  <option>Синий</option>
  <option>Фиолетовый</option>
</select>
<div></div>
</body>
</html>
```

Обсуждение

В листинге 1.8.3 HTML-разметка очень простая — элемент `select` с возможностью выбора сразу нескольких опций, две из которых уже заданы по умолчанию, и пустой элемент `div`, в который мы будем вставлять содержимое всех выбранных опций.

Посмотрим на JavaScript-код. Первое, что мы делаем, — находим элемент `select` и связываем с ним событие `change`, при наступлении которого будет вызвана callback-функция. Каждый раз при вызове функции в переменную `str` записывается пустая строка. Затем в элементе `select` будут обнаружены все выбранные опции. Это мы проделаем с помощью фильтра `:selected`. Поскольку таких опций может быть несколько, потребуется метод `each()`, который позволит применить функцию, указанную в качестве его аргумента, к каждому выбранному элементу. Таким образом, мы получаем возможность сформировать в переменной `str` строку, содержащую текст всех выбранных значений списка. После завершения итераций мы вставляем текст готовой строки в элемент `div`. И последнее, что нам осталось сделать, —

с помощью вызова метода `change()` без передачи ему аргументов, имитировать событие `change` при загрузке страницы.

1.9. Фильтры элементов-потомков

ЗАДАЧА

Требуется отыскать некоторых потомков указанных элементов, причем задача усложняется тем, что следует отыскать не только четные или нечетные потомки, или найти необходимый элемент-потомок по его индексу, но, например, нужно будет отыскать каждый третий элемент-потомок и т. д.

Решение

Решим задачу с помощью фильтра `:nth-child(index/even/odd/equation)` (листинг 1.9.1).

Листинг 1.9.1. Использование фильтра `:nth-child(index/even/odd/equation)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-9-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
button { width:100px; }
span { color:#00f; font-weight:bold; }
#inner { color:#f00; }
table {
  float:left;
  margin-left:50px;
  border:1px solid #ccc;
}
td {
  width:100px;
  text-align:center;
  border:1px dotted #ccc;
}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function () {
    var str = $(this).text();
    $("tr").css("background-color", "#fff");
    $("tr" + str).css("background-color", "#f00");
```

```
    $("#inner").text(str);
  });
});
</script>
</head>
<body>
<button>:nth-child(even)</button>
<button>:nth-child(odd)</button>
<button>:nth-child(3n)</button>
<button>:nth-child(2)</button>
<button>:nth-child(3n+1)</button>
<button>:nth-child(3n+2)</button>
<button>:even</button>
<button>:odd</button>
<hr />
<span>tr<span id="inner"></span></span>
<hr />
<table>
  <tr><td>Вадим</td></tr>
  <tr><td>Владимир</td></tr>
  <tr><td>Владислав</td></tr>
  <tr><td>Всеволод</td></tr>
  <tr><td>Вячеслав</td></tr>
</table>
<table>
  <tr><td>Эдуард</td></tr>
  <tr><td>Юрий</td></tr>
  <tr><td>Яков</td></tr>
</table>
<table>
  <tr><td>Георгий</td></tr>
  <tr><td>Дмитрий</td></tr>
  <tr><td>Евгений</td></tr>
  <tr><td>Игорь</td></tr>
  <tr><td>Константин</td></tr>
  <tr><td>Леонид</td></tr>
  <tr><td>Михаил</td></tr>
  <tr><td>Николай</td></tr>
  <tr><td>Павел</td></tr>
  <tr><td>Роман</td></tr>
</table>
</body>
</html>
```

Обсуждение

Для начала разберемся с HTML-разметкой, приведенной в листинге 1.9.1. Разметка состоит из набора кнопок, текст на которых представляет собой один из вариантов

применения рассматриваемых фильтров. Причем две последние кнопки с надписями :even и :odd предусмотрены для того, чтобы наглядно продемонстрировать разницу в работе фильтров. При нажатии одной из кнопок будет применен соответствующий фильтр. В элементы span мы будем выводить получившееся выражение, с помощью которого были выбраны элементы. И, наконец, три таблицы, над которыми и будем проводить наши опыты.

Посмотрим JavaScript-код. Сначала отбираем в объект jQuery все имеющиеся на веб-странице элементы button, с которыми связываем callback-функцию, вызываемую при возникновении события click на какой-либо кнопке. Вызванная функция сначала получит текст надписи на кнопке и сохранит его в переменной str. Следующая строка кода устанавливает белый цвет фона для всех элементов tr. Это делается, чтобы очистить результаты работы предыдущего фильтра, т. к. найденные с помощью фильтров строки мы отмечаем, устанавливая им красный цвет фона. За это действие отвечает следующая строка — в селекторе мы составляем нужное нам выражение, добавляя к записи tr текст, сохраненный в переменной str, т. е. какой-то из фильтров. Для строк, найденных с помощью этого выражения, задаем красный цвет фона, а в элемент span с идентификатором #inner вставляем переменную str, чтобы увидеть то выражение, с помощью которого выбирались строки таблиц.

Если с HTML-разметкой и JavaScript-кодом мы разобрались, то работу фильтров детально пока не изучили. Пора приступить к этому. Откройте соответствующий пример на компакт-диске, прилагаемом к книге. Попробуйте нажать сначала на кнопку :nth-child(even), а затем на :even. Результаты получаются разные. Фильтр :even, примененный к элементу tr, отмечает четные строки без учета принадлежности к конкретной таблице. Фильтр :nth-child(even) учитывает принадлежность строки к таблицам. Аналогичным образом, только для нечетных строк, можно сравнивать работу фильтров :odd и :nth-child(odd).

Другие варианты использования этих фильтров предлагаем разобрать самостоятельно.

ЗАДАЧА

Необходимо отыскать только первый и последний элементы-потомки во всех элементах набора.

Решение

Для решения задачи используем фильтры :first-child и :last-child (листинг 1.9.2).

Листинг 1.9.2. Использование фильтров :first-child и :last-child

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
```

```
<title>example-1-9-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
span { color:#008; }
</style>
<script type="text/javascript">
$(function(){
  $("div span:first-child").css("text-decoration", "underline");
  $("div span:last-child").css("text-decoration", "line-through");
});
</script>
</head>
<body>
<div>
  <span>Георгий,</span>
  <span>Дмитрий,</span>
  <span>Евгений</span>
</div>
<div>
  <span>Игорь,</span>
  <span>Константин,</span>
  <span>Леонид</span>
</div>
</body>
</html>
```

В листинге 1.9.2 приведена несложная HTML-разметка — два элемента `div`, внутри которых расположены по три элемента `span`. В JavaScript-коде все довольно просто — находим все `div`, внутри которых есть элементы `span`, и выбираем только первые из них с помощью фильтра `:first-child`. Подчеркиваем текст во всех элементах получившегося набора. Аналогично находим последние элементы `span` внутри `div`, только в этом случае используем фильтр `:last-child` и не подчеркиваем, а зачеркиваем текст в найденных элементах.

Подчеркнутыми оказались Георгий и Игорь, а зачеркнутыми — Евгений и Леонид.

> ***Примечание***
>
> Обратите внимание — если бы мы попробовали применить для решения задачи фильтры `:first` и `:last`, то смогли найти первый и последний элемент `span` только в первом элементе `div`, а не во всех.

ЗАДАЧА

Необходимо отыскать элементы, которые являются единственными наследниками своего родителя, причем текстовый узел в качестве наследника не должен учитываться.

Решение

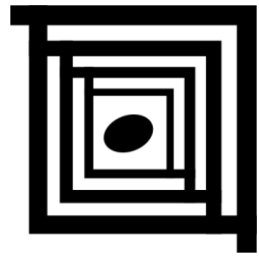
Решить задачу поможет фильтр `:only-child` (листинг 1.9.3).

Листинг 1.9.3. Использование фильтра `:only-child`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-1-9-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px;
  height:100px;
  margin:5px;
  padding:5px;
  float:left;
  background-color:#b9e;
}
</style>
<script type="text/javascript">
$(function(){
  $("div button:only-child").text("only-child").css("border", "1px solid #f00");
});
</script>
</head>
<body>
<div>
  <button>Кнопка</button>
  <button>Кнопка</button>
</div>
<div>
  <button>Кнопка</button>
</div>
<div>Это текст</div>
<div>
  <button>Кнопка</button>
  <button>Кнопка</button>
  <button>Кнопка</button>
</div>
<div>
  <button>Кнопка</button>
</div>
```

```
</body>
</html>
```

В HTML-коде из листинга 1.9.3 можно обнаружить пять элементов `div`. В одном из них содержится текст, в остальных — элементы `button`. Но только в двух элементах `div` кнопки `button` являются единственными наследниками. Посмотрим JavaScript-код — выражение `div button:only-child` позволяет быстро отыскать данные элементы. Для наглядности заменяем надписи и задаем кнопкам красную рамку.



ГЛАВА 2

Атрибуты элементов

2.1. Управление атрибутами элементов

Возможность управлять значениями атрибутов элементов объектной модели документа позволяет изменять визуальное представление этих элементов, заменять значения элементов форм программным способом, управлять их состоянием и т. п.

ЗАДАЧА

Необходимо получить значение атрибута `href` элемента `a` и установить его в качестве значения атрибута `title`.

Решение

Для решения такой задачи используем метод `attr()`, который поможет сначала получить, а затем установить значение требуемого атрибута по его имени (листинг 2.1.1).

Листинг 2.1.1. Использование метода `attr()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  var href = $("a").attr("href"); // получаем значение атрибута
  $("a").attr("title", href); // устанавливаем значение атрибута
});
```

```
</script>
</head>
<body>
<a href="http://jquery.com">Официальный сайт jQuery</a>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 2.1.1, — это обычная ссылка. Нас интересует ее атрибут `href`. Если посмотреть на JavaScript-код, то можно увидеть, что значение именно этого атрибута мы сохраняем в переменной `link`. А в следующей строке кода мы уже устанавливаем значение атрибута `title`.

После выполнения этого кода, при наведении указателя мыши на ссылку, будет появляться стандартная всплывающая подсказка с текстом "http://jquery.com", хотя изначально в HTML-разметке ссылка была вообще без атрибута `title`.

Получается, что передача методу `attr()` одного аргумента (названия атрибута) позволяет получить значение этого атрибута. А использование метода с двумя параметрами дает возможность устанавливать значение атрибута.

Аналогично можно работать с любыми атрибутами любых элементов DOM.

ЗАДАЧА

Для элемента `a` необходимо установить значения атрибутов `alt`, `title` и `href`, которые отсутствуют в HTML-разметке.

Решение

Решим задачу с помощью того же метода `attr()`, воспользовавшись возможностью передать ему в качестве единственного аргумента объект, свойствами которого являются названия атрибутов, а значениями этих свойств — значения атрибутов элемента (листинг 2.1.2).

Листинг 2.1.2. Использование метода `attr(map)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("a").attr({
    "alt": "Официальный сайт jQuery",
```

```
    "title": "jQuery.Com",
    "href": "http://jquery.com"
  }).click(function(){
    alert($(this).attr("alt"));
    return false;
  });
});
</script>
</head>
<body>
<a href="#">Официальный сайт библиотеки jQuery</a>
</body>
</html>
```

Обсуждение

В HTML-коде листинга 2.1.2 присутствует ссылка только с одним атрибутом `href`, значением которого является символ `#`. Рассмотрим JavaScript-код. Сначала находим ссылку — элемент `a`, затем с помощью метода `attr(map)` устанавливаем одновременно несколько атрибутов — `alt`, `title` и `href`. Продолжаем цепочку вызовов, связывая вызов callback-функции с событием `click` на элементе `a`. Эта функция отобразит в окне предупреждения значение атрибута `alt`, подтвердив таким образом, что мы успешно присвоили нужные атрибуты выбранной ссылке. Проверить значения остальных атрибутов можно, наведя указатель мыши на ссылку. При этом в стандартной всплывающей подсказке появится значение "jQuery.Com", а в строке состояния браузера — значение атрибута `href`, т. е. "http://jquery.com".

ЗАДАЧА

Для всех элементов `p`, обнаруженных на веб-странице, необходимо установить значение атрибута `name`, вычисленное на основании положения элемента.

Решение

Для решения задачи используем метод `attr(name, function(index,attr))` (листинг 2.1.3).

Листинг 2.1.3. Использование метода `attr(name, function(index,attr))`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
```

```
<script type="text/javascript">
$(function(){
  $("p").attr("name", function(n){
    return "pName-" + n;
  }).click(function(){
    alert("name = " + $(this).attr("name"));
  });
});
</script>
</head>
<body>
<p>Первый параграф</p>
<p>Второй параграф</p>
<p>Третий параграф</p>
<p>Четвертый параграф</p>
</body>
</html>
```

Обсуждение

В примере из листинга 2.1.3 с помощью HTML-разметки определены четыре параграфа `p` без каких-либо атрибутов. Рассмотрим JavaScript-код, который начнет свою работу, как только DOM (Document Object Model, объектная модель документа) будет готова. Сначала будут найдены все элементы `p` на странице, затем с помощью метода `attr(name, function(index,attr))` будет установлен атрибут `name` для всех выбранных параграфов. Функция, передаваемая методу вторым аргументом, в свою очередь также принимает два аргумента — индекс элемента в массиве объектов jQuery (отсчет начинается с нуля) и старое значение атрибута. В нашем примере мы указали только один аргумент. Функция вызывается для каждого элемента, попавшего в набор, и должна вернуть значение атрибута `name` для текущего элемента.

Мы продолжим цепочку вызовов, связав со всеми элементами `p` нашего набора событие `click`. Функция, которая вызывается при наступлении этого события, просто отобразит в окне предупреждения значение атрибута `name`.

ЗАДАЧА

Необходимо разрешить редактирование элемента `input`, которое было запрещено с помощью присвоения значения `disabled` атрибуту `disabled`.

Решение

Решим задачу простым удалением атрибута, применив для этого метод `removeAttr(name)` (листинг 2.1.4).

Листинг 2.1.4. Использование метода `removeAttr(name)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("a").click(function() {
    $(this).next().removeAttr("disabled").focus().val("Редактируется");
  });
});
</script>
</head>
<body>
<a href="#">Разрешить редактирование</a>
<input type="text" disabled="disabled" value="Не редактируется" />
</body>
</html>
```

Обсуждение

Итак, в HTML-коде, приведенном в листинге 2.1.4, имеется элемент `input`, редактирование которого запрещено с помощью атрибута `disabled`. Есть элемент `a`, при щелчке на котором с элемента `input` должен быть снят запрет редактирования.

Обратимся к JavaScript-коду. Сначала мы отыщем элемент `a`, чтобы связать с ним событие `click`, при наступлении которого будет вызываться функция, снимающая запрет редактирования. С этого момента разбираем код еще подробнее. В контексте функции, `this` ссылается на объект, описывающий элемент `a`, по которому был совершен щелчок. Но нам нужно работать со следующим элементом — `input`, поэтому в цепочку вызовов добавляем вызов метода `next()`, который поможет отыскать элемент `input`. Следующий метод в цепочке — `removeAttr(name)`, с помощью которого мы удаляем атрибут `disabled` элемента `input`. Дальше просто для наглядности передаем в `input` фокус, вызывая метод `focus()`, и вставляем текст "Редактируется".

2.2. Работа с атрибутом *class*

ЗАДАЧА

На разрабатываемой веб-странице подразумевается активная работа с классами элементов. Необходимо добавлять, удалять и переключать классы, а также выполнять какие-либо действия в зависимости от того, присутствует ли у элемента определенный класс.

Решение

Для решения этих задач используем имеющиеся в арсенале библиотеки jQuery методы: `addClass()`, `removeClass()`, `toggleClass()` и `hasClass()` (листинг 2.2.1).

Листинг 2.2.1. Использование методов `addClass()`, `removeClass()`, `toggleClass()` и `hasClass()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p { border:1px dotted #ccc; }
.test { background-color:#369; color:#fff; }
</style>
<script type="text/javascript">
$(function(){
  $("#add").click(function(){
    $("p").addClass("test");
  });
  $("#remove").click(function(){
    $("p").removeClass("test");
  });
  $("#toggle").click(function(){
    $("p").toggleClass("test");
  });
  $("#has").click(function(){
    alert($("p:first").hasClass("test"));
  });
});
</script>
</head>
<body>
<p>Параграф №1</p>
<p>Параграф №2</p>
<p>Параграф №3</p>
<button id="add">Добавить класс</button>
<button id="remove">Удалить класс</button>
<button id="toggle">Переключить класс</button>
<button id="has">Проверить класс</button>
</body>
</html>
```

Обсуждение

В листинге 2.2.1 есть три "подопытных" параграфа, оформление которых задано с помощью CSS-правила для элементов `p`, и четыре кнопки, с помощью которых мы сможем добавлять, удалять, переключать и проверять присутствие класса с именем `test` у элементов `p`.

Посмотрите JavaScript-код. Он очень простой — с каждой из кнопок связан обработчик события `click`, но callback-функции для каждой кнопки, конечно, вызываются разные. Так, щелчок по кнопке **Добавить класс** приведет к тому, что с помощью метода `addClass()` всем элементам `p` будет добавлен класс `test`, — изменится цвет фона и шрифта всех параграфов. Щелчок по кнопке **Удалить класс** — и элементы `p` возвращаются в свое первоначальное состояние. Щелкая по кнопке **Переключить класс** и вызывая таким образом метод `toggleClass()`, мы просто каждым следующим нажатием добавляем и удаляем класс `test`. При щелчке на кнопке **Проверить класс** вызываем метод `hasClass()`, который вернет `true` при наличии у выбранного элемента класса, с указанным именем, и `false` — в противном случае.

2.3. Работа с HTML и текстом

ЗАДАЧА

Необходимо получить внутреннее содержимое какого-либо элемента в виде HTML-кода или текста.

Решение

Для решения такой задачи в библиотеке jQuery предусмотрены методы `html()` и `text()` (листинг 2.3.1).

Листинг 2.3.1. Использование методов `html()` и `text()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("button:first").click(function(){
    alert($("#test").text());
  });
  $("button:last").click(function(){
    alert($("#test").html());
  });
});
```

```
</script>
</head>
<body>
<div id="test">
<p>Я шел, спускаясь в темные коридоры, и потом опять поднимаясь наверх. Я был один. Я кричал, мне не отвечали — я был один в этом обширном, запутанном, как лабиринт, доме.</p>
<p><em>Ги де Мопассан</em></p>
</div>
<button>Получить как текст</button>
<button>Получить как HTML</button>
</body>
</html>
```

Обсуждение

В качестве "подопытного" в листинге 2.3.1 выступает элемент `div` с идентификатором `test`, внутри которого содержится некоторый HTML-код. С помощью кнопок **Получить как текст** и **Получить как HTML**, с которыми связано событие `click`, мы вызываем методы `text()` и `html()`, получая внутреннее содержимое элемента `div` в требуемом виде, и выводим это содержимое в окне предупреждения.

ЗАДАЧА

В выбранный элемент необходимо вставить либо текст, либо HTML-код.

Решение

Для решения такой задачи в библиотеке jQuery существуют методы `html(val)` и `text(val)` (листинг 2.3.2).

Листинг 2.3.2. Использование методов `html(val)` и `text(val)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-2-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
#test { border:1px dotted #369; padding:5px; }
</style>
<script type="text/javascript">
$(function(){
  $("button:first").click(function(){
    var txt = $("div:first").text();
```

```
    $("#test").text(txt);
  });
  $("button:last").click(function(){
    var code = $("div:first").html();
    $("#test").html(code);
  });
});
</script>
</head>
<body>
<div>
<p>Я шел, спускаясь в темные коридоры, и потом опять поднимаясь наверх. Я был
один. Я кричал, мне не отвечали — я был один в этом обширном, запутанном, как
лабиринт, доме.</p><p><em>Ги де Мопассан</em></p>
</div>
<button>Получить и вставить как текст</button>
<button>Получить и вставить как HTML</button>
<div id="test">В этот div будет вставлен текст или html</div>
</body>
</html>
```

Обсуждение

В листинге 2.3.2 есть элемент `div` с некоторым HTML-кодом и две кнопки — **Получить и вставить как текст** и **Получить и вставить как HTML**. С этими кнопками свяжем событие `click`, при наступлении которого будем получать из первого элемента `div` его содержимое и вставлять в элемент `div` с идентификатором `test` в виде текста или HTML-кода. Различие легко заметить — при вставке HTML-кода сохранится форматирование, тогда как при вставке текста его не будет.

2.4. Работа с атрибутом *value*

ЗАДАЧА

По мере набора текста в поле ввода (элемент `input`) необходимо получать введенное значение и отображать его в другом элементе, например `p`.

Решение

Для решения задачи воспользуемся методом `val()` (листинг 2.4.1).

Листинг 2.4.1. Использование метода `val()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-2-4-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p { border:1px dotted #369; padding:5px; color:#00f; }
</style>
<script type="text/javascript">
$(function(){
  $("input").keyup(function () {
    var value = $(this).val();
    $("p").text(value);
  }).keyup();
});
</script>
</head>
<body>
  <input type="text" value="какой-то текст"/>
  <p></p>
</body>
</html>
```

Обсуждение

В HTML-коде листинга 2.4.1 описан элемент `input`, значением атрибута `value` которого является некоторый текст. В следующий за ним элемент `p` будем вставлять полученные значения.

Давайте разберемся в JavaScript-коде. Отыщем элемент `input` и свяжем с ним событие `keyup`, при наступлении которого будем получать значение атрибута `value` с помощью метода `val()` и вставлять его в элемент `p`. Следующий далее в цепочке вызов события `keyup` нужен для того, чтобы вставить значение атрибута `value` элемента `input` в элемент `p` при загрузке страницы.

ЗАДАЧА

Необходимо изменить значение атрибута `value` элемента `input` типа `text`.

Решение

Для решения применим метод `val(val)` (листинг 2.4.2).

Листинг 2.4.2. Использование метода `val(val)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
```

```
<title>example-2-4-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("button").click(function () {
    var txt = $(this).text();
    $("input").val(txt);
  });
});
</script>
</head>
<body>
<div>
  <button>Один</button>
  <button>Два</button>
  <button>Три</button>
</div>
<input type="text" value="Нажимайте кнопки" />
</body>
</html>
```

Обсуждение

В листинге 2.4.2 имеются три кнопки и элемент `input`, значение атрибута `value` которого мы попробуем изменять.

Код JavaScript несложен — ищем элементы `button` и связываем с ними событие `click`. Функция, вызываемая при наступлении события, получит текст надписи на кнопке и сохранит его в переменной `txt`. Значение этой переменной будет передано в качестве аргумента методу `val(val)` и внутри поля ввода мы сможем увидеть текст, соответствующий надписи на нажатой кнопке.

ЗАДАЧА

Необходимо получить значения атрибутов `value` элементов `radio`, `checkbox` и `select`.

Решение

Для решения также подходит метод `val()` (листинг 2.4.3).

Листинг 2.4.3. Использование метода `val()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-2-4-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p { border:1px dotted #369; padding:5px; color:#00f; }
</style>
<script type="text/javascript">
$(function(){
  $("#getSingle").click(function(){
    var single = $("#single option:selected").val();
    $("p").empty().text(single);
  });
  $("#getMultiple").click(function(){
    var multiple = $("#multiple").val();
    $("p").empty().text(multiple.join(", "));
  });
  $("#getCheckbox").click(function(){
    var checkbox = $("input:checkbox:checked").val();
    $("p").empty().text(checkbox);
  });
  $("#getRadio").click(function(){
    var radio = $("input:radio[name=r]:checked").val();
    $("p").empty().text(radio);
  });
});
</script>
</head>
<body>
<p></p>
<select id="single">
  <option value="s1">Single1</option>
  <option value="s2">Single2</option>
  <option value="s3">Single3</option>
  <option value="s4">Single4</option>
  <option value="s5">Single5</option>
</select>
<select id="multiple" multiple="multiple">
  <option value="m1" selected="selected">Multiple1</option>
  <option value="m2">Multiple2</option>
  <option value="m3" selected="selected">Multiple3</option>
  <option value="m4">Multiple4</option>
  <option value="m5">Multiple5</option>
</select><br />
<input type="checkbox" name="c" value="check" checked="checked"
/><label>check</label><br />
<input type="radio" name="r" value="radio1" /><label>radio1</label>
<input type="radio" name="r" value="radio2" /><label>radio2</label>
```

```
<input type="radio" name="r" value="radio3" checked="checked"
/><label>radio3</label><hr />
<button id="getSingle">select #single</button>
<button id="getMultiple">select #multiple</button>
<button id="getCheckbox">checkbox</button>
<button id="getRadio">radio</button>
</body>
</html>
```

Обсуждение

Рассмотрим код, приведенный в листинге 2.4.3. HTML-разметка представляет собой набор элементов, с которыми мы будем экспериментировать. В элемент `p` мы будем вставлять полученные значения. Далее следуют два элемента `select`, причем второй с возможностью выбора нескольких опций одновременно. Следом идут элемент `checkbox` и набор элементов `radio`. Заканчивается HTML-разметка четырьмя кнопками `button`, с помощью которых мы будем получать значения выбранных элементов.

Теперь разберем JavaScript-код. С каждой кнопкой мы связываем событие `click` и по наступлению этого события вызываем функцию-обработчик. Отметьте для себя то обстоятельство, что в функции-обработчике везде вызывается один и тот же метод `val()`, вопрос только в том, чтобы грамотно отыскать именно те элементы, к которым его нужно применить.

Давайте подробно рассмотрим, что именно будет делать функция, вызванная при щелчке на кнопке с идентификатором `#getMultiple`.

Первое — в переменной `multiple` сохраняются значения выбранных опций элемента `select`. Поскольку выбранных опций может быть несколько, в этом случае метод `val()` вернет массив значений. Второе — отыскивается элемент `p`, из которого с помощью метода `empty()` на всякий случай удаляется все внутреннее содержимое, а затем в него же в виде текста вставляются значения из `multiple`, предварительно объединенные в строку с помощью метода `join()`.

ЗАДАЧА

При загрузке веб-страницы необходимо установить начальные значения элементов `select`, `checkbox` и `radio`.

Решение

Используем метод `val(val)` для решения задачи (листинг 2.4.4).

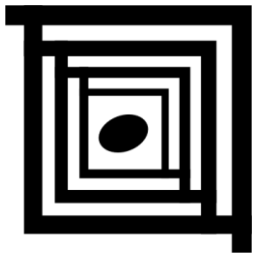
Листинг 2.4.4. Использование метода `val(val)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-2-4-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#single").val("Single3");
  $("#multiple").val(["Multiple2", "Multiple4", "Multiple5"]);
  $("input").val(["check", "radio3"]);
});
</script>
</head>
<body>
<select id="single">
  <option>Single1</option>
  <option>Single2</option>
  <option>Single3</option>
  <option>Single4</option>
  <option>Single5</option>
</select>
<select id="multiple" multiple="multiple">
  <option selected="selected">Multiple1</option>
  <option>Multiple2</option>
  <option selected="selected">Multiple3</option>
  <option>Multiple4</option>
  <option>Multiple5</option>
</select><br />
<input type="checkbox" name="c" value="check" /><label>check</label><br />
<input type="radio" name="r" value="radio1" /><label>radio1</label>
<input type="radio" name="r" value="radio2" /><label>radio2</label>
<input type="radio" name="r" value="radio3" /><label>radio3</label>
</body>
</html>
```

Обсуждение

HTML-код из листинга 2.4.4 — набор элементов `select`, `checkbox` и `radio`. Самое интересное не здесь, а в JavaScript-коде. В момент готовности DOM мы выполняем несколько строк кода. Например, сначала выбираем элемент `select` с идентификатором `#single` и применяем к нему метод `val(val)`, которому в качестве аргумента передаем строку `"Single3"`, устанавливая, таким образом, необходимое значение. Аналогично поступаем и с элементом `#multiple`, только в качестве аргумента методу `val(val)` передается массив значений. Для элементов `checkbox` и `radio` мы пользуемся тем, что с учетом нашей HTML-разметки их можно выбрать с помощью селектора `input` и задать значения для них одной строкой кода.



ГЛАВА 3

Визуальные эффекты

С помощью визуальных эффектов можно придать странице элементы интерактивности, показать пользователю, что страница реагирует на его действия. Визуальные эффекты — это не только переключение состояния видимости/невидимости элементов. Имея возможность управлять значениями большинства CSS-свойств элементов, можно создавать свою собственную анимацию.

3.1. Как показывать и скрывать элементы

ЗАДАЧА

Необходимо скрывать, показывать или переключать состояние видимости/невидимости для какого-либо элемента.

Решение

Для решения задачи воспользуемся имеющимися в составе библиотеки jQuery методами `hide()`, `show()` и `toggle()` (листинг 3.1.1).

Листинг 3.1.1. Использование методов `hide()`, `show()` и `toggle()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { border:1px dotted #369; padding:5px; color:#00f; }
</style>
<script type="text/javascript">
```

```
$(function(){
  $("button:eq(0)").click(function(){
    $("#test").hide();
  });
  $("button:eq(1)").click(function(){
    $("#test").show();
  });
  $("button:eq(2)").click(function(){
    $("#test").toggle();
  });
});
</script>
</head>
<body>
<button>Hide</button><button>Show</button><button>Toggle</button>
<div id="test">
<p>Я шел, спускаясь в темные коридоры, и потом опять поднимаясь наверх. Я был один. Я кричал, мне не отвечали — я был один в этом обширном, запутанном, как лабиринт, доме.</p>
<p><em>Ги де Мопассан</em></p>
</div>
</body>
</html>
```

Обсуждение

В листинге 3.1.1 HTML-код описывает три кнопки (с их помощью будем вызывать необходимые методы) и элемент `div` с идентификатором `test` и некоторым содержимым (им мы будем манипулировать).

Рассмотрим JavaScript-код. Все довольно просто — с каждой кнопкой связывается событие `click`, при наступлении которого вызывается соответствующий метод для элемента, имеющего идентификатор `test`. В итоге, при нажатии на кнопку **Hide** элемент будет скрыт, при нажатии на кнопку **Show** — показан. А каждое нажатие на кнопку **Toggle** будет приводить к скрытию и отображению элемента.

ЗАДАЧА

Необходимо не только скрывать, показывать или переключать состояние видимости/невидимости элемента, но и делать это в течение какого-то, заранее заданного времени. По истечении этого времени элемент должен быть скрыт/показан и дополнительно должна быть вызвана какая-либо функция.

Решение

Для решения используем уже знакомые методы `hide()`, `show()` и `toggle()`. Однако здесь мы учтем то обстоятельство, что эти методы могут принимать некоторые аргументы (листинг 3.1.2).

Листинг 3.1.2. Использование методов `hide()`, `show()` и `toggle()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { border:1px dotted #369; padding:5px; color:#00f; }
.result { border-bottom:1px dotted #369; }
</style>
<script type="text/javascript">
$(function(){
  $("button:eq(0)").click(function(){
    $("#test").hide(1000,function(){ $("p:first").text("hide"); });
  });
  $("button:eq(1)").click(function(){
    $("#test").show(2000,function(){ $("p:first").text("show"); });
  });
  $("button:eq(2)").click(function(){
    $("#test").toggle(800,function(){ $("p:first").text("toggle"); });
  });
});
</script>
</head>
<body>
<p class="result"> </p>
<button>Hide</button><button>Show</button><button>Toggle</button>
<div id="test">
<p>Я шел, спускаясь в темные коридоры, и потом опять поднимаясь наверх. Я был
один. Я кричал, мне не отвечали — я был один в этом обширном, запутанном, как
лабиринт, доме.</p>
<p><em>Ги де Мопассан</em></p>
</div>
</body>
</html>
```

Обсуждение

В HTML-коде, приведенном в листинге 3.1.2, к кнопкам **Hide**, **Show**, **Toggle** и элементу `div` с идентификатором `test` добавился параграф `p`. Немного позже поясним, для чего он понадобился.

Что касается JavaScript-кода, то точно так же, как в примере из листинга 3.1.1, с каждой кнопкой связывается событие `click`, при наступлении которого для эле-

мента с идентификатором `test` вызывается метод, соответствующий его названию на кнопке. Но, в отличие от примера из листинга 3.1.1, методам передаются аргументы. Первый аргумент — число в миллисекундах, определяющее интервал, в течение которого будет происходить скрытие/отображение элемента. Второй аргумент — функция, которая будет вызвана при завершении этого процесса. А элемент `p` в HTML-разметке требуется только для того, чтобы вставить туда текст с названием использованного метода.

Примечание

Методы `hide()`, `show()` и `toggle()` могут принимать еще один аргумент — название эффекта плагина `easing`, с использованием которого может происходить скрытие/отображение элемента, например `show(3000, 'swing', fn)`. По умолчанию возможны только эффекты `swing` и `linear`. Чтобы получить больше эффектов, необходимо подключение плагина `easing`. Подробнее см. **http://jqueryui.com/demos/effect/#easing**.

3.2. Эффекты "скольжения" и "затухания"

ЗАДАЧА

Необходимо скрывать или показывать элементы с использованием эффекта "скольжения".

Решение

Для решения в арсенале библиотеки jQuery найдутся соответствующие методы — `slideUp()`, `slideDown()` и `slideToggle()` (листинг 3.2.1).

Листинг 3.2.1. Использование методов `slideUp()`, `slideDown()` и `slideToggle()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { border:1px dotted #369; padding:5px; color:#00f; }
.result { border-bottom:1px dotted #369; }
</style>
<script type="text/javascript">
$(function(){
  $("button:eq(0)").click(function(){
    $("#test").slideUp(1000,function(){ $("p:first").text("slideUp"); });
  });
```

```
  $("button:eq(1)").click(function(){
    $("#test").slideDown(2000,function(){ $("p:first").text("slideDown"); });
  });
  $("button:eq(2)").click(function(){
    $("#test").slideToggle(800,function(){ $("p:first").text("slideToggle");
});
  });
});
</script>
</head>
<body>
<p class="result"> </p>
<button>Hide</button><button>Show</button><button>Toggle</button>
<div id="test">
<p>Я шел, спускаясь в темные коридоры, и потом опять поднимаясь наверх. Я был
один. Я кричал, мне не отвечали — я был один в этом обширном, запутанном, как
лабиринт, доме.</p>
<p><em>Ги де Мопассан</em></p>
</div>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 3.2.1, в общем уже знаком — три кнопки **Show**, **Hide** и **Toggle**, элемент `div`, над которым мы ставим свои эксперименты, и параграф `p`, куда вставляем какие-либо сообщения.

JavaScript-код тоже очень похож на код из листинга 3.1.2 — точно так же по событию `click` каждой из кнопок вызываются соответствующие методы — `slideUp()`, `slideDown()` и `slideToggle()`, которые применяются к элементу с идентификатором `test`. Методам передаются аргументы: первый — число, определяющее длительность (в миллисекундах) эффекта, и второй — функция, вызываемая при завершении процесса. Точно так же, как и в примере из листинга 3.1.2, эта функция просто вставляет название использованного метода в элемент `p`.

> ***Примечание***
>
> Методы `slideUp()`, `slideDown()` и `slideToggle()` могут принимать еще один аргумент — название эффекта плагина `easing`, с использованием которого может происходить скрытие/отображение элемента, например `slideDown (3000, 'swing', fn)`. По умолчанию возможны только эффекты `swing` и `linear`. Больше эффектов даст подключение плагина `easing`. Подробнее см. **http://jqueryui.com/demos/effect/#easing**.

ЗАДАЧА

Необходимо скрывать и показывать элементы с использованием эффекта "затухания".

Решение

Для решения применим методы `fadeOut()`, `fadeIn()` и `fadeToggle()` (листинг 3.2.2).

Листинг 3.2.2. Использование методов `fadeOut()`, `fadeIn()` и `fadeToggle()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { border:1px dotted #369; padding:5px; color:#00f; }
.result { border-bottom:1px dotted #369; }
</style>
<script type="text/javascript">
$(function(){
  $("button:eq(0)").click(function(){
    $("#test").fadeOut(2500,function(){ $("p:first").text("fadeOut"); });
  });
  $("button:eq(1)").click(function(){
    $("#test").fadeIn(2500,function(){ $("p:first").text("fadeIn"); });
  });
  $("button:eq(2)").click(function(){
    $("#test").fadeToggle(800,function(){
        $("p:first").text("fadeToggle");
    });
  });
});
</script>
</head>
<body>
<p class="result"> </p>
<button>Hide</button><button>Show</button><button>Toggle</button>
<div id="test">
<p>Я шел, спускаясь в темные коридоры, и потом опять поднимаясь наверх. Я был
один. Я кричал, мне не отвечали — я был один в этом обширном, запутанном, как
лабиринт, доме.</p>
<p><em>Ги де Мопассан</em></p>
</div>
</body>
</html>
```

Обсуждение

HTML-код в листинге 3.2.2 нам тоже знаком. В JavaScript-коде можно легко разобраться на основании предыдущих примеров. Методы `fadeIn()`, `fadeOut()` и

fadeToggle() принимают такие же аргументы — время, за которое выполняется эффект, и функция, вызываемая при завершении эффекта.

> ***Примечание***
>
> Методы fadeIn(), fadeOut() и fadeToggle() могут принимать еще один аргумент — название эффекта плагина easing, с использованием которого может происходить скрытие/отображение элемента, например fadeIn(3000, 'swing', fn). По умолчанию возможны эффекты swing и linear. Больше эффектов даст подключение плагина easing. Подробнее см. **http://jqueryui.com/demos/effect/#easing**.

ЗАДАЧА

При щелчке указателем мыши на каком-либо элементе необходимо плавно изменить его прозрачность к заданному значению и при его достижении вызвать функцию, которая выведет сообщение о завершении этого процесса.

Решение

Решить такую задачу поможет метод fadeTo() (листинг 3.2.3).

Листинг 3.2.3. Использование метода fadeTo()

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-2-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { border:1px dotted #369; padding:5px; color:#00f; }
.result { border-bottom:1px dotted #369; }
</style>
<script type="text/javascript">
$(function(){
  $("p:first").one("click", function(){
    $(this).fadeTo("slow",0.33,function(){ alert("Готово!"); });
  });
});
</script>
</head>
<body>
<p>Первый параграф — щелкните левой кнопкой мыши здесь.</p>
<p>Второй параграф — для сравнения.</p>
</body>
</html>
```

Обсуждение

С помощью HTML-кода, приведенного в листинге 3.2.3, описано два параграфа. По умолчанию оба параграфа со всем своим содержимым имеют прозрачность равную единице, т. е. непрозрачны.

Что мы делаем в JavaScript-коде? Находим первый параграф с помощью указанного в селекторе выражения `p:first` и назначаем ему обработчик события `click`, который выполнится только один раз. При наступлении этого события вызываем для данного параграфа метод `fadeTo()`, которому передаем аргументы. Первый аргумент `slow` — строка, определяющая продолжительность эффекта (возможны строковые значения `slow`, `normal` и `fast` или число миллисекунд). Второй аргумент — число, определяющее требуемую прозрачность элемента (от 0 до 1). Третий аргумент — функция, вызываемая при завершении работы.

В итоге, щелкнув мышью по первому параграфу, мы увидим, как он "побледнеет", и после этого появится окно предупреждения с текстом "Готово!".

3.3. Создание анимации

ЗАДАЧА

Необходимо создать простую анимацию для элемента `div` — при нажатии кнопок элемент должен сдвигаться влево или вправо на 50 пикселов.

Решение

Для решения задачи используем метод `animate()` (листинг 3.3.1).

Листинг 3.3.1. Использование метода `animate()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  position:absolute;
  background-color:#369;
  left:100px;
  width:90px;
  height:90px;
  margin:5px;
}
</style>
```

```
<script type="text/javascript">
$(function(){
  $("#right").click(function(){
    $(".block").animate({"left": "+=50px"}, "slow");
  });
  $("#left").click(function(){
    $(".block").animate({"left": "-=50px"}, "slow");
  });
});
</script>
</head>
<body>
<button id="left">влево</button><button id="right">вправо</button>
<div class="block"></div>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 3.3.1, определен условием задачи — кнопки **влево** и **вправо**, и элемент `div`, который необходимо двигать при их нажатии. Обработчики события `click` для кнопок очень похожи. В обоих случаях мы выбираем элемент по имени его класса — `block` и применяем метод `animate()`. Этот метод в состоянии принимать до четырех аргументов, из которых мы используем только два.

В первом аргументе передаем объект, состоящий из пар ключ/значение, которые определяют различные CSS-свойства элемента. В нашем случае мы пользуемся только свойством `left`, задавая его значение на 50 пикселов больше (или меньше) предыдущей величины.

Во втором аргументе передаем скорость выполнения эффекта в виде строкового значения. Можно указать и число (в миллисекундах), определяющее длительность эффекта.

В качестве третьего и четвертого аргументов можно передать название эффекта и функцию, вызываемую по окончании анимации.

ЗАДАЧА

При нажатии кнопки необходимо выполнить анимацию только одного элемента `div`, но в процессе анимации должны измениться значения многих его CSS-свойств и по завершении анимации необходимо выдать сообщение о ее окончании. В процессе выполнения анимации может появиться необходимость в ее остановке.

Решение

В решении задачи помогут методы `animate()` и `stop()` (листинг 3.3.2).

Листинг 3.3.2. Использование методов `animate()` и `stop()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px;
  border:1px solid #369;
  padding:5px;
}
</style>
<script type="text/javascript">
$(function(){
  $("#go").click(function(){
    $("#block").animate({
      "width": "70%",
      "opacity": 0.3,
      "marginLeft": "5em",
      "fontSize": "3em",
      "borderWidth": "10px"
    }, 2500, function(){ alert("Сделано!"); } );
  });
  $("#stop").click(function(){
    $("#block").stop();
  });
});
</script>
</head>
<body>
<button id="go">Выполнить</button>
<button id="stop">Остановить</button>
<div id="block">Hello!</div>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 3.3.2, очень простой — элемент `div`, над которым будем ставить эксперименты, и две кнопки, запускающие и останавливающие анимацию.

Поработаем с JavaScript-кодом. Сначала попробуем запустить анимацию. Для этого связываем обработчик события `click` с кнопкой `#go`. Функция-обработчик найдет

необходимый элемент по его идентификатору `#block` и применит к нему метод `animate()`. Первый аргумент метода — объект, где мы описываем CSS-свойства элемента и их значения, которые они должны получить по окончании анимации.

> ***Внимание!***
>
> Обратите внимание, как записывают имена некоторых свойств — свойства, названия которых состоят из двух слов, пишут не через дефис, как принято в CSS, а вместе, причем второе слово — с большой буквы. Жаргонное название такой формы записи — camel case.

Второй аргумент метода — длительность выполнения анимации, заданная в миллисекундах. Третий аргумент — функция, которая будет вызвана при завершении анимации и покажет окно предупреждения с надписью "Готово!".

Основная часть задачи выполнена, осталось только решить вопрос с принудительной остановкой анимации. Связываем обработчик события `click` с кнопкой `#stop`. При наступлении этого события находим нужный элемент `div` по его идентификатору `#block` и вызываем метод `stop()`, который остановит анимацию.

ЗАДАЧА

Необходимо построить относительно сложную анимацию, где некоторые свойства элемента будут меняться одновременно с другими, а некоторые — в порядке очереди.

Решение

Для решения задачи подходит все тот же метод `animate()` (листинг 3.3.3), но в этом примере мы воспользуемся возможностью передать ему в качестве аргументов два объекта. Первый объект содержит названия и значения CSS-свойств, второй — дополнительные настройки.

Листинг 3.3.3. Использование метода `animate()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  position:absolute;
  background-color:#69c;
  left:100px;
  width:90px;
```

```
  height:90px;
  margin:5px;
  border:1px solid #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("#go").click(function(){
    $("div").animate({ width:"300px" }, { queue:false, duration:3000 })
            .animate( { height:"400px" }, 1500 )
            .animate( { borderWidth:"15px" }, 1500);
  });
});
</script>
</head>
<body>
<button id="go">Выполнить</button>
<div class="block"></div>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 3.3.3, реализует кнопку, при щелчке на которой будет запускаться анимация, и элемент `div`, который мы будем анимировать.

С кнопкой связываем обработчик события `click`. Функция-обработчик, вызываемая при наступлении события, сначала отыщет элемент `div`, к которому мы применим цепочку из нескольких вызовов метода `animate()`, реализовав, таким образом, последовательное выполнение эффектов. Но если эффекты увеличения высоты и ширины рамки действительно выполняются строго друг за другом в течение полутора секунд каждый, то эффект увеличения ширины выполняется параллельно с ними в течение трех секунд. Мы добились такого поведения, передав во втором аргументе объект с настройками анимации — опция `queue:false` исключила эффект из общей очереди, а опция `duration:3000` задала время выполнения эффекта, равное трем секундам.

Также в этом объекте возможны и некоторые другие свойства. Например, в свойстве `step` можно определить функцию, которая будет вызываться после каждого шага анимации, а в свойстве `complete` — функцию, которая будет вызвана по завершении всей очереди.

ЗАДАЧА

При выполнении анимации необходимо задать некоторую задержку между двумя шагами очереди.

Решение

Для решения задачи воспользуемся методом `delay()` (листинг 3.3.4).

Листинг 3.3.4. Использование метода `delay()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-3-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { width: 60px; height: 60px; float: left; }
.first { background-color: #3f3; }
.second { background-color: #33f;}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function() {
    $("div.first").slideUp(700).fadeIn(1500);
    $("div.second").slideUp(700).delay(1000).fadeIn(1500);
  });
});
</script>
</head>
<body>
<p><button>Run</button></p>
<div class="first"></div>
<div class="second"></div>
</body>
</html>
```

HTML-код, приведенный в листинге 3.3.4, очень прост: два элемента `div` и кнопка `button`. С кнопкой связан обработчик события `click`, который выполняет анимацию для двух элементов. Разница в том, что для первого элемента `div` метод `fadeIn()` будет вызван сразу же после метода `slideUp()`. А во втором случае между этими методами "вклинится" задержка в 1 секунду.

3.4. Эффекты UI jQuery

ЗАДАЧА

Примеры, рассмотренные ранее в этой главе, слишком просты. Нужны более разнообразные эффекты.

Решение

Для решения применим эффекты UI jQuery — надстройки, которая расширяет возможности библиотеки. Таких эффектов всего 14, и они могут быть использованы как сами по себе, так и совместно с методами `show()`, `hide()` и `toggle()`. Реализацию эффекта `explode` совместно с методом `hide()` иллюстрирует листинг 3.4.1.

Листинг 3.4.1. Использование эффекта `explode` совместно с методом `hide()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-4-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.10.custom.min.js"
type="text/javascript"></script>
<style type="text/css">
div {
  position:absolute;
  top:200px; left:200px;
  width:200px; height:200px;
  border:1px solid #369; background-color:#69c;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").click(function(){
    $(this).hide(
      "explode",
      { pieces: 4 },
      1200,
      function(){ alert("Готово!"); }
    );
  });
});
</script>
</head>
<body>
<div></div>
</body>
</html>
```

Обсуждение

Рассмотрим код, который приведен в листинге 3.4.1, и сначала обратим внимание на раздел `head`. Помимо того, что мы подключили собственно библиотеку jQuery

(файл js/ jquery-1.4.4.min.js), нам понадобится еще один файл — js/jquery-ui-1.8.10.custom.min.js. В этом файле объединен код ядра надстройки Effects и необходимые эффекты, которые мы собираемся реализовать. Получить его можно на странице настраиваемой закачки **http://jqueryui.com/download**, отметив только необходимые файлы.

Весь HTML-код состоит из одного элемента `div`, которому с помощью CSS-свойств заданы ширина, высота, цвет фона, рамки и т. д.

Перейдем к JavaScript-коду. Для рассмотрения мы выбрали довольно интересный эффект `explode`, который можно комбинировать с методами `show()`, `hide()` или `toggle()`.

Сначала мы выбираем элемент `div` и связываем с ним событие `click`, при наступлении которого вызываем функцию-обработчик события. Что же делает эта функция? Она применяет к элементу, по которому был совершен щелчок левой кнопкой мыши, метод `hide()`, но посмотрите, каким именно образом: методу передается четыре аргумента и первый из них — это строка с названием эффекта. Второй аргумент — объект, определяющий значения возможных для используемого эффекта свойств. В данном случае свойство `pieces` со значением `4` заставит элемент `div` "рассыпаться" ровно на четыре части. Третий аргумент — длительность эффекта, указанная в миллисекундах. И четвертый — функция, вызываемая при завершении эффекта. Обязательными являются только первые два аргумента.

Посмотреть эффект вы можете или на компакт-диске, прилагаемом к книге, или на официальном сайте UI jQuery по адресу **http://jqueryui.com/demos/effect/**.

Листинг 3.4.2 иллюстрирует пример эффекта `bounce`.

Листинг 3.4.2. Использование эффекта `bounce`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-3-4-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="jquery-ui-1.8.10.custom.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  position:absolute;
  top:200px; left:200px;
  width:200px; height:200px;
  border:1px solid #369; background-color:#69c;
}
</style>
<script type="text/javascript">
```

```
$(function(){
  $("div").click(function(){
    $(this).effect(
      "bounce",
      { times:7, distance:50, direction:"right" },
      300,
      function(){ alert("Готово!"); }
    );
  });
});
</script>
</head>
<body>
<div></div>
</body>
</html>
```

Как и в примере из листинга 3.4.1, потребуется подключить дополнительный файл с функциональностью соответствующего эффекта. Воспользуйтесь страницей настраиваемой закачки на официальном сайте UI jQuery по адресу **http://jqueryui.com/download**.

В качестве "подопытного" элемента снова выберем `div`, при щелчке на котором левой кнопкой мыши вызовем функцию-обработчик этого события. Как только функция будет вызвана, она применит к элементу `div` метод `effect()`, которому будут переданы четыре аргумента (первый и второй — обязательные, третий и четвертый — необязательные). Первый аргумент — строка с названием эффекта. Второй аргумент — объект, определяющий некоторые свойства используемого эффекта. Третий аргумент — длительность эффекта. Четвертый — функция, вызываемая при завершении эффекта.

Посмотреть эффект вы можете или на компакт-диске, прилагаемом к книге, или на официальном сайте UI jQuery по адресу **http://jqueryui.com/demos/effect/**.

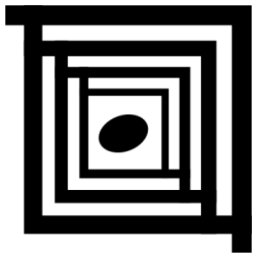
В табл. 3.1 для каждого эффекта приведены его доступные свойства.

Таблица 3.1. *Эффекты UI jQuery и их свойства*

Эффект	Свойства эффекта
`blind`	Имеет свойства `direction` и `mode`. Значения свойства `direction` — `vertical` (по умолчанию) или `horizontal`. Значения свойства `mode` — `show` или `hide` (по умолчанию)
`bounce`	Имеет свойства `direction`, `distance`, `mode` и `times`. Значения свойства `direction` — `left`, `right`, `up` (по умолчанию) и `down`. Значение свойства `distance` — число (по умолчанию — 20). Значения свойства `mode` — `show`, `hide` и `effect` (по умолчанию). Свойство `times` — число (по умолчанию — 5)
`clip`	То же, что и `blind`

Таблица 3.1 *(окончание)*

Эффект	Свойства эффекта
drop	Имеет свойства direction и mode. Значения свойства direction — left (по умолчанию), right, up, down. Значения свойства mode — show или hide (по умолчанию)
explode	Имеет свойства pieces и mode. Значение свойства pieces — число (по умолчанию — 9). Значения свойства mode — show или hide (по умолчанию)
fold	Имеет свойства horizFirst, mode и size. Значения свойства horizFirst — true и false (по умолчанию). Значения свойства mode — show или hide (по умолчанию). Значение свойства size — число (по умолчанию — 15)
highlight	Имеет свойства color и mode. Свойство color может принимать значения цвета подсветки (по умолчанию #ffff99). Значения свойства mode — show (по умолчанию) или hide
puff	Имеет свойства percent и mode. Значение свойства percent — число (по умолчанию — 150). Значения свойства mode — show или hide (по умолчанию)
pulsate	Имеет свойства times и mode. Значение свойства times — число (по умолчанию — 5). Значения свойства mode — show (по умолчанию) или hide
scale	Имеет свойства direction, from, origin, percent и scale. Значения свойства direction — both (по умолчанию), vertical и horizontal. Свойство from — объект {height:., width:.}, характеризующий начальные размеры. Обычно не требуется. Свойство origin — массив, описывающий точку исчезновения. По умолчанию — [middle, center]. Свойство percent — число (по умолчанию 0/100). Значения свойства scale — both (по умолчанию), box и content
shake	Имеет свойства direction, distance и times. Значения свойства direction — left (по умолчанию), right, up и down. Значение свойства distance — число (по умолчанию — 20). Свойство times — число (по умолчанию — 3)
size	Имеет свойства from, to, origin и scale. Свойство from — объект {height:., width:.}, характеризующий начальные размеры. Обычно не требуется. Свойство to — объект {height:., width:.}, характеризующий конечные размеры. Свойство origin — массив, описывающий точку исчезновения. По умолчанию — [middle, center]. Значения свойства scale — both (по умолчанию), box и content
slide	Имеет свойства direction, distance и mode. Значения свойства direction — left (по умолчанию), right, up и down. Значение свойства distance — число (по умолчанию — внешняя ширина элемента). Значения свойства mode — show или hide (по умолчанию)
transfer	Имеет свойства className и to. Свойство className — строка с именем класса. Свойство to — строка, содержащая jQuery-селектор



ГЛАВА 4

Работа с CSS-свойствами

4.1. Как получать и устанавливать значения CSS-свойств элементов

ЗАДАЧА

Необходимо решить задачу получения цвета фона какого-либо элемента, иными словами, значение CSS-свойства `background-color`.

Решение

Для решения такой задачи подойдет метод `css()` (листинг 4.1.1).

Листинг 4.1.1. Использование метода `css()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-4-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { width:60px; height:60px; margin:5px; float:left; }
</style>
<script type="text/javascript">
$(function(){
  $("div").click(function () {
    var color = $(this).css("background-color");
    $("#result").text("Значение свойства background-color = " + color);
  });
});
```

```
</script>
</head>
<body>
<span id="result"> </span>
<div style="background-color:blue;"></div>
<div style="background-color:rgb(15,99,30);"></div>
<div style="background-color:#123456;"></div>
<div style="background-color:#f11;"></div>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 4.1.1, описывает четыре элемента `div`, которым заданы разные цвета фона, и элемент `span`, необходимый нам лишь для вывода результатов.

В JavaScript-коде сначала выбираем все элементы `div`, указывая в селекторе название элемента, и связываем с ними функцию-обработчик события `click`. При наступлении этого события применяем к элементу, на котором был совершен щелчок, метод `css()`, а результат сохраняем в переменной `color`. Следующая строка кода вставляет в элемент `span` поясняющий текст и значение переменной `color`.

Конечно, этот пример можно было бы записать и короче, ведь нет никакой необходимости создавать переменную `color` только ради того, чтобы хранить промежуточный результат.

Передавая методу `css()` единственный аргумент — название CSS-свойства, мы можем получить значение этого свойства.

ЗАДАЧА

Необходимо установить цвет фона какого-либо элемента.

Решение

Снова решаем задачу с помощью метода `css()` (листинг 4.1.2).

Листинг 4.1.2. Использование метода `css()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-4-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
```

```
div { width:150px; height:150px; margin:5px; float:left; border:1px solid #369; }
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function () {
    var color = $("input").val();
    $("div").css("background-color", color);
  });
});
</script>
</head>
<body>
<input type="text" />
<button>Применить</button>
<div></div>
</body>
</html>
```

Обсуждение

В листинге 4.1.2 приведен HTML-код, описывающий элемент ввода, куда мы будем вводить код нужного цвета, кнопку, при нажатии на которую будет вызываться JavaScript-код, выполняющий требуемую операцию. Есть еще элемент `div`, для которого заданы некоторые CSS-свойства, такие как `width`, `height` или `border`. Для этого элемента мы будем изменять цвет фона.

Первое, что делает JavaScript-код при нажатии кнопки `button` — получает код цвета, введенный в элемент `input` с помощью метода `val()`, и сохраняет это значение в переменной `color`. Затем будет найден элемент `div`, для которого будет установлено значение CSS-свойства `background-color` с помощью метода `css()`.

Передавая методу `css()` два аргумента — название CSS-свойства и его значение, мы можем установить значение этого свойства.

ЗАДАЧА

Необходимо установить одновременно некоторые CSS-свойства какого-либо элемента, например, изменить визуальное представление элемента `a` в момент наведения на него указателя мыши.

Решение

И снова нам пригодится метод `css()` для решения этой задачи (листинг 4.1.3).

Листинг 4.1.3. Использование метода `css(properties)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-4-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
a { color:#000; background-color:#eee; }
</style>
<script type="text/javascript">
$(function(){
  $("a").hover(
    function(){
      $(this).css({
        "background-color":"#999",
        "color":"#fff",
        "font-weight":"bold"
      });
    },
    function(){
      $(this).css({
        "background-color":"#eee",
        "color":"#000",
        "font-weight":"normal"
      });
    }
  );
});
</script>
</head>
<body>
<a href="http://jquery.com/">Библиотека jQuery</a><br />
<a href="http://jqueryui.com/">UI jQuery</a><br />
<a href="http://blog.jqueryui.com/">Блог UI jQuery</a>
</body>
</html>
```

Обсуждение

В соответствии с условием задачи в HTML-коде, приведенном в листинге 4.1.3, присутствуют три элемента `a`, визуальное представление которых по умолчанию определено с помощью CSS.

Чтобы изменить отображение этих элементов, необходимо сначала отыскать их, что мы и делаем, указав в селекторе название тега. Затем применим к выбранным элементам метод `hover()`, который принимает в качестве аргументов две функции. Первая будет выполняться в момент наведения указателя мыши на элемент `a`, а вторая — при выходе его за пределы элемента. Если мы сможем внутри этих функций изменить сразу несколько значений CSS-свойств, то задача будет решена. А это сделать нетрудно.

В контексте рассматриваемых функций указатель `this` ссылается на тот элемент `a`, на который был наведен указатель мыши, соответственно, применив к объекту, на который ссылается указатель, метод `css()`, можно изменить сразу несколько CSS-свойств этого объекта, т. е. выбранного элемента `a`.

При наведении мыши на ссылку мы задаем темно-серый цвет фона, белый цвет шрифта и делаем его начертание полужирным, а в момент выхода указателя мыши за пределы ссылки возвращаем начальные значения (светло-серый фон, черный шрифт и отменяем полужирное начертание).

Передавая методу `css()` в качестве аргумента объект с названиями и значениями CSS-свойств, мы можем установить значения этих свойств.

4.2. Ширина и высота элементов

ЗАДАЧА

На разрабатываемой веб-странице подразумевается активная работа с размерами элементов. Необходимо получать и устанавливать размеры выбранных элементов.

Решение

Для решения таких задач библиотека jQuery предлагает целый набор методов, которые мы и используем (листинг 4.2.1).

Листинг 4.2.1. Использование методов `width()`, `height()` и др.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-4-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:120px; height:110px;
  margin:15px; padding:10px;
  border:1px dotted #f00;
  overflow:hidden;
}
button { width:170px; }
</style>
<script type="text/javascript">
$(function(){
  $("button:eq(0)").click(function(){
    $("input").val($("div").width());
  });
```

```
  $("button:eq(1)").click(function(){
    $("input").val($("div").height());
  });
  $("button:eq(2)").click(function(){
    $("input").val($("div").innerWidth());
  });
  $("button:eq(3)").click(function(){
    $("input").val($("div").innerHeight());
  });
  $("button:eq(4)").click(function(){
    $("input").val($("div").outerWidth(true));
  });
  $("button:eq(5)").click(function(){
    $("input").val($("div").outerHeight());
  });
  $("button:eq(6)").click(function(){
    $("div").width($("input").val()*1);
  });
  $("button:eq(7)").click(function(){
    $("div").height($("input").val()*1);
  });
});
</script>
</head>
<body>
<button>Получить width</button>
<button>Получить height</button>
<button>Получить innerWidth</button>
<button>Получить innerHeight</button>
<button>Получить outerWidth</button>
<button>Получить outerHeight</button>
<br /><br />
<input type="text" />
<button>Установить width</button>
<button>Установить height</button>
<div>Элемент div</div>
</body>
</html>
```

Обсуждение

Код, приведенный в листинге 4.2.1, — это демонстрационная веб-страница, с помощью которой мы познакомимся с методами, позволяющими получать и устанавливать ширину и высоту выбранных элементов. Страница состоит из набора кнопок, элемента ввода `input` и элемента `div`. Мы будем получать и изменять размеры элемента `div`, элемент `input` будет служить для ввода и вывода значений, а кнопки — для того, чтобы получать и устанавливать соответствующие значения.

Код JavaScript очень однообразен: для каждой кнопки определен обработчик события `click`. При наступлении события вызывается callback-функция, которая получает ширину (высоту) элемента `div` и вставляет это значение в поле ввода `input`. Отличаются только обработчики события для двух последних кнопок — они, наоборот, получают значение, содержащееся в поле ввода `input`, и устанавливают ширину (высоту) элемента `div`.

Поговорим о нюансах. Вы наверняка заметили, что значения, полученные из элемента `input`, умножаются на единицу. Дело в том, что метод `val()` возвращает строку, а методы `width(val)` и `height(val)` принимают число. Умножение на единицу — нехитрый прием в JavaScript, позволяющий преобразовать строковое значение в число.

А почему столько разных методов для получения ширины (высоты)? Здесь тоже все просто — метод `width()` получает ширину содержимого элемента, т. е. не учитывает ширину отступов (`padding`), рамок (`border`) и полей (`margin`). Метод `innerWidth()` учитывает ширину содержимого плюс отступы (`padding`). Метод `outerWidth()` — ширину содержимого, отступов (`padding`), рамок (`border`) и полей (`margin`). Аналогично обстоит дело и с методами, применяемыми для получения высоты элементов: `height()`, `innerHeight()` и `outerHeight()`.

Обратите внимание на значение, которое мы получаем в результате вызова метода `outerHeight()`. Попробуйте понять, почему оно не соответствует ожиданию. Подсказка: ответ нужно искать в руководстве по CSS.

4.3. Позиционирование элементов

ЗАДАЧА

Необходимо получить значения CSS-свойств `top` и `left` для выбранного элемента, причем относительно как документа, так и элемента-родителя.

Решение

Решить такую задачу помогут методы `offset()` и `position()` (листинг 4.3.1).

Листинг 4.3.1. Использование методов `offset()` и `position()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-4-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
#main {
  position:relative;
```

```
  top:20px; left:40px;
  width:200px; height:200px;
  border:1px solid #f00;
  padding:10px;
}
.inner {
  width:60px; height:60px;
  margin:5px; float:left;
  border:1px solid #369;
}
br { clear:left; }
p { margin-left:40px; }
</style>
<script type="text/javascript">
$(function(){
  $("div.inner").click(function(){
    var o = $(this).offset();
    $("p:first span").text("left: " + o.left + "px, top: " + o.top + "px");
    var p = $(this).position();
    $("p:last span").text("left: " + p.left + "px, top: " + p.top + "px");
  });
});
</script>
</head>
<body>
<div id="main">
  <div class="inner"></div>
  <div class="inner"></div>
  <br />
  <div class="inner"></div>
  <div class="inner"></div>
</div>
<br />
<p>Offset — <span>left: ?px, top: ?px</span></p>
<p>Position — <span>left: ?px, top: ?px</span></p>
</body>
</html>
```

Обсуждение

В HTML-разметке, приведенной в листинге 4.3.1, элемент `div` с идентификатором `#main` играет роль элемента-родителя для элементов `div`, имеющих класс `inner`. С помощью CSS для элементов заданы отступы, поля и рамки. Параграфы `p` с вложенными в них элементами `span` будут служить для отображения полученных значений CSS-свойств `top` и `left` для тех элементов `div`, находящихся внутри `div#main`, по которым был выполнен щелчок мышью.

Учитывая это, пишем JavaScript-код. Сначала ищем все элементы `div` с именем класса `inner`, указывая в селекторе выражение `div.inner`, а затем связываем с ними обработчик события `click`. Пользуясь тем, что в контексте функции указатель `this` ссылается на тот элемент, по которому был произведен щелчок, вызываем для него методы `offset()` и `position()`, а результаты сохраняем в переменных `o` и `p` соответственно.

Оба метода возвращают объекты, в свойствах `top` и `left` которых содержится необходимая информация. Но следует помнить, что в свойствах `top` и `left` объекта, возвращенного методом `offset()`, содержатся отступы относительно документа, а в соответствующих свойствах объекта, возвращенного методом `position()`, — относительно элемента-родителя.

Затем мы просто вставляем полученные результаты в элементы `span`, находящиеся внутри параграфов.

ЗАДАЧА

Для какого-либо элемента, использующего горизонтальную и вертикальную полосы прокрутки для отображения всего содержимого, необходимо при загрузке переместить вертикальную полосу прокрутки на 150 пикселов, а горизонтальную — на 300.

Решение

Найдутся подходящие методы и для решения такой задачи — `scrollTop()` и `scrollLeft()` (листинг 4.3.2).

Листинг 4.3.2. Использование методов `scrollTop()` и `scrollLeft()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-4-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
#demo {
  width:300px; height:200px;
  overflow:auto;
  border:1px solid #369;
}
p {
  width:1000px;
  background-color:#ddd;
  border-bottom:1px dotted #69c;
}
</style>
```

```
<script type="text/javascript">
$(function(){
  $("#demo").scrollTop(150).scrollLeft(300);
  $("button").click(function(){
    var d = $("#demo");
    alert("scrollTop = " + d.scrollTop() + ", scrollLeft = " + d.scrollLeft());
  });
});
</script>
</head>
<body>
<div id="demo">
  <p>Он поднял  палец ... приближался.</p>
  <p>Кровь застыла ... взором зверя,</p>
  <p>попавшего в ... сучьями,</p>
  <p>и все это ... самоотверженных.</p>
  <p>Воздух вокруг ... поворачиваться,</p>
  <p>и тут свершилось ... самые глаза.</p>
  <p>Лавр Федотович ... осинника.</p>
  <p>Справа от меня... действием.</p>
</div>
<button>Получить scrollTop и scrollLeft</button>
</body>
</html>
```

Примечание

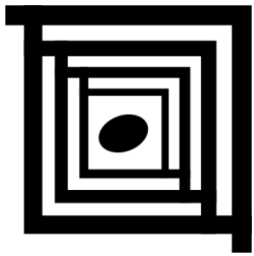
Часть текста в листинге 4.3.2 опущена (заменена многоточиями). В соответствующем примере на прилагаемом компакт-диске текст приведен полностью, чтобы продемонстрировать работу методов.

Обсуждение

HTML-код, приведенный в листинге 4.3.2, — это элемент `div` с идентификатором `#demo`, для которого с помощью CSS-свойств `width` и `height` заданы фиксированные значения ширины и высоты (300 и 200 пикселов соответственно). Значение `auto`, установленное для свойства `overflow`, определит появление вертикальной и горизонтальной полос прокрутки при необходимости. А такая необходимость будет, поскольку содержимое элемента `div` — несколько параграфов `p` шириной 1000 пикселов, что заведомо превышает размеры элемента `div` и по вертикали, и по горизонтали.

При загрузке страницы с помощью JavaScript-кода выбираем элемент `#demo` и применяем к нему методы `scrollTop()` и `scrollLeft()`, передавая им в виде аргументов числа 150 и 300. Когда страница загрузится, видим, что и вертикальная и горизонтальная полосы прокрутки находятся в требуемых положениях.

С помощью кнопки `button`, с которой связан обработчик события `click`, мы можем получить текущие значения. Попробуйте подвигать полосы прокрутки и нажать кнопку — в окне предупреждения будут выведены текущие значения.



ГЛАВА 5

Работа с данными в jQuery

5.1. Сохранение и извлечение данных

ЗАДАЧА

Требуется сохранить произвольные данные, связав их с набором элементов, чтобы иметь возможность несколько позже извлечь и использовать их. Необходимо также реализовать возможность удалить эти данные.

Решение

Для решения такой задачи подойдут методы `data(key,value)`, `data(key)` и `removeData([name])` (листинг 5.1.1).

Листинг 5.1.1. Использование методов `data()` и `removeData()`

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-5-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("div:first").data("employee", {
    name: "Иван",
    age: 27,
    phone: "+7(903)217-77-77"
  });
  $("button:first").click(function(){
    $("p span:eq(0)").text($("div:first")
                     .data("employee").name);
```

```
    $("p span:eq(1)").text($("div:first")
                      .data("employee").age);
    $("p span:eq(2)").text($("div:first")
                      .data("employee").phone);
  });
  $("button:last").click(function(){
    $("p span").empty();
    $("div:first").removeData("employee");
  });
});
</script>
</head>
<body>
<div></div>
<p>Сотрудник: <span></span>, возраст <span></span> лет, телефон:
<span></span></p>
<button>Получить данные</button><button>Удалить данные</button>
</body>
</html>
```

Обсуждение

Сначала рассмотрим HTML-код, приведенный в листинге 5.1.1. Элемент `div` послужит нам в качестве хранилища данных. В нашем примере он не имеет никакого оформления и содержимого, но это непринципиально — он мог быть оформлен очень красиво и содержать элементы-потомки. Следующий за ним элемент `p` с вложенными в него элементами `span` мы используем для отображения данных, а с помощью кнопок **Получить данные** и **Удалить данные** будем делать то, что собственно на кнопках и написано.

Перейдем к JavaScript-коду. При инициализации страницы выбираем элемент `div`, в котором собирались сохранить данные, и сохраняем их с помощью метода `data()`. Первый аргумент метода — имя, под которым данные будут сохранены в элементе и по которому к ним можно будет впоследствии обратиться. Второй аргумент — собственно данные. В нашем случае это объект, представляющий информацию на какого-то сотрудника (имя, возраст и контактный телефон).

Поскольку данные сохранены нами в элементе `div`, то в элементе `p` после загрузки страницы мы их видеть еще не можем. Там отображается только то, что определено HTML-кодом страницы.

Теперь необходимо извлечь данные из элемента `div` и вставить значение каждого свойства объекта данных в отведенное ему место — в элементы `span` внутри параграфа `p`. А заодно и убедиться в том, что данные действительно были сохранены.

Находим нужную кнопку с помощью выражения `button:first`, указанного в селекторе, и связываем с ней событие `click`, при наступлении которого вызывается функция-обработчик.

Что же делает эта функция? Сначала ищет первый элемент `span`, находящийся внутри параграфа `p`, и с помощью метода `text()` вставляет в него какой-то текст. Какой именно? В качестве аргумента записано относительно длинное выражение. Рассмотрим его подробнее.

Сначала с помощью выражения `div:first`, указанного в селекторе, находим элемент `div`, в котором были сохранены данные. И применяем к нему метод `data()`, которому в качестве аргумента передаем имя, под которым мы сохранили данные, — `employee`. Получив таким образом доступ к объекту, нам остается только обратиться к нужному свойству этого объекта — свойству `name`.

Остальные два элемента `span` заполняются подобным образом. В итоге мы увидим параграф `p`, заполненный нужными данными, — "Сотрудник: Иван, возраст 27 лет, телефон: +7(903)217-77-77".

Теперь возьмемся за удаление данных из элемента `div`. Сначала найдем кнопку **Удалить данные** с помощью выражения `button:last`, с которой свяжем обработчик события `click`. В функции-обработчике найдем и с помощью метода `empty()` очистим все элементы `span` внутри `p` (чтобы не создавалось впечатления, что данные не удалены). Затем отыщем элемент `div` и применим к нему метод `removeData()`, передав в качестве аргумента имя, под которым данные были сохранены.

Если теперь попробовать нажать кнопку **Получить данные**, это не произведет никакого эффекта, т. к. сохраненных данных в элементе `div` уже нет.

Необходимо отметить, что начиная с версии 1.4.3 метод `data()` может принимать в качестве единственного аргумента объект, содержащий данные, которые необходимо сохранить.

ЗАДАЧА

При создании веб-страницы в некоторых элементах используется атрибут `data`, определенный в спецификации HTML 5. Необходимо с помощью jQuery извлечь данные из этих элементов.

Решение

Снова применим метод `data()` (листинг 5.1.2).

Листинг 5.1.2. Использование метода `data()` в HTML 5

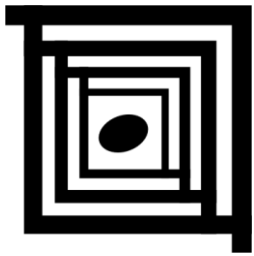
```
<!DOCTYPE html>
<html>
<head>
<title>example-5-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
```

```
<script type="text/javascript">
$(function(){
  $("li").hover(
    function(){
      var def = $(this).data("definition");
      var clr = $(this).data("color");
      var txt = $(this).text();
      $("p").text(txt + ' (' + def + ') - ' + clr);
    },
    function(){ $("p").empty(); }
  );
});
</script>
</head>
<body>
<ul>
  <li data-definition="животное" data-color="рыжая">Лиса</li>
  <li data-definition="птица" data-color="черный">Ворон</li>
  <li data-definition="дерево" data-color="белая">Береза</li>
</ul>
<p></p>
</body>
</html>
```

Обсуждение

В листинге 5.1.2 обратим внимание на элементы `li` списка `ul`. Точнее, на атрибуты `data` этих элементов. Таким образом можно хранить необходимую информацию непосредственно в HTML-разметке документа. А метод `data()` библиотеки jQuery позволит легко эти данные извлечь.

Посмотрим на JavaScript-код. При наведении указателя мыши на элемент `li` вызывается метод `data`, которому в качестве аргумента передается имя, под которым данные хранятся в разметке. Полученные данные вставляются в элемент `p` в виде текста. При выходе указателя мыши за пределы элемента `li` содержимое элемента `p` очищается.



ГЛАВА 6

Манипуляции над элементами

6.1. Изменение содержимого элементов

ЗАДАЧА

Необходимо вставить текст в какой-либо элемент.

Решение

Для решения задачи подойдет метод `text()` (листинг 6.1.1).

Листинг 6.1.1. Использование метода `text()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p {
  margin:5px;
  padding:5px;
  border-bottom:1px dotted #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("p").text("Если ввести здесь <a href='#'>html-код</a>, он отобразится как
обычный текст.");
});
</script>
```

```
</head>
<body>
<p></p>
</body>
</html>
```

Обсуждение

HTML-код из листинга 6.1.1 — это просто один элемент `p`, куда необходимо вставить текст.

