**Лекция. Библиотека jQuery**

**1. Введение в jQuery**

**jQuery** – библиотека JavaScript, которая позволяет получать доступ к любому элементу модели DOM, обращаться к атрибутам и содержимому, а также манипулировать ими. Кроме этого библиотека предоставляет удобный API для работы с технологией AJAX.

Библиотека была создана в 2006 году. В настоящее время практически каждый сайт использует данную библиотеку.

Подключив библиотеку jQuery вместо десятков команд на JavaScript можно написать несколько команд. Команды основаны на селекторах и классах CSS.

Библиотеку jQuery можно скачать с сайта [http://jquery.com/](http://aermolenko.ru/jexr/aHR0cDovL2pxdWVyeS5jb20v). Размер библиотеки в минимальном варианте составляет примерно 60 Кб.

Перед началом работы библиотеку надо подключить:

<script src=»jquery.js» type=»text/javascript»></script>

Вся работа с библиотекой ведется с использованием функции $. Общая идея использования: 1) выбирается элемент или группа элементов, 2) выполняются действия над выделенными элементами.

Пример. Выделение красным цветом каждой второй строки таблицы.

<script src=»jquery.js» type=»text/javascript»></script>

<style>

.rrr{color:red}

</style>

<body>

<table border=»1″>

<tr><td>1</td><td>1</td><td>1</td></tr>

<tr><td>1</td><td>1</td><td>1</td></tr>

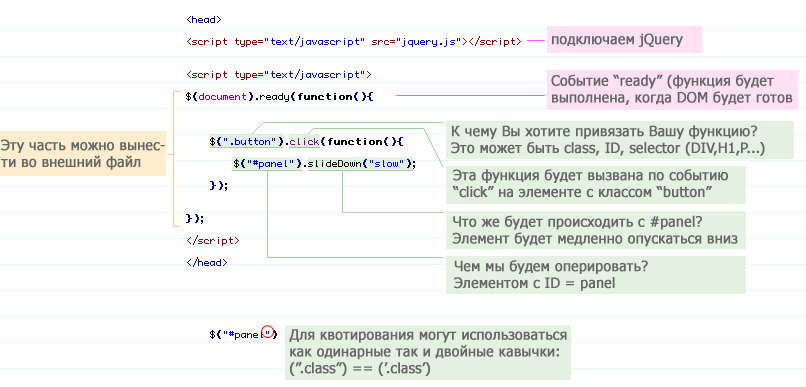
<tr><td>1</td><td>1</td><td>1</td></tr>

</table>

<script>

$(«table tr:even»).addClass(«rrr»)

</script>



Работу с jQuery можно разделить на 2 типа:

1. Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса.

2. Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

**2. Обращение к элементам**

Рассмотрим на примерах варианты обращения. Обращаю внимание, что все основано на синтаксисе CSS.

1. $(p a) выбирает все ссылки, расположенные в абзацах.

2. $(‘ #myId ‘) – выбор элемента с указанным идентификатором.

3. $(‘myClass’) – использование классов.

4. $(‘body>div’) – выбор элементов div, являющихся прямыми потомками элемента body.

5. $(p:even) – четные абзацы.

6. Использование ссылок.

* a[href^=http://] – этому селектору соответствуют все ссылки, которые начинаются с символов http://, на это указывает символ ^/
* a[href$=.pdf] – ссылки заканчиваются на «pdf»
* a[href\*=ya.ru] – ссылка содержит упоминание ya.ru в произвольном месте.

7. Имеется возможность выбора определенных элементов при условии, что они содержат другие элементы (в примере ниже двоеточие как в математике означает “такое что”). Например, селектор

li:has(a)

выбирает элементы <li>, которые содержат элемент <a>.

8.  Выбор по позиции (некоторые варианты)

:first – первое совпадение на странице;

:last – последнее совпадение на странице;

:first-child – первый дочерний элемент;

:last-child – последний дочерний элемент;

:nth-child(n) – n-й элемент;

:nth-child(even|odd) – четные или нечетные дочерние элементы;

:even – четные элементы;

:odd – нечетные элементы.

9. Выбор элементов на основе характеристик, не предусмотренных спецификацией CSS

:input – выбирает элементы формы;

:selected – выбранные элементы option;

:visible – выбор невидимых элементов.

**3. Обработчик готовности документа**

Первый способ – запуск обработчика, когда загрузилась страница

window.onload=function(){

$(«table tr:nth-child(even) «).addClass(«even»);

};

Недостаток данного метода заключается в том, что ничего не будет изменено, пока все не загрузится, включая картинки, т.е. пользователь может усмотреть как все изменяется, т.е. увидеть и исходный вариант, и преобразованный.

