**Web-дизайн**

Лабораторная работа №12

**Введение в jQuery**

**Подключение библиотеки jQuery**

Чтобы начать работать с данной библиотекой, нам первым делом надо ее загрузить. Ее найти можно на официальном сайте разработчика:

<https://jquery.com/download/>

Библиотека представлена в двух вариантах – *Compressed* (минимизированная) и *Uncompressed* (обычный).

Минимизированные версии предоставляют ту же функциональность, что и обычные, но отличаются тем, что не содержат всяких необязательных символов, наподобие пробелов, комментариев и т.д., и поэтому в своем названии имею суффикс *min*.

Библиотека *jQuery* подключается также, как и другие файлы *javascript*. Например:

<head>

<script src="jquery-3.4.1.js"></script>

</head>

Пример страницы:

<html>

<head>

<title>Мир jQuery</title>

<script src="jquery-3.4.1.js"></script>

<meta charset="utf-8">

</head>

<body>

<h2>Добро пожаловать в мир jQuery</h2>

<button id="btn1">jQuery</button>

<button id="btn2" onclick="alert('Мир JavaScript'); ">

JavaScript</button>

<script type="text/javascript">

$(function(){

$("#btn1").click(function(){

$(this).css('background-color', 'red');

alert('Мир jQuery');

});

});

</script>

</body>

</html>

У нас определены на странице две кнопки. У одной кнопки определен обработчик *onclick* в самой разметке кнопки: onclick="alert('Мир JavaScript'); ". Другая же кнопка делает практически тоже самое, но только с *jQuery*.

**Использование сетей CDN**

Вместо загрузки библиотеки можно использовать сети CDN (Content Delivery Networks), то есть можно указать ссылку на библиотеку.

<script src="https://code.jquery.com/jquery-3.4.1.js"></script>

**Функция jQuery**

Функция jQuery имеет следующий синтаксис:

jQuery(document).ready(function(){

// здесь код функции.

});

В данном случае в качестве объекта используется объект *document*, и к нему применяется обработчик **ready**, который и сигнализирует о том, что DOM-модель веб-страницы загружена.

Есть также еще один способ объявления, который также равнозначен предыдущему:

$(document).ready(function(){

// здесь код функции.

});

**Знак $** представляет псевдоним jQuery. Сокращенная запись:

$(function(){

// здесь код функции.

});

**Выборка элементов**

Библиотека предоставляет нам удобный способ выбора элементов, основанный на *селекторах* (как в CSS):

$("селектор")

Пример:

<h2>Словарь</h2>

<table id="**tab**">

<tr class="tabhead">

<td><p>Слово</p></td> <td><p>Перевод</p></td>

</tr>

<tr>

<td><p>table</p></td> <td><p>таблица</p></td>

</tr>

</table>

<script>

jQuery(document).ready(function(){

$("#**tab**").css('background-color', 'silver');

});

</script>

Здесь мы используем селектор *#tab*, то есть выбираем нашу таблицу и устанавливаем цвет элемента. Но также можем использовать набор селекторов, чтобы более детально произвести действие. Например, нам надо, чтобы была закрашена только первая строка, которая имеет атрибут *class="tabhead"*. Тогда можно использовать селектор: $("#tab .tabhead").

К примеру, селектор $("div#menu a.redStyle") может найти такой элемент:

<div id="menu">

<a href="m.tml" class="redStyle">Ссылка на сайт</a>

</div>

В этом случае мы применили простой **иерархический селектор**:

внешний\_селектор внутренний\_селектор

Кроме этого можно использовать еще ряд других иерархических селекторов:

родительский\_селектор **>** дочерний\_селектор

Выборка всех элементов, которые соответствуют второму селектору и являются непосредственно дочерними для элементов первого селектора.

**селектор1 + селектор2**

Выборка элементов с селектором 'селектор2', которые располагаются непосредственно за элементами 'селектор1'. Причем оба элемента должны находиться на одном уровне. Например, у нас есть следующий блок ссылок:

<div id="menu">

<a href="1.html" class="open">1.html</a>

<a href="2.html" class="closed"> 2.html</a>

**<a href="3.html" class="open">3.html</a>**

</div>

Тогда селектор $(".closed + .open") выберет ту ссылку, которая выделена жирным, так как она с атрибутом *class="open"* и идет за элементом с атрибутом *class="closed"*.

**селектор1 ~ селектор2**

Выборка элементов с селектором 'селектор2', которые располагаются на одном уровне с элементами с селектором 'селектор1'. Например:

<div id="menu">

**<a href="1.html" class="open">1.html</a>**

<a href="2.html" class="closed"> 2.html</a>

</div>

<a href="3.html" class="open">3.html</a>

Тогда селектор $(".closed ~ .open") выберет ту ссылку, которая выделена жирным, так как элемент с атрибутом *class="open"* находится на одном уровне с элементом с *class="closed"*.

