

# Фреймворк jQuery

# Функция **\$()**

**`$(html)`**  
**`$(elems)`**  
**`$(fn)`**  
**`$(expr, context)`**

## **`$(html)`**

Позволяет создать html-элементы «на лету» из «чистого» HTML. Например, мы можем создать элемент `div`, содержащий параграф с текстом «Ба-бах!» и добавить его к элементу с `id="body"` таким образом:

```
var my_div = $("<div><p>Ба-бах!</p></div>");  
my_div.appendTo("#body");
```

## **`$(elems)`**

Позволяет «прицепить» всю функциональность jQuery к уже существующим элементам на странице (а именно к элементам из объектной модели документа, из DOM).

```
$(document.body).css( "background-color", "black" );  
$( myForm.elements ).hide();
```

## **\$(expr[, context])**

- наиболее часто используемая функция. Первый, обязательный, параметр – это выражение, которое позволит jQuery найти элемент на странице. Второй, необязательный, параметр указывает, где искать этот элемент (по умолчанию jQuery будет искать по всей странице).

Например: Найдем все элементы p, находящиеся внутри всех элементов div на странице

**\$("#div > p");**

Найдем все радиокнопки в первой форме на странице

**\$("#input:radio", document.forms[0]);**

Найдем все элементы div в XML, что прислан с сервера с помощью технологии AJAX:

**\$("#div", xml.responseXML);**

Найдем все видимые элементы li внутри элемента ul

**\$("#ul > li:visible"); // CSS + селектор**

## **\$(fn)**

- выполняет код в тот момент, когда доступна объектная модель документа, то есть когда браузер уже получил исходный код страницы полностью, но, возможно, еще не подгрузил различное мультимедийное содержимое (рисунки, видео, флэш).

```
$(document).ready(  
  function() { /* ... */  
});
```

или его сокращенным вариантом, **\$(fn)**:

```
$(  
  function(){/* ... */  
});
```

Пример:

```
$(document).ready(  
  function()  
  {  
    // добавим ко всем ссылкам на странице некий текст  
    $("a").append("<span>(opens in new window)</span>");  
  }  
);
```

# Функции JQuery

**append(content)/prepend(content)** - Добавить переданный элемент или выражение в конец/в начало выбранного элемента.

**appendTo(expr)/prependTo(expr)** - Добавить выбранный элемент в конец/в начало переданного элемента.

**attr(name)** - Получить значение атрибута.

**attr(params)** - Установить значение атрибутов. Атрибуты передаются в виде {ключ1:значение1[, ключ2:значение2[, ...]]}

**attr(name, value)** - Установить значение одного атрибута.

**css(name)/css(params)/css(name, value)** - Получить/установить значение отдельных параметров CSS. Аналогично функции attr().

**text()/text(val)** - Получить/задать текст элемента. В случае ввода текста специальные символы HTML заменяются символьными примитивами (entities, например, знак ">" будет заменен &gt;).

**Empty()** - Удалить все подэлементы текущего элемента.

Пример:

Код

```
<div id="my-div">  
  <a href="http://google.com/" id="my-link">Ссылка</a>  
</div>
```

**`$("#my-div").html();`**

Вернет `<a href="#" id="my-link">Ссылка</a>`

**`$("#my-div").append("<strong>Полужирный текст</strong>");`**

**// или**

**`$("#<strong>Полужирный текст</strong>").appendTo($("#my-div");`**

HTML станет таким:

```
<div id="my-div">  
  <a href="http://google.com/" id="my-link">Ссылка</a>  
  <strong>Полужирный текст</strong>  
</div>
```

**`$("#my-div").empty();`**

HTML станет таким:

```
<div id="#my-div"></div>
```

Пример:

