БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ

Кафедра «Автоматизированные системы управления»

ТЕХНОЛОГИЯ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ

Методические указания к лабораторной работе № 12 для студентов специальности
09.03.01 Автоматизированные системы обработки информации и управления

Могилев 2020

Лабораторная работа №12
JQuery основные события

Цель работы: Изучить основные события JQuery

Порядок выполнения работы.
Изучить теоретические сведения.
Выполнить задание к лабораторной работе в соответствии с вариантом.
Оформить отчет.

Требования к отчету.
Цель работы.
Постановка задачи.
Текст программы.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

.text()

Возвращает или изменяет текстовое содержимое выбранных элементов страницы. Функция имеет три варианта использования:

.text():stringv:1.0

возвращает текст содержащийся в выбранном элементе. Если таких элементов несколько, метод возвратит строку, в которой будет содержимое всех элементов, расположенное через пробел.

.text(newText):jQueryv:1.0

заменяет все содержимое у выбранных элементов, на текст newText.

.text(function(index, value)):jQueryv:1.4

заменяет все содержимое у выбранных элементов на возвращенный пользовательской функцией текст. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, value — текущий текст элемента.

Замечание: 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 (см. результат в разделе "В действии"). Для вставки html-элементов нужно использовать метод .html().

Примеры использования:

$(".topBlock").text() вернет текстовое содержимое всех элементов с классом topBlock (одной строкой).

$(".topBlock").text("Новье!") заменит содержимое всех элементов с классом topBlock на текст "Новье!".

В действии

поместим в элемент с классом demo-container текст:

```
<!DOCTYPE html>
<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
<div class="demo-container">
  <div class="demo-box">Контейнер для демонстраций</div>
  <ul>
    <li>Первый</li>
    <li>Второй</li>
  </ul>
</div>
<script>
  $('div.demo-container').text('<p> А вот и текст! </p>');
</script>
</body>
</html>
```

Методы Hide и Show

Как вы наверное не раз замечали на различных сайтах, большинство javascript-эффектов, основываются на отображении и скрытии различных элементов страницы. Примеров этому огромное количество: тултипы. выпадающее меню, слайд-шоу и многое другое. В сегодняшней статье, вы освоите применение базовых методов .hide() и .show(), а также узнаете как построить простейшую анимацию.

Базовые методы .hide() и .show(), без каких-либо параметров, можно рассматривать как сокращенный вариант метода .css('display', 'string'), где string это соответствующее значение свойства display. Эффект от применения этих методов очевиден – выбранный набор элементов, немедленно спрячется или отобразится, без всякой анимации.

Метод .hide() устанавливает линейный атрибут стиля, выбранному набору элементов, в значение display: none. Сложная часть здесь в том, что он запоминает значение свойства display – обычно block или inline – прежде чем изменит его на значение none. В свою очередь, метод .show() восстанавливает выбранному набору элементов, любое видимое свойство display, которое у них было установлено, ДО применения display: none.

Эта функциональность методов .show() и .hide(), особенно полезна, когда свойство display скрытых элементов, переопределено в CSS-файле. Например, элемент li, по умолчанию имеет свойство display: block, но мы можем захотеть изменить его на display: inline, для создания горизонтального меню. К счастью,

использование метода .show() для скрытых элементов, таких как тэги li, не установит их дефолтное свойство display: block, потому что это сломает все горизонтальное меню, и перенесет каждый элемент li на отдельную строчку. Вместо этого, элементу будет восстановлено его прежнее, назначенное в CSS-файле, значение display: inline, и меню сохранится в прежнем виде.

