***Практическая работа №5***

***Создание сценариев с использованием JQuery***

***JQuery*** — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API по работе с Ajax.

Простота и лёгкость использования, библиотеки JQuery, в конечном результате и принесла ей такую популярность. Все движения, перемещения, изменения атрибутов объектов DOM, теперь стали необыкновенно просты, по сравнению с JavaScript. На основе этой библиотеки написаны множество плагинов на все случаи жизни. И они только множатся и усовершенствуются.

***Задание 1. Подключение и использование JQuery***

1. Как подключить JQuery к сайту, для того чтоб можно было использовать эту библиотеку?

Скачать, последнюю версию JQuery, можно с официального сайта [jquery.com](http://jquery.com/) ([jquery-1.7.2.min.js](http://lphp.ru/lessons/js/jquery-1.7.2.min.js)). При нажатии на ссылку откроется страница со скриптом, именно её и надо сохранить. Для этого выберите в меню вашего браузера пункт **Файл -> Сохранить как** и укажите директорию вашего сайта куда сохранить эту библиотеку JQuery. Пусть будет называться эта папка **/js**.

***Совет***: лучше не меняйте названия файла библиотеки JQuery. Официальная версия названия, поможет вам в дальнейшем ориентироваться, с какой версией вы работаете. Но если вы, всё же, решили изменить название на работоспособность это ни как не повлияет.

И так, после того как вы скачали файл JQuery и разместили его в папке /js, на своём сайте, можно приступать к подключению.

1. Подключение JQuery выглядит так:

<script type="text/javascript" src="/js/jquery-1.7.2.min.js"></script>

Размещаться эта строка должна между тегами **head** или **body**в любом месте, но перед файлом скриптов (о них мы поговорим ниже).

Размещение JQuery выглядит так:

<head>

<script type="text/javascript" src="/js/jquery-1.7.2.min.js"></script>

</head>

Чтобы эта ссылка была доступна на всём сайте, её нужно прописать в шапке сайта, её ещё называют header, т.е. в часть страницы, которая доступна на всех страницах сайта. Или же, для тех страниц, на которых будет присутствовать код jquery.

Теперь определимся, где мы будем писать сам код JQuery.

Для этого создадим в той же директории /js, файл jq.js (/js/jq.js). И пропишем (подключим) на него ссылку, после подключённой библиотеки JQuery. Выглядит это примерно так:

<head>

<script type="text/javascript" src="/js/jquery-1.7.2.min.js"></script>

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

</head>

В файле jq.js, мы станем писать код для выполнения на сайте. Первый скрипт jquery, мы напишем для проверки работоспособности всех наших подключений.

1. **Первый скрипт JQuery**

Откроем файл jq.js и впишем туда скрипт

$(document).ready(function(){

});

Весь код в большинстве своём, нужно писать в то пустое пространство, между фигурными скобками. Обусловлено это тем, что все манипуляции происходят с элементами HTML и прежде чем с ними начать работу, они должны появиться на странице, т.е. проще сказать, страница должна быть сначала загружена (сформирована браузером), а потом уже, можно приступать к обработке элементов.

Далее мы впишем обработку тегов div (контейнеров), они станут менять цвет и вид курсора, при наведении на них курсора.

$(document).ready(function(){

$("div").bind("mouseover", function(){

$(this).css({"background-color":"#FAFFEA",cursor:"pointer"});

}).bind("mouseout", function(){

$(this).css("background-color", "");

});

});

1. Второй, и более верный способ — подключать через **Google’s Content Delivery Network (CDN)**.Это гарантирует самую последнюю версию.

|  |  |
| --- | --- |
| 1 | <script type="text/javascript" src="[http://ajax.googleapis.com/ajax/l](http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js) |

Можно загрузить и определённую версию:

|  |
| --- |
| <script type="text/javascript" src="<http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js>"> |

В заголовке подключим определения с сервера стандартов

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

……….

