# Библиотеки JavaScript. jQuery

## Ввдение

**jQuery** — [библиотека JavaScript](http://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующаяся на взаимодействии [JavaScript](http://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript) и [HTML](http://ru.wikipedia.org/wiki/HTML" \o "HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](http://ru.wikipedia.org/wiki/Document_Object_Model" \o "Document Object Model), обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный [API](http://ru.wikipedia.org/wiki/API" \o "API) по работе с [Ajax](http://ru.wikipedia.org/wiki/Ajax" \o "Ajax).

## Основные возможности

* Движок кроссбраузерных CSS-селекторов [Sizzle](http://ru.wikipedia.org/w/index.php?title=Sizzle&action=edit&redlink=1" \o "Sizzle (страница отсутствует))[[2]](http://ru.wikipedia.org/wiki/JQuery#cite_note-1), выделившийся в отдельный проект;
* Переход по дереву DOM, включая поддержку [XPath](http://ru.wikipedia.org/wiki/XPath" \o "XPath) как плагина;
* События;
* Визуальные эффекты;
* [AJAX](http://ru.wikipedia.org/wiki/AJAX)-дополнения;
* JavaScript-[плагины](http://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD" \o "Плагин).

## Использование

jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл:

<head>

<script type="text/javascript" src="путь/к/jQuery.js"></script>

</head>

Вся работа с jQuery ведётся с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним — jQuery. Второй способ считается более правильным, а чтобы код не получался слишком громоздким можно писать его следующим образом:

jQuery(function($) {

// здесь код скрипта, где в $ будет находиться объект, предоставляющий доступ к функциям jQuery

})

Работу с jQuery можно разделить на 2 типа:

* Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции [текучего интерфейса](http://ru.wikipedia.org/wiki/Fluent_interface" \o "Fluent interface).
* Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора [CSS](http://ru.wikipedia.org/wiki/CSS" \o "CSS), что возвращает объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery. Например,

$("div.test").add("p.quote").addClass("blue").slideDown("slow");

находит все элементы div с классом test, а также все элементы p с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз. Здесь методы add, addClass и slideDown возвращают ссылку на исходный объект $("div.test"), поэтому возможно вести такую цепочку.

Методы, начинающиеся с $., удобно применять для обработки глобальных объектов. Например:

$.each([1,2,3], function() {

document.write(this + 1);

});

добавит на страницу 234.

$.ajax и соответствующие функции позволяют использовать методы [AJAX](http://ru.wikipedia.org/wiki/AJAX" \o "AJAX). Например:

$.ajax({

type: "POST",

url: "some.php",

data: {name: 'John', location: 'Boston'},

success: function(msg){

alert( "Data Saved: " + msg );

}

});

В этом примере идет обращение к скрипту some.php с параметрами name=John&location=Boston, и полученный результат выдается в сообщении посредством alert().

Пример добавления к элементу обработчика события click с помощью jQuery:

jQuery(function($) {

$("a").click(function() {

alert("Hello world!");

});

});

В данном случае при нажатии на элемент <A> происходит вызов alert("Hello world!").

## Селекторы

Селекторы в jQuery базируются на [CSS селекторах](http://www.w3.org/TR/CSS2/selector.html), а так же поддерживают [XPath](http://www.w3.org/TR/xpath" \l "location-paths). Дабы не закапываться в документацию буду приводить примеры, много примеров. Но начнем с самых азов…  
  
Для начала нам понадобиться макет HTML странички (вполне типичный макет):

