# Практическое занятие №16 JQuery

**Задание №1**

Проверьте запущен ли у Вас Open Server. Папка с материалами урока (jquery16.loc) должна быть размещена в директории /domains сервера. Заходим на http://jquery16.loc и начинаем урок.

**Занятие №2**

Скачать JQuery можно на странице http://jquery.com/download/, после чего он подключается на странице как обычный js файл.

```
<script src="/public/jquery.min.js"></script>
```

Если вы используете JQuery приучите себя подключать его первым JS файлом, поскольку остальные как правило будут использовать JQuery и может возникнуть ситуация, в которой библиотека будет использована раньше, чем подключена.

В ваших файлах с заданиями JQuery уже скачан и подключен.

Обращение с библиотекой JQuery идет через функцию-объект JQuery, которая также имеет короткую форму записи $.

Попробуем использовать JQuery. Сделаем заголовок на странице красным с использованием JQuery (не забудьте выключить кеш браузера):

```
$('h1').css('color', 'red');
```

Как вы можете убедиться, ничего не произошло. Это случилось потому что заголовок еще не существовал на момент подключения файла. У этой проблемы есть несколько вариантов решения:

1. Перенести подключение js файлов в конец html документа:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Подключение Jquery</title>
        <link href="/tasks/task-2/style.css" type="text/css" rel="stylesheet">
    </head>

    <body>
        <h1>Hello world!</h1>

        <script src="/public/jquery.min.js"></script>
        <script src="/tasks/task-2/script.js"></script>
    </body>

</html>
```

2. Использовать функцию $.ready

```
//3 варианта использования $.ready
//для Jquery 1 и 2 использовать можно любой
//однако в 3 версии первые 2,
//с явным использованием .ready
//объявлены устаревшим поэтому далее будем использовать 3 способ

$().ready(function(){
    $('h1').css('color', 'red');
});

$(document).ready(function(){
    $('h1').css('color', 'red');
});

$(function(){
    $('h1').css('color', 'red');
});
```

Использовать можно оба метода решения проблемы загрузки файлов, однако лучше всего использовать или 2 или оба сразу (и подключать js файлы в конце документа и использовать ready). Подключать js в конце рекомендуется чтобы контент страницы грузился быстрее, однако такой способ не всегда доступен. Использование же ready дает 1 значительное преимущество. Если разбивать файлы по функционалу (например, скрипты работой с корзиной помещать в отдельный файл basket.js), то весь функционал конкретного файла будет в отдельной области видимости (поскольку записан внутри функции), и у Вас не будут возникать конфликты имен переменных и функций.

```
//начало файла, используем быстрый вызов функции ready
$(function(){

    //эта функция существует только внутри этого файла
    function f(){
        $('h1').css('color', 'red');
    }

    f();

});
//конец файла, заканчивается функцией, которую мы передаем в ready
```

**Задание №3**

Итак, если первым аргументом функции $() (или JQuery) передать функцию, он воспримет это как короткую запись функции ready, однако если первым аргументом передать строку JQuery воспримет это как селектор и начнет искать по нему элементы. В качестве селекторов можно использовать любые селекторы CSS.

```
$(function(){

    console.log($('h1')); //все элементы h1
    console.log($('h1, h2')); //все элементы h1 или h2
    console.log($('.test')); //все элементы с классом .test
    console.log($('#test')); //элемент с id='test'
    console.log($('a[href^="#"]')); //все ссылки-якоря
    console.log($('a[href^="http"]')); //все ссылки, которые начинаются с http

});
```

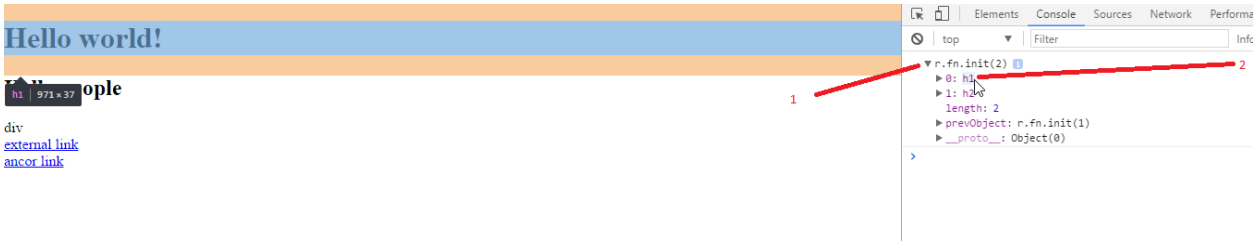
Также можно передать элементы, выбранные с использованием функций клиентского JS:

