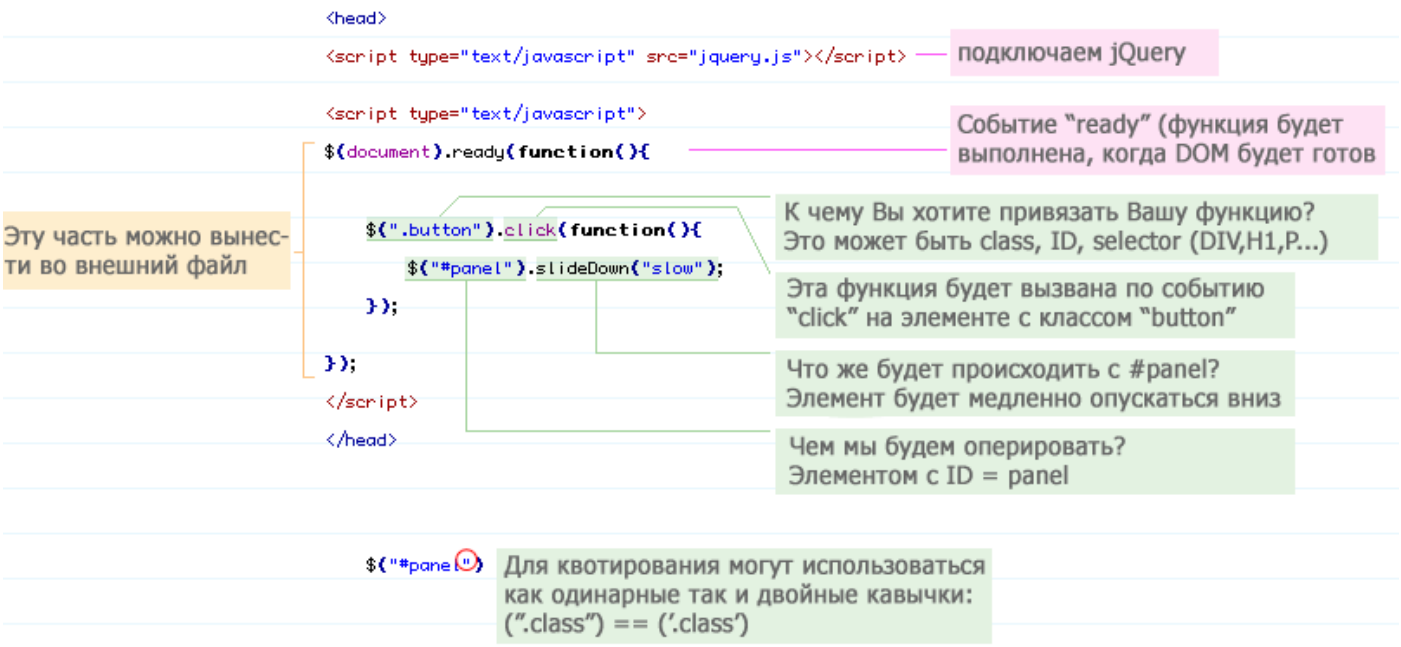
jQuery — библиотека, которая позволяет делать код короче, а также устраняет наиболее занятные кросс-браузерные различия. Она не содержит встроенных архитектурных решений, т.е. не навязывает свой стиль, просто делает код короче.

Библиотеку jQuery можно скачать с сайта http://jquery.com, а можно — вставить в документ, используя jQuery's CDN:

```
<script src="js/jquery-3.31.2.min.js"></script>
<script src="//code.jquery.com/jquery-3.31.2.min.js"></script>
<script src="//code.jquery.com/jquery-migrate-3.2.1.min.js"></script>
```

Основные моменты библиотеки поможет понять следующая диаграмма:



«Сердцем» jQuery является функция `$()`.
Все вызовы делаются через `$`. Это обычная функция, но объявлена библиотекой. Символ `$` является допустимым символом для имён в JavaScript.
Эта функция служит точкой входа в «волшебный мир» jQuery, а символ $ является удобным коротким псевдонимом пространства имен jQuery.