</HTML>

**Теперь напишем немного кода в документ**

Текст будем писать внутри тегов

<script type="text/javascript"></script>

Перед тем, как что-то туда писать, научимся выбирать нужные элементы. Это довольно просто — используется знак $.  
Т.е. $(‘выбранные элементы’). Для того, чтобы выбрать все div’ы на этой странице, придется прописать $(‘div’).

$('div p') //выбираем все параграфы, содержащиеся внутри div'ов.

$('#something') //выбираем элемент с id 'something'

$('.something') //выбираем элементы с class'ом 'something'Комбинировать можно как угодно.

$('div p#something a') //выбирает все ссылки из параграфа с id 'something' и с родителем div.

Создадим новый Div на нашей странице.

<div id="something">

<p>this is some text</p>

</div>

Пропишем ему стили в заголовке документа <head>:

<style type="text/css" media="screen">

#something {

width: 200px;

height: 200px;

background-color: red;

color: white;

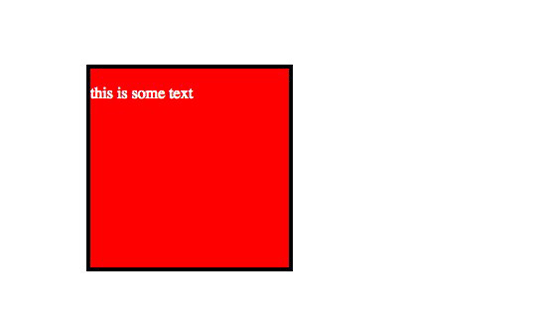
border: 4px solid black;

margin: 100px 0 0 80px;

}

</style>

получится что-то в этом духе:



Заставим его двигаться.

Финальной частью нашего первого урока будет движение этого блока на странице. Для начала выделим его:

$('#something')

Можно конечно использовать и $(‘div’), но лучше всё максимально конкретизировать.  
Чтобы заставить объект двигаться, используем функцию animate() :

$('#something').animate({'margin-top': '300px', 'margin-left': '400px'},1000);

Вообще есть у этой функции три опции, но одна необязательна.

animate(что поменять, скорость, возврат);

Если более детально взглянуть на «что поменять», видим:

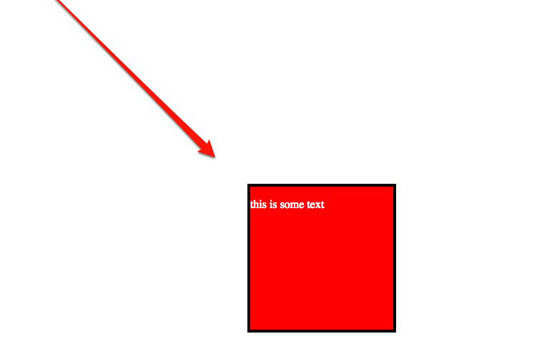
.animate({'параметр': 'величина', 'параметр2': 'величина'}, скорость, возврат)

То есть в итоге мы задаём отступ сверху 300, слева 400, и скорость 1000. 1000=1сек. ‘возврат’ выполняется после совершения движения, но сейчас мы обойдёмся без него.

Итак, возвращаемся к нашему движению

$('#something').animate({'margin-top': '300px', 'margin-left': '400px'},1000);

и вот результат:



***Задание 2*** Методы работы с атрибутами

Создайте код

<div class="code">

<p id="first" class="a">параграф0</p>

    <p class="z">параграф1</p>

    <p class="a" id="middle">параграф2</p>

    <p class="z">параграф3</p>

    <p class="a">параграф4</p>

    <p class="z">параграф5</p>

    <div class="block">

        <span id="b">родительский спан

            <span class="c">дочерний спан</span>

        </span>

        <br/>

        <a href="">ссылка</a>

        <br/>

        <span class="c">свободный спан</span>

    </div>

</div>

### Атрибуты

Форма записи — $(selector).метод(значение)