Код JavaScript тоже весьма прост. Находим элемент `p` и применяем к нему метод `text()`, передавая в качестве аргумента строку.

> ***Внимание!***
> Обратите внимание, если в этой строке передать HTML-код, то он будет не выполнен, а отображен в качестве текста.

Начиная с версии 1.4, метод `text()` может принимать в качестве аргумента функцию, которая в свою очередь принимает два аргумента — индекс элемента в наборе и старый текст, содержащийся в элементе.

Эту возможность иллюстрирует пример, приведенный в листинге 6.1.2.

Листинг 6.1.2. Использование метода `text()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("ul li").text(function(index,text) {
    return text + ' - ' + (index + 1);
  });
});
</script>
</head>
<body>
<ul>
  <li>Первый пункт списка</li>
  <li>Второй пункт списка</li>
  <li>Третий пункт списка</li>
</ul>
</body>
</html>
```

После выполнения JavaScript-кода, приведенного в этом листинге, к тексту в каждом пункте списка будет добавлен порядковый номер.

ЗАДАЧА

Необходимо вставить HTML-код в какой-либо элемент.

Решение

Для решения задачи подойдет метод `html()` (листинг 6.1.3).

Листинг 6.1.3. Использование метода `html()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p {
  margin:5px;
  padding:5px;
  border-bottom:1px dotted #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("p").html("Если здесь ввести <a href='#'>html-код</a>, он будет
выполнен.");
});
</script>
</head>
<body>
<p></p>
</body>
</html>
```

Обсуждение

HTML-код из листинга 6.1.3 — это тот же самый элемент `p`, как и в листинге 6.1.1, но только теперь в него необходимо вставить HTML-код.

Такой же простой и код JavaScript. Находим элемент `p` и применяем к нему метод `html()`, передавая в качестве аргумента строку.

ВНИМАНИЕ!

Обратите внимание, если в этой строке передать HTML-код, он будет выполнен.

Точно так же, как и метод `text()`, метод `html()`, начиная с версии 1.4, может принимать в качестве аргумента функцию, которая в свою очередь принимает два аргумента — индекс элемента в наборе и старый HTML-код, содержащийся в элементе.

6.2. Как вставлять элементы в DOM

ЗАДАЧА

Необходимо вставить в какой-либо элемент содержимое, которое может быть как обычным текстом, так и HTML-кодом. Существующее в элементе содержимое необходимо сохранить, а новое содержимое вставить после старого.

Решение

Для решения используем метод `append()`(листинг 6.2.1).

Листинг 6.2.1. Использование метода `append()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p {
  margin:10px;
  border-bottom:1px dotted #369;
}
em {
  color:#00f;
  font-weight:bold;
}
</style>
<script type="text/javascript">
$(function(){
  $("p").append("<em>Привет! </em>");
});
</script>
</head>
<body>
<p>Я хочу сказать </p>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 6.2.1, представляет собой элемент `p`, который содержит небольшой текст. В JavaScript-коде мы сначала найдем этот элемент `p`, а затем применим к нему метод `append()`, передав ему в качестве аргумента строку с фрагментом HTML-кода. В результате данный фрагмент кода действительно окажется внутри параграфа `p`, после его содержимого.

Метод `append()` может принимать функцию в качестве аргумента. В свою очередь функция принимает в качестве аргументов индекс элемента в наборе и старый HTML-код, находившийся в элементе.

ЗАДАЧА

Необходимо вставить в какой-либо элемент содержимое, которое может быть как обычным текстом, так и HTML-кодом. Существующее в элементе содержимое необходимо сохранить, а новое содержимое вставить перед старым.

Решение

Для решения воспользуемся методом `prepend()` (листинг 6.2.2).

Листинг 6.2.2. Использование метода `prepend()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p {
  margin:10px;
  border-bottom:1px dotted #369;
}
em {
  color:#00f;
  font-weight:bold;
}
</style>
<script type="text/javascript">
$(function(){
  $("p").prepend("<em>Привет, </em>");
});
</script>
</head>
```

```
<body>
<p>говорю я Вам!</p>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 6.2.2, представляет собой элемент `p`, который содержит небольшой текст. В JavaScript-коде мы сначала найдем этот элемент `p`, а затем применим к нему метод `prepend()`, передав ему в качестве аргумента строку с фрагментом HTML-кода. В результате этот фрагмент кода действительно окажется внутри параграфа `p`, перед его содержимым.

Метод `prepend()` может принимать функцию в качестве аргумента. В свою очередь функция принимает в качестве аргументов индекс элемента в наборе и старый HTML-код, находившийся в элементе.

ЗАДАЧА

Необходимо выбрать некоторые элементы веб-страницы и поместить их в определенный элемент после существующего содержимого.

Решение

Решим эту задачу с помощью метода `appendTo()` (листинг 6.2.3).

Листинг 6.2.3. Использование метода `appendTo()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  margin:5px;
  padding:5px;
  border:1px dotted #369;
}
span {
  color:#fff;
  font-weight:bold;
  background-color:#e4ae00;
  margin:0 5px;
  padding:0 5px;
}
```

```
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function() {
    $("span").appendTo("#test");
  });
});
</script>
</head>
<body>
<span>Первый элемент span</span>
<span>Второй элемент span</span>
<div id="test">Элементы span будут вставлены сюда --></div>
<button>Вставить</button>
</body>
</html>
```

Обсуждение

В HTML-разметке из листинга 6.2.3 определены два элемента `span`, элемент `div` с идентификатором `#test` и элемент `button`. По щелчку на кнопке **Вставить** мы должны поместить элементы `span` в элемент `div` после существующего в нем текста.

В этом нам поможет простой JavaScript-код. Сначала найдем элемент `button` и свяжем с ним обработчик события `click`. При наступлении этого события будет вызвана функция-обработчик, которая выберет в набор элементы `span` и применит к этому набору метод `appendTo()`. Поскольку в качестве селектора мы передаем этому методу идентификатор `#test`, оба элемента `span` будут вставлены в `div` с таким идентификатором после существующего текста.

Необходимо отметить следующее — если выбранные элементы вставляются в единственный целевой элемент, то они будут перемещены туда. Иначе говоря, при повторном нажатии на кнопку **Вставить** ничего не произойдет.

ЗАДАЧА

Необходимо выбрать некоторые элементы веб-страницы и поместить их в другие элементы перед существующим содержимым.

Решение

При решении этой задачи используем метод `prependTo()` (листинг 6.2.4).

Листинг 6.2.4. Использование метода `prependTo()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-6-2-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  margin:5px;
  padding:5px;
  border:1px dotted #369;
}
span {
  color:#fff;
  font-weight:bold;
  background-color:#e4ae00;
  margin:0 5px;
  padding:0 5px;
}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function() {
    $("span").prependTo("div.test");
  });
});
</script>
</head>
<body>
<span>Первый элемент span</span>
<span>Второй элемент span</span>
<div class="test"><-- Сюда будут вставлены элементы span</div>
<div class="test"><-- Сюда будут вставлены элементы span</div>
<button>Вставить</button>
</body>
</html>
```

Обсуждение

В HTML-разметке из листинга 6.2.4 определены два элемента `span`, два элемента `div` с именем класса `test` и элемент `button`. По щелчку на кнопке **Вставить** мы должны поместить элементы `span` в элементы `div` перед существующим в них текстом.

Посмотрим на JavaScript-код. Сначала найдем элемент `button` и свяжем с ним обработчик события `click`. При наступлении этого события будет вызвана функция-обработчик, которая выберет в набор элементы `span` и применит к этому набору метод `prependTo()`. Поскольку в качестве селектора мы передаем этому методу выражение `div.test`, оба элемента `span` будут вставлены в оба элемента `div` с таким именем класса перед существующим текстом.

Необходимо отметить следующее — если выбранные элементы вставляются в несколько целевых элементов, то они будут перемещены туда, но при этом произойдет их копирование. Значит, при каждом следующем нажатии на кнопку **Вставить** число элементов `span` внутри каждого `div.test` будет удваиваться.

ЗАДАЧА

Необходимо вставить некоторый HTML-код до и после определенного элемента.

Решение

В решении этой задачи помогут методы `before()` и `after()` (листинг 6.2.5).

Листинг 6.2.5. Использование методов `before()` и `after()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
input { margin-bottom:2px; }
label { margin-right:10px; }
em { color:#f00; }
</style>
<script type="text/javascript">
$(function(){
  $("form input:text").before("<label>Имя:</label>")
                      .after("<em>*</em>");
});
</script>
</head>
<body>
<form action="#" method="post">
  <input type="text" name="name" /><br />
  <input type="submit" value="Отправить" /><br />
</form>
</body>
</html>
```

Обсуждение

В HTML-разметке, приведенной в листинге 6.2.5, имеется форма, состоящая из одного текстового поля ввода и кнопки **Отправить**. Нам необходимо вставить перед

элементом `input` типа `text` поясняющую надпись, а после него — символ `*`, чтобы отметить поле как обязательное для заполнения.

Посмотрим, как этого можно добиться с помощью JavaScript-кода. Сначала находим нужный элемент, указав в селекторе выражение `form input:text`, а затем применяем поочередно методы `before()` и `after()`. Первому передаем в качестве аргумента HTML-код, состоящий из элемента `label` с поясняющей надписью внутри него, а второму — HTML-код элемента `em` с символом `*`.

Теперь при загрузке страницы форма выглядит так, как и положено.

Методы `before()` и `after()` могут принимать функцию в качестве аргумента. В свою очередь функция принимает в качестве единственного аргумента индекс элемента в наборе.

ЗАДАЧА

Необходимо вставить некоторый HTML-код до и после определенного элемента.

Решение

Перед нами та же самая задача, решение которой уже было приведено в листинге 6.2.5. Но ее можно решить несколько иным способом (листинг 6.2.6), например, с помощью методов `insertBefore()` и `insertAfter()`.

Листинг 6.2.6. Использование методов `insertBefore()` и `insertAfter()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
input { margin-bottom:2px; }
label { margin-right:10px; }
em { color:#f00; }
</style>
<script type="text/javascript">
$(function(){
  $("<em>*</em>").insertAfter("form input:text");
  $("<label>Имя:</label>").insertBefore("form input:text");
});
</script>
</head>
<body>
<form action="#" method="post">
```

```
  <input type="text" name="name" /><br />
  <input type="submit" value="Отправить" /><br />
</form>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 6.2.6, описывает уже знакомую по предыдущему примеру форму, текстовое поле ввода которой необходимо снабдить поясняющей надписью перед этим полем и символом * после него.

Посмотрите на JavaScript-код. Здесь мы сначала определяем нужный HTML-код (* в элементе `em` и поясняющая надпись в элементе `label`), а затем вызываем методы `insertAfter()` и `insertBefore()`, передавая им в качестве аргумента одно и то же выражение — `form input:text`.

Выполнение данного JavaScript-кода приведет к такому же результату, как и в предыдущем примере — форма будет выглядеть так, как требовалось по условию задачи.

ЗАДАЧА

Требуется изменить имеющуюся HTML-разметку веб-страницы. Из нескольких элементов `a`, следующих друг за другом, необходимо создать ненумерованный список, поместив каждую ссылку в тег `<li>` и, кроме того, в тег `<strong>`. Само собой разумеется, что элементы `li` должны быть заключены в элемент `ul`.

Решение

Для решения такой задачи нам понадобится сразу несколько методов — `wrap()`, `wrapInner()` и `wrapAll()` (листинг 6.2.7).

Листинг 6.2.7. Использование методов `wrap()`, `wrapInner()` и `wrapAll()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
ul {
  width:200px;
  border:1px solid #666;
  padding:20px;
}
```

```
li {
  border-bottom:1px dotted #369;
  padding-left:5px;
}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function() {
    $("a").wrapInner("<strong></strong>").wrap("<li></li>");
    $("li").wrapAll("<ul></ul>");
  });
});
</script>
</head>
<body>
<a href="http://www.google.com">Google</a>
<a href="http://www.msn.com">MSN</a>
<a href="http://www.yandex.ru">Yandex</a>
<a href="http://www.rambler.ru">Rambler</a>
<button>Изменить разметку</button>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 6.2.7, — это просто четыре ссылки, следующие друг за другом. Кнопка с надписью **Изменить разметку** послужит для того, чтобы запустить выполнение эффекта — по щелчку на ней HTML-разметка изменится на глазах.

Самое интересное ждет нас при знакомстве с JavaScript-кодом. Здесь уже знакомый нам прием — находим элемент `button` и связываем с ним событие `click`. А вот функцию-обработчик этого события разберем подробно.

Сначала поработаем с элементами `a`. Выберем их все в набор, к которому применим метод `wrapInner()`, передав ему в качестве аргумента HTML-код элемента `strong`. После этой операции внутреннее содержимое элементов `a` окажется "обернутым" в элементы `strong`. Следующим в цепочке вызовов будет метод `wrap()`. Ему в качестве аргумента передается HTML-код элемента `li` и каждый элемент `a` оказывается уже сам "обернут" в элемент `li`.

Осталось "обернуть" все элементы `li` в `ul`. Поскольку теперь элементы `li` уже существуют, их можно очень просто отыскать, указав в селекторе название тега. Выбрав элементы `li` в набор, применяем к нему метод `wrapAll()`, передав в аргументе HTML-код элемента `ul`. Этот метод "обернет" все элементы набора в указанный HTML-код.

В итоге мы увидим как при щелчке на кнопке **Изменить разметку** преобразится наша веб-страница.

Начиная с версии 1.4, методы `wrap()` и `wrapInner()` могут принимать функцию в качестве аргумента. Эта функция не принимает аргументов. Указатель `this` внутри функции ссылается на элемент набора.

ЗАДАЧА

Необходимо периодически изменять имеющуюся HTML-разметку веб-страницы — оборачивая и снимая обертку с выбранных элементов.

Решение

Для решения понадобятся методы `wrap()` и `unwrap()` (листинг 6.2.8).

Листинг 6.2.8. Использование методов `wrap()` и `unwrap()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-2-8</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div { border: 2px solid #00f; }
p { background-color:#ff0; margin:4px; }
</style>
<script type="text/javascript">
$(function(){
  $("button").toggle(function(){
    $("p").wrap("<div></div>");
  }, function(){
    $("p").unwrap();
  });
});
</script>
</head>
<body>
<button>Обернуть/Развернуть</button>
<p>Hello</p>
<p>World</p>
</body>
</html>
```

Обсуждение

HTML-разметка из листинга 6.2.8 — это пара элементов `p`. Каждое нажатие на кнопку с надписью **Обернуть/Развернуть** будет оборачивать и снимать обертку

с элементов p. Оборачивать параграфы будем в элементы div, для которых с помощью CSS-правила зададим рамку синего цвета.

Теперь обратим внимание на JavaScript-код. С помощью метода toggle() можно поочередно вызывать функции, которые передаются этому методу в качестве аргументов (подробнее об этом вы узнаете в *главе 8*). Используем эту возможность — при первом нажатии кнопки находим все элементы p и оборачиваем их в div, при втором нажатии вызываем метод unwrap() и удаляем обертку с элементов p.

6.3. Замена, удаление и копирование элементов

ЗАДАЧА

На веб-странице необходимо заменить выбранные элементы определенным HTML-кодом.

Решение

В решении этой задачи поможет метод replaceWith() (листинг 6.3.1).

Листинг 6.3.1. Использование метода replaceWith()

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
button { display:block; width:200px; }
div {
  color:#369;
  border:1px solid #369;
  width:200px;
  margin:3px;
  text-align:center;
}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function() {
    $(this).replaceWith("<div>" + $(this).text() + "</div>");
  });
});
```

```
</script>
</head>
<body>
<button>Один</button>
<button>Два</button>
<button>Три</button>
</body>
</html>
```

Обсуждение

HTML-разметка в листинге 6.3.1 — это просто три кнопки с надписями на них. При щелчке мышью на кнопке необходимо будет заменить кнопку, т. е. элемент `button` на элемент `div`, причем текст, написанный на кнопке, должен оказаться внутри элемента `div`.

Как связать функцию-обработчик события `click` с элементом `button`, мы разбирали уже много раз, поэтому перейдем сразу к функции-обработчику. Внутри нее указатель `this` ссылается на ту кнопку, по которой был совершен щелчок. Следовательно, мы спокойно можем применять метод `replaceWith()`, а в качестве его аргумента укажем HTML-код элемента `div`. Между открывающим и закрывающим тегами `<div>` напишем выражение, получающее текст надписи на той кнопке, для которой была вызвана функция-обработчик.

Необходимо отметить, что, начиная с версии 1.4, метод `replaceWith()` может принимать функцию в качестве аргумента.

ЗАДАЧА

Требуется заменить некоторые элементы со всем их внутренним содержимым, некоторым HTML-кодом.

Решение

Практически та же задача, которую мы решали в листинге 6.3.1, но теперь мы применим для ее решения метод `replaceAll()` (листинг 6.3.2).

Листинг 6.3.2. Использование метода `replaceAll()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
```

```
<style type="text/css">
button { display:block; width:200px; }
p {
  border-bottom:1px dotted #369;
  color:#369;
}
div {
  border-bottom:1px dotted #00f;
  color:#00f;
}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function() {
    $("<div>Параграф заменен на div</div>").replaceAll("p");
  });
});
</script>
</head>
<body>
<p>Первый параграф</p>
<p>Второй параграф</p>
<p>Третий параграф</p>
<button>Заменить</button>
</body>
</html>
```

Обсуждение

HTML-разметка в листинге 6.3.2 — это просто три параграфа с некоторым текстом и кнопка **Заменить**. При щелчке мышью на кнопке необходимо будет заменить все элементы `p` на `div`.

Разбирая JavaScript-код, перейдем сразу к функции-обработчику события `click`. Сначала определяем HTML-код, которым мы будем заменять элементы `p`. А затем вызываем метод `replaceAll()`, указывая в качестве аргумента имя тега — `p`.

ЗАДАЧА

Требуется удалить все элементы-потомки, содержащиеся в выбранном элементе. Необходимо также удалить все выбранные элементы.

Решение

Для решения поставленной задачи воспользуемся методами `empty()` и `remove()` (листинг 6.3.3).

Листинг 6.3.3. Использование методов `empty()` и `remove()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p, div {
  width:200px; height:25px;
  border:1px dotted #369;
  padding-left:5px;
  margin:1px;
  color:#369;
}
</style>
<script type="text/javascript">
$(function(){
  $("button:first").click(function() {
    $("p").empty();
  });
  $("button:last").click(function() {
    $("p").remove();
  });
});
</script>
</head>
<body>
<p>Первый параграф</p>
<div>Первый div</div>
<p>Второй <em>параграф</em></p>
<div>Второй div</div>
<p><a href="#">Третий</a> параграф</p>
<div>Третий div</div><br />
<button>Очистить</button>
<button>Удалить</button>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 6.3.3, — это чередующиеся между собой элементы `p` и `div`, а также две кнопки — **Очистить** и **Удалить**. При щелчке мышью по кнопке **Очистить** будем удалять все содержимое элементов `p`, а при щелчке по кнопке **Удалить** — уничтожать сами элементы `p`.

Первую кнопку выбираем с помощью выражения `button:first`, вторую — `button:last`. С обеими кнопками связываем обработчики события `click`. И в первом, и во втором случае сначала выбираем все элементы `p`. Затем для первой кнопки применяем метод `empty()`, который очистит все содержимое выбранных в набор элементов. А для второй кнопки вызовем метод `remove()`, удаляющий и сами элементы.

ЗАДАЧА

Необходимо иметь возможность копировать элементы как сами по себе, так и вместе с обработчиками событий, связанными с этими элементами.

Решение

Решим задачу с помощью метода `clone()` (листинг 6.3.4).

Листинг 6.3.4. Использование метода `clone()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-3-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
button { width:120px; }
</style>
<script type="text/javascript">
$(function(){
  $("button:first").click(function() {
    $(this).clone().insertAfter(this);
  });
  $("button:last").click(function() {
    $(this).clone(true).insertAfter(this);
  });
});
</script>
</head>
<body>
<button>clone()</button>
<hr />
<button>clone(true)</button>
</body>
</html>
```

Обсуждение

В листинге 6.3.4 HTML-разметка описывает две кнопки. С каждой из них связаны обработчики событий `click`. Функции-обработчики тоже очень похожи — пользу-

ясь тем, что указатель `this` ссылается на ту кнопку, по которой был выполнен щелчок мышью, применяем к ней метод `clone()`, который вернет объект jQuery, представляющий клонированный элемент. Значит, мы имеем возможность продолжить цепочку вызовов. Делаем это, вызывая метод `insertAfter()`, вставляющий клонированный элемент, т. е. кнопку, после той, по которой был совершен щелчок.

Вся разница между первым и вторым обработчиками состоит в том, что в первом случае мы использовали метод `clone()` без передачи ему аргументов, а во втором передали в качестве аргумента значение `true`, указав, таким образом, на необходимость копировать не только сам элемент, но и связанные с ним обработчики событий.

Если вы испытаете этот пример на компакт-диске, прилагаемом к книге, то обнаружите, что при копировании кнопок с надписью **clone()** кнопки добавляются, но только при щелчке мышью на самой первой кнопке. При копировании кнопок с надписью **clone(true)** кнопки также копируются, но продолжать копирование можно щелкая по любой, даже вновь появившейся кнопке.

ЗАДАЧА

Необходимо удалить выбранные элементы, но оставить для дальнейшего использования информацию, связанную с этими элементами. Например, класс стилевого оформления, добавленный по какому-либо событию, или текст, содержащийся в удаляемом элементе.

Решение

Задачу будем решать с помощью метода `detach([selector])` (листинг 6.3.5).

Листинг 6.3.5. Использование метода `detach()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-6-3-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p, div {
  width:250px; height:25px; border:1px dotted #369;
  padding-left:5px; margin:1px; color:#369;
}
.off { background-color:#369; }
</style>
<script type="text/javascript">
$(function(){
  $("p").click(function(){
    $(this).toggleClass("off");
  });
```

```
  var p;
  $("button").click(function() {
    if(p) {
      p.insertAfter("div:first");
      p = null;
    } else {
      p = $("p").detach();
    }
  });
});
</script>
</head>
<body>
<div>Первый <strong>div</strong></div>
<p>Первый параграф</p>
<p id="two">Второй <em>параграф</em></p>
<p><a href="#">Третий</a> параграф</p>
<div>Второй <strong>div</strong></div>
<button>Удалить/вставить параграф</button>
</body>
</html>
```

Обсуждение

HTML-разметка, приведенная в листинге 6.3.5, — это несколько элементов `p` и `div`. Нас будут интересовать элементы `p`, которые содержат некоторый текст, а также дочерние элементы.

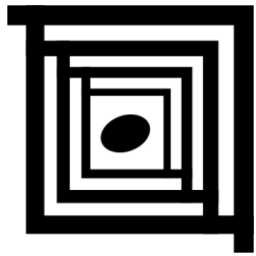
Если рассматривать JavaScript-код, то с элементами `p` мы связываем обработчик события `click`, в котором с помощью метода `toggleClass()` переключаем стилевое оформление элементов.

Затем объявляем переменную `p`, которая будет использоваться в обработчике нажатия кнопки **Удалить/вставить параграф**.

В зависимости от состояния переменной `p`, мы либо вставляем ее содержимое после первого элемента `div` в HTML-разметке, либо записываем результат выполнения метода `detach()`, примененного к набору элементов `p`, созданного с помощью соответствующего селектора.

Если теперь щелкать мышью на параграфах, то при каждом следующем щелчке будет изменяться их фон. Если же нажать на кнопку **Удалить/вставить параграф**, то параграфы будут удалены.

Самое интересное, что если нажать на ту же кнопку во второй раз, то исходное состояние всех параграфов будет восстановлено, т. е. параграфы будут содержать точно такой же текст и внутреннюю разметку, а также реагировать на событие `click`, связанное с ними.



ГЛАВА 7

Перемещение по элементам

7.1. Поиск нужных элементов в DOM

ЗАДАЧА

Для какого-либо известного элемента требуется найти один/все элементы, находящиеся в объектной модели документа непосредственно за/перед заданным элементом и на одном уровне с ним. Иными словами, необходимо немного "прогуляться по DOM".

Решение

Для решения задачи придется воспользоваться возможностями четырех методов библиотеки jQuery — `next()`, `prev()`, `nextAll()` и `prevAll()` (листинг 7.1.1).

Листинг 7.1.1. Использование методов `next()`, `prev()`, `nextAll()` и `prevAll()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:500px; padding:10px;
  margin:10px; border:2px solid #ccc;
}
p { border:2px solid #ccc; padding:0 5px; }
#main { background-color:#ddd; }
</style>
```

```
<script type="text/javascript">
$(function(){
  $("#main").next().css("border-color","#f00");
  $("#main").prev().css("border-color","#0c0");
  $("#main").nextAll().css("color","#f00");
  $("#main").prevAll().css("color","#0c0");
});
</script>
</head>
<body>
<p>Параграф p вне элемента div</p>
<p>Параграф p вне элемента div</p>
<div>
  <p>Параграф p внутри элемента div</p>
  <p>Параграф p внутри элемента div</p>
  <p>Параграф p внутри элемента div</p>
  <p id="main">Параграф p с идентификатором #main внутри элемента div</p>
  <p>Параграф p внутри элемента div</p>
  <p>Параграф p внутри элемента div</p>
  <p>Параграф p внутри элемента div</p>
</div>
<p>Параграф p вне элемента div</p>
<p>Параграф p вне элемента div</p>
</body>
</html>
```

Обсуждение

Рассмотрим HTML-разметку, показанную в листинге 7.1.1. Роль известного элемента, относительно которого будем осуществлять поиск, играет элемент `p` с идентификатором `#main`. На одном уровне DOM с ним, как до, так и после него, находятся еще несколько параграфов. Все эти параграфы, включая и параграф с идентификатором `#main`, — дочерние элементы по отношению к элементу `div`. В свою очередь, на одном уровне с элементом `div`, перед ним и после него, располагаются еще элементы `p`.

Перейдем к рассмотрению JavaScript-кода. Все четыре строки кода начинаются одинаково: с выбора известного нам элемента по его идентификатору `#main`. Применяя в первой и второй строках кода методы `next()` и `prev()`, мы находим соответственно следующий и предыдущий элементы, располагающиеся на том же уровне DOM, и отмечаем их рамками красного и зеленого цвета.

В третьей и четвертой строках кода мы применяем методы `nextAll()` и `prevAll()`, отыскивая соответственно все элементы, следующие за элементом с идентификатором `#main` и располагающиеся на том же уровне DOM в первом случае, и находящиеся перед ним и располагающиеся на том же уровне DOM — во втором случае. Отмечаем найденные элементы, устанавливая красный и зеленый цвета шрифта.

ЗАДАЧА

Для какого-либо заранее известного элемента необходимо выбрать все элементы, находящиеся за/перед заданным элементом и на одном уровне с ним, но только до того момента, пока не встретится другой элемент определенного типа. Причем этот элемент в набор попасть уже не должен.

Решение

Для решения такой задачи прекрасно подходят методы `nextUntil()` и `prevUntil()` (листинг 7.1.2).

Листинг 7.1.2. Использование методов `nextUntil()` и `prevUntil()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#term-2").nextUntil("dt").css("color", "#f00");
  $("#term-2").prevUntil("dt").css("color", "#00f");
});
</script>
</head>
<body>
<dl>
  <dt>термин 1</dt>
  <dd>определение 1-a</dd>
  <dd>определение 1-b</dd>
  <dd>определение 1-c</dd>
  <dd>определение 1-d</dd>
  <dt id="term-2">термин 2</dt>
  <dd>определение 2-a</dd>
  <dd>определение 2-b</dd>
  <dd>определение 2-c</dd>
  <dt>термин 3</dt>
  <dd>определение 3-a</dd>
  <dd>определение 3-b</dd>
</dl>
</body>
</html>
```

Обсуждение

Для проведения опытов выберем HTML-разметку, показанную в листинге 7.1.2. Здесь роль известного элемента, относительно которого будем осуществлять поиск, играет элемент `dt` с идентификатором `term-2`.

Соответственно в JavaScript-коде мы должны сначала выбрать этот элемент и затем применить к нему метод `nextUntil()`. Указывая в качестве аргумента название элемента `dt`, мы ограничим набор тремя элементами `dd`, следующими за элементом `dt` с идентификатором `term-2`. Затем, пользуясь методом `css()`, установим для них красный цвет шрифта.

Аналогично используем метод `prevUntil()`. Только теперь мы найдем элементы, которые предшествуют выбранному элементу `dt`. Для них установим синий цвет шрифта.

ЗАДАЧА

Для какого-либо известного элемента необходимо найти его непосредственного родителя и одного из элементов-потомков.

Решение

Для решения задачи понадобятся методы `parent()` и `children()` (листинг 7.1.3).

Листинг 7.1.3. Использование методов `parent()` и `children()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; padding:10px;
  margin:10px; border:2px solid #ccc;
}
p {
  border:2px solid #ccc; padding:0 5px;
  background-color:#ddd;
}
</style>
<script type="text/javascript">
$(function(){
  $("#main").parent().css("border-color","#f00");
  $("#main").children("span").css("background-color","#0c0");
});
```

```
</script>
</head>
<body>
<p>Параграф вне элемента div</p>
<div>
  <p>Параграф внутри элемента div</p>
  <p id="main"><em>Параграф</em> <strong>относительно</strong> которого
необходимо отыскать <span>элемент-потомок</span> и элемент-родитель.</p>
  <p>Параграф внутри элемента div</p>
</div>
<p>Параграф вне элемента div</p>
</body>
</html>
```

Обсуждение

Что представляет собой HTML-код из листинга 7.1.3? Параграф `p`, элемент `div` и снова параграф. Внутри элемента `div`, в свою очередь, находятся еще три параграфа `p`, в одном из которых встречаются элементы `em`, `strong` и `span`. Именно для этого параграфа с идентификатором `#main` необходимо найти и непосредственного родителя, и потомка, являющегося элементом `span`.

Познакомимся с JavaScript-кодом. В первой строке выбираем элемент с идентификатором `#main` и применяем к нему метод `parent()`, находя, таким образом, непосредственного родителя этого элемента, для которого установим рамку красного цвета.

Во второй строке точно так же выбрав элемент с идентификатором `#main`, применим к нему метод `children()`, передав в качестве аргумента название элемента — `span`. А чтобы было заметно, что мы действительно отыскали среди потомков нужный элемент, установим для него зеленый цвет фона.

ЗАДАЧА

Для какого-либо известного элемента необходимо найти один из его элементов-родителей определенного типа.

Решение

Попробуем решить эту задачу с помощью методов `parents()` и `closest()` (листинг 7.1.4), попутно выяснив, в чем же отличие этих очень похожих методов.

Листинг 7.1.4. Использование методов `parents()` и `closest()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-7-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
body { width:400px; }
body, h1, div, p, ul, li, a {
  padding:10px; margin:10px;
  border:2px solid #ccc;
}
a { display:block; }
</style>
<script type="text/javascript">
$(function(){
  $("a:first").click(function(){
    $("*").css("border-color","#ccc");
    $(this).parents("ul").css("border-color","#f0f");
  });
  $("a:last").click(function(){
    $("*").css("border-color","#ccc");
    $(this).closest("ul").css("border-color","#ff0");
  });
});
</script>
</head>
<body>
<h1>Заголовок</h1>
<p>Параграф вне элемента div</p>
<div>
  <p>Параграф внутри элемента div</p>
  <ul>
    <li><a href="#">Метод parents()</a></li>
    <li><a href="#">Метод closest()</a></li>
  </ul>
  <p>Параграф внутри элемента div</p>
</div>
<p>Параграф вне элемента div</p>
</body>
</html>
```

Обсуждение

Внимательно рассмотрите HTML-код из листинга 7.1.4 — в элементе `body` присутствуют некоторые элементы, среди которых нас будет интересовать список `ul`. Два элемента `li`, принадлежащие списку, содержат внутри себя элементы `a`. Именно для этих элементов `a` мы постараемся отыскать один из их элементов-родителей. Причем элемент-родитель должен быть элементом `ul`.

Посмотрим на JavaScript-код. С первым элементом `a` мы свяжем функцию-обработчик, которая будет использовать метод `.parents()`, а при щелчке на втором элементе `a` вызываем метод `.closest()`.

В функциях-обработчиках все довольно просто — поскольку указатель `this` ссылается на элемент, по которому был выполнен щелчок мышью, то, получив его, мы передаем название элемента `ul` в первом случае методу `parents()`, во втором — `closest()` и получаем искомого родителя. Далее, с помощью метода `css()` устанавливаем фиолетовую (в случае с `parents()`) и желтую (в случае с `closest()`) рамку найденному элементу. А первая строка кода в функциях-обработчиках просто устанавливает серый цвет рамки для всех элементов веб-страницы. Это делается для того, чтобы удалить фиолетовые рамки, если они были.

Так в чем же разница между методами `parents()` и `closest()`? Ведь визуально мы никаких отличий не обнаружили. Но визуальных отличий действительно нет — есть отличия технические.

Метод `parents()` поднимается по структуре DOM от непосредственного родителя выбранного элемента до корневого элемента документа, добавляя каждый родительский элемент во временную коллекцию, которую затем фильтрует на основании предоставленного селектора. Возвращает объект jQuery, содержащий ноль, один или несколько элементов.

Метод `closest()` поднимается по структуре DOM от самого элемента до момента обнаружения требуемого селектора. Возвращает объект jQuery, содержащий ноль или один элемент.

Метод `closest()` был добавлен в библиотеку, начиная с версии 1.3, и преимущества его использования вместо `parents()` в определенных случаях вполне очевидны.

ЗАДАЧА

Необходимо создать набор элементов, начиная с заданного. В набор должны попасть все элементы до родительского элемента какого-либо уровня.

Решение

Используем метод `parentsUntil()` для решения этой задачи (листинг 7.1.5).

Листинг 7.1.5. Использование методов `parentsUntil()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
```

```
ul.level-1 { width:300px; }
</style>
<script type="text/javascript">
$(function(){
  $("li.item-a").parentsUntil(".item-ii")
                .css("color", "#f00");
});
</script>
</head>
<body>
<ul class="level-1">
  <li class="item-i">I</li>
  <li class="item-ii">II
    <ul class="level-2">
      <li class="item-a">A</li>
      <li class="item-b">B
        <ul class="level-3">
          <li class="item-1">1</li>
          <li class="item-2">2</li>
          <li class="item-3">3</li>
        </ul>
      </li>
      <li class="item-c">C</li>
    </ul>
  </li>
  <li class="item-iii">III</li>
</ul>
</body>
</html>
```

Обсуждение

HTML-код в листинге 7.1.5 состоит из трех вложенных друг в друга списков `ul`. Заданным элементом будем считать элемент `li` с классом `item-a`, а родительским элементом, до которого необходимо "подняться", — элемент `li` с классом `item-ii`.

JavaScript-код начинается с того, что мы выбираем элемент `li` с классом `item-a` и применяем к нему метод `parentsUntil()`. В качестве аргумента этого метода указываем имя класса `item-ii`. Выбрав требуемые элементы, вызываем метод `css()`, чтобы установить шрифт красного цвета.

ЗАДАЧА

Для какого-либо известного элемента необходимо найти его сестринские элементы.

Решение

Решим задачу с помощью метода `siblings()` (листинг 7.1.6).

Листинг 7.1.6. Использование метода `siblings()`

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-1-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
body { width:400px; }
body, div, p {
  padding:10px; margin:10px;
  border:2px solid #ccc;
}
</style>
<script type="text/javascript">
$(function(){
  $("p").click(function(){
    $("*").css("border-color","#ccc");
    $(this).css("border-color","#f00")
           .siblings("p")
           .css("border-color","#f0f");
  });
});
</script>
</head>
<body>
<p>Параграф вне элемента div</p>
<div>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
</div>
<div>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
</div>
<p>Параграф вне элемента div</p>
</body>
</html>
```

Обсуждение

В листинге 7.1.6 HTML-разметка состоит из элементов `p` и `div`, причем внутри каждого элемента `div` имеется еще по три параграфа `p`.

С помощью JavaScript-кода будем находить все сестринские элементы для того элемента p, по которому будет совершен щелчок мышью (этот p выделим красной рамкой), и отмечать найденные элементы фиолетовой рамкой.

Итак, указываем в селекторе название элемента, который хотим найти — p, связываем со всеми найденными параграфами функцию-обработчик события click.

Внутри самой функции сначала устанавливаем всем элементам веб-страницы серую рамку, затем, выбрав с помощью указателя this тот элемент p, по которому был выполнен щелчок мышью, устанавливаем ему рамку красного цвета. И, наконец, применяем метод siblings() к набору, созданному с помощью все того же указателя this, получая все его сестринские элементы, и устанавливаем для них рамку фиолетового цвета методом css().

> ***Примечание***
>
> Отметьте, что в качестве необязательного аргумента мы передали название элемента, применив, таким образом, дополнительный фильтр — среди сестринских элементов остались только элементы p.

ЗАДАЧА

С помощью селектора jQuery создан набор элементов. Необходимо отыскать какой-либо элемент внутри этого набора.

Решение

Для решения такой задачи подойдет метод find() (листинг 7.1.7).

Листинг 7.1.7. Использование метода find()

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-1-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
p, div {
  padding:10px; margin:10px;
  border:1px solid #ccc;
}
</style>
<script type="text/javascript">
$(function(){
  $("p, div").find("span").css("color","#f00");
});
```

```
</script>
</head>
<body>
<p><span>Привет</span>, как дела?</p>
<p>У меня? Все отлично!</p>
<div>А как у <span>тебя</span>?</div>
<div>О! Тоже все <span>прекрасно</span>!</div>
</body>
</html>
```

Обсуждение

HTML-разметка, приведенная в листинге 7.1.7, — это два элемента `p` и два элемента `div`. Внутри некоторых из этих элементов присутствуют элементы `span`. Их и нужно будет отыскать.

В JavaScript-коде мы сначала создаем набор, состоящий из элементов `p` и `div`. К этому набору применяем метод `find()`, а в качестве аргумента указываем название нужного элемента — `span`. Далее устанавливаем красный цвет для текста, заключенного в элементы `span`.

7.2. Фильтрация элементов набора

ЗАДАЧА

Выбрав в набор некоторые элементы и выполнив над ними какие-либо операции, необходимо уменьшить набор, оставив в нем только те элементы, которые соответствуют определенному условию.

Решение

В решении такой задачи поможет метод `filter()` (листинг 7.2.1).

Листинг 7.2.1. Использование метода `filter()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:100px;
  margin:5px; float:left;
  border:3px double #369;
}
</style>
```

```
<script type="text/javascript">
$(function(){
  $("div").css("background","#c8ebcc")
  .filter(".test")
  .css("border-color", "#f00");
});
</script>
</head>
<body>
<div></div>
<div class="test"></div>
<div class="test"></div>
<div></div>
<div class="test"></div>
<div></div>
</body>
</html>
```

Обсуждение

HTML-код из листинга 7.2.1 — это шесть элементов `div`, половина из которых имеют атрибут `class` со значением `test`.

В JavaScript-коде мы сначала выбираем все элементы `div`, указывая в селекторе название элемента, и устанавливаем салатовый цвет фона. Дальше по цепочке вызовов следует вызов метода `filter()`. Аргумент, передаваемый методу, — селектор класса `.test`. После применения данного метода в наборе останутся только элементы `div` с этим классом. Для них будет установлена двойная рамка красного цвета.

ЗАДАЧА

Выбрав в набор некоторые элементы и выполнив над ними какие-либо операции, необходимо уменьшить набор, оставив в нем только те элементы, которые соответствуют определенному условию. Условия отбора элементов достаточно сложны, и их невозможно описать с помощью стандартных селекторов.

Решение

При решении такой задачи применим уже знакомый метод `filter()`, но воспользуемся тем обстоятельством, что в качестве аргумента этот метод может принимать функцию (листинг 7.2.2).

Листинг 7.2.2. Использование метода `filter()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-7-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:100px;
  margin:5px; float:left;
  border:3px double #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").css("background","#b4b0da")
          .filter(function(index){
            return index == 1 || $(this).attr("id") == "fourth";
          }).css("border", "3px double #f00");
});
</script>
</head>
<body>
<div id="first"></div>
<div id="second"></div>
<div id="third"></div>
<div id="fourth"></div>
<div id="fifth"></div>
<div id="sixth"></div>
</body>
</html>
```

Обсуждение

В HTML-разметке в листинге 7.2.2 присутствует шесть элементов `div`.

Что в JavaScript-коде? Сначала все очень похоже на решение задачи, приведенной в листинге 7.2.1, — точно так же выбираем все элементы `div` и устанавливаем для них цвет фона (только на этот раз — сиреневый).

Далее пробуем отфильтровать набор методом `filter()`. Аргументом метода выступает функция, возвращающая в наборе только те элементы, которые будут удовлетворять определенным условиям. Функция может принимать в качестве аргумента индекс элемента в исходном наборе. Итак, в нашем случае в наборе, который вернет функция, окажется элемент `div`, имеющий индекс 1 в исходном наборе, т. е. второй по счету, и элемент `div`, значение атрибута `id` которого соответствует `fourth`, т. е. четвертый по счету.

Затем, пользуясь методом `css()`, устанавливаем для этих двух элементов двойную рамку красного цвета.

ЗАДАЧА

Необходимо проверить, удовлетворяет ли элемент (или элементы) какому-либо условию, например, наличию определенного имени класса, или атрибута, или его состояния.

Решение

Решить задачу поможет метод `is()` (листинг 7.2.3).

Листинг 7.2.3. Использование метода `is()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:100px;
  margin:5px; float:left;
  border:3px double #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").click(function(){
    if($(this).is(".test")){
      alert("div имеет класс test");
    } else if($(this).is("#first, #fourth")) {
      alert("div имеет id #first или #fourth");
    } else {
      alert("Этот div не имеет класса test,\nне имеет id #first или #fourth");
    }
  });
});
</script>
</head>
<body>
<div id="first"></div>
<div id="second" class="test"></div>
<div id="third"></div>
<div id="fourth"></div>
<div id="fifth" class="test"></div>
<div id="sixth"></div>
</body>
</html>
```

Обсуждение

В листинге 7.2.3 приведена уже довольно знакомая HTML-разметка — шесть элементов `div`. Все элементы имеют атрибут `id`, а два из них еще и атрибут `class`.

Разберем JavaScript-код. Сначала выбираем все элементы `div`, с которыми связываем функцию-обработчик события `click`. Таким образом, мы будем проверять некоторые условия для того элемента `div`, на котором был совершен щелчок мышью.

Что же это за условия? Например, первое проверяемое условие — наличие атрибута `class` со значением `test`. Мы добиваемся этого, применяя метод `is()`, которому в качестве аргумента передаем селектор класса `".test"`. Если условие выполнено, то увидим окно предупреждения с соответствующим сообщением.

Следующее условие — наличие у элемента `div` какого-либо одного из значений идентификатора (`#first` или `#fourth`).

Подобным образом можно проверить и состояние элемента. Например, чтобы найти скрытый элемент, методу `is()` нужно передать выражение `":hidden"`.

ЗАДАЧА

Выбрав в набор некоторые элементы, необходимо его уменьшить, оставив только те элементы, которые не соответствуют определенному условию.

Решение

Решать задачу будем с помощью метода `not()` (листинг 7.2.4).

Листинг 7.2.4. Использование метода `not()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:100px;
  margin:5px; float:left;
  border:3px double #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").not(".test, #third").css("border-color","#f00");
});
```

```
</script>
</head>
<body>
<div id="first"></div>
<div id="second" class="test"></div>
<div id="third"></div>
<div id="fourth"></div>
<div id="fifth" class="test"></div>
<div id="sixth"></div>
</body>
</html>
```

Обсуждение

В листинге 7.2.4 опять та же самая HTML-разметка, состоящая из шести элементов `div`. Атрибут `id` присутствует у всех, атрибут `class` — только у двух элементов `div` из шести.

В JavaScript-коде создадим набор из всех элементов `div`. Попробуем удалить из набора элементы, которые имеют класс `.test` и значение атрибута `id #third`. Для этого применим к первоначальному набору метод `not()`, а в аргументе передадим выражение `".test, #third"`, которое удалит соответствующие этому правилу элементы из окончательного набора.

Для оставшихся в наборе элементов устанавливаем красную рамку с помощью метода `css()`. Проверяем — красную рамку получили первый, четвертый и шестой элементы `div`.

ЗАДАЧА

В существующем наборе необходимо выбрать некоторые элементы, например, с первого по третий или с третьего по пятый.

Решение

Для решения задачи очень хорошо подойдет метод `slice()` (листинг 7.2.5).

Листинг 7.2.5. Использование метода `slice()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
```

```
div {
  width:100px; height:100px;
  margin:5px; float:left;
  border:3px double #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").slice(2,5).css("border-color","#f00");
});
</script>
</head>
<body>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

HTML-код из листинга 7.2.5 — опять шесть элементов `div`. На этот раз без атрибутов — они нам просто не понадобятся.

Разбираем JavaScript-код. Первый шаг — выбираем все элементы `div`. Затем применяем к получившемуся набору метод `slice()`. Первый аргумент метода — это индекс первого элемента, который должен присутствовать в наборе, возвращаемом методом. Второй аргумент — тоже индекс элемента, и тоже первого. Но первого из тех, которые уже не должны попасть в набор. В примере из листинга 7.2.5 мы передаем аргументы 2 и 5.

Опять воспользуемся методом `css()` и установим для элементов окончательного набора красную рамку. Если выполнить написанный нами код, то такую рамку получат третий, четвертый и пятый элементы `div`.

ЗАДАЧА

Необходимо уменьшить существующий набор до одного элемента. Например, в наборе должен остаться только первый элемент, или последний, или вообще третий по счету.

Решение

В решении этой задачи нам помогут методы `first()`, `last()` и `eq()` (листинг 7.2.6).

Листинг 7.2.6. Использование методов `first()`, `last()` и `eq()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("li").first().css("background-color", "#f30");
  $("li").last().css("background-color", "#093");
  $("li").eq(2).css("background-color", "#ff3");
});
</script>
</head>
<body>
<ul>
  <li>Пункт 1</li>
  <li>Пункт 2</li>
  <li>Пункт 3</li>
  <li>Пункт 4</li>
  <li>Пункт 5</li>
</ul>
</body>
</html>
```

Обсуждение

Для своих опытов используем список из пяти пунктов. HTML-разметка в листинге 7.2.6 — проще не бывает. На JavaScript-код посмотрим внимательнее. Создадим набор из всех элементов `li` и применим к нему метод `first()`, оставив, таким образом, в наборе только один элемент — первый. С помощью метода `css()` установим для этого элемента зеленый цвет фона. Снова создадим такой же набор, но теперь применим к нему метод `last()`, который оставит в наборе только последний элемент. Методом `css()` устанавливаем зеленый цвет фона. В третий раз создаем тот же самый набор, но теперь применяем к нему метод `eq()`, которому передадим в качестве аргумента число 2, уменьшив набор до единственного элемента, третьего по счету (отсчет ведется от нуля), и ему установим фон желтого цвета.

ЗАДАЧА

Среди некоторых элементов необходимо отыскать только те, которые имеют элементы-потомки определенного типа.

Решение

В решении этой задачи нам поможет метод `has()` (листинг 7.2.7).

Листинг 7.2.7. Использование метода `has()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
ul { width: 400px; }
</style>
<script type="text/javascript">
$(function(){
  $("li").has("ul").css("background-color", "#ff3");
});
</script>
</head>
<body>
<ul>
  <li>Пункт 1</li>
  <li>Пункт 2
    <ul>
      <li>Пункт 2-а</li>
      <li>Пункт 2-б</li>
    </ul>
  </li>
  <li>Пункт 3</li>
  <li>Пункт 4</li>
  <li>Пункт 5
    <ul>
      <li>Пункт 5-а</li>
      <li>Пункт 5-б</li>
      <li>Пункт 5-в</li>
    </ul>
  </li>
  <li>Пункт 6</li>
</ul>
</body>
</html>
```

Обсуждение

Листинг 7.2.7 демонстрирует HTML-разметку, состоящую из списка `ul`, некоторые элементы `li` которого содержат в качестве потомков такие же списки.

Разберем JavaScript-код. Сначала выбираем все элементы `li` и к получившемуся набору применяем метод `has()`, передавая в качестве аргумента название того элемента, который хотим обнаружить внутри `li`, а именно `ul`. Для элементов `li`, удовлетворяющих этому условию, устанавливаем желтый цвет фона.

ЗАДАЧА

Необходимо получить значения какого-либо атрибута большого числа элементов, например, атрибута `id` элементов `checkbox`. Полученные значения должны быть представлены в виде массива.

Решение

Для решения задачи воспользуемся методом `map()` (листинг 7.2.8).

Листинг 7.2.8. Использование метода `map()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-2-8</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">

</style>
<script type="text/javascript">
$(function(){
    $("p").append($(":text").map(function() {
      return $(this).val();
    }).get().join(", "));
});
</script>
</head>
<body>
<p></p>
<form method="post" action="">
  <fieldset>
    <input type="text" value="один" id="one" name="number[]" />
    <input type="text" value="два" id="two" name="number[]" />
    <input type="text" value="четыре" id="four" name="number[]" />
    <input type="text" value="восемь" id="eight" name="number[]" />
  </fieldset>
</form>
</body>
</html>
```

Обсуждение

В листинге 7.2.8 описана форма с четырьмя элементами `input`. Перед формой есть параграф `p`, который послужит только для вывода результатов работы JavaScript-кода. Собственно с выбора элемента `p` мы и начинаем писать код. Отыскав его, применяем метод `append()`, а в качестве его аргумента указываем еще одно выражение.

Вот о нем-то и подробнее. Сначала выбираем все элементы `input` со значением `text` в атрибуте `type`, пользуясь соответствующим фильтром. К получившемуся набору применяем метод `map()`. Функция, которую этот метод принимает в качестве аргумента, применяется к каждому элементу набора. Это обстоятельство дает нам возможность легко получить значения необходимого атрибута каждого элемента. Метод `get()` в цепочке вызовов позволяет получить стандартный массив, поскольку метод `map()` возвращает не массив, а объект, похожий на массив. Заключительный метод в цепочке — `join()`, служит для объединения значений массива в строку, которая в итоге и будет вставлена в элемент `p`.

7.3. Прочие методы

ЗАДАЧА

Необходимо создать набор элементов, над которым осуществить некоторые операции. Затем нужно расширить этот набор дополнительными элементами, для того, чтобы следующие операции выполнялись уже над объединенным набором.

Решение

Для решения задачи используем метод `add()` (листинг 7.3.1).

Листинг 7.3.1. Использование метода `add()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
body { width:400px; }
body, div, p {
  padding:10px; margin:10px;
  border:2px solid #ccc;
}
</style>
<script type="text/javascript">
```

```
$(function(){
  $("p").css("background-color","#ddd")
        .add("div").css("border-color","#f00");
});
</script>
</head>
<body>
<div>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
</div>
<div>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
  <p>Параграф внутри элемента div</p>
</div>
</body>
</html>
```

Обсуждение

HTML-код в листинге 7.3.1 состоит из двух элементов `div`, внутри которых находится по три параграфа `p`. В начальный набор отберем элементы `p`, а расширять набор будем за счет элементов `div`.

JavaScript-код начинается с того, что мы выбираем все имеющиеся элементы `p`, к которым применяем метод `css()`, для того, чтобы установить всем параграфам серый цвет фона. Далее вызываем метод `add()`, передавая ему в качестве аргумента название элемента — `div`. После этого мы получаем объединенный набор из элементов `p` и `div`, но при этом элементы `p` уже имеют серый цвет фона. Снова применяем метод `css()` и с его помощью устанавливаем рамку красного цвета.

В итоге красную рамку получат элементы `p` и `div`, а вот серый цвет фона будет только у элементов `p`.

ЗАДАЧА

Необходимо отыскать заданный элемент в содержимом, загруженном в `iframe`.

Решение

Решаем задачу с помощью метода `contents()` (листинг 7.3.2).

Листинг 7.3.2. Использование метода `contents()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-7-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
iframe {
  border:1px dotted #00f;
}
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function(){
    $("iframe").contents().find("h2").css("color","#f00");
  });
});
</script>
</head>
<body>
<iframe src="test.htm" width="400" height="400"></iframe>
<button>Найти h2</button>
</body>
</html>
```

Обсуждение

HTML-код из листинга 7.3.2 — это элемент `iframe`, в который загружается содержимое файла `test.htm`. Кнопка **Найти h2** служит для придания наглядности — при щелчке на ней будет выполняться код, с помощью которого мы намерены решить задачу.

Займемся JavaScript-кодом. Выбираем элемент `button`, с которым необходимо связать функцию-обработчик события `click`. Самое интересное для нас заключено внутри функции. Естественно, что сначала нужно найти сам элемент `iframe`, что мы и делаем, указывая в селекторе соответствующее выражение. Применяем метод `contents()` и получаем таким образом содержимое, загруженное в элемент `iframe`. Это содержимое является обычным набором элементов, к которому мы уже привыкли, и даже знаем, что для поиска в наборе пригодится метод `find()`. Мы им и воспользуемся, передав в аргументе выражение `"h2"`. И, наконец, получив этот элемент, с помощью метода `css()`, заставим цвет шрифта тега `<h2>` сделаться красным.

Осталось только нажать кнопку **Найти h2**, чтобы убедиться, что так и произойдет.

ЗАДАЧА

После того как первоначальный набор элементов был изменен и над ним проводились какие-либо операции, необходимо вернуться к исходному набору и изменить его вновь, но используя уже другие критерии.

Решение

Для решения задачи подойдет метод `end()` (листинг 7.3.3).

Листинг 7.3.3. Использование метода `end()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:100px;
  margin:5px; float:left;
  border:3px double #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").filter(".test, #sixth")
          .css("background-color","#00f")
          .end()
          .filter(".test:first, #first")
          .css("border-color","#f00");
});
</script>
</head>
<body>
<div id="first"></div>
<div id="second" class="test"></div>
<div id="third"></div>
<div id="fourth"></div>
<div id="fifth" class="test"></div>
<div id="sixth"></div>
</body>
</html>
```

Обсуждение

Вновь будем работать с шестью элементами `div` на веб-странице, HTML-разметка которой приведена в листинге 7.3.3. Все элементы `div` имеют атрибуты `id`, а два из них еще и атрибут `class`.

С помощью JavaScript-кода, содержащегося в том же листинге 7.3.3, выбираем все элементы `div`. К получившемуся набору применяем уже знакомый метод `filter()`,

которому передаем выражение ".test, #sixth". Таким образом, мы оставляем в наборе только те элементы div, которые имеют имя класса test, а также элемент div с значением атрибута id, равным #sixth. Для этих элементов устанавливаем цвет фона — второй, пятый и шестой элементы div стали синими.

Теперь попробуем вернуться к первоначальному набору, и используем для этого в цепочке вызовов метод end(). Будем считать, что это у нас получилось, и в нашем распоряжении снова весь набор элементов div (но некоторые из них уже имеют синий фон). С помощью метода filter() снова изменяем набор, но уже с условием ".test:first, #first", оставляя в наборе первый найденный элемент с именем класса test и элемент div с идентификатором #first. Получается, что это первый и второй элементы div. Установим для них красную двойную рамку с помощью метода css().

Теперь первый элемент имеет красную рамку, второй — красную рамку и синий фон, пятый и шестой элементы — только синий фон. Ну, а третий и четвертый элементы никак не изменились.

ЗАДАЧА

В начальном наборе необходимо отыскать какие-либо элементы, после чего объединить получившийся и исходный наборы.

Решение

andSelf() — метод, который поможет решить эту задачу (листинг 7.3.4).

Листинг 7.3.4. Использование метода andSelf()

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-7-3-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div, p { margin:10px; padding:5px; }
.border { border:3px double #f00; }
.background { background-color:#ddd; }
</style>
<script type="text/javascript">
$(function(){
  $("div").find("p").andSelf().addClass("border");
  $("div").find("p").addClass("background");
});
</script>
</head>
```

```
<body>
<div>
  <p>Первый параграф</p>
  <p>Второй параграф</p>
  <p>Третий параграф</p>
</div>
</body>
</html>
```

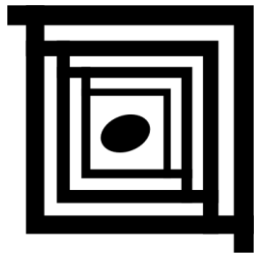
Обсуждение

В HTML-коде из листинга 7.3.4 в элемент `div` вложены три элемента `p` с некоторым текстом.

Смотрим на первую строку JavaScript-кода. Совершаем уже ставшее привычным действие — выбираем все элементы `div`, имеющиеся на веб-странице. К получившемуся набору применяем метод `find()`, а в качестве аргумента передаем выражение `"p"`, оставляя в наборе только элементы `p`.

Теперь настало время объединить набор, содержащий только элементы `p`, с начальным набором, в который, как мы помним, входили элементы `div` (вернее один элемент `div` — больше-то их и не было). В цепочку вызовов включаем метод `andSelf()` и к элементам набора, получившегося в результате этой операции, добавляем класс `border` с помощью метода `addClass()`. В итоге и элементы `p`, и элемент `div` получат двойную рамку красного цвета.

Вторая строка JavaScript-кода приведена только для сравнения. В ней почти все то же самое, только без метода `andSelf()`. В результате класс с именем `background` будет применен лишь к элементам `p`.



ГЛАВА 8

События и их обработка

8.1. События документа

ЗАДАЧА

Необходимо выполнить какую-либо функцию в тот момент, когда объектная модель документа (DOM) готова к обходу и манипуляциям с ней.

Решение

Для решения такой задачи в библиотеке jQuery имеется метод `ready(handler)` (листинг 8.1.1).

Листинг 8.1.1. Использование метода `ready(handler)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(document).ready(function () {
  $("<p>DOM готова к обходу и манипуляциям с ней!</p>").appendTo("body");
});
</script>
</head>
<body>
</body>
</html>
```

Обсуждение

Код, приведенный в листинге 8.1.1, добавит в `body` элемент `p` с поясняющим текстом в момент готовности DOM. Это хорошая замена использованию `window.onload`. Применение данного метода позволит вызывать необходимую функцию непосредственно в момент готовности объектной модели документа (DOM) к работе, т. е. пользователю не придется ожидать окончания загрузки каких-либо данных с удаленных ресурсов (баннеры, счетчики и т. п.), чтобы приступить к работе с приложением. Вот поэтому практически любой код jQuery начинается с тех нескольких строк, которые приведены в листинге 8.1.1.

Однако внимательные читатели наверняка сразу обратят внимание на тот факт, что на протяжении всей книги в листингах встречается несколько иная конструкция, с которой начинается jQuery-код (листинг 8.1.2).

Листинг 8.1.2. Использование метода `ready(handler)`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("<p>DOM готова к обходу и манипуляциям с ней!</p>").appendTo("body");
});
</script>
</head>
<body>
</body>
</html>
```

Все объясняется очень просто — код в листинге 8.1.2 представляет собой сокращенную запись выражения `$(document).ready();`. Вы вольны выбрать тот способ записи, который вам больше понравится.

На странице может быть сколь угодно много конструкций `$(function(){...});`, но при этом необходимо учесть, что выполняться код будет в порядке его следования. И еще один важный момент — необходимо убедиться, что элемент `body` не имеет обработчика события `onload`, в противном случае конструкция `$(document).ready();` или ее короткая форма записи `$(function(){...});` может и не сработать.

ЗАДАЧА

Требуется выполнить какую-либо функцию в тот момент, когда веб-страница полностью загружена. При выгрузке веб-страницы также необходимо выполнить некоторые действия.

Решение

Используем имеющиеся в библиотеке jQuery методы `load(handler(eventObject))` и `unload(handler(eventObject))` (листинг 8.1.3).

Листинг 8.1.3. Использование методов `load()` и `unload()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(window).load(function () {
  alert("Здравствуйте!");
}).unload(function () {
  alert("До свидания!");
});
</script>
</head>
<body>
<a href="example-8-1-1.html">Уйти с этой страницы</a>
</body>
</html>
```

Обсуждение

JavaScript-код, приведенный в листинге 8.1.3, связывает с объектом `window` функции-обработчики событий, передаваемые как аргументы методов `load()` и `unload()`. Теперь при входе на веб-страницу с нами поздороваются, а при попытке покинуть ее — попрощаются.

Начиная с версии 1.4.3, появилась возможность передавать методам `load()` и `unload()` необязательный аргумент — объект, содержащий какие-либо пользовательские данные. Если такой объект передается, то он должен быть первым аргументом, например: `.unload([eventData], handler(eventObject))`.

8.2. Назначение, удаление и вызов событий

ЗАДАЧА

Выбранному элементу необходимо назначить функцию-обработчик какого-либо события с передачей в эту функцию пользовательских данных. Требуется также иметь возможность в любой момент снять с этого элемента обработчик события.

Решение

Для решения такой задачи вполне подойдут методы `bind(eventType, [eventData], handler(eventObject))` и `unbind([eventType], [handler(eventObject)])` (листинг 8.2.1).

Листинг 8.2.1. Использование методов `bind()` и `unbind()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:300px; padding:10px;
  margin:10px; border:2px solid #ccc;
}
p {
  width:400px; padding:10px;
  margin:10px; border-bottom:1px dotted #369;
}
</style>
<script type="text/javascript">
$(function(){
  var data = {
    one: "Один",
    two: 2
  };
  $("div").bind("click", data, function(e){
    $("p").text("Данные: " + e.data.one + ", " + e.data.two +
                "; Событие click в координатах: " + e.pageX +
                ", " + e.pageY);
  });
  $("button").click(function(){
    $("div").unbind("click");
    $("p").empty();
  });
});
</script>
</head>
<body>
<div>Элемент DIV</div>
<p></p>
<button>unbind</button>
</body>
</html>
```

Обсуждение

Сначала разберем HTML-разметку, приведенную в листинге 8.2.1. С элементом `div` свяжем функцию-обработчик события `click`. В элемент `p` будем выводить поясняющие сообщения, а элемент `button` используем для того, чтобы при нажатии на него снять обработчик события `click` с элемента `div`.

Теперь поговорим о JavaScript-коде. Весь код заключен в конструкцию `$(function(){...});`, которая помогает отследить момент готовности DOM к работе (см. *разд. 8.1*). Как только DOM готова, мы создаем объект `data`. Это объект, содержащий пользовательские данные, которые необходимо передать функции-обработчику. Затем мы выбираем элемент `div` и применяем к получившемуся набору метод `bind()`, который поможет связать с элементами набора нужное событие. Первый аргумент метода — название того события, которое следует обрабатывать. В нашем случае — `"click"`. Второй, необязательный аргумент — объект, который может быть передан в функцию-обработчик события в качестве свойства объекта `event`. В примере из листинга 8.2.1 таким аргументом будет созданный в самом начале кода объект `data`. Наконец, третьим аргументом является сама функция-обработчик события, назначенного элементам набора.

Если предыдущий абзац показался вам не очень понятным, просто найдите соответствующее место в листинге 8.2.1 и многое сразу прояснится. Теперь давайте разберемся, какие действия выполняет функция-обработчик события `click`, назначенная нами элементу `div`.

Действие очень простое — в элемент `p` вставляется некоторый текст. Но обратите внимание, внутри нашей функции мы легко можем обратиться не только к свойствам объекта события `e`, получив значения его свойств `pageX` и `pageY`, но и к дополнительному свойству `data`, т. е. объекту, который мы передавали функции-обработчику во втором, необязательном аргументе.

Следующий шаг — выбираем элемент `button` и связываем с ним обработчик события `click`. При нажатии на кнопку будет вызвана функция, которая отыщет элемент `div` и применит к нему метод `unbind()`. Передав в аргументе `"click"`, мы снимаем обработчик соответствующего события с выбранного элемента. Следующая строка просто удаляет внутреннее содержимое элемента `p`, куда мы выводили поясняющий текст.

> ***Примечание***
>
> Возможные значения аргумента `eventType` методов `bind()` и `unbind()` — `blur`, `focus`, `focusin`, `focusout`, `load`, `resize`, `scroll`, `unload`, `click`, `dblclick`, `mousedown`, `mouseup`, `mousemove`, `mouseover`, `mouseout`, `mouseenter`, `mouseleave`, `change`, `select`, `submit`, `keydown`, `keypress`, `keyup`, `error`.

ЗАДАЧА

Выбранному элементу необходимо назначить обработчик какого-либо события, который должен отработать только один раз.

Решение

Используем для решения такой задачи метод `one(eventType, [eventData], handler(eventObject))` (листинг 8.2.2).

Листинг 8.2.2. Использование метода `one()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  float:left;
  width:150px; height:150px;
  padding:10px; margin:10px;
  border:3px double #ccc;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").one("dblclick", function(){
    alert($(this).attr("id"));
  });
});
</script>
</head>
<body>
<div id="one"></div>
<div id="two"></div>
<div id="three"></div>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 8.2.2, — три элемента `div`, каждый из которых имеет идентификатор `id`. Этим элементам попробуем назначить обработчик события `"dblclick"` так, чтобы он был вызван только один раз.

В JavaScript-коде мы отслеживаем момент готовности DOM, выбираем в набор все элементы `div` и применяем к получившемуся набору метод `one()`. В качестве первого аргумента передаем название нужного события — `"dblclick"`, второго аргумента не будет (его использование ничем не отличается от рассмотренного в листинге 8.2.1), и в третьем аргументе назначаем функцию-обработчик события.

Обработчик очень простой — при наступлении события появится окно предупреждения, где будет выведено значение атрибута `id` того элемента, в котором произошло событие.

Если вы испытаете этот пример, содержащийся на компакт-диске, прилагаемом к книге, то сможете убедиться, что обработчик события действительно вызывается только один раз.

Примечание

Возможные значения аргумента `eventType` метода `one(eventType, [eventData], handler(eventObject))` — `blur`, `focus`, `focusin`, `focusout`, `load`, `resize`, `scroll`, `unload`, `click`, `dblclick`, `mousedown`, `mouseup`, `mousemove`, `mouseover`, `mouseout`, `mouseenter`, `mouseleave`, `change`, `select`, `submit`, `keydown`, `keypress`, `keyup`, `error`.

ЗАДАЧА

Для выбранного элемента необходимо программно вызвать какое-либо событие.

Решение

Задачу решаем с использованием метода `trigger(eventType, [extraParameters])` (листинг 8.2.3).

Листинг 8.2.3. Использование метода `trigger()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-2-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  float:left;
  width:150px; height:150px;
  padding:10px; margin:10px;
  border:3px double #369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div:first").click(function(){
    $(this).css("background-color","#369");
  });
  $("div:last").mouseover(function(){
    $("div:first").trigger("click");
  });
});
```

```
</script>
</head>
<body>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

Два элемента `div` — это вся HTML-разметка для листинга 8.2.3. Первому элементу `div` назначим обработчик события `click`, а второму — обработчик события `mouseover`, где внутри обработчика попробуем вызвать событие `click` для первого элемента `div`.

Обратимся к JavaScript-коду. Сначала выбираем первый элемент `div` с помощью выражения `div:first`, указанного в селекторе, и связываем с ним обработчик события `click`, в котором методом `css()` устанавливаем цвет фона для выбранного элемента. Как вы уже догадались, теперь при щелчке мышью на первом элементе `div` он станет синего цвета.

Теперь выберем второй (он же последний) элемент `div` с помощью выражения `div:last` и назначим ему функцию-обработчик события `mouseover`. Обработчик этого события должен будет отыскать первый элемент `div` и применить к нему метод `trigger()`, передав в качестве первого аргумента значение `"click"`. Второй, необязательный аргумент передавать не будем. (Этот аргумент является массивом с дополнительными параметрами, которые можно передать в функцию-обработчик.)

Догадались, что произойдет? При наведении указателя мыши на второй элемент `div` первый моментально сделается синим, т. е. будет вызвана функция-обработчик события `click`, назначенная ему.

Необходимо отметить, что нашу задачу можно было бы решить и с помощью довольно похожего метода `triggerHandler()`. Попробуйте заменить в JavaScript-коде из листинга 8.2.3 метод `trigger()` на `triggerHandler()` — никаких изменений не произойдет. Код отработает точно так же.

Но при всей схожести этих методов они имеют и довольно существенные различия.

Во-первых, метод `triggerHandler()` в отличие от метода `trigger()` не вызывает поведение по умолчанию. В качестве примера можно привести нажатие кнопки `submit` в форме. При использовании метода `triggerHandler()` отправки формы не произойдет.

Во-вторых, метод `triggerHandler()` вызовет назначенное событие только для первого элемента, попавшего в набор, тогда как `trigger()` отработает для всех элементов набора.

В-третьих, события, вызванные с применением `triggerHandler()`, не всплывают по иерархии DOM.

В-четвертых, в отличие от метода `trigger()`, который возвращает объект jQuery (чтобы обеспечить возможность построения цепочек вызовов), метод `triggerHandler()` возвращает значение, которое вернул последний вызванный обработчик.

ЗАДАЧА

Некоторым элементам должен быть назначен обработчик события. Точно такие же элементы могут быть добавлены в DOM позже. Для них также должен действовать обработчик события. Необходимо иметь возможность удалить обработчик события со всех элементов.

Решение

Для решения задачи используем методы `live(eventType, [eventData], handler)` и `die(eventType, [handler])` (листинг 8.2.4).

Листинг 8.2.4. Использование методов `live()` и `die()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-2-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  float:left;
  width:100px; height:100px;
  padding:10px; margin:10px;
  background-color:#369;
}
</style>
<script type="text/javascript">
$(function(){
  $("button:first").click(function(){
    $("<div></div>").insertAfter("div:last");
  });
  /*
  $("div").click(function(){
    $(this).animate({ opacity:0.2 },100)
           .animate({ opacity:1 },1000);
  });
  */
  $("div").live("click",function(){
```

```
    $(this).animate({ opacity:0.2 },100)
           .animate({ opacity:1 },1000);
  });
  $("button:last").click(function(){
    $("div").die("click");
  });
});
</script>
</head>
<body>
<button>Добавить DIV</button>
<button>Отменить live</button><br />
<div></div>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

HTML-разметка, приведенная в листинге 8.2.4, описывает три элемента `div`, с которыми мы будем связывать обработчик события `click`, и две кнопки. С помощью первой кнопки мы будем добавлять в DOM точно такие же элементы `div`, которые присутствуют изначально. С помощью второй кнопки — удалять установленные обработчики события.

Давайте внимательно рассмотрим JavaScript-код. Сначала свяжем обработчик события `click` с первой кнопкой. При щелчке мышью на ней функция-обработчик должна будет добавить в DOM элемент `div` с помощью метода `insertAfter()`.

Посмотрите теперь на закомментированную часть JavaScript-кода. Что произошло бы, если мы решали поставленную задачу обычным способом? На первый взгляд — все логично. Назначили обработчик события `click` всем элементам `div`. При щелчке мышью на любом из элементов вызывается функция-обработчик, которая последовательно применяет к этому элементу два вызова метода `animate(params,[duration])`, с помощью которых прозрачность элемента `div` сначала изменяется до значения 0,2 в течение 100 мс, а затем в течение 1000 мс возвращается в значение 1. Таким образом, получается, что элемент "мигнул". Однако если бы мы теперь попробовали добавить в DOM еще один или несколько элементов `div`, то смогли бы убедиться, что для них обработчики события не действуют! Но почему?

Все просто — ведь обработчик события назначался набору элементов `div`, и в этот набор попали только элементы, присутствовавшие в DOM с самого начала. Добавляя в DOM дополнительные элементы `div`, мы не включали их в набор, связанный с обработчиком события, поэтому для новых элементов обработчик вызываться не будет.

Примечание

Это справедливо для элементов, добавленных в DOM как с помощью методов `after()`, `before()`, `insertAfter()`, `insertBefore()`, `replaceWith()`, `replaceAll()`, так и с помощью AJAX-запросов.

Для того чтобы решить эту проблему, применяется метод `live()` — первым аргументом мы передадим методу название необходимого события `"click"`, а вторым аргументом выступает функция-обработчик. Если теперь попробовать выполнить код из листинга 8.2.4, то можно убедиться, что обработчик события `click` действует для всех элементов `div` — как существовавших в DOM с самого начала, так и добавленных позднее.

Осталось только познакомиться с методом `die()`. Этот метод удаляет обработчики событий, назначенные с помощью метода `live()`. В примере из листинга 8.2.4 мы удаляем обработчик события `click` с элементов `div`.