На основе этих двух методов, можно построить простейшее выпадающее меню на JQuery. Итак, для начала нам понадобится вот такая html-разметка:

Пример:

```
<div id="menu">
        <ul>
          <li>
            <a href="#">первый пункт</a>
            <ul>
              <li><a href="#">пункт 1</a></li>
              <li><a href="#">пункт 2</a></li>
              <li><a href="#">пункт 3</a></li>
            </ul>
          </li>
          <li>
            <a href="#">второй пункт</a>
             <ul>
              <li><a href="#">пункт 1</a></li>
              <li><a href="#">пункт 2</a></li>
              <li><a href="#">пункт 3</a></li>
            </ul>
          </li>
          <li>
            <a href="#">третий пункт</a>
             <ul>
              <li><a href="#">пункт 1</a></li>
              <li><a href="#">пункт 2</a></li>
              <li><a href="#">пункт 3</a></li>
            </ul>
          </li>
          <li>
            <a href="#">четвертый пункт</a>
            <ul>
              <li><a href="#">пункт 1</a></li>
              <li><a href="#">пункт 2</a></li>
              <li><a href="#">пункт 3</a></li>
            </ul>
          </li>
        </ul>
```

```
        </div>
```

Далее добавим немного CSS-стилей, чтобы наше меню выглядело как самое настоящее.

```
#menu {
    margin: 20px 0;
    clear: both;
    font-size: 12px;
}

ul {
    list-style: none;
    display: block;
}

ul li {
    list-style: none;
    float: left;
    display: block;
}

ul li a {
    text-decoration: none;
    color: #fff;
    background: #055788 url(menu.png) repeat-x;
    padding: 7px;
    border: 1px #2E5C09 solid;
}

ul li ul{
    display: none;
    float: none;
    margin: 5px 0 0 0;
    padding: 0;
}

ul li ul li{
    display: block;
    float: none;
}

ul li ul li a{
    background:  #000;
    display: block;
    padding: 7px;
    border: 1px #2E5C09 solid;
```

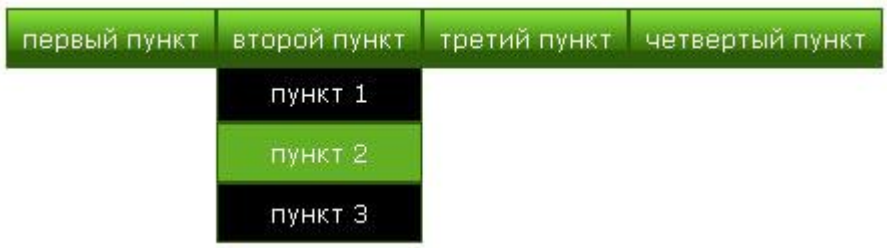
```
}

ul li ul li a:hover{
   display: block;
   background: #63B024;
}
```

Все основные приготовления сделаны, теперь перейдем непосредственно к коду, в котором нет, ничего сложного:

```
$(document).ready(function(){
        $("#menu ul li").hover(function(){
           $(this).find('ul').show();
        }, function(){
           $(this).find('ul').hide();
        })
     })
```

Как видно, из вышеприведенного примера, мы воспользовались событием .hover(), которое, как я уже писала ранее, включает в себя два обработчика событий: наведение мыши на элемент, и отведение мыши от элемента. Когда пользователь подводит мышь, отображается выпадающее меню, как только мышь перемещается в другое место, выпадающее меню снова исчезает.

Вот так просто, и без затей, мы создали простое выпадающее меню:



Как видите, использовать методы .hide() и .show() довольно быстро и удобно, но они кратковременные и не очень красивые. Для того чтобы добавить им изюминку, мы можем применить к ним скорость.

Эффекты и Скорость

Когда мы применяем скорость к методам .show() или .hide(), они становятся анимированными – в течении некоторого периода времени. Например, метод .hide('speed'), будет уменьшать высоту элемента, ширину и прозрачность, до тех пор, пока все три значения не станут равными 0, то есть пока не вступит в силу CSS-правило display: none. Метод .show('speed'), будет увеличивать высоту элемента, сверху вниз, ширину – слева направо, и прозрачность от 0 до 1, пока все

содержимое элемента не станет полностью видимым.