**attr(name)** — вернёт значение name для выбранных элементов

**attr(name,value)** — установит значение key, равное value, для выбранных элементов

**attr(name,fn)** — установит значение key, равное функции fn, для выбранных элементов

**attr(settings)** — установит атрибуты, указанные в settings, соотв. значениями.

**removeAttr(name)** — удаляет атрибут

**Примеры**

alert($('p').attr('id'))

ничего не высветится, т.к. существует такое правило: если указан определённый вид элементов, то attr будет обращаться к ПЕРВОМУ.  
А вот если мы к пераграфу0 добавим id= “first” , то высветится слово first.

|  |  |
| --- | --- |
| 1 | $('div span').attr('class','next'); |

в этом примере мы добавили для всех спанов, заключённых в div, атрибут class c названием name. Чтобы проверить, появился ли этот класс, воспользуйтесь firebug’ом, или же просто пропишите ему какой-нибудь border.

$('div span').attr('id', function(index){return 'next'+index});

чтобы задать каждому спану отдельный id, мы воспользовались функцией. Таким образом, каждый спан получит id nextX, где X — постоянно растущее число.

### Работа с классами

Форма записи — $(selector).метод(значение)  
 **hasClass(class)** — вернёт true, если выбранный элемент принадлежит к классу class

**addClass(class)** - добавляет класс  
 **removeClass(class)** — удаляет класс  
 **toggleClass(class)** - если выбранные элементы принадлежат к указанному классу — удалит его, а если нет — добавит.

**toggleClass (class,switch)** — если выбранные элементы принадлежат к указанному классу и если switch выставлен на true — удалит его, а если нет — добавит.

**Примеры**

|  |  |
| --- | --- |
| 1 | $('p').toggleClass('z'); |

мы видим, что теперь появились параграфы с классом ‘a z’, и исчезли параграфы с классом z

$('p').click(function(){

if($(this).hasClass('a'))

{

alert('у него класс A!');

$(this).removeClass('a').addClass('z')

}

else(alert('мы его не нашли!'));

});

при нажатии на параграф проверяется условие — если у элемента есть класс a, то появляется надпись ‘у него класс A!’, затем сам класс a ремувится, и добавляется класс Z.

### Значения Value

Форма записи — $(selector).метод(значение)

Работать будем с вот таким кодом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <form name="test" method="post" action="input1.php"/>    <p><b>Ваше имя:</b><br>     <input name="nick" type="text" size="40"/>    </p>    <p><b>Каким браузером в основном пользуетесь:</b><Br>     <input type="checkbox" name="browser" value="ie"/> Internet Explorer<Br>     <input type="checkbox" name="browser" value="opera"/> Opera<Br>     <input type="checkbox" name="browser" value="firefox"/> Firefox<Br>    </p>    <p>Комментарий<Br>     <textarea name="comment" cols="40" rows="3"></textarea></p>    <p><input type="submit" value="Отправить"/>     <input type="reset" value="Очистить"/></p>   </form> |

Эти операторы нужны для работы с Input’ами:

**val()** — возвращает значение атрибута value

**val(string)** — заменяет значение value на string  
 **val(array)** — отметит переданные значения в таких элементах как radio, checkbox, select.

**Примеры:**

|  |  |
| --- | --- |
| 1 | $('input:text').val('xxx') |

в текстовом поле Input’a появятся буквы ‘xxx’

|  |  |
| --- | --- |
| 1 | $('input:submit').val('xxx') |

Тут уже фраза «отправить» заменится на «ххх»

|  |  |
| --- | --- |
| 1 | alert($('input:submit').val()) |

выплывет слово «отправить»

|  |  |
| --- | --- |
| 1 | $('input:checkbox').val(['ie','opera']) |

данное выражение поставит галочки в checkbox’ах ie и opera .