Примечание

Возможные значения аргумента `eventType` методов `live(eventType, [eventData], handler)` и `die(eventType, [handler])` — `click`, `dblclick`, `mousedown`, `mouseup`, `mousemove`, `mouseover`, `mouseout`, `keydown`, `keypress`, `keyup`. Начиная с версии 1.4.1 поддерживаются `blur`, `focus`.

Следует отметить также, что начиная с версии 1.4.3 существует возможность передать методу `live()` в качестве единственного аргумента объект, свойствами которого являются названия типов событий JavaScript, а значениями этих свойств — функции-обработчики.

ЗАДАЧА

Некоторым элементам должен быть назначен обработчик события. Точно такие же элементы могут быть добавлены в DOM позже. Для них также должен действовать обработчик события. Необходимо иметь возможность удалить обработчик события со всех элементов.

Задача формулируется абсолютно так же, как и в предыдущем примере, но решать ее мы будем другими методами.

Решение

Для решения задачи используем методы `delegate(eventType, [eventData], handler)` и `undelegate(eventType, [handler])` (листинг 8.2.5), добавленные в библиотеку, начиная с версии 1.4.2.

Листинг 8.2.5. Использование методов `delegate()` и `undelegate()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-8-2-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
th, td { text-align:center; }
td { border-bottom:1px dotted #ccc; }
.hover { background-color:#333; color:#fff; }
</style>
<script type="text/javascript">
$(function(){
  $("button:first").click(function(){
    $("tbody").append("<tr><td>доб-1</td><td>доб-2</td><td>доб-3</td></tr>");
  });
  $("button:last").click(function(){
    $("table").undelegate("td", "hover");
  });
  $("table").delegate("td", "hover", function(){
    $(this).toggleClass("hover");
  });
});
</script>
</head>
<body>
<table>
  <thead>
    <tr><th>Колонка 1</th><th>Колонка 2</th><th>Колонка 3</th></tr>
  </thead>
  <tbody>
    <tr><td>1-1</td><td>1-2</td><td>1-3</td></tr>
    <tr><td>2-1</td><td>2-2</td><td>2-3</td></tr>
    <tr><td>3-1</td><td>3-2</td><td>3-3</td></tr>
    <tr><td>4-1</td><td>4-2</td><td>4-3</td></tr>
    <tr><td>5-1</td><td>5-2</td><td>5-3</td></tr>
  </tbody>
</table>
<button>Добавить строку</button>
<button>undelegate()</button>
</body>
</html>
```

Обсуждение

HTML-разметка, приведенная в листинге 8.2.5, — обыкновенная таблица и две кнопки. Если рассматривать JavaScript-код, то при нажатии на первую кнопку мы будем добавлять в таблицу еще одну строку с помощью метода `append()`, а при нажатии на вторую кнопку — вызовем метод `undelegate()`, чтобы удалить назначенный обработчик.

Дальше, несколько подробнее. Мы хотим сделать так, чтобы при наведении указателя мыши на любую ячейку `td` таблицы изменялся цвет ее фона и цвет шрифта. Для этого мы приготовили оформление в виде стилевого класса `hover`. Такого поведения мы могли бы добиться, выбрав все элементы `td` и применив к получившемуся набору метод `toggleClass()`. Но нам необходимо, чтобы таким же образом себя вели ячейки, которые были добавлены в таблицу с помощью `append()`. Значит, воспользуемся методом `delegate()`.

Но посмотрите, к какому набору мы его применяем — не к ячейкам `td`, а к элементу `table`. А вот в первом аргументе метода `delegate()` действительно уже указываем название нужного элемента — `td`, во втором — название события, и в третьем — функцию-обработчик.

Разница в коде по сравнению с примером из листинга 8.2.4 вполне понятна, но какова эта разница по своей сути?

Для примера не случайно была выбрана таблица. Если бы мы попытались использовать метод `live()`, то вынуждены были бы сначала создать набор из всех элементов `td`, затем с помощью метода `each()` обойти получившийся набор и назначить обработчик для каждого элемента `td`. Представьте, если ячеек в таблице несколько тысяч. А метод `delegate()` позволит назначить только один обработчик на родительском элементе.

8.3. События мыши, клавиатуры, браузера и форм

ЗАДАЧА

При наведении указателя мыши на какой-либо элемент требуется изменять визуальное представление элемента. Необходимо, чтобы при выходе указателя мыши за пределы элемента визуальное представление элемента возвращалось в начальное состояние.

Решение

Задачу можно решить с помощью пары методов `mouseover` – `mouseout`, а можно и по-другому (листинг 8.3.1), применив метод `hover(handlerIn(eventObject), handlerOut(eventObject))`.

Листинг 8.3.1. Использование метода `hover()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  float:left;
  width:100px; height:100px;
  padding:10px; margin:10px;
  border:3px double #369;
}
.hover {
  border:3px double #f00;
  background-color:#369;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").hover(
    function(){ $(this).addClass("hover"); },
    function(){ $(this).removeClass("hover"); }
  );
});
</script>
</head>
<body>
<div></div>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

В листинге 8.3.1 очень простая HTML-разметка — три элемента `div`, визуальное отображение которых задано с помощью CSS-свойств. Посредством CSS описан и класс `hover` — синий фон и двойная рамка красного цвета.

Рассмотрим JavaScript-код. Выбираем в набор элементы `div` и применяем метод `hover()`. Оба аргумента, принимаемых методом, — функции. Первая функция выполняется в момент вхождения указателя мыши в пределы выбранного элемента, вторая — при выходе указателя мыши за его пределы. Учитывая, что первая функция добавляет класс с именем `hover` элементу, над которым находится указатель мыши, а вторая удаляет этот же класс, мы видим, как при наведении мыши изменяется оформление элемента `div`: он получает двойную красную рамку и синий цвет фона.

ЗАДАЧА

Для выбранных элементов необходимо организовать переключение состояний по щелчку мышью на элементе. Причем состояний может быть больше двух.

Решение

Задачу будем решать с помощью имеющегося в арсенале библиотеки jQuery метода `toggle(handler(eventObject), handler(eventObject), [handler(eventObject)])` (листинг 8.3.2).

Листинг 8.3.2. Использование метода `toggle()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:100px;
  padding:10px; margin:10px;
  border:3px double #369;
  background-color:#fff;
}
</style>
<script type="text/javascript">
$(function(){
  $("div").toggle(
    function(){ $(this).css("background-color","#f00"); },
    function(){ $(this).css("background-color","#0f0"); },
    function(){ $(this).css("background-color","#00f"); },
    function(){ $(this).css("background-color","#fff"); }
  );
});
</script>
</head>
<body>
<div></div>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

В HTML-разметке, приведенной в листинге 8.3.2, описаны три элемента `div`, которым с помощью CSS-правил задано оформление (размеры, рамка и белый цвет фона). Попробуем изменять цвет фона элементов `div` в последовательности: белый, красный, зеленый, синий и снова белый.

Посмотрим, как это реализовано с помощью нехитрого JavaScript-кода. Сначала необходимо выбрать элементы `div`, а затем применить к этому набору метод `toggle()`. Аргументами метода являются простые и очень похожие функции, которые будут вызываться друг за другом при каждом следующем щелчке мышью на элементе и устанавливать соответствующий цвет фона с помощью метода `css()`.

ЗАДАЧА

Имеется HTML-код, описывающий два элемента `input`, элементы `select` и `button`. При получении и потере фокуса элементами `input` должны вызываться функции-обработчики этих событий. Для элемента `select` функция-обработчик должна вызываться при изменении выбранной опции. По щелчку мышью на элементе `button` первый элемент `input` должен получить фокус. При загрузке страницы фокус должен получить последний элемент `input`.

Решение

Для решения такой задачи понадобятся методы `focus(handler(eventObject))`, `blur(handler(eventObject))`, `change(handler(eventObject))` и `click(handler(eventObject))` (листинг 8.3.3).

Листинг 8.3.3. Использование методов `focus()`, `blur()`, `change()` и `click()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-8-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
input,select { width:120px; margin:1px; }
span { padding:4px; }
.focus { color:#f00; }
.blur { color:#00f; }
.change { color:#0f0; }
</style>
<script type="text/javascript">
$(function(){
  $("input").focus(function(){
    $("<span class='focus'>focus!</span>").insertAfter(this).animate({
opacity:0 },1500);
  });
  $("input").blur(function(){
    $("<span class='blur'>blur!</span>").insertAfter(this).animate({ opacity:0
},1000);
  });
```

```
  $("select").change(function() {
    var t = "name=" + $(this).attr('name') + "&value=" + $(this).val();
    $(this).after("<span class='change'>" + t +
"</span>").next("span").animate({ opacity:0 },2500);
  });
  $("button").click(function(){
    $("input:first").focus();
  });
  $("input:last").focus();
});
</script>
</head>
<body>
<input type="text" name="One" /><br />
<input type="text" name="Two" /><br />
<select name="Three">
  <option value="1">Один</option>
  <option value="2">Два</option>
  <option value="3">Три</option>
  <option value="4">Четыре</option>
  <option value="5">Пять</option>
</select><br />
<button>click()</button>
</body>
</html>
```

Обсуждение

Итак, HTML-разметка, приведенная в листинге 8.3.3, была задана в условии задачи — два элемента `input`, элементы `select` и `button`.

Рассмотрим JavaScript-код, с помощью которого будем решать поставленную задачу. Сначала назначим функцию-обработчик события `focus` для элементов `input`, для чего выберем их в набор и применим метод `focus()`. Сама функция-обработчик добавит элемент `span` с поясняющим текстом после того элемента `input`, в котором произошло событие. А метод `animate(params,[duration])` поможет плавно скрыть поясняющую надпись. Проделаем точно такие же операции для набора элементов `input`, но уже с использованием метода `blur()`, чтобы обрабатывать событие потери фокуса элементами `input`.

На следующем шаге выбираем элемент `select` и с помощью метода `change()` назначаем функцию-обработчик события, которое будет происходить при смене выбранной опции. Внутри функции получим значение атрибута `name` элемента `select` и значение атрибута `value` для выбранной опции. После чего выведем эту информацию подобно тому, как мы делали это для элементов `input`.

Остался элемент `button`, при щелчке мышью на котором первый элемент `input` должен получить фокус. Следовательно, с элементом `button` необходимо связать

обработчик события `click`, при наступлении которого вызывать нужную функцию. Эта функция выбирает первый элемент `input` с помощью выражения `input:first`, указанного в селекторе, и применяет к нему метод `focus()`. Обратите внимание, что в этом случае мы не передаем методу никаких аргументов и таким образом можем вызвать необходимое нам событие в выбранном элементе.

Осталось немного — найти последний элемент `input` и сделать так, чтобы при загрузке страницы именно он получал фокус. Это очень просто — в селекторе указываем `input:last` и к этому набору применяем метод `focus()` без передачи ему каких-либо аргументов.

В примере из листинга 8.3.3 мы познакомились с некоторыми методами, помогающими управлять обработкой событий или их вызовом.

Нерассмотренные в примерах методы используются подобным образом, а их полный перечень приведен в табл. 8.1.

Таблица 8.1. *Обработка и вызов событий в jQuery*

Событие	**Описание**
`ready(handler)`	Метод `ready(handler)` позволит вызвать функцию-обработчик непосредственно в момент готовности объектной модели документа (DOM) к работе
`click()`, `click(handler(eventObject))`	Метод `click()` вызывает событие `click` для каждого выбранного элемента. Метод `click(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `click` для каждого выбранного элемента. Событие `click` возникает, когда пользователь совершает щелчок мышью над элементом. По-другому это можно определить как нажатие и отпускание кнопки мыши в то время, когда ее указатель находится над некоторым элементом
`focus()`, `focus(handler(eventObject))`	Метод `focus()` вызывает событие `focus` для каждого выбранного элемента. Метод `focus(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `focus` для каждого выбранного элемента. Событие `focus` возникает, когда элемент принимает фокус через указатель мыши или через нажатие клавиши <Tab>
`blur()`, `blur(handler(eventObject))`	Метод `blur()` вызывает событие `blur` для каждого выбранного элемента. Метод `blur(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `blur` для каждого выбранного элемента. Событие `blur` возникает, когда элемент теряет фокус через указатель мыши или через нажатие клавиши <Tab>
`focusin(handler(eventObject))`	Метод `focusin(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `focus` для каждого выбранного элемента или элемента внутри него

Таблица 8.1 *(продолжение)*

Событие	Описание
`focusout(handler(eventObject))`	Метод `focusout(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `blur` для каждого выбранного элемента или элемента внутри него
`dblclick()`, `dblclick(handler(eventObject))`	Метод `dblclick()` вызывает событие `dblclick` для каждого выбранного элемента. Метод `dblclick(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `dblclick` для каждого выбранного элемента. Событие `dblclick` возникает, когда пользователь совершает двойной щелчок мышью над элементом
`change()`, `change(handler(eventObject))`	Метод `change()` вызывает событие `change` для каждого выбранного элемента. Метод `change(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `change` для каждого выбранного элемента. Событие `change` возникает, когда элемент теряет фокус, и его значение было изменено по сравнению с моментом получения фокуса
`select()`, `select(handler(eventObject))`	Метод `select()` вызывает событие `select` для каждого выбранного элемента. Метод `select(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `select` для каждого выбранного элемента. Событие `select` возникает, когда пользователь выделяет какой-либо текст в текстовом поле элементов `input` или `textarea`. Не следует путать событие `select` с событием `change`, которое возникает, когда пользователь изменяет выбранную опцию в HTML-элементе `select`
`keydown()`, `keydown(handler(eventObject))`	Метод `keydown()` вызывает событие `keydown` для каждого выбранного элемента. Метод `keydown(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `keydown` для каждого выбранного элемента. Событие `keydown` возникает, когда на клавиатуре нажата какая-либо клавиша
`keypress()`, `keypress(handler(eventObject))`	Метод `keypress()` вызывает событие `keypress` для каждого выбранного элемента. Метод `keypress(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `keypress` для каждого выбранного элемента. Событие `keypress` возникает, когда на клавиатуре нажата и затем отпущена какая-либо клавиша. Точнее это можно определить как "клик" по клавише или как последовательность событий `keydown` — `keypress` — `keyup`
`keyup()`, `keyup(handler(eventObject))`	Метод `keyup()` вызывает событие `keyup` для каждого выбранного элемента. Метод `keyup(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `keyup` для каждого выбранного элемента. Событие `keyup` возникает, когда на клавиатуре отпущена какая-либо клавиша

Таблица 8.1 *(продолжение)*

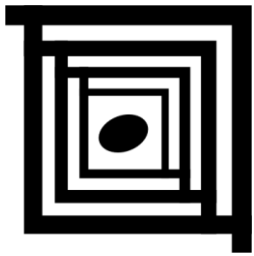
Событие	Описание
`error()`, `error(handler(eventObject))`	Метод `error()` вызывает событие `error` для каждого выбранного элемента. Метод `error(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `error` для каждого выбранного элемента. Событие `error` возникает, когда в каком-либо элементе происходит ошибка. Пример: загрузка отсутствующего изображения
`submit()`, `submit(handler(eventObject))`	Метод `submit()` вызывает событие `submit` для каждого выбранного элемента. Метод `submit(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `submit` для каждого выбранного элемента. Событие `submit` возникает, когда происходит отправка данных из формы. Иными словами, либо совершен "клик" мышью на элементе `input` с типом `submit`, либо нажата клавиша <Enter>, когда этот элемент имеет фокус
`mouseover(handler(eventObject))`	Метод `mouseover(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mouseover` для каждого выбранного элемента. Событие `mouseover` возникает, когда указатель мыши перемещен в пределы элемента
`mouseout(handler(eventObject))`	Метод `mouseout(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mouseout` для каждого выбранного элемента. Событие `mouseout` возникает, когда указатель мыши перемещен за пределы элемента
`mousedown(handler(eventObject))`	Метод `mousedown(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mousedown` для каждого выбранного элемента. Событие `mousedown` возникает, когда кнопка мыши нажата над элементом
`mouseup(handler(eventObject))`	Метод `mouseup(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mouseup` для каждого выбранного элемента. Событие `mouseup` возникает, когда кнопка мыши отпущена над элементом
`mouseenter(handler(eventObject))`	Метод `mouseenter(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mouseenter` для каждого выбранного элемента. Событие `mouseenter` возникает однократно, когда указатель мыши перемещен в пределы элемента. Удобно при использовании вложенных элементов
`mouseleave(handler(eventObject))`	Метод `mouseleave(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mouseleave` для каждого выбранного элемента. Событие `mouseleave` возникает однократно, когда указатель мыши перемещен за пределы элемента. Удобно при использовании вложенных элементов

Таблица 8.1 *(окончание)*

Событие	Описание
`mousemove(handler(eventObject))`	Метод `mousemove(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `mousemove` для каждого выбранного элемента. Событие `mousemove` возникает постоянно, пока указатель мыши находится в процессе перемещения
`hover(handlerIn(eventObject), handlerOut(eventObject))`	Метод `hover(handlerIn(eventObject), handlerOut(eventObject))` связывает две функции с событиями `mouseover` (для функции, переданной в первом аргументе) и `mouseout` (для функции, переданной во втором аргументе) для каждого выбранного элемента
`toggle (handler(eventObject), handler( eventObject ), [handler(eventObject)])`	Метод `toggle(handler(eventObject), handler(eventObject), [handler(eventObject)])` связывает n функций с событием `click` для каждого выбранного элемента. Для первого события вызывается функция, переданная в первом аргументе, для второго — переданная во втором аргументе, и т. д.
`resize(handler(eventObject))`	Метод `resize(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `resize` для каждого выбранного элемента. Событие `resize` возникает, когда область просмотра документа изменяет свои размеры
`scroll(handler(eventObject))`	Метод `scroll(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `scroll` для каждого выбранного элемента. Событие `scroll` возникает, когда прокручивается область просмотра документа
`load(handler(eventObject))`	Метод `load(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `load` для каждого выбранного элемента. Событие `load` возникает, когда загружены все компоненты элемента. Помните, что `load(handler(eventObject))` будет работать только тогда, когда `handler(eventObject)` установлена до того, как был загружен связанный с ней элемент
`unload(handler(eventObject))`	Метод `unload(handler(eventObject))` связывает функцию `handler(eventObject)` с событием `unload` для каждого выбранного элемента. Событие `unload` возникает, когда текущая страница выгружается из браузера

Важное замечание

Необходимо четко понимать, что вызов метода без аргумента не приведет к появлению этого события и не повлечет его распространения по объектной модели документа, а просто вызовет обработчики этого события.



ГЛАВА 9

Взаимодействие jQuery и AJAX

Для того чтобы выполнить большинство примеров этой главы, которые вы сможете найти на компакт-диске, прилагаемом к книге, понадобится веб-сервер.

9.1. Сокращенные методы

Речь пойдет о методе `load()`, вспомогательных функциях `$.get()`, `$.getJSON()`, `$.getScript()` и `$.post()`, которые не дают возможность гибко изменять настройки запроса так, как это позволяет делать вспомогательная функция `$.ajax()`. Поэтому они названы сокращенными методами.

ЗАДАЧА

В выбранный элемент необходимо с помощью AJAX-запроса загрузить веб-страницу целиком.

Решение

Для решения такой простой задачи подойдет метод `load(url, [data], [complete(responseText, textStatus, XMLHttpRequest)])` (листинг 9.1.1).

Листинг 9.1.1. Использование метода `load()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
```

```
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
</style>
<script type="text/javascript">
$(function () {
  $("button:first").click(function(){
    $("#target").load("testLoad.htm");
  });
  $("button:last").click(function(){
    $("#target").empty();
  });
});
</script>
</head>
<body>
<div id="target"></div>
<button>Загрузить</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

В HTML-коде из листинга 9.1.1 `div` с идентификатором `#target` будет служить тем элементом, куда необходимо загрузить веб-страницу целиком. Следующие за ним кнопки предназначены для выполнения тех действий, о которых сообщает надпись на них. Кнопка **Загрузить** загрузит указанную веб-страницу в элемент `div`. Кнопка **Очистить** просто очистит содержимое элемента `div`.

Перейдем к рассмотрению JavaScript-кода. Начнем с того, что найдем кнопку **Загрузить** с помощью выражения `button:first`, указанного в селекторе, и свяжем с ней обработчик события `click`. Внутри функции-обработчика выбираем элемент `div` по его идентификатору и применяем метод `load()`, передавая ему только один обязательный аргумент `url`. Нетрудно догадаться, что в аргументе передается адрес веб-страницы, которую нужно загрузить.

Затем находим кнопку **Очистить** и с ней также связываем обработчик события `click`. Эта функция-обработчик выбирает элемент `div` по идентификатору и применяет метод `empty()`, удаляя все внутреннее содержимое элемента `div`.

> ***Примечание***
>
> Исходный код файла `testLoad.htm` можно найти на компакт-диске, прилагаемом к книге.

ЗАДАЧА

В выбранный элемент необходимо с помощью AJAX-запроса загрузить только некоторые элементы указанной веб-страницы. После окончания загрузки должно появиться соответствующее сообщение.

Решение

Для решения задачи снова используем метод `load(url, [data], [complete(responseText, textStatus, XMLHttpRequest)])` (листинг 9.1.2).

Листинг 9.1.2. Использование метода `load()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
</style>
<script type="text/javascript">
$(function () {
  $("button:first").click(function(){
    $("#target").load("testLoad.htm h2,p:last", function(){
      alert("Готово!");
    });
  });
  $("button:last").click(function(){
    $("#target").empty();
  });
});
</script>
</head>
<body>
<div id="target"></div>
<button>Загрузить</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

Целевым элементом в HTML-коде из листинга 9.1.2 снова будет выступать элемент `div` с идентификатором `#target`. Кнопки **Загрузить** и **Очистить** также будут выполнять те операции, названия которых на них написаны.

Немного изменился JavaScript-код, поскольку теперь необходимо загрузить не всю веб-страницу, а только выбранные элементы этой страницы. Пусть такими элементами будут элемент `h2` и последний элемент `p`.

Обратите внимание на функцию-обработчик события `click` на кнопке **Загрузить**. Мы точно так же выбираем элемент `div` по идентификатору `#target` и применяем метод `load()`, но передаем уже два аргумента.

В аргументе `url` указываем адрес веб-страницы, а через пробел селекторы, по которым будут выбраны необходимые элементы (один или несколько). Мы указываем два — `h2` и `p:last`.

Кроме этого используем один из необязательных аргументов, в котором определяем callback-функцию. Она будет вызвана после окончания загрузки и покажет окно предупреждения с сообщением "Готово!".

> ***Примечание***
>
> Исходный код файла `testLoad.htm` можно найти на компакт-диске, прилагаемом к книге.

ЗАДАЧА

В выбранный элемент необходимо с помощью AJAX-запроса загрузить ответ сервера, сформированный на основании отправленных ему данных. При получении ответа от сервера нужно вывести сообщение в окно предупреждения.

Решение

И снова решаем задачу с помощью метода `load(url, [data], [complete(responseText, textStatus, XMLHttpRequest)])` (листинг 9.1.3).

Листинг 9.1.3. Использование метода `load()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
```

```
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
</style>
<script type="text/javascript">
$(function () {
  $("button:first").click(function(){
    $("#target").load("testLoad.php",
        { name: "John", age: "35" },
        function(responseText, textStatus, XMLHttpRequest){
          alert('textStatus = ' + textStatus +
          '\nСвойство readyState объекта XMLHttpRequest: ' +
XMLHttpRequest.readyState +
          '\nСвойство status объекта XMLHttpRequest: ' +
XMLHttpRequest.status);
        });
  });
  $("button:last").click(function(){
    $("#target").empty();
  });
});
</script>
</head>
<body>
<div id="target"></div>
<button>Загрузить</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

В листинге 9.1.3 приведена знакомая HTML-разметка: элемент `div` с идентификатором `#target` и две кнопки — **Загрузить** и **Очистить**.

При щелчке мышью на кнопке **Загрузить** выбираем элемент `div` по его идентификатору и применяем к нему метод `load()`, которому первым аргументом передается URL нужной веб-страницы, вторым — объект, состоящий из пар ключ/значение, представляющие данные, отправляемые на сервер, третьим — функция, вызываемая при получении ответа от сервера.

AJAX-запрос будет отправлен к файлу `testLoad.php`, получающему эти данные в виде элементов глобального массива `$_POST['name']` и `$_POST['age']`. После обработки этих данных сервером он должен вернуть результат, который будет вставлен в элемент `div` с идентификатором `#target`.

А в окне предупреждения, как и требовалось, будет выведено сообщение, содержащее некоторую информацию, которую можно получить из аргументов, переда-

ваемых в функцию. В частности, мы получили статус ответа сервера в виде текста, а обратившись к свойствам `readyState` и `status` объекта `XMLHttpRequest`, получили их значения.

Исходный код файла `testLoad.php` приведен в листинге 9.1.3а.

Листинг 9.1.3а. Исходный код файла `testLoad.php`

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
  if($_POST) {
    print 'Имя: ' . $_POST['name'] . ', возраст: ' . $_POST['age'] . ', время:
' . date('H:i:s', time());
  }
}
?>
```

ЗАДАЧА

Используя GET-запрос, необходимо отправить на сервер некоторые данные. Ответ, полученный от сервера, должен быть вставлен в определенный элемент веб-страницы.

Решение

Попробуем решить задачу, используя вспомогательную функцию `$.get(url, [data], [success(data, textStatus, jqXHR)], [dataType])` (листинг 9.1.4).

Листинг 9.1.4. Использование вспомогательной функции `$.get()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
</style>
<script type="text/javascript">
```

```
$(function () {
  $("button:first").click(function(){
    $.get("testGet.php",
          { name: "John", age: "35" },
            function(data){
              $("div").text(data);
          });
  });
  $("button:last").click(function(){
    $("div").empty();
  });
});
</script>
</head>
<body>
<div></div>
<button>Загрузить</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

В листинге 9.1.4 видим уже привычную разметку — элемент `div` и два элемента `button`. Интересующие нас действия происходят при щелчке мышью на кнопке **Загрузить**. Здесь мы вызываем вспомогательную функцию `$.get()`, которой передаем три аргумента. Первый аргумент — адрес файла, куда нужно отправить GET-запрос. Второй аргумент — объект, представляющий данные, отправляемые на сервер. На сервере они будут доступны как элементы глобального массива `GET['name']` и `GET['age']`. И третий аргумент — функция, вызываемая при получении успешного ответа сервера. Она выберет элемент `div` и с помощью метода `text()` вставит в него ответ сервера в виде текста.

В этой функции мы использовали только один из трех возможных аргументов — `data`, который содержит ответ сервера (в рассмотренном случае — в виде обычного текста). Но мы могли задействовать также аргумент `textStatus`, содержащий статус ответа сервера в виде текста, и аргумент `jqXHR` (до версии 1.5 объект `XMLHttpRequest`), представляющий собой расширенный объект `XMLHttpRequest`.

Исходный код файла `testGet.php` приведен в листинге 9.1.4а.

Листинг 9.1.4а. Исходный код файла `testGet.php`

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
```

```
  if($_GET) {
    print 'Имя: ' . $_GET['name'] . ', возраст: ' . $_GET['age'] . ', время: '
. date('H:i:s', time());
  }
}
?>
```

ЗАДАЧА

Необходимо реализовать возможность с помощью GET-запроса загружать и выполнять JavaScript-файлы, в том числе расположенные на другом домене.

Решение

Для решения задачи применим вспомогательную функцию `$.getScript(url, [success(data, textStatus)])` (листинг 9.1.5).

Листинг 9.1.5. Использование вспомогательной функции `$.getScript()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
</style>
<script type="text/javascript">
$(function () {
  $("button:eq(0)").click(function(){
    $.getScript("testGetScript.js");
  });
  $("button:eq(1)").click(function(){
$.getScript("http://dev.jquery.com/view/trunk/plugins/color/jquery.color.js",
    function(){
      $("div").animate({ backgroundColor: "#f00" }, 1500)
              .animate({ backgroundColor: "#00f" }, 1500)
              .animate({ backgroundColor: "#fff" }, 1500);
    });
  });
});
```

```
</script>
</head>
<body>
<div></div>
<button>Загрузить и выполнить локальный скрипт</button>
<button>Загрузить и выполнить скрипт с другого домена</button>
</body>
</html>
```

Обсуждение

Два элемента `button` и элемент `div` — это вся HTML-разметка, приведенная в листинге 9.1.5. При нажатии на кнопки мы будем загружать и выполнять JavaScript-файлы.

Итак, сначала разберем функцию, вызываемую при щелчке на первой кнопке. Здесь мы используем вспомогательную функцию `$.getScript()`, которой передаем только один аргумент — адрес загружаемого файла.

Исходный код файла `testGetScript.js` приведен в листинге 9.1.5а.

Листинг 9.1.5а. Исходный код файла `testGetScript.js`

```
alert("Этот небольшой javascript-файл загружен и выполнен!");
```

В результате, при нажатии на кнопку **Загрузить и выполнить локальный скрипт** мы получим окно предупреждения с поясняющим текстом, т. е. на самом деле загруженный JavaScript-файл будет выполнен.

При щелчке мышью на следующей кнопке действуем похожим образом, только в аргументе `url` указываем абсолютный адрес JavaScript-файла `jquery.color.js`, расположенного на домене **dev.jquery.com**. Это файл плагина jQuery Color Animation, и как только он будет загружен, мы сможем использовать его функциональность в своем приложении.

Во втором аргументе можно определить функцию, вызываемую по окончании загрузки данных. Мы воспользуемся этим и передадим здесь функцию, которая найдет элемент `div` и будет плавно, в течение полутора секунд, изменять цвет фона элемента `div` с белого на красный, затем на синий и вновь на белый.

ЗАДАЧА

Требуется загрузить данные в формате JSON с помощью GET-запроса, поскольку заранее известно, что сервер вернет данные в этом формате. Необходимо также реализовать загрузку данных в формате JSON с другого домена (подразумевается, что сервер предоставляет такую возможность).

Решение

Для решения задачи подходит вспомогательная функция `$.getJSON(url, [data], [success(data, textStatus, jqXHR)])` (листинг 9.1.6).

Листинг 9.1.6. Использование вспомогательной функции `$.getJSON()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
</style>
<script type="text/javascript">
$(function () {
  $("button:eq(0)").click(function(){
    $.getJSON("testGetJSON.php", function(data, textStatus, jqXHR){
      var items = [];
      $.each(data, function(i,item){
       items.push(item.name + ': ' + item.phone);
      });
      $("div").html(items.join('<br />'));
    });
  });
  $("button:eq(1)").click(function(){

       $.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?tags=n
ature&tagmode=any&format=json&jsoncallback=?",
    function(data){
      $.each(data.items, function(i,item){
        $("<img/>").attr("src", item.media.m)
                   .attr("title",item.title)
                   .appendTo("div");
        if (i == 2) return false;
      });
    });
  });
  $("button:last").click(function(){
    $("div").empty();
  });
});
</script>
</head>
```

```
<body>
<div></div>
<button>getJSON</button>
<button>getJSONP</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 9.1.6, — элемент `div` и три кнопки, при нажатии на которые будем отправлять запросы для получения необходимых данных и очищать содержимое элемента `div`.

Рассмотрим JavaScript-код, обрабатывающий щелчок мышью на первой кнопке. Здесь мы вызываем вспомогательную функцию `$.getJSON()`. Ее первый аргумент — относительный адрес файла, который должен вернуть необходимые нам данные, а второй — функция, вызываемая при получении ответа сервера. Эта функция послужит для обработки полученных данных. Пример данных в формате JSON, возвращаемых расположенным на сервере файлом `testGetJSON.php`, приведен в листинге 9.1.6а.

Листинг 9.1.6а. Исходный код файла `testGetJSON.php`

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
  print '[{"name":"John","phone":"555-66-77"},
          {"name":"Mary","phone":"566-77-77"},
          {"name":"Helen","phone":"577-88-77"},
          {"name":"Bob","phone":"588-66-99"}]';
}
?>
```

Функция, которая будет обрабатывать ответ сервера, в первом аргументе принимает данные в формате `JSON`. Следующие два аргумента — статус ответа сервера в виде текста и специальный объект `jqXHR`.

В функции-обработчике сначала объявим массив `items`, а дальше вызовем вспомогательную функцию `$.each(collection, callback(indexInArray, valueOfElement)` для обхода и обработки полученных данных. Каждый обработанный элемент вставляем в массив `items`, который затем объединим в строку и вставим в элемент `div`. В итоге мы увидим следующий результат — "John: 555-66-77" и так далее.

Продолжим решать задачу и разберем код функции, которая выполняется при щелчке мышью на второй кнопке. Здесь используется та же самая вспомогательная функция `$.getJSON()`, только в первом аргументе указан абсолютный адрес вебстраницы на сервере flickr.com, который предоставляет API для получения нужных данных в формате JSON. Во втором аргументе определим функцию, обрабатываю-

щую ответ сервера. Здесь мы применим ту же самую вспомогательную функцию `$.each(collection, callback(indexInArray, valueOfElement))`, которая позволяет совершать обход массивов и объектов и выполнять определенную функцию для каждого их элемента.

Для того чтобы понять, почему мы передаем в первом аргументе значение `data.items`, нужно просто посмотреть структуру данных, возвращаемых flickr.com. Фрагмент структуры данных в формате JSON можно увидеть, если набрать тот самый адрес, который мы указали в коде:

http://api.flickr.com/services/feeds/photos_public.gne?tags=nature&tagmode=any&format=json&jsoncallback=?.

В нашем случае `data.items` — это массив объектов, получив который мы выполняем для каждого его элемента одну и ту же операцию: создаем элемент `img` и добавляем ему атрибуты `src` и `title`. Значения этих атрибутов получаем из соответствующих свойств объекта, который является очередным, *i*-м элементом массива. Затем добавляем получившийся элемент `img` в элемент `div` с помощью метода `appentTo()`.

Как только будут созданы и помещены в `div` три картинки — цикл прервется на условии, которое вернет `false`.

ЗАДАЧА

Используя POST-запрос, необходимо отправить на сервер некоторые данные. Ответ, полученный от сервера, должен быть вставлен в определенный элемент вебстраницы.

Решение

Попробуем решить задачу, применив вспомогательную функцию `$.post(url, [data], [success(data, textStatus, jqXHR)], [dataType])` (листинг 9.1.7).

Листинг 9.1.7. Использование вспомогательной функции `$.post()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-1-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:400px; height:200px;
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
```

```
</style>
<script type="text/javascript">
$(function () {
  $("button:first").click(function(){
    $.post("testPost.php",
          { name: "Debora", age: "45" },
            function(data){
            $("div").text(data);
          });
  });
  $("button:last").click(function(){
    $("div").empty();
  });
});
</script>
</head>
<body>
<div></div>
<button>Загрузить</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

В HTML-коде листинга 9.1.7 видим привычную разметку — элемент `div` и два элемента `button`. Интересующие нас действия происходят при щелчке мышью на кнопке **Загрузить**. Здесь мы вызываем вспомогательную функцию `$.post()`, которой передаем некоторые аргументы. Первый аргумент — адрес файла, к которому нужно отправить POST-запрос. Второй аргумент — объект, представляющий данные, отправляемые на сервер. На сервере они будут доступны как элементы глобального массива `POST['name']` и `POST['age']`. И третий аргумент — функция, вызываемая при получении ответа сервера. Она выберет элемент `div` и с помощью метода `text(val)` вставит в него ответ сервера в виде текста.

В этой функции мы задействовали только один из трех возможных аргументов — аргумент `data`, который содержит ответ сервера (в рассмотренном случае — в виде обычного текста). Но есть также аргумент `textStatus`, содержащий статус ответа сервера в виде текста, и аргумент `jqXHR` (до версии 1.5 объект `XMLHttpRequest`), представляющий собой расширенный объект `XMLHttpRequest`.

Исходный код файла `testPost.php` приведен в листинге 9.1.7а.

Листинг 9.1.7а. Исходный код файла `testPost.php`

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
```

```
  if($_POST) {
    print 'Имя: ' . $_POST['name'] . ', возраст: ' . $_POST['age'] . ', время:
' . date('H:i:s', time());
  }
}
?>
```

9.2. Вспомогательные функции *$.ajax()* и *$.ajaxSetup()*

Эти вспомогательные функции библиотеки jQuery могут быть использованы для создания произвольных AJAX-запросов. Функция `$.ajax(options)` принимает в качестве аргумента объект, содержащий настройки конкретного запроса.

Начиная с версии 1.5 можно передать функции два аргумента `$.ajax(url, [settings])`, где в качестве первого аргумента выступает адрес, на который должен быть отправлен запрос, и вторым (необязательным) аргументом является объект, содержащий настройки конкретного запроса.

Вспомогательная функция `$.ajaxSetup(options)` принимает в качестве единственного аргумента объект с настройками, которые являются общими по умолчанию для всех AJAX-запросов, отправляемых с этой веб-страницы.

ЗАДАЧА

Поставлена задача отслеживать ход выполнения AJAX-запроса. Необходимо выполнить некоторые действия перед отправкой запроса на сервер. Кроме того, нужно контролировать максимальное время ожидания ответа от сервера и выполнять разные действия в случае успешного ответа и сообщения об ошибке.

Решение

Для решения задачи воспользуемся вспомогательной функцией `$.ajax()` (листинг 9.2.1), которая может предоставить нам гораздо больше возможностей управления AJAX-запросами, чем рассмотренные функции `$.get()` и `$.post()`, по сути, являющиеся высокоуровневыми абстракциями, более простыми в понимании и применении, но с ограниченной функциональностью.

Листинг 9.2.1. Использование вспомогательной функции `$.ajax()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
```

```
div {
  width:400px; height:200px;
  padding:10px; margin:10px;
  border:3px double #369;
  overflow:auto;
}
span { color:#f00; }
</style>
<script type="text/javascript">
$(function () {
  $("button:first").click(function(){
    $.ajax("testAjax.php", {
      type: "POST",
      data: "name=Jonh&age=35",
      timeout: 5000,
      beforeSend: function(){
        $("div").text("Загрузка...");
      },
      success: function(data, textStatus, jqXHR){
        $("div").html(data + '<br />Статус ответа: ' + textStatus +
                    '<br />Код ответа сервера: ' + jqXHR.status);
      },
      error: function(jqXHR, textStatus){
        $("div").html('<span>' + textStatus +
                '</span><br />Код ответа сервера: ' + jqXHR.status);
      }
    });
  });
  $("button:last").click(function(){
    $("div").empty();
  });
});
</script>
</head>
<body>
<div></div>
<button>Загрузить</button>
<button>Очистить</button>
</body>
</html>
```

Обсуждение

В листинге 9.2.1 мы снова видим надоевшую уже HTML-разметку — элемент `div`, предназначенный для помещения в него ответа от сервера, и кнопки **Загрузить** и **Очистить**, выполняющие те действия, названия которых на них написаны.

Подробно будем рассматривать JavaScript-код, который выполняется при щелчке мышью по кнопке **Загрузить**. Именно здесь мы применим вспомогательную функцию `$.ajax()` с двумя аргументами — `url`, к которому будем отправлять запрос, и объектом, состоящим из пар ключ/значение, для настройки этого AJAX-запроса. Все свойства этого объекта необязательные и используются только тогда, когда нужно установить значение, отличное от значения по умолчанию.

> ***Примечание***
>
> Значения по умолчанию для всех AJAX-запросов веб-страницы могут быть установлены для любой опции в `$.ajaxSetup()`.

Но давайте вернемся к примеру из листинга 9.2.1.

Итак, первым аргументом мы передаем `url`, к которому будет отправлен запрос — `testAjax.php`.

Во втором аргументе передаем объект с настройками и первая опция, которую мы определяем в этом объекте — опция `type`, задает требуемый тип запроса. В нашем случае на сервер будет отправлен POST-запрос.

Опция `data` — данные, которые нужно отправить на сервер. Данные представлены в виде строки запроса точно таким же образом, каким они представляются при отправке обычного GET-запроса, т. е. пары имя/значение, объединенные знаком `'&'`.

Опция `timeout` — число в миллисекундах, определяющее время ожидания ответа от сервера. При превышении этого времени запрос будет считаться не выполненным.

Опция `beforeSend` — тут можно определить функцию, выполняемую перед отправкой запроса на сервер. В нашем примере мы просто вставляем в элемент `div` текст, информирующий о том, что начался процесс загрузки. Вообще здесь можно написать функцию, проверяющую какие-либо условия и в зависимости от результата отменяющую выполнение AJAX-запроса, вернув `false`.

Опция `success` — здесь определяем функцию, которая исполняется только в случае успешного выполнения запроса. В примере все полученные от сервера данные вставляются в элемент `div` как HTML-код.

В опции `error` можно определить функцию, вызываемую при возникновении ошибки. В примере мы вставляем в элемент `div` HTML-код, содержащий сообщение об ошибке.

В листинге 9.2.1а приведен исходный код файла `testAjax.php`, к которому мы отправляли наш запрос. Попробуйте изменить количество итераций цикла `for` до 5, задав 5-секундную задержку ответа, и убедитесь, что вы получите сообщение об ошибке — `'timeout'`.

Листинг 9.2.1а. Исходный код файла `testAjax.php`

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
```

```
  for($i=0;$i<2;$i++){
    sleep(1);
  }
  if($_POST){
    print '<h3>Ответ сервера</h3><p>Время: '.date("H:i:s", time()).'</p>';
  }
}
?>
```

На примере из листинга 9.2.1 мы рассмотрели только несколько возможных опций, хотя нужно отметить, что именно эти опции встречаются наиболее часто. В табл. 9.1 приведен полный список и описания всех возможных опций, применяемых при настройках AJAX-запросов с помощью вспомогательных функций `$.ajax()` или `$.ajaxSetup()`.

Таблица 9.1. *Опции вспомогательных функций $.ajax() и $.ajaxSetup()*

Опция	Описание
`async`	По умолчанию все AJAX-запросы передаются в асинхронном режиме. Опция принимает значения `true` или `false`. По умолчанию установлено `true`. Если необходимо выполнить синхронный запрос, ее нужно установить в `false`. При этом следует помнить, что пока синхронный запрос не завершен, блокируется любая активность браузера. Кроссдоменные запросы и запросы с `dataType: 'jsonp'` не поддерживают синхронный запрос
`beforeSend(jqXHR, settings)`	Функция, которая может быть вызвана с целью изменения объекта `jqXHR` (в jQuery 1.4.*x* объект `XMLHTTPRequest`) перед его отправкой на сервер. Можно использовать для проверки условий, при которых должна состояться передача объекта на сервер и т. п. В этой функции можно вернуть `false`, чтобы отменить отправку запроса на сервер
`cache`	Опция добавлена в jQuery, начиная с версий 1.2. Принимает логические значения `true` или `false`. Если установлена в `false`, то браузер не будет кэшировать запрашиваемую страницу. По умолчанию задано значение `true`, но если для опции `dataType` установлено значение `script`, то для опции `cache` по умолчанию будет назначено `false`
`complete(jqXHR, textStatus)`	Функция, которая может быть вызвана после завершения AJAX-запроса (и после того как будут выполнены функции, определенные в опциях `success` и `error`). Функция принимает два аргумента: объект `jqXHR` (в jQuery 1.4.*x* объект `XMLHTTPRequest`) и строку, описывающую тип успешно выполненного запроса
`contents`	Добавлено в версии 1.5. Объект, состоящий из пар строка/регулярное выражение, которые определяют, как jQuery будет разбирать ответ с учетом типа его содержимого
`contentType`	Строка, определяющая используемый `content-type` при передаче данных на сервер. По умолчанию установлен `application/x-www-form-urlencoded`, как наиболее подходящий в большинстве случаев

Таблица 9.1 *(продолжение)*

Опция	Описание
`context`	Объект, определяющий контекст для callback-функций AJAX-запроса
`converters`	Добавлено в версии 1.5. Конвертер типа данных в тип данных. Значение по умолчанию `{"* text": window.String, "text html": true, "text json": jQuery.parseJSON, "text xml": jQuery.parseXML}`. Каждое значение объекта является функцией, которая возвращает преобразованное значение ответа сервера
`crossDomain`	Добавлено в версии 1.5. Значение по умолчанию: `true` для кроссдоменных запросов и `false` для запросов к своему домену. Если необходимо заставить кроссдоменный запрос (например JSONP) выполняться к своему домену, следует установить значение `true`
`data`	Объект или строка с данными, предназначенными для передачи на сервер. Преобразуются в строку запроса, если уже не являются таковой. Для запросов типа GET добавляются в URL. Объект должен представлять собой пары ключ/значение. Если значение будет являться массивом, jQuery упорядочит множество значений с одним и тем же ключом. Например: `{foo:["bar1", "bar2"]}` превратится в `'&foo=bar1&foo=bar2'`
`dataFilter(data, type)`	Функция, которая будет использоваться для обработки данных, возвращенных в объекте `XMLHttpRequest`. Это функция предварительной фильтрации, которая должна возвратить предварительно обработанные данные. Функция принимает два аргумента: данные, возвращенные с сервера в аргументе `data`, и значение опции `dataType` в аргументе `type`
`dataType`	Строка, определяющая тип данных, которые ожидаются в ответе сервера. Если опция не определена, то jQuery передаст в `success` либо `responseXML`, либо `responseText`. Выбор будет произведен на основании MIME-типа ответа. Доступные значения: • `xml` — возвращается XML-документ, который может быть обработан через jQuery; • `html` — возвращается HTML как простой текст. Включенные теги `<script>` расцениваются как вставленные в объектную модель документа (DOM); • `script` — ответ оценивается как JavaScript и возвращается как простой текст; • `json` — ответ оценивается как данные в формате JSON и возвращается как объект JavaScript; • `jsonp` — ответ оценивается как данные в формате JSON, загружаемые с удаленного домена с использованием JSONP; • `text` — строка, представляющая собой простой текст

Таблица 9.1 *(продолжение)*

Опция	Описание
`error(jqXHR, textStatus, errorThrown)`	Функция, которая может быть вызвана, если запрос потерпел неудачу. Функция принимает три аргумента: объект `jqXHR` (в jQuery 1.4.*x* объект `XMLHTTPRequest`), строку, описывающую тип ошибки, которая произошла, и дополнительный объект исключения, если он имеется. Возможные значения второго аргумента — `"timeout"`, `"error"`, `"abort"` и `"parsererror"`
`global`	Опция определяет, вызывать или нет глобальные обработчики событий для этого AJAX-запроса. По умолчанию установлено `true`. Необходимо установить опцию в `false`, чтобы предотвратить вызов глобальных обработчиков (например `ajaxStart` или `ajaxStop`) из этого запроса. Эту опцию можно использовать для управления различными AJAX-событиями
`headers`	Добавлено в версии 1.5. Объект, содержащий дополнительные заголовки, которые могут быть отправлены. Этот параметр устанавливается до вызова функции, определенной в `beforeSend` и поэтому на этом этапе имеется возможность внести изменения
`ifModified`	Опция принимает значения `true` или `false`. По умолчанию установлено `false`. Если установить значение `true`, то это позволит запросу быть успешным только в том случае, если ответ сервера изменился со времени последнего запроса к нему. Реализовано проверкой заголовка `Last-Modified`
`isLocal`	Добавлено в версии 1.5.1. Значение по умолчанию зависит от текущего значения свойства `protocol` объекта `location`. Позволяет распознавать текущее окружение как локальное, даже если jQuery не распознает его. Если требуется изменить значение опции, лучше делать это в `$.ajaxSetup()`
`jsonp`	Строка, которая переопределяет имя callback-функции при выполнении запроса с `jsonp`. Это значение будет использоваться вместо callback в той части строки запроса, где передается `callback=?` (в GET-запросе) или в данных (в POST-запросе). Например, вариант `{jsonp: onJsonPLoad}` приведет к передаче на сервер данных `onJsonPLoad=?`
`jsonpCallback`	Указывается имя callback-функции для запроса JSONP. Это значение будет использоваться вместо случайных имен, создаваемых jQuery
`mimeType`	Добавлено в версии 1.5.1. Строка, в которой можно указать значение MIME-типа, чтобы изменить значение, определенное в объекте `XMLHTTPRequest`
`password`	Строка, где можно указать пароль для ответа на запрос HTTP-аутентификации
`processData`	Опция может принимать значения `true` или `false`. По умолчанию установлено значение `true`. Технически, данные передаются как объект, который будет обработан и преобразован в строку запроса, соответствующую `content-type application/x-www-form-urlencoded`. Если же необходимо передать DOM-документ или другие данные, не подлежащие обработке, необходимо установить значение этой опции в `false`

Таблица 9.1 *(окончание)*

Опция	Описание
`scriptCharset`	Строка, определяющая кодировку данных. Может применяться исключительно с запросами типа GET и только, если опция `DataType` имеет значения `jsonp` или `script`
`statusCode`	Добавлено в версии 1.5. Объект, где ключом является числовой код ответа сервера, а значением — функция, которая будет вызвана при получении соответствующего кода ответа от сервера
`success(data, textStatus, jqXHR)`	Функция, которая будет вызвана, когда запрос будет успешно завершен. Функция принимает три аргумента: данные, возвращенные с сервера и отформатированные согласно значению, установленному в опции `dataType`, строку, описывающую статус ответа сервера, и объект `jqXHR` (в jQuery 1.4.*x* объект `XMLHTTPRequest`)
`timeout`	Число в миллисекундах. С помощью этой опции устанавливается локальный тайм-аут для AJAX-запроса. Переопределяет значение глобального тайм-аута, если он определен через `$.ajaxSetup()`
`traditional`	Может принимать значения `true` и `false`. Устанавливается в `true`, если необходим традиционный способ упорядочивания параметров в строке запроса
`type`	По умолчанию — "GET". Строка, определяющая тип запроса, — "POST" или "GET". Необходимо отметить, что другие типы запросов, такие как "PUT" и "DELETE", также допустимы здесь, но они поддерживаются не всеми браузерами
`url`	Строка, содержащая адрес ресурса в Интернете, к которому будет отправлен запрос. Значение по умолчанию — текущая страница
`username`	Строка, где можно указать имя пользователя, которое будет использоваться в ответ на запрос HTTP-аутентификации
`xhr`	Функция обратного вызова для создания объекта `XMLHTTPRequest`. По умолчанию — `ActiveXObject` (если доступен), в противном случае `XMLHTTPRequest`. Переопределение позволяет обеспечить свою собственную реализацию создания объекта `XMLHTTPRequest`
`xhrFields`	Добавлено в версии 1.5.1. Объект, содержащий пары имя/значение для установки в собственном объекте `XMLHTTPRequest`

ЗАДАЧА

На веб-странице планируется несколько элементов, которые будут отправлять AJAX-запросы на сервер. Все AJAX-запросы будут отправляться одному и тому же файлу, одним и тем же методом. Различаться для всех запросов будут только отправляемые на сервер данные. Учитывая это, необходимо по возможности оптимизировать код.

Решение

Решить такую проблему вполне реально, если использовать вспомогательную функцию `$.ajaxSetup(options)` (листинг 9.2.2).

Листинг 9.2.2. Использование вспомогательной функции `$.ajaxSetup()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:350px; height:150px;
  padding:10px; margin:5px;
  border:3px double #369;
  overflow:auto;
}
#result {
  padding:10px; margin:5px;
  border:1px dotted #f00;
  width:350px; height:50px;
}
span { color:#f00; }
</style>
<script type="text/javascript">
$(function () {
  $.ajaxSetup({
    url: "testAjaxSetup.php",
    type: "POST",
    timeout: 3000,
    beforeSend: function(){
      $("div:last").empty();
      $("#result").text("Загрузка...");
    },
    success: function(data){
      $("div:last").html(data);
      $("#result").text("Готово!");
    },
    error: function(jqXHR, textStatus){
      $("#result").html("<span>" + status + "</span>");
    }
  });
```

```
  $("button:eq(0)").click(function(){
    $.ajax({ data: "q=1&er=none" });
  });
  $("button:eq(1)").click(function(){
    $.ajax({ data: "q=2&er=none" });
  });
  $("button:eq(2)").click(function(){
    $.ajax({ data: "q=3&er=yes" });
  });
});
</script>
</head>
<body>
<div id="result"></div>
<div></div>
<button>Запрос №1</button>
<button>Запрос №2</button>
<button>Запрос №3 (с ошибкой)</button>
</body>
</html>
```

Обсуждение

Сначала ознакомимся с HTML-разметкой, приведенной в листинге 9.2.2. Элемент `div` с идентификатором `#result` будет служить контейнером для отображения процесса выполнения AJAX-запросов. В следующий за ним элемент `div` будем помещать ответ сервера (при его наличии). С помощью кнопок — элементов `button` — отправлять AJAX-запросы на сервер.

Теперь примемся за JavaScript-код. С самого начала мы вызываем функцию `$.ajaxSetup()`. В качестве аргумента эта функция принимает объект `options`, содержащий необходимые настройки AJAX-запросов. Очень важно запомнить, что это настройки, которые будут действовать для *всех AJAX-запросов*, отправляемых с этой страницы. А это как раз то, что нам нужно, чтобы попытаться уменьшить наш код.

Настройки, которые можно передать в объекте `options`, абсолютно те же, что мы рассматривали ранее.

Давайте начнем определять настройки для всех AJAX-запросов нашей веб-страницы.

В опции `url` зададим относительный адрес файла на сервере, к которому будут отправляться все запросы, — `testAjaxSetup.php`.

Определим тип запроса как POST в опции `type` и установим тайм-аут 3000 мс в опции `timeout`.

В опции `beforeSend` назначим функцию, которая очищает внутреннее содержимое того элемента `div`, в который мы собираемся вставлять ответ от сервера, а в элемент

с идентификатором #result вставляет текст, сообщающий о том, что началась загрузка.

Определим функцию, которая будет выполняться при успешном завершении запроса в опции success. Эта функция будет вставлять ответ сервера в виде HTML-кода в элемент div, который мы для этого выбрали, а также вставит в элемент с идентификатором #result сообщение об успешном завершении запроса.

Осталось задать функцию, отвечающую за обработку сообщения об ошибке в опции error — сообщение об ошибке будет помещено в тег и вставлено в виде HTML-кода в элемент #result.

Самое сложное мы уже сделали, и нам осталось только определить обработчики события click для каждой из трех кнопок, с помощью которых будут отправляться разные AJAX-запросы. Естественно, что функции-обработчики для всех трех кнопок используют вспомогательную функцию $.ajax(). Но посмотрите, во всех трех случаях мы передаем всего лишь одну опцию — data, в которой передаются данные для отправки на сервер, файл testAjaxSetup.php (листинг 9.2.2а).

Листинг 9.2.2а. Исходный код файла testAjaxSetup.php

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
  if($_POST['er'] == "yes") { for($i=0;$i<5;$i++) sleep(1); }
  if($_POST){
    print '<h3>Ответ сервера</h3>
           <p>В '.date("H:i:s d-m-Y", time()).' получены данные:<br />
           <strong>q = ' . $_POST['q'] . ',
           er = ' . $_POST['er'] . '</strong></p>';
  }
}
?>
```

Если сервер получит значение 'yes' в параметре er, сработает 5-секундная задержка, что приведет к превышению времени ожидания ответа от сервера и, следовательно, к сообщению об ошибке. Это произойдет при нажатии на третью кнопку. При нажатии же на первую и вторую кнопки запросы будут успешно выполнены, и в элементе div можно будет увидеть сообщение о том, какие данные были получены, а также дату и время их получения.

9.3. События AJAX

ЗАДАЧА

Необходимо познакомить читателя с локальными и глобальными событиями, происходящими в ходе выполнения AJAX-запроса.

Решение

Для того чтобы познакомить читателя с локальными и глобальными событиями, которые происходят при выполнении AJAX-запросов и последовательностью их возникновения, написан код, который приведен в листинге 9.3.1.

Листинг 9.3.1. Локальные и глобальные события при выполнении AJAX-запросов

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
#result {
  width:410px; height:180px;
  margin:5px; padding:3px;
  border:1px solid #369;
}
button {
  width:200px;
  margin:5px;
}
</style>
<script type="text/javascript">
$(function () {
  // глобальные настройки
  $.ajaxSetup({
    url: "testEvents.php",
    type: "POST",
    timeout: 4000,
    beforeSend: function() {
      $("#result").append(new Date().getTime()+' ==> beforeSend<br />');
    },
    success: function() {
      $("#result").append(new Date().getTime()+' ==> success<br />');
    },
    error: function() {
      $("#result").append(new Date().getTime()+' ==> error<br />');
    },
    complete: function() {
      $("#result").append(new Date().getTime()+' ==> complete<br />');
    }
  });
```

```
  // успешный запрос
  $("#well").click(function() {
    $("#result").empty();
    $.ajax({
      data: "er=none"
    });
  });
  // ошибочный запрос
  $("#bad").click(function() {
    $("#result").empty();
    $.ajax({
      data: "er=yes"
    });
  });
  // обработка глобальных событий
  $("#result").ajaxStart(function() {
    $(this).append(new Date().getTime()+' ==> ajaxStart<br />');
  }).ajaxSend(function() {
    $(this).append(new Date().getTime()+' ==> ajaxSend<br />');
  }).ajaxSuccess(function() {
    $(this).append(new Date().getTime()+' ==> ajaxSuccess<br />');
  }).ajaxError(function() {
    $(this).append(new Date().getTime()+' ==> ajaxError<br />');
  }).ajaxComplete(function() {
    $(this).append(new Date().getTime()+' ==> ajaxComplete<br />');
  }).ajaxStop(function() {
    $(this).append(new Date().getTime()+' ==> ajaxStop<br />');
  });
});
</script>
</head>
<body>
<div id="result"></div>
<button id="well">Успешный ajax-запрос</button>
<button id="bad">Ошибочный ajax-запрос</button>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 9.3.1, не представляет собой ничего особенного. Две кнопки — элементы `button`, с помощью которых мы будем отправлять AJAX-запросы. Кнопка с идентификатором `#well` отправит корректный запрос, а кнопка с идентификатором `#bad` — запрос, который заведомо должен завершиться ошибкой.

Поскольку мы имеем два AJAX-запроса с одной веб-страницы, и отличаться они будут только данными, отправляемыми на сервер, то написание JavaScript-кода

начнем со вспомогательной функции `$.ajaxSetup()`. Вспомним, что с ее помощью можно определять настройки по умолчанию для всех AJAX-запросов на веб-странице.

Начнем определять настройки по умолчанию с опции `url`, в которой передадим относительный адрес файла для отправки AJAX-запросов — `testEvents.php`.

В опции `type` определим тип запроса POST, а в опции `timeout` установим максимальное время ожидания ответа от сервера 4000 мс.

Функции, которые мы определяем в опциях `beforeSend`, `success`, `error` и `complete`, выполняют одинаковые операции — получают число миллисекунд, прошедших с 1 января 1970 г., объединяют его с названием события и добавляют в элемент `div` с идентификатором `#result`. Это сделано для того, чтобы наглядно показать последовательность возникновения события при отправке AJAX-запроса.

Далее назначаем обработчики события `click` для двух кнопок. В обоих случаях при возникновении этого события, т. е. при щелчке мышью на элементе `button`, сначала очищаем содержимое элемента `div` для того, чтобы выводить туда текущие результаты. А затем реализуем AJAX-запросы с помощью вспомогательной функции `$.ajax()`, передавая данные в опции `data`. Но если в первом случае для кнопки с идентификатором `#well` мы передаем данные `'er=none'`, то во втором случае для кнопки с идентификатором `#bad` будут переданы данные `'er=yes'`. Это делается для того, чтобы иметь возможность имитировать возникновение ошибки на сервере, в файле `testEvents.php`, исходный код которого приведен в листинге 9.3.1а.

Листинг 9.3.1а. Исходный код файла `testEvents.php`

```
<?php
header('Content-Type: text/html; charset=utf-8');
if($_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest') {
  if($_POST['er'] == "yes") {
     sleep(5);
  }
}
?>
```

Если мы получаем `'yes'` в качестве значения `$_POST['er']`, то задерживаем ответ сервера на 5 с, превышая таким образом время ожидания ответа от сервера, и вызываем ошибку выполнения AJAX-запроса.

Если обработчики для локальных событий мы уже определили внутри вспомогательной функции `$.ajaxSetup()`, то до обработчиков глобальных событий добрались только теперь.

Выберем элемент `div` по его идентификатору `#result` и свяжем с ним целую цепочку вызовов функций-обработчиков глобальных событий: `ajaxStart`, `ajaxSend`, `ajaxSuccess`, `ajaxError`, `ajaxComplete` и `ajaxStop`. Обработчики всех перечисленных глобальных и локальных событий действуют одинаково, т. е. получают число миллисекунд, прошедших с 1 января 1970 г., объединяют его с названием события и добавляют в элемент `div` с идентификатором `#result`.

Если теперь попробовать нажать кнопку **Успешный ajax-запрос**, то можно будет увидеть порядок следования событий при выполнении успешного AJAX-запроса (рис. 9.1).

Если нажать кнопку **Ошибочный ajax-запрос**, тоже можно будет увидеть порядок следования событий, но только немного других — характерных для ошибочных AJAX-запросов (рис. 9.2).

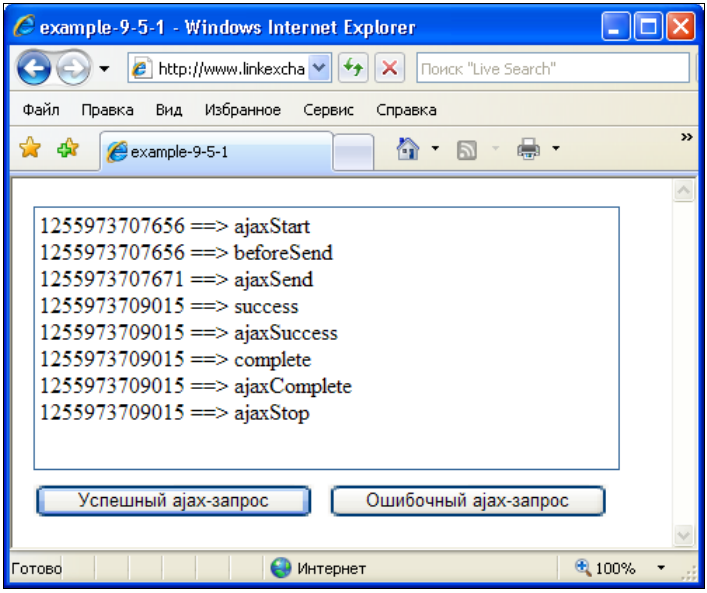


Рис. 9.1. Порядок возникновения событий при выполнении успешного AJAX-запроса

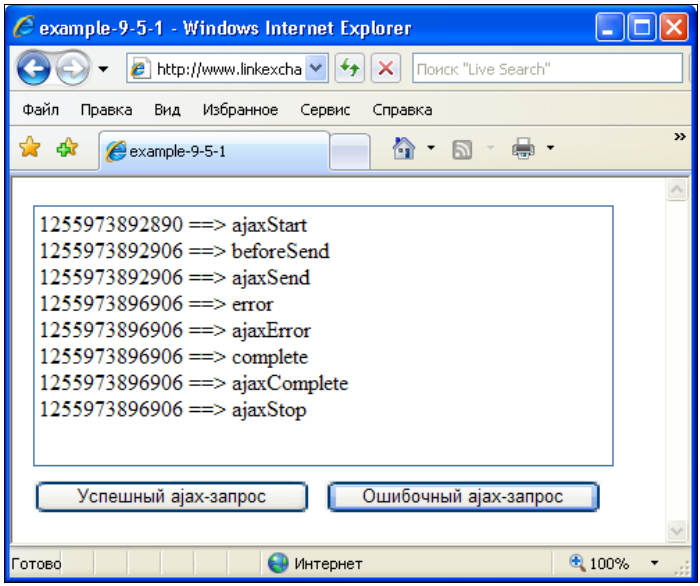


Рис. 9.2. Порядок возникновения событий при выполнении ошибочного AJAX-запроса

Описание локальных и глобальных событий, происходящих при выполнении AJAX-запросов, приведено в табл. 9.2.

Таблица 9.2. *Локальные и глобальные события при выполнении AJAX-запросов*

Опция	Описание
`ajaxStart`	Глобальное событие. Распространяется, если AJAX-запрос начат и нет других AJAX-запросов, выполняющихся в это же время. Вызывается до создания объекта `XMLHttpRequest`
`beforeSend`	Локальное событие. Вызывается до того, как начнет выполняться AJAX-запрос, позволяет изменить объект `XMLHttpRequest` (например, при необходимости можно установить дополнительные заголовки)
`ajaxSend`	Глобальное событие. Вызывается до начала выполнения AJAX-запроса, но после создания объекта `XMLHttpRequest`
`success`	Локальное событие. Вызывается только, если AJAX-запрос был успешно выполнен (нет ошибок с сервера, нет ошибок в данных)
`ajaxSuccess`	Глобальное событие. Вызывается только при успешном выполнении AJAX-запроса, т. е. при получении ответа от сервера, содержащего код статуса, свидетельствующего об успешном выполнении
`error`	Локальное событие. Вызывается только при возникновении ошибок AJAX-запроса (никогда не могут возникнуть оба события вместе — успешное и ошибочное выполнение AJAX-запроса)
`ajaxError`	Глобальное событие. Вызывается только в случае получения ответа от сервера, содержащего код статуса, свидетельствующего об ошибке выполнения запроса
`complete`	Локальное событие. Вызывается независимо от того, был ли AJAX-запрос успешно выполнен или нет. Вы будете всегда получать это событие, даже для AJAX-запросов, которые выполнялись синхронно
`ajaxComplete`	Глобальное событие. Это событие ведет себя точно так же, как локальное событие `complete` и будет вызываться каждый раз, когда AJAX-запрос завершается
`ajaxStop`	Глобальное событие. Вызывается, если больше нет выполняющихся AJAX-запросов

9.4. Полезные вспомогательные функции

ЗАДАЧА

Данные из формы должны быть отправлены на сервер с помощью AJAX-запроса. Необходимо собрать значения всех полей формы и подготовить из них строку запроса.

Решение

Конечно, такую задачу можно было бы решать, получая значение каждого элемента формы и объединяя их в строку запроса. Но гораздо проще использовать метод `$.serialize()`. Такое решение приведено в листинге 9.4.1.

Листинг 9.4.1. Использование метода `$.serialize()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-9-4-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">

</style>
<script type="text/javascript">
$(function () {
  $("form").submit(function() {
    alert($(this).serialize());
    return false;
  });
});
</script>
</head>
<body>
<form>
  <div><input type="text" name="a" value="1" id="a" /></div>
  <div><input type="text" name="b" value="2" id="b" /></div>
  <div><input type="hidden" name="c" value="3" id="c" /></div>
  <div>
    <textarea name="d" rows="8" cols="40">4</textarea>
  </div>
  <div><select name="e">
    <option value="5" selected="selected">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
  </select></div>
  <div>
    <input type="checkbox" name="f" value="8" id="f" />
  </div>
  <div>
    <input type="submit" name="g" value="Submit" id="g" />
  </div>
</form>
</body>
</html>
```

Обсуждение

В листинге 9.4.1 приведена форма с некоторыми возможными типами полей, данные из которых должны быть отправлены на сервер с помощью AJAX-запроса. Сам

AJAX-запрос нас сейчас интересовать не будет — главное, получить данные в том виде, в котором они будут использованы.

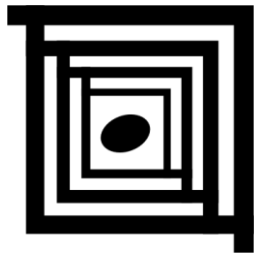
Давайте посмотрим на JavaScript-код. Мы назначаем обработчик события `submit`, в котором применяем к форме метод `.serialize()`. В окне предупреждения `alert` мы увидим готовую строку запроса, представляющую собой пары ключ/значение, объединенные знаком `&`.

Нельзя не упомянуть также, что существует метод `.serializeArray()`. Если бы мы применили его вместо `.serialize()` в листинге 9.4.1, то он вернул бы нам данные из формы в виде массива объектов. См. листинг 9.4.1a.

Листинг 9.4.1a. Результат использования метода `$.serializeArray()`

```
[
  { name: "a", value: "1" },
  { name: "b", value: "2" },
  { name: "c", value: "3" },
  { name: "d", value: "4" },
  { name: "e", value: "5" }
]
```

Использовать методы `.serialize()` и `.serializeArray()` несложно, однако нужно помнить, что для успешного применения этих методов форма должна соответствовать стандартам W3C. В частности, каждое поле формы должно иметь атрибут `name`.



ГЛАВА 10

Полезные вспомогательные функции и методы jQuery

10.1. Некоторые операции с массивами и объектами в jQuery

ЗАДАЧА

Необходимо применить какую-либо функцию к каждому элементу массива или объекта.

Решение

Для решения задачи воспользуемся вспомогательной функцией `$.each(collection, callback(indexInArray, valueOfElement))`. Не путайте ее с методом `each()`, который применяется для обхода набора элементов jQuery. Функция `$.each(collection, callback(indexInArray, valueOfElement)` позволяет совершать обход как массивов, так и объектов (листинг 10.1.1).

Листинг 10.1.1. Использование вспомогательной функции `$.each()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  float:left;
  width:100px; height:100px;
  padding:10px; margin:10px;
  border:1px solid #369;
}
```

```
p { color:#f00; }
</style>
<script type="text/javascript">
$(function () {
  var arr = [ "one", "two", "three", "four", "five" ];
  var obj = { one:1, two:2, three:3, four:4, five:5 };
  $.each(arr, function() {
    $("#" + this).text("id='" + this + "'");
    return (this != "four");
  });
  $.each(obj, function(n, val) {
    $("#" + n).append("<p>" + val + "</p>");
  });
});
</script>
</head>
<body>
<div id="one"></div>
<div id="two"></div>
<div id="three"></div>
<div id="four"></div>
<div id="five"></div>
</body>
</html>
```

Обсуждение

Пять элементов `div`, каждый из которых имеет атрибут `id`, — вот вся HTML-разметка, показанная в листинге 10.1.1. Она послужит нам исключительно в демонстрационных целях.

Займемся рассмотрением JavaScript-кода. В самом начале объявляем массив `arr` и объект `obj`, обход которых попробуем совершить.

Вспомогательная функция `$.each()` принимает первым аргументом объект или массив, а вторым аргументом — функцию, которую необходимо выполнить для каждого элемента массива или объекта. В свою очередь эта функция тоже может принимать два аргумента. Первый — ключ (для объектов) или индекс (для массивов), второй — значение.

Попробуем совершить обход массива `arr`, передав его в качестве первого аргумента вспомогательной функции `$.each()`. Во втором аргументе передаем функцию, добавляющую в каждый элемент `div` текст со значением его идентификатора. Выражение, на основании которого выбирается нужный элемент `div`, состоит из знака `#` и указателя `this`, содержащего при каждой итерации значение очередного элемента массива.

Однако мы не будем совершать обход всего массива и прервемся сразу после того, как обнаружим в массиве значение `'four'`. Для того чтобы прервать итерации, функция, применяемая к каждому элементу массива, просто должна вернуть `false`.

Похожим образом обстоит дело и с обходом объекта `obj`. Передадим его в качестве первого аргумента вспомогательной функции `$.each()`, а во втором аргументе передаем функцию, которая добавит в каждый элемент `div` параграф, содержащий значение соответствующего свойства объекта `obj`.

ЗАДАЧА

Требуется объединить два объекта в один.

Решение

Для решения такой задачи вполне подойдет вспомогательная функция `$.extend(target,object1,[objectN])` (листинг 10.1.2).

Листинг 10.1.2. Использование вспомогательной функции `$.extend()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var empty = {}
  var defaults = { validate: false, limit: 5, name: "foo" };
  var options = { validate: true, foo: "bar" };
  var settings = $.extend(empty, defaults, options);
  $.each(settings, function(n, val) {
    $("body").append("<p>" + n + ": " + val + "</p>");
  });
});
</script>
</head>
<body>
</body>
</html>
```

Обсуждение

В примере из листинга 10.1.2 внутри `body` нет вообще никаких элементов — они будут созданы позже с помощью JavaScript-кода и послужат только в целях демонстрации.

Что касается JavaScript-кода, то здесь мы объявляем три объекта: `empty`, `defaults` и `options`. В переменной `settings` сохраним результат выполнения вспомогательной

функции `$.extend()`. Мы передаем функции три объявленных ранее объекта и ожидаем, что в переменной `setting` будет представлен результирующий объект.

Попробуем убедиться в этом с помощью функции `$.each()`, передав ей объект `settings` в первом аргументе. Во втором аргументе передаем функцию, добавляющую в `body` элементы `p`, в которые будут выведены название свойства объекта и его значение.

Действительно, в результате выполнения операции объединения объектов мы получили итоговый объект `{'validate': true, 'limit': '5', 'name': 'foo', 'foo': 'bar'}`.

ЗАДАЧА

Необходимо объединить два массива в один.

Решение

Для решения задачи подходит вспомогательная функция `$.merge(first,second)` (листинг 10.1.3).