**Селекторы атрибутов**

Чтобы сузить поиск кроме базовых и иерархических селекторов можно применять *селекторы атрибутов*:

|  |  |  |
| --- | --- | --- |
| **Шаблон** | **Описание** | **Пример** |
| [attr] | Выборка всех элементов с атрибутом *attr* | Выражение $("a[class]") выберет все элементы ссылок, которые имеют атрибут *class* |
| [attr='value'] | Выборка всех элементов, у которых значение атрибута *attr* равно *value* | Выражение $("a[class='redStyle']")выберет все элементы ссылок, которые имеют атрибут *class="redStyle"* |
| [attr~='value'] | Выборка всех элементов, у которых атрибут *attr* имеет список значений, разделенных пробелом, и одно из значений – *value* | Выражение $("a[class~='redStyle']")выбрать элементы ссылок, которые могут иметь следующее значение атрибута  *class="redStyle* *closed"*. |

и т.д.

**Контекст селекторов**

Можно ограничить поиск область выборки, применив *контекст поиска*. Контекст представляет собой селектор, в рамках которого надо производить выборку элементов. Пример:

<div id="menu">

<a href="1.html" class="**open**"> Ссылка 1 </a>

</div>

<a href="2.html" class="**open**"> Ссылка 2 </a>

<script>

jQuery(document).ready(function(){

$(".open", **"div#menu"**).css('background-color', 'silver');

});

</script>

На странице две ссылки, у всех у них определен один и тот же класс, но одна из них находится в элементе *div*. Для ее получения используем выражение $(".open", "div#menu") – здесь *второй параметр* будет являться **контекстом выборки**.

**Фильтры jQuery**

В дополнение к селекторам применяются *фильтры*. Они позволяют отфильтровать найденные элементы по определенному принципу. По своей сути фильтры совпадают с *псевдоклассами CSS*. Например:

<table>

<tr> <td>Слово</td> <td>Перевод</td> </tr>

<tr> <td>Cabbage</td> <td>Капуста</td> </tr>

<tr> <td>Carrot</td> <td>Морковь</td> </tr>

</table>

<script>

jQuery(document).ready(function(){

$("tr**:even**").css('background-color', 'silver');

});

</script>

К примеру, можно выделить несколько из *базовых фильтров*:

|  |  |
| --- | --- |
| **Фильтр** | **Описание** |
| :eq(n) | Выбирает n-й элемент выборки (нумерация начинается с нуля). |
| :even | Выбирает элементы с четными номерами. |
| :first | Выбирает первый элемент выборки. |
| :not(селектор) | Выбирает все элементы, которые не соответствуют селектору, указанному в скобках. |

и т.д.

Мы можем комбинировать в одном выражении несколько селекторов и фильтров.

*Специальный набор фильтров* – фильтры контента обеспечивают доступ к элементам, имеющим определенное содержимое:

|  |  |
| --- | --- |
| **Фильтр** | **Описание** |
| :contains('text') | Получает все элементы, которые содержат текст text |
| :has('селектор') | Получает все элементы, которые содержат хотя бы один дочерний элемент, соответствующий селектору |
| :first-child | Получает все элементы, которые являются первыми дочерними элементами в своих родителях |
| :nth-child(n) | Получает все элементы, которые являются n-ными элементами в своих родителях (нумерация идет с единицы) |
| :only-child | Получает все элементы, которые являются единственными дочерними элементами в своих родителях |

и т.д.

Например, если мы хотим получить все строки, содержащие текст «Капуста», можно применить следующее выражение: $("tr:contains('Капуста')").

**Работа с выборкой**

**Свойство length и метод size**

Свойство *length* возвращает нам количество выбранных элементов:

var num = $("tr:nth-child(odd)").length;

Альтернативой является использование метода *size*, который также возвращает число выбранных элементов: var num = $("tr:nth-child(odd)").size();

**Получение элемента**

Мы можем получать элементы выборки по индексу:

var firstElement = $("tr:nth-child(odd)")[0];

**Функция eq**

Функция **eq(index)** в качестве параметра получает индекс элемента и возвращает новый набор, состоящий из одного элемента по данному индексу. Например:

$(function(){

var array = $("tr:even");

array.eq(1).css('background-color', 'silver');

array.eq(-1).css('background-color', 'silver');

});

При отрицательном значении аргумента отсчет будет производиться с конца.

**Получение первого и последнего элемента**

Для этого используются *last* и *first* соответственно.