## Задание 3Функциональный разбор DOM

### Фильтрация

Форма записи — $(селектор).метод(параметры)

**is(exp)** - вернёт true, если есть хотя бы одно значение , равное exp

**not(exp)** — вернёт значения , не равные exp

**filter(exp)** — вернёт те значения, которые соответствуют expr

**eq(exp)** — вернёт объект с соответствующим индексом

**slice(start,end)** — выбирает объекты в промежутке от start и до end

**filter(fn)** - вернёт те элементы, для которе соответствуют функции fn

**Примеры:**

|  |  |
| --- | --- |
| 1 | alert($('body > \*').is('span') ? 'ЕСТЬ такое' : 'НЕТ ТАКОГО'); |

у body нет дочерних элементов на первом уровне вложенности(у нас там только div.code), так что ответ будет НЕТ ТАКОГО. Кстати, обратите внимание на знак вопроса — такая конструкция предлагает выбор — если ответ на вопрос будет да, то ‘ЕСТЬ такое’, если нет, то ‘НЕТ ТАКОГО’.

|  |  |
| --- | --- |
| 1 | alert($('body > \*').is('div') ? 'ЕСТЬ такое' : 'НЕТ ТАКОГО'); |

Тут ответ будет ‘ЕСТЬ такое’

|  |  |
| --- | --- |
| 1 | $('p').slice(1,5).css('border','1px solid #f06eaa') |

выбрали все параграфы, и затем отрезали с 1-го по 4-й включительно (поэтому на конце надо писать 5 — до 5-го).

|  |  |
| --- | --- |
| 1 | $('p').slice(-4,-1).css('border','1px solid #f06eaa') |

Тут идёт отсчёт с конца списка. Пишется сверху вниз. Данный код выбирает с первого по 3й элементы с конца, т.е. это параграф2, параграф3, параграф4

|  |  |
| --- | --- |
| 1 | $('p').eq(3).css('border','1px solid #f06eaa') |

выбран параграф3

|  |  |
| --- | --- |
| 1 | $('.code > \*').not('span').css('border','1px solid #f06eaa') |

выбран будет только div.block

|  |  |
| --- | --- |
| 1 | $('.code > \*').not('p[id^=mi]').css('border','1px solid #f06eaa') |

выделит div.block и все параграфы, кроме параграфа2, т.к. он содержит атрибут id, в значении которого есть сочетание mi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $('p').filter(function(){            if(this.hasAttribute('id'))              return true;          else              return false;  }).css('border','1px solid #f06eaa') |

На данном примере мы задали функцию — если у текущего элемента есть атрибут id — возвращаем true. Этой функции соответствует только параграф 2.

### Фильтрация отобранных элементов

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $('p').filter(function(index){          if(index==0 || index==2)              return true;          else              return false;  }).css('border','1px solid #f06eaa') |

На этом примере мы уже работаем с функцией Index , и проверяем индекс элементов. Если он равен 0 или 2 ( ||- это знак ‘или’), то параграф будет подсвечен.

### Относительный поиск элементов

**siblings(exp)** — выбирает все другие элементы на этом уровне (всех соседей)  
 **next(exp)** - выбирает следующий элемент на этом уровне

**nextAll(exp)** — выбирает все следующие элементы на этом уровне

**prev(exp)** - выбирает предыдущий элемент на этом уровне

**prevAll(exp)** — — выбирает все предыдущие элементы на этом уровне

**parent(exp)** — выбирает родительский элемент

**parents(exp)** — выбирает для данного элемента всех родителей , всех уровней  
 **children(exp)** — выбирает дочерний элемент.