Листинг 10.1.3. Использование вспомогательной функции `$.merge()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var a = [ 1, 2, 3, 4 ];
  var b = [ 4, 5, 6, 7 ];
  c = $.merge(a, b);
  $("div").text(c.join(", "));
});
</script>
</head>
<body>
<div></div>
</body>
</html>
```

Обсуждение

В HTML-разметке из листинга 10.1.3 только один элемент `div`, куда будет вставлена строка, представляющая собой объединенный в строку результирующий мас-

сив `c`, полученный при объединении массивов `a` и `b` с помощью вспомогательной функции `$.merge()`. Строка будет иметь вид: 1, 2, 3, 4, 4, 5, 6, 7.

ЗАДАЧА

Из существующего массива необходимо выбрать в новый массив только те элементы, которые удовлетворяют определенному условию.

Решение

Задачу решаем, используя вспомогательную функцию `$.grep(array, function(elementOfArray, indexInArray), [invert])` (листинг 10.1.4).

Листинг 10.1.4. Использование вспомогательной функции `$.grep()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var arr = [ 1, 9, 3, 8, 6, 1, 5, 9, 4, 7, 3, 8, 6, 9, 1 ];
  $("div:eq(0)").text(arr.join(", "));
  arr = $.grep(arr, function(n, i){
    return (n != 5 && i > 4);
  });
  $("div:eq(1)").text(arr.join(", "));
  arr = $.grep(arr, function (a) { return a != 9; },true);
  $("div:eq(2)").text(arr.join(", "));
});
</script>
</head>
<body>
<div></div>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 10.1.4, описывает три элемента `div`, куда мы будем вставлять результаты наших исследований.

В самом начале JavaScript-кода объявляем исходный массив `arr`. Чтобы видеть его содержимое, мы объединим массив в строку и вставим в первый элемент `div`. Затем попробуем отфильтровать некоторые элементы массива, для чего передадим массив `arr` вспомогательной функции `$.grep()` в качестве первого аргумента. Второй аргумент — функция, которая будет применена к каждому элементу массива. Эта функция может принимать два аргумента. Первый — значение элемента массива, второй — его индекс. Если функция возвращает `false` — элемент не будет включен в результирующий массив. Мы выбираем в новый массив все элементы, значение которых не равно пяти и индекс элемента массива более четырех. В результате во второй элемент `div` будет вставлена строка: 1, 9, 4, 7, 3, 8, 6, 9, 1.

Если передать функции `$.grep()` необязательный аргумент `invert`, то в результирующий массив будут включены только те значения, которые не соответствуют условию. В последний элемент `div` будет вставлена строка: 9, 9. Заметьте, что мы указывали условие `a!=9`, но, передав в необязательном аргументе `invert` значение `true`, получили противоположный результат.

ЗАДАЧА

В массиве элементов DOM необходимо оставить только уникальные элементы.

Решение

Решение задачи — использование вспомогательной функции `$.unique(array)` (листинг 10.1.5).

Листинг 10.1.5. Использование функции `$.unique()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var divs = $("div").get(); // 6
  var dups = $("div.dup").get(); // 3
  arr = $.merge(divs, dups);
  alert('Всего: ' + arr.length); // 9
  arr = $.unique(arr);
  alert('Уникальных: ' + arr.length); // 6
});
</script>
</head>
```

```
<body>
<div></div>
<div class="dup"></div>
<div class="dup"></div>
<div></div>
<div></div>
<div class="dup"></div>
</body>
</html>
```

Обсуждение

Рассмотрим HTML-разметку, показанную в листинге 10.1.5. Есть шесть элементов `div`, три из которых имеют атрибут `class` со значением `'dup'`.

Если с помощью выражений `div` и `div.dup`, указанных в селекторе, и метода `get()` получить в переменных `divs` и `dups` массивы, каждый элемент которых будет представлять собой элемент DOM, то совершенно очевидно, что `divs` будет содержать шесть элементов, а `dups` — только три. Соответственно при объединении их с помощью вспомогательной функции `$.merge()` в результирующем массиве окажется уже девять элементов.

Применим к получившемуся массиву `arr` вспомогательную функцию `$.unique()` и получим массив, содержащий только шесть уникальных элементов.

ЗАДАЧА

Необходимо определить наличие/отсутствие в массиве элемента с определенным значением.

Решение

Решаем задачу с помощью вспомогательной функции `$.inArray(value,array)` (листинг 10.1.6).

Листинг 10.1.6. Использование функции `$.inArray()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var arr = [ "one", "two", "three", "four", "five", "six", "seven", "eight",
"nine", "ten" ];
```

```
  $("p").text(arr.join(", "));
  $("button").click(function(){
    alert($.inArray($("input").val(), arr));
  });
});
</script>
</head>
<body>
<p></p>
<input type="text" />
<button>Искать</button>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 10.1.6, — это элементы `p`, `input` и `button`. Объединим массив в строку и выведем ее в параграф `p`. В элемент `input` будем вводить какое-либо значение и при нажатии кнопки **Искать** выполним поиск в массиве, передавая во вспомогательную функцию `$.inArray()` первым аргументом значение, полученное из `input`, а вторым — массив.

В итоге при нажатии кнопки **Искать** получим окно предупреждения, в котором будет выведен индекс элемента в массиве или `'-1'`, если значение, введенное в элемент `input`, в массиве отсутствует.

ЗАДАЧА

Над всеми элементами массива необходимо провести какие-либо преобразования.

Решение

Для решения такой задачи хорошо подходит вспомогательная функция `$.map(array, callback(elementOfArray, indexInArray))` (листинг 10.1.7).

Листинг 10.1.7. Использование вспомогательной функции `$.map()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var arr = ["zero", "one", "two", "three", "four", "five", "six", "seven",
"eight", "nine"];
```

```
  $("p:first").text(arr.join(", "));
  arr = $.map(arr, function(n, i){
    return (n.toUpperCase() + i);
  });
  $("p:last").text(arr.join(", "));
});
</script>
</head>
<body>
<p></p>
<p></p>
</body>
</html>
```

Обсуждение

HTML-код, приведенный в листинге 10.1.7, — два элемента `p`, которые придают наглядность примеру.

В самом начале JavaScript-кода определяем массив `arr`. Объединим этот массив в строку и вставим в первый элемент `p`. Затем вызовем вспомогательную функцию `$.map()`, которой в качестве первого аргумента передадим массив `arr`, в качестве второго аргумента выступает функция, выполняющая преобразования над каждым элементом массива. Функция может принимать два аргумента — значение элемента массива и его индекс.

Значение каждого элемента массива переведем в верхний регистр и добавим к нему его индекс. Получившийся массив объединим в строку и выведем во втором элементе `p`.

ЗАДАЧА

Необходимо проверить, является ли некий набор элементов массивом, и, если нет, преобразовать его в массив.

Решение

Для решения задачи подойдут вспомогательные функции `$.isArray(obj)` и `$.makeArray(obj)` (листинг 10.1.8).

Листинг 10.1.8. Использование вспомогательных функций `$.isArray()` и `$.makeArray()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-8</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:20px;
  margin:10px; padding:5px;
  color:#fff; background-color:#00f;
}
</style>
<script type="text/javascript">
$(function () {
  $("button:first").click(function(){
    $.isArray($("div")) ? alert("Это массив!") : alert("Это НЕ массив!");
  });
  $("button:last").click(function(){
    var arr = $.makeArray($("div"));
    $.isArray(arr) ? alert("Это массив!") : alert("Это НЕ массив!");
  });
});
</script>
</head>
<body>
<div>Первый</div>
<div>Второй</div>
<div>Третий</div>
<div>Четвертый</div>
<button>.isArray()</button>
<button>.makeArray()</button>
</body>
</html>
```

Обсуждение

Прежде чем начать обсуждение, необходимо заметить, что некоторые методы JavaScript, и в частности jQuery, возвращают объекты, подобные массивам. Такие объекты обладают некоторыми свойствами массива, но в действительности массивами не являются, поскольку в них отсутствуют отдельные методы, например `pop()` и `reverse()`.

HTML-код, приведенный в листинге 10.1.8, — четыре элемента `div`, используемые в качестве "подопытных", и две кнопки `button`, при нажатии которых мы будем выполнять некоторые действия.

Смотрим JavaScript-код. При нажатии на кнопку **.isArray()** передаем набор элементов `div` вспомогательной функции `$isArray()`. Если переданный ей аргумент является массивом — получим соответствующее сообщение. Но поскольку набор элементов массивом не является, мы получаем сообщение "Это НЕ массив!". Нажима-

ем кнопку **.makeArray()**. При ее нажатии передаем набор элементов `div` вспомогательной функции `$.makeArray()`, которая преобразует набор в полноценный массив. Сохраняем его в переменной `arr` и передаем на проверку в `$.isArray()`. Получаем сообщение "Это массив!".

Необходимо отметить, что полученный полноценный теперь массив состоит из элементов DOM.

ЗАДАЧА

Имеется несколько объектов. Необходимо выяснить, является ли объект пустым (т. е. не содержащим свойств) или простым (т. е. созданным с помощью `{}` или `new Object`).

Решение

Для решения этой задачи воспользуемся вспомогательными функциями `$.isEmptyObject(object)` и `$.isPlainObject(object)` (листинг 10.1.9).

Листинг 10.1.9. Использование вспомогательных функций `$.isEmptyObject()` и `$.isPlainObject()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-9</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:100px; height:20px;
  margin:10px; padding:5px;
  color:#fff; background-color:#00f;
}
</style>
<script type="text/javascript">
$(function () {
  var test = { "one": "один", "two": "два" }
  var empty = new Object;
  $("button:eq(0)").click(function(){
    $.isPlainObject($("div")) ? alert("Это объект!") : alert("Это НЕ объект!");
  });
  $("button:eq(1)").click(function(){
    $.isPlainObject(test) ? alert("Это объект!") : alert("Это НЕ объект!");
  });
```

```
  $("button:eq(2)").click(function(){
    $.isEmptyObject(empty) ? alert("Объект пустой!") : alert("Объект НЕ
пустой!");;
  });
});
</script>
</head>
<body>
<div>Первый</div>
<div>Второй</div>
<div>Третий</div>
<div>Четвертый</div>
<button>div's</button>
<button>test</button>
<button>empty</button>
</body>
</html>
```

Обсуждение

HTML-разметка, приведенная в листинге 10.1.9, — это четыре элемента `div`, которые послужат для создания специального (не простого) объекта — набора элементов jQuery. С помощью трех кнопок будем проводить "испытания".

В самом начале JavaScript-кода создадим обычные объекты, один из которых пустой. Затем рассмотрим, что будет происходить при нажатии кнопок.

Нажимаем кнопку с надписью **div's**. Передав вспомогательной функции набор элементов, созданный с помощью селектора jQuery, сможем убедиться, что "Это НЕ объект!". На самом деле это конечно объект, но не простой, а специальный.

Нажимаем кнопку с надписью **test** и передаем в `$.isPlainObject()` объект `test`, созданный в начале. Убеждаемся, что это действительно объект, причем объект простой, созданный с помощью `{}` или `new Object`.

Наконец, нажимая кнопку с надписью **empty**, передаем во вспомогательную функцию `$.isEmptyObject()` объект `empty` и убеждаемся, что он действительно пустой — видим сообщение "Объект пустой!".

ЗАДАЧА

Имеется массив JavaScript, некоторые элементы которого являются функциями. Требуется отыскать эти элементы.

Решение

Для решения задачи подойдет вспомогательная функция `$.isFunction(obj)` (листинг 10.1.10).

Листинг 10.1.10. Использование вспомогательной функции `$.isFunction()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-1-10</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  function stub() {  }
  var objs = [ null,
               function () {},
               { x:15, y:20 },
               stub,
               "function"
             ];
  $.each(objs, function (i) {
    var isFunc = jQuery.isFunction(objs[i]);
    $("span").eq(i).text(isFunc);
  });
});
</script>
</head>
<body>
<div>$.isFunction(objs[0]): <span></span></div>
<div>$.isFunction(objs[1]): <span></span></div>
<div>$.isFunction(objs[2]): <span></span></div>
<div>$.isFunction(objs[3]): <span></span></div>
<div>$.isFunction(objs[4]): <span></span></div>
</body>
</html>
```

Обсуждение

HTML-разметка из листинга 10.1.10 — это несколько элементов `div` с вложенными в них элементами `span`. Разметка служит исключительно в демонстрационных целях — в элементы `span` мы будем вставлять результаты.

Подробнее рассмотрим JavaScript-код. Массив `objs` — это "подопытный" элемент. Самостоятельно рассмотрите каждый элемент массива.

Для обхода массива `objs` вызовем вспомогательную функцию `$.each()`, в которую первым аргументом передадим массив `obj`, а вторым — функцию, применяемую к каждому элементу массива.

Проверяем каждый элемент массива, передавая его вспомогательной функции `$.isFunction()`. Результат проверки вставляем в соответствующий элемент `span`.

Во второй и четвертый элементы span будет вставлено значение true — значит, эти элементы массива являются функциями.

10.2. Некоторые операции с наборами элементов jQuery

ЗАДАЧА

Необходимо узнать число элементов, выбранных в набор jQuery с помощью какого-либо селектора.

Решение

Для решения такой задачи можно использовать метод size() или просто обратиться к свойству length (листинг 10.2.1).

Листинг 10.2.1. Использование метода size() и свойства length

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:60px; height:60px;
  margin:5px; float:left;
  border:1px solid #369;
}
.test {
  background-color:#369;
}
br { clear:left; }
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function(){
    var size = $("div.test").size();
    var length = $("div.test").length;
    alert("size = " + size + " & length = " + length);
  });
});
</script>
</head>
```

```
<body>
<div></div>
<div class="test"></div>
<div></div>
<div class="test"></div>
<div class="test"></div>
<div></div>
<div class="test"></div><br />
<button>Получить</button>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 10.2.1, описывает семь элементов `div`, четыре из которых имеют класс с именем `test`. С помощью кнопки **Получить** будем выводить в окне предупреждения число элементов, выбранных в набор jQuery.

Код JavaScript начинается с того, что с элементом `button` связывается обработчик события `click`. Внутри функции-обработчика мы выбираем элементы `div` с классом `test`, получаем их число с помощью метода `size()` и сохраняем в переменной `size`. Затем проделываем то же самое, только обращаясь к свойству `length`, а результат сохраняем в переменной `length`. После этого выводим значения этих переменных в окне предупреждения вместе с поясняющей надписью.

Ничего удивительного, что результаты совпали: в обеих переменных оказалось число 4, именно столько элементов `div.test` попали в набор jQuery.

> ***Примечание***
>
> Следует отметить, что вариант с обращением к свойству `length` работает несколько быстрее.

ЗАДАЧА

В набор jQuery необходимо выбрать элементы с помощью селектора, получить для каждого из них внутреннее содержимое в виде текста и завершить работу при обнаружении в наборе элемента с определенным идентификатором.

Решение

Для решения задачи применим один из наиболее полезных методов — `each(function(index, Element))`, который позволяет совершать итерации по набору элементов и выполнять функцию, переданную в качестве аргумента в контексте каждого элемента набора. Наверное, звучит не очень понятно, но после того как мы подробно разберем решение задачи, все прояснится (листинг 10.2.2).

Примечание

Не путайте этот метод со вспомогательной функцией `$.each(collection, callback(indexInArray, valueOfElement))`, которая позволяет совершать обход массивов и объектов (см. листинг 10.1.1).

Листинг 10.2.2. Использование метода `each()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:60px; height:60px;
  margin:5px; float:left;
  border:1px solid #369;
}
.test {
  background-color:#369;
}
br { clear:left; }
</style>
<script type="text/javascript">
$(function(){
  $("button").click(function(){
    var a = new Array();
    $("div.test").each(function(i){
      a.push(i + "-" + $(this).text());
       if($(this).is("#stop")) {
         $("p").text(a.join(", "));
        return false;
       }
    });
  });
});
</script>
</head>
<body>
<p> </p>
<div>Один</div>
<div class="test">Два</div>
<div>Три</div>
<div class="test">Четыре</div>
<div id="stop" class="test">Пять</div>
```

```
<div>Шесть</div>
<div class="test">Семь</div><br />
<button>Применить</button>
</body>
</html>
```

Обсуждение

Познакомимся с HTML-разметкой, приведенной в листинге 10.2.2. Семь элементов `div`, четыре из которых имеют класс с именем `test` и в свою очередь один из этих элементов обладает еще и атрибутом `id` со значением `stop`. Мы должны будем выбрать все элементы с классом `test`, получить внутреннее содержимое каждого элемента в виде текста и сохранить его в массиве `a`. Но, как только в наборе нам встретится элемент `div` с идентификатором `stop`, мы должны будем прекратить обход оставшихся элементов, объединить содержимое массива `a` в строку и поместить результат в параграф `p`.

Начнем разбираться с JavaScript-кодом. Начало должно быть знакомым — выбираем элемент `button`, с которым связываем функцию-обработчик события `click`. Внутри функции-обработчика объявляем массив `a`, где будем хранить полученные данные. Затем с помощью указанного в селекторе выражения `div.test` выбираем все интересующие нас элементы и применяем к ним метод `each()`. Единственный аргумент метода — функция, которая будет выполняться в контексте каждого элемента набора.

Еще раз обращаю ваше внимание — в контексте каждого элемента набора. Это действительно важно.

Функция, которая будет выполняться для каждого элемента набора, может принимать аргумент `i`, являющийся индексом элемента в наборе (отсчет начинается от нуля).

Внутри функции мы для каждого элемента набора вставляем в массив `a` его индекс `i` и, через дефис, содержимое элемента в виде текста. Затем с помощью метода `is()` проверяем условие — если очередной элемент набора имеет идентификатор `#stop`, то мы объединяем массив в строку с помощью метода `join()`, вставляем результат в параграф `p` и завершаем обход набора элементов, возвращая `false`.

Если выполнить пример, приведенный в листинге 10.2.2, то в результате в параграф `p` будет вставлен текст "0-Два, 1-Четыре, 2-Пять".

ЗАДАЧА

Требуется выбрать единственный элемент из набора по значению его индекса, необходимо также иметь возможность узнать индекс любого элемента набора.

Решение

Для решения этой задачи применим методы `eq(index)` и `index(element)` (листинг 10.2.3).

Листинг 10.2.3. Использование методов `eq()` и `index()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-2-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<style type="text/css">
div {
  width:60px; height:60px;
  margin:5px; float:left;
  border:1px solid #369;
}
br { clear:left; }
</style>
<script type="text/javascript">
$(function(){
  $("div").eq(2).css("background-color", "#f00");
  $("div").click(function(){
    alert("Индекс элемента в наборе: " + $("div").index(this));
  });
});
</script>
</head>
<body>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
<div></div>
</body>
</html>
```

Обсуждение

HTML-разметка, показанная в листинге 10.2.3, — это просто семь элементов `div`, которые оформлены с помощью CSS-свойств и имеют размеры, рамку и белый цвет фона, установленный по умолчанию. Больше ничего интересного нет, поэтому перейдем к рассмотрению JavaScript-кода.

При готовности DOM выбираем все элементы `div`, которые имеются на странице, и применяем к этому набору метод `eq()`, передавая в качестве аргумента число 2. После этого в наборе останется единственный элемент `div`, тот самый, имевший индекс 2 в исходном наборе, или просто третий по счету (отсчет начинается от нуля).

Чтобы иметь возможность узнать индекс любого элемента `div` при щелчке на нем мышью, свяжем с набором этих элементов обработчик события `click`. Функция-обработчик этого события будет выводить в окне предупреждения индекс соответствующего элемента, полученный с помощью метода `index()`, и поясняющий текст.

ЗАДАЧА

Необходимо получить элементы DOM из набора элементов jQuery.

Решение

Задача решается с помощью метода `get([index])` (листинг 10.2.4).

Листинг 10.2.4. Использование метода `get()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-2-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("li").click(function(){
    var elem = $(this).get(0);
    alert(elem + ', id=' + elem.id + ' и текст: ' + elem.innerHTML);
  });
});
</script>
</head>
<body>
<ul>
  <li id="One">Один</li>
  <li id="Two">Два</li>
  <li id="Three">Три</li>
  <li id="Four">Четыре</li>
</ul>
</body>
</html>
```

Обсуждение

В листинге 10.2.4 имеется элемент `ul`, содержащий четыре пункта списка, у каждого из которых присутствует атрибут `id` и текст внутри элемента.

При щелчке на элементе `li` будем применять к "кликнутому" элементу метод `get()` с передачей в качестве аргумента числа "0" (поскольку в текущем наборе это вооб-

ще единственный элемент). Выводим информацию в окне предупреждения и видим примерно следующее: `[object HTMLLIElement], id=Four и текст: Четыре`. Нам действительно удалось получить элемент DOM.

10.3. Другие полезные вспомогательные функции

ЗАДАЧА

Необходимо получить текущее время.

Решение

Для решения этой простой задачи воспользуемся вспомогательной функцией `$.now()`, которая была добавлена в библиотеку, начиная с версии 1.4.3 (листинг 10.3.1).

Листинг 10.3.1. Использование вспомогательной функции `$.now()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  $("button").click(function(){
       alert($.now());
  });
});
</script>
</head>
<body>
<button>.now()</button>
</body>
</html>
```

Обсуждение

Строго говоря, в коде, который приведен в листинге 10.3.1, при нажатии кнопки **.now()** мы получаем не текущее время, а число миллисекунд, прошедших с полуночи 1 января 1970 года GMT. То же самое значение возвращает метод `getTime` объекта `Date` в "чистом" JavaScript.

ЗАДАЧА

Необходимо конвертировать строку, представленную в формате JSON, в полноценный JavaScript-объект.

Решение

Для решения используем вспомогательную функцию `$.parseJSON(json)` (листинг 10.3.2).

Листинг 10.3.2. Использование вспомогательной функции `$.parseJSON()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var row = '{"name":"John","phone":"555-66-77","age":"45"}';
  $("button").click(function(){
       var obj = $.parseJSON(row);
       alert("Имя: " + obj.name + "\nВозраст: " + obj.age + "\nТелефон: " +
obj.phone);
  });
});
</script>
</head>
<body>
<button>JSON</button>
</body>
</html>
```

Обсуждение

В HTML-разметке из листинга 10.3.2 — одна кнопка. При нажатии на нее будем производить конвертировать строку, записанную в формате JSON, в полноценный объект. А для проверки попробуем получить значения свойств этого объекта.

Итак, исходную строку сохраняем в переменной `row`.

При нажатии на кнопку **JSON** передаем строку в качестве аргумента во вспомогательную функцию `$.parseJSON()` и сохраняем результат в переменной `obj`.

А затем обращаемся к свойствам получившегося объекта по их именам, и полученные значения выводим в окне предупреждения.

ЗАДАЧА

Необходимо конвертировать строку в XML-документ и проверить корректность выполненного преобразования.

Решение

Для решения подойдут вспомогательные функции `$.parseXML(data)` и `$.isXMLDoc(node)` (листинг 10.3.3).

Листинг 10.3.3. Использование вспомогательных функций `$.parseXML()` и `$.isXMLDoc()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var row = '<rss version="2.0"><channel><title>RSS
Title</title></channel></rss>';
  $("button").click(function(){
       var xml = $.parseXML(row);
       $.isXMLDoc(xml) ? alert('TRUE') : alert('FALSE');
  });
});
</script>
</head>
<body>
<button>XML</button>
</body>
</html>
```

Обсуждение

В HTML-разметке из листинга 10.3.3 присутствует всего одна кнопка, при нажатии на которую будем преобразовывать полученную строку в XML-документ и проверять, действительно ли получился требуемый документ.

В JavaScript-коде, в переменной `row` содержится строка, представляющая собой содержимое для создания XML-документа.

При нажатии на кнопку **XML** мы передаем эту строку во вспомогательную функцию `$.parseXML()`, а результат сохраняем в переменной `xml`.

С помощью вспомогательной функции `$.isXMLDoc()` проверяем переменную `xml`. Если мы действительно сформировали XML-документ, с содержимым которого

можно работать с помощью jQuery, получим окно предупреждения с сообщением "TRUE", иначе — сообщение "FALSE".

ЗАДАЧА

Необходимо одну и ту же функцию выполнить в разных контекстах.

Решение

В составе библиотеки jQuery есть замечательная вспомогательная функция, которая подойдет для решения такой задачи, — `$.proxy(function, context)` или `$.proxy(context, name)`. Она может принимать аргументы в двух вариантах — функция и контекст ее выполнения или контекст выполнения и имя функции. А возможности иллюстрирует пример, код которого приведен в листинге 10.3.4.

Листинг 10.3.4. Использование вспомогательной функции `$.proxy()`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-10-3-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function () {
  var obj1 = {
    name: "John",
    surname: "Resig",
    test: function() {
      $("#log").append('<p>' + this.name + ' ' + this.surname + '</p>');
    }
  };
   var obj2 = {
    name: "Yehuda",
    surname: "Katz"
  };
  var obj3 = {
    name: "Jörn",
    surname: "Zaefferer",
    test3: function() {
      $("#log").append('<p>' + this.surname + ', ' + this.name + '</p>');
    }
  };
  $("#test1").click($.proxy(obj1.test, obj1));
  $("#test2").click($.proxy(obj1.test, obj2));
  $("#test3").click($.proxy(obj3, 'test3'));
});
```

```
</script>
</head>
<body>
<div>
  <button id="test1">Объект 1</button>
  <button id="test2">Объект 2</button>
  <button id="test3">Объект 3</button>
</div>
<div id="log"></div>
</body>
</html>
```

Обсуждение

Сначала разберем простейшую HTML-разметку, приведенную в листинге 10.3.4. Три кнопки, при нажатии на которые будем выполнять некоторые действия, и элемент `div` с идентификатором `log` — сюда будем добавлять результаты.

Теперь изучим JavaScript-код. В начале создаем три объекта. Каждый из них содержит свойства `name` и `surname` с разными значениями. Кроме этого первый и третий объекты имеют свойства `test` и `test3`, которые являются функциями.

Функция `test` добавляет значения свойств `name` и `surname` в элемент `p` и вставляет его в элемент с идентификатором `log`. Функция `test3` выполняет похожие действия, только вставляет значения свойств `name` и `surname` в элемент `p` в другом порядке и через запятую.

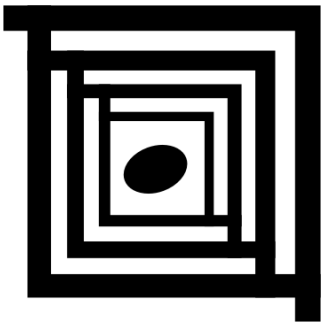
Далее следуют три обработчика для трех кнопок — **Объект 1**, **Объект 2**, **Объект 3**.

На этом подготовительная часть закончена и дальше мы попробуем разобраться, что же такого интересного есть в этом примере.

Итак, нажимаем кнопку **Объект 1** — передаем вспомогательной функции `$.proxy()` функцию `test`, которую необходимо выполнить, и контекст ее выполнения — объект `obj1`. Получаем результат — "John Resig". Следовательно, функция, определенная как свойство объекта `obj1`, выполнилась в контексте этого объекта.

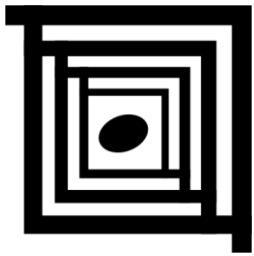
Нажмем кнопку **Объект 3** — вспомогательной функции `$.proxy()` будут переданы в качестве аргументов контекст выполнения `obj3` и строка с именем функции `test3`. Результат — "Zaefferer, Jörn". Следовательно, функция, определенная как свойство объекта `obj3`, выполнилась в контексте этого объекта. Просто мы передавали ей аргументы в другом виде.

А теперь нажимаем кнопку **Объект 2**, чтобы заставить функцию `test`, определенную в объекте `obj1`, выполниться в контексте другого объекта — `obj2`. Ожидаем результат "Yehuda Katz" и действительно его получаем!



ЧАСТЬ II

Расширения для библиотеки jQuery



ГЛАВА 11

Меню для веб-сайта

В Интернете можно встретить самые разные сайты, ведь фантазия разработчиков безгранична, как и сама Всемирная паутина. Но и миниатюрные сайты-визитки, и огромные порталы объединяет одно обстоятельство — их невозможно представить без системы навигации или, говоря по-простому, без меню. Хорошее и понятное меню побудит посетителя посмотреть много страниц сайта. Неудобное или запутанное меню может привести к тому, что пользователь просто закроет сайт и больше туда не вернется.

11.1. Плагин jQuery Superfish

ЗАДАЧА

На веб-странице необходимо организовать многоуровневое меню, раскрывающееся при наведении указателя мыши. При этом все пункты меню должны быть доступными для индексации роботами поисковых систем. И конечно, страница должна соответствовать стандартам W3C.

Решение

Для решения этой задачи используем плагин Superfish, разработанный для библиотеки jQuery (листинг 11.1.1).

Листинг 11.1.1. Использование плагина jQuery Superfish

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-11-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/superfish.css" />
```

```
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/hoverIntent.js"></script>
<script type="text/javascript" src="js/superfish.js"></script>
<script type="text/javascript">
$(function(){
  $("ul.sf-menu").superfish();
});
</script>
</head>
<body>
<ul class="sf-menu">
  <li class="current">
    <a href="#a">Пункт меню</a>
    <ul class="A">
      <li><a href="#aa">Пункт меню, где написано много</a></li>
      <li class="current">
        <a href="#ab">Пункт меню</a>
        <ul class="AA">
          <li class="current"><a href="#">Пункт меню</a></li>
          <li><a href="#aba">Пункт меню</a></li>
          <li><a href="#abb">Пункт меню</a></li>
          <li><a href="#abc">Пункт меню</a></li>
          <li><a href="#abd">Пункт меню</a></li>
        </ul>
      </li>
      <li>
        <a href="#">Пункт меню</a>
        <ul class="AB">
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
        </ul>
      </li>
      <li>
        <a href="#">Пункт меню</a>
        <ul class="AC">
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
        </ul>
      </li>
    </ul>
  </li>
```

```
<li><a href="#">Пункт меню</a></li>
<li>
  <a href="#">Пункт меню</a>
  <ul class="B">
    <li>
      <a href="#">Пункт меню</a>
      <ul class="BA">
        <li><a href="#">Мало</a></li>
        <li><a href="#">Мало</a></li>
        <li><a href="#">Мало</a></li>
        <li><a href="#">Мало</a></li>
        <li><a href="#">Мало</a></li>
      </ul>
    </li>
    <li>
      <a href="#">Пункт меню</a>
      <ul class="BB">
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
      </ul>
    </li>
    <li>
      <a href="#">Пункт меню</a>
      <ul class="BC">
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
      </ul>
    </li>
    <li>
      <a href="#">Пункт меню</a>
      <ul class="BD">
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
        <li><a href="#">Пункт меню</a></li>
      </ul>
    </li>
    <li>
      <a href="#">Пункт меню</a>
      <ul class="BE">
        <li><a href="#">Пункт меню</a></li>
```

```
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
          <li><a href="#">Пункт меню</a></li>
        </ul>
      </li>
    </ul>
  </li>
  <li><a href="#">Пункт меню</a></li>
</ul>
</body>
</html>
```

Обсуждение

Рассмотрим пример, приведенный в листинге 11.1.1, и прежде всего разберемся, какие файлы необходимо указать в разделе `head` веб-страницы. Сначала подключен css/superfish.css — файл стилевого оформления меню, затем файл библиотеки jQuery js/jquery-1.5.2.min.js, а также файл плагина js/superfish.js.

Файл js/hoverIntent.js подключать необязательно, но если он указан, плагин Superfish определит его присутствие и будет использовать его возможности.

> ***Примечание***
>
> hoverIntent — плагин, который несколько замедляет вызов событий при наведении указателя мыши на элемент или выходе за его пределы, чтобы предотвратить случайный вызов каких-либо функций.

Теперь обратим свое внимание на HTML-разметку. По ней несложно понять, что многоуровневое меню состоит из вложенных друг в друга ненумерованных списков.

Все, что осталось проделать, — написать буквально одну строку JavaScript-кода: выбрать корневой элемент `ul` нашего меню по значению `sf-menu` его атрибута `class` и связать с ним метод `superfish()`.

Полученный результат можно увидеть на рис. 11.1.

Продолжим знакомство с плагином Superfish и попробуем передать ему некоторые пользовательские настройки, которые он умеет принимать.

Листинг 11.1.2. Использование плагина jQuery Superfish

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-11-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/superfish.css" />
```

```
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/hoverIntent.js"></script>
<script type="text/javascript" src="js/superfish.js"></script>
<script type="text/javascript">
$(function(){
  $("ul.sf-menu").superfish({
    animation: { height: "show" },
    delay: 1200,
    onShow: function(){ alert($(this).attr("class")); }
  });
});
</script>
</head>
<body>
<ul class="sf-menu">
. . . . . . . . . .
</ul>
</body>
</html>
```

В листинге 11.1.2 мы опустили практически всю HTML-разметку, поскольку она не изменилась. И подключаемые файлы остались прежними. Изменения коснулись только JavaScript-кода.

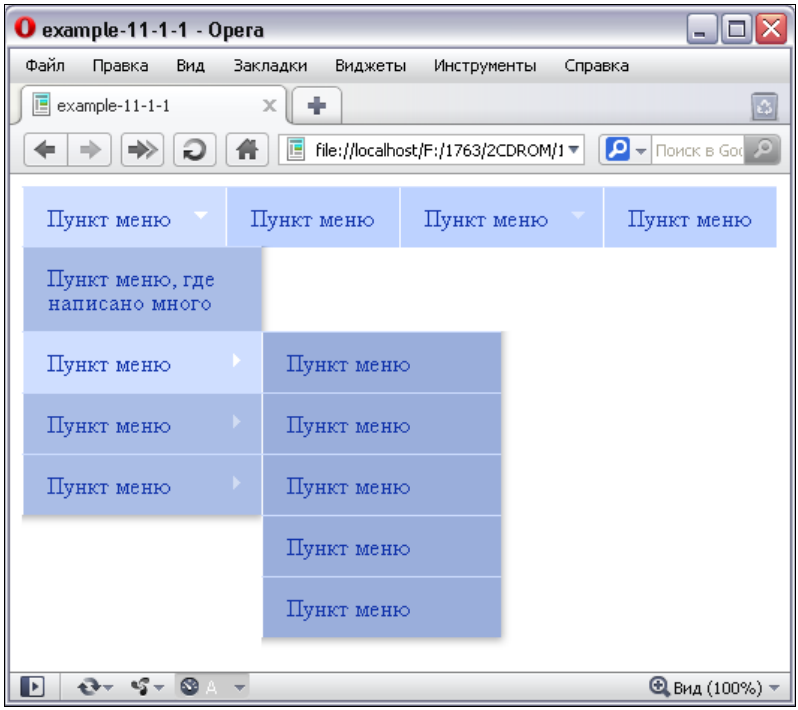


Рис. 11.1. Меню Superfish — горизонтальное расположение

Посмотрите, теперь мы не только выбрали необходимый элемент `ul` и связали с ним метод `superfish()`, но и передали ему в качестве аргумента объект с некоторыми настройками.

В свойстве `animation` передаем объект с настройками для анимации, выполняемой при открытии уровней меню. Это свойство может принимать объект с любыми настройками, которые "понимает" метод `animate()` самой библиотеки jQuery.

В свойстве `delay` передаем задержку 1,2 с, т. е. вложенные уровни меню будут закрываться с этой задержкой.

Для свойства `onShow` в качестве значения выступает пользовательская функция, которая вызывается сразу после того, как открыт очередной уровень меню (т. е. сразу по завершении анимации, если она используется). Внутри функции объект `$(this)` представляет тот элемент `ul`, пункты списка `li` которого были открыты. Если вы испытаете пример, находящийся на прилагаемом к книге компакт-диске, то сможете убедиться, что в окне предупреждения действительно выводится значение атрибута `class` этого элемента.

Остальные опции используются аналогично, а полный их список с описаниями приведен в табл. 11.1.

Таблица 11.1. *Опции плагина Superfish*

Опция	Описание
`hoverClass`	Имя класса, который применяется к пункту меню, когда над ним находится указатель мыши. Значение по умолчанию — `sfHover`
`pathClass`	Имя класса, который применяется к пункту списка, указывающему на текущую страницу
`delay`	Задержка закрытия меню после выхода указателя мыши за его пределы. Значение по умолчанию — 800 мс
`animation`	Принимает объект с настройками. Идентичен объекту, который может быть передан первым аргументом методу `animate()` библиотеки jQuery. Значение по умолчанию — `{opacity:'show'}`
`speed`	Время выполнения анимации. Указывается аналогично второму аргументу метода `animate()` библиотеки jQuery. Значение по умолчанию — `'normal'`
`autoArrows`	Значение по умолчанию — `true`. Стрелки автоматически генерируются в разметке
`dropShadows`	Значение по умолчанию — `true`. Чтобы группы пунктов подменю не отбрасывали тень, необходимо установить значение `false`
`disableHI`	Значение по умолчанию — `false`. Установка значения `true` приведет к тому, что плагин hoverIntent.js (если подключен) не будет обнаруживаться и его возможности не будут использоваться
`onInit`	В этой опции может быть определена пользовательская функция, которая будет вызвана при инициализации меню
`onBeforeShow`	В этой опции может быть определена пользовательская функция, которая будет вызвана непосредственно перед открытием очередного уровня меню

***Таблица 11.1** (окончание)*

Опция	Описание
`onShow`	В этой опции может быть определена пользовательская функция, которая будет вызвана сразу после открытия очередного уровня подменю (после завершения анимации, если используется)
`onHide`	В этой опции может быть определена пользовательская функция, которая будет вызвана сразу после закрытия очередного уровня подменю (после завершения анимации, если используется)

До сих пор мы рассматривали возможности плагина на примере горизонтального меню. А если нам понадобится меню вертикальное?

Оставим для примера ту же самую HTML-разметку и посмотрим, как этот вопрос можно решить с помощью плагина Superfish (листинг 11.1.3).

Листинг 11.1.3. Использование плагина jQuery Superfish

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-11-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/superfish.css" />
<link rel="stylesheet" type="text/css" href="css/superfish-vertical.css" />
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/hoverIntent.js"></script>
<script type="text/javascript" src="js/superfish.js"></script>
<script type="text/javascript">
$(function(){
  $("ul.sf-menu").superfish({
    delay: 1200
  });
});
</script>
</head>
<body>
<ul class="sf-menu sf-vertical">
. . . . . .
</ul>
</body>
</html>
```

В примере из листинга 11.1.3 мы подключили дополнительный стилевой файл css/superfish-vertical.css — он отвечает за вертикальную ориентацию меню. В HTML-разметке изменения минимальны — корневому элементу `ul` нашего меню

добавлен дополнительный класс `sf-vertical`. А в JavaScript-коде вообще никаких изменений не потребовалось!

На рис. 11.2 показано, как будет выглядеть развернутое многоуровневое меню при вертикальном расположении.

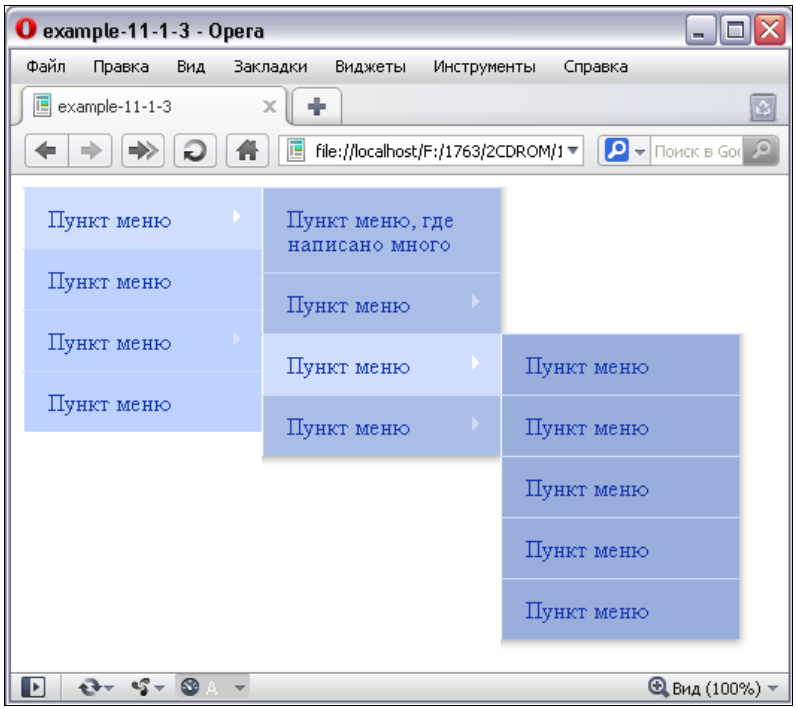


Рис. 11.2. Результат использования плагина jQuery Superfish

И наконец, познакомимся с еще одной возможностью плагина — попробуем организовать меню в виде навигационной панели. При этом опять HTML-разметка останется практически без изменений (листинг 11.1.4).

Листинг 11.1.4. Использование плагина jQuery Superfish

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-11-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/superfish.css" />
<link rel="stylesheet" type="text/css" href="css/superfish-navbar.css" />
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/hoverIntent.js"></script>
<script type="text/javascript" src="js/superfish.js"></script>
```

```
<script type="text/javascript">
$(function(){
  $("ul.sf-menu").superfish();
});
</script>
</head>
<body>
<ul class="sf-menu sf-navbar">
. . . . . . . . . .
</ul>
</body>
</html>
```

Из примера, приведенного в листинге 11.1.4, ясно, что для создания меню в виде навигационной панели потребовалось только подключить стилевой файл css/superfish-navbar.css и добавить имя класса `sf-navbar` корневому элементу `ul` меню. В JavaScript-коде вновь никаких изменений.

Результат можно увидеть на рис. 11.3.

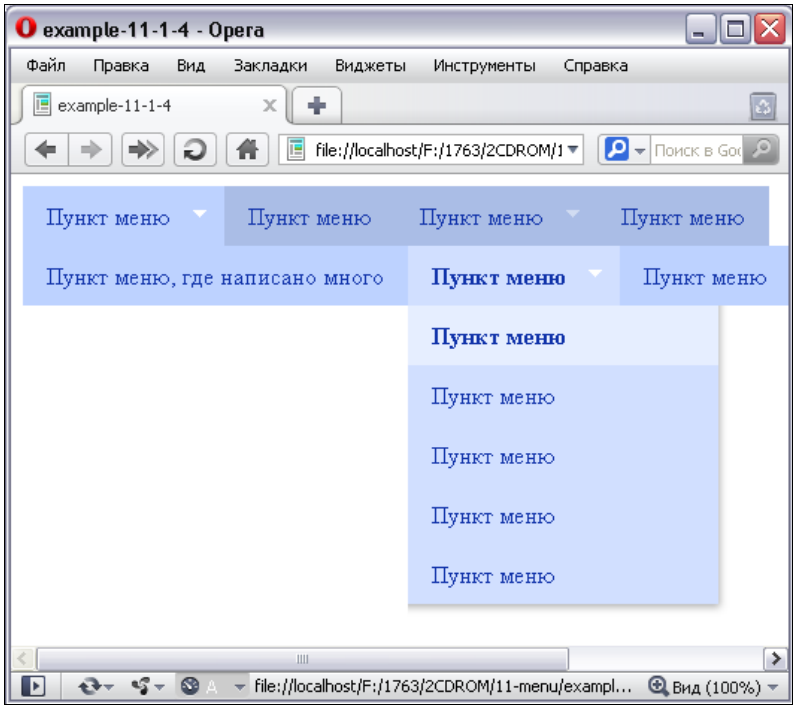
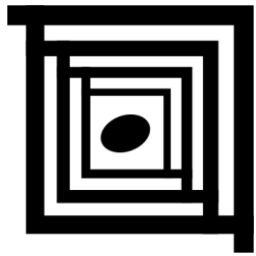


Рис. 11.3. Результат использования плагина jQuery Superfish

Найти оригинальное описание плагина Superfish в Интернете можно по адресу **http://users.tpg.com.au/j_birch/plugins/superfish/**.



ГЛАВА 12

Работа с таблицами

12.1. Плагин jqGrid

ЗАДАЧА

Посетителям веб-страницы нужно обеспечить возможность комфортной работы с данными, представленными в табличной форме. Необходимо реализовать сортировку данных по любому столбцу таблицы, изменение числа строк таблицы, выводимых на одной странице, постраничную навигацию с сохранением порядка сортировки, а также поиск требуемых данных в таблице.

Решение

Для решения такой многогранной задачи хорошо подойдет плагин jqGrid. Чтобы рассказать про все возможности этого плагина, придется написать отдельную книгу. Поэтому мы будем рассматривать только самые основные варианты его применения. В листинге 12.1.1 приведен пример наполнения таблицы локальными данными.

Листинг 12.1.1. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-12-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
```

```
<style type="text/css">
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
</style>
<script type="text/javascript">
$(function(){
  $("#example").jqGrid({
    datatype: "local",
    height: 150,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', width:50, sorttype:"int",resizable:false},
      {name:'country_code', width:75, sorttype:"text"},
      {name:'region_code', width:75, sortable:false},
      {name:'city', width:300, align:"left", sorttype:"text"},
      {name:'latitude', width:100, align:"center", sorttype:"float"},
      {name:'longitude', width:100, align:"right", sorttype:"float"}
    ],
    multiselect: true,
    caption: "Наполнение таблицы локальными данными"
  });
  var mydata = [
    {id:"8039", country_code:"CH", region_code:"19", city:"Aadorf",
latitude:"47.4833", longitude:"8.9"},
    {id:"4286", country_code:"BE", region_code:"08", city:"Aalst",
latitude:"50.9333", longitude:"4.0333"},
    {id:"4287", country_code:"BE", region_code:"08", city:"Aalter",
latitude:"51.0833", longitude:"3.45"},
    {id:"7405", country_code:"CH", region_code:"01", city:"Aarau",
latitude:"47.3833", longitude:"8.05"},
    {id:"7936", country_code:"CH", region_code:"14", city:"Aarberg",
latitude:"47.05", longitude:"7.2833"},
    {id:"7406", country_code:"CH", region_code:"01", city:"Aarburg",
latitude:"47.3167", longitude:"7.9"},
    {id:"3564", country_code:"BE", region_code:"02", city:"Aarschot",
latitude:"50.9833", longitude:"4.8333"},
    {id:"4437", country_code:"BE", region_code:"09", city:"Aartrijke",
latitude:"51.1167", longitude:"3.0833"},
    {id:"3443", country_code:"BE", region_code:"01", city:"Aartselaar",
latitude:"51.1333", longitude:"4.3833"},
    {id:"7407", country_code:"CH", region_code:"01", city:"Aarwangen",
latitude:"47.2333", longitude:"7.75"}
  ];
  $.each(mydata, function(i,val){
    $("#example").addRowData(i+1,val);
  });
});
```

```
</script>
</head>
<body>
<table id="example"></table>
</body>
</html>
```

Обсуждение

Для того чтобы иметь возможность применять плагин jqGrid на веб-странице, необходимо подключить к ней файл библиотеки js/jquery-1.5.2.min.js, файл самого плагина — js/jquery.jqGrid.min.js и файл локализации js/i18n/grid.locale-ru.js. Стилевое оформление задается в подключаемых файлах css/ui.jqgrid.css и css/redmond/jquery-ui-1.8.11.custom.css — плагин поддерживает темы оформления UI jQuery.

HTML-разметка, показанная в листинге 12.1.1, до смешного проста — открывающий и закрывающий теги `table`. Открывающий тег имеет идентификатор `example`. И это абсолютно вся разметка!

Самое интересное ждет нас при изучении JavaScript-кода. Не стоит пугаться его внушительного объема — на самом деле не все так сложно.

Сначала с помощью селектора выбираем необходимый элемент по его идентификатору и применяем к нему плагин jqGrid. А дальше передаем плагину объект с настройками, которые рассмотрим несколько подробнее.

Свойство `datatype`, определяющее тип информации, в нашем примере содержит строку `local` — мы будем наполнять таблицу локальными данными. Другие возможные варианты этого свойства — `xml`, `xmlstring`, `json`, `jsonstring`, `javascript` и `function`.

В свойстве `height` передаем число `150`. Так задается высота области, в которой будут отображаться строки таблицы. Если строк окажется больше, чем можно отобразить в этой области, справа появится полоса прокрутки. Похожим образом можно установить и ширину таблицы. Но мы выберем другой вариант, и несколько позже попробуем установить ширину для каждой ячейки в строке.

В свойстве `colNames` описываем массив, содержащий заголовки колонок таблицы.

В свойстве `colModel` тоже массив, но каждый его элемент — объект, свойства которого описывают модели ячеек соответствующего столбца таблицы.

Здесь свойство `name` является обязательным и задает уникальное имя колонки в таблице. С помощью свойства `width` можно задать ширину колонки, а свойство `align` поможет выровнять содержимое ячеек.

Щелкая указателем мыши по заголовку колонки, можно сортировать данные. В свойстве `sorttype` указывается строка, которая определяет, как именно будут сортироваться данные. Например, значения `int` или `integer` указывают при сортировке данных, как целых чисел. Значения `float/number/currency` — данные сортируются как десятичные числа. Другие возможные значения — `data`, `text`. В этом

свойстве можно определить даже пользовательскую функцию для сортировки. Нужно отметить, что свойство `sorttype` служит только для работы с локальными данными.

Кстати, сортировку по отдельным колонкам таблицы можно запретить, передав в настройках свойство `sortable` со значением `false`. В примере из листинга 12.1.1 не сортируется колонка **Код региона**.

Установив указатель на границу колонки в заголовке таблицы, нажав и удерживая левую кнопку мыши и перемещая указатель, можно менять ширину колонок. Впрочем, при необходимости, эту возможность можно запретить, передав в настройках свойство `resizable` со значением `false`. В примере из листинга 12.1.1 колонка **Id** не имеет возможности изменения ширины.

Следующее далее по коду свойство — `multiselect`. Передав здесь значение `true`, разрешаем выделение одновременно более одной строки. К тому же слева добавляется столбец с элементами `checkbox` для каждой строки. В заголовке таблицы появляется общий `checkbox`.

Последнее свойство, с которым мы познакомимся на примере из листинга 12.1.1, — свойство `caption`. Тут все должно быть ясно сразу — заголовок, отображаемый сверху таблицы.

Далее нам встречается массив `mydata` — это те данные, которыми будет наполнена таблица. Каждый элемент массива — объект, названия свойств которого соответствуют значениям `name`, заданным в модели колонок `colModel`.

И наконец, следующие три строки кода — это собственно заполнение таблицы содержимым. Для этой операции мы используем вспомогательную функцию `$.each` библиотеки jQuery. Напомню, что она позволяет совершать обход массивов (или объектов) с применением некоторой функции для каждого элемента. В нашем случае мы вызываем метод `addRowData` для каждого элемента приготовленного массива и передаем методу индекс элемента +1 и его значение.

Результат наших усилий можно увидеть на рис. 12.1.

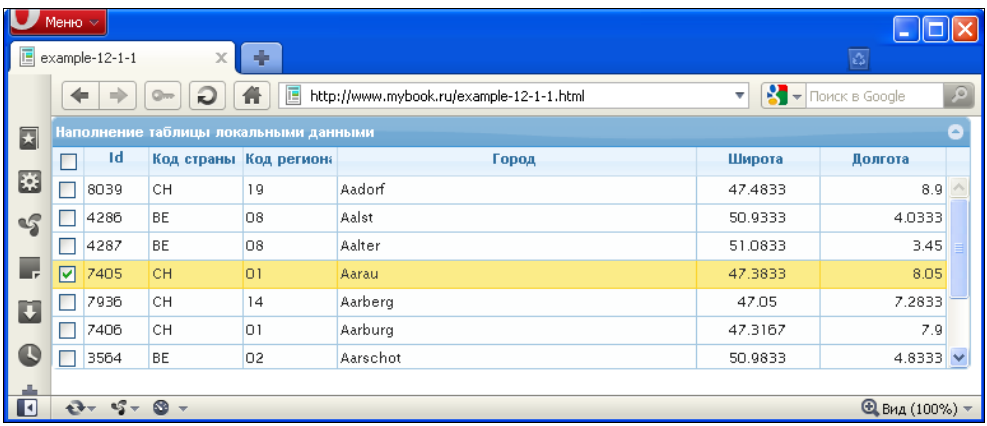


Рис. 12.1. Использование плагина jqGrid

Продолжим изучение возможностей плагина jqGrid при работе с локальными данными и попробуем выполнять над ними некоторые манипуляции. Например, узнаем, как можно получать данные из ячеек выбранной строки, как добавлять, редактировать и удалять строки таблицы. Пример, демонстрирующий эти возможности, приведен в листинге 12.1.2.

Листинг 12.1.2. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-12-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
<style type="text/css">
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
p { font-size: 2em; }
p a { margin:0 5px; }
</style>
<script type="text/javascript">
$(function(){
  $("#example").jqGrid({
    datatype: "local",
    height: 250,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', width:50, sorttype:"int",resizable:false},
      {name:'country_code', width:75, sorttype:"text"},
      {name:'region_code', width:75, sortable:false},
      {name:'city', width:300, align:"left", sorttype:"text"},
      {name:'latitude', width:100, align:"center", sorttype:"float"},
      {name:'longitude', width:100, align:"right", sorttype:"float"}
    ],
    caption: "Работа с данными, представленными в таблице"
  });
  var mydata = [ … ];
  $.each(mydata, function(i,val){
```

```
    $("#example").addRowData(i+1,val);
  });

  $("p a:first").click(function(){
    var id = $("#example").getGridParam('selrow');
    if(id) {
      var row = $("#example").getRowData(id);
      alert(row.city + ' [' + row.latitude + ', ' + row.longitude + ']');
    } else {
      alert("Нужно выбрать строку!");
    }
  }).next().click(function(){
    var datarow = {id:"9999", country_code:"RU", region_code:"643",
city:"Moscow", latitude:"55.755786121111", longitude:"37.617633343333"};
    var add = $("#example").addRowData(9999,datarow);
    if(add) { alert("Строка добавлена!"); }
  }).next().click(function(){
    var id = $("#example").getGridParam('selrow');
    if(id) {
      var upd = $("#example").setRowData(id, {city:"North Pole",
latitude:"90.0000", longitude:"-", region_code:"<img src='user.gif'/>"});
      if(upd) { alert("Строка " + id + " обновлена!"); }
    } else {
      alert("Нужно выбрать строку!");
    }
  }).next().click(function(){
    var id = $("#example").getGridParam('selrow');
    if(id) {
      var del = $("#example").delRowData(id);
      if(del) { alert("Строка " + id + " удалена!"); }
    } else {
      alert("Нужно выбрать строку!");
    }
  });
});
</script>
</head>
<body>
<table id="example"></table>
<p><a href="#">Получить данные</a> <a href="#">Добавить строку</a> 
<a href="#">Редактировать строку</a> <a href="#">Удалить строку</a></p>
</body>
</html>
```

В HTML-разметке, приведенной в листинге 12.1.2, добавился элемент `p`, внутри которого есть четыре ссылки. При нажатии на них мы будем выполнять соответствующие действия.

Настройки, передаваемые плагину при инициализации, изменились незначительно. Массив данных mydata остался прежним, поэтому его элементы не показаны в листинге. Процедура наполнения таблицы данными также не претерпела изменений.

А вот следующий далее JavaScript-код пока нам не знаком — займемся его подробным рассмотрением.

Сначала мы выбираем первый элемент a внутри параграфа p и связываем с ним обработчик события click. Здесь мы применяем к таблице метод getGridParam, которому передаем аргумент selrow. Аргумент selrow содержит идентификатор последней выбранной строки, и мы имеем возможность сохранить его в переменной id. Если эта операция прошла удачно, то мы вызовем метод getRowData и передадим ему переменную id. Этот метод возвращает объект, названия свойств которого соответствуют названиям, определенным в colModel, а значения являются данными, содержащимися в ячейках выбранной строки. Таким образом, сохранив объект в переменной row, мы получаем возможность обратиться к любому его свойству, т. е. другими словами, — получить любые значения, содержащиеся в ячейках выбранной строки.

Двигаемся дальше, добавляем в цепочку вызовов метод next, который позволяет нам переместиться к следующему элементу a, а затем также назначить ему обработчик события click.

В этом обработчике попробуем добавить еще одну строку в таблицу. Сначала приготовим данные: в переменной datarow сохраним объект, свойства которого нам уже хорошо знакомы — это "имена" ячеек. Каждому свойству зададим какие-либо значения. Осталось применить к таблице метод addRowData, которому в качестве аргументов передать значение идентификатора и приготовленный ранее объект datarow.

Следующий next в цепочке — и мы перемещаемся к третьему элементу a. Обработчик события click для него начинается с получения значения идентификатора выбранной строки с помощью метода getGridParam, так же, как и в первом случае. Если идентификатор получен, применяем метод setRowData, которому в первом параметре передаем значение идентификатора строки, а во втором — объект с "именами" ячеек и их новым содержимым.

Последний вызов метода next — получаем четвертый элемент a, и, связав с ним обработчик click, пробуем удалить выбранную строку. Сначала опять получаем значение идентификатора строки с помощью getGridParam, а затем применяем метод delRowData. Аргументом метода является полученный идентификатор.

До сих пор мы изучали возможности плагина jqGrid, используя локальные данные. Перейдем к более серьезным примерам — попробуем загружать данные с сервера в формате JSON, увеличим объемы обрабатываемых данных и начнем знакомиться с навигационной панелью плагина.

Листинг 12.1.3. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-12-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
<style type="text/css">
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
</style>
<script type="text/javascript">
$(function(){
  $("#example").jqGrid({
    url: "datafromserver.php",
    datatype: "json",
    height: 250,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', index:'id', width:50, sorttype:"int"},
      {name:'country_code', index:'country_code', width:75, sorttype:"text"},
      {name:'region_code', index:'region_code', width:75, sorttype:false},
      {name:'city', index:'city', width:300, align:"right", sorttype:"text"},
      {name:'latitude', index:'latitude', width:100, align:"right",
sorttype:"float"},
      {name:'longitude', index:'longitude', width:100, align:"right",
sorttype:"float"}
    ],
    rowNum:10,
    rowList:[10,20,30],
    pager: "#pager",
    sortname: "city",
    sortorder: "desc",
    viewrecords: true,
    caption: "Загрузка данных в таблицу с использованием JSON"
  });
  $("#example").jqGrid('navGrid','#pager',{edit:false, add:false, del:false,
search:false});

});
</script>
</head>
<body>
```

```
<table id="example"></table>
<div id="pager"></div>
</body>
</html>
```

В примере, приведенном в листинге 12.1.3, несколько изменилась HTML-разметка. Помимо пары тегов `table`, появился элемент `div` с идентификатором `pager` — он послужит для создания навигационной панели.

Что касается JavaScript-кода, то в его начале мы, так же как и раньше, выбираем таблицу по идентификатору и связываем с ней плагин jqGrid. Но поскольку теперь мы будем оперировать данными, полученными с сервера, то изменятся и некоторые настройки. Мы будем разбирать те из них, которые нам не были знакомы до сих пор.

И первое же свойство в объекте настроек — `url` — нам пока не встречалось. Но ничего непонятного тут нет. В этом свойстве указывается адрес файла, к которому следует обращаться за получением данных.

Свойство `datatype` нам уже встречалось. Но если при работе с локальными данными мы указывали в нем `local`, то теперь задали значение `json` — мы ожидаем получить от сервера данные именно в этом формате.

Свойства `height` и `colNames` пропускаем, а вот в `colModel` произошло небольшое изменение. В объектах, описывающих настройки колонок таблицы, появилось свойство `index`. Значение, определенное в этом свойстве, передается на сервер в параметре `sidx` запроса (с самим запросом мы познакомимся несколько позже) и служит для указания имени поля базы данных, по которому должна производиться сортировка.

Перейдем к следующим незнакомым свойствам — `rowNum` и `rowList`. В первом указывается число, определяющее, сколько строк таблицы будет отображаться на одной странице. Во втором можно задать массив, элементы которого сформируют выпадающий список. С помощью этого списка можно изменять число строк таблицы, выводимых на одной странице.

В свойстве `pager` указывается идентификатор элемента, в котором будет построена навигационная панель плагина.

Свойство `sortname` содержит "название" колонки таблицы, по которому должна осуществляться сортировка при инициализации плагина, а свойство `sortorder` — направление сортировки. Значение свойства `sortorder` передается в запросе к серверу в параметре `sord`.

Указав в свойстве `viewrecords` значение `true`, отображаем в правом нижнем углу, в навигационной панели, информацию об общем числе записей в таблице и о номерах строк, отображаемых в текущий момент.

Нам осталось разобраться всего в одной строке JavaScript-кода, в той самой, которая отвечает за формирование элементов управления в навигационной панели. Мы вновь выбираем таблицу по идентификатору, применяем к ней метод `jqGrid` и пе-

редаем ему три аргумента. Первый аргумент — строка с названием метода `navGrid`, который необходимо вызвать, второй — строка с идентификатором элемента `pager`, который будет служить навигационной панелью, и третий — объект, с незнакомыми пока свойствами `edit`, `del`, `add` и `search`. Знакомство с ними нам еще предстоит, а пока ограничимся тем, что всем четырем свойствам присвоим значение `false`.

Внешний вид таблицы с навигационной панелью показан на рис. 12.2.

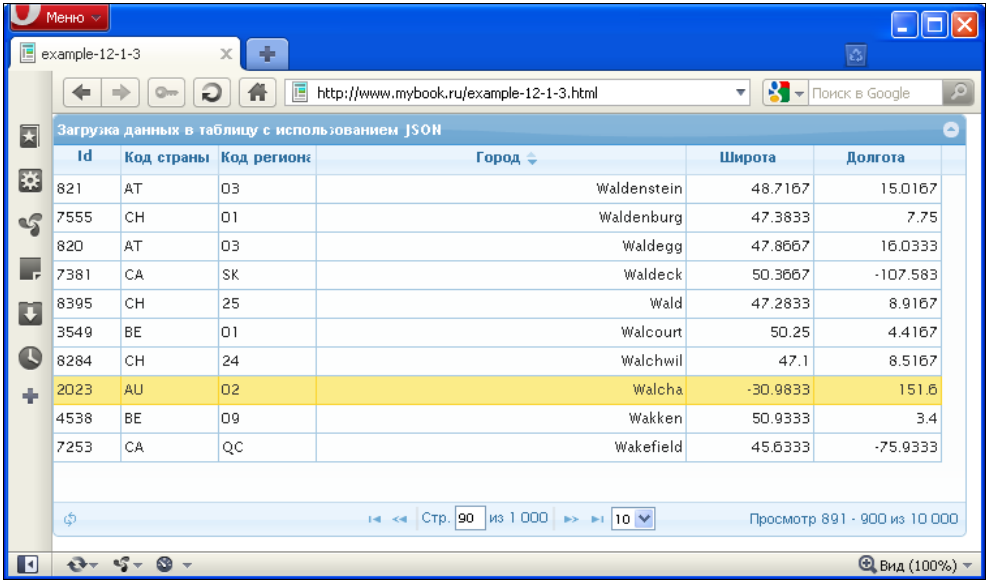


Рис. 12.2. Использование плагина jqGrid с навигационной панелью

Примечание

При создании навигационной панели с помощью метода `navGrid` мы использовали новый API, тогда как в примерах из листингов 12.1.1 и 12.1.2 при вызове методов `addRowData`, `getGridParam`, `getRowData`, `addRowData`, `setRowData` и `delRowData` был применен старый API. В своих приложениях вы можете выбрать любую форму записи.

Чтобы отправить данные на сервер, потребуется GET-запрос, пример которого приведен в листинге 12.1.3а.

Листинг 12.1.3а. Использование плагина jqGrid

```
http://www.server.ru/datafromserver.php?_search=false&nd=1301680124043&rows=
10&page=90&sidx=city&sord=desc
```

Первый параметр `_search` может принимать значения `false` или `true`. В нашем случае передается значение `false`, т. к. панели поиска пока у нас нет (с ней знакомство еще предстоит). Параметр `nd` передается для предотвращения кэширования запроса. Следующие далее параметры уже немного знакомы: `rows` — число строк таблицы, которые выводятся на одной странице, `page` — номер страницы, `sidx` — поле, по которому необходимо проводить сортировку, и `sord` — направление сортировки.

Пример JSON-объекта, который возвращается сервером в ответ на такой запрос, приведен в листинге 12.1.3б.

Листинг 12.1.3б. Использование плагина jqGrid

```
{
"page":"90",
"total":1000,
"records":"10000",
"rows":[
 {"id":"821","cell":["821","AT","03","Waldenstein","48.7167","15.0167"]},
 {"id":"7555","cell":["7555","CH","01","Waldenburg","47.3833","7.75"]},
 {"id":"820","cell":["820","AT","03","Waldegg","47.8667","16.0333"]},
 {"id":"7381","cell":["7381","CA","SK","Waldeck","50.3667","-107.583"]},
 {"id":"8395","cell":["8395","CH","25","Wald","47.2833","8.9167"]},
 {"id":"3549","cell":["3549","BE","01","Walcourt","50.25","4.4167"]},
 {"id":"8284","cell":["8284","CH","24","Walchwil","47.1","8.5167"]},
 {"id":"2023","cell":["2023","AU","02","Walcha","-30.9833","151.6"]},
 {"id":"4538","cell":["4538","BE","09","Wakken","50.9333","3.4"]},
 {"id":"7253","cell":["7253","CA","QC","Wakefield","45.6333","-75.9333"]}
]
}
```

Исходный код файла datafromserver.php, который служит для получения информации из базы данных и формирования объекта JSON, помещен на компакт-диске, прилагаемом к книге. Там же находится файл, содержащий SQL Dump таблицы, рассматриваемой в примерах этой главы.

Далее мы попробуем добавлять, редактировать и удалять данные. В начале главы мы разбирали аналогичные возможности плагина при использовании локальных данных, теперь же нас ждет взаимодействие с сервером.

Листинг 12.1.4. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-12-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
<style type="text/css">
```

```
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
</style>
<script type="text/javascript">
$(function(){
  $("#example").jqGrid({
    url:"datafromserver.php",
    datatype: "json",
    height: 250,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', index:'id', width:50, sorttype:"int"},
      {name:'country_code', index:'country_code', width:75, sorttype:"date",
editable:true},
      {name:'region_code', index:'region_code', width:75, sorttype:false,
editable:true},
      {name:'city', index:'city', width:300, align:"right", sorttype:"date",
editable:true},
      {name:'latitude', index:'latitude', width:100, align:"right",
sorttype:"float", editable:true},
      {name:'longitude', index:'longitude', width:100, align:"right",
sorttype:"float", editable:true}
    ],
    rowNum:10,
    rowList:[10,20,30],
    pager: "#pager",
    sortname: "country_code",
    viewrecords: true,
    caption: "Добавление, редактирование и удаление данных",
    editurl: "dataoperation.php"
  });
$("#example").jqGrid('navGrid','#pager',{edit:true,add:true,del:true,search:fal
se});
});
</script>
</head>
<body>
<table id="example"></table>
<div id="pager"></div>
</body>
</html>
```

Рассматривать код, приведенный в листинге 12.1.4, будем, только отмечая изменения по сравнению с листингом 12.1.3.

Итак, в JavaScript-коде изменения коснулись свойства `colModel` объекта настроек плагина. В некоторых элементах массива, которые в свою очередь являются объек-

тами, добавилось свойство `editable` со значением `true` — эти ячейки таблицы мы сможем теперь редактировать.

Поменялся заголовок таблицы, но это изменение непринципиальное.

Добавилось свойство `editurl` с присвоенным значением dataoperation.php — адресом файла на сервере, которому будут адресованы данные не только для редактирования, но и для добавления и удаления.

Некоторые изменения произошли в последней строке JavaScript-кода, которая отвечает за создание навигационной панели. Значения `true` присвоены свойствам `edit`, `add` и `del`.

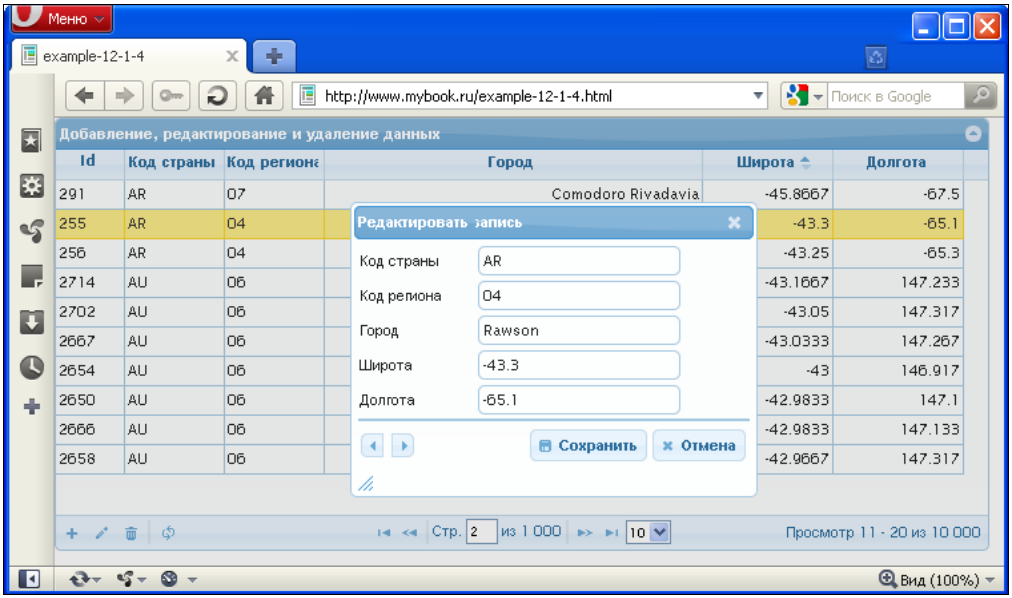


Рис. 12.3. Использование плагина jqGrid. Диалог редактирования записи

Обратите внимание на рис. 12.3 — в левом нижнем углу, в навигационной панели появились три дополнительных значка. При нажатии на них вызываются соответствующие диалоги (на рисунке показан диалог редактирования записи).

Исходный код файла dataoperation.php помещен на компакт-диск, прилагаемый к книге.

Следующий пример демонстрирует редактирование данных непосредственно в ячейках таблицы. Для того чтобы реализовать эту возможность, нам потребуется познакомиться с тем, как плагин jqGrid умеет реагировать на события.

Листинг 12.1.5. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-12-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
<style type="text/css">
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
</style>
<script type="text/javascript">
$(function(){
  var lastsel;
  $("#example").jqGrid({
    url:"datafromserver.php",
    datatype: "json",
    height: 250,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', index:'id', width:50, sorttype:"int"},
      {name:'country_code', index:'country_code', width:75, sorttype:"date",
editable:true},
      {name:'region_code', index:'region_code', width:75, sorttype:false,
editable:true},
      {name:'city', index:'city', width:300, align:"right", sorttype:"date",
editable:true},
      {name:'latitude', index:'latitude', width:100, align:"right",
sorttype:"float", editable:true},
      {name:'longitude', index:'longitude', width:100, align:"right",
sorttype:"float", editable:true}
    ],
    rowNum:10,
    rowList:[10,20,30],
    pager: "#pager",
    sortname: "country_code",
    viewrecords: true,
    caption: "Редактирование данных в таблице",
    onSelectRow: function(id) {
      if(id && id !== lastsel) {
        $('#example').jqGrid('restoreRow', lastsel);
        $('#example').jqGrid('editRow', id, true);
```

```
        lastsel = id;
      }
    },
    editurl: "dataoperation.php"
  });
  $("#example").jqGrid('navGrid','#pager',{edit:false,add:false,del:false,
search:false});

});
</script>
</head>
<body>
<table id="example"></table>
<div id="pager"></div>
</body>
</html>
```

В коде, приведенном в листинге 12.1.5, произошли небольшие изменения. Во-первых, в самом начале мы объявили переменную `lastsel`. Во-вторых, в настройках навигационной панели (это последняя строка JavaScript-кода) свойствам `edit`, `add`, `del` и `search` вновь присвоено значение `false`. И, в-третьих, в настройках самого плагина появилось свойство, которое до сих пор не встречалось — `onSelectRow`.

В свойстве `onSelectRow` можно определить функцию, вызываемую сразу после щелчка на строке.

Давайте разберемся, что же делает функция, определенная в этом свойстве, и попутно познакомимся с еще двумя методами плагина — `restoreRow` и `editRow`.

В примере из листинга 12.1.5 мы передаем этой функции аргумент `id` — идентификатор строки. Вызов методов `restoreRow` и `editRow` находится внутри условия `if`. Это сделано для того, чтобы предотвратить возможность редактирования более одной строки одновременно, поскольку оба метода "работают" в паре. При вызове метода `editRow` в ячейки таблицы вставляются элементы `input` (по умолчанию), а содержимое ячеек становится значением атрибута `value` элементов `input`. При вызове метода `restoreRow` уже значение атрибута `value` элементов `input` становится содержимым ячейки.

Среди свойств плагина jqGrid есть возможность организации поиска данных. В листинге 12.1.6 приведен пример кода, с помощью которого можно реализовать вызов и использование поискового диалога.