<div id="header">  
    <h1><a href="/" title="homepage">Title</a></h1>  
    <h2>Sub-title <span>small description</span></h2>  
</div>  
<div id="wrapper">  
    <div id="content">  
        <div class="post">  
            <h3>Post Title</h3>  
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra tempus sapien.  
               Cras condimentum, tellus id lacinia fermentum, tortor lectus tincidunt sapien,  
               vel varius augue tortor vel tortor.</p>  
            <span>Image Title</span>  
            <img src="/image1.jpg" alt="Image Alt Text"/>  
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra tempus sapien.  
               Cras condimentum, tellus id lacinia fermentum, tortor lectus tincidunt sapien,  
               vel varius augue tortor vel tortor.</p>  
            <span class="inner-banner">Banner Text</span>  
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra tempus sapien.  
               Cras condimentum, tellus id lacinia fermentum, tortor lectus tincidunt sapien,  
               vel varius augue tortor vel tortor.</p>  
        </div>  
        <span id="banner"><img src="/banner1.jpg" alt="Big Banner"/></span>  
        <div class="post">  
            <h3>Post Title</h3>  
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra tempus sapien.  
               Cras condimentum, tellus id lacinia fermentum, tortor lectus tincidunt sapien,  
               vel varius augue tortor vel tortor.</p>  
            <span>Image Title</span>  
            <img src="/image2.jpg" alt="Image Alt Text"/>  
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra tempus sapien.  
               Cras condimentum, tellus id lacinia fermentum, tortor lectus tincidunt sapien,  
               vel varius augue tortor vel tortor.</p>  
            <span class="inner-banner">Banner Text</span>  
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra tempus sapien.  
               Cras condimentum, tellus id lacinia fermentum, tortor lectus tincidunt sapien,  
               vel varius augue tortor vel tortor.</p>  
        </div>  
    </div>  
</div>  
<div id="sidebar">  
    <ul>   
         <li><a href="/item0.html">Menu Item 0</a></li>  
         <li><a href="/item1.html">Menu Item 1</a></li>  
         <li><a href="/item2.html">Menu Item 2</a></li>  
         <li><a href="/item3.html">Menu Item 3</a></li>  
    </ul>  
</div>  
<div id="footer">  
    Copyright &copy; 2008  
</div>

А теперь приступим к выборкам:  
  
Выбор элементов по Id либо ClassName аналогично используемому в CSS

$('#sidebar');    // выбор элемента с id = sidebar  
$('.post');       // выбор элементов с class = post  
$('div#sidebar'); // выбор элемента div с id = sidebar  
$('div.post');    // выбор элементов div с class = post

Примечание: используйте валидные имена классов и id

### Бродим по иерархии объектов в DOM'е

Простой выбор потомков:

$('div span');            // выбор всех span элементов в элементах div

Аналогичный результат так же можно получить используя следующую конструкцию:

$('div').find('span');    // выбор всех span элементов в элементах div

Выбор только непосредственных потомков

$('div > span');          // выбор всех span элементов в элементах div, где span является прямым потомком div'a

Как же лучше поступить, что работает быстрее? Надо бы протестировать…  
  
Так же селекторы можно группировать:

$('div, span');          // выбор всех div и span элементов

Поиск по соседям:

$('span + img');         // выбор всех img элементов перед которыми идут span элементы  
$('span ~ img');         // выбор всех img элементов после первого элемента span  
$('#banner').prev();     // выбор предыдущего элемента от найденого  
$('#banner').next();     // выбор следующего элемента от найденого

Выбор всех элементов, всех предков, всех потомков

$('\*');                // выбор всех элементов  
$('p > \*');            // выбор всех потомков элементов p  
$('p').children();     // --  
$('p').parent();       // выбор всех прямых предков элементов p  
$('\* > p');            // выбор всех предков элементов p (скорей всего Вам не понадобится)  
$('p').parents();      // --  
$('p').parents('div'); // выбор всех предков элемента p которые есть div (parents принимает в качестве параметра селектор)

### Фильтры

Фильтров в jQuery реализовано достаточно много, и пользоваться ими одно удовольствие:

$('div:first');     // выбираем первый div в доме  
$('div:last');      // выбираем последний div в доме  
$('div:not(.red)'); // выбираем div'ы у которых нету класса red  
$('div:even');      // выбираем четные div'ы  
$('div:odd');       // выбираем нечетные div'ы  
$('div:eq(N)');     // выбираем div идущим под номером N в DOMe  
$('div:gt(N)');     // выбираем div'ы, индекс которых больше чем N в DOMe  
$('div:lt(N)');     // выбираем div'ы, индекс которых меньше чем N в DOMe  
$(':header');       // выбо заголовоков h1, h2, h3 и т.д.  
$('div:animated');  // выбор элементов с активной анимацией

Фильтры по контенту и видимости:

$('div:contains(text)'); // выбираем div'ы содержащие текст  
$('div:empty');          // выбираем пустые div'ы  
$('div:has(p)');         // выбираем div'ы которые содержат p  
$('div.red').filter('.bold') // выбираем div'ы которые содержат класс red и класс bold  
$('div:hidden');         // выбираем скрытые div'ы  
$('div:visible');        // выбираем видимые div'ы

Так же есть фильтры по атрибутам:

$("div[id]");           // выбор всех div с атрибутом id  
$("div[title='my']");   // выбор всех div с атрибутом title=my  
$("div[title!='my']");  // выбор всех div с атрибутом title не равного my  
$("div[title^='my']");  // выбор всех div с атрибутом title начинающихся с my   
                        // <div title="myCat">,<div title="myCoffee">, <div title="my...">  
$("div[title$='my']");  // выбор всех div с атрибутом title заканчивающихся на my   
                        // <div title="itsmy">,<div title="somy">, <div title="...my">  
$("div[title\*='my']");  // выбор всех div с атрибутом title содержащим my  
                        // <div title="itsmy">,<div title="myCat">, <div title="its my cat">,<div title="...my...">

так же стоит отдельно отметить следующий фильтр:

$("a[rel~='external']"); // выбор всех A с атрибутом rel содержащим external в списке значений разделенных пробелом

В результате его работы будут выбраны следующие теги:

<a href="" rel="external">link</a> — да  
<a href="" rel="nofollow external">link</a> — да  
<a href="" rel="external nofollow">link</a> — да  
<a href="" rel="friend external follow">link</a> — да  
<a href="" rel="external-link">link</a> — нет

Для работы с элементами форм есть ряд селекторов позволяющий выбирать по типу элемента и фильтров — enabled/disabled/selected/checked:

$(":text");            // выбор всех input элементов с типом =text   
$(":radio");           // выбор всех input элементов с типом =radio  
                       // и так далее  
$("input:enabled");    // выбор всех включенных элементов input  
$("input:checked");    // выбор всех отмеченных чекбоксов

Фильтры так же можно группировать:

$("div[name=city]:visible:has(p)"); // выбор видимого div'a с именем city, который содержит тег p

Приведу так же ряд полезных селекторов для работы с элементами форм:

$("form select[name=city] option:selected").val(); // получение выбранного(-ых) элементов в селекте city  
$("form :radio[name=some]:checked").val(); // получение выбранного значения радиобатона с именем some  
$("form :checkbox:checked"); // выбор всех выбранных чекбоксов

Еще советую почитать статью [jQuery: 8 полезных советов при работе с элементом SELECT](http://www.kachayev.ru/2009/01/08/jquery-8-poleznyx-sovetov-pri-rabote-s-elementom-select/)  
  
Если Вам хочеться опробывать как это все работает — то для этого можете воспользоваться [тестовой страничкой](http://anton.shevchuk.name/wp-demo/jquery-tutorials-4/index.html)

## События

Для удобства разработки, на AJAX запросах висит несколько event'ов, их можно задавать для каждого AJAX запроса в отдельности, либо глобально. На все event'ы можно повесить свою функцию.  
  