Использование скорости

С любым эффектом JQuery, мы можем использовать одну из трех скоростей: slow, normal и fast. Использование .show(‘slow’) выполнит эффект появления в течение 0.6 секунд, .show(‘normal’) за 0.4 секунды, и show(‘fast’) за 0.2 секунды. Для еще большей точности, мы можем указать количество миллисекунд, например .show(850). В отличие от имен скоростей, числовые значения можно не заключать в кавычки. Давайте включим скорость в наш пример, и вы сразу увидите, насколько изящнее стало появляться выпадающее меню:

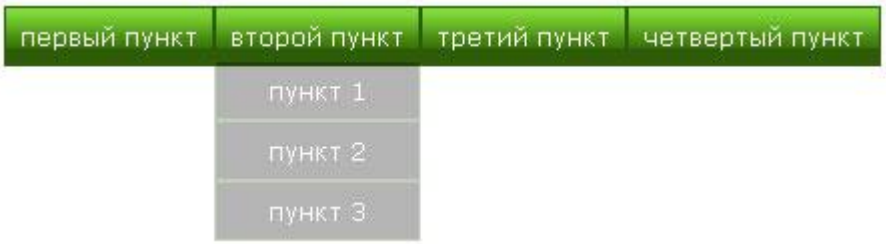
```
$(document).ready(function(){
        $("#menu ul li").hover(function(){
            $(this).find('ul').show('slow');
        }, function(){
            $(this).find('ul').hide();
        })
    })
```

Использование затухания

Если мы хотим, чтобы наше выпадающее меню появлялось только путем постепенного изменения прозрачности, мы можем использовать метод .fadeIn(‘slow’):

```
$(document).ready(function(){
        $("#menu ul li").hover(function(){
    $(this).find('ul').fadeIn('slow');
            $(this).find('ul').show();
        }, function(){
            $(this).find('ul').hide();
        })
    })
```

Теперь выпадающее меню будет появляться следующим образом:



Attr:

Имя	*Тип*
attr(имя)	***Возвращает: Объект***

Получение доступа к свойству первого совпавшего элемента.

Используя этот метод можно легко получить значение свойства первого совпавшего элемента. Если элемент не имеет указанного атрибута, то возвращается undefined (не определено). Атрибутами могут быть: title, alt, src, href, width, style и т.д.

attr(свойства) ***Возвращает: jQuery***

Устанавливает атрибуты всех элементов набора, используя при этом объект, который содержит пары ключ/значение.

attr(ключ, значение) ***Возвращает: jQuery***

Изменяет значение единственного свойства для каждого совпавшего элемента.

attr(ключ, функция) ***Возвращает: jQuery***

Изменяет значение единственного свойства для каждого совпавшего элемента.

Но вместо указания непосредственно значения указывается функция, которая возвращает значение как свой результат.

removeAttr(имя) ***Возвращает: jQuery***

Удаляет указанный атрибут из каждого совпавшего элемента.

attr(name) - обеспечивает доступ к значению указанного атрибута первого элемента в наборе.

Пример:

```
var a=$("i").attr("title");
$("div").text(a);
```

Данная инструкция найдет первый элемент в тегах i, найдет атрибут title этого элемента и добавит его значение в div.

Метод attr(name) получает значение заданного атрибута соответствующего

элемента набора jQuery, либо первого элемента в наборе jQuery (если их несколько). Возвращает значение undefined, если у элемента указанный атрибут отсутствует или в наборе нет элементов.

Возвращаемое значение: (строка) Значение искомого атрибута или undefined.

Параметры:

name – (строка) Имя атрибута, значение которого необходимо получить.

Примечание:

Чтобы получить значения атрибутов для каждого элемента в наборе jQuery, можно использовать методы .each() или .map().

Примечание:

Можно использовать собственные нестандартные имена атрибутов.

Примечание:

Библиотека jQuery предоставляет дополнительные нормализованные имена атрибутов для кроссбраузерной работы команды .attr(): class (от ClassName), float, cssFloat, styleFloat, for (от HtmlFor), maxlength, readonly.

Примечание:

Удалить атрибут средствами jQuery можно с помощью функции .removeAttr().