Листинг 12.1.6. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-12-1-6</title>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
<style type="text/css">
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
</style>
<script type="text/javascript">
$(function(){
  $("#example").jqGrid({
    url:"datafromserver2.php",
    datatype: "json",
    height: 250,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', index:'id', width:50, sorttype:"int"},
      {name:'country_code', index:'country_code', width:75, sorttype:"date",
search:false},
      {name:'region_code', index:'region_code', width:75, sorttype:false},
      {name:'city', index:'city', width:300, align:"right", sorttype:"date",
search:false},
      {name:'latitude', index:'latitude', width:100, align:"right",
sorttype:"float"},
      {name:'longitude', index:'longitude', width:100, align:"right",
sorttype:"float"}
    ],
    rowNum:10,
    rowList:[10,20,30],
    pager: "#pager",
    sortname: "country_code",
    viewrecords: true,
    caption: "Поиск в таблице"
  });
$("#example").jqGrid('navGrid','#pager',{ edit:false, add:false, del:false,
search:true });
$("#example").jqGrid('searchGrid', {sopt:['eq','lt','gt'], caption:
"Поищем..."});
});
</script>
</head>
<body>
<table id="example"></table>
```

```
<div id="pager"></div>
</body>
</html>
```

Практически все свойства в объекте с настройками плагина нам уже знакомы. Остановимся на последней строке JavaScript-кода — по аналогии с инициализацией навигационной панели, мы создаем панель поиска с помощью метода `searchGrid`. В первом аргументе указываем имя метода, во втором — передаем объект с настройками панели. Значение свойства `sopt` — массив, в котором передаются условные обозначения, используемые на сервере для построения запроса к базе данных. В нашем случае мы передаем `eq` — равно, `lt` — меньше и `gt` — больше.

На рис. 12.4 показана таблица с поисковым диалогом.

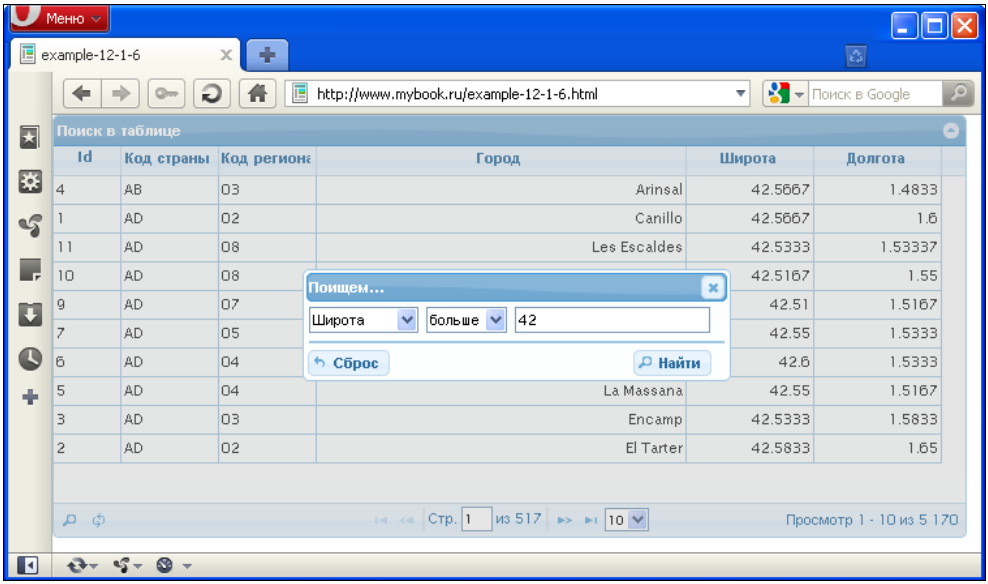


Рис. 12.4. Использование плагина jqGrid. Поисковый диалог

Не забывайте, что плагин jqGrid будет отображать в таблице именно то, что вернет сервер в ответ на поисковый запрос. В нашем примере был использован файл datafromserver2.php, исходные коды которого вы сможете найти на компакт-диске, прилагаемом к книге.

Рассмотрим еще одну возможность — организацию фильтрации данных с помощью дополнительной встроенной панели. При этом большой объем данных будет загружен с помощью только одного запроса к серверу, а все дальнейшие операции будут происходить только на стороне клиента (листинг 12.1.7).

Листинг 12.1.7. Использование плагина jqGrid

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-12-1-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="stylesheet" type="text/css" href="css/redmond/jquery-ui-
1.8.11.custom.css" />
<link rel="stylesheet" type="text/css" href="css/ui.jqgrid.css" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/i18n/grid.locale-ru.js" type="text/javascript"></script>
<script src="js/jquery.jqGrid.min.js" type="text/javascript"></script>
<style type="text/css">
html, body {
  margin: 0;
  padding: 0;
  font-size: 62.5%;
}
</style>
<script type="text/javascript">
$(function(){
  $("#example").jqGrid({
    url:"datafromserver2.php",
    datatype: "json",
    height: 250,
    colNames:['Id','Код страны','Код региона','Город','Широта','Долгота'],
    colModel:[
      {name:'id', index:'id', width:50, sorttype:"int"},
      {name:'country_code', index:'country_code', width:75},
      {name:'region_code', index:'region_code', width:75},
      {name:'city', index:'city', width:300, align:"right"},
      {name:'latitude', index:'latitude', width:100, align:"right"},
      {name:'longitude', index:'longitude', width:100, align:"right"}
    ],
    rowNum:100,
    rowTotal: 10000,
    rowList:[30,50,100],
    loadonce:true,
    rownumbers: true,
    rownumWidth: 40,
    pager: "#pager",
    sortname: "country_code",
    viewrecords: true,
    caption: "Поиск в таблице"
  });
  $("#example").jqGrid('navGrid','#pager', {edit:false, add:false, del:false,
search:false});
  $("#example").jqGrid('filterToolbar', {searchOnEnter:false});

});
```

```
</script>
</head>
<body>
<table id="example"></table>
<div id="pager"></div>
</body>
</html>
```

Несколько новых свойств появилось в объекте настроек плагина jqGrid. Если свойство `rowNum` нам знакомо — число строк, отображаемых на одной странице, то свойство `rowTotal` встречается впервые. Здесь передается число строк, которые должен вернуть сервер в ответе на единственный запрос. То, что запрос необходимо выполнить только один раз, определяет свойство `loadonce` со значением `true`. Свойство `rownumbers` со значением `true` позволяет отобразить в дополнительной колонке слева нумерацию строк по порядку, а свойство `rownumWidth` со значением `40` задает ширину этой колонки.

В самой последней строке JavaScript-кода мы создаем дополнительную панель для фильтрации данных, отображаемую в верхней части таблицы. Применяем для этого метод `filterToolbar`, которому передаем объект настроек с единственным свойством `searchOnEnter`. Значение `false`, установленное для этого свойства, позволяет фильтровать данные не по нажатию клавиши <Enter>, а непосредственно при вводе каждого очередного символа.

На рис. 12.5 показана таблица с панелью фильтрации данных.

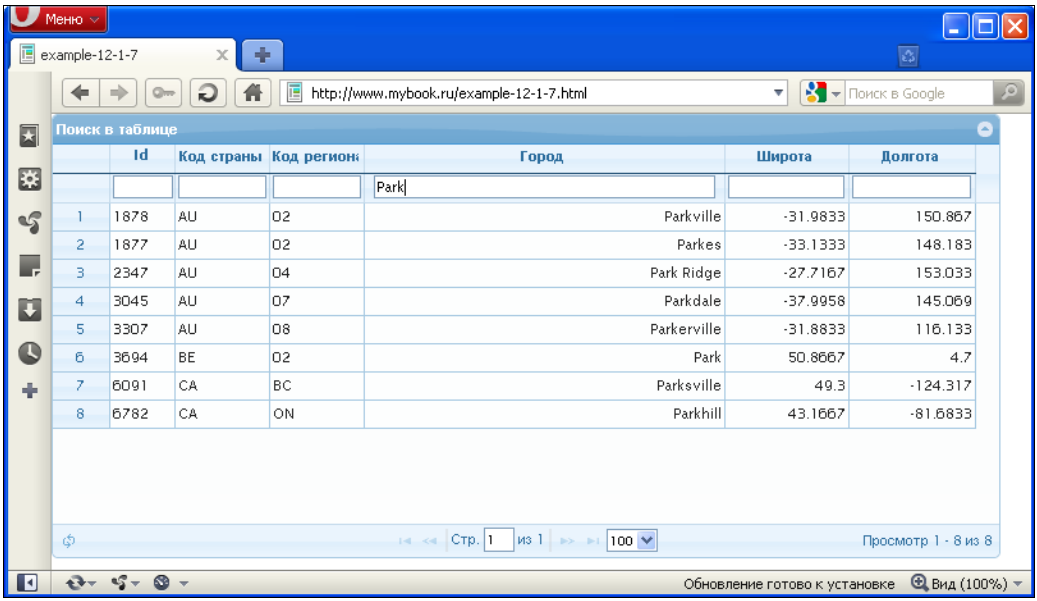
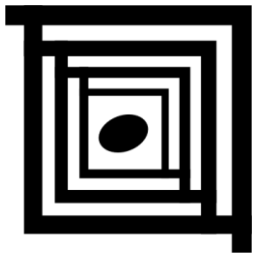


Рис. 12.5. Использование плагина jqGrid. Панель фильтрации данных

Внимание!

Файлы datafromserver.php, datafromserver2.php и dataoperation.php в примерах этой главы созданы исключительно в целях демонстрации возможностей плагина jqGrid и не имеют никаких проверок поступающих данных. Во избежание проблем с безопасностью ваших приложений не используйте их в существующем виде.

К сожалению, в рамках этой книги невозможно полностью рассказать о замечательном плагине jqGrid — мы рассмотрели лишь небольшую часть его возможностей. Однако хочется надеяться, что примеры, которые мы разобрали, послужат хорошим фундаментом для дальнейшей самостоятельной работы. Множество примеров использования плагина jqGrid находится на сайте разработчика по адресу **http://trirand.com/blog/jqgrid/jqgrid.html**. Официальную документацию на плагин можно найти по адресу **http://www.trirand.com/jqgridwiki/doku.php?id=wiki:jqgriddocs**.



ГЛАВА 13

Графики и диаграммы

Представьте себе веб-сайт, на котором можно посмотреть изменение курсов валют за какой-либо период времени, или систему сбора статистики посещаемости сайта. Различные графики и диаграммы в настоящее время стали неотъемлемой частью Интернета. Попробуем и мы разобраться, как с помощью библиотеки jQuery и плагинов к ней существенно облегчить создание графиков и диаграмм для веб-сайта.

13.1. Плагин jqPlot

ЗАДАЧА

На веб-странице необходимо организовать визуальное представление данных в виде графиков или диаграмм.

Решение

Для решения поставленной задачи применим jqPlot — один из самых удачных плагинов к библиотеке jQuery и уж совершенно точно самый мощный среди плагинов, позволяющих реализовывать графики и диаграммы. Плагин имеет очень много всевозможных настроек, описанию которых можно посвятить отдельную книгу. Поэтому мы остановимся на рассмотрении основных возможностей и на относительно простых примерах познакомимся с принципами реализации различных типов графиков, что позволит в дальнейшем самостоятельно решить задачи, не описанные в этой главе. Листинг 13.1.1 иллюстрирует простейший пример использования плагина jqPlot.

Листинг 13.1.1. Использование плагина jqPlot

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
```

```
<head>
<title>example-13-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!--[if IE]><script type="text/javascript"
src="js/excanvas.min.js"></script><![endif]-->
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jqplot.min.js"></script>
<link rel="stylesheet" type="text/css" href="js/jquery.jqplot.min.css" />
<script type="text/javascript">
$(function(){
  $.jqplot("example",
[[[0,0],[1,2],[3,5.12],[5,13.1],[7,33.6],[9,85.9],[11,219.9]]]);
});
</script>
</head>
<body>
<div id="example" style="height:300px;width:400px; "></div>
</body>
</html>
```

Обсуждение

Прежде чем начать обсуждение, посмотрите на рис. 13.1, на котором показан результат выполнения кода, приведенного в листинге 13.1.1.

График, представленный на рис. 13.1, выглядит очень изящно, но еще более впечатляет код в листинге 13.1.1 — буквально одна строка JavaScript-кода и простейшая HTML-разметка, состоящая из элемента `div` с идентификатором `example` и размерами, заданными в атрибуте `style`.

Но конечно далеко не все так просто — чтобы получить такой график, сначала нужно подключить к веб-странице несколько необходимых файлов, среди которых, конечно же, файл библиотеки jQuery — js/jquery-1.5.2.min.js. Кроме этого потребуются файл плагина jquery.jqplot.min.js и файл его стилевого оформления jquery.jqplot.css.

Я не упомянул еще об одном подключаемом файле, который предназначен для браузеров IE — excanvas.min.js. Для того чтобы создавать графические формы, рисовать линии, вращать изображения, плагин использует тег `<canvas>` — элемент HTML 5, предназначенный для создания растрового изображения при помощи JavaScript. Браузеры FireFox, Opera, Chrome, Safari поддерживают тег `<canvas>`. Как обычно, исключением оказался Internet Explorer — в 8-й версии этот тег еще не поддерживается. Тут на помощь пришла компания Google, реализовав для IE полноценный API Canvas, — это и есть файл excanvas.js.

Вернемся теперь к обсуждению JavaScript-кода из листинга 13.1.1. Совершенно очевидно, что мы передали плагину два параметра — идентификатор элемента, в который будет осуществляться вывод графика, и координаты точек, по которым этот график будет построен. Все остальное плагин jqPlot проделал вполне самостоятельно.

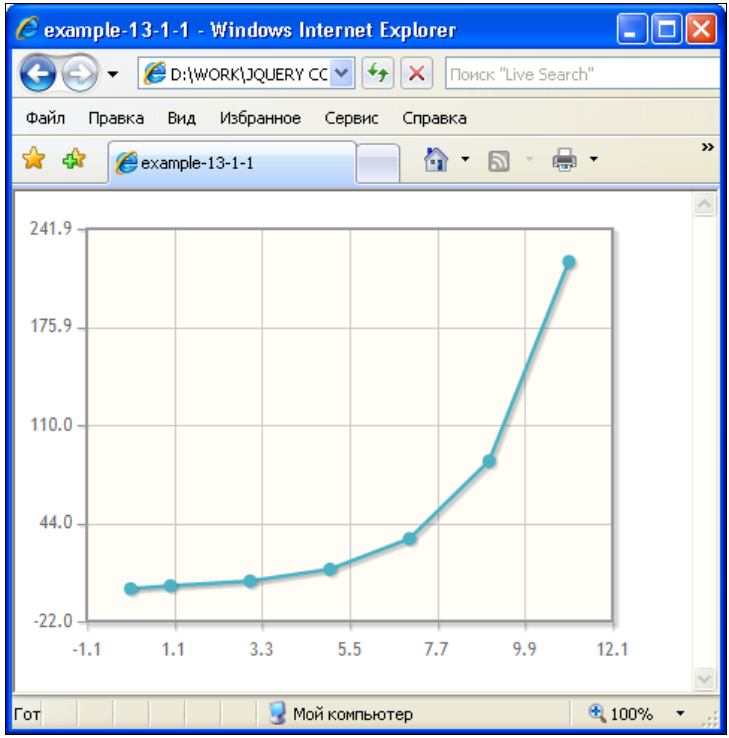


Рис. 13.1. Результат использования плагина jqPlot с настройками по умолчанию

Есть еще необязательный (но на практике очень важный) параметр — объект, в котором плагину передаются многочисленные настройки. Получается, что применить на веб-странице плагин jqPlot с настройками по умолчанию очень легко, но гораздо сложнее изучить эти настройки.

Этим мы и начнем заниматься на примере, приведенном в листинге 13.1.2.

Листинг 13.1.2. Использование плагина jqPlot

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-13-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!--[if IE]><script type="text/javascript"
src="js/excanvas.min.js"></script><![endif]-->
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jqplot.min.js"></script>
<link rel="stylesheet" type="text/css" href="js/jquery.jqplot.min.css" />
<script type="text/javascript">
$(function(){
  line1 = [[1,2],[2,4],[3,8],[4,16],[5,8],[6,4],[7,2],[8,1]];
```

```
  line2 = [4, 8, 16, 24, 16, 16, 8, 4];
  line3 = [16, 16, 12, 18, 26, 24, 6, 2];
  $.jqplot("example",
           [line1,line2,line3],
           {
                title: "Демонстрационный график",
                axes: {
                      yaxis: {
                            min:0, max:30
                       },
                       xaxis: {
                            min:0, max:8
                       }
                },
                series: [
                  { color:"#4bb2c5",label:"Николай" },
                  { color:"#EAA228",label:"Алексей" },
                  { color:"#839557",label:"Василий" }
                ],
                legend: {
                      show: true,
                      location: "ne",
                      xoffset: 12,
                      yoffset: 12 }
           }
          );
});
</script>
</head>
<body>
<div id="example" style="height:300px;width:600px; "></div>
</body>
</html>
```

С первого взгляда становится ясно, насколько усложнился код в листинге 13.1.2. Мы подключили те же самые внешние файлы, в HTML-разметке по-прежнему присутствует элемент `div` с идентификатором `example`, но JavaScript-код значительно изменился. Его изучению мы и уделим основное внимание. Но сначала посмотрим на результат выполнения этого кода (рис. 13.2).

Первое, что мы делаем в JavaScript-коде, — подготавливаем три массива, по которым будут построены три графика. Затем мы вызываем плагин jqPlot, в первом аргументе передаем ему идентификатор элемента, в который будут выводиться графики, во втором аргументе — массив из приготовленных ранее данных и, наконец, в третьем аргументе — объект с настройками плагина. Если с первыми двумя аргументами все более или менее понятно, то с третьим аргументом знакомство только предстоит.

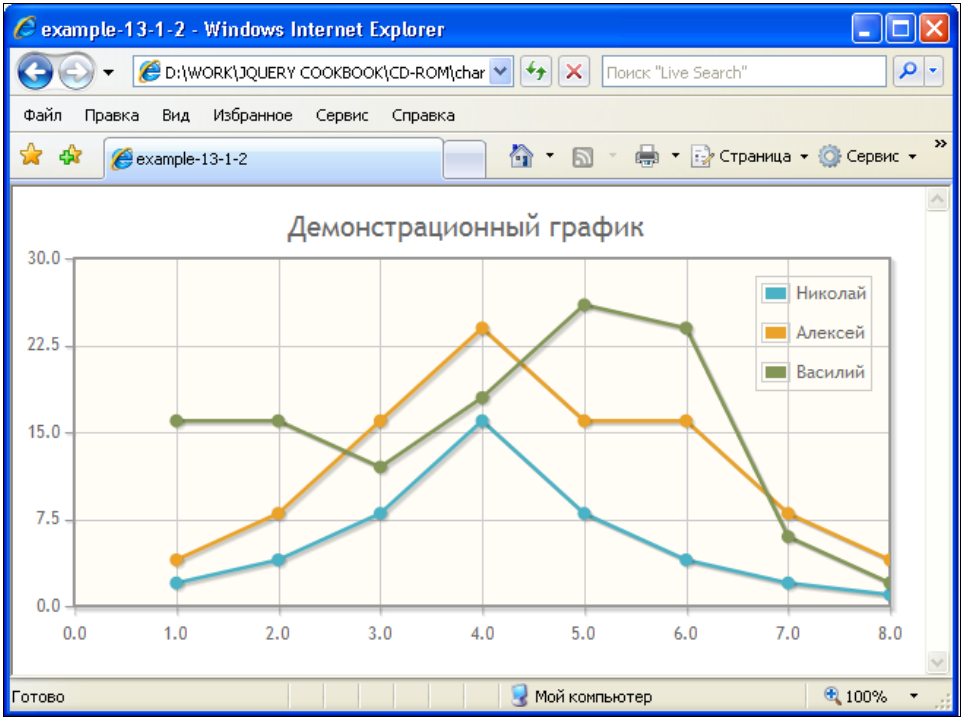


Рис. 13.2. Результат использования плагина jqPlot

Итак, первое свойство объекта называется `title`, его значение — строка, в которой можно указать название графика.

Следующее свойство — `axes`. Его значение также является объектом, в свойствах `yaxis` и `xaxis` которого описываются соответствующие оси координат. Значения свойств `yaxis` и `xaxis` тоже объекты, но уже описывающие конкретные характеристики осей координат. В примере из листинга 13.1.2, в свойствах `min` и `max` указаны минимальные и максимальные значения для соответствующих шкал.

Следующее свойство объекта настроек — `series` — массив, каждый элемент которого есть объект, характеризующий какой-либо из графиков. Нетрудно догадаться, что свойство `color` описывает цвет, а в свойстве `label` хранится метка для соответствующего графика.

Последнее свойство, представленное в примере из листинга 13.1.2, — `legend` — объект, характеризующий легенду графика. Свойство `show` этого объекта принимает логические значения. В случае `true` плагин будет отображать легенду. Свойство `location` — расположение легенды. Значение `ne` заставит плагин поместить легенду в правом верхнем углу (`ne` — это nord-east, северо-восток). Свойства `xoffset` и `yoffset` определят отступы легенды от края.

В общем, все оказалось не так уж и сложно, поэтому двигаемся дальше и рассмотрим пример, приведенный в листинге 13.1.3.

Листинг 13.1.3. Использование плагина jqPlot

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-13-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!--[if IE]><script type="text/javascript"
src="js/excanvas.min.js"></script><![endif]-->
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jqplot.min.js"></script>
<link rel="stylesheet" type="text/css" href="js/jquery.jqplot.min.css" />
<script type="text/javascript"
src="js/plugins/jqplot.pieRenderer.min.js"></script>
<script type="text/javascript">
$(function(){
  line1 = [["груши",3], ["яблоки",7], ["апельсины",2.5], ["бананы",6],
["арбузы",5], ["дыни",4]];
  $.jqplot("example", [line1], {
    title: "Пример круговой диаграммы",
    seriesDefaults: {
      renderer: $.jqplot.PieRenderer,
      rendererOptions: {
        sliceMargin:8
      }
    },
    legend: { show:true }
  });
});
</script>
</head>
<body>
<div id="example" style="height:340px; width:540px;"></div>
</body>
</html>
```

Сразу же посмотрим на рис. 13.3, на котором можно увидеть результат выполнения кода.

А теперь начнем разбирать пример из листинга 13.1.3. HTML-разметка страницы осталась прежней, только немного изменились размеры, заданные в атрибуте `style` элемента `div`.

А вот среди подключаемых внешних файлов произошли изменения. Дополнительно был подключен файл js/plugins/jqplot.pieRenderer.min.js, который служит как раз для того, чтобы иметь возможность организовать круговую диаграмму.

Вообще, следует отметить, что jqPlot, который является плагином для библиотеки jQuery, в свою очередь имеет набор собственных плагинов, реализующих различ-

ные виды графиков и действия над ними. Один из таких плагинов — файл jqplot.pieRenderer.min.js.

Займемся рассмотрением JavaScript-кода. Вначале все практически так же, как и в предыдущих примерах. Готовим данные для построения диаграммы, которые храним в переменной `line1`. Обратите внимание, что метку можно указать непосредственно в массиве, описывающем конкретную величину. Затем вызываем плагин jqPlot, в первом аргументе передаем ему идентификатор элемента, в котором будет построена диаграмма. Второй аргумент — массив, содержащий данные для построения диаграммы. И, наконец, третий аргумент — объект с настройками.

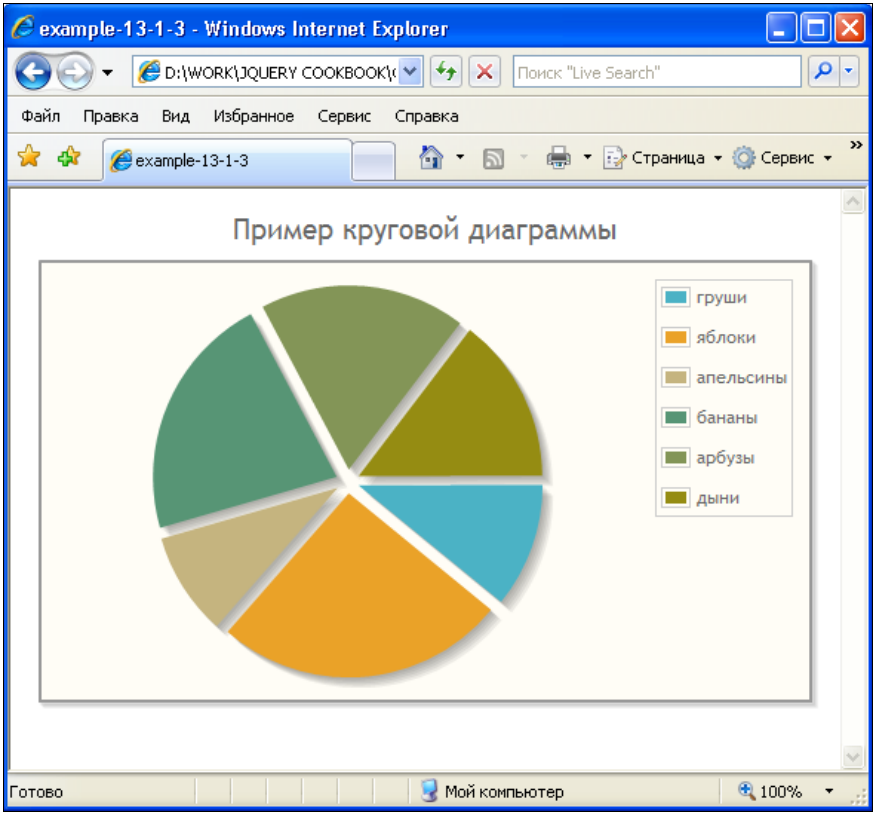


Рис. 13.3. Результат использования плагина jqPlot

Первое и третье свойства этого объекта нам уже знакомы. Свойство `title` определяет заголовок диаграммы, а `legend` в данном случае заставит плагин отображать легенду диаграммы.

А вот с важным свойством `seriesDefaults` мы пока не встречались. Значение этого свойства — объект, в котором определяются настройки по умолчанию. Передавая в этом объекте его отдельные свойства, можно переопределить поведение плагина. Так мы изменяем значение опции `renderer` на `$.jqplot.PieRenderer`, в отличие от определенного по умолчанию значения `$.jqplot.LineRenderer`, чтобы рисовать не

линейные графики, а круговую диаграмму. В опции `rendererOptions`, значение которой в свою очередь тоже является объектом, передаем одну из настроек — `sliceMargin` для того, чтобы организовать отступы между сегментами круговой диаграммы.

Следующий пример — диаграмма, где значения представлены в виде столбцов. Код примера приведен в листинге 13.1.4.

Листинг 13.1.4. Использование плагина jqPlot

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-13-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!--[if IE]><script type="text/javascript"
src="js/excanvas.min.js"></script><![endif]-->
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jqplot.min.js"></script>
<link rel="stylesheet" type="text/css" href="js/jquery.jqplot.min.css" />
<script type="text/javascript" src="js/plugins/jqplot.barRenderer.min.js"
/></script>
<script type="text/javascript"
src="js/plugins/jqplot.categoryAxisRenderer.min.js" /></script>
<script type="text/javascript" src="js/plugins/jqplot.pointLabels.min.js"
/></script>
<script type="text/javascript">
$(function(){
  line1 = [14, 32, 41, 44, 40, 37, 29];
  line2 = [7, 12, 15, 17, 20, 27, 39];
  $.jqplot("example", [line1, line2], {
    title: "Пример диаграммы с метками значений",
    stackSeries: true,
    seriesDefaults: {
      renderer: $.jqplot.BarRenderer,
      rendererOptions: { barMargin: 25 },
      pointLabels: { show: true, stackedValue: true }
    },
    axes: {
      xaxis:{ renderer:$.jqplot.CategoryAxisRenderer },
      yaxis:{ ticks:[0, 20, 40, 60, 80] }
    }
  });
});
</script>
</head>
<body>
```

```
<div id="example" style="height:320px; width:480px;"></div>
</body>
</html>
```

Сразу же посмотрим, как выглядит результат работы этого кода (рис. 13.4).

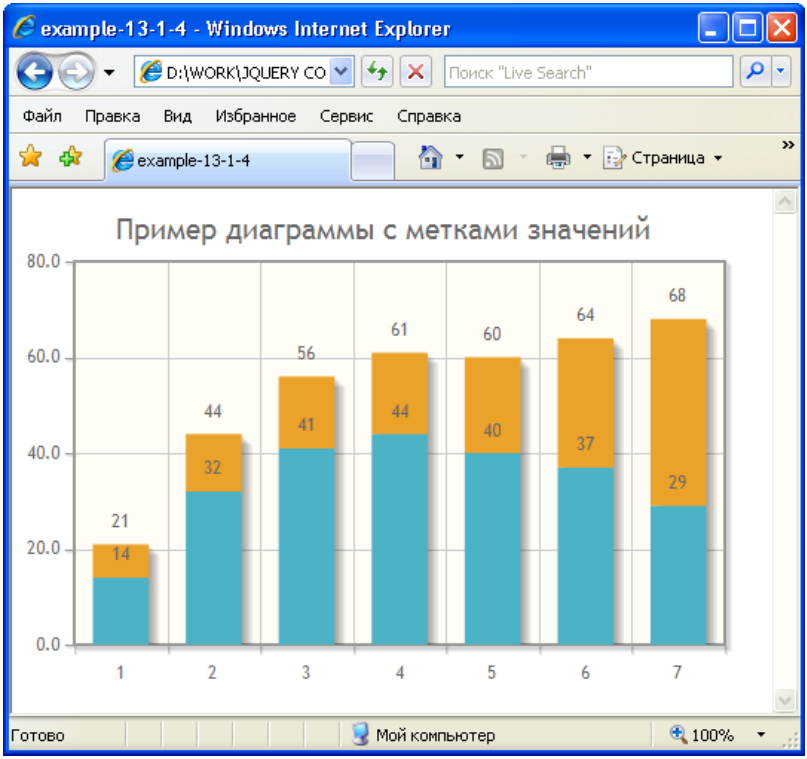


Рис. 13.4. Результат использования плагина jqPlot

Опять HTML-разметка почти не изменилась, но мы были вынуждены подключить довольно много внешних файлов для реализации диаграммы, представленной на рис. 13.4. Помимо библиотеки jQuery, стилевого файла плагина jqPlot и собственно самого файла плагина, потребовались файлы js/plugins/jqplot.barRenderer.min.js, js/plugins/jqplot.categoryAxisRenderer.min.js и js/plugins/jqplot.pointLabels.min.js.

В JavaScript-коде тоже все довольно понятно, но, памятуя, что "повторение — мать учения", разберем JavaScript-код из листинга 13.1.4, как всегда, весьма подробно.

Как обычно, сначала готовим данные. На этот раз это два массива, которые мы храним в переменных `line1` и `line2`. Вызываем плагин jqPlot. Первый параметр — идентификатор элемента `div`, куда выводится диаграмма. Второй параметр — данные, третий — объект с настройками.

Опция `title` нам уже хорошо знакома — это заголовок диаграммы. С опцией `stackSeries` только предстоит познакомиться. Она может принимать логические значения. В данном случае значение `true` заставит плагин расположить столбцы

друг за другом — "стопкой". Следующая опция `seriesDefaults` нам тоже немного знакома. В свойстве `renderer` указываем значение `$.jqplot.BarRenderer`, чтобы рисовать диаграмму в виде столбцов. В свойстве `rendererOptions` передаем объект только с одним свойством `barMargin`, чтобы установить размеры полей для столбцов диаграммы. И, наконец, опция `pointLabels`, в которой передаем плагину еще один объект со свойствами `show` и `stackedValue`. Значение `true` в первом случае разрешит отображать метки, а во втором — определит отображение значений столбцов диаграммы такой же "стопкой", как и сами столбцы.

Последние свойство в объекте настроек — `axes` — поможет нам описать оси координат в том виде, в котором мы их хотим видеть. В свойстве `xaxis` передаем объект, где в опции `renderer` указываем значение `$.jqplot.CategoryAxisRenderer` — т. е. плагин для jqPlot, который будет использован для построения оси *X*. Теперь значения шкалы не будут рассчитываться автоматически. Вместо этого под каждым столбцом диаграммы появится название категории, и поскольку мы не задали явно эти названия, то они будут обозначены цифрами. В свойстве `yaxis` передаем объект, где в опции `ticks` указываем массив, значения которого необходимы для построения оси *Y*.

Осталось разобрать последний и очень интересный пример, код которого приведен в листинге 13.1.5.

Листинг 13.1.5. Использование плагина jqPlot

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-13-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!--[if IE]><script type="text/javascript"
src="js/excanvas.min.js"></script><![endif]-->
<script type="text/javascript" src="js/jquery-1.5.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jqplot.min.js"></script>
<link rel="stylesheet" type="text/css" href="js/jquery.jqplot.min.css" />
<script type="text/javascript" src="js/plugins/jqplot.dragable.min.js"
/></script>
<script type="text/javascript" src="js/plugins/jqplot.trendline.min.js"
/></script>
<script type="text/javascript">
$(function(){
  $.jqplot.config.enablePlugins = true;
  line1=[4, 25, 13, 22, 14, 17, 15];
  $.jqplot("example", [line1],
    { title: "Изменяемые значения и линия тренда" }
  );
});
</script>
```

```
</head>
<body>
<div id="example" style="height:320px; width:480px;"></div>
</body>
</html>
```

Сразу же посмотрим на результат выполнения кода (рис. 13.5).

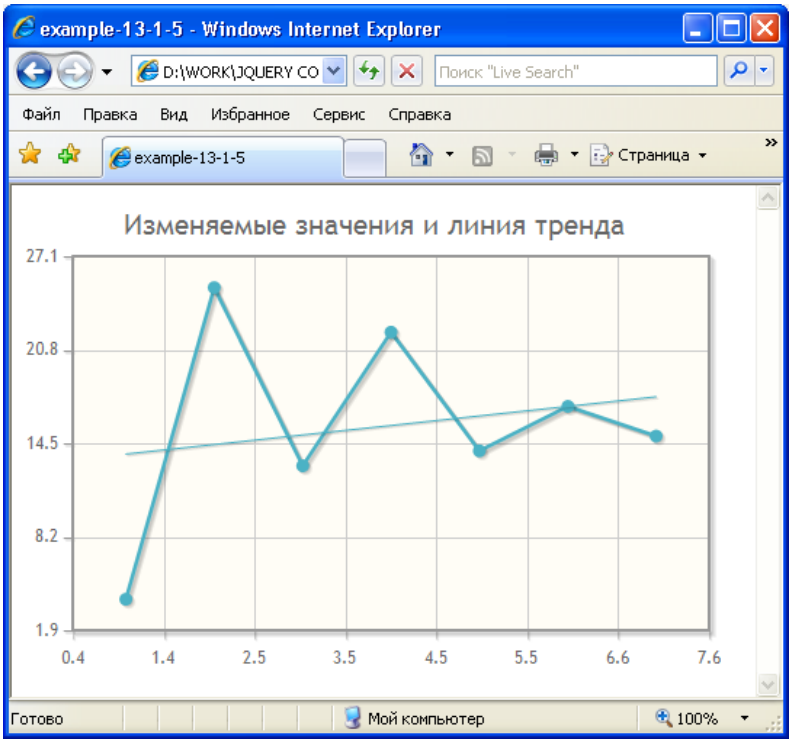


Рис. 13.5. Результат использования плагина jqPlot

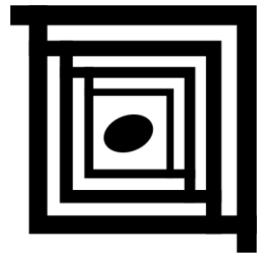
Это очень полезная возможность плагина jqPlot. Во-первых, данный график можно изменять, перемещая его узлы (отмеченные точками) с помощью указателя мыши. Во-вторых, при этом автоматически рассчитывается линия тренда.

Для создания такой функциональности потребуется подключить дополнительные файлы. Посмотрите код, приведенный в листинге 13.1.5, — мы подключили два плагина: js/plugins/jqplot.dragable.min.js и js/plugins/jqplot.trendline.min.js. Первый реализует возможность перемещения узлов графика с помощью указателя мыши, второй — рассчитывает линию тренда.

Сам JavaScript-код довольно прост, т. к. мы не передаем никаких дополнительных настроек, за исключением заголовка. Остальные действия выполняем как обычно — готовим данные, вызываем плагин, которому передаем идентификатор целевого элемента, данные и немного настроек.

Обратить внимание следует разве на строку `$.jqplot.config.enablePlugins = true`, которая раньше нигде не встречалась. Начиная с версии 0.9.7 необходимо явно включить плагины, установив значение `true` для опции конфигурации `enablePlugins` или для опции `show` в настройках, передаваемых плагину (так было сделано в примере из листинга 13.1.4 для опции `pointLabels`).

В заключение остается сказать, что мы рассмотрели далеко не все возможности плагина jqPlot по организации графиков и диаграмм, но получили достаточно информации, чтобы суметь самостоятельно решать подобного рода задачи. В качестве дополнительных материалов можно воспользоваться большой коллекцией примеров на сайте разработчика плагина **http://www.jqplot.com/**, а увидеть перечень всех возможных опций можно на странице **http://www.jqplot.com/docs/files/jqPlotOptions-txt.html**.



ГЛАВА 14

AJAX-формы

14.1. Плагин jQuery Form

ЗАДАЧА

На веб-странице уже существует традиционная HTML-форма для отправки данных. Поставлена задача как можно проще доработать существующую форму так, чтобы при передаче данных на сервер использовалась технология AJAX.

Решение

Решим эту задачу с помощью плагина jQuery Form, который позволит легко и ненавязчиво модернизировать существующую форму (листинг 14.1.1).

Листинг 14.1.1. Использование плагина jQuery Form

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/form.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.form.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  var options = {
    target: "#output",
    beforeSubmit: showRequest,
    success: showResponse,
    timeout: 3000
  };
```

```
  $("#myForm").submit(function() {
    $(this).ajaxSubmit(options);
    return false;
  });

  // вызов перед передачей данных
  function showRequest(formData, jqForm, options) {
    var queryString = $.param(formData);
    alert('Вот что мы передаем: \n\n' + queryString);
    return true;
  }

  // вызов после получения ответа
  function showResponse(responseText, statusText)  {
    alert('Статус ответа сервера: ' + statusText +
          '\n\nТекст ответа сервера: \n' + responseText +
          '\n\nбудет помещен в элемент #output.');
  }
});
</script>
</head>
<body>
<div id="output"> </div>

<form id="myForm" action="form.php" method="post">
<div>
<input type="hidden" name="Hidden" value="hidden Value" /><br />
<label>Имя:</label>
<input name="Name" type="text" value="Моё имя" /><br />
<label>Пароль:</label>
<input name="Password" type="password" /><br />
<label>Мультиселект:</label>
<select class="ms" name="Multiple[]" multiple="multiple">
  <optgroup label="Группа 1">
    <option value="one" selected="selected">Первый элемент</option>
    <option value="two">Второй элемент</option>
    <option value="three">Третий элемент</option>
  </optgroup>
  <optgroup label="Группа 2">
    <option value="four">Четвертый элемент</option>
    <option value="five">Пятый элемент</option>
    <option value="six">Шестой элемент</option>
    <option value="seven">Седьмой элемент</option>
  </optgroup>
</select><br />
```

```
<label>Элемент select:</label>
<select name="Single">
  <option value="one" selected="selected">Первый элемент</option>
  <option value="two">Второй элемент</option>
  <option value="three">Третий элемент</option>
</select><br />
<label>Элемент select 2:</label>
<select name="Single2">
  <optgroup label="Группа 1">
    <option value="A" selected="selected">Буква A</option>
    <option value="B">Буква B</option>
    <option value="C">Буква C</option>
  </optgroup>
  <optgroup label="Группа 2">
    <option value="D">Буква D</option>
    <option value="E">Буква E</option>
    <option value="F">Буква F</option>
    <option value="G">Буква G</option>
  </optgroup>
</select><br />
<label>Загрузка файла:</label>
<input name="File" type="file" /><br />
<label>Чекбоксы:</label>
<input class="cb" type="checkbox" name="Check" value="1" />
<input class="cb" type="checkbox" name="Check" value="2" />
<input class="cb" type="checkbox" name="Check" value="3" /><br />
<label>Радиобаттоны:</label>
<input class="rb" type="radio" name="Radio" value="1" />
<input class="rb" type="radio" name="Radio" value="2" />
<input class="rb" type="radio" name="Radio" value="3" /><br />
<label>Просто текст:</label>
<textarea class="ta" name="Text" rows="2" cols="20">Это элемент
textarea</textarea><br />
<input id="reset" type="reset" name="resetButton" value="Reset" />
<input id="submit2" type="image" name="submitButton" value="Submit2"
src="img/form.gif" />
<input id="submit1" type="submit" name="submitButton" value="Submit1" />
</div>
</form>
</body>
</html>
```

Обсуждение

В листинге 14.1.1 приведена HTML-разметка, описывающая форму, которая включает все доступные для использования в ней элементы. Форма обеспечивает от-

правку данных файлу form.php методом POST. Она будет прекрасно работать без каких-либо плагинов, и даже при отключенной поддержке JavaScript в браузере.

Но перед нами стоит задача сделать из обычной формы AJAX-форму. Для этого мы подключили к веб-странице два внешних файла — файл библиотеки js/jquery-1.5.2.min.js и файл плагина js/jquery.form.js. Еще один подключаемый файл — css/form.css отвечает за стилевое оформление.

В самой HTML-разметке формы нет абсолютно ничего необычного. Отметим только, что кроме самой формы на странице присутствует элемент `div` с идентификатором `output` — в него мы будем вставлять ответ, полученный от сервера.

Займемся рассмотрением JavaScript-кода. В самом начале приготовим объект `options`. В нем будут содержаться некоторые настройки, которые впоследствии мы передадим методу `ajaxSubmit(options)` используемого плагина. Этот небольшой фрагмент кода приведен в листинге 14.1.1а.

Листинг 14.1.1а. Использование плагина jQuery Form

```
var options = {
  target: "#output",
  beforeSubmit: showRequest,
  success: showResponse,
  timeout: 3000
};
```

В свойстве `target` объекта `options` укажем идентификатор целевого элемента, в который необходимо вставить полученные от сервера данные.

В свойстве `beforeSubmit` указана пользовательская функция, вызываемая перед отправкой данных. При желании здесь можно описать какие-либо проверки. Функция должна возвращать `true`, чтобы запрос был отправлен, и `false`, чтобы предотвратить отправку (листинг 14.1.1б).

Листинг 14.1.1б. Использование плагина jQuery Form

```
// вызов функции перед передачей данных
function showRequest(formData, jqForm, options) {
  var queryString = $.param(formData);
  alert('Вот что мы передаем: \n\n' + queryString);
  return true;
}
```

Функция `showRequest` принимает три аргумента — данные формы, объект, представляющий саму форму, и объект `options`, где содержатся не только определенные нами настройки, но и данные, собранные непосредственно из формы, например, целевой `url`, form.php и метод отправки данных POST. Вспомогательная функция `$.param` в примере служит только для демонстрационных целей, чтобы вывести в окне предупреждения данные, приготовленные к отправке.

В свойстве `success` указана функция `showResponse`, которая будет вызвана при успешном получении ответа от сервера. Рассмотрим ее внимательнее, пользуясь листингом 14.1.1в.

Листинг 14.1.1в. Использование плагина jQuery Form

```
// вызов после получения ответа
function showResponse(responseText, statusText)  {
  alert('Статус ответа сервера: ' + statusText +
        '\n\nТекст ответа сервера: \n' + responseText +
        '\n\nбудет помещен в элемент #output.');
}
```

Функция `showResponse` принимает два аргумента — содержимое, которое возвращает сервер в ответ на наш запрос, и статус ответа сервера. Аргумент `statusText` содержит строку со значением `success` при успешном ответе. Другие варианты — `error` или, например, `timeout`.

Но что же конкретно будет содержать аргумент `responseText`? Давайте разберем фрагмент кода, приведенный в листинге 14.1.1г, и посмотрим, что именно должен вернуть сервер в своем ответе.

Листинг 14.1.1г. Использование плагина jQuery Form

```
<?php
header('Content-Type: text/html; charset=utf-8');
$time = date("H:i:s", time());
print '<h3>Ответ сервера</h3>Время: '.$time.'<br />';
var_dump($_POST);
foreach($_FILES as $file) {
  $n = $file['name'];
  $s = $file['size'];
  if (!$n) continue;
  print "File: $n ($s bytes)";
}
?>
```

В первой строке PHP-кода из листинга 14.1.1г устанавливается заголовок, определяющий тип и кодировку содержимого. Затем получаем текущее время, приводим к формату часы, минуты, секунды и сохраняем в переменную `$time`. Пользуясь оператором `print`, выводим небольшой поясняющий заголовок, после которого вставляем текущее время. И наконец, выводим содержимое глобальных массивов `$_POST` и `$_FILES`, которые нас интересуют в первую очередь, поскольку содержат данные, полученные сервером.

Дополнительные материалы по работе с плагином можно найти на сайте разработчика по адресу **http://malsup.com/jquery/form/**.

14.2. Плагин jQuery Validate

ЗАДАЧА

На веб-странице уже существует традиционная HTML-форма, которую модернизировали для отправки данных через AJAX с помощью плагина jQuery Form. Теперь в форме необходимо организовать предварительную проверку данных, вводимых пользователем.

Решение

Для решения этой задачи воспользуемся плагином jQuery Validate (листинг 14.2.1).

Листинг 14.2.1. Использование плагина jQuery Validate

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/form.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.validate.js" type="text/javascript"></script>
<script src="js/jquery.form.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#myForm").validate({
    rules: {
      Name: {
        required: true,
        minlength: 2,
        maxlength: 12
      },
      Email: {
        required: true,
        email: true
      },
      Url: {
        url: true
      },
      Colors: {
        required: true
      },
      Text: {
        required: true,
        maxlength: 24
      },
```

```
    Password: {
      required: true,
      rangelength: [6, 24]
    },
    ConfirmPassword: {
     required: true,
            equalTo: "#Password"
    },
    Avatar: {
      accept: "jpg|gif|png"
    },
    Agree: "required",
    Radio: "required",
    Examine: {
      required: true,
      equal: 13
    }
  },
  messages: {
    Name: {
      required: "Укажите свое имя!",
      minlength: "Не менее 2 символов",
      maxlength: "Не более 12 символов"
    },
    Email: {
      required: "Нужно указать email адрес",
      email: "e-mail введен неверно"
    },
    Url: {
      url: "Неправильно введен URL"
    },
    Colors: {
      required: "Нужно выбрать цвет"
    },
    Text: {
      required: "Вы забыли ввести текст",
      maxlength: "Не более 24 символов"
    },
    Password: {
      required: "Надо ввести пароль",
      rangelength: "От 6 до 24 символов"
    },
    ConfirmPassword: {
     required: "Подтвердите пароль",
            equalTo: "Подтверждение не принято"
    },
    Avatar: {
      accept: "Только jpg, gif или png"
    },
```

```
      Agree: "Нужно отметить чекбокс",
      Radio: "Нужно сделать выбор...",
      Examine: {
        required: "Надо решить этот пример",
        equal: "Вы в школе учились?"
      }
    },
    submitHandler: function(form) {
      $(form).ajaxSubmit({
        target: "#output",
        timeout: 3000
      });
    },
    errorPlacement: function(error, element) {
      error.appendTo(
        element.parent()
        .find("label[for='" + element.attr("name") + "']")
        .find("em")
      );
    }
  });
  $.validator.addMethod("equal", function(value, element, params) {
    return value == params;
  });
});
</script>
</head>
<body>
<div id="output"></div>
<form id="myForm" action="form.php" method="post">
<div>
<label for="Name">Имя: (*)<em></em></label>
<input id="Name" name="Name" type="text" value="" /><br />
<label for="Email">E-mail: (*)<em></em></label>
<input id="Email" name="Email" type="text" value="" /><br />
<label for="Url">URL:<em></em></label>
<input id="Url" name="Url" type="text" value="" /><br />
<label for="Colors">Select: (*)<em></em></label>
<select id="Colors" name="Colors">
  <option value=""></option>
  <option value="red">Красный</option>
  <option value="orange">Оранжевый</option>
  <option value="yellow">Желтый</option>
  <option value="green">Зеленый</option>
  <option value="blue">Голубой</option>
  <option value="deepblue">Синий</option>
  <option value="violet">Фиолетовый</option>
</select><br />
```

```
<label for="Text">Textarea: (*)<em></em></label>
<textarea id="Text" name="Text" rows="1" cols="20"></textarea><br />
<label for="Password">Пароль: (*)<em></em></label>
<input id="Password" name="Password" type="password" /><br />
<label for="ConfirmPassword">Еще раз: (*)<em></em></label>
<input id="ConfirmPassword" name="ConfirmPassword" type="password" /><br />
<label for="Avatar">Файл:<em></em></label>
<input id="Avatar" name="Avatar" type="file" /><br />
<label for="Agree">Чекбокс (*)<em></em></label>
<input class="cb" id="Agree" name="Agree" type="checkbox" /><br />
<label for="Radio">Радио: (*)<em></em></label>
<input class="rb" type="radio" name="Radio" value="1" />
<input class="rb" type="radio" name="Radio" value="2" />
<input class="rb" type="radio" name="Radio" value="3" /><br />
<label for="Examine">6 + 7 = (*)<em></em></label>
<input id="Examine" name="Examine" type="text" value="" /><br />
<input id="reset" type="reset" name="resetButton" value="Reset" />
<input class="img" type="image" name="submitButton" value="Submit"
src="img/form.gif" />
</div>
</form>
</body>
</html>
```

Обсуждение

Не стоит пугаться внушительного объема кода, который содержит листинг 14.2.1. Настолько большой пример приведен с практической целью — показать организацию проверки данных, подразумевающих ввод по определенным правилам. Например, правильность ввода e-mail-адреса, URL страницы и т. д.

Детальное знакомство с примером начнем с рассмотрения HTML-кода. Первый элемент, который нам встретится в `body`, — элемент `div` с идентификатором `#output`. Сюда будет выводиться ответ от сервера. Конечно в том случае, если данные из формы будут успешно отправлены. А пока про него можно забыть.

Дальше следует форма, содержащая практически все возможные элементы, которые только могут встречаться в формах. Элементы `input` типа `text`, `password`, `file`, `checkbox`, `radio`, `reset`, `image`, а также элементы `select` и `textarea`.

В таком виде, даже без подключения внешних JavaScript-файлов, эта форма вполне функциональна, и данные, введенные пользователем, будут отправлены на сервер. Но поскольку перед нами стоит задача проверки данных на стороне клиента, вернемся к разделу `head` веб-страницы и посмотрим, какие файлы потребуется подключить.

Первым идет файл стилевого оформления формы — css/form.css, за ним подключаем необходимые JavaScript-файлы. Файл библиотеки jQuery — js/jquery-1.5.2.min.js, файл плагина jQuery Validate — js/jquery.validate.js и файл плагина jQuery Form — js/jquery.form.js.

Примечание

Совместное использование плагинов jQuery Form и jQuery Validate совершенно не обязательно. Плагин jQuery Validate может функционировать и отдельно.

Вот мы добрались до самого главного и интересного — наиболее подробно разберем JavaScript-код, который будет обеспечивать проверку правильности заполнения полей формы.

Сначала отслеживаем момент готовности DOM, затем выбираем форму по ее идентификатору `#myForm` и связываем с ней функциональность плагина jQuery Validate. При этом мы передаем плагину некоторые опции — `rules`, `messages`, `submitHandler` и `errorPlacement`.

Опции `rules` и `messages` кажутся очень похожими. Значения обеих опций — объекты, названия свойств которых соответствуют атрибутам `name` полей формы. Значения этих свойств — тоже объекты (допустима строка, если правило проверки единственное), пары ключ/значение которых описывают правила проверки для полей с соответствующим именем (для опции `rules`), и сообщения, выводимые пользователю при нарушении этих правил (для опции `messages`).

Объяснение получилось довольно запутанным, но у нас есть листинг кода, изучая который можно все хорошо понять. Разберем для примера одно поле формы, самое первое, предназначенное для ввода имени. В атрибуте `name` этого элемента указано значение `Name`, поэтому соответствующим ему свойством объекта `rules` является свойство с именем `Name`. Его значение — объект, в котором определены три свойства: `required`, `minlength` и `maxlength`. Значение `true`, указанное в свойстве `required`, сообщит плагину о том, что поле `Name` обязательно для заполнения, значение `2`, указанное в свойстве `minlength`, ограничивает минимальную длину текста в этом поле двумя знаками, а значение `12` в свойстве `maxlength` задает максимальную длину текста.

Правила проверки для других полей создаются аналогично. Обратите внимание на свойства `email` или `url`. Значение `true`, переданное в этих опциях, заставит плагин проверять данные, введенные в эти поля, на соответствие форматам записи e-mail-адреса и URL страницы.

В `messages` все похоже — для каждого правила из опции `rules` в опции `messages` создается пара. Только значениями в этом случае являются текстовые сообщения, выводимые пользователю при некорректном заполнении соответствующего поля формы.

Отдельно от этого ряда стоит проверка поля `Examine`. Это поле обязательно для заполнения, о чем свидетельствует опция `required` со значением `true`. В это поле должно быть введено число `13`, указанное в свойстве `equal`. Но проблема в том, что такого метода проверки в плагине нет и, если мы не предпримем какие-либо действия, все наши труды не увенчаются успехом.

К счастью, в плагине предусмотрена возможность добавления собственных методов проверки. Воспользуемся этим и создадим нужный метод (листинг 14.2.1а).

Листинг 14.2.1а. Использование плагина jQuery Validate

```
$.validator.addMethod("equal", function(value, element, params) {
    return value == params;
  });
```

Первый аргумент, принимаемый методом `addMethod`, — строка, в которой передается имя добавляемого метода. В нашем случае это `equal`. Второй аргумент — функция, выполняющая необходимую проверку. Функция принимает три аргумента: аргумент `value` — это значение, введенное в поле формы, `element` — объект, характеризующий соответствующий HTML-элемент, и `params` — значение опции, с которой сравнивается введенное значение. Функция должна возвращать значения `true` или `false` в зависимости от результатов проверки. В примере из листинга 14.2.1а видно, что при совпадении значений аргументов `value` и `params` функция вернет `true`, в противном случае — `false`.

Метод `addMethod` может принимать еще и третий, необязательный аргумент, который может быть как строкой, так и функцией и определяет сообщение по умолчанию, выводимое пользователю при несоблюдении условий проверки.

Теперь вернемся к еще не рассмотренным опциям — `submitHandler` и `errorPlacement`.

Функция, которую мы определим в опции `submitHandler`, будет вызвана при наступлении события `submit` формы в том случае, если все поля формы заполнены в соответствии с указанными правилами. В качестве единственного аргумента функция принимает форму. Внутри функции для этой формы вызываем метод `ajaxSubmit()` уже знакомого нам плагина jQuery Form и передаем ему некоторые настройки. Например, в опции `target` указываем значение `#output` — идентификатор того элемента `div`, куда будет выводиться ответ от сервера.

Еще одна опция, присутствующая в этом примере, — `errorPlacement`. В ней можно определить функцию, которая будет обрабатывать ошибки проверки полей формы. Функция принимает два аргумента — `error` и `element`. Аргумент `error` содержит текст сообщения о произошедшей ошибке, а аргумент `element` — это объект, характеризующий элемент, в котором ошибка произошла.

Листинг 14.2.1б. Использование плагина jQuery Validate

```
function(error, element) {
  error.appendTo(element.parent()
       .find("label[for='" + element.attr("name") + "']")
       .find("em"));
}
```

Функция, которую мы определили в опции `errorPlacement`, приведена еще раз в листинге 14.2.1б. Внутри функции мы вызываем метод `appendTo()` библиотеки jQuery. Метод `appendTo()` применен к `error` (это сообщение об ошибке), а аргументом метода `appendTo()` является выражение, которое помогает обнаружить тот са-

мый элемент, куда необходимо вставить сообщение об ошибке. Читайте внимательно! Сначала, с помощью метода `parent()` находим родительский элемент по отношению к тому, что был передан в аргументе `element`, затем вызываем метод `find()`, чтобы обнаружить элемент `label` с атрибутом `for`, значение которого соответствует атрибуту `name` того элемента, где произошла ошибка (если внимательно посмотрите HTML-код формы, то обнаружите, что они одинаковые), и наконец снова используем метод `find()` — получаем элемент `em`, находящийся внутри только что найденного элемента `label`. Именно в элемент `em` и будет вставлен текст сообщения об ошибке.

Конечно, мы разобрали не все возможные варианты применения плагина jQuery Validate, но он обладает настолько большим набором разнообразных опций, дополнительных методов, что для полного описания потребовалась бы отдельная книга. Более подробную информацию по работе с плагином можно обнаружить на официальном сайте jQuery по адресу **http://docs.jquery.com/Plugins/Validation**. Много примеров доступно на сайте разработчика плагина по адресу **http://jquery.bassistance.de/validate/demo/**.

14.3. Плагин jQuery Uploadify

ЗАДАЧА

Посетителям веб-страницы необходимо предоставить возможность загрузки файлов на сервер. Требуется реализовать выбор для загрузки не только одного, но и сразу нескольких файлов. Процесс загрузки файлов должен отображаться с помощью шкалы. Веб-страница не должна перезагружаться.

Решение

Для решения этой задачи используем плагин jQuery Uploadify (листинг 14.3.1).

Листинг 14.3.1. Использование плагина jQuery Uploadify

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/uploadify.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/swfobject.js" type="text/javascript"></script>
<script src="js/jquery.uploadify.v2.1.4.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#fileInput").uploadify({
    "uploader": "scripts/uploadify.swf",
```

```
    "script": "scripts/uploadify.php",
    "cancelImg": "img/cancel.png",
    "folder": "uploads"
  });
  $("a").click(function(){
    $("#fileInput").uploadifyUpload();
  });
});
</script>
</head>
<body>
<div>
  <input id="fileInput" name="fileInput" type="file" />
  <a href="#">Загрузить файл</a>
</div>
</body>
</html>
```

Обсуждение

Листинг 14.3.1 иллюстрирует самый простой вариант работы с плагином jQuery Uploadify. Давайте разберем код подробнее.

HTML-разметка несложная — элемент `input` типа `file` с атрибутами `id` и `name`, и элемент `a`, при щелчке мышью на котором будет начинаться загрузка файла.

Посмотрим на раздел `head` нашей веб-страницы. Здесь нам необходимо подключить следующие файлы: файл стилевого оформления css/uploadify.css, js/jquery-1.5.2.min.js — файл библиотеки jQuery, js/swfobject.js — вспомогательный файл для работы с swf-объектами (плагин задействует flash-технологии) и js/jquery.uploadify.v2.1.4.js — файл самого плагина jQuery Uploadify.

После подключения всех необходимых файлов мы в состоянии реализовать возможности плагина на нашей веб-странице. Посмотрим JavaScript-код и узнаем, как именно это делается.

Находим нужный нам элемент `input` по его идентификатору `#fileInput` и применяем к нему метод `uploadify(options)`. Сразу же передаем плагину объект с минимально необходимым количеством настроек.

В опции `uploader` указываем путь (относительный или абсолютный) к файлу `uploader.swf`.

В опции `script` — указываем относительный или абсолютный путь к файлу, который занимается обработкой загружаемых данных на сервере.

Опция `cancelImg` содержит путь к картинке, при щелчке на которой можно отменить загрузку.

В опции `folder` определяем путь к папке, куда будут загружаться файлы. Обратите внимание, что завершающего слэша быть не должно. Кроме того, может потребоваться выставить для этой папки права на запись.

Осталось связать с элементом `a` обработчик события `click` и при наступлении этого события отыскать тот самый элемент `input`, с которым мы связали функциональность плагина jQuery Uploadify, и вызвать для него метод `uploadifyUpload()`. Метод `uploadifyUpload()` инициализирует загрузку выбранного файла (или файлов).

Если вы решите испытать в действии пример из листинга 14.3.1 и не обнаружите ожидаемого вами традиционного элемента `input` типа `file` с кнопкой **Обзор** — не удивляйтесь. Этот элемент заменен на создаваемую плагином по умолчанию flash-кнопку **Browse**. Ну, а если такая кнопка пришлась вам не по вкусу (или цвету), ее можно заменить любым изображением (листинг 14.3.2).

Листинг 14.3.2. Использование плагина jQuery Uploadify

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/uploadify.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/swfobject.js" type="text/javascript"></script>
<script src="js/jquery.uploadify.v2.1.4.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#fileInput").uploadify({
    "uploader": "scripts/uploadify.swf",
    "script": "scripts/uploadify.php",
    "cancelImg": "img/cancel.png",
    "folder": "uploads",
    "multi": true,
    "buttonImg": "img/browse.jpg",
    "width": 130,
    "height": 32,
    "queueSizeLimit": 3
  });
  $("a:first").click(function(){
    $("#fileInput").uploadifyUpload();
  });
  $("a:last").click(function(){
    $("#fileInput").uploadifyClearQueue();
  });
});
</script>
</head>
<body>
<div>
```

```
  <input id="fileInput" name="fileInput" type="file" />
  <a href="#">Загрузить файл(ы)</a> <a href="#">Очистить очередь</a>
</div>
</body>
</html>
```

Для того чтобы определить свою кнопку, необходимо указать путь к ней в опции `buttonImg` и задать ее размеры в опциях `width` и `height`.

Но в JavaScript-коде из листинга 14.3.2, кроме опций, связанных с кнопкой выбора файлов для загрузки, мы определили еще две дополнительные опции.

Значение `true`, переданное в опции `multi`, позволит выбирать для загрузки сразу несколько файлов, но в очередь для загрузки попадут не более трех из них, за что отвечает опция `queueSizeLimit`.

В HTML-разметке листинга 14.3.2 мы добавили еще один элемент `a`, для которого в JavaScript-коде определили функцию-обработчик события `click`. При наступлении события будет найден элемент с идентификатором `#fileInput`, для которого будет вызван метод `uploadifyClearQueue()`. Этот метод очищает очередь файлов.

Настала пора поговорить о том, какие варианты предварительной проверки файлов перед загрузкой может предоставить нам плагин jQuery Uploadify. Такие возможности есть, и пример их использования продемонстрирован в листинге 14.3.3.

Листинг 14.3.3. Использование плагина jQuery Uploadify

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/uploadify.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/swfobject.js" type="text/javascript"></script>
<script src="js/jquery.uploadify.v2.1.4.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#fileInput").uploadify({
    "uploader": "scripts/uploadify.swf",
    "script": "scripts/uploadify.php",
    "cancelImg": "img/cancel.png",
    "folder": "uploads",
    "auto": true,
    "sizeLimit": 2097152,
    "fileDesc": "*.gif, *.jpg и *.png",
    "fileExt": "*.gif;*.jpg;*.png"
  });
});
```

```
</script>
</head>
<body>
<div>
  <input id="fileInput" name="fileInput" type="file" />
</div>
</body>
</html>
```

Сначала обратим внимание на HTML-разметку в листинге 14.3.3. Здесь присутствует только элемент `input` типа `file` и нет ссылки, с помощью которой мы начинали загрузку файлов.

Не волнуйтесь — она нам здесь не понадобится. Вместо этого мы назначим опцию `auto`, в которой передаем значение `true`. Теперь достаточно только выбрать нужный файл и загрузка его на сервер начнется автоматически.

Что касается предварительной проверки файла, то мы имеем возможность ограничить размер, указав предельно допустимый в опции `sizeLimit` (в байтах). Кроме этого можно задать опцию `fileExt`, с помощью которой устанавливаются разрешенные для загрузки типы файлов. В примере из листинга 14.3.3 при щелчке мышью на кнопке **Browse** откроется окно выбора файлов, где будут отображаться только GIF-, JPG- и PNG-файлы. Опции `fileExt` и `fileDesc` должны использоваться совместно. Здесь можно написать любой текст, который будет отображен в выпадающем списке "Тип файлов" в открывшемся окне выбора.

Осталось запомнить самое главное — если вы всерьез заботитесь о безопасности своих приложений, ни в коем случае не стоит полагаться только на предварительную проверку загружаемых файлов. Уделите самое серьезное внимание проверкам на стороне сервера.

Познакомимся еще с несколькими возможностями плагина jQuery Uploadify. Если вы уже испытывали примеры из предыдущих листингов, то смогли заметить, что шкала загрузки файла в любом случае появляется ниже кнопки **Browse**. А как быть, если кнопку выбора файлов хочется поместить в одном месте веб-страницы, а шкалу загрузки выводить совершенно в другом? Такая возможность существует, и в листинге 14.3.4 показано, как ее реализовать.

Листинг 14.3.4. Использование плагина jQuery Uploadify

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-3-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/uploadify.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/swfobject.js" type="text/javascript"></script>
```

```
<script src="js/jquery.uploadify.v2.1.4.js" type="text/javascript"></script>
<style type="text/css">
#fileQueue {
  width:400px; height:200px;
  border:1px solid #E5E5E5;
  padding:0; margin-bottom:10px;
  overflow:auto;
}
</style>
<script type="text/javascript">
$(function(){
  $("#fileInput").uploadify({
    "uploader": "scripts/uploadify.swf",
    "script": "scripts/uploadify.php",
    "cancelImg": "img/cancel.png",
    "folder": "uploads",
    "multi": true,
    "buttonImg": "img/browse.jpg",
    "width": 130,
    "height": 32,
    "queueID": "fileQueue",
    "simUploadLimit": 2
  });
  $("a").click(function(){
    $("#fileInput").uploadifyUpload();
  });
  $("a:last").click(function(){
    $("#fileInput").uploadifyClearQueue();
  });
});
</script>
</head>
<body>
<div id="fileQueue"></div>
<div>
  <input id="fileInput" name="fileInput" type="file" /><br />
  <a href="#">Загрузить файл(ы)</a> <a href="#">Очистить очередь</a>
</div>
</body>
</html>
```

Начнем с рассмотрения HTML-разметки из листинга 14.3.4. В основном вся разметка нам уже знакома, но кроме элемента `input` и двух элементов `a` появился элемент `div` с идентификатором `#fileQueue`. Его можно разместить в любом месте веб-страницы, и именно в нем мы будем отображать очередь стоящих на загрузку файлов. Сделать это можно, передав в опции `queueID` значение атрибута `id`, т. е. `fileQueue`.

Заодно познакомимся еще с одной опцией — `simUploadLimit`. По умолчанию она имеет значение `1`. В примере из листинга 14.3.4 мы указали в этой опции значение `2`, разрешив, таким образом, параллельную загрузку сразу двух файлов из очереди.

Нам осталось рассмотреть еще одну очень интересную возможность плагина jQuery Uploadify — его способность реагировать на различные события. Пример использования опций, в которых определяются callback-функции, вызываемые при наступлении определенных событий, приведен в листинге 14.3.5.

Листинг 14.3.5. Использование плагина jQuery Uploadify

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-14-3-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/uploadify.css" rel="stylesheet" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/swfobject.js" type="text/javascript"></script>
<script src="js/jquery.uploadify.v2.1.4.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#fileInput").uploadify({
    "uploader": "scripts/uploadify.swf",
    "script": "scripts/uploadify.php",
    "cancelImg": "img/cancel.png",
    "folder": "uploads",
    "onSelect": function(e,id,obj){
      $("p").html("Событие: " + e.type +
            "<br />Идентификатор: " + id +
            "<br />Имя файла: " + obj.name +
            "<br />Размер файла: " + obj.size +
            " байт<br />Дата создания: " + obj.creationDate.date +
            "/" + obj.creationDate.month +
            "/" + obj.creationDate.fullYear +
            "<br />Дата модификации: " + obj.modificationDate.date +
            "/" + obj.modificationDate.month +
            "/" + obj.modificationDate.fullYear +
            "<br />Тип файла: " + obj.type);
    },
    "onProgress": function(e,id,obj,data){
      $("p").html("Событие: " + e.type +
            "<br />Идентификатор: " + id +
            "<br />Имя файла: " + obj.name +
            "<br />Размер файла: " + obj.size +
            " байт<br />Тип файла: " + obj.type +
```

```
            "<br />Процент загрузки: " + data.percentage +
            "%<br />Загружено : " + data.bytesLoaded +
            " байт<br />Текущая скорость загрузки: " + data.speed + " KB/s");
    },
    "onComplete": function(e,id,obj,response,data){
      $("p").html("Событие: " + e.type +
            "<br />Идентификатор: " + id +
            "<br />Имя файла: " + obj.name +
            "<br />Путь к файлу: " + obj.filePath +
            "<br />Размер файла: " + obj.size +
            " байт<br />Тип файла: " + obj.type +
            "<br />Ответ сервера: " + response +
            "<br />Осталось в очереди: " + data.fileCount +
            " файлов<br />Средняя скорость загрузки: " + data.speed + " KB/s");
    }
  });
  $("a").click(function(){
    $("#fileInput").uploadifyUpload();
  });
});
</script>
</head>
<body>
<div>
  <input id="fileInput" name="fileInput" type="file" />
  <a href="#">Загрузить файл</a>
</div>
<p></p>
</body>
</html>
```

Начнем сразу с рассмотрения JavaScript-кода из листинга 14.3.5, а точнее с трех новых опций, передаваемых плагину, которые мы определили в объекте настроек, — `onSelect`, `onProgress` и `onComplete`.

В опции `onSelect` определяется функция, которая будет вызвана при выборе файла для загрузки. Эта функция принимает три аргумента — объект события, строку с шестизначным идентификатором выбранного файла, и объект, содержащий некоторую информацию о выбранном для загрузки файле.

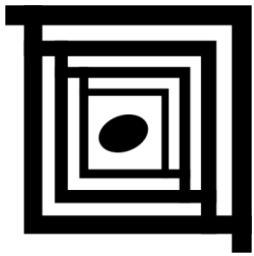
В опции `onProgress` определяется функция, вызываемая постоянно во время процесса загрузки файлов на сервер. Функция принимает четыре аргумента — объект события, строку с шестизначным идентификатором файла, объект, содержащий информацию о загружаемом файле, и объект с информацией о самом процессе загрузки.

Наконец, в опции `onComplete` определяется функция, которая будет вызвана по окончании загрузки файла на сервер. Функция принимает пять аргументов — объект события, строку с шестизначным идентификатором файла, объект, содержащий

информацию о загруженном файле, данные, возвращенные в ответе сервера, и объект с информацией о процессе загрузки.

Если вы внимательно изучите пример из листинга 14.3.5, то при выборе файла сможете легко получать такие данные, как имя, размер и тип файла, дата его создания и модификации. Вы сможете отображать ход загрузки не только в процентном выражении, но и указывать число загруженных байтов, а также текущую скорость загрузки. А по окончании процесса сможете показать среднюю скорость загрузки или, например, путь к файлу на сервере.

Мы рассмотрели наиболее распространенные возможности плагина jQuery Uploadify, но, конечно, не все. Более подробную информацию по применению плагина можно найти на сайте разработчиков по адресу **http://www.uploadify.com/**. Там же можно загрузить самую последнюю версию плагина.



ГЛАВА 15

Фотогалерея для сайта

В настоящее время трудно себе представить, например, достойный интернет-магазин, который не выставляет "товар лицом". Ведь потенциальный покупатель, конечно, может прочитать много информации о товаре, но один из самых важных факторов, побуждающих к покупке, — качественное изображение товара. Ряд хороших фотографий, сделанных с разных ракурсов, порой расскажут о товаре больше, чем несколько страниц текста.

Демонстрацию изображений на сайте обычно организуют с помощью плагинов фотогалерей. Для библиотеки jQuery подобных плагинов написано немало, мы разберем принципы их работы на примере популярной фотогалереи FancyBox.

15.1. Фотогалерея FancyBox

ЗАДАЧА

Нужно организовать фотогалерею на веб-странице.

Решение

Для решения этой задачи мы используем очень популярный плагин FancyBox (листинг 15.1.1).