Пример для отображения элемента с id=«loading» во время выполнения любого AJAX запроса:

$("#loading").bind("ajaxSend", function(){  
    $(this).show(); // показываем элемент  
}).bind("ajaxComplete", function(){  
    $(this).hide(); // скрываем элемент  
});

Для локальных событий — вносим изменения в опции метода ajax():

$.ajax({  
    beforeSend: function(){  
        // Handle the beforeSend event  
    },  
    complete: function(){  
        // Handle the complete event  
    }  
    // ...  
});

Для большей наглядности, приведу следующую диаграмму (кликабельно):  
[](http://anton.shevchuk.name/wp-content/uploads/2008/10/events.png)  
  
  
Ну и собственно список всех event'ов:

* **ajaxStart** — Данный метод вызывается в случае когда побежал AJAX запрос, и при этом других запросов нету
* **beforeSend** — Срабатывает до отправки запроса, позволяет редактировать XMLHttpRequest. Локальное событие
* **ajaxSend** — Срабатывает до отправки запроса, аналогично beforeSend
* **success** — Срабатывает по возвращению ответа, когда нет ошибок ни сервера, ни вернувшихся данных. Локальное событие
* **ajaxSuccess** — Срабатывает по возвращению ответа, аналогично success
* **error** — Срабатывает в случае ошибки. Локальное событие
* **ajaxError** — Срабатывает в случае ошибки
* **complete** — Срабатывает по завершению текущего AJAX запроса (с ошибкои или без — срабатывает всегда).Локальное событие
* **ajaxComplete** — Глобальное событие, аналогичное complete
* **ajaxStop** — Данный метод вызывается в случае когда больше нету активных запросов

## AJAX

**Примечание**: Во всех примерах используется сокращенный вариант вызова jQuery методов, используя функцию **$** (знак доллара)

### [jQuery(..).load](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/load.html)

Начнем с самого простого — загрузка HTML кода в необходимый нам DOM элемент на странице. Для этой цели нам подойдет метод load. Данный метод может принимать следующие параметры:

1. url запрашиваемой страницы
2. передаваемые данные (необязательный параметр)
3. функция которой будет скормлен результат (необязательный параметр)

Приведу пример JavaScript кода:

// по окончанию загрузки страницы  
$(document).ready(function(){                
    // вешаем на клик по элементу с id = example-1  
    $('#example-1').click(function(){  
        // загрузку HTML кода из файла example.html  
        $(this).load('ajax/example.html');         
    })   
});

Пример подгружаемых данных (содержимое файла example.html):

Example<[br](http://december.com/html/4/element/br.html)/>  
Data Loaded By AJAX<[br](http://december.com/html/4/element/br.html)/>  
Bye-Bye

[Пример работы](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/load.html)

### jQuery.ajax

Это самый основной метод, а все последующие методы лишь обертки для метода jQuery.ajax. У данного метода лишь один входной параметр — объект включающий в себя все настройки (выделены параметры которые стоит запомнить):

* async — асинхронность запроса, по умолчанию true
* cache — вкл/выкл кэширование данных браузером, по умолчанию true
* contentType — по умолчанию «application/x-www-form-urlencoded»
* **data** — передаваемые данные — строка иль объект
* dataFilter — фильтр для входных данных
* **dataType** — тип данных возвращаемых в callback функцию (xml, html, script, json, text, \_default)
* global — тригер — отвечает за использование глобальных AJAX Event'ов, по умолчанию true
* ifModified — тригер — проверяет были ли изменения в ответе сервера, дабы не слать еще запрос, по умолчанию false
* jsonp — переустановить имя callback функции для работы с JSONP (по умолчанию генерируется на лету)
* processData — по умолчанию отправляемые данный заворачиваются в объект, и отправляются как «application/x-www-form-urlencoded», если надо иначе — отключаем
* scriptCharset — кодировочка — актуально для JSONP и подгрузки JavaScript'ов
* timeout — время таймаут в миллисекундах
* **type** — GET либо POST
* **url** — url запрашиваемой страницы

Локальные [AJAX Event'ы](http://habrahabr.ru/post/42426/#ajax-events):

* beforeSend — срабатывает перед отправкой запроса
* error — если произошла ошибка
* success — если ошибок не возникло
* complete — срабатывает по окончанию запроса

Для организации HTTP авторизации (О\_о):