$(function(){

var array = $("tr:even");

var firstEl = array.first(); // Первый элемент

var lastEl = array.last(); // Последний элемент

});

**Перебор элементов выборки**

Для перебора выборки мы можем использовать специальный метод **each**:

$(function(){

$("tr:even").each(function(index, elem){

console.log(index + ". " + elem.innerHTML);

});

});

В метод *each* в качестве параметра передается функция, которая принимает два параметра: *index* – индекс элемента в наборе и *elem* – сам элемент.

**Добавление новых элементов**

С помощью метода **add** мы можем создать новый набор. Варианты синтаксиса:

* *add(селектор)*: добавление элементов, соответствующих селектору, в набор
* *add(элемент)*: добавление в набор элемента
* *add(массив элементов)*: добавление массива элементов
* *add(jQuery)*: добавление объекта, который является результатом функции jQuery
* и т.д.

Например:

$(function(){

var array = $("tr:even").add("tr:odd:first");

//данное выражение эквивалентно следующему

// var array = $("tr:even, tr:odd:first");

});

**Фильтрация выборки**

**Метод filter**

Метод *filter(условие)*принимает в качестве параметра условие фильтрации. И если элементы выборки не соответствуют этому условию, то они исключаются из выборки. Варианты использования:

* *filter(селектор)*: если объекты выборки не удовлетворяют селектору, то они исключаются
* *filter(элемент)*: если объекты выборки не представляют данный элемент html, то они исключаются
* *filter(jQuery)*: если объекты выборки не представляют данный объект jQuery, то они исключаются
* *filter(функция)*: для каждого элемента выборки вызывается функция, и если эта функция возвращает false, то элемент исключается из выборки

$(function(){

//Фильтрация по селектору

var array = $(".header").**filter**("tr");

// Фильтрация по элементу html

var elem = document.getElementsByTagName("tr")[0];

var array1 = $(".header").**filter**(elem);

// Фильтрация функцией

var array2 = $("tr").**filter**(function(){

if($(this).hasClass("header")) { return true;}

else {return false;}

});

});

**Метод not**

Противоположным по действию по сравнению с методом *filter* является метод *not*. Он исключает из выборки те элементы, которые соответствуют условию. Варианты использования аналогичны *filter*.

**Метод slice**

Метод *slice(begin, end)* исключает из выборки те элементы, которые не попадают в диапазон, задаваемый параметрами *begin* и *end*. Подразумевается, что индексация начинается с 0. Негативные индексы означают, что выборка ведется с конца набора. Необязательный параметр *end* задает позицию, с которой выборка элементов уже прекращена, если этот параметр отсутствует, то выборка ведется до конца набора:

$(function(){

var array = $("tr").slice(1,4);

});

**Метод has**

Метод *has(вложенный элемент)* проверяет объект на наличие вложенного элемента. И если объект содержит вложенный элемент, то остается в выборке. Варианты вызова:

* *has(селектор):* если объекты содержат элемент, соответствующий селектору, то они остаются в выборке
* *has(элемент):* если объекты содержат данный элемент html, то они остаются в выборке

**Преобразование выборки**

Метод *map(callback)* предназначен для преобразования элементов выборки с помощью функции, которая передается в параметре *callback*. Пример:

<ul>

<li>Пункт 1</li>

<li>Пункт 2

<ul>

<li>Подпункт 2.1</li>

</ul>

</li>

<li>Пункт 3</li>

</ul>

Отберем только те элементы, которые имеют вложенные списки:

$(function(){

var array = $('li').**map**(function(index, elem){

return $(elem).children()[0];

});

array.each(function(index, elem){

console.log(elem.innerHTML);});

array.css('background-color', 'silver');

});

Так, в данном случае мы передаем перебираемый элемент в функцию jQuery, чтобы нам стали доступными методы jQuery.

**Метод is**

Метод *is(условие)* определяет, имеются ли в данной выборке объекты, соответствующие условию (возвращает *true* или *false*). Варианты использования аналогичны *filter* и *not*.

**Поиск в выборке**

Для поиска в выборке предназначен метод **find**. Варианты использования:

* *find(селектор)*: поиск в выборке объектов, соответствующих селектору
* *find(элемент)*: поиск в выборке объектов, соответствующих данному элементу html
* *find(jQuery)*: поиск в выборке объектов, соответствующих объекту jQuery

Например:

<ul class="menu">

<li>Пункт 1</li>

<li>Пункт 2

<ul class="submenu">

<li>Подпункт 2.1</li>

</ul>

</li>

<li>Пункт 3

<ul class="submenu">

<li>Подпункт 3.1</li>

</ul>

</li>

</ul>

Теперь последовательно осуществим поиск по селектору, элементу и объекту jQuery:

$(function(){

// поиск по селектору

var array0 = $('ul').**find**('.submenu');

// поиск по элементу

var elem = $('ul.submenu')[0]; // получаем первый элемент выборки

var array1= $('ul').**find**(elem);

// Поиск объекта jQuery

var jQueryObject = $('ul.submenu');

var array2= $('ul').**find**(jQueryObject);

});

**Навигация по странице**

**Получение дочерних элементов**

Для получения дочерних элементов объекта используется метод *children([селектор])*.