Второй способ – дождаться только загрузки структуры документа, а именно:

$(documen).ready(function(){

$(«table tr:nth-child(even) «).addClass(«even»);

});

Т.е. читаем буквально – “Запуск функции после загрузки”. У этой конструкции имеется сокращенная форма:

$(function(){

$(«table tr:nth-child(even) «).addClass(«even»);

});

**4. Создание элементов DOM**

Создание элемента

$(‘<div>Hi</div>’)

Пустой элемент можно создать так:

$(‘<div> ‘)

Можно использовать кавычки, можно апострофы. Но наличие тегов обязательно. Задавать просто текст нельзя. Вспомните (лекция 7), сколько было команд, чтобы создать веточку элементов без jquery. Сама по себе эта команда ничего не выведет, надо эту веточку опять же привязать к родительскому элементу. Делается это так:

$(‘<div>Hi</div>’).insertAfter(‘#a1’)

Здесь a1 – это идентификатор объекта, после которого необходимо вставить элемент.

**5. Работа с полученным набором значений**

В терминах jQuery эти наборы называют обернутыми.

8.5.1. Определение размера. Для этого можно использовать свойство length, а можно метод size().

Пример.  Замена содержимого с идентификатором a1 на количество элементов класса b2

$(«#a1»).html($(«.b2»).size())

8.5.2. Полученный набор значений рассматривается как массив, и с ним мы соответственно и работаем. Получить любой элемент в обернутом наборе можно по индексу. Например, получить второй элемент в наборе всех гиперссылок:

$(‘a’)[1].

Вместо индексов можно использовать метод get(), т.е. последнюю команду можно записать так:

$(‘a’).get(1).

При помощи метода get() можно получить обычный массив JavaScript, содержащий все обернутые элементы. Пример:

var allanchors=$(‘a’).get().

8.5.3. Существуют специальные команды библиотеки, которые позволяют объединять различные полученные наборы, убирать определенные значения по некоторым правилам

**6. Манипулирование объектами на странице**

Команды jQuery позволяют манипулировать свойствами, атрибутами, стилями и содержанием элементов.

Для обращения к свойствам и их значениям используются методы JavaScript, нет методов непосредственно библиотеки.

Пример. Изменяет свойство title у всех элементов класса b2

$(‘.b2’).each(function(n){this.title=»New «+n})

Здесь использована команда **each(функция).** Она выполняет обход всех элементов в наборе и вызывает для них функцию. В качестве параметра функции передается индекс элемента в наборе.

**7. Обработка событий**

8.7.1. Различные браузеры по-своему могут обрабатывать события, jQuery пытается сгладить эти неприятности. Поэтому обращаемся только к методам jQuery, а библиотека уже сама смотрит, что за браузер и применяет то или иное свойство.

* Модель событий jQuery обладает следующими свойствами:
* поддерживает единый метод установки событий;
* позволяет устанавливать несколько обработчиков для события;
* использует стандартные названия типов событий;
* предоставляет единые методы отмены события и блокирования действий по умолчанию.

8.7.2. Подключение обработчиков. Рассмотрим пример функции, которая будет срабатывать при щелчке по любому рисунку:

$(‘img’).blind(‘click’,fuction(event){alert(‘Приветствую!’);});

Для удаления обработчика используется команда unbind()

**8. Скрытие и отображение объектов**

8.1. Функции **hide**() и **show**(). С ними есть некоторые нюансы (например, show показывает изначальное состояние свойства display, поэтому при загрузке рекомендуют скрыть объект при помощи hide()), но в целом все как в примере.

Пример.     $(‘#b3’).hide()

А затем, скажем, по щелчку,         $(‘#b3’).show()

8.2. Имеются различные эффекты – раскрывающиеся списки, слайдеры, увеличение рисунков (когда-нибудь надо разобрать подробнее)

**9. Создания слайдера**

Рассмотрим пример создания слайдера – блока с контентом, выползающего по нажатию на конец закладки.

8.9.1. В текст вставляем следующий фрагмент:

<div id="container">  
<div id="panel">  
Содержимое панели  
</div>  
<p><a href="#">Справка</a></p>  
</div>

9.2. При помощи стилей изначально скрываем панель. Остальные параметры просто описательные – размер, цвет и т.д.

<!—Из-за наличия auto заодно и центрует в Mozila, т.к. auto  Указывает, что размер отступов будет автоматически рассчитан браузером.—>

<!—buttontext можно вообще не задавать—>

<style>

.rrr    {color:red}

#container {

margin: 0 auto;

width: 152px;

}

#panel {

background: #1ca8f6;

height: 230px;

display: none;

}

.button {

width: 152px;

height: 40px;

border-top: #333 dotted 1px;

text-align: center;

}

.buttontext {

font-weight: bold;

font-size: 1.2em;

text-shadow: 1px 1px 1px #666;

}

</style>

9.3. Описываем функцию

<script type="text/javascript">  
$(document).ready(function(){  
$(".buttontext").click(function(){  
$("#panel").slideToggle("normal"); return false;  
});  
});  
</script>

Комментарии

* Команда **.slideToggle**позволяет чередовать два простых действия: показать и скрыть. normal – это скорость, означает, что панель выедет за 400 миллисекунд, есть еще fast, slow.
* Команда **return false** означает, что мы не даем перейти по ссылке, в качестве которой используется кнопка-закладка. То есть, она просто работает в качестве удобной ручки, за которую дергают, но не как непосредственно ссылка.