jquery.js education.eksmo.ru/downloads/ jquery

http://www.codething.ru/ajax.php

**AJAX и jQuery. Динамическое обновление контента. Основы (изменения от 03.01.2012)**

**Автор:** Михаил Пестречихин  
**Источник:** <http://www.codething.ru/ajax.php>

В данной статье речь пойдет о том, что же такое AJAX и jQuery и будут рассмотрены примеры, как их использовать.

AJAX - инструмент для построения веб-приложений, обменивающихся данными с сервером в фоновом режиме. При этом пользователь получает приложение с динамическим обновлением контента, без перезагрузки всей страницы.

jQuery - JavaScript-framework, библиотека, позволяющая более удобно использовать некоторые возможность Javascript, такие как: создание визуальных эффектов, обработка событий, работа с DOM и поддержка AJAX.

Скачать последнюю версию jQuery и подробно ознакомиться со всеми возможностями можно на сайте разработчика: [http://www.jquery.com/](http://codething.ru/go.php?site=http://www.jquery.com/)

В данной статье мы будем рассматривать только одну функцию библиотеки jQuery, а именно функцию $.ajax(). Эта функция позволяет нам как передавать данные на сервер, так и получать ответы от сервера и все это в фоновом режиме, без перезагрузки страницы. Настройка приема или передачи данных зависит от параметров, с которыми вызывается функция $.ajax(). Основные из них будут рассмотрены ниже. Подробнее о параметрах можно прочесть в руководстве по jQuery.

Перейдем к рассмотрению примеров.

**Важно!**  
Для того, чтобы примеры работали корректно, необходимо:  
1. Все файлы должны быть записаны в кодировке UTF-8.  
2. Скрипты должны выполняться на веб-сервере, а не запускаться в браузере, как файл.

**Пример 1. Динамическое обновление контента по таймеру**

Создадим простенькую программу, отображающую текущее время, обновляемое раз в секунду по таймеру. Особенностью данной программы будет получение информации о текущем времени из другого внешнего файла.

Содержимое файла index.html.

1. <html>
2. <head>
3. <meta http-equiv="Content-Type" content="text/html; Charset=UTF-8">
4. <script type="text/javascript" src="jquery.js"></script>
5. </head>
7. <body>
9. <div id="content"></div>
11. <script>
12. function show()
13. {
14. $.ajax({
15. url: "time.php",
16. cache: false,
17. success: function(html){
18. $("#content").html(html);
19. }
20. });
21. }
23. $(document).ready(function(){
24. show();
25. setInterval('show()',1000);
26. });
27. </script>
29. </body>
30. </html>

В коде имеются несколько особенностей, поясним их.

1. Подключение библиотеки jQuery происходит в заголовке HTML файла, для этого записана данная строка.

1. <script type="text/javascript" src="jquery.js"></script>

Сам файл jquery.js - находится в той же папке, что и файлы примера.

2. В теле документа создается контейнер, в который мы будем загружать контент.

1. <div id="content"></div>

3. Странная, на первый взгляд, функция $(document).ready() требуется для корректной работы jQuery, к тому же в ней мы можем выполнить все приготовления к работе программы. В нашем случае мы вызываем функцию show(), в которой прописан механизм получения контента из другого файла, и настраиваем таймер, так, чтобы функция show() вызывалась один раз в секунду.

1. $(document).ready(function(){
2. show();
3. setInterval('show()',1000);
4. });

4. Функция show() состоит из обращения к функции $.ajax() с определенным рядом параметров, которая позволяет нам в фоновом режиме получить информацию из внешнего файла на сервере.

1. $.ajax({
2. url: "time.php",
3. cache: false,
4. success: function(html){
5. $("#content").html(html);
6. }
7. });

Рассмотрим используемые параметры функции $.ajax().

1. url: "time.php"

Обращается к файлу time.php для получения контента.

1. cache: false

Результаты запросов не кэшируются.

1. success: function(html){
2. $("#content").html(html);
3. }

При успешном выполнении запроса, управление переходит функции, которая получает контент в качестве параметра и записывает его контейнер. Запись в контейнер происходит вот в этой строке:

1. $("#content").html(html);

Содержимое файла time.php.

1. <?php echo date("H:i:s"); ?>

Смысл работы файла time.php - выводим текущее время на экран.

Скачать исходные файлы примера (16,6 кб):  
<http://codething.ru/files/ajax_ex1.zip>

**Пример 2. Динамическое обновление контента по выбору пользователя**

Программа, динамически загружающая контент, по выбору пользователя.

Содержимое файла index.html.

1. <html>
2. <head>
3. <meta http-equiv="Content-Type" content="text/html; Charset=UTF-8">
4. <script type="text/javascript" src="jquery.js"></script>
5. </head>
7. <body>
9. <p>Какую страницу желаете открыть?</p>
10. <form>
11. <input id="btn1" type="button" value="Страница 1">
12. <input id="btn2" type="button" value="Страница 2">
13. </form>
14. <div id="content"></div>
16. <script>
17. $(document).ready(function(){
19. $('#btn1').click(function(){
20. $.ajax({
21. url: "page1.html",
22. cache: false,
23. success: function(html){
24. $("#content").html(html);
25. }
26. });
27. });
29. $('#btn2').click(function(){
30. $.ajax({
31. url: "page2.html",
32. cache: false,
33. success: function(html){
34. $("#content").html(html);
35. }
36. });
37. });
39. });
40. </script>
42. </body>
43. </html>

В теле документа создана форма, имеющая две кнопки, посредством которых пользователь осуществляет выбор желаемого контента. И контейнер для загрузки контента.