На основе предыдущего примера получим дочерние узлы:

// дочерние узлы без селектора

var lists = $('li').**children**('');

lists.each(function(index, elem){

console.log(elem.innerHTML);

});

//дочерние узлы по селектору

var lists2 = $('li').**children**('.submenu');

lists2.each(function(index, elem){

console.log(elem.innerHTML);

});

**Метод closest**

Метод *closest(условие)* возвращает ближайшего родителя по отношению к данному элементу. Варианты использования:

* *closest(селектор[, контекст])*: возвращает ближайшего родителя, соответствующего селектору. В качестве необязательного параметра может принимать контекст, в рамках которого будет проводиться поиск.
* *closest(элемент)*: возвращает ближайшего родителя, соответствующего элементу html
* *closest(jQuery)*: возвращает ближайшего родителя, соответствующего объекту jQuery

**Метод next**

Для получения следующего по порядку элемента, находящегося на одном уровне с текущим, используется метод *next([селектор])*.

Мы можем также получить все следующие элементы одного уровня по отношению к текущему, использовав метод *nextAll([селектор])*.

Кроме того, получить все следующие элементы одного уровня с текущим можно с помощью метода *nextUntil([селектор])*.

**Метод prev**

На группу методов *next / nextAll / nextUntill* похожа группа методов *prev / prevAll / prevUntil*, только получает предыдущие элементы по отношению к текущему.

**Получение всех элементов одного уровня (сиблингов)**

Метод **siblings([селектор])** представляет собой комбинацию методов *nextAll* и *prevAll* и позволяет получить всех сиблингов текущего элемента.

**Получение родителей**

Метод **parent([селектор])** возвращает непосредственных родителей для данного элемента.

Похожим действием обладает метод *parents([селектор])*. Только он возвращает не только непосредственных родителей, но и *вообще всех родителей*.

Самоучитель jQuery - <https://webref.ru/dev/jqfundamentals>

Самоучитель jQuery - <https://metanit.com/web/jquery/>

Справочник jQuery - <https://jquery-docs.ru/>

Справочник jQuery - <http://jquerybook.ru/api/>

**Общие задания**

Все задания необходимо выполнять с помощью **jQuery**.

* 1. Создайте два отдельных списка. Наполните их какими-либо данными (создайте несколько элементов списка). С помощью *jquery*-выборки выберите элементы первого списка и измените размер шрифта (к примеру, на *20px*). Выберите элементы второго списка и сделайте шрифт жирным (*bold*). Далее выберите все элементы обоих списков и поменяйте цвет заднего фона четных элементов на синий, а нечетных – на красный.
  2. Создайте блок <*div>* на странице. Добавьте несколько абзацев <*p*>. Разместите некоторые из абзацев внутри блока <div>.

Выберите только те абзацы, которые находятся в <*div*> и поменяйте им цвет заднего фона. Из них выберите все абзацы кроме первого. Для них установите цвет шрифта.

Выберите только абзацы, которые не находятся в <*div*>. Установите у них какой-нибудь размер шрифта. У центрального элемента среди них сделайте шрифт курсивом.

* 1. Создайте два больших блока <*div>.* В каждом должно быть по 5 маленьких блоков <*div>* на странице. Установите одинаковые размеры всех маленьких блоков и сделайте так, чтобы они располагались в одной строке (*display: inline-block*).

Напишите функцию, которая принимает номер блока, выбирает его и устанавливает у него границу. После этого выбирает все следующие после него блоки (на том же уровне иерархии) и установите цвет заднего фона. Тоже самое делает с блоками, идущими до текущего (но другой цвет). Продемонстрируйте работу на данных двух группах блоков, передавая функции разные значения аргумента.

* 1. Напишите функцию, которая принимает один элемент и проходится по всем элементам этого же уровня, начиная с полученного в качестве аргумента. Устанавливает для них цвет заднего фона.

Для принятого элемента цвет белый, но с каждым следующим элементом он должен становиться темнее. Также для предыдущих элементов. Для перебора элементов используйте метод *next*() и *prev*().

Продемонстрируйте работу функции на списке.