HTML

`<p><span>Hello</span>, how are you?</p>`

jQuery

`$("p").find("span").html("Ahoy").end().append("<p>I'm fine<p>");`

Код	Текущая коллекция элементов
<code>\$("p")</code>	<code>[&lt;p&gt;..&lt;/p&gt;]</code>
<code>.find("span")</code>	<code>[&lt;span&gt;..&lt;/span&gt;]</code>
<code>.html("Ahoy")</code>	<code>[&lt;span&gt;..&lt;/span&gt;]</code>
<code>.end()</code>	<code>[&lt;p&gt;..&lt;/p&gt;]</code>
<code>.append(...);</code>	<code>[&lt;p&gt;..&lt;/p&gt;]</code>

Результат HTML:

`<p><span>Ahoy</span>, how are you?<p>I'm fine</p></p>`

Пример:

Цвет всех ссылок на странице станет зеленым:

```
$("a").css("color", "green");
```

## **Индексы и get([index])**

Возвращаемая функцией `$()` коллекция является «псевдо»-массивом DOM-элементов. Поэтому можно обратиться к индивидуальным элементам по их индексу:

```
$("p")[0].className = "test";
```

```
$("p"); // Получили объект jQuery
```

```
$("p").get(); // Получили соответствующую коллекцию DOM-элементов
```

```
$("a").get(0); // Получили первую из всех ссылок на странице
```

## **each(fn)**

Иногда нужно пробежаться по всем элементам коллекции и выполнить над ними какие-то действия. Для этого нам понадобится функция `each`. Эта функция принимает в качестве аргумента другую функцию. `each()` работает в контексте найденных элементов и поэтому каждый раз, когда выполняется переданная ей функция, в этой функции доступна переменная `this`, указывающая на соответствующий DOM-элемент.



# Animate

Ключевой функцией, на которой базируются все остальные, является функция `animate`:

**`animate(params, speed, easing, callback);`**

Здесь:

- `params` – свойства, которые участвуют в анимации в виде пар {ключ: значение}. Например: {height: "show"} или {opacity: 50, width: 100, height: 200}.
- `speed` – скорость в миллисекундах.
- `easing` – замедление анимации (замедляется ли к концу, "easein", или, наоборот, ускоряется, "easeout". Дополнительные типы доступны в модулях расширения).
- `callback` – функция, которая будет вызвана после завершения анимации.

Пример.

Пусть у нас есть элемент `div` с каким-нибудь текстом. Мы хотим скрыть этот текст, заменить новым, и показать обновленный текст.

```
$("#mydiv")  
  .animate({height: "hide"}, 300)  
  .text("Новый текст")  
  .animate({height: "show"}, 300);
```

## Эффекты

Метод `animate` является основой большинства, если не всех, эффектов jQuery и плагинов. Например, jQuery предлагает следующие методы для показа и скрытия элементов:

- `show([speed[, callback]])` – показать элемент;
  - `hide([speed[, callback]])` – скрыть элемент;
  - `fadeIn(speed[, callback])` – показать элемент путем изменения его прозрачности;
  - `fadeOut(speed[, callback])` – скрыть элемент путем изменения его прозрачности;
  - `slideDown(speed, callback)` – показать элемент, спустив его сверху;
  - `slideUp(speed, callback)` – показать элемент, подняв его снизу;
- где

`speed` – скорость в миллисекундах или одно из "slow" (600 миллисекунд) или "fast" (200 миллисекунд);

`callback` – функция, которая будет вызвана после выполнения анимации.

# AJAX

Базовыми функциями для работы с AJAX являются `post()` и `get()` (есть еще более низкоуровневая, `ajax()`, но мы ее не будем рассматривать):

**`$.post(url[, params[, callback]])`**

**`$.get(url[, params[, callback]])`**

Здесь:

- `url` – адрес страницы, на которую будет отправлен запрос;
- `params` – параметры, передаваемые в запросе в виде пар «ключ : значение»;
- `callback` – функция, которая будет вызвана в случае успешного завершения вызова.

Пример:

```
$.post(
  '/ajaxtest.php',
  {
    type: "test-request",
    param1: "param1",
    param2: 2
  },
  onAjaxSuccess
);
function onAjaxSuccess(data)
{ // Здесь мы получаем данные, отправленные сервером
  alert(data);}
```