Урок 4

Введение в jQuery

Знакомство с библиотекой jQuery. Подключение и работа с библиотекой. Асинхронные запросы в jQuery.

На прошлых занятиях мы рассматривали чистый JavaScript. Его знание очень важно для JS-программиста, однако согласитесь, что, к примеру, строя дом, мы не стремимся сначала нарубить деревьев в лесу, выплавить гвозди из руды и так далее. Мы идём в магазин и покупаем готовый товар, по сути покупая у самих себя свободное время.
Так и готовые библиотеки во многих языках позволяют сэкономить время, не изобретая велосипеды. Одной из таких крайне полезных библиотек является jQuery. О нём и пойдёт речь.

Что такое jQuery

jQuery – это одна из наиболее популярных JavaScript-библиотек, которые обеспечивают кроссбраузерную поддержку front-end приложений. jQuery совместим со следующими браузерами:

- Internet Explorer 6.0+
- Mozilla Firefox 2+
- Safari 3.0+
- Opera 9.0+
- Chrome

В чём же его преимущества? Основным плюсом, с которого начинается jQuery, является крайне удобный механизм селекторов, который даёт возможность очень быстро и легко получать доступ к любому элементу DOM-модели. Именно за это (но не только) разработчики полюбили jQuery.
Для получения ссылки на DOM-элемент в Javascript нужно вызвать метод getElementByld(). Например, для управления HTML-кодом элемента с идентификатором myid нужно будет сделать следующее:

```
document.getElementByld('myid').innerHTML = "Lorem ipsum";
```

А теперь оцените, как лаконично записывается та же операция на jQuery:

```
$('myid').html("Lorem ipsum");
```

Согласитесь, весьма красиво!

Подключение jQuery

Прежде чем начинать работу с jQuery, её надо подключить к вашей странице. Для этого есть два пути:

1. Скачать к себе нужную версию библиотеки.
2. Загружать библиотеку их репозитория Google.

И тот, и другой способы по-своему хороши – в первом случае вы всегда имеете подключенную библиотеку вне зависимости от доступности удалённого репозитория. Во втором - очень удобно реализуется смена версий – достаточно просто поменять ссылку.
Так или иначе, подключение производится с помощью тега script, в параметре src которого указывается абсолютный или относительный путь к библиотеке.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

Сейчас доступно 3 поколения jQuery. В рамках этого курса мы изучим 1 поколение как самое базовое.
Сам тег script размещается внутри тега head Вашего HTML-документа.
Если пользователь в процессе сёрфинга ранее посещал другой сайт, который использует библиотеку jQuery, загруженную из репозитория Google, то браузер не станет ещё раз загружать библиотеку, а

использует кэшированный файл, что однозначно скажется на скорости работы вашего сайта. Это несомненно ещё один плюс использования удалённой библиотеки.

Селекторы jQuery

Для того, чтобы понять, как работает селектор, вам нужно вспомнить свои знания CSS, ведь концепция селекторов базируется именно на них:

- $(“#header”) – получение элемента с id=”header”;
- $(“h3”) – получить все <h3> элементы;
- $(“div#content .photo”) – получить все элементы с классом =”photo”, которые находятся в элементе div с id=”content”;
- $(“ul li”) – получить все <li> элементы из списка <ul>;
- $(“ul li:first”) – получить только первый элемент <li> из списка <ul>.

В качестве параметров функции $() могут быть указаны следующие селекторы:

1. * - коллекция всех тегов:

```
var n = $("*").size() ; // Количество тегов
```

2. Коллекция определённых (по имени) тегов:

```
$("p").css("backgroundColor", "red");
```

3. #id — ссылка на элемент с указанным идентификатором:

```
$("#myid").css("backgroundColor", "red");
```

4. Тег#Идентификатор — ссылка на элемент с указанным идентификатором, который расположен в определённом теге. Если элементов с указанным идентификатором несколько, то будет возвращена ссылка только на первый найденный элемент.

```
$("p#myid").css("backgroundColor", "red");
```

5. Класс — коллекция элементов, имеющих указанный класс:

```
$(".cls2").css("backgroundColor", "red");
```

6. Тег. Класс — коллекция элементов, имеющих указанный класс в определённом теге:

```
$("p.cls2").css("backgroundColor", "red");
```