Примеры:

```
// Получить значение атрибута alt первого изображения на странице
// и, используя его, установить значение атрибута title.

var title = $("img").attr("alt") + " -> Увеличить";
$("img:first").attr("title", title);

// Сохранить в массив значения атрибутов id
// всех элементов <div> в документе.

var arr = new Array();
$("div").each(function(){
   arr[] = $(this).attr("id");
});
```

attr(properties) - установит атрибуты во всех отобранных элементах.

Пример:

```
$("img").attr({src:"images/pict.gif", alt:"рисунок"});
```

Данная инструкция найдет все картинки и установит им соответствующие атрибуты.

attr(key,value) - установит значение (value) атрибута (key) для всех отобранных элементов.

Пример:

```
$("button").attr("disabled", "disabled");
```

Данная инструкция установит для всех кнопок значение "disabled" атрибута "disabled".

removeAttr(name) - удалит указанный атрибут в всех элементов.

Пример:

```
$("img").removeAttr("alt");
```

Данная инструкция удалит атрибут "alt" у всех картинок.

Animate

Запомните – все эффекты анимации в jQuery крутятся вокруг функции animate – данная функция берет один или несколько CSS свойств элемента и изменяет их исходного до конечного за N-ое количество итераций (количество итераций зависит от указанного времени, но не реже одной итерации в 13ms (если я правильно накопал это значение)).

Функция **animate** понимает следующие параметры:
params – описание CSS свойств элемента, до которых будет происходить анимация (т.е. есть у нас div с высотой 100px – говорим animate({height:200}) – и высота плавно изменяется до 200px)
duration – скорость анимации – указываем в миллисекундах, или используя ключевые слова “fast” = 200ms, “normal” = 400ms или “slow” = 600ms
easing – указываем какую функцию будем использовать для наращивания значений, на выбор “linear” или “swing” (хотите больше см. Easing Plugin)
callback – функция, которая будет вызвана после окончания анимации

Альтернативный способ инициализации:

params – описание CSS свойств элемента, до которых будет происходить анимация

options – объект настроек:

duration – см. выше

easing – см. выше

complete – аналогичен ранее описанному callback-параметру

step – еще одна callback функция – отвечает за пошаговое изменение параметров – пример ниже

queue – флаг очереди, если выставить в false – то данная анимация будет игнорировать очередь и запустится сразу

Если заглянуть в руководство пользователя – то в разделе эффектов можно найти еще несколько вспомогательных функций:

show( ) – отображает выбранные элементы

hide( ) – скрывает выбранные элементы

toggle( ) – переключатель между show/hide

slideDown(speed, callback) – выдвигает объект(ы) вниз – увеличивает высоту от 0 до 100%

slideUp(speed, callback) – задвигает объект(ы) вверх – уменьшает высоту от 100% до 0

slideToggle(speed, callback) – переключатель между slideDown/slideUp

fadeIn(speed, callback) – отображает выбранные элементы – изменяет прозрачность элементов

fadeOut(speed, callback) – скрывает выбранные элементы – изменяет прозрачность элементов

fadeTo(speed, opacity, callback)- изменяет прозрачность элементов до указанного значения

Самые простые методы hide и show обходятся без функции animate, т.к. манипулируют лишь атрибутом display (демо):

Пример:

```
// вызов метода
$('#my').hide();

// аналогичен
$('#my').css({display:"none"});

// но если задать скорость анимации либо callback функцию,
// то будут изменятся значения height и width
```

Как я и говорилось ранее – остальные вспомогательные функции лишь обертки над animate, приведу пример (демо):

Пример:

```
// вызов метода
$('#my').slideUp();

// аналогичен
$('#my').animate({height:0,padding:0}, function(){
   $(this).css({display:"none"});
});

// вызов метода
$('#my').fadeOut();

// аналогичен
$('#my').animate({opacity:0}, function(){
   $(this).css({display:"none"});
});
```

В действительности, реализация данных методов чуть более сложная, но сути это не меняет

Стоит так же обратить внимание на способ задания параметров CSS для animate:

```
// установит прозрачность элемента в ноль, прозрачность изменяется от 0 до 1
$('#my').animate({opacity:0});

// наращиваем высоту элемента на 200px
$('#my').animate({height:'+=200px'});

// уменьшаем ширину элемента на 50px
$('#my').animate({width:'-=50px'});

// наращиваем высоту элемента до 20in
$('#my').animate({height:'20in'});
```

Append

append(контент)
Добавляет контент внутрь каждого элемента набора. Добавляемый контент следует за уже существующим.
Данный метод подобен применению appendChild.
Аргументы:контент Строка, Элемент, jQuery
Контент, который необходимо добавить.