**Примеры**

|  |  |
| --- | --- |
| 1 | $('p').siblings().css('border','1px solid #f06eaa') |

выделились все p, и заодно div.block

|  |  |
| --- | --- |
| 1 | $('p:eq(3)').next().css('border','1px solid #f06eaa') |

выделился параграф 4, т.к. он следующий сосед p с индексом 3

|  |  |
| --- | --- |
| 1 | $('div.block').children('a').css('border','1px solid #f06eaa') |

определилась только ссылка

|  |  |
| --- | --- |
| 1 | $('div.block, #middle').parent.css('border','1px solid #f06eaa') |

для обоих элементов родитель — div.code

### Манипуляторы набором элементов

**add(expr)** — добавит к expr\_элементам

**closest(expr)** — найдёт ближайшие родительские элементы , попадающие под expr

**find(expr)** — добавит дочерние элементы , попадающие под expr

**contents()** - все дочерние узлы элементов набра (1-й уровень вложенности)

**end()** — отменяет последнюю диструктивную операцию над набором. Диструктивные операции любым способом отсеивают или изменяют вид набора элементов. К диструктивным операциям относятся add, andSelf, children, filter, find, map, next, not, parent, prev, slice, siblings.

**andSelf()** — объединит теукщий набор с прошлым.

**Примеры**

|  |  |
| --- | --- |
| 1 | $('div.block > span').add('p[class]').css('border','1px solid #f06eaa') |

выделены будут все p с классами, и два дочерних спана div.block

|  |  |
| --- | --- |
| 1 | $('div.block > span').closest('div').css('border','1px solid #f06eaa') |

этот код находит div-родителя элементов span, родителем которых является div.block)

|  |  |
| --- | --- |
| 1 | $('div.code').contents().css('border','1px solid #f06eaa') |

выберется родительский спан, ссылка, свободный спан

|  |  |
| --- | --- |
| 1 | $('div.code').find(':not(p)').css('border','1px solid #f06eaa') |

выделится единственный параграф с id — параграф2

|  |  |
| --- | --- |
| 1 | $('div.code').find(':not(p)').css('border','1px solid #f06eaa') |

найдёт все не-’p'-элементы внутри div.code

## *Задание 5* фильтры

Работать будем с этим HTML-кодом:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <div class="code">      <p class="a">параграф1</p>      <p id="middle">параграф2</p>      <p class="a">параграф3</p>      <div class="block">          <span id="b">родительский спан              <span class="c">дочерний спан</span>          </span>          <br/>          <a href="">ссылка</a>          <br/>          <span class="c">свободный спан</span>        </div>  </div> |

вот его схема:

### Базовые Фильтры

Форма записи — $(селектор:фильтр)

**odd** — нечётные элементы (индексация, мы помним, с 0)

**even** — чётные элементы  
 **gt(index)** — элементы, индекс которых больше, чем index  
 **eq (index)** — элементы, индекс которых равно index

**lt(index)** — элементы, индекс которых меньше, чем index

**header** — заголовки h1,h2..

**animated** — анимированные элементы

**first** — первый из отобранных элементов

**last** — последний из отобранных элементов

not(selector) — все элементы не-selector

Примеры:

|  |  |
| --- | --- |
| 1 | $('p:first').css('border','1px solid #f06eaa') |

выберется первый элемент ‘p’ в документе, т.е. параграф1

|  |  |
| --- | --- |
| 1 | $('p:even').css('border','1px solid #f06eaa') |

выбираем параграф2, т.к. он нечётный (т.к. нумерация начинается с 0, то его порядковый номер — 1.)

|  |  |
| --- | --- |
| 1 | $('\*:not(span)').css('border','1px solid #f06eaa') |

выделяться все элементы, не являющиеся span. Обратите внимание, что наш селектор и фильтр заключены в кавычки.

|  |  |
| --- | --- |
| 1 | $('.code \*:gt(2):not(a)').css('border','1px solid #f06eaa') |

в блоке .code индекс 0 — у параграфа1, 1- у параграфа 2, 2 — у параграфа 3. Значит, выделеине пойдёт после параграфа 2. Рамка будет у всех элементов после параграфа 2, которые не являются ссылками.