Простейший поиск элементов выглядит так:

```
var result = $('...CSS-селектор...');
```

Например, `$("div a")` — ссылки `a` внутри `div`. Возможны и более сложные запросы, например:

```
var list = $('li > a:odd:not([href^="http://"])');
// расшифровка: нечётные (:odd) ссылки внутри li,
// кроме тех (:not), у которых атрибут href начинается с http://
```

Вызов `$(селектор)` аналогичен `document.querySelectorAll(селектор)`, но jQuery предоставляет свои поисковые расширения, например псевдофильтр `:checked` — выбранные элементы `input` и `option`, `:header` – заголовок `h1..6`, и другие, более полный список которых доступен в документации.
Для того чтобы понимать, как работает селектор необходимы базовые знания CSS, т.к. именно от принципов CSS отталкивается селектор jQuery:

- $("#header") – получение элемента с id="header"

- $(“h3”) – получить все <h3> элементы
- $(“div#content .photo”) – получить все элементы с классом =”photo” которые находятся в элементе div с id=”content”
- $(“ul li”) – получить все <li> элементы из списка
- $(“ul li:first”) – получить только первый элемент <li> из списка

Следующие два примера аналогичны (выводят сообщение после загрузки страницы):

```
jQuery(document).ready(function(e) {
        alert('Привет')
});

$(document).ready(function(e) {
        alert('Привет')
});
```

Библиотека jQuery — не единственная библиотека JavaScript, в которой используется переменная $, что может привести к конфликтам имен, если в одном документе используется одновременно несколько библиотек. Чтобы не допустить возникновения проблем такого рода, можно передать контроль над переменной $ другим библиотекам, вызвав метод jQuery.noConflict(), как показано ниже:

```
jQuery.noConflict();

jQuery(document).ready(function(e) {
       jQuery("img:odd").mouseenter(function(e) {
               jQuery(this).css("opacity", 0.5);
console.log('Вы навели мышь на элемент <img src="' + jQuery(this).attr('src') + '">');
        }).mouseout(function(e) {
           jQuery(this).css("opacity", 1.0);
console.log('Вы убрали мышь с элемента <img src="' + jQuery(this).attr('src') + '">');
        });
});
```

Вы можете сами определить псевдоним для функции jQuery(). Это делается путем присваивания выбранной вами переменной результата вызова метода noConflict():

```
var jq = jQuery.noConflict();

jq(document).ready(function(e) {
   jq("img:odd").mouseenter(function(e) {
              jq(this).css("opacity", 0.5);
console.log('Вы навели мышь на элемент <img src="' + jq(this).attr('src') + '">');
   }).mouseout(function(e) {
   jq(this).css("opacity", 1.0);
console.log('Вы убрали мышь с элемента <img src="' + jq(this).attr('src') + '">');
   });
});
```

В этом примере для функции jQuery() создается новый псевдоним jq, который можно использовать в последующих сценариях. Независимо от выбранного способа обращения к основной функции jQuery(), ей передается один и тот же набор аргументов. Возможные варианты вызова этой функции перечислены в таблице ниже:

Вариант вызова	*Описание*
$(функция)	Позволяет указать функцию, которая должна быть выполнена по завершении построения DOM
$(селектор) $(селектор, контекст)	Осуществляет выбор группы элементов в документе с помощью селектора
$(HTMLElement) $(HTMLElement[])	Создает объект jQuery из объекта или массива объектов HTMLElement
$()	Создает пустой набор элементов
$(HTML-код)	Создает новые элементы из фрагмента HTML-кода

Перебор результатов

Результатом поиска является jQuery-объект. Он похож на массив: в нём есть нумерованные элементы и `length`, но методы у него совсем другие.
jQuery-объект также называют «jQuery-коллекцией», «элементами, обёрнутыми в jQuery» и десятком других жаргонных терминов.

Используем jQuery, чтобы выбрать все элементы по селектору `li > a` и перебрать их:

```
<ul>
  <li><a href="http://jquery.com">jQuery</a></li>
  <li><a href="http://jqueryui.com">jQuery UI</a></li>
  <li><a href="http://blog.jquery.com">jQuery Blog</a></li>
</ul>

<script>
var links = $('li > a');

// перебор результатов
for(var i=0; i<links.length; i++) {
  alert( links[i].href );
}
</script>
```

Контекст поиска

Для поиска внутри какого-либо элемента — можно передать его вторым аргументом `$`.
Например, найдём все ***a*** внутри *#menu*:

```
var menu = document.getElementById('menu');
$('a', menu); // поиск аналогичен menu.querySelectorAll('a')
```

Второй аргумент в этом случае называются «контекстом поиска».