```
$(function(){

  console.log( $(document.getElementById('test')) );

});
```

Также это могут быть элементы, выбранные с помощью JQuery:

```
$(function(){

    var test = $('#test');
    console.log($(test));

});
```

**Примечание**: для только чтобы увидеть в браузере какие элементы были выбраны и выведены в консоль откройте выведенный в консоль объект и наведите на элемент




**Задание №4**

Повторим селекторы

1. Выберите все элементы списков
2. Выберите все элементы списков из нумерованных списков
3. Выберите все элементы списков из списков с классом «test»
4. Выберите все заголовки на странице
5. Всех прямых потомков элемента с ид main_content

**Задание №5**

После поиска с помощью конструкции $(selector) JQuery возвращает набор элементов, с которыми мы можем дальше работать. Но перед тем как мы приступим к работе с методами модификации элементов, рассмотрим дополнительные методы выборки, которые работают с уже сформированными выборками:

1. .find(). Этот метод осуществляет поиск элементов внутри уже выбранных элементов. Этот метод следует и удобно использовать, когда некоторые элементы уже найдены, и вам необходимо осуществить поиск элементов внутри них.

```
$(function(){
    //вернет тоже самое что и $('#test span')
    console.log($('#test').find('span'));

});
```

2. .children(). Этот метод возвращает потомков выбранных элементов. Если передать селектор, вернет детей, отфильтрованных по селектору. Этот метод следует и удобно использовать, когда некоторые элементы уже найдены, и вам необходимо осуществить поиск элементов внутри них.

```
$(function(){

   //аналогично $('#test > div')
   console.log($('#test').children('div'));

});
```

3. .filter(). Фильтрует набор выбранных элементов. Этот метод следует и удобно использовать, когда некоторые элементы уже найдены, и вам необходимо отфильтровать их по дополнительному параметру.

```
$(function(){

   var allDiv = $('div');
   console.log(allDiv.filter('.test'));
   console.log(allDiv.filter('#test'));

});
```

4. .is(). Проверяет, соответствует ли хотя бы один из выбранных элементов селектору.

```
$(function(){

    var allDiv = $('div');
    console.log(allDiv.is('#test'));
    console.log(allDiv.is('.test'));
    console.log(allDiv.is('.test2'));

});
```

5. .has(). Фильтрует набор выбранных элементов, оставляя только те, которые имеют определенных потомков.

```
$(function(){

    var allDiv = $('div');
    console.log(allDiv.has('span'));

});
```

6. .parent(). Осуществляет поиск родительских элементов всех заданных элементов. Возвращает родительские элементы всех выбранных элементов. Можно указать селектор для фильтрации результата, но чаще всего этот метод используется просто для поиска родителя конкретного элемента.

```
$(function(){

    console.log($('.test').parent());

});
```

7. .parents(). Осуществляет поиск всех предков выбранных элементов, то есть, не только прямых родителей, но и прародителей, прапрародителей и так далее, до начала дерева DOM. можно указать селектор для фильтрации результата.

```
$(function(){

    console.log($('.test').parents());
    console.log($('.test').parents('div'));

});
```

8. .eq(). Принимает целое число – индекс элемента в выборке и возвращает этот элемент.

```
$(function(){

    //второй див на странице
    console.log($('div').eq(1));

});
```

Методы find, filter, is и has, как и выборка с помощью $(selector) помимо селекторов могут принимать элементы, выбранные с помощью JQuery или элементы, выбранные с использованием функций клиентского JS. Поэкспериментируйте с этими функциями передавая им элементы, выбранные с помощью JQuery.

**Задание №6**

Метод .css() возвращает или изменяет значения css-величин у выбранных элементов страницы. Основные варианты использования функции:

1. Получить значение css свойства. Для этого необходимо передать 1 параметр с именем свойства:

```
$(function(){

    console.log($('h1').css('color'));

});
```

2. Установить значение css свойства. Для этого необходимо передать 2 параметра – имя свойства и новое значение.

```
$(function(){

    $('h1').css('color', 'blue');

});
```

3. Установить множество значений css свойств. Для этого необходимо передать 1 параметр объект с набором новых свойств.

```
$(function () {

    $('h1').css({
        'color': 'blue',
        'font-size': '45px',
        textDecoration: 'underline'
    });

});
```

Обратите внимание как можно записывать свойства, которые пишутся через дефис (такие как font-size и text-decoration).

**Задание №7**

Используя JQuery

1. Измените задний цвет всех ненумерованных списков на синий, а цвет текста их элементов на белый
2. Ударим по глазам пользователя.  Каждые 10 секунд меняйте цвет заголовка на один из цветов радуги случайным образом.

**Задание №8**

Метод. text() возвращает или изменяет текстовое содержимое выбранных элементов страницы. Вызов без параметра возвращает текстовое содержание элементов:

```
$(function () {

    console.log($('.test').text());

});
```

Вызов с параметром заменяет все содержимое у выбранных элементов, на переданный текст.