Листинг 15.1.1. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
```

```
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
img { padding:2px; border:1px solid #ccc; margin:4px; }
a { outline:none; }
</style>
<script type="text/javascript">
$(function(){
  $("a").fancybox();
});
</script>
</head>
<body>
<p>
  <a title="Россия - Греция 1:0" href="img/russia-greece.jpg">
    <img src="img/russia-greece-min.jpg" alt="Россия - Греция 1:0" />
  </a>
  <a title="Россия - Швеция 2:0" href="img/russia-sweden.jpg">
    <img src="img/russia-sweden-min.jpg" alt="Россия - Швеция 2:0" />
  </a>
  <a title="Россия - Голландия 3:1" href="img/russia-holland.jpg">
    <img src="img/russia-holland-min.jpg" alt="Россия - Голландия 3:1" />
  </a>
</p>
</body>
</html>
```

Обсуждение

Итак, для применения плагина нам понадобится сначала подключить файл библиотеки jQuery js/jquery-1.5.2.min.js и файл плагина js/jquery.fancybox-1.3.4.js. Также потребуется файл стилевого оформления плагина js/jquery.fancybox-1.3.4.css. Если предполагаются анимационные эффекты при открытии и закрытии полноразмерных изображений, то необходимо подключить еще и файл — jquery.easing-1.3.pack.js — плагин, который обеспечивает поддержку анимации, поскольку по умолчанию поддерживается только пара эффектов. Хотите, чтобы полноразмерные изображения в галерее управлялись колесиком мыши? Тогда подключите файл jquery.mousewheel-3.0.4.pack.js.

> ***Примечание***
>
> Все дополнительные файлы, как и файлы примеров, находятся в соответствующей папке на компакт-диске, поставляемом с книгой.

В листинге 15.1.1 приведен наиболее простой пример работы с плагином FancyBox — его функциональность применена к одному-единственному изображению. При щелчке на миниатюре в отдельном слое открывается полноразмерное изображение.

Обратите внимание на HTML-разметку — изображение-миниатюра заключено в элемент `a`, атрибут `href` которого указывает на полноразмерное изображение. Содержимое атрибута `title` — это подпись к полноразмерному изображению.

В JavaScript-коде тоже нет ничего сложного — с помощью селектора jQuery выбирается необходимый элемент, к которому применяется метод `fancybox()`.

Результат выполнения кода показан на рис. 15.1.

Плагин FancyBox способен принимать пользовательские настройки, которые передаются плагину в виде объекта, содержащего пары ключ/значение (листинг 15.1.2).

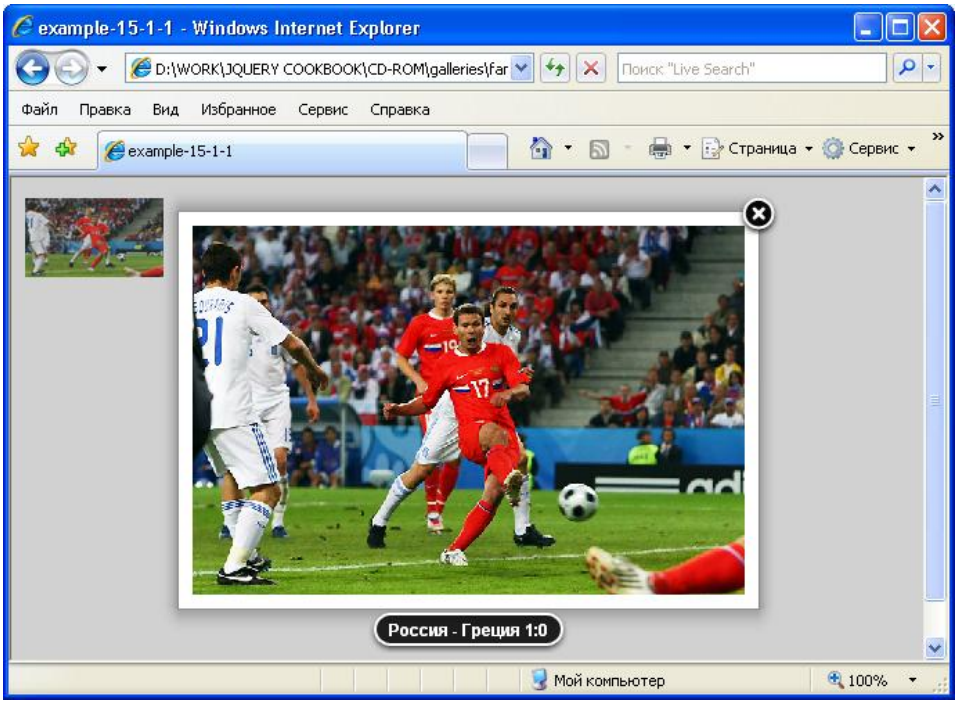


Рис. 15.1. Результат использования плагина FancyBox

Листинг 15.1.2. Использование плагина FancyBox

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
```

```
img { padding:2px; border:1px solid #ccc; margin:4px; }
a { outline:none; }
</style>
<script type="text/javascript">
$(function(){
  $("a[rel^='group']").fancybox({
    "cyclic": true,
    "speedIn": 2000,
    "speedOut": 1000,
    "overlayColor": "#f90",
    "overlayOpacity": 0.7,
    "transitionIn": "elastic",
    "transitionOut": "elastic"
  });
});
</script>
</head>
<body>
<p>
  <a rel="group" title="Россия - Швеция 2:0" href="img/russia-sweden.jpg">
    <img src="img/russia-sweden-min.jpg" alt="" />
  </a>
  <a rel="group" title="Гус Хиддинк" href="img/guus-hiddink.jpg">
    <img src="img/guus-hiddink-min.jpg" alt="" />
  </a>
  <a rel="group" title="Роман Павлюченко" href="img/roman-pavlyuchenko.jpg">
    <img src="img/roman-pavlyuchenko-min.jpg" alt="" />
  </a>
  <a rel="group" title="Россия - Голландия 3:1" href="img/russia-holland.jpg">
    <img src="img/russia-holland-min.jpg" alt="" />
  </a>
</p>
</body>
</html>
```

Разбирая пример из листинга 15.1.2, сначала обратим внимание на HTML-разметку. Заметьте, что все элементы `a` имеют атрибут `rel` со значением `group`. Одинаковое значение атрибута `rel` в данном примере заставляет плагин "понять", что все эти изображения принадлежат к одной группе — из них нужно создать одну фотогалерею.

В JavaScript-коде мы выбрали все элементы `a`, атрибут `rel` которых начинается со значения `group`, и применили к ним метод `fancybox()`. Этому методу мы передаем объект с настройками, которые мы хотим сделать отличными от заданных по умолчанию.

Передав в опции `cyclic` значение `true`, мы добьемся того, что полноразмерные изображения будут показываться по кругу, т. е. после последнего изображения галереи будет снова отображено первое.

В опциях `speedIn` и `speedOut` передаем время, за которое должна выполняться анимация при открытии/закрытии полноразмерных изображений.

Свойства `overlayColor` и `overlayOpacity` определяют соответственно цвет и прозрачность слоя, перекрывающего основное содержимое страницы.

А `transitionIn` и `transitionOut` — анимационные эффекты. В примере мы выбрали эффект `elastic`, доступный без подключения дополнительных файлов.

Названия и описания всех возможных опций приведены в табл. 15.1.

Таблица 15.1. *Опции плагина FancyBox*

Опция	Описание
`padding`	Определяет ширину рамки вокруг полноразмерного изображения. Значение по умолчанию — 10
`margin`	Определяет поля — минимальное расстояние между полноразмерным изображением (включая рамку) и областью просмотра. Значение по умолчанию — 20
`opacity`	По умолчанию — `false`. Если установить значение `true`, будет использоваться изменение прозрачности изображения во время анимации при показе/скрытии области просмотра полноразмерного изображения при использовании эффекта `elastic`
`modal`	По умолчанию — `false`. Если установить значение `true`, то для опции `overlayShow` будет установлено значение `true`, а для опций `hideOnOverlayClick`, `hideOnContentClick`, `enableEscapeButton`, `showCloseButton` — значение `false`. Таким образом, открытую область полноразмерного изображения можно будет закрыть только программно
`ciclyc`	Если установить значение `true`, галерея станет замкнутой, т. е. после показа последнего изображения галереи будет показано первое. По умолчанию установлено значение `false`
`scrolling`	Устанавливает css-свойство `overflow` для создания или скрытия полосы прокрутки. Принимает значения `'auto'`, `'yes'` или `'no'`
`width`	Опция определяет ширину, когда содержимое загружается с помощью AJAX-запроса или через `iframe`. По умолчанию задано значение 560
`height`	Опция определяет высоту, когда содержимое загружается через AJAX-запрос или `iframe`. Значение по умолчанию — 340
`autoScale`	Значение по умолчанию для этой опции — `true`. В этом случае область просмотра полноразмерного изображения в случае необходимости масштабируется, чтобы уместиться в окне просмотра
`autoDimensions`	Значение по умолчанию — `true`. Используется для содержимого, загруженного с помощью AJAX-запросов, или содержимого, размещенного на текущей веб-странице. Определяет размеры загружаемых элементов. Убедитесь, что размеры для них явно определены, иначе возможен некорректный результат
`senterOnScroll`	Полноразмерное изображение будет центрироваться при прокрутке страницы, если значением этой опции будет `true`. Это значение задано по умолчанию

Таблица 15.1 *(продолжение)*

Опция	Описание
`ajax`	Объект, в котором могут быть переданы настройки AJAX-запроса. В том числе могут быть переопределены опции `error` и `success`
`swf`	Параметры для передачи swf-объекту. По умолчанию — `{wmode: 'transparent'}`
`hideOnOverlayClick`	Значение `true`, установленное по умолчанию для этой опции, скрывает полноразмерное изображение при щелчке на дополнительном слое
`hideOnContentClick`	Значение `true`, установленное по умолчанию для этой опции, скрывает полноразмерное изображение при щелчке на нем. Установка значения `false` может потребоваться, когда в области просмотра содержимое выводится через `iframe` или загружается с помощью AJAX-запроса
`overlayShow`	По умолчанию имеет значение `true` — показывать дополнительный слой между областью просмотра полноразмерных изображений и основным содержимым веб-страницы
`overlayOpacity`	Прозрачность дополнительного слоя. Принимает значения от 0 до 1 с шагом 0.1. Значение по умолчанию — 0.3
`overlayColor`	Определяет цвет дополнительного слоя. По умолчанию установлено значение `#666`
`titleShow`	По умолчанию — `true`. Если установить значение `false`, подпись к изображению отображаться не будет
`titlePosition`	Опция определяет расположение подписи к изображению. Может принимать значения `'outside'`, `'inside'` или `'over'`. По умолчанию установлено значение `'outside'`
`titleFormat`	Функция обратного вызова для настройки подписи к изображению. С ее помощью можно задать любую html-разметку для счетчика изображений и навигации
`transitionIn`, `transitionOut`	Опция определяет тип анимационных эффектов при открытии/закрытии полноразмерных изображений. Может принимать значения `'elastic'`, `'fade'` or `'none'`. По умолчанию установлено значение `'fade'`
`speedIn`, `speedOut`	Время выполнения анимации при открытии/закрытии полноразмерного изображения. Указывается в миллисекундах. Значение по умолчанию — 300
`changeSpeed`	Время выполнения анимации при изменении размеров области просмотра полноразмерных изображений при переходе от одного к другому. Указывается в миллисекундах. Значение по умолчанию — 300
`changeFade`	Время затухания полноразмерного изображения при переходе от одного к другому. Значение по умолчанию — `'fast'`
`easingIn`, `easingOut`	В этих опциях можно указать названия анимационных эффектов, применяемых при показе/скрытии области просмотра полноразмерных изображений. Использование этих опций подразумевает подключение файла jquery.easing.js. Дополнительную информацию об эффектах см. на **http://gsgd.co.uk/sandbox/jquery/easing/**
`showCloseButton`	Значение по умолчанию — `true`. Можно не отображать кнопку закрытия полноразмерного изображения, установив значение `false`

Таблица 15.1 *(окончание)*

Опция	Описание
showNavArrows	Значение по умолчанию — true. Можно не отображать навигационные кнопки для перехода к предыдущему/следующему изображению, установив значение false
enableEscapeButton	По умолчанию установлено значение true — при нажатии на клавишу <Esc> полноразмерное изображение будет закрыто. Можно отменить такое поведение, установив значение false
onStart	В этой опции можно определить пользовательскую функцию, которая будет вызываться перед открытием полноразмерного изображения
onCancel	В этой опции можно определить пользовательскую функцию, которая будет вызываться при отмене загрузки полноразмерного изображения
onComplete	В этой опции можно определить пользовательскую функцию, которая будет вызываться сразу после открытия полноразмерного изображения
onCleanup	В этой опции можно определить пользовательскую функцию, которая будет вызываться непосредственно перед закрытием полноразмерного изображения
onClosed	В этой опции можно определить пользовательскую функцию, которая будет вызываться после закрытия полноразмерного изображения

Пример, приведенный в листинге 15.1.3, демонстрирует возможность применения пользовательских функций, определенных в опциях onStart, onComplete и onClosed.

Листинг 15.1.3. Использование плагина FancyBox

```
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
img { padding:2px; border:1px solid #ccc; margin:4px; }
a { outline:none; }
</style>
<script type="text/javascript">
$(function(){
  $("a").fancybox({
    "titlePosition": "over",
    "onStart": function(){ alert('START'); },
    "onComplete": function(){ alert('COMPLETE'); },
    "onClosed":        function(){ alert('CLOSED'); }
  });
});
```

```
</script>
</head>
<body>
<p>
  <a title="Юрий Жирков" href="img/yuri-zhirkov.jpg">
    <img src="img/yuri-zhirkov-min.jpg" alt="Юрий Жирков" />
  </a>
</p>
</body>
</html>
```

HTML-разметка в листинге 15.1.3 вполне обычная — миниатюра изображения, заключенная в тег `a`, атрибут `href` которого содержит ссылку на полноразмерное изображение.

Нас больше интересует JavaScript-код. Выбрав элемент `a` и применив к нему функциональность плагина, мы передаем несколько настроек. В опциях `onStart`, `onComplete` и `onClosed` мы написали небольшие пользовательские функции, которые будут всего лишь выводить сообщения в окне предупреждения при наступлении соответствующего события.

При щелчке на миниатюре, но перед открытием полноразмерного изображения появится сообщение START. Сразу после открытия изображения увидим сообщение COMPLETE. А после того, как полноразмерное изображение будет закрыто — сообщение CLOSED.

Аналогично используются опции `onCancel` и `onCleanup`.

Пример из листинга 15.1.4 демонстрирует еще некоторые возможности плагина FancyBox. Мы попробуем программно управлять фотогалереей, пользуясь предоставляемыми ею методами, чтобы "спрятать" целый набор полноразмерных изображений за единственным элементом.

Листинг 15.1.4. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<script type="text/javascript">
$(function(){
  $("#open").click(function() {
    $.fancybox([
      "img/yuri-zhirkov.jpg",
      "img/roman-pavlyuchenko.jpg",
```

```
      {
        "href": "img/guus-hiddink.jpg",
        "title": "Гус Хиддинк"
      }],
      {
      "titlePosition": "over",
      "overlayShow": false,
      "type": "image"
      });
  });
  $("#prev").click($.fancybox.prev);
  $("#next").click($.fancybox.next);
  $("#close").click($.fancybox.close);
});
</script>
</head>
<body>
<p><a id="open" href="#">Открыть</a></p>
<p><a id="prev" href="#">Предыдущая</a></p>
<p><a id="next" href="#">Следующая</a></p>
<p><a id="close" href="#">Закрыть</a></p>
</body>
</html>
```

Посмотрим на HTML-разметку из листинга 15.1.4. Четыре элемента `a` будут играть роль элементов управления — открывать и закрывать полноразмерные изображения, а также осуществлять переход от одного изображения галереи к другому.

В JavaScript-коде мы выбираем необходимые элементы `a` с помощью селектора и связываем с ними обработчики события `click`.

Сначала займемся обработчиком для элемента `a` с идентификатором `open`. Данные об изображениях, из которых должна состоять галерея, передаются в первом аргументе — массиве. Каждый элемент массива может быть либо строкой, либо объектом. В первом случае в строке указывается адрес полноразмерного изображения, во втором — объект со свойствами `href` и `title`. Свойство `href` — адрес изображения, свойство `title` — комментарий к нему. Вторым аргументом передается объект с настройками плагина, точно так же, как это делалось и в предыдущих примерах. В рассматриваемом примере мы переопределили опцию `titlePosition`, установили для опции `overlayShow` значение `false`, чтобы иметь возможность щелкать по ссылкам, и принудительно установили тип содержимого `image` в опции `type`.

Таким образом, мы создали полноценную галерею, "спрятав" ее за одной-единственной ссылкой.

Организовать управление галереей извне тоже просто — необходимо связать обработчик события `click` на выбранном элементе с соответствующим методом плагина. Если вновь взглянуть на код из листинга 15.1.4, то можно увидеть, что обработчик события `click` для элемента `a` с идентификатором `next` вызывает метод

`$.fancybox.next`. Аналогичным образом вызываются методы `$.fancybox.prev` и `$.fancybox.close`.

Следующий пример использования плагина демонстрирует загрузку flash-содержимого в область просмотра. В примере мы попробуем загрузить в область просмотра видеоклип с YouTube (листинг 15.1.5).

Листинг 15.1.5. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<script type="text/javascript">
$(function(){
  $("#youtube").click(function() {
    $.fancybox({
      "padding": 10,
      "autoScale": false,
      "transitionIn": "none",
      "transitionOut": "none",
      "title": this.title,
      "width": 680,
      "height": 495,
      "href": this.href.replace(new RegExp("watch\\?v=", "i"), 'v/'),
      "type": "swf",
      "swf": {
        "wmode": "transparent",
        "allowfullscreen": "true"
      }
    });
  return false;
  });
});
</script>
</head>
<body>
<p><a id="youtube" title="Безумный пенальти"
href="http://www.youtube.com/watch?v=mof6hZ-8Glw&feature=related">Клип на
YouTube</a></p>
</body>
</html>
```

HTML-разметка, приведенная в листинге 15.1.5, описывает элемент `a` с идентификатором `youtube`. Атрибут `href` содержит адрес видеоклипа, а в атрибут `title` мы поместили заголовок.

В JavaScript-коде тоже нет ничего необычного — щелчком по ссылке мы будем вызывать плагин и передавать ему заданные настройки. Большинство настроек нам уже знакомо. Однако некоторые мы встретили впервые. Например, опция `title` позволяет принудительно задать заголовок к содержимому. Мы воспользовались тем, что `this` в нашем случае указывает на элемент, по которому был совершен щелчок. Следовательно, в `this.title` находится содержимое атрибута `title` этой ссылки.

Названия и описания дополнительных опций приведены в табл. 15.2.

***Таблица 15.2.** Дополнительные опции плагина FancyBox*

Опция	Описание
`type`	Определяет тип содержимого. Может принимать значения `'image'`, `'ajax'`, `'iframe'`, `'swf'` или `'inline'`
`href`	Назначает источник содержимого
`title`	Определяет заголовок
`content`	Устанавливает содержимое (может использоваться только HTML)
`orig`	Задает объект, чье положение и размеры будут использоваться при открытии/закрытии галереи с использованием эффекта `elastic`
`index`	Индекс изображения, которое будет показано при открытии галереи при "ручном" создании. См. пример в листинге 15.1.4

Познакомимся с еще одной интересной возможностью плагина FancyBox — возможностью загрузки в `iframe` данных из внешнего источника. Пример приведен в листинге 15.1.6.

Листинг 15.1.6. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
img { padding:2px; border:1px solid #ccc; margin:4px; }
a { outline:none; }
</style>
```

```
<script type="text/javascript">
$(function(){
  $("a:first").fancybox({
    "width": "75%",
    "height": "75%",
    "autoScale": false,
    "transitionIn": "none",
    "transitionOut": "none",
    "type": "iframe"
  });
});
</script>
</head>
<body>
<p>
  <a href="http://www.google.ru">
    <img src="img/google-min.jpg" alt="Google" />
  </a>
</p>
</body>
</html>
```

К HTML-разметке, приведенной в листинге 15.1.6, мы уже привыкли — миниатюрное изображение внутри элемента `a`. В атрибуте `href` указан адрес поисковой системы Google.

Пример довольно простой, чтобы разбирать его детально. Обратить внимание стоит на опцию `type` со значением `iframe`.

FancyBox — очень популярная фотогалерея, но пионером этого направления, пожалуй, можно назвать галерею LightBox. В следующем примере (см. листинг 15.1.7) мы попробуем с помощью плагина FancyBox придать фотогалерее стиль оформления, характерный для LightBox, и попутно показать действие опции `titleFormat`.

Листинг 15.1.7. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
img { padding:2px; border:1px solid #ccc; margin:4px; }
a { outline:none; }
```

```
#tip-title { text-align:left; }
#tip-title strong { display:block; margin-right:80px; }
#tip-title span { float:right; }
#tip-title img { border:none; }
</style>
<script type="text/javascript">
$(function(){
  function formatTitle(title, currentArray, currentIndex, currentOpts) {
    return '<div id="tip-title"><span><a href="javascript:;"
onclick="$.fancybox.close();"><img src="img/closelabel.gif" /></a></span>' +
(title && title.length ? '<strong>' + title + '</strong>' : '' ) + 'Изображение
' + (currentIndex + 1) + ' из ' + currentArray.length + '</div>';
  }
  $("a[rel='group']").fancybox({
    "showCloseButton": false,
    "titlePosition": "inside",
    "titleFormat": formatTitle
  });
});
</script>
</head>
<body>
<p>
  <a rel="group" title="Россия - Швеция 2:0" href="img/russia-sweden.jpg">
    <img src="img/russia-sweden-min.jpg" alt="" />
  </a>
  <a rel="group" title="Гус Хиддинк" href="img/guus-hiddink.jpg">
    <img src="img/guus-hiddink-min.jpg" alt="" />
  </a>
  <a rel="group" title="Роман Павлюченко" href="img/roman-pavlyuchenko.jpg">
    <img src="img/roman-pavlyuchenko-min.jpg" alt="" />
  </a>
  <a rel="group" title="Россия - Голландия 3:1" href="img/russia-holland.jpg">
    <img src="img/russia-holland-min.jpg" alt="" />
  </a>
</p>
</body>
</html>
```

Начнем разбираться сразу с JavaScript-кода, поскольку HTML-разметка, приведенная в листинге 15.1.7, уже хорошо знакома. Более того, мы уже умеем с легкостью выбирать элементы, с которыми связываем плагин, и передавать плагину различные настройки. Остановимся только на одной опции, которая ранее в примерах не использовалась, — `titleFormat`. В ней можно определить пользовательскую функцию, с помощью которой описывается формат комментариев к изображению.

В JavaScript-коде мы определили такую функцию `formatTitle(title, currentArray, currentIndex, currentOpts)`, которая принимает в качестве аргументов собственно

заголовок, массив изображений, индекс текущего изображения и текущие настройки, а возвращает эта функция HTML-разметку.

Обратите внимание на эту функцию, а также на то, что в теги `style` мы добавили некоторые CSS-правила для элемента с идентификатором `tip-title` и некоторых вложенных в него элементов.

Итак, с помощью дополнительной HTML-разметки, заданной в функции `formatTitle`, и некоторых CSS-правил для нее, мы добились, что наша FancyBox визуально практически ничем не отличается от галереи LightBox.

Разберем еще один полезный пример (листинг 15.1.8), демонстрирующий возможность загрузки в область просмотра галереи содержимого, которое находится на той же странице, но скрыто с помощью CSS-правила `display:none;`.

Листинг 15.1.8. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-8</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src=" js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
a { outline:none; }
#information {
  overflow:auto;
  width:500px; height:100px;
}
#outer {
  display:none;
}
</style>
<script type="text/javascript">
$(function(){
  $("#show").fancybox({
    "titlePosition": "inside",
    "transitionIn": "none",
    "transitionOut": "none"
  });
});
</script>
</head>
<body>
<p><a id="show" href="#information" title="Полезная информация">Информация</a></p>
<div id="outer"><div id="information">Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Etiam quis mi eu elit tempor facilisis id et neque. Nulla sit
```

```
amet sem sapien. Vestibulum imperdiet porta ante ac ornare. Nulla et lorem eu
nibh adipiscing ultricies nec at lacus. Cras laoreet ultricies sem, at blandit
mi eleifend aliquam. Nunc enim ipsum, vehicula non pretium varius, cursus ac
tortor. Vivamus fringilla congue laoreet. Quisque ultrices sodales orci, quis
rhoncus justo auctor in. Phasellus dui eros, bibendum eu feugiat ornare,
faucibus eu mi. Nunc aliquet tempus sem, id aliquam diam varius ac. Maecenas
nisl nunc, molestie vitae eleifend vel, iaculis sed magna. Aenean tempus lacus
vitae orci posuere porttitor eget non felis. Donec lectus elit, aliquam nec
eleifend sit amet, vestibulum sed nunc.</div></div>
</body>
</html>
```

Строго говоря, в HTML-разметке, приведенной в листинге 15.1.8, с помощью `display:none;` скрыт не сам элемент `div` с идентификатором `information`, а его родитель — `div#outer`. Дело здесь в том, что элемент, который загружается в FancyBox, должен иметь явно определенные размеры, иначе нужный результат можно просто не получить. Больше никаких хитростей в этом коде нет.

И наконец, продемонстрируем еще одну возможность плагина — загрузку содержимого с помощью AJAX-запроса. Пример приведен в листинге 15.1.9.

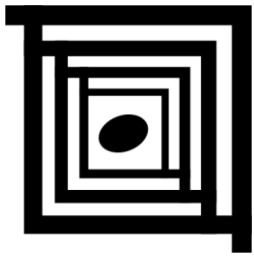
Листинг 15.1.9. Использование плагина FancyBox

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-15-1-9</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.fancybox-1.3.4.js" type="text/javascript"></script>
<link type="text/css" href="js/jquery.fancybox-1.3.4.css" rel="stylesheet" />
<style type="text/css">
a { outline:none; }
</style>
<script type="text/javascript">
$(function(){
  $("a:first").fancybox();
});
</script>
</head>
<body>
<p><a href="ajax.htm" title="Полезная информация">AJAX-запрос</a></p>
</body>
</html>
```

И HTML-разметка, и JavaScript-код, приведенные в листинге 15.1.9, настолько просты, что пояснять их не имеет смысла.

В заключение необходимо отметить, что разработчики плагина постоянно совершенствуют его, добавляя все новые интересные возможности.

Самую свежую версию плагина FancyBox всегда можно скачать на веб-сайте по адресу **http://fancybox.net**.



ГЛАВА 16

Несколько полезных плагинов

В этой главе будут рассмотрены несколько небольших, но очень полезных плагинов к библиотеке jQuery, которые наверняка когда-либо пригодятся в практической деятельности. Плагин jQuery Cookie поможет в установке, считывании и удалении `cookie`, jQuery Timers позволит легко создавать таймеры и управлять ими, а плагин jQuery Cluetip реализует красивые всплывающие подсказки на веб-странице.

16.1. jQuery Cookie

ЗАДАЧА

Довольно часто возникает необходимость запоминать измененное состояние элементов DOM на веб-странице, и при ее обновлении сохранять это состояние для пользователя, инициировавшего данные изменения.

Решение

Для решения этой задачи используем очень простой, легкий и полезный плагин jQuery Cookie (листинг 16.1.1).

Листинг 16.1.1. Использование плагина jQuery Cookie

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.cookie.js" type="text/javascript"></script>
<script type="text/javascript">
```

```
$(function(){
  var options = { path: "/", expires: 10 };
  $("#setCookie").click(function(){
    $.cookie("testCookie", "Это значение cookie", options);
  });
  $("#getCookie").click(function(){
    alert($.cookie("testCookie"));
  });
  $("#delCookie").click(function(){
    $.cookie("testCookie", null, options);
  });
});
</script>
</head>
<body>
<button id="setCookie">Set Cookie</button>
<button id="getCookie">Get Cookie</button>
<button id="delCookie">Delete Cookie</button>
</body>
</html>
```

Обсуждение

Итак, для работы с плагином нам понадобится сначала подключить файл библиотеки jQuery и затем файл плагина jquery.cookie.js. HTML-разметка описывает три кнопки, с помощью которых мы будем устанавливать (**Get Cookie**), считывать (**Set Cookie**) и удалять (**Delete Cookie**) `cookie`.

Для того чтобы установить `cookie`, необходимо вызвать функцию `$.cookie(name, value, [options])`, предоставляемую плагином, и передать ей в качестве аргументов имя и значение `cookie`. Третьим (необязательным) параметром можно передать объект `options`, содержащий дополнительные настройки `cookie`, такие как `expires`, `path`, `domain`, `secure`.

Для того чтобы прочитать значение `cookie`, достаточно вызвать функцию `$.cookie(name)`, передав только один аргумент — имя `cookie`.

Удалить `cookie` тоже несложно — вызвать функцию с передачей в качестве второго параметра значения `null`.

Все очень просто, только необходимо отметить, что если вы устанавливали `cookie` с передачей необязательного аргумента `options`, то для корректного удаления этого `cookie` его также необходимо будет указать.

Дополнительную информацию можно найти по адресу **https://github.com/carhartl/jquery-cookie**.

16.2. jQuery Timers

ЗАДАЧА

При разработке веб-страниц иногда приходится сталкиваться с необходимостью выполнить какое-либо действие по истечении некоторого времени.

Решение

В решении этой задачи очень поможет плагин jQuery Timers.

В листинге 16.2.1 приведен пример простейшего таймера, который запускается при загрузке веб-страницы и работает до тех пор, пока она не будет закрыта.

Листинг 16.2.1. Использование плагина jQuery Timers

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.timers.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("span").everyTime(1000,function(i) {
    $(this).text(i);
  });
});
</script>
</head>
<body>
<div>С момента загрузки страницы прошло <span>0</span> секунд</div>
</body>
</html>
```

Для начала потребуется подключить файл библиотеки jQuery и файл плагина jquery.timers.js. HTML-разметка примера очень проста — элемент `div` с обыкновенным текстом и элемент `span`, куда будет вставляться число секунд, прошедшее с момента загрузки страницы.

Посмотрим на JavaScript-код. Мы отыскали элемент `span` и применили к нему метод `everyTime()`, предоставляемый плагином. Первым аргументом передается интервал в миллисекундах, через который будет срабатывать таймер. Второй аргумент — callback-функция с единственным аргументом `i` — числом срабатываний таймера. Учитывая, что таймер будет срабатывать один раз в секунду, можно счи-

тать, что этот аргумент и есть число секунд, прошедших с момента загрузки страницы. Далее мы просто вставляем его в качестве текста в элемент `span`.

Немного усложним задачу и посмотрим, как можно с помощью плагина jQuery Timers реализовать управляемый таймер (листинг 16.2.2).

Листинг 16.2.2. Использование плагина jQuery Timers

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-2-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.timers.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#start").click(function() {
    $("span").everyTime(1000, "myTimer", function(i) {
      $(this).text(i);
    }, 60);
  });
  $("#stop").click(function() {
    $("span").stopTime("myTimer");
  });
});
</script>
</head>
<body>
<div>С момента запуска таймера прошло <span>0</span> секунд</div>
<button id="start">Запустить</button>
<button id="stop">Остановить</button>
</body>
</html>
```

Теперь у нас добавились две кнопки — **Запустить** и **Остановить**. С ними мы связали событие `click()`, при наступлении которого будут вызываться соответствующие callback-функции.

При щелчке на кнопке **Запустить** мы отыщем элемент `span` и применим к нему уже знакомый метод `everyTime()`. Но, обратите внимание, теперь мы передаем этому методу дополнительные аргументы, с которыми еще не знакомы. Во втором аргументе передается так называемая метка таймера. Поскольку с одним и тем же элементом может быть связано сразу несколько таймеров, метка служит для идентификации конкретного таймера. Число, переданное в четвертом аргументе, — интервал времени в секундах, через который таймер прекратит свою работу.

В элемент span будем вставлять число секунд, прошедших с момента запуска таймера.

При щелчке на кнопке **Остановить** мы отыщем элемент span и, применив метод stopTimer(), остановим таймер. Обратите внимание, что в качестве аргумента мы передаем методу метку таймера, который необходимо остановить.

И, наконец, мы познакомимся с третьим и последним из методов, предоставляемых плагином. Это метод oneTime() — с его помощью можно легко реализовать "одноразовый" таймер. Посмотрите пример, приведенный в листинге 16.2.3.

Листинг 16.2.3. Использование плагина jQuery Timers

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-2-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.timers.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("div").oneTime("5s", function() {
    $(this).hide(1500);
  });
});
</script>
</head>
<body>
<div>Через 5 секунд этот текст будет скрыт</div>
</body>
</html>
```

В HTML-разметке имеется только элемент div, текст в котором сообщает, что через 5 с он будет скрыт. Посмотрим на JavaScript-код. Первое, что мы делаем при готовности страницы, — отыскиваем элемент div и применяем к нему метод oneTime(). Первый аргумент — время, через которое должен сработать таймер, второй — callback-функция, вызываемая, как только таймер сработает. Обратите внимание, в первом аргументе можно передавать как число — тогда это число должно быть представлено в миллисекундах, так и строку — тогда возможна понятная для человека форма записи.

Дополнительную информацию можно найти на сайте библиотеки jQuery по адресу **http://plugins.jquery.com/project/timers** или на сайте разработчика плагина **http://jquery.offput.ca/timers/**.

16.3. jQuery Cluetip

ЗАДАЧА

Не секрет, что стандартные всплывающие подсказки, в которых отображается содержимое атрибута `title`, имеют очень непрезентабельный вид. Зачастую необходимо реализовать всплывающие подсказки с оформлением, не только гармонирующим со стилем веб-страницы, но и, например, загружающим текст из внешних файлов.

Решение

В решении этой задачи очень поможет плагин jQuery Cluetip.

В листинге 16.3.1 приведен пример самого простого варианта использования плагина — всплывающая подсказка формируется из содержимого атрибута `title` элемента `a`.

Листинг 16.3.1. Использование плагина jQuery Cluetip

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.cluetip.js" type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="css/jquery.cluetip.css" />
<script type="text/javascript">
$(function(){
  $("a:first").cluetip({
    splitTitle: "|"
  });
});
</script>
</head>
<body>
<a href="#" title="Плагин jQuery Cluetip|Поможет организовать
на веб-странице оригинальные всплывающие подсказки.">jQuery Cluetip</a>
</body>
</html>
```

Сначала потребуется подключить библиотеку jQuery, файл плагина js/jquery.cluetip.js и файл стилевого оформления всплывающей подсказки css/jquery.cluetip.css. С его помощью можно оформить всплывающую подсказку в соответствии со своим вкусом, а по умолчанию он содержит три варианта стилевого оформления.

В HTML-разметке присутствует только одна ссылка, но она имеет атрибут `title`, из которого будет сформирована всплывающая подсказка.

Посмотрим JavaScript-код. Сначала посредством селекторов jQuery мы выбираем необходимый элемент, а затем применяем к нему функциональность плагина.

Обратите внимание на единственную опцию `splitTitle`, которая передается плагину. Значение этой опции — вертикальная черта, служащая разделителем. Все, что находится в атрибуте `title` до разделителя, будет заголовком всплывающей подсказки, все, что после, — текстом подсказки.

Пример всплывающей подсказки в одном из готовых вариантов оформления приведен на рис. 16.1.

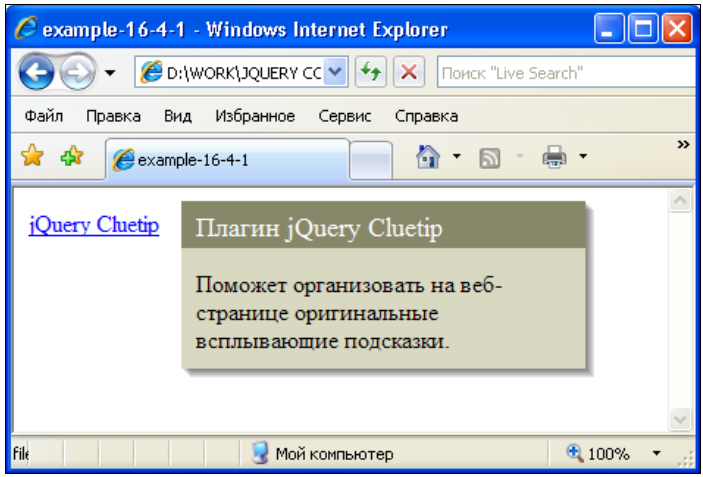


Рис. 16.1. Результат использования плагина jQuery Cluetip

Но это еще не все возможности плагина jQuery Cluetip — он в состоянии загружать подсказки, используя AJAX-запросы (листинг 16.3.2).

Листинг 16.3.2. Использование плагина jQuery Cluetip

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-3-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.cluetip.js" type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="jquery.cluetip.css" />
<script type="text/javascript">
$(function(){
  $("a.basic").cluetip();
});
```

```
</script>
</head>
<body>
Содержимое подсказки <a class="basic" href="#" rel="ajax.htm">загружается с
помощью ajax-запроса</a>, а не из атрибута title.
</body>
</html>
```

Основное внимание следует обратить даже не на JavaScript-код, а на HTML-разметку. В атрибуте `rel` элемента `a` указано, откуда следует забирать информационное наполнение для тела всплывающей подсказки. Это значение можно переопределить в опциях плагина, указав какой-либо другой атрибут.

Еще одна полезная возможность плагина jQuery Cluetip — он может загружать в подсказку содержимое, находящееся на той же странице (листинг 16.3.3).

Листинг 16.3.3. Использование плагина jQuery Cluetip

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-16-3-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.5.2.min.js" type="text/javascript"></script>
<script src="js/jquery.cluetip.js" type="text/javascript"></script>
<link rel="stylesheet" type="text/css" href="css/jquery.cluetip.css" />
<style type="text/css">
body { position:relative; }
</style>
<script type="text/javascript">
$(function(){
  $("a.local").cluetip({
    local: true,
    arrows: true,
    sticky: true,
    closePosition: "bottom",
    closeText: "Закрыть",
    cluetipClass: "jtip"
  });
});
</script>
</head>
<body>
<p>Плагин jQuery Cluetip умеет <a class="local" href="#" rel="p.loadme"
title="Локальный контент">загружать</a> локальный контент.</p>
```

```
<p class="loadme">В данном случае в подсказку загружено содержимое первого
элемента p, у которого имеется класс <strong>loadme</strong> и который
находится на этой же странице.</p>
</body>
</html>
```

В примере из листинга 16.3.3 мы передали плагину объект с настройками, в том числе указали `local:true`, а в атрибуте `rel` элемента `a` — имя класса элемента, содержимое которого нужно загрузить в подсказку. При этом самого элемента на странице не видно — это определяется настройками плагина по умолчанию, но такое поведение можно легко переопределить. В этом же примере мы указали еще несколько опций, а названия всех опций и их описания приведены в табл. 16.1.

Таблица 16.1. *Опции плагина jQuery Cluetip*

Опция	Описание
`width`	Определяет ширину всплывающей подсказки. Значение по умолчанию — 275
`height`	Определяет высоту всплывающей подсказки. Значение по умолчанию — `auto`
`cluezIndex`	Устанавливает значение CSS-свойства `z-index` для всплывающей подсказки. Значение по умолчанию — 97
`positionBy`	Устанавливает тип позиционирования всплывающей подсказки. Возможные значения: `auto` (по умолчанию), `mouse`, `bottomTop`, `fixed`. При использовании значения `fixed` необходимо задать значения опций `topOffset` и `leftOffset`
`topOffset`	Отступ всплывающей подсказки от верхнего края вызывающего элемента в пикселах. По умолчанию 15 px
`leftOffset`	Отступ всплывающей подсказки от левого края вызывающего элемента в пикселах. По умолчанию 15 px
`local`	Значение по умолчанию — `false`. Если установить значение опции `true`, то всплывающая подсказка будет наполнена локальным содержимым
`hideLocal`	Применяется совместно с опцией `local`. По умолчанию имеет значение `true`, что приводит к скрытию загружаемого в подсказку локального содержимого, расположенного на этой же странице. Если установить значение `false`, содержимое будет не только загружаться в подсказку, но и отображаться на странице
`attribute`	Атрибут, значение которого будет использовано для получения информационного наполнения всплывающей подсказки. Значение по умолчанию — `rel`
`titleAttribute`	Атрибут, значение которого будет использовано для получения заголовка всплывающей подсказки. Значение по умолчанию — `title`
`splitTitle`	Символ, который будет использован для разбиения значения атрибута `title` при создании из него заголовка и информационного наполнения подсказки

Таблица 16.1 *(продолжение)*

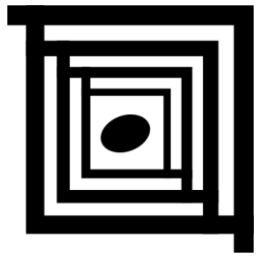
Опция	Описание
showTitle	Значение по умолчанию — true, что определяет обязательный показ области заголовка всплывающей подсказки
cluetipClass	Имя класса, автоматически добавляемое к префиксу 'cluetip-'. Значение по умолчанию — default. Кроме того, в стилевом файле плагина есть и еще предопределенные классы — jtip и rounded
hoverClass	Имя стилевого класса, добавляемого к вызывающему элементу по событию mouseover и удаляемого по событию mouseout
waitImage	Показывать ли картинку ожидания загрузки, которая установлена в стилевом файле jquery.cluetip.css. Значение по умолчанию — true (показывать)
arrows	Значение по умолчанию — false. Если установить true, то на соответствующей стороне подсказки будет отображена стрелка
dropShadow	Значение по умолчанию — true. Если вы не хотите, чтобы всплывающая подсказка не отбрасывала тень, то следует установить значение false
dropShadowSteps	Установка размера отбрасываемой тени. По умолчанию — 6
sticky	Значение по умолчанию — false. Если установить true, подсказка будет сохранять видимость до тех пор, пока ее не закроют принудительно
mouseOutClose	По умолчанию — false. Если установить true, подсказка будет закрыта в том случае, если указатель мыши покинет ее пределы
activation	Значение по умолчанию — hover, приводит к отображению подсказки при наведении указателя мыши и скрытию подсказки при выходе указателя мыши за пределы вызывающего элемента. Установите значение click для вызова подсказки по щелчку
clickThrough	По умолчанию — true. Если установлено значение по умолчанию и в опции activation содержится значение, отличное от click, то при нажатии на ссылку происходит переход по ней
tracking	По умолчанию — false. Если установить true, то всплывающая подсказка будет перемещаться вместе с указателем мыши (экспериментальная опция)
delayedClose	По умолчанию — 0. В опции может быть передано число миллисекунд, определяющее задержку закрытия подсказки (экспериментальная опция)
closePosition	Позиция элемента, щелчок по которому приводит к закрытию подсказки (при sticky: true). Значение по умолчанию — top. Другие возможные значения — bottom и title
closeText	Значение по умолчанию — close. Текст или HTML-код, щелчок по которому приводит к закрытию подсказки (при sticky: true)
truncate	Число символов, до которого будет усечено информационное наполнение. Значение по умолчанию — 0 (усечения не происходит)
fx	В этой опции можно передать объект для настройки эффекта и скорости при открытии всплывающей подсказки. Опции, передаваемые в объекте: open и openSpeed. Эффект, применяемый при открытии, определяет опция open — может принимать значения show (по умолчанию), slideDown и fadeIn. Опция openSpeed определяет скорость открытия — может быть или числом в миллисекундах, или строкой, в которой возможны значения slow, normal или fast

Таблица 16.1 *(окончание)*

Опция	Описание
`onActivate`	Принимает пользовательскую функцию, которая будет вызвана перед появлением всплывающей подсказки. Обратите внимание, что для корректной работы функция обязательно должна возвращать `true`
`onShow`	Принимает пользовательскую функцию, которая будет вызвана при появлении всплывающей подсказки
`ajaxCache`	По умолчанию — `true`. Используется для кэширования результатов AJAX-запросов, чтобы избежать ненужных обращений к серверу
`ajaxProcess`	В этой опции можно определить пользовательскую функцию, которая будет обрабатывать данные, полученные от сервера перед их отображением
`ajaxSettings`	В качестве значения этой опции можно передать объект, содержащий стандартные настройки AJAX-запроса, за исключением таких опций, как: `error`, `complete`, `success`, и `url`

Дополнительную информацию можно найти на веб-сайте разработчиков плагина jQuery Cluetip **http://plugins.learningjquery.com/cluetip/** или на официальном веб-сайте jQuery **http://plugins.jquery.com/project/cluetip**.

ГЛАВА 17



UI jQuery — виджеты

Виджеты — это готовые элементы пользовательских интерфейсов, имеющие множество различных опций, с помощью которых разработчик может очень гибко настроить виджет для решения своих задач. Виджеты можно расценивать как обычные плагины для библиотеки jQuery, просто плагины наиболее удачные, которые приобрели официальный статус. В этой главе будут приведены примеры проектирования пользовательских интерфейсов на основе виджетов. Общее правило при использовании любого виджета — подключение к странице помимо самого файла библиотеки jQuery, файла соответствующего виджета. На странице настраиваемой закачки **http://jqueryui.com/download** можно совместить все нужные виджеты в одном файле. Кроме того, здесь же есть и готовые темы оформления виджетов.

17.1. Виджет Accordion

Щелчок по заголовку скрывает/отображает содержимое, разбитое на логические секции. При отображении содержимого одной секции открытая ранее секция обязательно закрывается.

> ***Примечание***
>
> Если необходимо открывать одновременно несколько секций, то можно обойтись без виджета Accordion.

Внешний вид виджета Accordion в одном из многочисленных вариантов оформления представлен на рис. 17.1.

ЗАДАЧА

Необходимо применить на веб-странице виджет Accordion.

Решение

Решение этой задачи приведено в листинге 17.1.1. Для стилевого оформления виджета выбрана тема "smoothness".

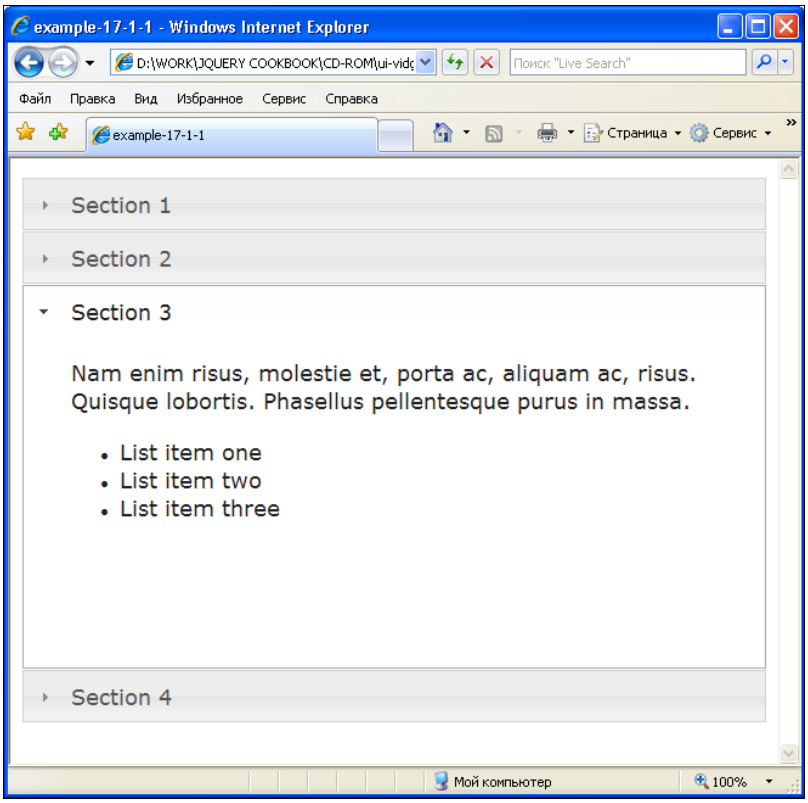


Рис. 17.1. Виджет Accordion в одном из многочисленных вариантов оформления

Листинг 17.1.1. Использование виджета Accordion

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/smoothness/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#accordion").accordion();
});
</script>
</head>
<body>
<div id="accordion">
```

```
  <h3><a href="#">Section 1</a></h3>
  <div>
    <p>Mauris mauris ante, blandit et, ultrices a, suscipit eget, quam. Integer
ut neque.</p>
  </div>
  <h3><a href="#">Section 2</a></h3>
  <div>
    <p>Sed non urna. Donec et ante. Phasellus eu ligula. Vestibulum sit amet
purus.</p>
  </div>
  <h3><a href="#">Section 3</a></h3>
  <div>
    <p>Nam enim risus, molestie et, porta ac, aliquam ac, risus. Quisque
lobortis. Phasellus pellentesque purus in massa.</p>
    <ul>
      <li>List item one</li>
      <li>List item two</li>
      <li>List item three</li>
    </ul>
  </div>
  <h3><a href="#">Section 4</a></h3>
  <div>
    <p>Cras dictum. Pellentesque habitant morbi tristique senectus et netus et
malesuada fames ac turpis egestas.</p>
    <p>Suspendisse eu nisl. Nullam ut libero. Integer dignissim consequat
lectus.</p>
  </div>
</div>
// Набор кнопок будет использоваться только для демонстрации
// работы методов виджета Accordion
<button id="disable">Disable</button>
<button id="enable">Enable</button>
<button id="destroy">Destroy</button>
<button id="getter">Get Option</button>
<button id="setter">Set Option</button>
<button id="activate">Activate</button>
</body>
</html>
```

Обсуждение

Следует заметить, что виджет Accordion требует определенной HTML-разметки, пример которой приведен в листинге 17.1.1. Обратите внимание, что по умолчанию виджет считает заголовками секций элементы `a` (в нашем примере они заключены в тег `<h3>`), а содержимое секций должно быть заключено в следующий непосредственно за ним элемент `div`. Внутри элемента `div` содержимое может быть практически любым.

Итак, сначала мы подключили файл стилей css/smoothness/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл

библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и виджета Accordion. За связь виджета с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.1.2.

Листинг 17.1.2. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion();
});
</script>
```

Код из листинга 17.1.2 реализует базовую функциональность виджета. Говоря иначе, все возможные настройки, которые можно было бы передать этому виджету, при такой реализации имеют значения по умолчанию.

Теперь попробуем передать некоторые настройки, чтобы понять, как это работает. Чтобы не повторяться, в листинге 17.1.3 приведен только интересующий нас фрагмент кода, поскольку и подключаемые файлы, и HTML-разметка останутся без изменений.

Листинг 17.1.3. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    autoHeight: false,
    active: 2
  });
});
</script>
```

Если вы воспроизведете пример из листинга 17.1.3, то сможете обнаружить, что по умолчанию открывается третья секция и каждая секция имеет высоту, соответствующую ее содержимому. Тогда как в примере из листинга 17.1.1 по умолчанию открывалась первая секция, высота всех секций была одинаковой и равной высоте секции с максимальным количеством содержимого. Мы добились этого, передав виджету при инициализации всего две опции с соответствующими значениями. В табл. 17.1 приведены описания всех доступных опций.

Таблица 17.1. *Опции виджета Accordion*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Accordion будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия

Таблица 17.1 *(окончание)*

Опция	Описание
`active`	Выбирает секцию, открытую при инициализации. По умолчанию имеет значение `first child`, т. е. открывается первая секция. Установка значения в `false` не приведет к тому, что все секции будут закрыты. Для того чтобы иметь возможность закрывать все секции, используется `collapsible: true`
`animated`	Включает или отключает анимацию. Установка опции в `false` отключит анимацию, применяемую по умолчанию (эффект скольжения). Поддерживаются еще эффекты `'bounceslide'` и `'easeslide'`, но оба эффекта требуют подключения дополнительного плагина jQuery Easing
`autoHeight`	Автоматическая установка высоты секции. Значение по умолчанию — `true` и высота всех секций определяется секцией с максимальной высотой. При установке значения `false` все секции будут иметь высоту, определяемую внутренним содержимым секции
`clearStyle`	Значение по умолчанию — `false`. Если установить значение `true`, то после окончания анимации будут очищены CSS-свойства `height` и `overflow`. Эта опция позволяет работать с динамически изменяемым содержимым секций. Не используется вместе с опцией `autoHeight`
`collapsible`	Значение по умолчанию — `false`. Если при инициализации установить значение `true`, можно будет закрывать открытую секцию щелчком по заголовку, так, чтобы все секции были закрыты
`event`	Определяет событие, по которому происходит переключение секций. Значение по умолчанию — `'click'`. Можно задать значение `'mouseover'`
`fillSpace`	Значение по умолчанию — `false`. При установке значения `true` открытая секция будет занимать всю возможную высоту, определяемую родительским элементом. Переопределяет опцию `autoHeight`
`header`	По умолчанию заголовком секции считается элемент `a`. Но заголовок можно определить явно, например `header: 'h3'`
`icons`	Значки для использования в заголовках. Это объект, состоящий из двух пар ключ/значение. Может быть определен как `'header': 'ui-icon-plus'` и `'headerSelected': 'ui-icon-minus'`. Значения являются именами классов CSS, с помощью которых выбирается соответствующая картинка. Рекомендуется применять значки, предоставляемые jQuery UI ThemeRoller вместе с многочисленными вариантами стилевого оформления
`navigation`	Значение по умолчанию — `false`. Если установить `true`, разыскивает и активирует ссылку. Служит, как правило, для построения меню на основе виджета, поскольку при перезагрузке страницы позволяет сохранить состояние. Обычно применяют HTML-разметку на основе списков (см. листинг 17.1.4)
`navigationFilter`	В этой опции можно определить функцию, которая будет каким-либо образом обрабатывать ссылку, по которой осуществляется переход. Используется совместно с опцией `navigation`

В листинге 17.1.4 приведен пример построения меню на основе виджета Accordion. На этот раз мы для разнообразия не используем вообще никаких тем оформления.

Листинг 17.1.4. Использование виджета Accordion

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-1-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    header: '.head',
    event: 'mouseover',
    fillSpace: true,
    navigation: true
  });
});
</script>
<style type="text/css">
#accordion {
  width:200px;
}
</style>
</head>
<body>
<ul id="accordion">
  <li>
    <a class="head" href="?p=1.1">Guitar</a>
    <ul>
      <li><a href="?p=1.1.1">Electric</a></li>
      <li><a href="?p=1.1.2">Acoustic</a></li>
      <li><a href="?p=1.1.3">Amps</a></li>
      <li><a href="?p=1.1.4">Effects A</a></li>
      <li><a href="?p=1.1.5">Effects B</a></li>
      <li><a href="?p=1.1.6">Effects C</a></li>
      <li><a href="?p=1.1.7">Effects D</a></li>
      <li><a href="?p=1.1.8">Accessories</a></li>
    </ul>
  </li>
  <li>
    <a class="head" href="?p=1.2">Bass</a>
    <ul>
      <li><a href="?p=1.2.1">Electric</a></li>
      <li><a href="?p=1.2.2">Acoustic</a></li>
      <li><a href="?p=1.2.3">Amps</a></li>
```

```
      <li><a href="?p=1.2.4">Effects</a></li>
      <li><a href="?p=1.2.5">Accessories</a></li>
    </ul>
  </li>
  <li>
    <a class="head" href="?p=1.3">Drums</a>
    <ul>
      <li><a href="?p=1.3.1">Acoustic Drums</a></li>
      <li><a href="?p=1.3.2">Electronic Drums</a></li>
      <li><a href="?p=1.3.3">Cymbals</a></li>
      <li><a href="?p=1.3.4">Hardware</a></li>
      <li><a href="?p=1.3.5">Accessories</a></li>
    </ul>
  </li>
</ul>
</body>
</html>
```

Обратите внимание на вариант HTML-разметки, используемый для построения меню. Это ненумерованный список, в каждом элементе которого заголовком служит элемент `a`, а информационным наполнением секции — тоже ненумерованный список. Передавая в опциях `navigation: true`, мы переходим по ссылке, на которой был совершен щелчок, но при перезагрузке страницы мы обнаружим, что открыта как раз та секция, по ссылке в которой был совершен переход.

Однако это еще не все. Кроме возможности довольно гибко настроить виджет, можно заставить его реагировать на события. Очень простой пример приведен в листинге 17.1.5.

Листинг 17.1.5. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    autoHeight: false,
    active: 2,
    change: function(event, ui) {
      alert('Произошло событие ' + event.type);
    }
  });
});
</script>
```

В листинге 17.1.5 мы определили опцию `change`, значением которой является callback-функция, вызываемая при смене секции. Эта callback-функция может принимать два аргумента. Первый — объект `event`, второй — объект `ui`, в свойствах которого содержится информация:

- `ui.newHeader` — об активизируемом заголовке;
- `ui.oldHeader` — о предыдущем заголовке;
- `ui.newContent` — об активизируемом содержимом;
- `ui.oldContent` — о предыдущем содержимом.

На примере из листинга 17.1.6 показано, как можно извлечь необходимую информацию из доступного объекта.

Листинг 17.1.6. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    autoHeight: false,
    active: 2,
    change: function(event, ui) {
      alert('Старый заголовок: ' + ui.oldHeader.find('a').text() +
            '\nНовый контент: ' + ui.newContent.text());
    }

  });
});
</script>
```

В примере из листинга 17.1.6 в опции `change` определяем callback-функцию, которая, обращаясь к свойствам объекта `ui`, выводит в окне предупреждения полученную информацию. Таким образом, при каждом переключении секции мы будем видеть в окне предупреждения заголовок той секции, которую мы покидаем, и содержимое той секции, куда мы переходим.

В табл. 17.2 приведены описания событий виджета и опции, в которых могут быть определены callback-функции, связанные с этими событиями.

Таблица 17.2. *События виджета Accordion*

Опция	Событие	Описание
`create`	`accordioncreate`	Наступает в момент инициализации
`changestart`	`accordionchangestart`	Происходит каждый раз при начале смены секций
`change`	`accordionchange`	Наступает каждый раз после того, как изменилась активная секция. Если используется анимация, то событие наступает по ее завершении, если нет — то событие наступает немедленно

Но и это еще не все. Виджет Accordion имеет еще и несколько методов, расширяющих и без того богатую функциональность (табл. 17.3).

Таблица 17.3. *Методы виджета Accordion*

Метод	Описание
destroy `.accordion('destroy')`	Полностью удаляет всю функциональность виджета Accordion. Возвращает элементы в состояние, предшествующее инициализации
disable `.accordion('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.accordion('enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`
option `.accordion('option', optionName, [value])`	Устанавливает значение любой опции виджета после инициализации
option **`.accordion('option', optionName)`**	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Accordion
activate `.accordion('activate', index)`	С помощью этого метода можно программно активизировать необходимые секции. Аргумент `index` может быть номером секции (отсчет ведется от нуля) или селектором jQuery, который выберет необходимый элемент. Передавая значение –1, можно закрыть все секции (только при использовании опции `collapsible:true`)
resize	Пересчитывает высоту содержимого секций аккордиона при ее изменении (при использовании опции `fillSpace: true`)

Продемонстрируем работу методов `destroy`, `disable` и `enable` в листинге 17.1.7. Здесь, наконец, нам пригодятся кнопки, на которые мы до этой поры внимания не обращали.

Листинг 17.1.7. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    autoHeight: false,
    active: 2
  });
  $("#disable").click(function(){
    $("#accordion").accordion('disable');
  });
```

```
  $("#enable").click(function(){
    $("#accordion").accordion('enable');
  });
  $("#destroy").click(function(){
    $("#accordion").accordion('destroy');
  });
});
</script>
```

Разберем код из листинга 17.1.7. Мы добавили только обработчики событий для трех кнопок. По щелчку на кнопке будем вызывать метод с соответствующим именем. После инициализации виджет Accordion функционирует так, как ему и положено. Если щелкнуть указателем мыши по кнопке **Disable**, окажется, что возможности виджета стали недоступны. Восстановить их можно щелчком по кнопке **Enable**, вызывая тем самым нужный метод. Ну а щелчок по кнопке **Destroy** удалит функциональность виджета и вернет все элементы в состояние, предшествующее инициализации. Восстановить работоспособность виджета можно будет только перезагрузкой страницы.

В листинге 17.1.8 приведен пример того, как можно получить значение какой-либо опции и изменить это значение уже после инициализации виджета.

Листинг 17.1.8. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    autoHeight: false,
    active: 2,
    animated: false
  });
  $("#getter").click(function(){
    alert('animated: '+$("#accordion").accordion('option', 'animated'));
  });
  $("#setter").click(function(){
    $("#accordion").accordion('option', 'animated', 'slide');
    alert('animated: '+$("#accordion").accordion('option', 'animated'));
  });
});
</script>
```

В примере из листинга 17.1.8 при инициализации виджета мы указали опцию `animated: false`, отключив, таким образом, применяемую по умолчанию анимацию. Попробуйте переключать секции и убедитесь в том, что никаких эффектов при переключении действительно нет. Если щелкнуть по кнопке **Get Option**, мы увидим окно предупреждения, в котором будет выведено текущее значение опции `animated`.

Изменим значение этой опции на `'slide'`, щелкнув по кнопке **Set Option**. Получили окно предупреждения с сообщением: `animated: slide`. Проверим это, переключая секции. Действительно, теперь при переключении секций есть анимация.

В листинге 17.1.9 приведен пример использования метода `activate`.

Листинг 17.1.9. Использование виджета Accordion

```
<script type="text/javascript">
$(function(){
  $("#accordion").accordion({
    active: 2,
    collapsible:true
  });
  $("#activate").click(function(){
    $("#accordion").accordion('activate', 'h3:last');
  });
});
</script>
```

При инициализации виджета активной секцией будет третья по счету, а при щелчке на кнопке **Activate** откроется (закроется, если открыта) самая последняя секция. Мы добились этого, передав методу непосредственно селектор jQuery. Попробуйте изменить селектор, например, на `'h3:first'` или передать значение `1`. Интересно попробовать передать методу значение `–1` — тогда все секции окажутся закрытыми. Важно отметить, что эта возможность будет доступна только при инициализации виджета с опцией `'collapsible: true'`.

17.2. Виджет Datepicker

Datepicker — это интерактивный календарь, который связан с полем ввода. Щелчок в поле ввода приводит к тому, что в отдельном слое открывается небольшой календарь. Выбираете нужную дату, щелкаете мышью в любом месте страницы (или нажимаете клавишу <Esc>) и готово — дата занесена в поле ввода. Внешний вид виджета Datepicker в одном из вариантов оформления представлен на рис. 17.2.

ЗАДАЧА

Необходимо применить на веб-странице виджет Datepicker.

Решение

Решение этой задачи приведено в листинге 17.2.1. Для стилевого оформления выбрана тема "lightness".

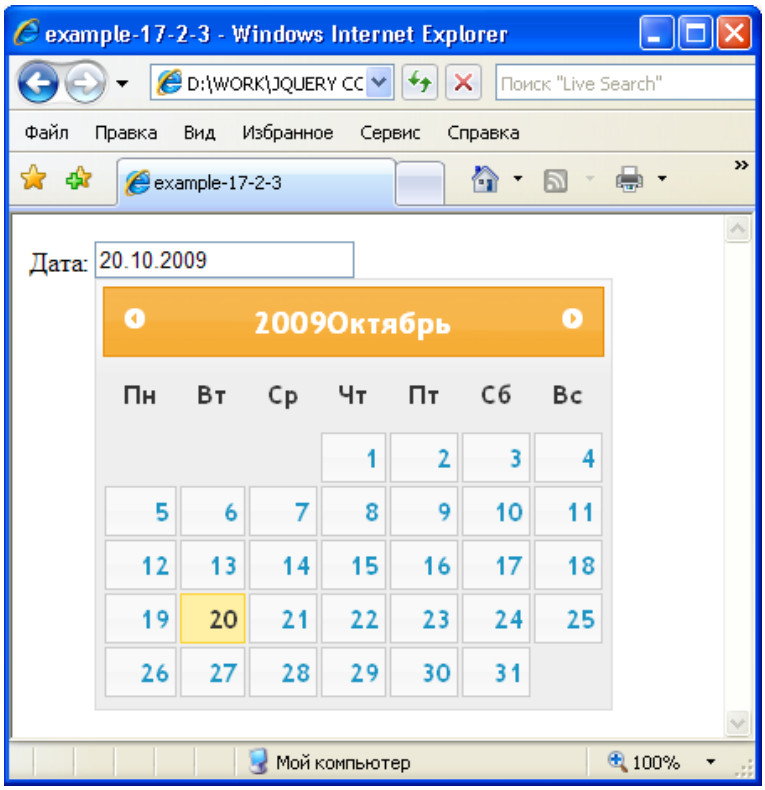


Рис. 17.2. Виджет Datepicker в одном из вариантов оформления

Листинг 17.2.1. Использование виджета Datepicker

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/ui-lightness/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script src="js/i18n/jquery-ui-i18n.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#datepicker").datepicker();
});
</script>
</head>
<body>
```

```
<p>Дата: <input id="datepicker" type="text" /></p>
<a href="#">Открыть в диалоге</a>
</body>
</html>
```

Обсуждение

Сначала мы подключили файл стилей css/ui-lightness/jquery-ui-1.8.9. custom.css одной из многочисленных тем оформления. Вы можете выбрать для себя любую тему, которая придется вам по душе. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором заключена функциональность виджета Datepicker. Файл js/i18n/jquery-ui-i18n.js обеспечивает локализацию виджета для нужного языка.

Datepicker не предъявляет особенных требований к HTML-разметке. Обычное поле ввода — элемент `input`.

За связь виджета с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.2.2.

Листинг 17.2.2. Использование виджета Datepicker

```
<script type="text/javascript">
$(function(){
  $("#datepicker").datepicker();
});
</script>
```

В листинге 17.2.2 приведен пример кода, который реализует виджет с настройками по умолчанию. Отлично, но мы хотим видеть в календаре названия месяцев и дней недели, написанных по-русски (или по-украински). Листинг 17.2.3 иллюстрирует пример локализации виджета для русского языка.

Листинг 17.2.3. Использование виджета Datepicker

```
<script type="text/javascript">
$(function(){
  $.datepicker.setDefaults($.extend($.datepicker.regional['ru']));
  $("#datepicker").datepicker();
});
</script>
```

Приведенный в листинге 17.2.3 код обеспечит поддержку русского языка, а если вы замените `ru` на `uk`, то получите календарь уже на украинском языке.

Но, чтобы решить свои задачи, вам, вероятно, понадобится изменить какие-то настройки Datepicker. Давайте попробуем это сделать (листинг 17.2.4).

Листинг 17.2.4. Использование виджета Datepicker

```
<script type="text/javascript">
$(function(){
  $.datepicker.setDefaults($.extend($.datepicker.regional['ru']));
  $("#datepicker").datepicker({
    minDate: '-30',
    maxDate: '+1m +1w +3d'
  });
});
</script>
```

В примере из листинга 17.2.4 мы передали виджету две опции, названия которых говорят сами за себя — мы установили минимальную и максимальную дату, которую можно выбрать в календаре. Безусловно, данные опции полезны, но таких опций гораздо больше, а принцип работы с ними аналогичен рассмотренному в листинге 17.2.4. Полный список возможных опций приведен в табл. 17.4.

Таблица 17.4. *Опции виджета Datepicker*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Datepicker будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия
`altField`	jQuery селектор для другого поля, которое должно быть обновлено, как только будет выбрана дата в Datepicker. Формат даты в этом дополнительном поле устанавливается с помощью опции `altFormat`
`altFormat`	Формат даты, который будет использоваться для опции `altField`. Эти настройки позволяют показывать пользователю один формат даты, тогда как для вычислений может быть установлен другой формат. Полный список возможных форматов можно найти на **http://docs.jquery.com/UI/Datepicker/formatDate**
`appendText`	Текст, который будет отображаться после каждого поля выбора даты. Можно, например, пометить поле как обязательное для заполнения
`autoSize`	Значение по умолчанию — `false`. Если установить значение `true`, поле ввода будет автоматически изменять свой размер, чтобы полностью вместить выбранную дату в текущем формате
`buttonImage`	Адрес картинки, отображающейся на кнопке, с помощью которой можно вызывать календарь. Используется совместно с опцией `showOn`, когда та принимает значения `'button'` или `'both'`. Если был определен текст в опции `buttonText`, он становится значением атрибута `alt` картинки
`buttonImageOnly`	Если для этой опции установить значение `true`, то картинка, адрес которой определен в опции `buttonImage`, будет отображаться не на кнопке, а сама по себе

Таблица 17.4 *(продолжение)*

Опция	Описание
buttonText	Текст, отображающийся на кнопке, с помощью которой можно вызывать календарь. Используется совместно с опцией showOn, когда та принимает значения 'button' или 'both'
calculateWeek	В этой опции можно определить собственную функцию, которая будет вычислять порядковый номер недели в году. По умолчанию для вычисления используется формат ISO 8601
changeMonth	Если в этой опции передать значение true, то можно выбирать месяц из выпадающего списка
changeYear	Если в этой опции передать значение true, появится возможность выбирать год из выпадающего списка
closeText	Эта опция используется совместно с showButtonPanel, если последняя имеет значение true. Значение опции closeText определяется в файле локализации, если он используется, но это значение можно переопределить, явно указав его
constrainInput	По умолчанию опция имеет значение true и принуждает строго соблюдать формат даты, определенный в опциях виджета, в текстовом поле ввода. Чтобы иметь возможность не соблюдать этот формат, следует установить опцию в false
currentText	Эта опция используется совместно с showButtonPanel, если последняя имеет значение true. Значение опции currentText определяется в файле локализации, когда он присутствует, но это значение можно переопределить, явно указав его
dateFormat	Определяет формат даты. Значение опции dateFormat определяется в файле локализации, если он задействован, но это значение можно переопределить, явно указав его. Полный список возможных форматов можно найти на **http://docs.jquery.com/UI/Datepicker/formatDate**
dayNames	Массив, содержащий полные названия дней недели, начиная с воскресенья. Определяется в файле локализации, если он есть, но это значение можно переопределить, явно указав его
dayNamesMin	Массив, содержащий двухбуквенную аббревиатуру названия дней недели, начиная с воскресенья. Определяется в файле локализации, если он используется, но это значение можно переопределить, явно указав его
dayNamesShort	Массив, содержащий трехбуквенную аббревиатуру названия дней недели, начиная с воскресенья. Определяется в файле локализации, если он задействован, но это значение можно переопределить, явно указав его
defaultDate	Устанавливает дату, которая будет подсвечена при первом открытии, если поле выбора даты пустое. Опция может быть определена через объект Date, либо как число дней от текущего дня (например +7 или –15), либо как строка значений, определяющих период ('y' для лет, 'm' для месяцев, 'w' для недель, 'd' для дней, например '+1m +7d'), и, наконец, как null для текущего дня

Таблица 17.4 *(продолжение)*

Опция	Описание
duration	Длительность эффекта анимации при открытии (закрытии) календаря. Может принимать значения в виде строки — 'fast', 'normal' (по умолчанию), 'slow' или в виде числа в миллисекундах. Если передать пустую строку, календарь будет открываться и закрываться без эффекта анимации
firstDay	Опция устанавливает первый день недели: воскресенье — 0, понедельник — 1,... Определяется в файле локализации, если он используется, но это значение можно переопределить, явно указав его
gotoCurrent	Если установить значение опции в true, то кнопка **Сегодня** (доступная при showButtonPanel: true) будет указывать на выбранную дату взамен текущей
hideIfNoPrevNext	Если вы ограничиваете диапазон доступных дат с помощью опций minDate и maxDate, то при достижении края диапазона стрелки "Назад" и "Вперед" становятся неактивными. Но их можно и совсем скрыть, передав в опции hideIfNoPrevNext значение true
isRTL	Для этой опции необходимо установить значение true, если используется язык с написанием справа налево. Определяется в файле локализации, если он задействован
maxDate	Устанавливает максимальную возможную для выбора дату через объект Date, или как число дней от текущего (например +7), или как строку значений, определяющих период ('y' для лет, 'm' для месяцев, 'w' для недель, 'd' для дней, например '+1y +1m'), или null при отсутствии ограничения
minDate	Устанавливает минимальную возможную для выбора дату через объект Date, или как число дней от текущего (например -7), или как строку значений, определяющих период ('y' для лет, 'm' для месяцев, 'w' для недель, 'd' для дней, например '-1y -1m'), или null при отсутствии ограничения
monthNames	Массив, содержащий полные названия месяцев. Определяется в файле локализации, если он используется, но это значение можно переопределить, явно указав его
monthNamesShort	Массив, содержащий трехбуквенную аббревиатуру названия месяцев. Определяется в файле локализации, если он задействован, но это значение можно переопределить, явно указав его
navigationAsDateFormat	Значение по умолчанию — false. Если установить значение true, функция dateFormat будет примерена к значениям опций nextText, prevText и currentText, чтобы отображать при навигации, например, предыдущее и последующее названия месяцев
nextText	Текст, который отображается в качестве ссылки на следующий месяц. Определяется в файле локализации, если он используется, но это значение можно переопределить, явно указав его. Если подключен файл стилей ThemeRoller, это значение заменяется иконкой

Таблица 17.4 *(продолжение)*

Опция	Описание
numberOfMonths	Эта опция определяет, сколько месяцев сразу нужно показать. Значение опции может быть просто числом или массивом, состоящим из двух элементов, которые определяют соответственно количество строк и столбцов. Например, значение [2, 3] отобразит календарь в две строки по три месяца
prevText	Текст, который отображается в качестве ссылки на предыдущий месяц. Определяется в файле локализации, если он используется, но это значение можно переопределить, явно указав его. Если подключен файл стилей ThemeRoller, это значение заменяется иконкой
selectOtherMonths	Значение по умолчанию — false. Если указать true, то даты предыдущего/следующего месяца можно будет выбирать так же, как и даты текущего месяца. Используется только совместно с showOtherMonths: true
shortYearCutoff	Значение по умолчанию +10. Эта опция возможна только при установке в dateFormat двузначной записи года и играет роль компенсатора для определения века. Если значение передано в виде числа, используется как есть. Если значение передано в виде строки, конвертируется в число и добавляется к текущему значению года. После того как значение определено, все данные, выступающие в качестве значения года, и меньшие и равные ему считаются годами текущего века. Значения бóльшие — считаются годами предыдущего века
showAnim	Определяет тип анимации при открывании календаря. Значение по умолчанию — show (при закрывании будет использован hide). Без подключения дополнительных файлов возможны эффекты 'slideDown' и 'fadeIn' ('slideUp' и 'fadeOut' — при закрывании). Также возможны любые эффекты jQuery UI Effects при условии их дополнительного подключения
showButtonPanel	Установка значения true для этой опции приведет к тому, что будет показана панель с двумя кнопками — переход к сегодняшней дате и закрытие календаря
showCurrentAsPos	Когда отображаются сразу несколько месяцев, число, переданное в этой опции, определяет позицию текущего месяца. Значение по умолчанию — 0 (текущий месяц отображается в верхнем левом углу)
showMonthAfterYear	Значение по умолчанию — false (в заголовке название месяца идет перед годом). При указании значения true название месяца будет следовать за годом
showOn	Значение по умолчанию — 'focus' (календарь открывается при щелчке в поле ввода). Другие возможные значения — 'button' и 'both'. Рядом с полем ввода появится кнопка. В первом случае календарь будет открываться по щелчку на кнопке, а во втором случае как по щелчку на кнопке, так и при получении фокуса полем ввода
showOptions	Если вы применяете один из эффектов jQuery UI Effects, с помощью этой опции ему можно передать дополнительные настройки. Например: showOptions: {direction: 'up'}

Таблица 17.4 *(окончание)*

Опция	Описание
showOtherMonths	Значение по умолчанию — false. Если указать true, то на календаре будут отображаться дни предыдущего и/или следующего месяца без возможности их выбора. Чтобы иметь возможность выбора этих дат, используйте опцию selectOtherMonths
showWeek	Значение по умолчанию — false. Установка значения true приведет к добавлению дополнительной колонки, где будет отображаться порядковый новер недели в году. Для определения собственной функции вычисления порядкового номера недели предусмотрена опция calculateWeek
stepMonths	Определяет, насколько месяцев сдвигать календарь при щелчке по ссылкам **Следующий** и **Предыдущий**. По умолчанию — 1
weekHeader	В этой опции можно определить свой текст, который будет выводиться в заголовке колонки с порядковыми номерами недель. Опция используется вместе с showWeek: true
yearRange	Управление диапазоном лет, отображаемых в выпадающем списке (при использовании опции changeYear). Значение по умолчанию — '-10:+10' относительно текущего года. Возможно указание и абсолютного формата, например '1980:2025'
yearSuffix	В этой опции можно передать дополнительный текст, который будет отображаться в заголовке календаря, после цифр года

Попробуем теперь заставить виджет Datepicker реагировать на события (листинг 17.2.5).

