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:

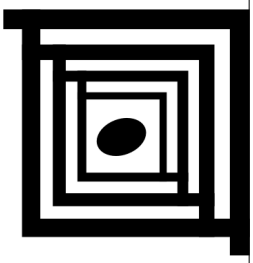
- **http://jquery.com/** — официальный сайт библиотеки jQuery (англ.);
- **http://api.jquery.com/** — оригинальная документация по библиотеке jQuery (англ.);
- **http://jqueryui.com/** — официальный сайт UI jQuery (англ.);
- **http://www.linkexchanger.su** — блог автора книги, содержит много статей с 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.

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

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