### Фильтры контента и видимости

Форма записи — $(селектор:фильтр)

**contains(text)** - элемент содержит какой-то текст

**empty** - пустой элемент

**has(selector)** — элемент содержит selector

**parent** — все родительские элементы

**hidden** — скрытые элементы

**visible** — видимые элементы

**Примеры:**

|  |  |
| --- | --- |
| 1 | $('span:contains(че)').css('border','1px solid #f06eaa') |

вёрнёт нам родительский и дочерний спаны, т.к. фраза ‘дочерний спан’ заключена в них обоих.

|  |  |
| --- | --- |
| 1 | $('span:parent').css('border','1px solid #f06eaa') |

выделились все спаны. Почему? Потому что текст тоже является дочерним элементом. Поэтому все span’ы являются родителями.

|  |  |
| --- | --- |
| 1 | $('span:parent:contains(ро)').css('border','1px solid #f06eaa') |

выделится уже только ‘родительский спан’

|  |  |
| --- | --- |
| 1 | $('span:parent:has(span)').css('border','1px solid #f06eaa') |

и опять выделится только ‘родительский спан’ , т.к. только он содержит span внутри себя.

### Фильтры атрибутов

Форма записи — $(селектор[фильтр])

**atribute** — элементы, у которых есть атрибут

**atribute=value** — элементы, у которых есть атрибут, равный value

**atribute!=value** — элементы, у которых есть атрибут, не равный value

**atribute^=value** — элементы, у которых есть атрибут, начинающийся с value

**atribute$=value** — элементы, у которых есть атрибут, заканчивающийся value

**atribute\*=value** — элементы, у которых есть атрибут, содержащий value

**Примеры**

|  |  |
| --- | --- |
| 1 | $('div[class^=bl]').css('border','1px solid #f06eaa') |

div, у которого атрибут class начинается на bl — это .block.

|  |  |
| --- | --- |
| 1 | $('\*[class\*=a]').css('border','1px solid #f06eaa') |

Из всех элементов только у параграфа1 и параграфа3 класс содержит букву a.

### Семейные фильтры

Форма записи — $(селектор:фильтр)

**nth-child(index)** — дочерний элемент с индексом Index

**nth-child(even)** — чётный дочерний элемент

**nth-child(odd)** — нечётный дочерний элемент

**nth-child(equation)** — все дочерние элементы, подходящие под определённое выражение(2n, 2n+1 ..)

**first-child** — первый дочерний элемент

**last-child** — последний дочерний элемент

**only-child** — вернёт все элементы, которые являются единственными дочерними элементами

**Примеры**

|  |  |
| --- | --- |
| 1 | $('span:first-child').css('border','1px solid #f06eaa') |

Выделяться родительский и дочерний спаны, как дети .block и родительского спана соответственно.

|  |  |
| --- | --- |
| 1 | $('.code \*:nth-child(odd)').css('border','1px solid #f06eaa') |

тут выделиться на первый взгляд больше элементов, чем нужно: параграф1, параграф3, родительский спан, дочерний спан,ссылка,свободный спан..Дело в том, что теги br тоже считаются, как дочерние элементы. (отсчёт тут начинается не с 0, а с 1)

|  |  |
| --- | --- |
| 1 | $('.code \*:nth-child(2)').css('border','1px solid #f06eaa') |

второй дочерний элемент — это ‘парагарф2′

|  |  |
| --- | --- |
| 1  2 | $('.code \*:nth-child(2n+3)').css('border','1px solid #f06eaa') |

n — это 0,1,2,3 и т.п. Значит, мы отбираем 3,5,7,9 и т.п.

|  |  |
| --- | --- |
| 1 | $('.code \*:only-child').css('border','1px solid #f06eaa') |

выделится один единственный ‘дочерний спан’

### Селекторы форм.

Форма записи — $(input:selector)

Все они соответствуют обычным html-инпутам, так что даже объяснять не буду . Виды:

**text**

**password**

**radio**

**checkbox**

**submit**

**reset**

**button**

**image**

**file**

### А так же фильтры форм:

Форма записи — $(селектор:фильтр)

**enabled** - активные элементы

**disabled** — неактивные элементы  
 **checked** — выбирает checkbox’ы, в которых стоит галка  
 **selected** — возвращает выбранный элемент.