Листинг 17.2.5. Использование виджета Datepicker

```
<script type="text/javascript">
$(function(){
  $.datepicker.setDefaults($.extend($.datepicker.regional['ru']));
  $("#datepicker").datepicker({
    beforeShow: function(input) {
      $(input).css("background-color","#ff9");
    },
    onSelect: function(dateText, inst) {
      $(this).css("background-color","");
      alert("Выбрано: " + dateText +
            "\n\nid: " + inst.id +
            "\nselectedDay: " + inst.selectedDay +
            "\nselectedMonth: " + inst.selectedMonth +
            "\nselectedYear: " + inst.selectedYear);
    },
```

```
    onClose: function(dateText, inst) {
      $(this).css("background-color","");
    }
  });
});
</script>
```

В примере из листинга 17.2.5 мы передаем виджету сразу три опции, в которых определены callback-функции. В опции `beforeShow` — функция, вызываемая перед отображением календаря. С помощью CSS-свойства `background-color` эта функция установит желтый цвет фона для элемента `input`. В опции `onSelect` — функция, которая будет вызвана в момент выбора какой-либо даты в календаре и передаст пустое значение CSS-свойству `background-color`, возвратив его в начальное состояние. И, наконец, в опции `onClose` — функция, вызываемая при закрытии календаря, если никакое значение выбрано не было. Она тоже вернет CSS-свойство `background-color` в его начальное состояние.

Но, обратите внимание, все callback-функции принимают некоторые аргументы. С ними нужно познакомиться немного подробнее.

В опции `beforeShow` функция принимает в качестве аргумента объект, характеризующий элемент `input`. В примере из листинга 17.2.5 мы указали этот объект в качестве селектора jQuery и получили, таким образом, возможность работать с его CSS-свойствами с помощью методов библиотеки.

В опции `onSelect` функция принимает два аргумента. Первый аргумент `dateText` — строка, представляющая собой тот текст, который появится и в элементе `input`. Второй аргумент — объект `datepicker`. В листинге 17.2.5 показано, как можно обратиться к некоторым из свойств этого объекта.

Функция, определенная в опции `onClose`, принимает те же два аргумента, что и функция из опции `onSelect`.

И еще одно важное обстоятельство для callback-функций, определенных в опциях `onSelect` и `onClose`, — в контексте этих функций ссылка `this` указывает на объект, характеризующий элемент `input`.

В табл. 17.5 приведены описания всех возможных событий для виджета Datepicker.

Таблица 17.5. *События виджета Datepicker*

Опция	Тип	Описание события
`create`	`datepickercreate`	Событие наступает в момент инициализации
`beforeShow`	`function(input, inst)`	Здесь можно определить функцию, которая будет вызываться перед открытием календаря. Функция принимает в качестве аргумента объект, характеризующий текстовое поле ввода, с которым работает виджет

Таблица 17.5 *(окончание)*

Опция	Тип	Описание события
beforeShowDay	function(date)	В этой опции можно определить пользовательскую функцию, которая будет принимать в качестве аргумента выбранную дату. Функция должна возвратить массив, где элемент с индексом [0] — true или false показывает, возможен или нет выбор этой даты. Элемент с индексом [1] содержит имя класса (классов) для отображения даты, элемент с индексом [2] (опционально) — текст всплывающей подсказки для даты. Функция будет вызываться для каждой даты в календаре в момент наведения указателя мыши
onChangeMonthYear	function(year, month, inst)	Здесь можно определить функцию, которая будет вызываться при смене месяца или года в календаре. Функция принимает три аргумента. Первые два аргумента — это новые значения года и месяца, третий аргумент — объект datepicker
onClose	function(dateText, inst)	В этой опции определяется функция, которая будет вызвана, когда календарь был закрыт без выбора какой-либо даты
onSelect	function(dateText, inst)	В этой опции определяется функция, которая будет вызвана, когда в календаре выбрана какая-либо дата

Познакомимся с методами, которые предлагает виджет Datepicker. В листинге 17.2.6 приведен пример использования одного из методов.

Листинг 17.2.6. Использование виджета Datepicker

```
<script type="text/javascript">
$(function(){
  $.datepicker.setDefaults($.extend($.datepicker.regional['ru']));
  $("#datepicker").datepicker({
    changeYear: true
  });
  $('a').click(function(){
    $("#datepicker").datepicker('dialog',
                                '23.04.2011',
                                function(){
                                  alert('Событие onSelect');
                                },
                                { showButtonPanel: true },
                                [300, 300]);
  });
});
</script>
```

Пример из листинга 17.2.6 демонстрирует работу метода `dialog`. Мы вызываем этот метод при щелчке на ссылке **Открыть в диалоге**. Первый аргумент — это название метода, второй аргумент `textDate` — дата, на которой по умолчанию будет открываться календарь. Остальные три аргумента необязательные, но мы рассмотрим и их. В аргументе `onSelect` можно передать callback-функцию, которая будет вызвана при выборе конкретной даты в календаре. В аргументе `settings` можно передать объект с новыми настройками виджета, которые будут применены к календарю, открытому в диалоговом окне. И, наконец, последний аргумент — `pos` — здесь можно задать координаты, в которых будет появляться календарь. Координаты можно задавать числами, но если передать в функцию, вызываемую по щелчку на ссылке, объект `event`, то можно будет определить координаты, через свойства объекта `event`. Например, `event.clientX` и `event.clientY`.

В табл. 17.6 приведены описания всех методов виджета Datepicker.

Таблица 17.6. *Методы виджета Datepicker*

Метод	Описание
destroy `.datepicker ('destroy')`	Полностью удаляет всю функциональность виджета Datepicker. Возвращает элементы в состояние, предшествующее инициализации
disable `.datepicker ('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.datepicker (enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`
option `.datepicker ('option', optionName, [value])`	Устанавливает значение любой опции виджета после инициализации
option `('option', optionName)`	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Datepicker
dialog `.datepicker ('dialog', dateText, [onSelect], [settings], [pos] )`	Открывает виджет Datepicker в режиме диалогового окна. В аргументе `dateText` передается дата, на которой должен быть открыт календарь. Остальные аргументы необязательные. В аргументе `onSelect` можно передать функцию, которая будет вызвана в момент выбора даты в календаре, в аргументе `settings` можно передать объект с новыми настройками виджета, в аргументе `pos` — координаты, в которых будет открыто диалоговое окно. Здесь можно использовать события мыши, чтобы определить координаты
isDisabled `.datepicker ('isDisabled')`	Метод возвращает значение `true`, если к виджету был применен метод `disable`, и `false` — в противном случае
hide `.datepicker ('hide', [speed])`	Скрывает ранее открытый календарь

Таблица 17.6 *(окончание)*

Метод	Описание
show `.datepicker ('show')`	Открывает календарь
getDate `.datepicker ('getDate')`	Метод возвращает дату, выбранную в календаре
setDate `.datepicker ('setDate', date)`	Метод позволяет установить дату в календаре. Значением аргумента `date` может быть строка (например: `25.10.1917`). Число, определяющее количество дней от текущей даты (например: `+7` или `–14`). Строка, определяющая период (`'y'` для лет, `'m'` для месяцев, `'w'` для недель, `'d'` для дней, например: `'+1m +7d'`). Значение `null` установит текущую дату

Пожалуй, можно дать еще пару полезных советов.

По умолчанию Datepicker открывается в новом слое, когда текстовое поле ввода получает фокус, и закрывается, когда это поле теряет фокус. Чтобы календарь был открыт постоянно, просто свяжите его функциональность с элементами `div` или `span`.

Управлять календарем можно посредством горячих клавиш:

- <Page Up>/<Down> — предыдущий/следующий месяц;
- <Ctrl>+<Page Up>/<Down> — предыдущий/следующий год;
- <Ctrl>+<Home> — выбор текущего месяца или открытие календаря после того, как он был закрыт;
- <Ctrl>+<Left>/<Right> — предыдущий/следующий день;
- <Ctrl>+<Up>/<Down> — предыдущая/следующая неделя;
- <Enter> — выбор отмеченной даты;
- <Ctrl>+<End> — закрытие и удаление даты из поля ввода;
- <Escape> — закрытие календаря без выбора даты.

17.3. Виджет Dialog

Dialog — это плавающее окно, которое содержит область заголовка и область информационного наполнения. Окно диалога можно перемещать, изменять его размеры и закрыть, щелкнув по значку '**x**'. Если информационное наполнение превысит максимальную высоту, то полоса прокрутки появится автоматически. Внешний вид виджета Dialog в одном из многочисленных вариантов оформления представлен на рис. 17.3.

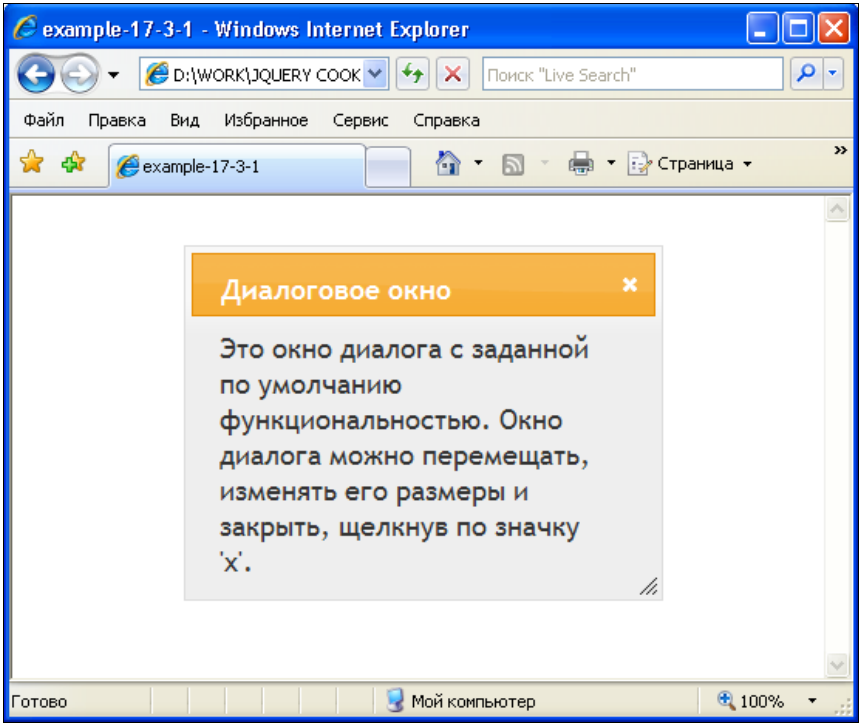


Рис. 17.3. Виджет Dialog в одном из многочисленных вариантов оформления

ЗАДАЧА

Необходимо применить на веб-странице виджет Dialog.

Решение

Решение этой задачи с использованием темы оформления "cupertino" приведено в листинге 17.3.1.

Листинг 17.3.1. Использование виджета Dialog

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/cupertino/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
```

```
$(function(){
  $("#dialog").dialog();
});
</script>
</head>
<body>
<div id="dialog" title="Диалоговое окно">
  <p>Это окно диалога с заданной по умолчанию функциональностью. Окно диалога
можно перемещать, изменять его размеры и закрыть, щелкнув по значку 'x'.</p>
</div>
</body>
</html>
```

Обсуждение

Сначала мы подключили файл стилей css/cupertino/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Вы можете выбрать для себя любую тему, которая придется вам по душе. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором заключена функциональность виджета Dialog и плагины, позволяющие перемещать диалоговое окно (`draggable`) и изменять его размеры (`resizable`).

К HTML-разметке нет никаких особенных требований. Dialog — это элемент `div`, в атрибуте `title` которого указывается текст, отображаемый в заголовке диалогового окна. Все, что содержится внутри элемента `div`, считается информационным наполнением диалогового окна.

За связь виджета Dialog с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.3.2.

Листинг 17.3.2. Использование виджета Dialog

```
<script type="text/javascript">
$(function(){
  $("#dialog").dialog();
});
</script>
```

Пример из листинга 17.3.2 реализует только базовую функциональность виджета Dialog. Но мы попробуем передать значения некоторых опций, чтобы научиться настраивать виджет для решения своих задач (листинг 17.3.3).

Листинг 17.3.3. Использование виджета Dialog

```
<script type="text/javascript">
$(function(){
  $("#dialog").dialog({
    position: ['right','top'],
```

```
    buttons: {
      "Применить": function() { alert('Нажата кнопка "Применить"'); },
      "Закрыть": function() { $(this).dialog("close"); }
    }
  });
});
</script>
```

В примере из листинга 17.3.3 мы передали методу `dialog` две опции — `position` и `buttons`. С помощью опции `position` мы заставляем диалоговое окно появляться в правом верхнем углу. В опции `buttons` определили две кнопки — **Применить** и **Закрыть**, с каждой из которых связали callback-функцию. При нажатии кнопки **Применить** будет показано окно предупреждения, а при нажатии кнопки **Закрыть** диалоговое окно будет закрыто.

В табл. 17.7 приведен полный список доступных опций.

Таблица 17.7. *Опции виджета Dialog*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Dialog будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия
`autoOpen`	Значение по умолчанию — `true`, что означает автоматическое появление диалогового окна при вызове метода `dialod`. Если установить значение опции в `false`, то диалоговое окно будет находиться в скрытом состоянии и сделать его видимым можно будет с помощью `.dialog('open')`
`buttons`	В этой опции передают объект, в котором можно определить кнопки, отображаемые в диалоговом окне, и связать их с callback-функциями (см. листинг 17.3.3). Название свойства — текст, отображаемый на кнопке, значение — функция, которая будет вызвана при нажатии кнопки. Контекст функции — элемент, являющийся диалогом. Доступ к самой кнопке можно получить через свойство `target` объекта `event`
`buttons`	Также в этой опции можно передать массив, в котором можно определить кнопки, отображаемые в диалоговом окне. Каждый элемент массива должен представлять собой объект, описывающий необходимые кнопки
`closeOnEscape`	Значение по умолчанию — `true` (диалоговое окно закрывается при нажатии клавиши <Escape>). Установив значение `false` можно запретить это действие
`closeText`	Задает текст для кнопки **Закрыть**. Обратите внимание, что при использовании стандартных тем оформления этой кнопки просто не видно
`dialogClass`	Указанное в этой опции имя класса (или классов) будет применено к диалоговому окну для дополнительного оформления
`draggable`	Значение по умолчанию — `true`, что дает возможность перемещения диалогового окна. Если установить значение `false`, перемещение станет невозможным

Таблица 17.7 *(окончание)*

Опция	Описание
height	Значение по умолчанию — auto (высота диалогового окна определяется его содержимым). Можно передать значение высоты диалогового окна в пикселах, например: height: 300. В этом случае, если содержимое будет превышать установленный размер, появится вертикальная полоса прокрутки
hide	В этой опции можно определить эффект, который будет использоваться при закрытии диалога, например: hide: 'slide'. Значение по умолчанию — null (никаких эффектов не применяется)
maxHeight	Максимальная высота, до которой может быть изменен размер диалога, в пикселах
maxWidth	Максимальная ширина, до которой может быть изменен размер диалога, в пикселах
minHeight	Минимальная высота, до которой может быть изменен размер диалога, в пикселах
minWidth	Минимальная ширина, до которой может быть изменен размер диалога, в пикселах
modal	Если установить значение true для этой опции, диалог станет модальным. То есть другие элементы на веб-странице будут заблокированы, и пользователь не сможет с ними взаимодействовать. Это будет достигнуто с помощью создания дополнительного слоя, находящегося ниже диалога, но выше остальных элементов веб-страницы
position	Значением этой опции может быть строка или массив, которые определяют начальное положение диалогового окна. Возможные значения: 'center', 'left', 'right', 'top', 'bottom'. Другой вариант — указание этих же значений в массиве. Например ['right', 'top'] для того, чтобы расположить диалог в правом верхнем углу
resizable	Значение по умолчанию — true, что позволяет изменять размеры диалогового окна. Если установить значение false, то изменение размеров станет невозможным
show	В этой опции можно определить эффект, который будет использоваться при открытии диалога, например: show: 'slide'. Значение по умолчанию — null (никаких эффектов не применяется)
stack	Значение по умолчанию — true, что позволяет окну (при использовании нескольких диалоговых окон на одной веб-странице), получившему фокус, быть отображенным поверх остальных окон. Установив эту опцию в false, можно отменить такое поведение для выбранного окна
title	Значением опции может быть строка, с помощью которой можно переопределить заголовок окна, заданный в HTML-разметке
width	Ширина диалога в пикселах (по умолчанию — 300)
zIndex	Значение z-index диалогового окна (по умолчанию — 1000)

Следующий наш шаг — начнем учиться тому, как можно заставить диалоговое окно реагировать на различные события (листинг 17.3.4).

Листинг 17.3.4. Использование виджета Dialog

```
<script type="text/javascript">
$(function(){
  $("#dialog").dialog({
    position: ['right','center'],
    open: function(event, ui) {
      alert('Произошло событие ' + event.type);
    },
    dragStop: function(event, ui) {
      alert('Произошло событие ' + event.type);
    }
  });
});
</script>
```

В листинге 17.3.4 мы определили опции `open` и `dragStop`, значением которых являются callback-функции, вызываемые при наступлении событий открытия диалогового окна и окончания его перемещения. Эти callback-функции могут принимать два аргумента. Первый — объект `event`, второй аргумент — специальный объект `ui`, в свойствах которого содержится следующая информация:

- `ui.position` — объект, в свойствах `top` и `left` которого содержатся значения положения перемещаемого элемента относительно родительского элемента;
- `ui.offset` — объект, в свойствах `top` и `left` которого содержатся значения положения перемещаемого элемента относительно документа.

На примере из листинга 17.3.5 показано, как можно извлечь необходимую информацию из доступного объекта jQuery.

Листинг 17.3.5. Использование виджета Dialog

```
<script type="text/javascript">
$(function(){
  // добавим к body красную рамку шириной 15px (для наглядности)
  $("body").css("border","15px dotted #F00");
  $("#dialog").dialog({
    dragStop: function(event, ui) {
      alert("ui.offset.top = " + ui.offset.top +
            "\nui.offset.left = " + ui.offset.left +
            "\n\nui.position.top = " + ui.position.top +
            "\nui.position.left = " + ui.position.left);
    }
  });
});
</script>
```

Обратите внимание на разницу между значениями ui.offset.top и ui.position.top в появившемся после окончания перемещения окне предупреждения — она будет составлять 15 пикселов. Это как раз ширина заданной нами рамки.

В табл. 17.8 приведены описания событий виджета Dialog и названия опций, в которых можно определить callback-функции, вызываемые при наступлении события.

Таблица 17.8. *События виджета Dialog*

Опция	Событие	Описание
`create`	`dialogcreate`	Наступает в момент инициализации
`beforeClose`	`dialogbeforeclose`	Наступает, когда вы пытаетесь закрыть диалоговое окно. Если связанная с этим событием функция вернет `false` — это предотвратит закрытие диалога
`open`	`dialogopen`	Наступает при открытии диалога
`focus`	`dialogfocus`	Наступает в момент получения фокуса диалоговым окном
`dragStart`	`dragStart`	Наступает в начале перемещения диалогового окна
`drag`	`drag`	Происходит постоянно во время процесса перемещения диалогового окна
`dragStop`	`dragStop`	Наступает в конце перемещения диалогового окна
`resizeStart`	`resizeStart`	Наступает в начале изменения размеров диалогового окна
`resize`	`resize`	Происходит постоянно во время процесса изменения размеров диалогового окна
`resizeStop`	`resizeStop`	Наступает в конце изменения размеров диалогового окна
`close`	`dialogclose`	Наступает, когда диалоговое окно было успешно закрыто

А теперь рассмотрим несколько методов, с помощью которых можно еще более гибко управлять виджетом Dialog. Описания методов приведены в табл. 17.9.

Таблица 17.9. *Методы виджета Dialog*

Метод	Описание
`destroy` `.dialog('destroy')`	Полностью удаляет всю функциональность виджета Dialog. Возвращает элементы в состояние, предшествующее инициализации
`disable` `.dialog ('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
`enable` `.dialog ('enable')`	Разрешает всю функциональность виджета, если ранее она была запрещена методом `disable`
`option` `.dialog ('option', optionName, [value])`	Устанавливает значение любой опции виджета после инициализации

Таблица 17.9 *(окончание)*

Метод	Описание
option `.dialog ('option', optionName)`	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget `.dialog ('widget')`	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Dialog
close `.dialog ('close')`	Закрывает диалоговое окно
isOpen `.dialog ('isOpen')`	Метод вернет `true`, если диалоговое окно уже открыто
moveToTop `.dialog ('moveToTop')`	Помещает диалоговое окно поверх других диалоговых окон, если их больше двух
open `.dialog ('open')`	Открывает диалоговое окно

В листинге 17.3.6 приведен пример использования методов `open` и `close`.

Листинг 17.3.6. Использование виджета Dialog

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-3-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/cupertino/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#dialog").dialog({
    autoOpen: false
  });
  $("#openD").click(function(){
    $("#dialog").dialog("open");
  });
  $("#closeD").click(function(){
    $("#dialog").dialog("close");
  });
});
</script>
</head>
```

```
<body>
<a id="openD" href="#">Открыть диалог</a>
<div id="dialog" title="Диалоговое окно">
  <p>Это информационное наполнение.</p>
</div>
<a id="closeD" href="#">Закрыть диалог</a>
</body>
</html>
```

В примере, приведенном в листинге 17.3.6, мы инициализируем виджет, передавая значение `false` в опции `autoOpen`, скрывая, таким образом, диалоговое окно. В HTML-разметке появились две ссылки — **Открыть диалог** и **Закрыть диалог**. Мы связали щелчок по ссылкам с вызовом соответствующих методов — `open` и `close`. Теперь открывать и закрывать диалог можно, щелкая по ссылкам.

17.4. Виджет Progressbar

Этот виджет разработан, чтобы просто отображать процент выполнения какого-либо процесса. Внешний вид виджета Progressbar в одном из вариантов оформления представлен на рис. 17.4.

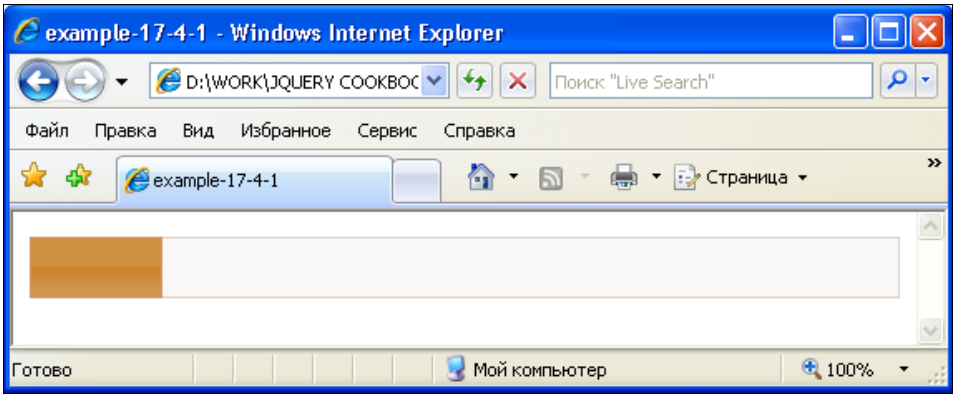


Рис. 17.4. Виджет Progressbar в одном из вариантов оформления

ЗАДАЧА

Необходимо применить на веб-странице виджет Progressbar.

Решение

Решение этой задачи приведено в листинге 17.4.1. Выбрана тема оформления "humanity".

Листинг 17.4.1. Использование виджета Progressbar

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-4-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/humanity/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#progressbar").progressbar({
    value: 15
  });
});
</script>
</head>
<body>
<div id="progressbar"></div>
<button>Добавить 5</button>
</body>
</html>
```

Обсуждение

В примере из листинга 17.4.1 мы подключили файл стилей css/humanity/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Вы можете выбрать для себя любую тему, которая придется вам по душе. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором заключена функциональность виджета Progressbar. Этот виджет имеет всего две доступные опции (табл. 17.10), одну из которых мы сразу же задали в примере. При инициализации виджета с помощью опции `value` можно установить начальное значение, отображаемое на шкале.

Таблица 17.10. *Опции виджета Progressbar*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Progressbar будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия
`value`	Значением является число, которое задает положение шкалы

Событий, на которые виджет Progressbar умеет реагировать, всего три (табл. 17.11). Например, в опции change можно определить обработчик события, наступающего при изменении значения на шкале виджета (листинг 17.4.2).

Таблица 17.11. *События виджета Dialog*

Опция	Событие	Описание
create	progressbarcreate	Наступает в момент инициализации
change	progressbarchange	Наступает, когда изменяется значение шкалы виджета
complete	progressbarcomplete	Наступает при достижении максимального значения на шкале

Листинг 17.4.2. Использование виджета Progressbar

```
<script type="text/javascript">
$(function(){
  $("#progressbar").progressbar({
    value: 15,
    change: function(event, ui) {
      alert('Событие ' + event.type);
    },
    complete: function(event, ui) {
      alert('Событие ' + event.type);
    }
  });
  $("button").click(function(){
    var currentVal = $("#progressbar").progressbar("option", "value");
    if(currentVal<100) {
      $("#progressbar").progressbar("option", "value", currentVal+5);
    }
  });
});
</script>
```

В листинге 17.4.2 мы определили опцию change, значением которой является callback-функция, вызываемая при наступлении события изменения значения шкалы, и опцию complete, позволяющую вызвать функцию при достижении максимально возможного значения шкалы. Эти callback-функции могут принимать два аргумента. Первый — объект event, второй аргумент — специальный объект ui. Нужно заметить, что хотя сам объект ui существует, но ни одно из его свойств не определено.

При щелчке на кнопке мы получаем значение опции value и, если значение меньше 100, устанавливаем новое значение шкалы с помощью одного из методов, описания которых приведены в табл. 17.12.

В момент, когда шкала достигнет максимально возможного значения, появится окно предупреждения, в котором будет выведен тип события `progressbarcomplete`.

Таблица 17.12. *Методы виджета Progressbar*

Метод	Описание
destroy `.progressbar('destroy')`	Полностью удаляет всю функциональность виджета Progressbar. Возвращает элементы в состояние, предшествующее инициализации
disable `.progressbar ('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.progressbar ('enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`
option `.progressbar ('option', optionName , [value])`	Устанавливает значение любой опции виджета после инициализации
option **`.progressbar ('option', optionName)`**	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget `.progressbar ('widget')`	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Progressbar
value `.progressbar ('value', [value])`	С помощью этого метода можно получить или установить значение любой опции виджета после инициализации

Пример использования метода `option` был приведен в листинге 17.4.2.

17.5. Виджет Slider

Виджет Slider позволяет превратить обычный элемент `div` в шкалу с бегунком, который можно перемещать с помощью указателя мыши или клавишами. Внешний вид виджета Slider в одном из многочисленных вариантов оформления представлен на рис. 17.5.

ЗАДАЧА

Необходимо применить на веб-странице виджет Slider.

Решение

Решение этой задачи приведено в листинге 17.5.1. Для оформления использована тема "hot-sneaks".

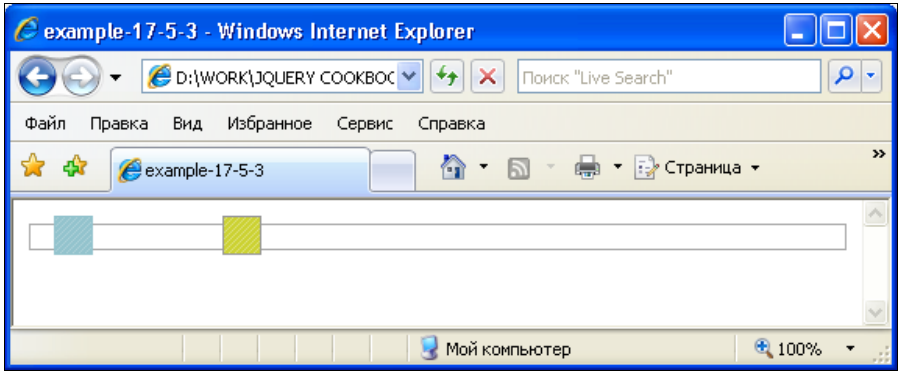


Рис. 17.5. Виджет Slider в одном из многочисленных вариантов оформления

Листинг 17.5.1. Использование виджета Slider

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-5-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/hot-sneaks/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#slider").slider();
});
</script>
</head>
<body>
<div id="slider"></div>
<button id="getter">Get Option</button>
<button id="setter">Set Option</button>
</body>
</html>
```

Обсуждение

В примере из листинга 17.5.1 мы подключили файл стилей css/hot-sneaks/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и виджета Slider. Обратите внимание на HTML-разметку — это всего-навсего один элемент

div. За связь виджета с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.5.2.

Листинг 17.5.2. Использование виджета Slider

```
<script type="text/javascript">
$(function(){
  $("#slider").slider();
});
</script>
```

В листинге 17.5.2 приведен фрагмент кода, реализующий базовую функциональность виджета Slider. Иными словами, все возможные настройки, которые можно было бы передать этому виджету, при такой реализации имеют значения по умолчанию.

Постепенно продвигаясь в освоении виджета, попробуем передать ему некоторые настройки. В листинге 17.5.3 приведен только интересующий нас фрагмент кода, поскольку и подключаемые файлы, и HTML-разметка остаются без изменений.

Листинг 17.5.3. Использование виджета Slider

```
<script type="text/javascript">
$(function(){
  $("#slider").slider({
    min: 10,
    max: 200,
    values: [20,60]
  });
});
</script>
```

Давайте разберем пример из листинга 17.5.3, в котором мы передали виджету некоторые настройки, отличные от настроек по умолчанию.

Во-первых, с помощью опций min и max мы определили минимальное и максимальное значения шкалы, а во-вторых, с помощью опции values определили начальные значения сразу для двух бегунков, т. е. в таком виде шкала позволит выбирать не одно значение, а диапазон.

Описания всех доступных опций приведены в табл. 17.13.

Таблица 17.13. *Опции виджета Slider*

Опция	Описание
disabled	Значение по умолчанию — false. Если установить значение true, то при инициализации функциональность виджета Slider будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия

Таблица 17.13 *(окончание)*

Опция	Описание
animate	Значение по умолчанию — false. В этом случае щелчок на шкале приведет к тому, что бегунок моментально переместится в указанное место. Если установить значение true — перемещение бегунка будет выполнено с анимацией
max	Максимальное значение шкалы. По умолчанию — 100
min	Минимальное значение шкалы. По умолчанию — 0
orientation	Определяет расположение виджета. Может принимать значения 'horizontal' и 'vertical'. Значение по умолчанию — 'auto' (при этом нужная ориентация определяется правильно). Если этого не происходит, можно указать ориентацию принудительно
range	Значение по умолчанию — false. Если установить значение true и при этом используются два бегунка, то между ними шкала будет отображаться иным стилем, нежели основная. Два других возможных значения — 'min' и 'max' при одном бегунке. В первом случае иным стилем будет отображаться часть шкалы слева (или ниже) от бегунка, во втором случае — справа (или выше)
step	Позволяет задать шаг изменения значений шкалы между минимальным и максимальным значением. Разность между максимальным и минимальным значением шкалы должна без остатка делиться на значение этой опции. Значение по умолчанию — 1
value	Определяет положение бегунка. При наличии более чем одного бегунка, определяет положение первого. Значение по умолчанию — 0
values	Массив, в котором могут быть переданы значения, определяющие положения бегунков на шкале. Если при этом опция range имеет значение true, длина массива должна быть равна 2. Значение по умолчанию — null

От настройки виджета Slider с помощью доступных опций перейдем к изучению его возможностей по реагированию на различные события. Сделаем это на примере из листинга 17.5.4.

Листинг 17.5.4. Использование виджета Slider

```
<script type="text/javascript">
$(function(){
  $("#slider").slider({
    min: 10,
    max: 200,
    values: [20,60],
    stop: function(event, ui) {
      alert("Событие " + event.type);
    }
  });
});
</script>
```

В листинге 17.5.4 определяем опцию `stop`. Значением опции является callback-функция, вызываемая по окончании перемещения бегунка. Эта callback-функция может принимать два аргумента. Первый — объект `event`, второй аргумент — объект `ui`, в свойствах которого содержится некоторая информация, представляющая интерес:

- `ui.value` — текущее значение опции `value` при использовании одного бегунка (при наличии двух и более бегунков — положение первого бегунка);
- `ui.values` — текущее значение опции `values` для двух и более бегунков.

На примере из листинга 17.5.5 показано, как можно получить доступ к свойствам объекта `ui`.

`ui.handle` — DOM-элемент, который представляет собой бегунок, находящийся в процессе перемещения.

Листинг 17.5.5. Использование виджета Slider

```
<script type="text/javascript">
$(function(){
  $("#slider").slider({
    min: 10,
    max: 200,
    values: [20,60],
    stop: function(event, ui) {
      alert("Положение бегунков: " + ui.values);
    }
  });
});
</script>
```

В табл. 17.14 приведены описания событий виджета Slider и названия опций, в которых можно определять callback-функции, вызываемые при наступлении соответствующих событий.

Таблица 17.14. *События виджета Slider*

Опция	Событие	Описание
`create`	`slidecreate`	Наступает в момент инициализации
`start`	`slidestart`	Происходит в момент начала перемещения бегунка
`slide`	`slide`	Происходит постоянно во все время перемещения бегунка
`change`	`slidechange`	Происходит при окончании перемещения бегунка или если положение бегунка было изменено программно
`stop`	`slidestop`	Наступает в момент окончания перемещения бегунка

Перейдем к рассмотрению методов виджета Slider (табл. 17.15).

Таблица 17.15. *Методы виджета Slider*

Метод	Описание
destroy `.slider('destroy')`	Полностью удаляет всю функциональность виджета Slider. Возвращает элементы в состояние, предшествующее инициализации
disable `.slider('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.slider('enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`
option `.slider('option', optionName , [value])`	Устанавливает значение любой опции виджета после инициализации
option `.slider('option', optionName)`	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget `.slider ('widget')`	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Slider
value `.slider('value', [value])`	С помощью этого метода можно получить или установить значение бегунка (при наличии единственного бегунка)
values `.slider('values', index, [value])`	С помощью этого метода можно получить или установить значения бегунков (для двух и более бегунков)

В листинге 17.5.6 приведен пример того, как можно получить значение какой-либо опции и изменить это значение уже после инициализации виджета.

Листинг 17.5.6. Использование виджета Slider

```
<script type="text/javascript">
$(function(){
  $("#slider").slider({
    min: 10,
    max: 200,
    values: [20,60]
  });
  $("#getter").click(function(){
    alert('Положение: ' + $("#slider").slider('option', 'values'));
  });
  $("#setter").click(function(){
    $("#slider").slider('values', 0, 15);
```

```
    $("#slider").slider('values', 1, 195);
  });
});
</script>
```

В примере из листинга 17.5.6, при инициализации виджета, мы задали опции, с помощью которых ограничили диапазон значений от 10 до 200 и установили начальные значения двух бегунков (для первого — 20, а для второго — 60). Затем наступило время использовать две кнопки, на которые до сих пор мы внимания не обращали. Мы связали с этими кнопками событие `click`. Если теперь щелкнуть по кнопке с надписью **Get Option**, мы увидим окно предупреждения, где будут выведены значения, характеризующие текущие значения бегунков. При щелчке по кнопке с надписью **Set Option** для первого бегунка будет установлено значение 15, а для второго — 195. Попробуйте щелкнуть по кнопке, и вы увидите, как на ваших глазах бегунки изменят свое положение.

17.6. Виджет Tabs

Виджет Tabs помогает разделить информационное наполнение между несколькими вкладками. Это может быть полезно при дефиците свободного места на веб-странице. Информационное наполнение вкладок можно загрузить с помощью AJAX-запроса. Внешний вид виджета Tabs в одном из вариантов оформления представлен на рис. 17.6.

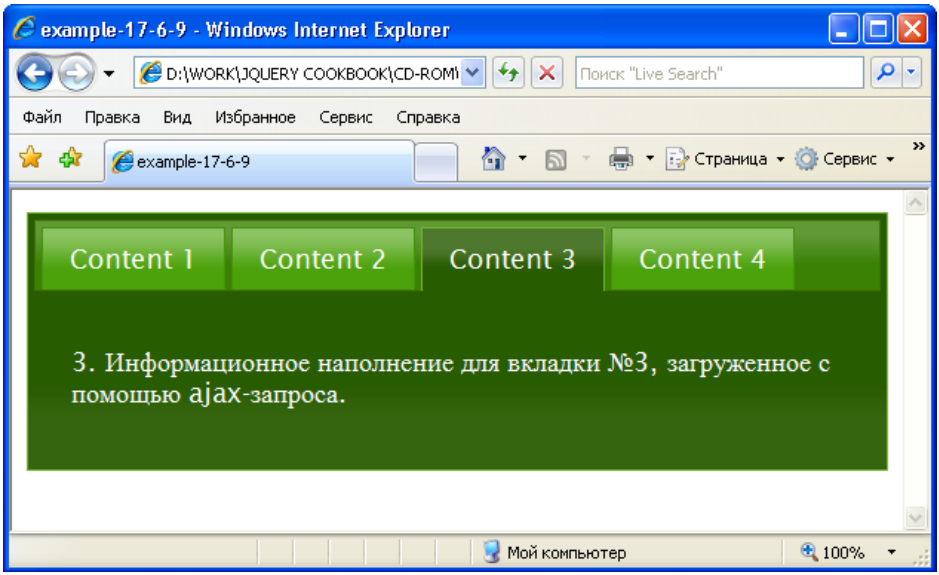


Рис. 17.6. Виджет Tabs в одном из вариантов оформления

ЗАДАЧА

Необходимо применить на веб-странице виджет Tabs.

Решение

Решение этой задачи с использованием темы оформления "le-frog" приведено в листинге 17.6.1.

Листинг 17.6.1. Использование виджета Tabs

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-6-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script src="js/jquery.cookie.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#tabs").tabs();
});
</script>
</head>
<body>
<div id="tabs">
  <ul>
    <li><a href="#tabs-1">Nunc tincidunt</a></li>
    <li><a href="#tabs-2">Proin dolor</a></li>
    <li><a href="#tabs-3">Aenean lacinia</a></li>
  </ul>
  <div id="tabs-1">
    <p>Tab 1 content</p>
  </div>
  <div id="tabs-2">
    <p>Tab 2 content</p>
  </div>
  <div id="tabs-3">
    <p>Tab 3 content</p>
  </div>
</div>
<button id="addTab">Add</button>
<button id="removeTab">Remove</button>
<button id="enableTab">Enable</button>
<button id="disableTab">Disable</button>
<button id="startRotate">Start Rotate</button>
```

```
<button id="stopRotate">Stop Rotate</button>
<button id="getCookie">Get Cookie</button>
</body>
</html>
```

Обсуждение

Виджет Tabs требует определенной HTML-разметки (см. листинг 17.6.1). Ярлыками вкладок являются элементы `li`, в атрибутах `href` которых содержатся идентификаторы соответствующих им элементов `div`, представляющие собой область информационного наполнения вкладки. Внутри элементов `div` содержимое может быть практически любым.

Итак, сначала мы подключили файл стилей css/le-frog/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и виджета Tabs. Файл js/jquery.cookie.min.js используется при необходимости работы с `cookie`. За связь виджета с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.6.2.

Листинг 17.6.2. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs();
});
</script>
```

Виджет Tabs способен принимать много различных опций, но пока в листинге 17.6.2 приведен фрагмент кода, который реализует виджет с настройками по умолчанию.

В примере из листинга 17.6.3 попробуем передать виджету некоторые опции. Мы приведем только интересующий нас фрагмент кода, поскольку и подключаемые файлы, и HTML-разметка остаются без изменений.

Листинг 17.6.3. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs({
    event: "mouseover",
    selected: 2
  });
});
</script>
```

В примере из листинга 17.6.3 мы заставили виджет Tabs переключать вкладки по событию `mouseover` (взамен установленного по умолчанию `click`) и открывать при инициализации третью вкладку (отсчет ведется от нуля).

Приведем еще пару интересных примеров (листинги 17.6.4 и 17.6.5).

Листинг 17.6.4. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs({
    fx: { opacity: "toggle", duration: "slow" }
  });
});
</script>
```

В примере из листинга 17.6.4 с помощью опции `fx` создаем анимационный эффект при переключении вкладок.

А пример из листинга 17.6.5 демонстрирует приемы использования `cookie` для запоминания последней открытой вкладки для того, чтобы впоследствии открыть ее при инициализации.

Листинг 17.6.5. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs({
    cookie: { expires: 7, name: 'startTab' }
  });
  $("#getCookie").click(function(){
    var cook = $("#tabs").tabs("option", "cookie");
    alert("name: " + cook.name + "\nexpires: " + cook.expires);
  });
});
</script>
```

В примере из листинга 17.6.5 мы установили `cookie` с именем `startTab` сроком на семь дней. Щелчок по кнопке с идентификатором `getCookie` приведет к тому, что в окне предупреждения мы увидим имя установленной `cookie` и срок, на который она была установлена. Нужно отметить, что для реализации этой возможности необходимо подключать дополнительный файл плагина jQuery Cookie (см. *главу 16*).

Описания всех возможных опций приведены в табл. 17.16.

Таблица 17.16. *Опции виджета Tabs*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Tabs будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия

Таблица 17.16 *(окончание)*

Опция	Описание
`ajaxOptions`	Опции AJAX-запроса (подробная информация об опциях, используемых для управления AJAX-запросами, приведена в *главе 9*), которые служат для загрузки удаленного содержимого в область информационного наполнения. Значение по умолчанию — `null`
`cache`	Опция определяет, кэшировать или нет информационное наполнение вкладки, загружаемое с помощью AJAX-запросов, т. е. загрузить содержимое только однажды или загружать каждый раз, когда совершен щелчок на ярлыке вкладки. Значение по умолчанию — `false` (кэширование не производится). Обратите внимание — чтобы воспрепятствовать фактическому кэшированию данных браузером, вы должны дополнительно обеспечить значение `false` в опциях самого AJAX-запроса
`collapsible`	Значение по умолчанию — `false`. Если передать в этой опции значение `true`, то каждый следующий щелчок на ярлыке вкладки будет скрывать и открывать область информационного наполнения
`cookie`	Запоминает последнюю выбранную вкладку в `cookie`. В дальнейшем служит для определения вкладки, открываемой по умолчанию (если не используется опция `selected`). Требует подключения плагина cookie (см. *главу 16*). Возможно присвоение имени `cookie` через опцию `name`
`disabled`	Массив, содержащий индексы вкладок (отсчет ведется от нуля), которые должны быть недоступными при инициализации виджета. Например, `disabled: [1, 2]` сделает недоступными вторую и третью вкладки
`event`	Тип события, которое используется для переключения вкладок. Значение по умолчанию — `'click'`. Второе возможное значение — `'mouseover'`
`fx`	Устанавливает анимационные эффекты при открытии/закрытии вкладок. Например: `fx: { opacity: "toggle", duration: "slow" }`
`idPrefix`	Значение по умолчанию — строка `'ui-tabs-'`. Используется для генерации идентификатора при создании дополнительных вкладок после инициализации виджета, для вкладок, в которые содержимое загружается с помощью AJAX-запросов
`panelTemplate`	HTML-шаблон, из которого создается новая область информационного наполнения вкладки при ее создании с помощью метода `add` или "на лету" посредством AJAX-запроса. Значение по умолчанию — `'<div></div>'`
`selected`	Индекс вкладки (отсчет ведется от нуля), которая должна быть открыта при инициализации виджета. Значение по умолчанию — 0, т. е. открывается первая вкладка. Чтобы сделать невыбранными все вкладки, нужно задать значение `-1`
`spinner`	HTML-код, указанный в этой строке, отображается на ярлыке вкладки во время загрузки удаленного содержимого в область информационного наполнения. Если передать пустую строку, эта возможность будет деактивирована. По умолчанию используется `<em>Loading</em>`
`tabTemplate`	HTML-шаблон, из которого создаются новые ярлыки вкладок при их создании методом `add`. `#{href} and #{label}` заменяются на URL и название вкладки, переданными в аргументах метода `add`. По умолчанию имеет значение `'<li><a href="#{href}"><span>#{label}</span></a></li>'`

Точно так же, как и другие виджеты, виджет Tabs умеет реагировать на разнообразные события (листинг 17.6.6).

Листинг 17.6.6. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs({
    selected: 2,
    select: function(event, ui) {
      alert("Событие: " + event.type +
            "\nui.tab: " + ui.tab +
            "\nui.panel: " + ui.panel.innerHTML +
            "\nui.index: " + ui.index);
    }
  });
});
</script>
```

В примере из листинга 17.6.6 мы определили в опции `select` callback-функцию, которая будет вызываться при наступлении события `tabsselect`, т. е. в тот момент, когда был совершен щелчок по ярлыку вкладки. Callback-функция принимает два аргумента: первый — объект `event`, второй — специальный объект `ui`. Организовав доступ к свойствам этих объектов, можно получить довольно полезную информацию (мы выводим ее в окне предупреждения):

- `ui.tab` — содержит полный URL активизированной в настоящий момент вкладки;
- `ui.panel` — элемент, представляющий информационное наполнение активизированной вкладки;
- `ui.index` — индекс активизированной вкладки (отсчет ведется от нуля).

В табл. 17.17 приведены описания событий виджета Tabs и названия опций, в которых можно определить callback-функции, вызываемые при наступлении события.

Таблица 17.17. *События виджета Tabs*

Опция	Событие	Описание
`create`	`tabscreate`	Наступает в момент инициализации
`select`	`tabsselect`	Происходит в момент щелчка по ярлыку вкладки (при наведении указателя мыши на вкладку, если используется опция `event: 'mouseover'`). При щелчке на ярлыке активной вкладки ничего не происходит
`load`	`tabsload`	Происходит, когда в область информационного наполнения вкладки было загружено содержимое из внешнего файла
`show`	`tabsshow`	Наступает в момент отображения содержимого вкладки

Таблица 17.17 *(окончание)*

Опция	Событие	Описание
add	tabsadd	Происходит каждый раз, когда добавлена новая вкладка
remove	tabsremove	Происходит каждый раз, когда удалена какая-либо вкладка
enable	tabsenable	Наступает в момент, когда ранее недоступная вкладка становится доступной
disable	tabsdisable	Наступает в момент, когда вкладка становится недоступной

Конечно же, виджет Tabs имеет и некоторые методы, позволяющие еще более расширить функциональность.

В листинге 17.6.7 мы перейдем к изучению методов виджета Tabs.

Листинг 17.6.7. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs({
    selected: 2,
  });
  // добавляем вкладку при щелчке на кнопке add
  $("#addTab").click(function(){
    $("#tabs").tabs("add", "#tabs-4", "Tab 4", 3);
    $("#tabs-4").append("<p>Tab 4 content</p>");
    $(this).attr("disabled","disabled");
    $("#removeTab").attr("disabled","");
  });
  // удаляем вкладку при щелчке на кнопке remove
  $("#removeTab").click(function(){
    $("#tabs").tabs("remove", 3);
    $(this).attr("disabled","disabled");
    $("#addTab").attr("disabled","");
  }).attr("disabled","disabled");
  // активируем вкладку при щелчке на кнопке enable
  $("#enableTab").click(function(){
    $("#tabs").tabs("enable", 2);
  });
  // деактивируем вкладку при щелчке на кнопке disable
  $("#disableTab").click(function(){
    $("#tabs").tabs("disable", 2);
  });
});
</script>
```

Подробно разберем пример, приведенный в листинге 17.6.7. Здесь нам пригодятся кнопки, которые мы до этого времени не задействовали. Для кнопки с идентифика-

тором **Add** определяем обработчик события `click`. При щелчке на этой кнопке будет вызвана callback-функция, которая применит к виджету метод `add` и добавит еще одну вкладку с идентификатором `#tabs-4` и надписью на ярлыке — **Tab4**. Последний аргумент метода — индекс создаваемой вкладки. Поскольку отсчет ведется от нуля, число 3 означает, что создаваемая вкладка — четвертая по счету. Вторая строка добавит параграф `p` в область информационного наполнения вкладки. Подобным образом определим обработчик для кнопки **Remove**. Здесь мы используем метод `remove`, чтобы удалить ранее созданную вкладку с индексом 3. С кнопками **Enable** и **Disable** связаны обработчики, которые активируют и деактивируют третью по счету вкладку.

В листинге 17.6.8 приведен пример использования еще одного очень интересного метода — `rotate`.

Листинг 17.6.8. Использование виджета Tabs

```
<script type="text/javascript">
$(function(){
  $("#tabs").tabs({
    selected: 2
  });
  $("#startRotate").click(function(){
    $("#tabs").tabs('rotate', 3000, true);
  });
  $("#stopRotate").click(function(){
    $("#tabs").tabs('rotate', null);
  });
});
</script>
```

В примере из листинга 17.6.8 при щелчке на кнопке с идентификатором **Start Rotate** будет вызван метод `rotate`, который заставит виджет самостоятельно переключать вкладки каждые три секунды. При щелчке на кнопке с идентификатором **Stop Rotate** автоматическое переключение вкладок будет остановлено.

В табл. 17.18 приведены описания всех доступных методов виджета Tabs.

Таблица 17.18. *Методы виджета Tabs*

Метод	Описание
destroy `.tabs( 'destroy' )`	Полностью удаляет всю функциональность виджета Tabs. Возвращает элементы в состояние, предшествующее инициализации
disable `.tabs('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.tabs('enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`

Таблица 17.18 *(продолжение)*

Метод	Описание
option `.tabs('option', optionName, [value])`	Устанавливает значение любой опции виджета после инициализации
option `.tabs('option', optionName)`	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget `.tabs ('widget')`	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Tabs
add `.tabs('add', url, label, [index])`	Добавляет новую вкладку. Второй аргумент содержит либо идентификатор создаваемой вкладки, (см. требования к разметке), либо является полным URL (относительным или абсолютным, но без поддержки кроссдоменной загрузки) при создании вкладки с использованием AJAX. Третий аргумент — название вкладки, отображаемое на ее ярлыке. Четвертый — индекс создаваемой вкладки (позиции отсчитываются от нуля)
remove `.tabs('remove', index)`	Удаляет вкладку. Второй аргумент — индекс удаляемой вкладки (позиции отсчитываются от нуля)
enable `.tabs('enable', index)`	Делает доступной ранее недоступную вкладку. Чтобы сделать доступными одновременно несколько вкладок, укажите `$('#tabs').data('disabled.tabs', [ ]);`
disable `.tabs('disable', index)`	Делает вкладку недоступной. Активная вкладка не может быть сделана недоступной. Чтобы сделать недоступными одновременно несколько вкладок, укажите `$('#tabs').data('disabled.tabs', [1, 2, 3]);`
select `.tabs('select', index)`	Метод позволяет выбрать вкладку так, как будто был сделан щелчок по ее ярлыку. Второй аргумент — индекс нужной вкладки (позиции отсчитываются от нуля). Он же может быть идентификатором вкладки (см. требования к разметке)
load `.tabs('load', index)`	Программно перезагружает содержимое вкладки, используя AJAX-запрос. Этот метод всегда загружает содержимое, даже если опция `cache` имеет значение `true`. Второй аргумент — индекс нужной вкладки (позиции отсчитываются от нуля)
url `.tabs('url', index, url)`	Изменяет URL, откуда с помощью AJAX-запроса будет загружаться содержимое. Второй аргумент — индекс нужной вкладки (позиции отсчитываются от нуля). Указанный в третьем аргументе URL будет использоваться и для дальнейших загрузок
length `.tabs('length')`	Метод просто возвращает число вкладок

Таблица 17.18 *(окончание)*

Метод	Описание
abort `.tabs('abort')`	Завершает все выполняющиеся AJAX-запросы и анимацию
rotate `.tabs('rotate', ms, [continuing])`	Устанавливает автоматический перебор всех вкладок. Второй аргумент — интервал времени (в миллисекундах), в течение которого очередная вкладка будет активной. Чтобы остановить перебор вкладок во втором аргументе, необходимо передать `0` или `null`. Третий аргумент определяет логику при выборе вкладки пользователем. Значение `true` — перебор вкладок будет продолжен. Значение по умолчанию — `false`

Одна из самых интересных возможностей виджета Tabs — способность загружать содержимое в область информационного наполнения через AJAX-запросы. Эта возможность будет продемонстрирована в листинге 17.6.9. Код приведен полностью, поскольку требования к HTML-разметке здесь несколько иные.

Листинг 17.6.9. Использование виджета Tabs

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-6-9</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  $("#tabs").tabs();
});
</script>
</head>
<body>
<div id="tabs">
  <ul>
    <li><a href="ajax_1.html"><span>Content 1</span></a></li>
    <li><a href="ajax_2.html"><span>Content 2</span></a></li>
    <li><a href="ajax_3.html"><span>Content 3</span></a></li>
    <li><a href="#tabs-4">Content 4</a></li>
  </ul>
  <div id="tabs-4">
    <p>4. Информационное наполнение для вкладки №4.</p>
```

```
  </div>
</div>
</body>
</html>
```

В примере из листинга 17.6.9 мы не передавали виджету никаких дополнительных опций, оставив все настройки по умолчанию. Тем не менее, содержимое области информационного наполнения трех вкладок теперь загружается из внешних файлов с использованием AJAX-запросов. Уделите особое внимание изучению HTML-разметки. Имейте в виду, что вкладки с информацией, загружаемой с помощью AJAX-запросов, вполне успешно могут соседствовать с вкладками, созданными обычным способом.

17.7. Виджет Button

Виджет Button задает стилевое оформление таких стандартных элементов, как `button`, `input` типа `submit` или `reset`, а также элементов `a`, включая соответствующее поведение при наведении указателя мыши. Также виджет может быть применен к элементам `input` типа `radio` и `checkbox`. Внешний вид элементов, оформленных с использованием темы "sunny", представлен на рис. 17.7.

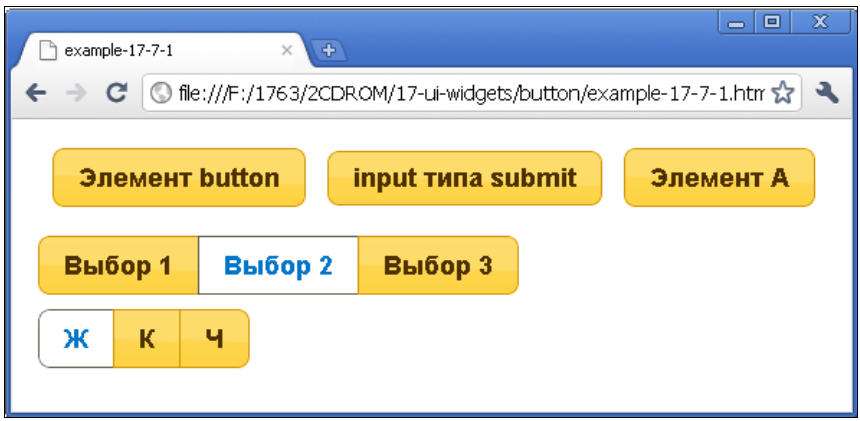


Рис. 17.7. Элементы, стилизованные с помощью виджета Button

ЗАДАЧА

Необходимо применить на веб-странице виджет Button.

Решение

Решение этой задачи с использованием темы оформления "sunny" приведено в листинге 17.7.1.

Листинг 17.7.1. Использование виджета Button

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-7-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/sunny/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
div, button, input, a { margin:10px; }
</style>
<script type="text/javascript">
$(function(){
  $("button, input:submit, a").button().click(function(e){
    alert(e.type);
    return false;
  });
  $("div.set").buttonset();
});
</script>
</head>
<body>
<div>
  <button>Элемент button</button>
  <input value="input типа submit" type="submit" />
  <a href="#">Элемент A</a>
</div>
<div class="set">
  <input type="radio" id="radio1" name="radio" /><label for="radio1">Выбор
1</label>
  <input type="radio" id="radio2" name="radio" checked="checked" /><label
for="radio2">Выбор 2</label>
  <input type="radio" id="radio3" name="radio" /><label for="radio3">Выбор
3</label>
</div>
<div class="set">
  <input type="checkbox" id="check1" /><label for="check1">Ж</label>
  <input type="checkbox" id="check2" /><label for="check2">К</label>
  <input type="checkbox" id="check3" /><label for="check3">Ч</label>
</div>
</body>
</html>
```

Обсуждение

Сначала мы подключили файл стилей css/sunny/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, объединяющий функциональность ядра UI и виджета Button.

Рассмотрим сначала HTML-разметку. В первом элементе `div` находятся элементы `button`, `input` типа `submit` и элемент `a`. В следующих элементах `div`, которые имеют класс `set`, находятся по три элемента `input` типа `radio` и `checkbox`.

За связь виджета с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.7.1a.

Листинг 17.7.1a. Использование виджета Button

```
<script type="text/javascript">
$(function(){
  $("button, input:submit, a").button().click(function(e){
    alert(e.type);
    return false;
  });
  $("div.set").buttonset();
});
</script>
```

Рассмотрим JavaScript-код, приведенный в листинге 17.1.1a. Сначала мы выбираем необходимые элементы, к которым применяем метод `button`. Собственно на этом все и заканчивается. Обработчик события `click`, примененный к этому же набору, служит только в демонстрационных целях.

Но в JavaScript-коде есть еще одна строка, в ней к элементам `div` класса `set` применяется метод `buttonset`. Это дополнительный метод, с помощью которого можно работать с элементами, объединенными в группы, как сделано в примере с элементами `radio` и `checkbox`.

Виджет Button может принимать несколько опций, но в листинге 17.7.1a приведен фрагмент кода, который реализует виджет с настройками по умолчанию.

В примере из листинга 17.7.2 попробуем передать виджету некоторые опции.

Листинг 17.7.2. Использование виджета Button

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-7-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<link type="text/css" href="css/sunny/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
button { margin:5px; }
</style>
<script type="text/javascript">
$(function(){
  $("button:first").button({
      icons: {
        primary: "ui-icon-locked"
      },
      text: false
    }).next().button({
      label: "Иконка слева",
      icons: {
        primary: "ui-icon-locked"
      }
    }).next().button({
      icons: {
        primary: "ui-icon-gear",
        secondary: "ui-icon-triangle-1-s"
      }
    }).next().button({
      icons: {
        primary: "ui-icon-gear",
        secondary: "ui-icon-triangle-1-s"
      },
      text: false
  });
});
</script>
</head>
<body>
<div>
  <button>Иконка без текста</button>
  <button></button>
  <button>Две иконки</button>
  <button>Две иконки, но без текста</button>
</div>
</body>
</html>
```

HTML-код из листинга 17.7.2 — всего лишь четыре элемента `button`, находящиеся внутри элемента `div`.

В JavaScript-коде сначала отышем первую кнопку с помощью выражения `button:first`, указанного в селекторе, и применим к ней метод `button`. Только те-

перь передадим этому методу объект с настройками. В качестве значения свойства `icons` также выступает объект, где возможны два свойства — `primary` и `secondary`. Это имена классов, с помощью которых в файле стилевого оформления позиционируются иконки, представленные в виде CSS-спрайта. Значения этих свойств определяют первую и вторую иконку для соответствующего элемента. В свойстве `text` передаем значение `false`, с помощью чего скрываем надпись на кнопке, которая имеется в разметке.

Далее мы строим цепочку вызовов и, применяя метод `next`, перемещаемся к следующему элементу `button`, с которым также связываем функциональность виджета. На этот раз на кнопке никакой надписи нет, и мы зададим ее, используя свойство `label`.

Дальнейшие операции над оставшимися кнопками аналогичны уже рассмотренным действиям.

Описания всех опций приведены в табл. 17.19.

Таблица 17.19. *Опции виджета Button*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Button будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия
`text`	Значение по умолчанию — `true`. Чтобы скрыть надпись на кнопке, необходимо установить значение `false`. Однако в этом случае должна быть определена опция `icons`, иначе настройка будет проигнорирована
`icons`	Значение по умолчанию — `{ primary: null, secondary: null }`. Иконка **primary** располагается слева от текста (если он присутствует), иконка **secondary** — справа. Значениями свойств `primary` и `secondary` являются имена классов, с помощью которых изображения иконок, представленные в виде CSS-спрайта, позиционируются в стилевом файле
`label`	Строка, определяющая надпись на кнопке. Может содержать HTML

У виджета Button нет никаких специальных событий, на которые он мог бы реагировать, кроме стандартного для всех виджетов `create` (табл. 17.20).

Однако никто не мешает вам связать с кнопками обработчик события `.click()` или например, с элементами `input` обработчики событий `.focusin()` и `.focusout()`.

Таблица 17.20. *События виджета Button*

Опция	Событие	Описание
`create`	`buttoncreate`	Наступает в момент инициализации

В табл. 17.21 приведено описание доступных методов виджета Button.

Таблица 17.21. *Методы виджета Button*

Метод	Описание
destroy `.button( 'destroy' )`	Полностью удаляет всю функциональность виджета Button. Возвращает элементы в состояние, предшествующее инициализации
disable `.button('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.button('enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`
option `.button('option', optionName, [value])`	Устанавливает значение любой опции виджета после инициализации
option `.button('option', optionName)`	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget `.button('widget')`	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Button
refresh `.button('refresh')`	Обновляет визуальное состояние кнопки

17.8. Виджет Autocomplete

Виджет Autocomplete помогает организовать список подходящих значений при заполнении пользователем поля ввода. Внешний вид виджета Autocomplete в одном из вариантов оформления представлен на рис. 17.8.

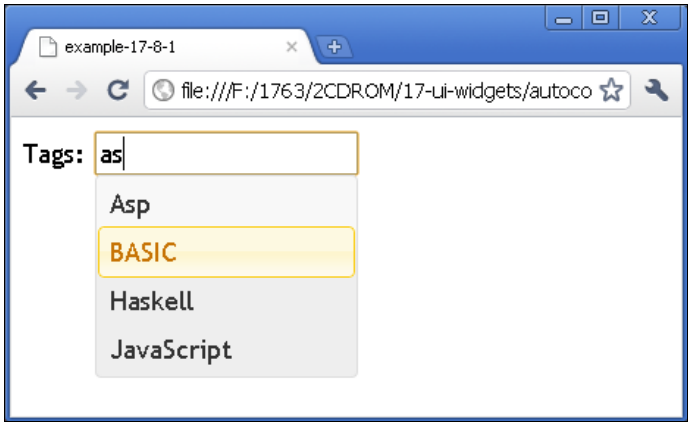


Рис. 17.8. Виджет Autocomplete в одном из вариантов оформления

ЗАДАЧА

Необходимо применить на веб-странице виджет Autocomplete.

Решение

Решение этой задачи с использованием темы оформления "ui-lightness" приведено в листинге 17.8.1.

Листинг 17.8.1. Использование виджета Autocomplete

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-8-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/ui-lightness/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  var availableTags = ["ActionScript","AppleScript","Asp","BASIC",
  "C","C++","Clojure","COBOL","ColdFusion","Erlang","Fortran",
  "Groovy","Haskell","Java","JavaScript","Lisp","Perl","PHP",
  "Python","Ruby","Scala","Scheme"];
  $("#tags").autocomplete({
    source: availableTags
  });
});
</script>
</head>
<body>
<div class="ui-widget">
  <label for="tags">Tags: </label>
  <input id="tags" type="text" />
</div>
</body>
</html>
```

Обсуждение

Виджет Autocomplete не требует сложной HTML-разметки (см. листинг 17.8.1). Достаточно всего лишь поля для ввода текста, т. е. обычного элемента `input`, который имеет значение `text` в атрибуте `type`. Дополнительная разметка добавлена лишь для того, чтобы иметь возможность задать стили в соответствии с выбранной темой оформления.

Итак, сначала мы подключили файл стилей css/ui-lightness/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, объединяющий функциональность ядра UI и виджета Autocomplete. За связь виджета с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 17.8.1а. Обратите внимание, что здесь же определяется массив данных для выпадающего списка подсказок. Это самый простой вариант использования виджета.

Листинг 17.8.1а. Использование виджета Autocomplete

```
<script type="text/javascript">
$(function(){
  var availableTags = ["ActionScript","AppleScript","Asp","BASIC",
  "C","C++","Clojure","COBOL","ColdFusion","Erlang","Fortran",
  "Groovy","Haskell","Java","JavaScript","Lisp","Perl","PHP",
  "Python","Ruby","Scala","Scheme"];
  $("#tags").autocomplete({
    source: availableTags
  });
});
</script>
```

В примере, приведенном в листинге 17.8.1а, мы сразу же передали виджету Autocomplete обязательную опцию `source`, в которой определяется источник используемых данных. В рассмотренном примере источником данных является массив `availableTags`.

Описания всех возможных опций приведены в табл. 17.22.

Таблица 17.22. *Опции виджета Autocomplete*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность виджета Autocomplete будет недоступна, однако ее можно включить впоследствии, например, при выполнении какого-либо условия
`appendTo`	Значение по умолчанию — `'body'`. В качестве значения этой опции допустим селектор jQuery. Определяет, к какому элементу должен быть добавлен выпадающий список подсказок
`delay`	Значение по умолчанию — 300. В этой опции указывается число миллисекунд, которое должно пройти после нажатия очередной клавиши, чтобы активизировался запрос на получение данных. Нулевое значение имеет смысл для локальных данных. При использовании запросов к серверу нулевое значение в этой опции может породить серьезную нагрузку
`minLength`	Значение по умолчанию — 1. В этой опции указывается количество символов, которое должно быть введено в поле ввода прежде, чем активизируются подсказки. Нулевое значение полезно при использовании локальных данных и списков из нескольких позиций. Это значение нужно увеличить при реализации запросов к серверу для получения данных и при наличии больших списков, где одному введенному символу может соответствовать несколько тысяч наименований

Таблица 17.22 *(окончание)*

Опция	Описание
`source`	Опция не имеет значения по умолчанию и должна быть обязательно определена. Значением опции может являться строка, массив или функция. В любом случае в этой опции должен быть указан источник данных

Виджет Autocomplete может реагировать на некоторые события (листинг 17.8.2).

Листинг 17.8.2. Использование виджета Autocomplete

```
<script type="text/javascript">
$(function(){
  var availableTags = ["ActionScript","AppleScript","Asp","BASIC",
  "C","C++","Clojure","COBOL","ColdFusion","Erlang","Fortran",
  "Groovy","Haskell","Java","JavaScript","Lisp","Perl","PHP",
  "Python","Ruby","Scala","Scheme"];
  $("#tags").autocomplete({
    source: availableTags,
    select: function(event, ui) {
      alert('Событие: ' + event.type +
            ',\nзначение: ' + ui.item.value);
    }
  });
});
</script>
```

В примере из листинга 17.8.2 мы определили в опции `select` callback-функцию, которая будет вызываться при наступлении события `autocompleteselect`, т. е. в тот момент, когда был выбран один из пунктов списка подсказок. Callback-функция принимает два аргумента: первый — объект `event`, второй — специальный объект `ui`. Организовав доступ к свойствам этих объектов, можно получить полезную информацию (мы выводим ее в окне предупреждения).

`ui.item` — объект со свойствами `label` и `value`. Здесь `value` — выбранное из списка значение. Свойство `label` содержит одинаковое значение с `value`, если данные представлены в виде массива строк, и может отличаться, если каждый элемент массива данных представлен в виде объекта.

В табл. 17.23 приведены описания событий виджета Autocomplete и названия опций, в которых можно определить callback-функции, вызываемые при наступлении события.

Виджет Autocomplete имеет и некоторые методы, позволяющие еще более расширить функциональность.

В листинге 17.8.3 мы перейдем к изучению методов виджета Autocomplete.

Таблица 17.23. *События виджета Autocomplete*

Опция	Событие	Описание
create	autocompletecreate	Наступает в момент инициализации
search	autocompletesearch	Наступает перед выполнением запроса. Если функция, определенная в этой опции вернет false, запрос не будет отправлен
open	autocompleteopen	Наступает в момент, когда открывается выпадающий список подсказок
focus	autocompletefocus	Происходит всякий раз, когда один из пунктов списка подсказок получает фокус
select	autocompleteselect	Наступает, когда выбран один из пунктов списка подсказок
close	autocompleteclose	Наступает, когда список подсказок закрывается. Событие наступает независимо от того, был выбран один из пунктов или нет
change	autocompletechange	Наступает после того, как выбран один из пунктов списка. Событие всегда наступает после close

Листинг 17.8.3. Использование виджета Autocomplete

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-8-3</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/ui-lightness/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<script type="text/javascript">
$(function(){
  var availableTags = ["ActionScript","AppleScript","Asp","BASIC",
  "C","C++","Clojure","COBOL","ColdFusion","Erlang","Fortran",
  "Groovy","Haskell","Java","JavaScript","Lisp","Perl","PHP",
  "Python","Ruby","Scala","Scheme"];
  $("#tags").autocomplete({
    source: availableTags,
    minLength: 0
  });
  $("#search").click(function(){ $("#tags").autocomplete("search", "as") });
  $("#close").click(function(){ $("#tags").autocomplete("close") });
});
</script>
</head>
```

```
<body>
<button id="search">Искать "as"</button>
<button id="close">Закрыть</button>
<div class="ui-widget">
  <label for="tags">Tags: </label>
  <input id="tags" type="text" />
</div>
</body>
</html>
```

В примере из листинга 17.8.3 в HTML-разметку добавлено две кнопки. При нажатии на кнопку **Search** вызываем одноименный метод используемого виджета и с передачей ему символов "as" — откроется список подсказок с подходящими значениями. Передавать искомую подстроку необязательно. Если она не передана, то откроется весь список полностью. При щелчке на кнопке **Close** будет применен метод `close`, который заставит закрыться список подсказок, если он был открыт.

В табл. 17.24 приведены описания всех доступных методов виджета Autocomplete.

Таблица 17.24. *Методы виджета Autocomplete*

Метод	Описание
destroy `.autocomplete('destroy')`	Полностью удаляет всю функциональность виджета Autocomplete. Возвращает элементы в состояние, предшествующее инициализации
disable `.autocomplete('disable')`	Временно запрещает использование всей функциональности виджета. Вновь разрешить ее можно с помощью метода `enable`
enable `.autocomplete('enable')`	Разрешает использование всей функциональности виджета, если ранее она была запрещена методом `disable`
option `.autocomplete('option', optionName, [value])`	Устанавливает значение любой опции виджета после инициализации
option `.autocomplete('option', optionName)`	С помощью этого метода можно получить значение любой опции виджета после инициализации
widget `.autocomplete('widget')`	Обеспечивает доступ к объекту, который представляет собой элемент с функциональностью Autocomplete
search `.autocomplete('search, [value]')`	С помощью этого метода без передачи второго параметра можно открыть весь список. Если передать во втором параметре строку символов, то будет открыт список с подходящими подсказками
close `.autocomplete('close')`	Закрывает список подсказок, если он был открыт ранее

До сих пор мы рассматривали примеры с использованием локальных данных, но виджет Autocomplete может формировать список подсказок из данных, хранящихся на сервере, в том числе и на удаленном. Пример приведен в листинге 17.8.4. Мы попробуем получать данные с реально работающего сервиса, расположенного по адресу **http://geonames.org**.