Примеры:

Добавляет код HTML ко всем параграфам.

```
$("p").append("<strong>Hello</strong>");
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.js"></script>

  <script>

  $(document).ready(function(){
    $("p").append("<strong>Hello</strong>");
  });
  </script>
  <style>p { background:yellow; }</style>

</head>
<body>
  <p>I would like to say: </p>
</body>
</html>
```

Добавляет элемент ко всем параграфам.

```
$("p").append(document.createTextNode("Hello"));
```

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

<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.js"></script>

  <script>
  $(document).ready(function(){
    $("p").append(document.createTextNode("Hello"));
```

```
  });
  </script>

  <style>p { background:yellow; }</style>
</head>
<body>
  <p>I would like to say: </p>
</body>
</html>
```

Добавляет объект jQuery (аналогично массиву элементов DOM) ко всем параграфам.

```
$("p").append( $("strong") );


<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <script src="http://code.jquery.com/jquery-latest.js"></script>

  <script>

  $(document).ready(function(){
    $("p").append( $("strong") );
  });
  </script>
  <style>p { background:yellow; }</style>
</head>
<body>

  <strong>Hello world!!!</strong><p>I would like to say: </p>
</body>
</html>
```

Цепные функции

Те же действия можно применить в одну строчку. С помощью, так называемых «цепных функций».

```
$(document).ready(function(){
   $('#example_id').hide(2000).show(2000);
});
```

Это так же будет работать, и очень часто практикуется при написании на jquery.

Автоматические циклы

Давайте задумаемся, все замечательно работает с одним элементом, однако что делать, если нужно, допустим, скрыть сразу несколько элементов с одним id.

Разработчики jQuery так же об это позаботились, и не нужно писать никаких циклов, для этого просто делаем выборку по нужному id и применяем функции. Все элементы исчезнут.

Изменение ширины/высоты элементов

Иногда требуется узнать или изменить ширину/высоту. Чтобы получить значение, используем следующие функции:

```
$(document).ready(function(){
  var wExample = $('#example_id').width();
  var hExample = $('#example_id').height();
});
```

В данном примере соответствующие переменные получат значение ширины и высоты элемента.

Для того, чтобы изменить их, передаем в функции нужные параметры:

```
$(document).ready(function(){
  $('#example_id').width(200);
  $('#example_id').height(300);
});
```

Или же делаем это с применением «цепных функций»:

```
$(document).ready(function(){
  $('#example_id').width(200).height(300);
});
```

Изменение HTML

Функция text() позволяет получить/изменить текст из выбранного элемента. Однако, если требуется получить/изменить HTML код элемента, существует функция html()

Допустим, у вас есть абзац:

```
<p><strong>Это жирный абзац</strong></p>
```

Применяя метод text() мы получим лишь «Это жирный абзац». Если применить html()

```
$(document).ready(function(){
  $('p').html();
});
```

в выборку попадёт " Это жирный абзац ".
Аналогично методу text(), метод html() позволяет изменить код.

Плавное исчезновение/появление

Функции show() и hide() позволяют прятать и показывать элементы, но без какой- либо анимации. Функции fadeOut() и fadeIn() позволяют прятать и показывать элементы, но уже с плавной анимацией. Функции принимаю 2 параметра: время исчезновения/появления и функция, которая будет выполнена после.