Для достижения оптимальной скорости работы выборки, не стоит указывать название тега перед идентификатором, т.е. указатель #myid работает быстрее p#myid. Очевидно, выборка производится последовательно, а элементов #myid гораздо меньше, чем параграфов, ведь во втором случае JS сначала выберет все параграфы, а затем будет их фильтровать.
В случае с классами нужно делать с точностью до наоборот. В начале следует указывать название тега.
Если название указателя содержит специальные символы (например, точку или квадратные скобки), их требуется экранировать двумя слэшами.

```
<div id="divl.index[5]"></div>
```

```
$("#divl\\.index\\[5\\]").html("Текст");
```

При необходимости применения одного действия к группе элементов с разными идентификаторами селекторы допускается указывать через запятую.

```
$("#idl, div.hello").addClass("newClass");
```

Привязка к элементам документа

Поиск элемента внутри другого элемента возможно производить следующими способами.

1) Элемент1 Элемент2 — находим Элемент2, который располагается внутри контейнера Элемента1:

```
$("div a").css("color", "red");
```

Цвет текста ссылки станет красным, если тег А находится внутри тега div.

```
<div><a href="/link.html">Ссылка</a></div>
```

2) Элемент1 > Элемент2 — находим Элемент2, который является дочерним для Элемента А.

```
$("div > a").css("color","red");
```

Цвет текста ссылки станет красным, если тег А находится внутри тега div и не вложен в другой тег.

```
<div>
   <a href="#">Ссылка I</a><br>
   <span>
      <a href="#">Ссылка 2</a>
   </span>
</div>
```

3) Элемент1 + Элемент2 — находим Элемент2, который является соседним для Элемент А и следует сразу после него.

```
$("div + a").css("color", "red");
```

Цвет текста ссылки станет красным, если тег А следует сразу после тега div.

```
<div>Lorem ipsum</div><a href="#">href</a>
```

4) Элемент 1 ~ Элемент2 — находим Элемент2, который следует после Элемента 1, причём необязательно сразу.

```
$("div ~ a").css("color", "red");
```

Цвет текста ссылки станет красным, если тег A следует после тега div.

```
<div>Lorem ipsum</div>
<span> Lorem ipsum </span><br>
<a href="#">Href1</a><br>
<a href="#">Href2</a><br>
<span><a href="#">Href3</a></span><br>
<a href="#">Href4</a><br>
```

В этом примере ссылки 1, 2 и 4 станут красного цвета. Ссылка 3 не станет красного цвета, так как расположена внутри тега span.

При необходимости можно составлять выражения из нескольких селекторов:

```
$("div span a").css("color", "red");
```

Цвет текста ссылки станет красным, если тег A расположен внутри тега span, а тот в свою очередь вложен в тег div:

```
<div>
   <a href="#">Href l</a><br>
   <span>
      <a href ="#">Href 2</a><br>
   </span>
</div>
```

Применение на практике

Возможностей применения селекторов и различных методов в jQuery очень много. И перечислять их все – нецелесообразно. Их надо наращивать на практике, чем мы и займёмся.
Создадим простую реализацию - слайд-панель, которая двигает вверх/вниз по клику на ссылке.

Слайдер

По клику на ссылку у самой ссылки будет переключаться класс (между «active» и «btn-slide»), а панель с id=«panel» будет выдвигаться/прятаться.

```
<div id="panel"></div>
<p class="slide"><a href="#" class="btn-slide">Slide Panel</a></p>
```

```
$(document).ready(function(){
	$(".btn-slide").on("click", function(){
		$("#panel").slideToggle("slow");
		$(this).toggleClass("active"); return false;
	});
});
```

Что же происходит в jQuery-коде? Разберём по порядку.
Запись $(document).ready(function(){}); говорит о том, что все действия будут выполняться только после того, как полностью загрузится DOM-модель документа. Это практически необходимое требование для того, чтобы все скрипты работали корректно. При медленном Интернет-соединении возможна ситуация, когда скрипты уже загрузились и начали работать, а некоторых элементов еще попросту нет на стороне клиента.
Далее мы навешиваем обработчик события на элемент с классом btn-slide. При помощи метода on() мы говорим, что при клике на данный элемент должна выполниться ниже указанная функция.
После этого мы показываем или скрываем панель при помощи slideToggle – этот метод позволяет не задумываться о том, показана ли сейчас панель или нет, он просто меняет её состояние на противоположное. В качестве аргумента указывается скорость анимации.
Похожие манипуляции проводятся с классом active.