* username — логин
* password — пароль

Пример javaScript'а:

$.ajax({  
    url: '/ajax/example.html',             // указываем URL и  
    dataType : "json",                     // тип загружаемых данных  
    success: function (data, textStatus) { // вешаем свой обработчик на функцию success  
        $.each(data, function(i, val) {    // обрабатываем полученные данные  
            /\* ... \*/  
        });  
    }   
});

### jQuery.get

Загружает страницу, используя для передачи данных GET запрос. Может принимать следующие параметры:

1. url запрашиваемой страницы
2. передаваемые данные (необязательный параметр)
3. callback функция, которой будет скормлен результат (необязательный параметр)
4. тип данных возвращаемых в callback функцию (xml, html, script, json, text, \_default)

### [jQuery.post](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/post.html)

Данный метод аналогичен предыдущему, лишь передаваемые данные уйдут на сервер посредством POST'а. Может принимать следующие параметры:

1. url запрашиваемой страницы
2. передаваемые данные (необязательный параметр)
3. callback функция, которой будет скормлен результат (необязательный параметр)
4. тип данных возвращаемых в callback функцию (xml, html, script, json, text, \_default)

JavaScript:

$(document).ready(function(){                          // по завершению загрузки страницы  
    $('#example-3').click(function(){                  // вешаем на клик по элементу с id = example-3  
        $.post('ajax/example.xml', {}, function(xml){  // загрузку XML из файла example.xml     
            $('#example-3').html('');  
            $(xml).find('note').each(function(){       // заполняем DOM элемент данными из XML  
                $('#example-3').append('To: '   + $(this).find('to').text() + '<br/>')  
                               .append('From: ' + $(this).find('from').text() + '<br/>')  
                               .append('<b>'    + $(this).find('heading').text() + '</b><br/>')  
                               .append(           $(this).find('body').text() + '<br/>');  
            });  
        }, 'xml');                                     // указываем явно тип данных  
    })  
});

Файл example.xml:

<?xml version="1.0" encoding="UTF-8"?>  
<note>  
<to>Tove</to>  
<from>Jani</from>  
<heading>Reminder</heading>  
<body>Don't forget me this weekend!</body>  
</note>

[Пример работы](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/post.html)

### [jQuery.getJSON](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/json.html)

Загружает данные в формате JSON (удобней и быстрее нежели XML). Может принимать следующие параметры:

1. url запрашиваемой страницы
2. передаваемые данные (необязательный параметр)
3. callback функция, которой будет скормлен результат (необязательный параметр)

JavaScript:

$(document).ready(function(){                            // по завершению загрузки страницы  
    $('#example-4').click(function(){                    // вешаем на клик по элементу с id = example-4  
        $.getJSON('ajax/example.json', {}, function(json){  // загрузку JSON данных из файла example.json     
            $('#example-4').html('');  
                                                         // заполняем DOM элемент данными из JSON объекта  
            $('#example-4').append('To: '   + json.note.to + '<br/>')  
                           .append('From: ' + json.note.from + '<br/>')  
                           .append('<b>'    + json.note.heading + '</b><br/>')  
                           .append(           json.note.body + '<br/>');  
        });                  
    })  
});

Файл example.json:

{  
    note:{  
        to:'Tove',  
        from:'Jani',  
        heading:'Reminder',  
        body:'Don\'t forget me this weekend!'  
    }  
}

[Пример работы](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/json.html)

### [jQuery.getScript](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/javascript.html)

данная функция загружает и выполняет локальный JavaScript. Может принимать следующие параметры:

1. url запрашиваемого скрипта
2. callback функция, которой будет скормлен результат (необязательный параметр)

JavaScript:

$(document).ready(function(){                          // по завершению загрузки страницы  
    $('#example-5').click(function(){                  // вешаем на клик по элементу с id = example-5  
        $.getScript('ajax/example.js', function(){     // загрузку JavaScript'а из файла example.js   
            testAjax();                                // выполняем загруженный JavaScript  
        });                  
    })  
});