В качестве контекста можно передать не только DOM-элемент, но и селектор:

```
$('a', '#menu' );
```

Также можно передать результат другого поиска:

```
var menu = $('#menu');
$('a', menu);
```

Не важно, в каком виде мы хотим указать контекст: DOM-элемент, строка или результат поиска — jQuery понимает всё.

А что, если контекст поиска содержит много элементов? Например, как будет работать запрос `$('a', 'li')`, если `<li>` в документе много?

Здесь все немного сложнее, но, тем не менее, интуитивно понятно.

Если в контексте много элементов, то поиск будет произведён в каждом из них, а затем результаты будут объединены. Повторы элементов при этом отфильтровываются, то есть два раза один и тот же элемент в результате не появится.

Например, найдём `$('a', 'li')` в многоуровневом списке:

```
<ul>
  <li>
    <a href="http://jquery.com">jQuery</a>
    <ul>
      <li><a href="http://blog.jquery.com">jQuery Blog</a></li>
    </ul>
  </li>
  <li><a href="http://sizzlejs.com">Sizzle</a></li>
</ul>

<script>
var links = $('a', 'li');

for(var i=0; i<links.length; i++) {
  alert( i + ": " +  links[i].href ); // 3 ссылки по очереди
}
</script>
```

Метод each

Для более удобного перебора у jQuery-коллекции есть метод each. Его синтаксис похож на `forEach` массива:

`.each( function(index, item) )`

Он выполняет для каждого элемента коллекции перед точкой функцию-аргумент, и передаёт ей номер `index` и очередной элемент `item`.

Используем его вместо for, чтобы перебрать коллекцию найденных ссылок:

```
$('li a').each(function(i, a) {
  alert( i + ": " + a.href);
});
```

У `.each` есть важная возможность, которой нет в `forEach`: возврат false из функции прекращает перебор. Например:

```
<a href="http://wikipedia.ru">Википедия</a>
<ul>
  <li><a href="http://jquery.com">jQuery</a></li>
  <li><a href="http://sizzlejs.com">Sizzle</a></li>
  <li><a href="http://blog.jquery.com">jQuery Blog</a></li>
</ul>

<script>
var links = $('li a'); // найти все ссылки на странице внутри LI

links.each(function(i, a) {
  alert(i + ': ' + a.innerHTML);

  if (i == 1) return false; // стоп на элементе коллекции с индексом 1
});
</script>
```

При переборе `each`, текущий элемент передаётся как `this`. Можно использовать это, чтобы сделать код короче:

```
$('li a').each(function(i) {
  alert(i + ': ' + this.innerHTML);

  if (i == 1) return false;
});
```

Получение конкретного элемента

Даже если найден только один элемент, всё равно результатом поиска будет jQuery-коллекция. Но нам же нужен один элемент - посмотрим, как его получить.

Для получения одного элемента из jQuery-коллекции есть три способа:

1. Прямой доступ по номеру:

```
 alert( $('body')[0] ); // BODY
```

   Это — самый быстрый и прямой доступ, он использует внутреннюю структуру jQuery-коллекции: элементы там хранятся по индексам, но этим он нарушает принцип инкапсуляции.

   Но в jQuery есть специальные методы для получения элемента по номеру.

3. Метод `get(индекс)`, работает так же, как прямой доступ:
   ```
   alert( $('body').get(0) ); // BODY
   ```
   Если элемента с таким номером нет — вызов `get` возвратит undefined.
4. Метод `eq(индекс)` возвращает коллекцию из одного элемента — с данным номером. Он отличается от метода `get(индекс)` и прямого обращения по индексу тем, что возвращает именно jQuery-коллекцию с одним элементом, а не сам элемент. Например:

   ```
   // DOM-элемент для первой ссылки
   $('a').get(0);

   // jQuery-объект из одного элемента: первой ссылки
   $('a').eq(0);
   ```
   Во втором случае вызов `eq` создаёт новую jQuery-коллекцию, добавляет в нее нулевой элемент и возвращает. Это удобно, если мы хотим дальше работать с этим элементом, используя методы jQuery. Если элемента с таким номером нет – `eq` возвратит пустую коллекцию.

«Сцепление» вызовов

Почти все методы jQuery-объекта возвращают jQuery-объект – либо новый, либо тот на котором вызваны, поэтому можно вызывать методы один за другим.