Растворение

Этот пример покажет способ, при помощи которого можно красиво и легко растворять элементы на странице:

```
<div class="pane">
	<h3>Sample heading</h3>
	<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Morbi malesuada, ante at feugiat tincidunt, enim massa gravida metus, commodo lacinia massa diam vel eros. Proin eget urna. Nunc fringilla neque vitae odio. Vivamus vitae ligula.</p>
	<p class="delete">X</p>
</div>
```

```
$(document).ready(function(){
	$(".pane .delete").on("click", function(){
		$(this).parents(".pane").animate({ opacity: 'hide' }, "slow");
	});
});
```

Здесь мы применяем метод animate, задавая некое CSS-свойство в его параметрах и скорость перехода от текущего значения к требуемому.

Можно и дальше заострять внимание на аспектах анимации в jQuery, но это будет непрактично. В списке ссылок к этому уроку вы найдёте ресурс, на котором очень подробно разбирается применение той или иной анимации.
jQuery предоставляет крайне удобный инструментарий для работы с AJAX.

AJAX в jQuery

jQuery имеет ряд функций, позволяющих обмениваться данными с сервером при помощи технологии AJAX. Базовым методом для данного функционала является метод jQuery.ajax. Другие методы библиотеки по сути являются обертками для него. Метода имеет лишь один входной параметр — это объект, который включает в себя все настройки.

```
$.ajax({
    url: '/ajax/example.html',
    dataType : "json",
    success: function (data, textStatus) {
        $.each(data, function(i, val) {
            /* ... */
        });
    }
});
```

Заметьте, что в данном случае уже не нужно задумываться о том, как создавать объект XHR. Нужно лишь правильно задать все необходимые настройки.

- async — асинхронность запроса, по умолчанию true;
- cache — вкл/выкл кэширование данных браузером, по умолчанию true;
- contentType — по умолчанию «application/x-www-form-urlencoded»;
- data — передаваемые данные — строка или объект;
- dataFilter — фильтр для входных данных;
- dataType — тип данных, возвращаемых в callback-функцию (xml, html, script, json, text, _default);
- global — триггер — отвечает за использование глобальных AJAX Event'ов, по умолчанию true;
- ifModified — триггер — проверяет, были ли изменения в ответе сервера, дабы не слать ещё запрос, по умолчанию false;
- jsonp — переустановить имя callback-функции для работы с JSONP (по умолчанию генерируется на лету);
- processData — по умолчанию отправляемые данный заворачиваются в объект, и отправляются как «application/x-www-form-urlencoded», если надо иначе — отключаем;
- scriptCharset — кодировка — актуально для JSONP и подгрузки JavaScript'ов;
- timeout — время, таймаут в миллисекундах;
- type — GET либо POST;
- url — url запрашиваемой страницы.

Также метод позволяет обработать локальные события:
- beforeSend — срабатывает перед отправкой запроса;
- error — если произошла ошибка;
- success — если ошибок не возникло;

- complete — срабатывает по окончанию запроса.

Расширениями данного метода являются jQuery.get и jQuery.post. Первый загружает страницу, используя для передачи данных GET-запрос. Может принимать следующие параметры:

1) url запрашиваемой страницы.
2) передаваемые данные (необязательный параметр).
3) callback-функция, которой будет отдан результат (необязательный параметр).
4) тип данных возвращаемых в callback функцию (xml, html, script, json, text, _default).

jQuery.post аналогичен .get, но данные уйдут на сервер посредством метода POST.
Может принимать следующие параметры:

1) url запрашиваемой страницы.
2) передаваемые данные (необязательный параметр).
3) callback-функция, которой будет отдан результат (необязательный параметр).
4) тип данных возвращаемых в callback-функцию (xml, html, script, json, text, _default).

Практика

Обладая знаниями по jQuery, можно переписать наше AJAX-меню следующим образом:

1) Загружать его через jQuery AJAX.
2) Скрывать его (сворачивать).

Домашнее задание

1. С помощью jQuery создать контрол, работающий с вкладками. Пример - http://dimox.name/examples/universal-jquery-tabs-script/ . Можно использовать любую анимацию, методы show, hide и подобные. Код примера желательно не смотреть.
2. В форму обратной связи добавить возможность выбора города обращения. Сам список должен загружаться после загрузки страницы через AJAX.
3. * Список из п.2 превратить в текстовое поле-автокомплит. Если пользователь ввёл 3 и более символов, нужно подгрузить список городов и показать подходящие по вводу. При клике на подходящий город, ввести его полное название в текстовое поле.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. https://habrahabr.ru/post/38208/ - анимация в jQuery
2. https://habrahabr.ru/post/42426/ - AJAX в jQuery