```
$(function () {
    $('.test').eq(0).text('new text');
});
```

Если вы попытаетесь с помощью метода text() поместить в элемент другие элементы с помощью html-текста, то JQuery будет экранировать все теги, и в результате на странице появится html-текст, вместо html-элементов.

```
$(function () {
    $('.test').eq(0).text('<p>new text</p>');
});
```

Для вставки html-элементов необходимо воспользоваться методом из следующего задания.

**Задание №9**

Метод .html() возвращает или изменяет html-содержимое выбранных элементов страницы. Вызов без параметра возвращает html-содержимое элемента. Если элементов в выборке несколько, то значение будет взято у первого.

```
$(function () {
    console.log($('.test').html());
});
```

Вызов с параметром заменяет содержимое всех выбранных элементов на переданный html-код.

```
$(function () {
    console.log($('.test').html('<p>new</p>'));
});
```

**Задание №10**

Сделать упражнение на скорочтение. В файле задания дано 5 параграфов. При загрузке страницы она должна быть пустой, а у пользователя необходимо спросить готов ли он. Как только он нажмет “ОК” должен появится первый абзац, а над ним полоса загрузки. В течении 10 секунд полоса должна равномерно окрашиваться в синий цвет, а когда она заполнится текст должен исчезнуть. У пользователя снова спрашивают готовность, и так пока не будут показаны все абзацы.

**Задание №11**

Для вставки html-кода в конец выбранных элементов можно использовать один из 2 методов – .append() или .appendTo():

```
$(function () {

    //вставляет элемент в конец всех списков на странице
    $('ul').append('<li>three</li>');

    //вставляет элемент в конец всех списков на странице
    $('<li>three</li>').appendTo($('ul'));

});
```

Аналогично, для добавления html-кода в начало выбранных элементов можно использовать – .prepend() и  .prependTo()

```
$(function () {

    //вставляет элемент в начало всех списков на странице
    $('ul').prepend('<li>zero</li>');

    //вставляет элемент в начало всех списков на странице
    $('<li>zero</li>').prependTo($('ul'));

});
```

**Задание №12**

Для вставки html-кода после выбранных элементов можно использовать функции – .after() и .insertAfter()

```
$(function () {

    $('p.test').after('<p>after p.test</p>');

    $('<p>after p.test</p>').insertAfter($('p.test'));


});
```

Для вставки html-кода до выбранных элементов можно использовать функции – .before() и .insertBefore()

```
$(function () {

    $('p.test').before('<p>before p.test</p>');

    $('<p>before p.test</p>').insertBefore($('p.test'));


});
```

**Задание №13**

С помощью функций .show()  .hide() можно мгновенно или плавно (за счет изменения размера и прозрачности) показывать и скрывать выбранные элементы на странице. Вызов функций без аргументов мгновенно показывает/скрывает выбранные элементы, установив их css-свойство display в none, не изменяя при этом их прозрачность и размеры.

Если передать 1 параметр как число duration (продолжительность выполнения анимации в миллисекундах) появление/скрытие будет происходить плавно.

Если передать 2 параметр функцию, она будет вызвана по окончании анимации

```
$(function () {

    $('.fast').show();
    $('.slow').show(3000, function(){
        alert('Анимация закончилась');
    });

});
```

Также функцию можно вызвать, используя первый аргумент как объект с опциями

```
$(function () {

    var options = {
        //опции
    };

    $('slow').show(options);

});
```

Все возможные опции можно посмотреть в оф. документации (http://api.jquery.com/show/).

**Задание №14**

На странице находится 5 скрытых блоков. Написать скрипт последовательного появления этих блоков начиная с первого, а затем последовательного скрытия в обратном порядке (пример в виде гиф изображения находится в папке в папке с заданием).

**Задание №15**

Метод .attr() возвращает или изменяет значение атрибутов у выбранных элементов страницы. Если вызвать метод с одним параметром строкой, то он возвращает значение атрибута переданного в качестве параметра у выбранного элемента. Если выбрано несколько элементов, то значение будет взято у первого.

```
$(function () {

   console.log($('.test').attr('title'));

});
```

Если передать 2 параметра атрибуту, который передан в качестве первого параметра будет присвоено значение, переданное в качестве второго параметра, у всех выбранных элементов.