**Примеры:**

Рассмотрим немного другой код, уже с input’ами

<form name="test" method="post" action="input1.php">

 <p><b>Ваше имя:</b><br>

  <input name="nick" type="text" size="40">

 </p>

 <p><b>Каким браузером в основном пользуетесь:</b><Br>

  <input type="radio" name="browser" value="ie"> Internet Explorer<Br>

  <input type="radio" name="browser" value="opera"> Opera<Br>

  <input type="radio" name="browser" value="firefox"> Firefox<Br>

 </p>

 <p>Комментарий<Br>

  <textarea name="comment" cols="40" rows="3"></textarea></p>

 <p><input type="submit" value="Отправить">

  <input type="reset" value="Очистить"></p>

</form>

выделиться кнопка ‘отправить’, т.к. только она является submit

$(':submit').css('border','1px solid #f06eaa')

$(':enabled').css('border','1px solid #f06eaa')

вылеляться все активные элементы.

## *Задание 6*

## Orbit — очень лёгкий jQuery-слайдер

## 

## **Orbit** — Симпатичный и очень(!) лёгкий jQuery-слайдер - **всего 2.8kb**. Лучше всего работает в Chrome, Safari, FF3.5+ .

## http://zurb.com/playground/orbit-jquery-image-slider

Для начала работы нам потребуется jQuery и Orbit-плагин. Подключаем в этом порядке.

|  |  |
| --- | --- |
| 1  2 | <script src="js/jquery.min.js" type="text/javascript"></script>  <script src="js/jquery.orbit.min.js" type="text/javascript"></script> |

И подключаем CSS:

В заголовке опишем стиль

<style type="text/css">

#featured { width: 940px; height: 450px; background: #009cff url('orbit/loading.gif') no-repeat center center; overflow: hidden; }

</style>

<link rel="stylesheet" href="css/orbit.css">

А в теле теперь сам код:

<div id="featured">

<img src="link.jpg" title="Link galliantly riding the majestic Epona across the hyrulian plains" alt="Link" />

<img src="ezio.jpg" alt="Ezio" />

<img src="masterchief.jpg" title="Master Chief surveying Reach, home of his upcoming game" alt="Master Chief" />

<img src="marcusfenix.jpg" alt="Marcus Fenix" />

</div>

Замечу, что этот плагин сам распознаёт ширину и высоту картинки, и масштаб. Нестандартные изображения будут обрезаться по бокам.

… И размещаем скрип для активации

<script type="text/javascript">

$(window).load(function() {

$('#featured').orbit({

'bullets': true,

'timer' : true,

'animation' : 'horizontal-slide'

});

});

</script>

В общем-то, всё готово. Напоследок, немного рассмотрим в опции слайдера:

$('#featured').orbit({

animation: 'fade', //fade, horizontal-slide, vertical-slide

animationSpeed: 800, // скорость анимации

advanceSpeed: 4000, // скорость между переходами по таймеру

startClockOnMouseOut: true, // таймер перезагрузится на MouseOut

startClockOnMouseOutAfter: 3000, //через сколько старутет таймер, если мышка отведена.

directionalNav: true, //подсказки при навигации captions: true,

// заголовок будет подписью к картинке

captionAnimationSpeed: 800, //как быстро загружать описание, как я понял: how quickly to animate in caption on load and between captioned and uncaptioned

timer: false //если нужен таймер

## });

## Контрольные вопросы

## Что такое JQuery?

## Для чего используется знак $ при работе с JQuery?

## Как обратиться к JQuery из веб-страницы?

## Перечислите достоинства JQuery

## Перечислите недостатки JQuery