```
$(document).ready(function(){
  $('img').fadeOut(1000).fadeIn(1000);
});
```

в данном примере изображение в выборке исчезнет плавно за 1 секунду, затем на следующем шаге плавно появится за 1 секунду.

Функция fadeTo() позволяет настроить степень прозрачности выбранного элемента. Принимает 3 параметра: время исчезновения, степень исчезновения и функция, которая будет выполнена после. Степень исчезновения варьируется от 0–1

```
$(document).ready(function(){
  $('img').fadeTo(1000,0.3).fadeTo(1000,1);
});
```

В данном примере изображение станет прозрачным до 0.3 за 1 секунду, а затем появится за 1 секунду.

Функции slideUp() и slideDown() позволяют плавно исчезнуть элементу(снизу вверх) и плавное появится(сверху вниз) соответственно.

```
$(document).ready(function(){
  $('img').slideUp(1000).slideDown(1000);
});
```

Работа с атрибутами элементов

Очень часто возникают задачи, связанные с работой над атрибутами

элемента. Допустим у нас есть изображение:

<img src="http://адрес" height="100" width="200" alt="изображение 1">

При помощи функции attr возможно получить доступ к значениям конкретных атрибутов, и даже изменить их. При помощи функции removeAttr() возможно удаление конкретных атрибутов

```
$(document).ready(function(){
  var imgAdress = $('img').attr('src'); // переменная будет содержать адрес изображения
  var imgHeight = $('img').attr('height'); // переменная будет содержать ширину изображения
  $('img').attr('width', '400'); // изменим значение атрибута width на 400
  $('img').removeAttr('alt'); // удалим атрибут alt у изображения
});
```

Добавление/удаление класса у элемента

У любого сайта существуют CSS стили для конкретных элементов. Чтобы добавить/удалить класс к выбранному элементу в jQuery есть функции addClass() и removeClass()

Представим, что вам нужно задать стиль для вновь созданного абзаца

```
<p id="main">Простой абзац</p>
```

Также, у вас в CSS прописан следующий класс:

```
.tagText
{
font-family: arial;
margin-right: 20pt;
color:#ffffff
}
```

Для того, чтобы применить этот стиль к заданному абзацу, необходимо сделать следующее:

```
$(document).ready(function(){
  $('#main').addClass('tagText');
});
```

или же удалить класс, если он не нужен

```
$(document).ready(function(){
  $('#main').removeClass('tagText');
});
```

Работа с CSS

Функции, описанные выше, позволяют добавлять стили из CSS при помощи классов. Однако, функция css() позволяет получить доступ непосредственно к таблицам CSS

```
.tagText
{
font-family: arial;
margin-right: 20pt;
color:#ffffff
}
```

для того, чтобы узнать/изменить шрифт, воспользуемся функцией css()

```
$(document).ready(function(){
  var textFont = $('#main').css('font-family');
});
```

Переменная textFont получит значение “arial”.

Для того, чтобы изменить какое-либо значение, допустим, значение цвета, необходимо через запятую после названия атрибута указать значение:

```
$(document).ready(function(){
  $('#main').css('color', '#ff00ff');
});
```

Если же необходимо изменить 2 атрибута, допустим цвет и шрифт, то тут возможны варианты. Можно применить «цепные функции»

```
$(document).ready(function(){
  $('#main').css('color', '#ff00ff').css('font-family');
});
```

Так же, возможно это делать в виде, более похожим на традиционный CSS:

```
$(document).ready(function(){
  $('#main').css({
    'color' : '#ff00ff',
    'font-family' : 'verdana'
  });
});
```

Обратите внимание, что в этом случае имя атрибута отделяется от значения, которое ему присваивается двоеточием!

Аналогичные действия возможно совершать за определенный промежуток

времени, для этого следует использовать функцию animate()

```
$(document).ready(function(){
  $('#main').animate({
    'marginRight':'10px'
  },5000);
});
```

за 5 секунд отступ справа уменьшится до 10 пикселей.

Добавление контента

В jQuery существуют функции, позволяющие очень гибко добавлять контент. Предположим, у вас в DOM дереве присутствует картинка:

```
...
<img id="simple" src="http://адрес">
...
```

Для того, чтобы добавить контент ДО выбранного элемента, следует пользоваться функцией before()