Это называют «сцепление» вызовов, или «чейнинг» (от англ. «chaining»):

```
$('li a[href$=".pdf"]') // ссылки, оканчивающиеся на pdf
  .each(function() {
    this.className = "pdf"; // дать им класс
  })
  .each(function() {
    alert(this.href); // вывести
  });
```

Для удобства чтения каждый новый вызов в чейнинге идёт с отступом и с новой строки.

В итоге – для поиска элементов можно вызвать `$(CSS-селектор[, контекст])`. Результатом будет jQuery-объект, который можно:

- Перебирать как массив.
- Перебирать методом `each`.
- Получить один элемент прямым доступом по индексу, либо вызовом `get` или `eq`.

Селекторы в jQuery базируются на CSS селекторах, а также поддерживают XPath.

Для начала нам понадобиться макет HTML странички (вполне типичный макет):

```
<div id="header">
    <h1><a href="/" title="homepage">Title 1</a></h1>
    <h2>Sub-title <span>small description</span></h2>
</div>
<div id="wrapper">
    <div id="content">
        <div class="post">
            <h3>Post Title 2</h3>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed viverra 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 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 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 3</h3>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed tempus 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 tempus 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 tempus 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
</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`.

Ищем в иерархии объектов в DOM'е

Простой выбор потомков:

```
$('div span');  // выбор всех span элементов в элементах div
```
Аналогичный результат также 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'а
```
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
$('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');       // выбор заголовоков h1, h2, h3 и т.д.
$('div:animated');  // выбор элементов с активной анимацией
```

Фильтры по контенту и видимости:

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

```
<code lang="HTML4strict">
<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 также можно группировать:

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

Пример полезных селекторов для работы с элементами форм:

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

Еще примеры:

```
// 1. удалить все элементы списка mySelect
$('select[@name=mySelect] option').remove();
// 2. добавить в список новый элемент
$('select[@name=mySelect]').append('<option>Новый элемент списка</option>');
// 3. сделать выделенным первый пунки списка
$('select[@name=loadFileName] option:first').attr('selected', 'yes');
// 4. принудительно снять выделение с элемента списка
$('select[@name=loadFileName] option:first').removeAttr('selected');
// 5. получить значение выделенного пункта из списка
// если вы используете атрибут <option value="some value">:
```

```
var file = $('select[@name=loadFileName] option:selected').val();
    // если вас интересует то, что заключено между <option>...</option>:
var file = $('select[@name=loadFileName] option:selected').text();
// 6. проверить, выбран ли какой-нибудь элемент списка
if( typeof $('select[@name=loadFileName] option:selected').text() === 'undefined' ){
       alert('Ни один элемент списка не выбран');
}
// 7. превратить список в "автомасштабируемый"
$('select[@name=loadFileName]').attr('size', $('select[@name=loadFileName]
option').size());
// 8. сделать недоступны для выбора отдельный элемент
$('select[@name=loadFileName] option:contains('текст нужного
элемента')').attr('disabled', 'disabled');
    // разрешить выделение всех ранее недоступных элементов можно так:
$('select[@name=loadFileName] option:disabled').removeAttr('disabled');
```

Задача для проверки понимания

Предположим, что есть документ:

```
<ul>
  <li><a href="http://jquery.com">jQuery</a></li>
  <li><a href="http://sizzlejs.com">Sizzle</a></li>
  <li><a href="http://blog.jquery.com">jQuery Blog</li>
</ul>
```

Подумайте, что вернёт такой вызов:
1) `$('UL')[0]`?
2) `$('UL').get(0).eq(0)`?
3) `$('UL').eq(0).get(0)`?

А что они бы вернули, если бы `UL` не было в документе?

Решение

Вначале посмотрим, что будет, если элементы UL есть.

1. Первый элемент коллекции — это как раз DOM-элемент `<ul>`
2. Вызов `$('UL').get(0)` также вернёт DOM-элемент, и у него нет метода `eq`, так что будет ошибка.
3. Этот вызов отработает корректно: вызов `eq(0)` возвратит коллекцию из первого `<ul>`, из которой `get(0)` далее возьмёт сам элемент.

А что, если UL в документе нет?

1. Первый вызов вернёт undefined.
2. Второй вызов вернёт ошибку, т.к. `$('UL').get(0)` возвратит undefined, и дальше `eq` не сработает.
3. Третий вызов отработает корректно: вызов `eq(0)` возвратит пустую коллекцию, из которой `get(0)` вернёт undefined.