Файл example.js:

function testAjax() {  
    $('#example-5').html('Test completed');  // изменяем элемент с id = example-5  
}

[Пример работы](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/javascript.html)

### Отправка Формы

Для отправки формы посредством jQuery можно использовать любой из перечисленных способов, а вот для удобства «сбора» данных из формы лучше использовать плагин [jQuery Form](http://malsup.com/jquery/form/)

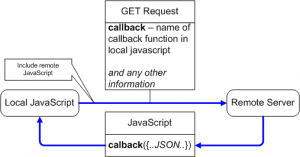
### Отправка Файлов

Для отправки файлов посредством jQuery можно использовать плагин [Ajax File Upload](http://www.phpletter.com/Our-Projects/AjaxFileUpload/) иль [One Click Upload](http://www.michaelmitchell.co.nz/one-click-upload/)

### Взаимодействие с PHP

Для организации работы с PHP использую бибилотеку [jQuery-PHP](http://jquery.hohli.com), удобно если Вам нравится jQuery ;), подробней читаем в статье [PHP библиотека для jQuery](http://anton.shevchuk.name/php/php-library-for-jquery/)

### Примеры использования JSONP

Отдельно стоит отметить использование [JSONP](http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/) — ибо это один из способов осуществления кросс-доменной загрузки данных. Если немного утрировать — то это подключение удаленного JavaScript'a, содержащего необходимую нам информациию в формате JSON, а так же вызов нашей локальной функции, имя которой мы указываем при обращении к удаленному серверу (обычно это параметр **callback**). Чуть более наглядно это можно продемонстрировать следующая диаграмма (кликабельно):  
  
[](http://anton.shevchuk.name/wp-content/uploads/2008/10/jsonp.png)  
При работе с jQuery имя callback функции генерируется автоматически для каждого обращения к удаленному серверу, для этого достаточно использовать GET запрос ввида:

http://api.domain.com/?type=jsonp&query=test&callback=?

Вместо последнего знака вопроса (?) будет подставлено имя callback функции. Если же Вы не хотите использовать данный способ, то Вам необходимо будет явно указать имя callback функции, используя опцию jsonp при вызове метода jQuery.ajax().

#### [Google Поиск](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/google-search.html)

Пример получения и обработки результатов поиска используя Google, более подробную информацию найдете в статье "[jQuery + AJAX + (Google Search API || Yahoo Search API)](http://anton.shevchuk.name/javascript/jquery-and-google-search-api-or-yahoo-search-api-yahoo-search-api/)"

#### [Yahoo Поиск](http://anton.shevchuk.name/wp-demo/jquery-tutorials-3/yahoo-search.html)

Пример получения и обработки результатов поиска используя Yahoo, более подробную информацию найдете в статье "[jQuery + AJAX + (Google Search API || Yahoo Search API)](http://anton.shevchuk.name/javascript/jquery-and-google-search-api-or-yahoo-search-api-yahoo-search-api/)"

#### JSONP API

Приведу так же небольшой список открытых API с поддержкой JSONP:

* [Google](http://code.google.com/apis/ajaxsearch/documentation/reference.html) — поиск и большинство сервисов
* [Yahoo](http://developer.yahoo.com/search/) — поиск и большинство сервисов
* [Flickr](http://www.flickr.com/services/api/)
* [MediaWiki](http://en.wikipedia.org/w/api.php) — соответственно и все производные — Wikipedia, Wiktionary и т.д.
* [Digg](http://apidoc.digg.com/)
* [CNET](http://api.cnet.com/)
* [aideRSS](http://apidoc.aiderss.com/Main_Page)

## Источники

<http://ru.wikipedia.org/wiki/JQuery> – Википедия. jQuery

<http://habrahabr.ru/post/51717/> - jQuery для начинающих. Часть 4. Селекторы

<http://habrahabr.ru/post/42426/> - jQuery для начинающих. Часть 3. AJAX