Листинг 17.8.4. Использование виджета Autocomplete

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-17-8-4</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/ui-lightness/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
.ui-autocomplete-loading {
  background: #FFF url('css/ui-lightness/images/ui-anim_basic_16x16.gif') right
center no-repeat;
}
#city { width: 25em; }
#log { height: 200px; width: 600px; overflow: auto; }
</style>
<script type="text/javascript">
$(function() {
  $("#city").autocomplete({
    source: function(request,response) {
      $.ajax({
        url: "http://ws.geonames.org/searchJSON",
        dataType: "jsonp",
        data: {
          featureClass: "P",
          style: "full",
          maxRows: 12,
          name_startsWith: request.term
        },
        success: function(data) {
          response($.map(data.geonames, function(item) {
            return {
              label: item.name + ", " + item.countryName,
              value: item.name + " (" + item.countryName + ")" + " [" +
item.lat + ", " + item.lng + "]"
            }
          }));
        }
      });
    },
```

```
    minLength: 3,
    select: function(event,ui) {
      $("<p/>").text(ui.item ? ui.item.value : "Ничего не
                                                выбрано!").prependTo("#log");
      $("#log").attr("scrollTop", 0);
    }
  });
});
</script>
</head>
<body>
<div class="ui-widget">
  <label for="city">Город: </label><input id="city" /><br />
  <span style="font-size:.8em;">Поддерживается <a
href="http://geonames.org">geonames.org</a></span>
</div>
<div id="log" class="ui-widget ui-widget-content"></div>
</body>
</html>
```

Давайте сначала познакомимся с HTML-разметкой из листинга 17.8.4. В первом элементе `div` нас интересует только элемент `input` с идентификатором `city`. Сюда будем вводить начальные буквы (на латинице) населенного пункта, информацию о котором мы хотели бы получить. Элемент `div` с идентификатором `log` служит для занесения в него полученной информации.

Теперь займемся JavaScript-кодом. Метод `autocomplete` свяжем с элементом `input#city` и передадим этому методу объект, содержащий три свойства, определяющие пользовательские настройки, — обязательное свойство `source`, а также свойства `minLength` и `select`.

Мы уже знаем, что обязательное свойство `source` определяет источник используемых данных и может быть массивом (в случае локальных данных) или строкой, в которой указывается `url` обработчика на сервере. Но мы избрали наиболее гибкий способ — в качестве значения свойства `source` определили функцию. Эта функция принимает два аргумента. Первый аргумент — `request` — объект, содержащий единственное свойство `term`, в котором хранится строка, введенная пользователем в поле ввода. Второй аргумент — `response` — функция, с помощью которой будет обрабатываться полученный ответ. Внутри функции, определенной в свойстве `source`, мы имеем практически не ограниченную свободу действий.

Воспользовавшись такой свободой, напишем AJAX-запрос, который будет получать и обрабатывать необходимые данные. В настройках запроса указываем `url`, к которому будем обращаться, а также тип данных, ожидаемых в ответе. В опции `data` определим дополнительные параметры, которые будут отправлены в запросе. Чтобы знать, какие именно параметры необходимо передавать, следует внимательно ознакомиться с документацией по API сервиса, который предполагается использовать.

Перейдем к рассмотрению опции `success`, предполагая, что мы получили ответ на свой запрос.

Внутри функции, определенной нами в опции `success`, вызываем функцию обработки ответа `response`. В аргументе, который мы передаем этой функции, мы можем обрабатывать данные, полученные в ответе сервера так, как нам будет угодно.

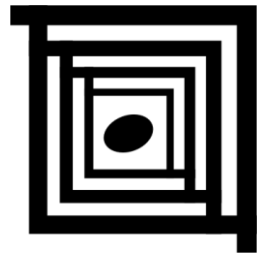
В примере из листинга 17.8.4 мы вызываем метод `$.map`, чтобы применить некоторую функцию к каждому элементу объекта, переданному в первом аргументе. Внутри функции мы можем обращаться к свойствам объекта — `item.countryName`, `item.lng`, `item.lat`. Для того чтобы узнать перечень возможных свойств, необходимо обращаться к документации по API, которую предоставляет веб-сервис.

Написанная нами функция для каждого элемента возвращает объект, содержащий два свойства, которые мы определили самостоятельно на основе полученных данных. Из получившегося массива таких объектов и строится список подсказок.

После такого подробного описания опции `source`, про опцию `minLength` даже говорить нечего — в ней передается количество введенных пользователем символов, прежде чем будет отправлен запрос.

Перейдем к последней опции — `select`. Здесь определяем функцию, вызываемую при выборе одного пункта из списка подсказок. Функция принимает два аргумента. Первый — стандартный объект события, второй — специальный объект `ui`, у которого имеются свойства `label` и `value`. По сути, это один из объектов, массив которых создавался при обработке ответа сервера, а именно выбранный пользователем объект.

ГЛАВА 18



UI jQuery — взаимодействие с элементами страницы

За взаимодействие с элементами пользовательского интерфейса отвечает группа плагинов, входящих в состав UI jQuery. С их помощью можно перемещать (Draggable) и "сбрасывать" (Droppable) любые элементы объектной модели документа (DOM), изменять их размеры (Resizable), делать элементы или группы элементов выбираемыми (Selectable), осуществлять сортировку элементов (Sortable). Все эти действия выполняются с помощью указателя мыши и без перезагрузки страницы.

При использовании любого плагина UI jQuery к странице, кроме самого файла библиотеки jQuery, нужно подключать некоторые дополнительные файлы. Все необходимое есть на странице настраиваемой закачки **http://jqueryui.com/download**, где можно совместить все требуемые файлы в одном. При желании, здесь же можно выбрать и готовые темы оформления. В этой главе мы рассмотрим различные темы оформления.

Во всех примерах этой главы мы будем использовать библиотеку jQuery версии 1.4.4 и библиотеку для построения пользовательских интерфейсов UI jQuery версии 1.8.9 — наиболее проверенную связку на момент написания книги.

18.1. Draggable — перемещение элементов

ЗАДАЧА

Необходимо сделать элементы DOM перемещаемыми по веб-странице с помощью указателя мыши.

Решение

Для решения этой задачи используем плагин Draggable и тему оформления "le-frog" (листинг 18.1.1).

Листинг 18.1.1. Использование плагина Draggable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-1-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #draggable { width: 150px; height: 150px; padding: 0.5em; }
</style>
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
});
</script>
</head>
<body>
<div id="draggable" class="ui-widget-content">
  <p>Drag me around</p>
</div>
<button id="getOption">Get Option</button>
<button id="setOption">Set Option</button>
</body>
</html>
```

Обсуждение

Итак, сначала мы подключили файл стилей css/le-frog/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме того, мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и плагинов Draggable, Droppable и Sortable (для примеров этой главы). За связь плагина с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 18.1.2.

Листинг 18.1.2. Использование плагина Draggable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
});
</script>
```

Таким нехитрым способом мы добились того, что теперь, нажимая и удерживая левую кнопку мыши на элементе с идентификатором `draggable`, мы имеем возможность совершенно свободно перемещать его по всей странице. В примере из листинга 18.1.2 мы применили плагин Draggable с настройками по умолчанию. Но этот плагин может принимать более 20 различных опций, которые помогают очень гибко настроить его под конкретные условия.

В листинге 18.1.3 мы попробуем передать плагину несколько настроек.

Листинг 18.1.3. Использование плагина Draggable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable({
    cursor: "move",
    grid: [20, 20],
    revert: true,
    revertDuration: 3000
  });
});
</script>
```

В примере из листинга 18.1.3, передав плагину новые значения для четырех опций, мы изменили вид курсора во время перемещения элемента, заставили элемент двигаться дискретно, по сетке с шагом в 20 пикселов по горизонтали и вертикали, и добились того, что по окончании перемещения элемент самостоятельно вернется в свое начальное положение в течение 3 с.

Остальные опции используются аналогично, а полный их список с описаниями приведен в табл. 18.1.

Таблица 18.1. *Опции плагина Draggable*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность Draggable будет недоступна, однако может быть включена впоследствии, например, при выполнении какого-либо условия
`addClasses`	Значение по умолчанию — `true`. Если установить значение `false`, то это предотвратит добавление класса `ui-draggable` к перемещаемым элементам. Это может потребоваться из соображений оптимизации при вызове `.draggable()`, например, для нескольких сотен элементов
`appendTo`	Значение по умолчанию — `parent`. В ней можно передать селектор jQuery или элемент, который будет контейнером для объекта, представляющего перемещаемый элемент во все время его перемещения
`axis`	Вынуждает перетаскиваемый элемент перемещаться только по оси *X* или только по оси *Y*. Значение по умолчанию — `false`, что позволяет элементу перемещаться свободно

Таблица 18.1 *(продолжение)*

Опция	Описание
`cancel`	Значение по умолчанию — `'input, option'`. Если эти элементы находятся внутри перемещаемого элемента, то его нельзя захватить, при нахождении указателя мыши над определенными в этой опции элементами. Для выбора элементов служит селектор jQuery
`connectToSortable`	Разрешает "сброс" перемещаемых элементов в сортируемый (sortable) список. При этом "сброшенный" элемент становится частью списка. В качестве значения указывается селектор jQuery. Значение по умолчанию — `false`. Для корректной работы необходимо дополнительная опция `helper` со значением `'clone'`
`containment`	Ограничивает перемещение внутри определенного элемента или области. В качестве значения может принимать селектор jQuery, элемент, строку или массив. Примеры возможных значений: `'parent'`, `'document'`, `'window'`, `[x1, y1, x2, y2]`
`cursor`	Строка, в которой может быть передано CSS-значение, определяющее значок для курсора в процессе перетаскивания элемента. По умолчанию — `auto`
`cursorAt`	При перемещении элемента устанавливает курсор в определенную позицию. Например, значение `{left: 5, top: 10}` при перемещении установит курсор в 10 пикселах от верхнего и в 5 пикселах от левого краев
`delay`	Отсрочка начала времени перемещения элемента в миллисекундах. Опция необходима для того, чтобы предотвратить нежелательное перемещение элемента во время случайного щелчка по нему. Значение по умолчанию — 0
`distance`	Расстояние в пикселах, на которое должен переместиться курсор при удерживаемой в нажатом положении левой кнопки мыши прежде, чем начнется перемещение элемента. Опция служит для того, чтобы предотвратить нежелательное перемещение элемента во время случайного щелчка по нему. Значение по умолчанию — 1
`grid`	Заставляет перетаскиваемый элемент перемещаться дискретно, по сетке, определяемой массивом передаваемых в опцию значений. Значение по умолчанию — `false`, что позволяет элементу перемещаться свободно. Пример передаваемого значения: `[10, 10]`
`handle`	В этой опции можно определить некоторый элемент, расположенный внутри перемещаемого элемента, только при нахождении над которым указателя мыши будет возможен захват элемента для перемещения. Для выбора элементов можно использовать селектор jQuery
`helper`	Возможные значения опции — `'original'`, `'clone'`, `Function`. Значение по умолчанию — `'original'` (визуально наблюдается перемещение самого элемента). При значении `'clone'` перемещаться будет клон элемента, а сам элемент при этом остается на своей позиции. В опции можно определить функцию. Функция обязательно должна возвращать элемент DOM
`iframeFix`	Значение по умолчанию — `false`. Если установить значение `true`, будет предотвращен перехват события `mousemove` во время перемещения элемента при прохождении указателя мыши над `iframe`. Значением может быть селектор jQuery — при этом предотвращение перехвата будет производиться только при прохождении указателя мыши над выбранными `iframe`. Опция бывает полезна при использовании совместно, например, с `cursorAt`

Таблица 18.1 *(продолжение)*

Опция	Описание
`opacity`	Устанавливает значение CSS-свойства `opacity` для элемента, находящегося в процессе перемещения. Значение по умолчанию — `false` (соответствует полной непрозрачности)
`refreshPositions`	Значение по умолчанию — `false`. При значении `true` позиция "сброса" элемента будет пересчитываться при каждом событии `mousemove`. Использовать с осторожностью! Такой подход может помочь создать очень динамичное приложение, но, безусловно, снизит производительность
`revert`	Значение по умолчанию — `false`. Если установить значение `true`, то элемент возвратится на свою исходную позицию после окончания перетаскивания. Также можно передать значения в виде строки. Возможные значения — `'valid'` или `'invalid'`. Эти значения необходимы в случае, если перемещаемый элемент будет "сброшен" в какой-либо целевой элемент. При указании значения `'valid'` перемещаемый элемент возвратится в предыдущую позицию только в том случае, если был "сброшен" в целевой элемент. При значении `'invalid'` — не возвратится в предыдущую позицию только в том случае, если он был "сброшен" в целевой элемент
`revertDuration`	Длительность эффекта возвращения элемента на исходную позицию, в миллисекундах. Значение по умолчанию — 500. Опция игнорируется, если опция `revert` имеет значение `false`
`scope`	Значением этой опции является строка. По умолчанию — `'default'`. Перемещаемые элементы, имеющие в опции `scope` точно такое же значение, как и значение опции `scope` для элементов с функциональностью `droppable`, будут приниматься этими элементами. По сути, дополняет опцию `accept` плагина Droppable (см. *разд. 18.2*)
`scroll`	Значение по умолчанию — `true` — при перемещении элемента к краю области просмотра, она автоматически прокручивается бесконечно. Если установить значение `false`, эта возможность будет запрещена
`scrollSensitivity`	Число, определяющее расстояние в пикселах от края области просмотра, после которого она начинает прокручиваться. Расстояние определяется относительно указателя мыши, а не перемещаемого элемента. Значение по умолчанию — 20
`scrollSpeed`	Число, определяющее скорость, с которой прокручивается область просмотра при приближении указателя мыши к ее краю на расстояние, определенное в опции `scrollSensitivity`. Значение по умолчанию — 20
`snap`	Значение по умолчанию — `false`. Если установить значение `true` (что эквивалентно `'.ui-draggable'`, поскольку значение может определяться также селектором jQuery), то перемещаемый элемент будет "прилипать" к краям выбранных элементов при прохождении около них
`snapMode`	Опция определяет, как именно перемещаемый элемент будет "прилипать" к выбранным элементам. Например, только к внешним или только к внутренним сторонам элементов. Возможные значения — `'inner'`, `'outer'`, `'both'`. Любое значение игнорируется при `snap: false`. Значение по умолчанию — `'both'`

Таблица 18.1 *(окончание)*

Опция	Описание
snapTolerance	Расстояние в пикселах от перемещаемого элемента до выбранного, при котором произойдет "прилипание". Значение по умолчанию — 20. Любое значение игнорируется при `snap: false`
stack	Объект, с помощью которого осуществляется автоматический контроль за свойством `z-index` определенной группы перемещаемых элементов, всегда помещая перемещаемый элемент поверх остальных. Значение свойства `'group'` определяет селектор jQuery. Дополнительно можно определить свойство `'min'`, число, ниже которого значение `z-index` опуститься не сможет. Например: `$('.selector').draggable({ stack: { group: '.selector', min: 50 } });`
zIndex	Число, определяющее значение свойства `z-index`, перемещаемого элемента. По умолчанию — `false`

Кроме гибкой настройки плагина Draggable, есть возможность заставить его реагировать на некоторые события. Листинг 18.1.4 иллюстрирует пример одного из возможных событий плагина.

Листинг 18.1.4. Использование плагина Draggable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable({
    cursor: "move",
    stop: function(event, ui) {
      alert("Событие: " + event.type +
            "\n\ntop: " + ui.offset.top +
            " px\nleft: " + ui.offset.left +
            " px");
    }
  });
});
</script>
```

В примере, приведенном в листинге 18.1.4, мы определили в опции `stop` callback-функцию. Функция принимает два аргумента — объекты `event` и `ui`. На простом примере показано, как можно обратиться к свойствам этих объектов. В итоге, по окончании перемещения элемента в окне предупреждения мы увидим информацию о типе наступившего события — `dragstop` — и текущее абсолютное положение элемента на странице.

Любая callback-функция, определенная в опциях `start`, `drag` или `stop`, принимает два аргумента — `event` и `ui`. Объект `event` — это объект события, а `ui` — специальный объект, в свойствах которого содержится следующая информация:

- `ui.helper` — объект, характеризующий элемент, находящийся в процессе перемещения;

- `ui.position` — объект, в свойствах `top` и `left` которого содержатся сведения о положении перемещаемого элемента относительно родительского элемента;
- `ui.offset` — объект, в свойствах `top` и `left` которого содержатся данные об абсолютном положении перемещаемого элемента.

В табл. 18.2 приведены описания всех возможных событий плагина Draggable и названия опций, в которых могут быть определены callback-функции.

Таблица 18.2. *События плагина Draggable*

Опция	Событие	Описание
`create`	`dragcreate`	Наступает в момент инициализации
`start`	`dragstart`	Наступает каждый раз при начале перемещения элемента
`drag`	`drag`	Происходит все время, пока элемент находится в процессе перемещения
`stop`	`dragstop`	Наступает каждый раз при завершении перемещения элемента

Плагин Draggable имеет четыре метода, с помощью которых можно управлять им уже после инициализации. Познакомимся с одним из методов на примере, приведенном в листинге 18.1.5. Здесь нам пригодятся две кнопки, которые мы до этого момента не замечали.

Листинг 18.1.5. Использование плагина Draggable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable({
    cursor: "move"
  });
  $("#getOption").click(function(){
    alert($("#draggable").draggable("option", "grid"));
  });
  $("#setOption").click(function(){
    $("#draggable").draggable("option", "grid", [50,50]);
  });
});
</script>
```

В примере из листинга 18.1.5 мы связали с кнопками **Get Option** и **Set Option** событие `click`. При щелчке на кнопке **Get Option** мы вызовем метод `option`, имя которого указываем в первом аргументе, а во втором аргументе передаем название опции, значение которой мы хотели бы получить. В итоге мы увидим окно предупреждения, где будет показано значение `false`, поскольку при инициализации эта опция не была определена, а по умолчанию она имеет именно такое значение. По-

пробуйте перетаскивать элемент с помощью мыши — он будет плавно перемещаться в любых направлениях.

Сделаем щелчок на кнопке **Set Option**, снова вызывая метод `option`, но теперь мы передаем три аргумента. Первый — название метода, второй — название опции, значение которой мы хотим изменить, и третий аргумент — новое значение опции. Таким образом, передав опции `grid` значение `[50,50]`, мы заставим элемент перемещаться дискретно, вдоль сетки с шагом 50×50 пикселов.

Описания всех методов плагина Draggable приведены в табл. 18.3.

Таблица 18.3. *Методы плагина Draggable*

Метод	Описание
destroy `.draggable('destroy')`	Полностью удаляет всю функциональность плагина Draggable. Возвращает элементы в состояние, предшествующее инициализации
disable `.draggable('disable')`	Временно запрещает использование всей функциональности плагина. Вновь разрешить ее можно с помощью метода `enable`
enable `.draggable('enable')`	Разрешает использование всей функциональности плагина, если ранее она была запрещена методом `disable`
option `.draggable('option', optionName, [value])`	С помощью этого метода можно установить значение любой опции плагина после инициализации
option **`.draggable('option', optionName)`**	С помощью этого метода можно получить значение любой опции плагина после инициализации
widget **`.draggable('widget')`**	С помощью этого метода можно получить доступ к объекту, который представляет собой перемещаемый элемент

Рассмотрим еще один интересный пример, демонстрирующий возможность "сброса" перемещаемых элементов в сортируемый список (листинг 18.1.6).

Листинг 18.1.6. Использование плагина Draggable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-1-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
```

```
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
.draggable {
  width:200px; margin:5px; padding-left:10px;
  border:1px solid #369; background-color:#eee;
  cursor:move;
}
#sortable li {
  width:200px; margin:5px; padding-left:10px;
  border:1px solid #000; background-color:#333;
  color:#fff; cursor:move;
}
</style>
<script type="text/javascript">
$(function(){
  $("#sortable").sortable();
  $(".draggable").draggable({
    connectToSortable: "#sortable",
    helper: "clone"
  });
});
</script>
</head>
<body>
<ul>
  <li class="draggable">List 1, item 1</li>
  <li class="draggable">List 1, item 2</li>
  <li class="draggable">List 1, item 3</li>
</ul>
<br />
<ul id="sortable">
  <li>List 2, item 1</li>
  <li>List 2, item 2</li>
  <li>List 2, item 3</li>
  <li>List 2, item 4</li>
  <li>List 2, item 5</li>
</ul>
</body>
</html>
```

HTML-разметка, приведенная в листинге 18.1.6, — это два ненумерованных списка. Элементы `li` первого списка имеют атрибут `class` со значением `draggable`, с помощью JavaScript-кода мы сделаем их перемещаемыми. Второй список имеет идентификатор `#sortable`, и опять же с помощью JavaScript-кода получит возможность сортировки элементов `li` в пределах списка. Кроме того, появится возможность перемещать и "сбрасывать" элементы `li` из первого списка во второй.

Посмотрим теперь, как такие возможности реализуются с помощью JavaScript.

Сделать второй список сортируемым не составляет труда — выбираем список по его идентификатору `#sortable`, указанному в селекторе, и применяем к нему метод `sortable`, не определяя никаких опций.

Займемся первым списком. Сначала нужно сделать все его элементы перемещаемыми. Это несложно — выбираем нужные элементы по названию класса, указанному в селекторе, и применяем метод `draggable`.

Для того чтобы реализовать перемещение и "сброс" элементов первого списка во второй, нужно определить опцию `connectToSortable`, указав в качестве значения селектор jQuery. В данном случае это `#sortable`, т. е. идентификатор сортируемого списка. Необходимо отметить, что для корректной работы следует определить еще и опцию `helper`, указав в ней значение `clone`.

Это все. Теперь элементы первого списка можно перемещать и "сбрасывать" во второй список.

18.2. Droppable — "сброс" элементов

ЗАДАЧА

Необходимо сделать элемент или элементы DOM не только перемещаемыми, но и "сбрасываемыми" в другой определенный элемент на веб-странице с помощью указателя мыши.

Решение

Используем плагин Droppable для решения этой задачи (листинг 18.2.1). Выберем тему оформления "trontmastic".

Листинг 18.2.1. Использование плагина Droppable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-2-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/trontastic/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #draggable { width: 100px; height: 100px; padding: 0.5em; float: left;
margin: 10px 10px 10px 0; }
  #droppable { width: 150px; height: 150px; padding: 0.5em; float: left;
margin: 10px; overflow:auto; }
```

```
  .active { border: 3px solid #f00; }
  .hover { border: 3px solid #00f; }
</style>
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
  $("#droppable").droppable();
});
</script>
</head>
<body>
<div id="draggable" class="ui-widget-content">
  <p>Drag me to my target</p>
</div>
<div id="droppable" class="ui-widget-header">
  <p>Drop here</p>
</div>
<button id="disable">Disable</button>
<button id="enable">Enable</button>
<button id="destroy">Destroy</button>
</body>
</html>
```

Обсуждение

Итак, сначала мы подключили файл стилей css/trontastic/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и плагинов Droppable и Draggable (ведь прежде, чем "сбросить" элемент, его нужно переместить). За связь плагина с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 18.2.2.

Листинг 18.2.2. Использование плагина Droppable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
  $("#droppable").droppable({
    drop: function(){
      $("#result").append("Dropped!<br />");
    }
  });
});
</script>
```

В примере из листинга 18.2.2 мы сразу же определили callback-функцию в опции `drop`. Это сделано в первую очередь для наглядности. Как только перемещаемый

блок `div` будет "сброшен" в целевой блок `div`, в элемент с идентификатором `result` добавится соответствующий текст.

Попробуем передать плагину еще пару опций, чтобы на примере понять, как это делается (листинг 18.2.3).

Листинг 18.2.3. Использование плагина Droppable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
  $("#droppable").droppable({
    activeClass: "active",
    hoverClass: "hover",
    drop: function(){
      $("#result").append("Dropped!<br />");
    }
  });
});
</script>
```

В примере из листинга 18.2.3 мы передали плагину Droppable опции `activeClass` и `hoverClass`, в которых указали имена классов. Посмотрите, как они определены в листинге 18.2.1. Таким образом, мы добились того, чтобы все время в процессе движения перемещаемого элемента у целевого элемента присутствовала красная рамка, а в тот момент, когда перемещаемый элемент оказывается над целевым, рамка меняла свой цвет на синий.

Познакомьтесь с описаниями возможных опций плагина Droppable (табл. 18.4).

Таблица 18.4. *Опции плагина Droppable*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность Droppable будет недоступна, однако может быть включена впоследствии, например, при выполнении какого-либо условия
`accept`	В опции может быть указан селектор или функция. Все перемещаемые элементы, которые будут выбраны этим селектором, получат возможность быть "сброшенными" в элемент, к которому применена функциональность плагина Droppable. Если в этой опции определена функция, то она будет вызываться для каждого перемещаемого элемента на странице (передаваемого в функцию как первый аргумент), с целью применения пользовательского фильтра. Функция должна будет возвратить `true`, если элемент должен получить возможность быть "сброшенным". Значение по умолчанию — `*`
`activeClass`	Если в этой опции определить имя класса, то он будет добавлен к целевому элементу в момент начала перемещения элемента, который должен быть "сброшен". По умолчанию — `false`

Таблица 18.4 *(окончание)*

Опция	Описание
addClasses	Значение по умолчанию — true. Если установить false, то это предотвратит добавление класса с именем ui-droppable. Это может потребоваться в целях оптимизации при вызове .droppable() для нескольких сотен элементов
greedy	Когда для этой опции установлено значение true, это предотвращает всплывание события, т. е. если "сброс" был выполнен во внутренний элемент, в то время как его родитель тоже может принимать "сбрасываемый" элемент, события drop для элемента-родителя не произойдет. Значение по умолчанию — false
hoverClass	Если в этой опции определить имя класса, то он будет добавлен к целевому элементу в тот момент, когда перемещаемый элемент окажется над целевым элементом. По умолчанию — false
scope	Опция применяется в дополнение к опции accept. Значение по умолчанию — 'default'. Может быть определено любое значение в виде строки и все перемещаемые элементы (т. е. элементы с функциональностью draggable), с таким же значением соответствующей опции, будут иметь возможность "сброса" в целевой элемент
tolerance	Опция определяет, в какой момент необходимо выполнить "сброс" элемента, при нахождении его над целевым элементом. По умолчанию имеет значение 'intersect', что определит "сброс" в тот момент, когда перемещаемый элемент перекрыл целевой как минимум на 50%. Другие возможные значения: • fit — перемещаемый элемент должен полностью войти в целевой элемент; • pointer — указатель мыши должен войти в целевой элемент; • touch — перемещаемый элемент должен перекрыть целевой элемент на любую самую малую величину

Как и другие плагины, входящие в состав UI jQuery, плагин Droppable умеет реагировать на некоторые события. Опций, в которых можно определить callback-функции, вызываемые при наступлении соответствующего события, всего шесть (листинг 18.2.4).

Листинг 18.2.4. Использование плагина Droppable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
  $("#droppable").droppable({
    create: function(event, ui){
      $("#result").empty().append("Событие: " + event.type + "<br />");
    },
    activate: function(event, ui){
      $("#result").append("Событие: " + event.type + "<br />");
    },
```

```
    deactivate: function(event, ui){
      $("#result").append("Событие: " + event.type + "<br />");
    },
    over: function(event, ui){
      $("#result").append("Событие: " + event.type +
      " -> ui.position.top/left: " + ui.position.top +
      "/" + ui.position.left + " px & ui.offset.top/left: " +
      ui.offset.top + "/" + ui.offset.left + " px<br />");
    },
    out: function(event, ui){
      $("#result").append("Событие: " + event.type +
      " -> ui.position.top/left: " + ui.position.top +
      "/" + ui.position.left + " px & ui.offset.top/left: " +
      ui.offset.top + "/" + ui.offset.left + " px<br />");
    },
    drop: function(event, ui){
      $("#result").append("Событие: " + event.type +
      " -> ui.helper.attr('id'): " + ui.helper.attr('id') +
      ", ui.draggable.text(): " + ui.draggable.text() + "<br />");
    }
  });
});
</script>
```

Названия и описания событий для плагина Droppable приведены в табл. 18.5.

Таблица 18.5. *События плагина Droppable*

Опция	Событие	Описание
create	dropcreate	Наступает в момент инициализации
activate	dropactivate	Происходит в момент начала движения перемещаемого элемента
deactivate	dropdeactivate	Происходит в момент окончания движения перемещаемого элемента, независимо от того, был ли он "сброшен" в целевой элемент или нет
over	dropover	Происходит в момент входа перемещаемого элемента в пределы целевого элемента
out	dropout	Происходит в момент выхода перемещаемого элемента за пределы целевого элемента
drop	drop	Происходит в момент "сброса" перемещаемого элемента в целевой элемент

Нужно отметить, что любая callback-функция, вызываемая при наступлении какого-либо события, может принимать два аргумента. Первый аргумент — объект собы-

тия, второй — специальный объект ui, в свойствах которого содержится следующая информация:

- ui.draggable — объект, характеризующий перемещаемый элемент;
- ui.helper — объект, характеризующий перемещаемый хелпер;
- ui.position — объект, в свойствах top и left которого содержатся сведения о положении перемещаемого элемента относительно родительского элемента;
- ui.offset — объект, в свойствах top и left которого содержатся данные об абсолютном положении перемещаемого элемента.

Объект, представленный свойством helper объекта ui, обычно является клоном объекта, характеризующего перемещаемый элемент, т. е. ui.draggable.

Давайте еще раз посмотрим JavaScript-код из листинга 18.2.4. Callback-функции, определенные во всех шести опциях, используют свойство type объекта event для того, чтобы вывести название события, вызвавшего соответствующую функцию. Кроме того, в callback-функциях, определенных в опциях over и out, мы задействуем свойства объекта ui — position и offset. Поскольку эти свойства в свою очередь тоже являются объектами, мы обращаемся уже к их свойствам, top и left, и получаем таким образом положение перемещаемого элемента в момент наступления соответствующих событий. А в опции drop callback-функция работает с объектом ui.helper, получая значение идентификатора и текст перемещаемого элемента.

Плагин Droppable имеет четыре метода, с помощью которых можно управлять им уже после инициализации. Познакомимся с некоторыми из методов на примере, приведенном в листинге 18.2.5.

Листинг 18.2.5. Использование плагина Droppable

```
<script type="text/javascript">
$(function(){
  $("#draggable").draggable();
  $("#droppable").droppable({
    activate: function(event, ui){
      $("#result").empty().append(event.type + "<br />");
    },
    deactivate: function(event, ui){
      $("#result").append(event.type + "<br />");
    },
    over: function(event, ui){
      $("#result").append(event.type + "<br />");
    },
    out: function(event, ui){
      $("#result").append(event.type + "<br />");
    },
```

```
    drop: function(event, ui){
      $("#result").append(event.type + "<br />");
    }
  });
  $("#disable").click(function(){
    $("#droppable").droppable("disable");
  });
  $("#enable").click(function(){
    $("#droppable").droppable("enable");
  });
  $("#destroy").click(function(){
    $("#droppable").droppable("destroy");
  });
});
</script>
```

С кнопками, имеющими идентификаторы `#disable`, `#enable` и `#destroy`, связаны обработчики события `click`, которые вызывают одноименные методы плагина.

Используя `.droppable("disable")`, можно временно удалить функциональность плагина из выбранного элемента, а с помощью `.droppable("enable")` — восстановить ее. Указание `.droppable("destroy")` приведет к окончательному удалению функциональности. Восстановить ее можно будет только перезагрузкой страницы.

Методы плагина Droppable приведены в табл. 18.6.

Таблица 18.6. *Методы плагина Droppable*

Метод	Описание
destroy `.droppable('destroy')`	Полностью удаляет всю функциональность плагина Droppable. Возвращает элементы в состояние, предшествующее инициализации
disable `.droppable('disable')`	Временно запрещает использование всей функциональности плагина. Вновь разрешить ее можно с помощью метода `enable`
enable `.droppable('enable')`	Разрешает использование всей функциональности плагина, если ранее она была запрещена методом `disable`
option `.droppable('option', optionName, [value])`	Устанавливает значение любой опции плагина после инициализации
option **`.droppable('option', optionName)`**	С помощью этого метода можно получить значение любой опции плагина после инициализации
widget **`.droppable('widget')`**	Обеспечивает доступ к объекту, который представляет собой элемент с возможностью приема "сбрасываемых" в него элементов

18.3. Resizable — изменение размеров элементов

ЗАДАЧА

Необходимо добавить возможность изменять размеры элемента (или элементов) DOM с помощью указателя мыши.

Решение

Решим эту задачу с помощью плагина Resizable (листинг 18.3.1). Выберем тему оформления "start".

Листинг 18.3.1. Использование плагина Resizable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-3-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/start/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #resizable { width: 150px; height: 150px; padding: 0.5em; }
  #resizable h3 { text-align: center; margin: 0; }
  .ui-resizable-helper { border: 1px dotted #999; }
</style>
<script type="text/javascript">
$(function(){
  $("#resizable").resizable();
});
</script>
</head>
<body>
<div id="resizable" class="ui-widget-content">
  <h3 class="ui-widget-header">Resizable</h3>
</div>
<!--
<button id="getter">Get Option</button>
<button id="setter">Set Option</button>
-->
</body>
</html>
```

Обсуждение

Сначала подключаем файл стилей css/start/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Вероятно, у вас будет свое собственное оформление, но для изучения примеров лучше воспользоваться готовым решением. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и плагина Resizable. За связь плагина с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 18.3.2.

Листинг 18.3.2. Использование плагина Resizable

```
<script type="text/javascript">
$(function(){
  $("#resizable").resizable();
});
</script>
```

В примере из листинга 18.3.2 мы указали в селекторе jQuery идентификатор того элемента, размеры которого будем изменять, и применили к нему функциональность плагина. Все возможные настройки при этом будут установлены по умолчанию.

Попробуем передать плагину несколько опций, чтобы на примере понять, как это делается и насколько гибко можно его настраивать (листинг 18.3.3).

Листинг 18.3.3. Использование плагина Resizable

```
<script type="text/javascript">
$(function(){
  $("#resizable").resizable({
    animate: true,
    ghost: true
  });
});
</script>
```

В примере из листинга 18.3.3 мы передали плагину две опции — `animate` и `ghost`. В обоих случаях передали логическое значение `true`. Но, чего же мы этим смогли добиться?

Передав значение `true` в опции `animate`, мы добились того, что теперь при изменении размеров элемента будет реализован анимационный эффект.

В случае с `ghost: true` при изменении размеров элемента будут отображаться его полупрозрачные контуры.

В табл. 18.7 приведены все доступные опции плагина Resizable и их описания.

Таблица 18.7. *Опции плагина Resizable*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность Resizable будет недоступна, однако может быть включена впоследствии, например, при выполнении какого-либо условия
`alsoResize`	Значение по умолчанию — `false`. Может принимать селектор jQuery, объект jQuery или элемент. Элементы, определенные в этой опции, также меняют свои размеры
`animate`	Значение по умолчанию — `false`. Если установить значение `true`, то изменение размера будет сопровождаться анимацией
`animateDuration`	В качестве значения опция принимает строку с предопределенными значениями — `'slow'`, `'normal'`, `'fast'` или число в миллисекундах, определяющее длительность анимационного эффекта при изменении размера элемента
`animateEasing`	Опция принимает строку с названием анимационного эффекта. Большое количество разнообразных эффектов доступно при подключении дополнительного плагина jQuery Easing. Значение по умолчанию — `'swing'`
`aspectRatio`	Значение по умолчанию — `false`. Если установить `true`, то при изменении размеров элемента будет сохраняться пропорция в соотношении сторон. Можно также передавать значения в виде 16/9 или 0,75. В этом случае пропорции в соотношении сторон будут установлены в соответствии с указанным значением
`autoHide`	Значение по умолчанию — `false`. Если установить значение `true`, картинка, расположенная в правом нижнем углу элемента, будет показываться только в случае нахождения указателя мыши над элементом
`cancel`	В качестве значения указывается селектор jQuery. Предотвращает изменение размеров для элементов, указанных в этой опции. Значение по умолчанию — `':input,option'`
`containment`	Ограничивает изменение размеров внутри элемента, определенного в этой опции. Значением может быть строка (например `'parent'`, `'document'`), элемент DOM или селектор jQuery. Значение по умолчанию — `false`
`delay`	В этой опции можно указать число в миллисекундах, определив задержку старта изменения размеров элемента. Значение по умолчанию — 0. Эта опция помогает предотвратить случайное изменение размеров элемента при щелчке мышью
`distance`	Значение по умолчанию — 1. Опция определяет число пикселов, на которое переместится указатель мыши, прежде чем начнется изменение размеров элемента. Эта опция помогает предотвратить случайное изменение размеров элемента при щелчке мышью
`ghost`	Значение по умолчанию — `false`. Если установить `true`, то при изменении размеров элемента будут видны его полупрозрачные контуры
`grid`	В качестве значения принимает массив элементов `[x, y]`, где `x` и `y` — числа, определяющие шаг сетки. Изменение размеров будет происходить дискретно, в соответствии с заданными значениями. Значение по умолчанию — `false`

Таблица 18.7 *(окончание)*

Опция	Описание
handles	Принимает строку в качестве значения. По умолчанию установлено значение `'e, s, se'`. Допустимы любые из следующих обозначений: `n`, `e`, `s`, `w`, `ne`, `se`, `sw`, `nw`, `all` или их комбинации
helper	Значение по умолчанию — `false`. Принимает строку с названием CSS-класса, который добавляется к области, демонстрирующей во время перемещения размеры, в которые установится элемент
maxHeight	Число, определяющее максимальную высоту, до которой можно изменять размеры элемента. По умолчанию — `null`
maxWidth	Число, определяющее максимальную ширину, до которой можно изменять размеры элемента. По умолчанию — `null`
minHeight	Число, определяющее минимальную высоту, до которой можно изменять размеры элемента. По умолчанию — 10
minWidth	Число, определяющее минимальную ширину, до которой можно изменять размеры элемента. По умолчанию — 10

Resizable, как и остальные плагины из UI jQuery, может реагировать на события. Он имеет четыре опции, в которых можно определить callback-функции, вызываемые при наступлении соответствующего события. Названия опций и описания соответствующих событий для плагина Resizable приведены в табл. 18.8. Пример использования опции `stop` иллюстрирует листинг 18.3.4.

Таблица 18.8. *События плагина Resizable*

Опция	Событие	Описание
create	resizecreate	Наступает в момент инициализации
start	resizestart	Наступает при начале изменения размеров элемента
resize	resize	Происходит постоянно во время изменения размеров элемента
stop	resizestop	Наступает по окончании изменения размеров элемента

Листинг 18.3.4. Использование плагина Resizable

```
<script type="text/javascript">
$(function(){
  $("#resizable").resizable({
    ghost: true,
    stop: function(event, ui) {
      alert("Событие " + event.type);
    }
  });
});
</script>
```

В примере из листинга 18.3.4 мы определили в опции `stop` callback-функцию, которая будет вызвана в момент окончания изменения размеров элемента. Функция может принимать два аргумента. Первый аргумент — это объект события, к свойству `type` которого можно легко обратиться. Теперь по окончании изменения размеров выбранного элемента мы получим окно предупреждения, где будет выведено название события, вызвавшего эту функцию.

Что касается второго аргумента, то это объект, в свойствах которого содержится весьма полезная информация. В примере из листинга 18.3.5 показано, как можно обратиться к некоторым свойствам этого объекта.

Листинг 18.3.5. Использование плагина Resizable

```
<script type="text/javascript">
$(function(){
  $("#resizable").resizable({
    ghost: true,
    stop: function(event, ui) {
      alert("Начальные размеры: " + ui.originalSize.width +
            "x" + ui.originalSize.height +
            "px\nКонечные размеры: " + ui.size.width +
            "x" + ui.size.height + "px");
    }
  });
});
</script>
```

В примере из листинга 18.3.5 мы обращаемся к свойствам объекта `ui originalSize` и `size`. Оба свойства также представляют собой объекты, в свойствах `width` и `height` которых сохраняются значения ширины и высоты начальных размеров элемента (`originalSize`) и его конечных размеров (`size`).

Перечислим основные свойства специального объекта `ui`:

- `ui.helper` — объект, представляющий изменяемый элемент;
- `ui.size` — объект, в свойствах `width` и `height` которого сохраняются текущие ширина и высота элемента;
- `ui.originalSize` — объект, в свойствах `width` и `height` которого сохраняются ширина и высота элемента до изменения его размеров;
- `ui.position` — объект, в свойствах `top` и `left` которого сохраняется текущее положение элемента;
- `ui.originalPosition` — объект, в свойствах `top` и `left` которого сохраняется положение элемента до изменения его размеров.

Кроме перечисленных возможностей, плагин имеет несколько методов, с помощью которых можно управлять им уже после инициализации. Пример использования метода `.resizable('option', optionName, [value])` приведен в листинге 18.3.6.

Листинг 18.3.6. Использование плагина Resizable

```
<script type="text/javascript">
$(function(){
  $("#resizable").resizable({
    animate: true,
    ghost: true,
    animateDuration: 500
  });
  $("#getter").click(function(){
    alert($("#resizable").resizable("option", "animateDuration"));
  });
  $("#setter").click(function(){
    $("#resizable").resizable("option", "animateDuration", 3000);
  });
});
</script>
```

В листинге 18.3.6 мы создаем кнопки с идентификаторами `#getter` и `#setter`, с которыми свяжем событие `click`. При щелчке мышью на кнопке `#getter` будем вызывать метод `.resizable('option', optionName, [value])` с двумя аргументами, получая, таким образом, текущее значение опции `animateDuration`. А при щелчке мышью на кнопке `#setter` вызовем тот же самый метод `.resizable('option', optionName, [value])`, но уже с тремя аргументами. В качестве третьего аргумента передадим значение, которое мы хотим установить для опции `animateDuration`.

В табл. 18.9 приведены описания всех доступных методов плагина Resizable.

Таблица 18.9. *Методы плагина Resizable*

Метод	Описание
destroy `.resizable('destroy')`	Полностью удаляет всю функциональность плагина Resizable. Возвращает элементы в состояние, предшествующее инициализации
disable `.resizable('disable')`	Временно запрещает использование всей функциональности плагина. Вновь разрешить ее можно с помощью метода `enable`
enable `.resizable('enable')`	Разрешает использование всей функциональности плагина, если ранее она была запрещена методом `disable`
option `.resizable('option', optionName, [value])`	Устанавливает значение любой опции плагина после инициализации
option **`.resizable('option', optionName)`**	С помощью этого метода можно получить значение любой опции плагина после инициализации
widget **`.resizable('widget')`**	Обеспечивает доступ к объекту, который представляет собой элемент с возможностью изменения размеров

18.4. Selectable — выбор элементов

ЗАДАЧА

Необходимо добавить на веб-страницу возможность выбора элементов с помощью передвижения указателя мыши с удержанием в нажатом состоянии ее левой кнопки или комбинации щелчка мышью с нажатием и удержанием клавиши <Ctrl> (<Meta>), подобно тому, как это делается в традиционных приложениях.

Решение

Применим плагин Selectable вместе с темой оформления "le-frog" для решения этой задачи (листинг 18.4.1).

Листинг 18.4.1. Использование плагина Selectable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-4-1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; height: 18px;
}
</style>
<script type="text/javascript">
$(function(){
  $("#selectable").selectable();
});
</script>
</head>
<body>
<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
```

```
  <li class="ui-widget-content">Item 6</li>
  <li class="ui-widget-content">Item 7</li>
</ol>
</body>
</html>
```

Обсуждение

Сначала подключаем файл стилей css/le-frog/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Вполне возможно, что у вас будет свое собственное оформление, но для изучения примеров лучше воспользоваться готовым примером. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, объединяющий функциональность ядра UI и плагина Selectable. Для примера создадим список `ol`, пункты которого нам предстоит сделать выделяемыми. За связь плагина с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 18.4.2.

Листинг 18.4.2. Использование плагина Selectable

```
<script type="text/javascript">
$(function(){
  $("#selectable").selectable();
});
</script>
```

В примере из листинга 18.4.2 мы выбрали нужный нам элемент по его идентификатору `#selectable` и связали с ним возможности плагина. В этом примере мы оставили настройки по умолчанию.

Однако мы имеем возможность передавать плагину при инициализации некоторые настройки. Как это можно делать, показано в листинге 18.4.3.

Листинг 18.4.3. Использование плагина Selectable

```
<script type="text/javascript">
$(function(){
  $("#selectable").selectable({
    distance: 100
  });
});
</script>
```

В примере из листинга 18.4.3 мы передали плагину значение `100` в опции `distance` и этим определили расстояние в пикселах, которое должен пройти указатель мыши (с одновременно нажатой левой кнопкой), прежде, чем начнется выделение указанных элементов. Значения для остальных опций можно передать плагину подобным образом.

В табл. 18.10 приведено описание всех возможных опций плагина Selectable.

Таблица 18.10. *Опции плагина Selectable*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность Selectable будет недоступна, однако может быть включена впоследствии, например, при выполнении какого-либо условия
`autoRefresh`	Значение по умолчанию — `true`, что определяет пересчет положения и размеров области выделения при попадании туда очередного элемента. Если таких элементов слишком много, может понадобиться отключить пересчет. Для этого следует установить значение `false`. При необходимости можно будет использовать для пересчета метод `.selectable('refresh')`
`cancel`	Опция принимает в качестве значения селектор jQuery. Предотвращает начало выбора элементов, если при начале выбор указателя мыши находится над указанными элементами. Значение по умолчанию — `':input,option'`
`delay`	В опции указывается число, устанавливающее время в миллисекундах, определяющее задержку начала выбора элементов. По умолчанию — 0
`distance`	В опции указывается число, устанавливающее расстояние в пикселах, которое должен пройти указатель мыши до начала выбора элементов. По умолчанию — 0
`filter`	В качестве значения опции указывается селектор jQuery. Значение по умолчанию — `*`. В этой опции можно указывать элементы-потомки, которые будут сделаны выбранными (при условии, если они могут быть таковыми)
`tolerance`	Опция принимает в качестве значения строку. Значение по умолчанию — `touch`. Возможные значения `'touch'` и `'fit'`. При указании значения `'touch'` выбор элемента производится, когда область выбора перекрывает элемент какой-либо частью. При значении `'fit'` выбор элемента производится, только когда область выбора перекрывает элемент полностью

Посмотрим теперь, на какие события может реагировать плагин Selectable. Таких событий всего семь. Названия событий и их описания приведены в табл. 18.11.

Таблица 18.11. *События плагина Selectable*

Опция	Событие	Описание
`create`	`selectablecreate`	Наступает в момент инициализации
`selected`	`selected`	Наступает по окончании выбора для каждого выбранного элемента
`selecting`	`selecting`	Происходит в процессе выбора для каждого выбранного элемента
`start`	`selectablestart`	Наступает при начале процесса выбора
`stop`	`selectablestop`	Наступает по окончании процесса выбора

Таблица 18.11 *(окончание)*

Опция	Событие	Описание
unselected	unselected	Наступает по окончании выбора для каждого элемента, не попавшего в выбор по сравнению с предыдущим состоянием
unselecting	unselecting	Происходит в процессе выбора, когда какой-либо элемент, присутствовавший в выборе, удаляется из него

В листинге 18.4.4 приведен пример события, которое происходит по окончании выбора элементов. Callback-функцию, вызываемую при наступлении этого события, можно определить в опции `stop`.

Листинг 18.4.4. Использование плагина Selectable

```
<script type="text/javascript">
$(function(){
  $("#selectable").selectable({
    stop: function(event, ui) {
      alert(event.type);
    }
  });
});
</script>
```

Callback-функция, которую мы определили в опции `stop`, может принимать два аргумента. Первый аргумент — объект события. Мы используем его внутри функции, обращаясь к свойству `type`, чтобы вывести в окне предупреждения название события, вызвавшего функцию.

В листинге 18.4.5 приведен пример задания опций `start`, `selected` и `stop`. В опциях `start` и `stop` определяются callback-функции, вызываемые при начале и окончании выбора элементов. В опции `selected` можно определить callback-функцию, которая вызывается для каждого выбранного элемента по окончании выбора.

Листинг 18.4.5. Использование плагина Selectable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-4-5</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
```

```
<style type="text/css">
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; height: 18px; }
  p { width:400px; height:200px; border:1px solid #369; background-color:#eee;
overflow:auto; font-size:.8em; }
</style>
<script type="text/javascript">
$(function(){
  $("#selectable").selectable({
    start: function(event) {
      $("#result").empty().append("Событие: " +
      event.type + "<br />");
    },
    selected: function(event, ui) {
      $("#result").append("Событие: " + event.type +
       " для " + ui.selected.innerHTML + "<br />");
    },
    stop: function(event) {
      $("#result").append("Событие: " +
      event.type + "<br />");
    }
  });
});
</script>
</head>
<body>
<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
  <li class="ui-widget-content">Item 6</li>
  <li class="ui-widget-content">Item 7</li>
</ol>
<p id="result"></p>
</body>
</html>
```

Давайте посмотрим, что будет происходить, если мы попробуем испытать в действии пример из листинга 18.4.5. Как только мы начнем выбор элементов, будет вызвана функция, определенная нами в опции `start`. Эта функция очистит все содержимое элемента с идентификатором `#result` и добавит в него название события, вызвавшего функцию.

Предположим, что мы выбираем элементы, передвигая указатель мыши и удерживая в нажатом состоянии ее левую кнопку. Договоримся также, что мы закончим выбор, как только будут выбраны два элемента `li`.

В этом случае, по окончании выбора будет сначала вызвана функция, определенная в опции `selected`. Причем у нас она будет вызвана дважды, для каждого выбранного элемента. Затем будет вызвана функция, которую мы определили в опции `stop`.

Обратите внимание, что в функцию, которая была определена в опции `selected`, мы передаем аргумент `ui` и используем его внутри функции, обращаясь к его свойству `selected`. Это свойство является объектом, представляющим элемент DOM, который был выбран.

Плагин Selectable имеет несколько методов, с помощью которых можно управлять им даже после инициализации. Примеры иллюстрирует листинг 18.4.6.

Листинг 18.4.6. Использование плагина Selectable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-4-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/le-frog/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #selectable .ui-selecting { background: #FECA40; }
  #selectable .ui-selected { background: #F39814; color: white; }
  #selectable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
  #selectable li { margin: 3px; padding: 0.4em; font-size: 1.4em; height: 18px;
}
</style>
<script type="text/javascript">
$(function(){
  $("#selectable").selectable();
  $("#disable").click(function(){
    $("#selectable").selectable("disable");
  });
  $("#enable").click(function(){
    $("#selectable").selectable("enable");
  });
  $("#destroy").click(function(){
    $("#selectable").selectable("destroy");
  });
});
```

```
</script>
</head>
<body>
<ol id="selectable">
  <li class="ui-widget-content">Item 1</li>
  <li class="ui-widget-content">Item 2</li>
  <li class="ui-widget-content">Item 3</li>
  <li class="ui-widget-content">Item 4</li>
  <li class="ui-widget-content">Item 5</li>
  <li class="ui-widget-content">Item 6</li>
  <li class="ui-widget-content">Item 7</li>
</ol>
<button id="disable">Disable</button>
<button id="enable">Enable</button>
<button id="destroy">Destroy</button>
</body>
</html>
```

В листинге 18.4.6 мы создаем три кнопки с идентификаторами `#disable`, `#enable` и `#destroy`, связывая с ними события `click`. При нажатии на кнопки вызываем соответствующий метод плагина Selectable, временно отменяя функциональность плагина, вновь восстанавливая эту функциональность, и удаляя ее окончательно, возвращая элементы в состояние, предшествующее инициализации.

Описания всех доступных методов плагина Selectable приведены в табл. 18.12.

Таблица 18.12. *Методы плагина Selectable*

Метод	Описание
destroy `.selectable('destroy')`	Полностью удаляет всю функциональность плагина Selectable. Возвращает элементы в состояние, предшествующее инициализации
disable `.selectable('disable')`	Временно запрещает всю функциональность плагина. Вновь разрешить ее можно с помощью метода `enable`
enable `.selectable('enable')`	Разрешает использование всей функциональности плагина, если ранее она была запрещена методом `disable`
option `.selectable('option', optionName, [value])`	Устанавливает значение любой опции плагина после инициализации
option **`.selectable('option', optionName)`**	С помощью этого метода можно получить значение любой опции плагина после инициализации
widget **`.selectable('widget')`**	Обеспечивает доступ к объекту, который представляет собой выбираемый элемент

Таблица 18.12 *(окончание)*

Метод	Описание
refresh `.selectable('refresh')`	Этот метод обычно применяется, если для опции `autoRefresh` установлено значение `false`. С помощью него можно пересчитать положение и размеры каждого выделенного элемента

18.5. Sortable — сортировка элементов

ЗАДАЧА

Необходимо добавить на веб-страницу возможность сортировки элементов с помощью указателя мыши.

Решение

Для решения этой задачи воспользуемся плагином Sortable (листинг 18.5.1). Для оформления выберем тему "sunny".

Листинг 18.5.1. Использование плагина Sortable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-5-2</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/sunny/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #sortable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
  #sortable li { margin: 0 3px 3px 3px; padding: 0.4em; padding-left: 1.5em;
font-size: 1.4em; height: 18px; }
  #sortable li span { position: absolute; margin-left: -1.3em; }
</style>
<script type="text/javascript">
$(function(){
  $("#sortable").sortable();
});
</script>
</head>
<ul id="sortable">
  <li class="ui-state-default">Item 1</li>
```

```
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>
</body>
</html>
```

Обсуждение

Итак, сначала мы подключили файл стилей css/sunny/jquery-ui-1.8.9.custom.css одной из многочисленных тем оформления. Кроме этого мы подключили файл библиотеки — js/jquery-1.4.4.min.js и файл js/jquery-ui-1.8.9.custom.min.js, в котором объединена функциональность ядра UI и плагина Sortable. За связь плагина с HTML-разметкой отвечает фрагмент кода, приведенный в листинге 18.5.2.

Листинг 18.5.2. Использование плагина Sortable

```
<script type="text/javascript">
$(function(){
  $("#sortable").sortable();
});
</script>
```

В примере из листинга 18.5.2 мы используем плагин с настройками по умолчанию, но поскольку его можно очень гибко настраивать с помощью множества различных опций, то обязательно нужно познакомиться с тем, как это делается (листинг 18.5.3).

Листинг 18.5.3. Использование плагина Sortable

```
<script type="text/javascript">
$(function(){
  $("#sortable").sortable({
    placeholder: "ui-state-highlight",
    opacity: 0.6
  });
});
</script>
```

В примере из листинга 18.5.3 мы передаем плагину две опции. Опция `placeholder` передает плагину имя CSS-класса, который будет применен к той позиции, откуда элемент начал перемещение или куда он может быть помещен, а опция `opacity` устанавливает прозрачность элемента во время перемещения.

Значения остальных опций передаются плагину подобным образом. Список доступных опций плагина Sortable приведен в табл. 18.13.

Таблица 18.13. *Опции плагина Sortable*

Опция	Описание
`disabled`	Значение по умолчанию — `false`. Если установить значение `true`, то при инициализации функциональность Sortable будет недоступна, однако может быть включена впоследствии, например, при выполнении какого-либо условия
`appendTo`	Значение по умолчанию — `'parent'`. В ней можно передать селектор jQuery или элемент, который будет контейнером для объекта, представляющего сортируемый элемент во время его перемещения
`axis`	Значением является строка. Доступные значения `'x'` или `'y'`. Значение по умолчанию — `false`. Если определить эту опцию, то сортируемые элементы смогут перемещаться только по вертикали или горизонтали
`cancel`	В качестве значения используется селектор jQuery. Предотвращает сортировку, если она начинается на элементе, указанном в этой опции. Значение по умолчанию — `':input,button'`
`connectWith`	В качестве значения принимает селектор jQuery, где можно указать другой сортируемый список, в который перемещать сортируемые элементы так, что они станут частью другого сортируемого списка
`containment`	Ограничивает перемещение внутри определенного элемента или области. В качестве значения может принимать селектор jQuery, элемент или строку. Примеры возможных значений: `'parent'`, `'document'`, `'window'`. По умолчанию — `false`
`cursor`	Строка, определяющая вид курсора в процессе перемещения элемента. По умолчанию установлено значение `'auto'`
`cursorAt`	Объект, который определяет положение указателя мыши во время перемещения элемента. Для задания координат используется комбинация одного или двух свойств из четырех возможных — `'top'`, `'right'`, `'bottom'` и `'left'`. Например: `{top: 10, left: 20}` или `{bottom: 5}`. По умолчанию — `false`
`delay`	Число, указанное в этой опции, определяет отсрочку начала перемещения элемента. Задается в миллисекундах. Опция помогает предотвратить нежелательное перемещение элемента во время случайного щелчка мышью. Значение по умолчанию — 0
`distance`	Число в пикселах, определяющее расстояние, которое должен пройти указатель мыши (с одновременно нажатой левой кнопкой), чтобы начался процесс перемещения элемента. Опция помогает предотвратить нежелательное перемещение элемента во время случайного щелчка мышью. Значение по умолчанию — 1
`dropOnEmpty`	Значение по умолчанию — `true`. В этом случае, если один из связанных списков пустой, в него можно перемещать элементы из другого списка. Если установить значение `false`, эта возможность будет запрещена
`forceHelperSize`	Значение по умолчанию — `false`. Установка значения `true` принудит хелпер иметь размеры

Таблица 18.13 *(продолжение)*

Опция	Описание
`forsePlaceholderSize`	Значение по умолчанию — `false`. Установка значения `true` потребует задать размеры того места списка, откуда элемент начал перемещение или куда он может быть помещен
`grid`	По умолчанию — `false`. Значением может быть массив из двух чисел, определяющий шаг сетки, по которой будет перемещаться сортируемый элемент, например `grid: [20,20]`
`handle`	Значение по умолчанию — `false`. В качестве значения можно указать селектор jQuery или элемент. В этом случае процесс сортировки можно будет начать только в том случае, если левая кнопка мыши нажата в момент нахождения указателя мыши над элементом, определенным в этой опции
`helper`	Значение по умолчанию — `'original'`. В этом случае при сортировке перемещаемый элемент представлен самим элементом. Другое возможное значение — `'clone'` — элемент при перемещении представлен своей копией
`items`	В качестве значения используется селектор jQuery. Значение по умолчанию — `'> *'`. Опция определяет, какие элементы, являющиеся элементами-потомками того элемента, с которым связана функциональность плагина, могут быть сортируемыми
`opacity`	Значение опции может изменяться от 0,01 до 1 и определяет прозрачность перемещаемого элемента. По умолчанию — `false`
`placeholder`	Имя CSS-класса, который будет применен к той позиции, откуда элемент начал перемещение. Если значение опции не определено, "пустая" позиция никак не оформляется
`revert`	Значение по умолчанию — `false`. Если задать значение `true`, то при перемещении элемента, после того, как будет отпущена кнопка мыши, элемент переместится на свою новую позицию с использованием плавного анимационного эффекта
`scroll`	По умолчанию установлено значение `true` — при перемещении элемента к краю области просмотра она автоматически прокручивается бесконечно. Если установить значение `false`, эта возможность будет запрещена
`scrollSensitivity`	Число, определяющее расстояние в пикселах от края области просмотра, после которого она начинает прокручиваться. Расстояние отсчитывается относительно указателя мыши, а не перемещаемого элемента. Значение по умолчанию — 20
`scrollSpeed`	Число, определяющее скорость, с которой прокручивается область просмотра при приближении указателя мыши к ее краю на расстояние, заданное в опции `scrollSensitivity`. Значение по умолчанию — 20
`tolerance`	Значение по умолчанию — `'intersect'`. При этом элемент, перемещаемый во время сортировки, должен перекрыть любой другой сортируемый элемент как минимум на 50%, чтобы тот "освободил" занимаемое место. Другое возможное значение — `'pointer'`. При этом над сортируемым элементом должен оказаться только указатель мыши

Таблица 18.13 *(окончание)*

Опция	Описание
zIndex	Число, которое определяет значение CSS-свойства z-index для элемента в момент его сортировки

Настало время познакомиться с событиями, на которые умеет реагировать плагин Sortable. Пример использования одного из возможных событий плагина приведен в листинге 18.5.4, а описания событий для плагина Sortable — в табл. 18.14.

Таблица 18.14. *События плагина Sortable*

Опция	Событие	Описание
create	sortcreate	Наступает в момент инициализации
start	sortstart	наступает в момент начала сортировки
sort	sort	Происходит постоянно в течение сортировки
change	sortchange	Наступает во время сортировки, но только в том случае, если изменилось положение сортируемого элемента в объектной модели документа
beforeStop	sortbeforeStop	Наступает в момент сортировки, перед ее окончанием (когда отпущена левая кнопка мыши)
stop	sortstop	Наступает в момент окончания сортировки
update	sortupdate	Наступает в момент окончания сортировки, но только в том случае, если порядок сортируемых элементов был изменен
receive	sortreceive	Происходит, когда связанный сортируемый список принимает элемент из другого списка
remove	sortremove	Наступает, когда элемент покидает один связанный список и перемещается в другой
over	sortover	Происходит, когда сортируемый элемент перемещен в связанный список
out	sortout	Наступает, когда сортируемый элемент перемещен из связанного списка
activate	sortactivate	Для каждого связанного списка наступает при начале процесса сортировки
deactivate	sortdeactivate	Для каждого связанного списка наступает по окончании процесса сортировки

Листинг 18.5.4. Использование плагина Sortable

```
<script type="text/javascript">
$(function(){
  $("#sortable").sortable({
    placeholder: "ui-state-highlight",
```

```
    opacity: 0.6,
    stop: function(event, ui) {
      alert(event.type);
    }
  });
});
</script>
```

В JavaScript-коде примера из листинга 18.5.4 мы с помощью опции `stop` определяем функцию, вызываемую в момент окончания сортировки. Функция может принимать два аргумента. Первый аргумент — объект события, второй — специальный объект `ui`, в свойствах которого можно обнаружить разнообразную полезную информацию.

Но пока в примере из листинга 18.5.4 мы задействуем только свойство `type` объекта события, чтобы вывести в окне предупреждения название события, вызвавшего нашу функцию.

Пример того, как можно использовать специальный объект `ui`, приведен в листинге 18.5.5.

Листинг 18.5.5. Использование плагина Sortable

```
<script type="text/javascript">
$(function(){
  $("#sortable").sortable({
    placeholder: "ui-state-highlight",
    opacity: 0.6,
    helper: "clone",
    change: function(event, ui) {
      ui.helper.css("color","#f00");
    }
  });
});
</script>
```

В примере из листинга 18.5.5 мы задаем опцию `change`, чтобы вызвать функцию в момент изменения положения сортируемого элемента. Внутри функции мы обращаемся к одному из свойств объекта `ui` — к свойству `helper`. Это свойство является объектом, который представляет перемещаемый элемент во время сортировки. С помощью метода `css(name,value)` мы устанавливаем красный цвет шрифта на перемещаемом элементе. Но, как только этот элемент займет свое новое место в списке, цвет шрифта вернется к исходному значению. Догадались почему? Если нет, то обратите внимание на опцию `helper`. Установив значение `clone` в этой опции, мы использовали во время перемещения не сам исходный объект, а его копию.

Кстати, `helper` — это не единственное полезное свойство объекта `ui`, есть и другие:

- `ui.helper` — объект, характеризующий элемент, находящийся в процессе перемещения (обычно клон ui.item);

- `ui.position` — объект, в свойствах `top` и `left` которого содержится информация о положении перемещаемого элемента относительно родительского элемента;
- `ui.offset` — объект, в свойствах `top` и `left` которого содержится информация об абсолютном положении перемещаемого элемента;
- `ui.item` — объект, представляющий сортируемый элемент;
- `ui.placeholder` — объект, представляющий место, откуда был перемещен сортируемый элемент или куда он будет перемещен;
- `ui.sender` — объект, который представляет элемент-контейнер для сортируемых элементов, откуда сортируемый элемент был перемещен (при наличии связанных списков).

Нам осталось познакомиться с теми методами, которые предоставляет плагин Sortable. Пример использования метода `.sortable("cancel")` приведен в листинге 18.5.6, а описания других методов плагина — в табл. 18.15.

Таблица 18.15. *Методы плагина Sortable*

Метод	Описание
destroy `.sortable('destroy')`	Полностью удаляет всю функциональность плагина Sortable. Возвращает элементы в состояние, предшествующее инициализации
disable `.sortable('disable')`	Временно запрещает использование всей функциональности плагина. Вновь разрешить ее можно с помощью метода `enable`
enable `.sortable('enable')`	Разрешает всю функциональность плагина, если ранее она была запрещена методом `disable`
option `.sortable('option', optionName, [value])`	С помощью этого метода можно установить значение любой опции плагина после инициализации
option **`.sortable('option', optionName)`**	С помощью этого метода можно получить значение любой опции плагина после инициализации
widget **`.sortable('widget')`**	С помощью этого метода можно получить доступ к объекту, который представляет собой сортируемый элемент
serialize `.sortable('serialize', [option])`	Упорядочивает значение атрибутов `id` элементов сортируемого списка в строку, которую можно передать на сервер с помощью AJAX-запроса. Предъявляет требования к формату записи значения атрибута `id`. Допустимые форматы: `id='name_number'` или `id='name-number'`. В этом случае вид строки получается `'name[]=number& name[]=number'`. Вторым, необязательным параметром, можно передать объект. Возможные значения: • `'key'` — заменит часть `'name[]'` на необходимую вам;

Таблица 18.15 *(окончание)*

Метод	Описание
	• 'attribute' — попробует получить значения из атрибута, отличного от id; • 'expression' — можно использовать свое регулярное выражение
toArray .sortable('toArray')	Упорядочивает значение атрибутов id элементов сортируемого списка в массив
cancel .sortable('cancel')	Отменяет результат последней операции сортировки и возвращает элемент в состояние, предшествующее этой операции. Метод полезен при использовании в callback-функциях, связанных с событиями stop или receive

Листинг 18.5.6. Использование плагина Sortable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-5-6</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/sunny/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #sortable { list-style-type: none; margin: 0; padding: 0; width: 60%; }
  #sortable li { margin: 0 3px 3px 3px; padding: 0.4em; padding-left: 1.5em;
font-size: 1.4em; height: 18px; }
  #sortable li span { position: absolute; margin-left: -1.3em; }
</style>
<script type="text/javascript">
$(function(){
  $("#sortable").sortable({
  placeholder: "ui-state-highlight",
  opacity: 0.6
  });
  $("#cancelSort").click(function(){
    $("#sortable").sortable("cancel");
  });
});
</script>
</head>
<ul id="sortable">
```

```
  <li class="ui-state-default">Item 1</li>
  <li class="ui-state-default">Item 2</li>
  <li class="ui-state-default">Item 3</li>
  <li class="ui-state-default">Item 4</li>
  <li class="ui-state-default">Item 5</li>
  <li class="ui-state-default">Item 6</li>
  <li class="ui-state-default">Item 7</li>
</ul>
<button id="cancelSort">Cancel Sort</button>
</body>
</html>
```

Метод `.sortable("cancel")`, пример которого приведен в листинге 18.5.6, наверное, один из самых полезных. Он позволяет отменить результат последней операции сортировки и вернуть элементы к предшествующему состоянию.

Обратите внимание на элемент `button` с идентификатором `#cancelSort`. С кнопкой связан обработчик события `click`, по которому и вызывается соответствующий метод.

Если вы попробуете переместить один из элементов списка на новое место, а затем нажать кнопку **cancelSort**, список вернется в предыдущее состояние.

И в заключение самый интересный пример, реализующий связанные списки. В примере из листинга 18.5.7 сортируемые элементы можно переносить из одного списка в другой. Элемент, перемещенный из одного списка в другой, становится частью списка, в который он был перемещен.

Листинг 18.5.7. Использование плагина Sortable

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="ru" xml:lang="ru">
<head>
<title>example-18-5-7</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link type="text/css" href="css/sunny/jquery-ui-1.8.9.custom.css"
rel="stylesheet" />
<script src="js/jquery-1.4.4.min.js" type="text/javascript"></script>
<script src="js/jquery-ui-1.8.9.custom.min.js" type="text/javascript"></script>
<style type="text/css">
  #sortable1, #sortable2 { list-style-type: none; margin: 0 30px; padding: 0;
width: 200px; float:left; }
  #sortable1 li, #sortable2 li { margin: 0 3px 3px 3px; padding: 0.4em;
padding-left: 1.5em; font-size: 1.4em; height: 18px; }
  #sortable1 li span, #sortable2 li span { position: absolute; margin-left: -
1.3em; }
</style>
<script type="text/javascript">
```

```
$(function(){
  $("#sortable1, #sortable2").sortable({
    connectWith: ".connectedSortable",
    tolerance: "pointer"
  });
});
</script>
</head>
<ul id="sortable1" class="connectedSortable">
  <li class="ui-state-default">Item 1 [1]</li>
  <li class="ui-state-default">Item 2 [1]</li>
  <li class="ui-state-default">Item 3 [1]</li>
  <li class="ui-state-default">Item 4 [1]</li>
  <li class="ui-state-default">Item 5 [1]</li>
  <li class="ui-state-default">Item 6 [1]</li>
  <li class="ui-state-default">Item 7 [1]</li>
</ul>
<ul id="sortable2" class="connectedSortable">
  <li class="ui-state-default">Item 1 [2]</li>
  <li class="ui-state-default">Item 2 [2]</li>
  <li class="ui-state-default">Item 3 [2]</li>
  <li class="ui-state-default">Item 4 [2]</li>
  <li class="ui-state-default">Item 5 [2]</li>
  <li class="ui-state-default">Item 6 [2]</li>
  <li class="ui-state-default">Item 7 [2]</li>
</ul>
</body>
</html>
```

В HTML-коде страницы, приведенной в листинге 18.5.7, ничего особенного нет — просто два ненумерованных списка `#sortable1` и `#sortable2`. С помощью JavaScript-кода мы сделаем их сортируемыми, указав их идентификаторы в селекторе jQuery и связав с ними функциональность Sortable. Самое главное заключается в использовании опции `connectWith`. Здесь мы указываем имя класса для того списка, который хотим сделать связанным с другим списком.

Обратите внимание, что атрибут `class` со значением `connectedSortable` мы присвоили обоим спискам и добились того, что элементы могут перемещаться не только из первого списка во второй, но и наоборот. Если бы мы захотели организовать только "одностороннее движение", скажем, из первого списка во второй, то присвоили бы класс `connectedSortable` только второму списку.

ПРИЛОЖЕНИЕ

Описание компакт-диска

На компакт-диске, прилагаемом к книге, находятся примеры, разобранные в *частях I* и *II*.

Таблица П1. *Содержание компакт-диска*

Каталог	Описание
01-selectors	Примеры к главе 1 — "Выбор элементов"
02-attributes	Примеры к главе 2 — "Атрибуты элементов"
03-effects	Примеры к главе 3 — "Визуальные эффекты"
04-css	Примеры к главе 4 — "Работа с CSS-свойствами"
05-data	Примеры к главе 5 — "Работа с данными в jQuery"
06-manipulation	Примеры к главе 6 — "Манипуляции над элементами"
07-traversing	Примеры к главе 7 — "Перемещение по элементам"
08-events	Примеры к главе 8 — "События и их обработка"
09-ajax	Примеры к главе 9 — "Взаимодействие jQuery и AJAX"
10-additional	Примеры к главе 10 — "Полезные вспомогательные функции и методы jQuery"
11-menu	Примеры к главе 11 — "Меню для веб-сайта"
12-tables	Примеры к главе 12 — "Работа с таблицами"
13-charts	Примеры к главе 13 — "Графики и диаграммы"
14-forms	Примеры к главе 14 — "AJAX-формы"
14-forms/jqueryForm	То же jQuery Form
14-forms/jqueryUploadify	То же jQuery Uploadify
14-forms/jqueryValidate	То же jQuery Validate
15-galleries	Примеры к главе 15 — "Фотогалерея для сайта"
16-plug-ins	Примеры к главе 16 — "Несколько полезных плагинов"

Таблица П1 *(окончание)*

Каталог	Описание
16-plug-ins/cluetip	Примеры использования плагина jQuery ClueTip
16-plug-ins/cookie	То же jQuery Cookie
16-plug-ins/timers	То же jQuery Timers
17-ui-widgets	Примеры к главе 17 — "UI jQuery — виджеты"
17-ui-widgets/accordion	Примеры использования виджета Accordion
17-ui-widgets/autocomplete	То же Autocomplete
17-ui-widgets/button	То же Button
17-ui-widgets/datepicker	То же Datepicker
17-ui-widgets/dialog	То же Dialog
17-ui-widgets/progressbar	То же Progressbar
17-ui-widgets/slider	То же Slider
17-ui-widgets/tabs	То же Tabs
18-ui-interactions	Примеры к главе 18 — "UI jQuery — взаимодействие с элементами страницы"
18-ui-interactions/draggable	Примеры использования плагина jQuery Draggable
18-ui-interactions/droppable	То же jQuery Droppable
18-ui-interactions/resizable	То же jQuery Resizable
18-ui-interactions/selectable	То же jQuery Selectable
18-ui-interactions/sortable	То же jQuery Sortable

Литература

1. Флэнаган Д. JavaScript. Подробное руководство. — СПб.: Символ-плюс, 2008.
2. Рейсиг Дж. JavaScript. Профессиональные приемы программирования. — СПб.: Питер, 2008.
3. Бибо Б., Кац И. jQuery. Подробное руководство по продвинутому JavaScript. — СПб.: Символ-плюс, 2008.
4. Бенкен Е., Самков Г. AJAX: Программирование для Интернета. — СПб.: БХВ-Петербург, 2009.
5. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP. Разработка динамических Web-приложений. — СПб.: Символ-плюс, 2006.
6. Харрингтон Дж. PHP. Трюки. — СПб.: Питер, 2008.
7. Закас Н., Мак-Пит Дж., Фосетт Дж. AJAX для профессионалов. — СПб.: Символ-плюс, 2008.
8. Мейер Э. CSS — каскадные таблицы стилей. Подробное руководство, 2-е издание. — СПб.: Символ-плюс, 2007.
9. Шмит К. CSS. Рецепты программирования. — СПб.: БХВ-Петербург, 2007.
10. Кузнецов М., Симдянов И., Голышев С. PHP5. Практика разработки Web-сайтов. — СПб.: БХВ-Петербург, 2006.
11. Кузнецов М., Симдянов И. MySQL на примерах. — СПб.: БХВ-Петербург, 2007.

Предметный указатель

О

П

С

Ф

Э