```
$(function () {

   $('.test').attr('title', 'new title');

});
```

Если вызвать метод с одним параметром объектом, можно установить значения сразу множества атрибутов

```
$(function () {

   $('.test').attr({
       'title': 'new title',
       'lang' : 'en'
   });

});
```

Для удаления атрибута необходимо использовать метод .removeAttr() которому надо передать имя атрибута. Атрибут будет удален у всех элементов в выборке.

**Задание №16**

Используя JQuery

1. Найдите все ссылки без атрибута href и добавьте им этот атрибут со значением ‘#top’.
2. Найдите все картинки без атрибута alt и добавьте им этот атрибут со значением ‘image’.
3. Активируйте все отключенные кнопки.

**Задание №17**

Несмотря на то что классы являются атрибутами, для работы с классами не стоит использовать метод .attr(), для этого есть более удобные методы:

1. .addClass(). Добавляет класс всем выбранным элементам.

```
$(function () {
   $('div').addClass('test');
});
```

2. .removeClass(). Убирает класс у выбранных элементов.

```
$(function () {
   $('div').removeClass('test');
});
```

3. .hasClass(). Проверяет наличие класса у элементов страницы. Проверяет наличие класса className у выбранных элементов страницы. Если хотя бы один из элементов содержит этот класс, то функция вернет true, иначе вернет false.

```
$(function () {
   console.log($('div').hasClass('test'));
});
```

**Задание №18**

Метод .each() выполняет заданную функцию для каждого из выбранных элементов в отдельности. Это дает возможность обрабатывать выбранные элементы отдельно друг от друга.

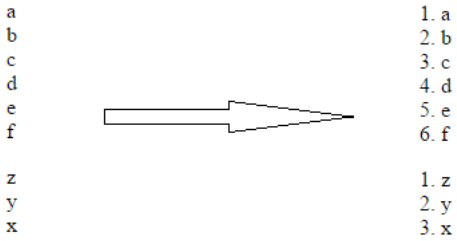
Метод принимает в качестве параметра функцию, которая будет вызвана для каждого элемента. В функцию будет передано 2 параметра: 1 – индекс элемента в выборке, 2 – сам элемент. Также внутри функции элемент будет доступен в ссылке this.

```
$(function () {

   $('div').each(function(index, element){

        console.log('div №' + index + ' html code: ' + $(element).html());

        //тут element и this совпадают
        console.log($(element));
        console.log($(this));

        function f(){
            //тут element и this отличаются
            console.log($(element));
            console.log($(this));
        }

        f();


   });
});
```

**Задание №19**

Используя JQuery

1. Выведите алертом содержимое всех элементов списков (по очереди)
2. Напишите скрипт, который проставит номер элемента в списке в начале элемента




3. Найдите все картинки без атрибута alt и добавьте им этот атрибут со значением равным имени картинки <img src="/tasks/task-19/img/nature1.jpg"> -> <img src="/tasks/task-19/img/nature1.jpg" alt="nature1.jpg">

**Задание №20**

В HTML5 для любого элемента можно использовать собственные атрибуты, начинающиеся с префикса data-. Это позволяет хранить разную информацию, которая может помочь в работе скриптов.

Для работы с этими атрибутами в JQuery есть отдельный метод .data()

Вызов метода без параметров .data() возвращает объект со всеми переменными, прикрепленными к первому из выбранных элементов.

```
$(function () {

   console.log($('.product').data());

});
```

Вызов с одним параметром строкой .data() возвращает значение переменной у первого выбранного элемента.

```
$(function () {

   console.log($('.product').data('type'));
   console.log($('.product').data('productId'));

});
```

Вызов с одним параметром объектом .data() устанавливает группу переменных всем выбранным элементам страницы.

```
$(function () {

   $('.product').data({'key': 5, 'key2': 6});
   console.log($('.product').data());

});
```

Вызов с двумя параметрами строками устанавливает переменную в первой строке со значением во второй строке всем выбранным элементам страницы.

```
$(function () {

   $('.product').data('key', 5);
   $('.product').data('type', 'product2');
   console.log($('.product').data());

});
```

Заметьте, что значение атрибутов в исходном коде страницы не меняется.

Важно отметить, что используя метод .data() для получения значения, вы получите значение переменных только у первого элемента из всех выбранных. Если вам нужны значения всех элементов, то нужно использовать метод .each().

С использованием этого метода можно сохранить не только строку или число, но и любой объект JS.

```
$(function () {

   $('.product').data('test', {'1':1, 'test': 2});
   console.log($('.product').data());

});
```

Метод data предоставляет массу возможностей, для передачи данных, связанные с различными элементами в JS, без использования глобальных переменных.