```
$(document).ready(function(){
  $('#simple').before('<p>Абзац, добавленный до картинки, с помощью before()');
});
```

Для того, чтобы добавить контент ПОСЛЕ выбранного элемента, следует пользоваться функцией after()

```
$(document).ready(function(){
  $('#simple').after('<p>Абзац, добавленный после картинки, с помощью after()');
});
```

Перебор в цикле

Функция .each() производит обход всех элементов, содержащихся в наборе jQuery и вызывает функцию обратного вызова callback для каждого из них.

callback([index, Element]) – (функция) Функция, вызываемая для каждого элемента в наборе jQuery. С каждой итерацией в качестве первого параметра index ей передается индекс текущего элемента в наборе Query (начиная с 0) как индекс текущей итерации цикла. Во втором аргументе Element передается ссылка на сам DOM элемент. Контекст this вызова функции также каждый раз ссылается на текущий элемент, задействованный в данной итерации (Element == this).

```
// Выводить в цикле содержимое пунктов списка до тех пор,
// пока не попадется <li> с классом 'stop'.
```

```
$('li').each(function(i,elem) {
    if ($(this).is(".stop")) {
        alert("Остановлено на " + i + "-м пункте списка.");
        return false;
         } else {
               alert(i + ': ' + $(elem).text());
              });
```

Количество элементов в выборке

С помощью функции size() возможно узнать количество элементов, которое попало в выборку. Она возвращает число.

```
$(document).ready(function(){
  $('img').size();
});
```

в выборке находится количество картинок на странице

Доступ к конкретному элементу

С помощью функции get() возможно получить доступ к конкретному номеру элемента в выборке.

Внимание! данная функция возвращает не jQuery объект, а просто javascript объект. Будьте внимательны!

Клонирование элемента

Иногда возникают ситуации, когда необходимо клонировать элемент. Для этого следует использовать функцию clone()

Допустим, в DOM дереве присутствует картинка, и вам необходимо её отобразить в другой части экрана.

```
$(document).ready(function(){
  var Image = $('img').clone();
});
```

Запомним элемент в переменную, затем выведем в нужное нам место с помощью before() или after(). Для этого нужно просто передать эту переменную как параметр в скобках.

Выбор разнотипных элементов

Иногда необходимо произвести определенный действия (например, скрыть) разные элементы. Для этого укажем это в выборке через запятую, в ковычках

Пример:

```
$(document).ready(function(){
  var s = $('img, p').size(); // выбор всех изображений и абзацев
  s.hide(); // скрытие
});
```

Задания к лабораторной работе

Вариант 1

У нас есть пример в котором мы отбираем из всего дерева картинку из шапки сайта logo.jpg

и сделали мы это таким способом: $('img[src*=logo.jpg]').hide();

Сделать выборку этого же элемента двумя другими способами на Ваше усмотрение.

Пример:

```
$(document).ready(function(){
var texH1 = $('#main_h1').text('текст измененный с помощью jquery');
var myLogo = $('img[src*=logo.jpg]').hide();
myLogo.hide(3000);
myLogo.show(3000);
});
```

Отобрать в jquery набор все картинки кроме первой и скрыть их с помощью функции hide() за 5 секунд. Подсказка: для формирования набора используйте фильтр not.

Напишите функцию, которая получает 3 параметра.

id элемента, с которым надо работать.

Название атрибута, который нужно изменить.

Новое значение указанного атрибута.

Функция в результате своей работы должна изменить указанный атрибут указанного элемента.

Сделайте выборку для картинки второго мотоцикла moto2.jpg.

С помощью метода css, добавьте ей рамку 1px solid #333333.

Примените анимацию, во время которой увеличьте рамку до значения в 5px за 5 секунд.

После выполнения анимации плавно скройте шапку с помощью функции fadeOut() за 5 секунд.

Используйте цепные функции.

Подсказка: свойство для увеличения размера рамки в обычном css записывается как border-width.

Ваша задача добавить в самый конец тэга <body> новый абзац с любым текстом

и дать этому абзацу id="newparagraph".

После этого изменить цвет фона этого параграфа на черный, а цвет текста на белый.

Используйте цепные функции.