Загрузка динамического контента в контейнер осуществляется аналогично примеру 1. Разница лишь в том, что грузится различный контент, если происходят разные события.

Событие нажатия на кнопку "Страница 1" обрабатывается функцией $('#btn1').click(), а событие нажатия на кнопку "Страница 2" обрабатывается функцией $('#btn2').click().

Содержимое файлов page1.html и page2.html, которые загружаются в область контента динамически, представляет собой простые HTML-страницы или текстовые файлы с контентом.

Скачать исходные файлы примера (18,4 кб):  
<http://codething.ru/files/ajax_ex2.zip>

**Пример 3. Отправка данных на сервер в фоновом режиме и получение контента**

Рассмотрим пример, отправляющий введенное имя пользователя на сервер. Сервер при получении имени выдает приветствие и подсчитывает количество символов в введенном имени.

Содержимое файла index.html.

1. <html>
2. <head>
3. <meta http-equiv="Content-Type" content="text/html; Charset=UTF-8">
4. <script type="text/javascript" src="jquery.js"></script>
5. </head>
7. <body>
9. <form id="myForm">
10. Введите имя:<br/>
11. <input id="username" type="text" size="20"><br/><br/>
12. <input type="submit" value="Отправить">
13. </form>
15. <div id="content"></div>
17. <script>
18. $(document).ready(function(){
20. $('#myForm').submit(function(){
21. $.ajax({
22. type: "POST",
23. url: "greetings.php",
24. data: "username="+$("#username").val(),
25. success: function(html){
26. $("#content").html(html);
27. }
28. });
29. return false;
30. });
32. });
33. </script>
35. </body>
36. </html>

В теле документа создана форма для ввода имени пользователя. И контейнер для загрузки динамического контента.

Заметим, что сама форма не имеет привычных полей action и method. В качестве обработчика события нажатия на кнопку "Отправить", выступает функция $('#myForm').submit(). Рассмотрим эту функцию.

1. $('#myForm').submit(function(){
2. $.ajax({
3. type: "POST",
4. url: "greetings.php",
5. data: "username="+$("#username").val(),
6. success: function(html){
7. $("#content").html(html);
8. }
9. });
10. return false;
11. });

Как мы видим, основная работа опять связана с функцией $.ajax(). В этот раз появляются дополнительные параметры, не рассмотренные в примерах 1 и 2. А именно:

1. type: "POST"

Тип передачи данных.

1. data: "username="+$("#username").val()

Параметры, передаваемые серверу. В данном случае мы передаем содержимое поля username - имя пользователя. В общем случае, параметры записываются также, как в методе GET, одной строкой, например:

1. data: "username=Vasya&age=18&sex=male"

Обратим внимание, что в конце написана строка:

1. return false;

Это сделано для того, чтобы форма не пыталась передать данные файлу, из которого она запускается и не происходило перезагрузки страницы.

Содержимое файла greetings.php.

1. <?php
2. echo "Приветствую, <b>".$\_REQUEST['username']."</b>!<br>";
3. echo "В вашем имени букв: ".strlen($\_REQUEST['username']).".";
4. ?>

Выводим на экран приветствие и подсчитываем количество символов в имени.

Скачать исходные файлы примера (16,8 кб):  
<http://codething.ru/files/ajax_ex3.zip>

В качестве еще одного примера использования AJAX и jQuery можно посмотреть гостевую книгу из статьи   
«[Гостевая книга своими руками на PHP и MySQL. Добавляем динамизма. AJAX + jQuery](http://www.codething.ru/guestbook2.php)».

Стоит сказать в заключении, что кроме очевидных плюсов создания страниц с использованием динамического обновляющегося контента, есть ряд недостатков, которые следует учитывать при разработке сайта, а именно:

1. На страницах с динамическим обновлением контента, кнопка "Назад" в браузере не работает корректно.

2. Страницы с динамическим обновлением контента не меняют URL, в зависимости от своего содержимого, поэтому не могут быть добавлены в закладки.

3. Страницы с динамическим обновлением контента не индексируются поисковыми системами, т.к. они не выполняют команды JavaScript.

От приведенных недостатов можно избавиться программным путем. В данной статье такие способы не рассматриваются.

Хотите дополнить - пишите: [studio@pestr.ru](mailto:studio@pestr.ru)

**Статьи по теме**

[AJAX и JavaScript. Загрузка контента без перезагрузки страницы](http://www.codething.ru/ajax_js.php);  
[AJAX и jQuery. Как загрузить скрипт JavaScript динамически?](http://www.codething.ru/ajax_script.php);  
[AJAX и jQuery. Передача параметров из PHP в JavaScript. Использование JSON](http://www.codething.ru/ajax_json.php);  
[Гостевая книга своими руками на PHP и MySQL. Добавляем динамизма. AJAX + jQuery](http://www.codething.ru/guestbook2.php).