Focused forward

Netcracker

# Лекция 2
# Библиотека jQuery

## Клиентские технологии web-программирования

# План лекции

- Библиотека jQuery.
- Начало работы.
- Селекторы.
- События.
- Атрибуты элементов и CSS.
- Анимация.

# Библиотека jQuery

- **Обращение к элементам ДОМ с помощью мульти-браузерного движка Sizzle, который является спин-офф проектом jQuery.**
- **Манипуляции с ДОМ на основе CSS селекторов, которые используют имена и атрибуты элементов, такие как id и class**
- **События**
- **Эффекты и анимация**
- **AJAX**
- **JSON Parsing**
- **Расширяемый за счет плагинов**
- **Поддержка многими браузерами**

# Начало работы

## Подключение jQuery

1. Скачать с официального сайта http://jquery.com/download/ одну из версий (jQuery 1.x, 2.x, 3.x) в сжатом или не сжатом варианте.

2. Закачать файл на сервер, где располагается сайт и подключить его на страницы сайта, прописав путь к файлу:

```
<head>
    <script type="text/javascript" src="путь к jQuery.js"></script>
</head>
```

- **Несжатую версию** удобно использовать во время разработки и отладки сайта.
- **Сжатую версию** целесообразно применять на выпущенной версии сайта, для более оптимальной работы (сжатый скрипт будет быстрее подключен к странице, а так же быстрее обработан).

# Начало работы

## Доступные версии

| | | | |
|---|---|---|---|
| 1.9 | January 15, 2013 | 1.9.1 (February 4, 2013) | 90 |
| 1.10 | May 24, 2013 | 1.10.2 (July 3, 2013) | 91 |
| 1.11 | January 24, 2014 | 1.11.3 (April 28, 2015) | 95.9 |
| 1.12 | January 8, 2016 | 1.12.4 (May 20, 2016) | 95 |
| 2.0 | April 18, 2013 | 2.0.3 (July 3, 2013) | 81.1 |
| 2.1 | January 24, 2014 | 2.1.4 (April 28, 2015) | 82.4 |
| 2.2 | January 8, 2016 | 2.2.4 (May 20, 2016) | 85.6 |
| 3.0[28] | June 9, 2016 | 3.0.0 (June 9, 2016) | 86.3 |
| 3.1 | July 7, 2016 | 3.1.1 (September 23, 2016) | 86.3 |
| 3.2 | March 16, 2017 | 3.2.1 (March 20, 2017) | 84.6 |
| 3.3 | January 19, 2018 | 3.3.1 (January 20, 2018) | 84.8 |

# Начало работы

## Подключение jQuery с CDN

Можно подключить библиотеку, которая находится не на Вашем сервере, а на серверах CDN.
Существуют несколько таких хранилищ, наиболее известные и надежные из них Google CDN, Microsoft CDN, а так же CDN который организовали создатели jQuery.

Необходимо просто прописать соответствующий путь к файлу на сервере CDN.

**Google CDN:**

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.m
in.js"></script>
```

**Microsoft CDN:**

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-
3.3.1.min.js"></script>
```

# Начало работы

```
<head>
<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">
$(document).ready(function(){

    $(".button").click(function(){
        $("#panel").slideDown("slow");
    });

});
</script>
</head>
```

подключаем jQuery

Событие "ready" (функция будет выполнена, когда DOM будет готов

К чему Вы хотите привязать Вашу функцию?
Это может быть class, ID, selector (DIV,H1,P...)

Эта функция будет вызвана по событию "click" на элементе с классом "button"

Что же будет происходить с #panel?
Элемент будет медленно опускаться вниз

Чем мы будем оперировать?
Элементом с ID = panel

```
$("#panel")
```

Для квотирования могут использоваться как одинарные так и двойные кавычки:
("class") == ('class')

# Начало работы

```
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
```

# Начало работы

## Синтаксис jQuery

Синтаксис jQuery специально разработан для выбора элементов HTML и выполнения некоторых действий над элементом (элементами)

```
$(селектор).метод();
```

**Знак $** сообщает, что символы идущие после него, являются jQuery кодом.

**Селектор** позволяет выбрать элемент на странице.

**Метод** задает действие, которое необходимо совершить над выбранным элементом:

- Методы для манипулирования DOM;
- Методы для оформления элементов;
- Методы для создания AJAX запросов;
- Методы для создания эффектов;
- Методы для привязки обработчиков событий.

# Начало работы

```
<script>
	// мы пытаемся найти все элементы <h2> на странице
	// и изменить цвет шрифта на красный
	jQuery("h2").css("color", "red");
</script>
```

Этот **код ничего не сделает**, так как на момент выполнения на странице **ещё не будет тегов <h2>,** слишком рано выполняется скрипт - до загрузки всего HTML документа.

Для того, чтобы **код сработал верно**, мы должны **либо поместить код в самый низ страницы** (главное после искомого <h2>), либо использовать **функцию ready()** для отслеживания события «**load»** нашего «**document**»:

```
<script>
	// ждём загрузки всего документа
	// после этого будет выполнена анонимная функция
	// которую мы передали в качестве параметра
	jQuery(document).ready(function(){
		jQuery("h2").css("color", "red");
	});
</script>
```

# Начало работы

Можно использовать сокращённый вариант без явного вызова метода jQuery():

```
<script>
	$(function(){
		$("h2").css("color", "red");
	});
</script>
```

**$() = jQuery()**

# Начало работы

## Цепочки методов

Важной особенностью большинства методов jQuery, является возможность связывать их в цепочки. Методы, манипулирующие элементами документа, обычно возвращают эти объекты для дальнейшего использования, что позволяет писать примерно следующее:

```
$("#bigIt").empty().attr("class", "noContent");
// в результате, у элемента с идентификатором bigIt будет удалено все содержимое,
// после чего ему будет установлен класс noContent.
```

Эти цепочки могут состоять из гораздо большего числа методов.
Для удобочитаемости, цепочки часто пишут "в столбик":

```
$("div")                    // найдем все div-элементы
.parent()                   // найдем их родительские элементы
.css("heigth", "10px")      // установим последним высоту в 10 пикселей
.fadeTo(0, 0.5)             // установим им (родителям div'ов) прозрачность в 50%
.addClass("divOwner");      // добавим им же класс divOwner
```

# Селекторы

С помощью селекторов можно выбирать элементы на странице для применения к ним определенных действий

| Пример | Результат |
|---|---|
| **$("p")** | Будут выбраны все элементы p, которые находятся на странице. |
| **$(".par")** | Будут выбраны все элементы на странице с class="par". |
| **$("#par")** | Будет выбран первый элемент на странице с id="par". |
| **$(this)** | Позволяет выбрать текущий HTML элемент. |
| **$("p.par")** | Будут выбраны все элементы p на странице с class="par". |
| **$("p#par")** | Будут выбраны все элементы p на странице с id="par". |
| **$(".par,.header,#heat")** | Будут выбраны все элементы на странице со значениями атрибутов class="par", class="header" и id='heat'. |
| **$("[src]")** | Будут выбраны все элементы на странице, имеющие атрибут src. |

# Селекторы

| Пример | Результат |
|---|---|
| **$("[src='значение']")** | Будут выбраны все элементы со значениям атрибута src="значение". |
| **$("[src$='.gif']")** | Будут выбраны все элементы со значениями атрибута src заканчивающимися на .gif. |
| **$("div#wrap .par1")** | Будут выбраны все элементы с class=par1, которые находятся внутри элементов div с id=wrap. |
| **$(":input")** | Будут выбраны все input элементы на странице. |
| **$(":тип")** | Будут выбраны все input элементы с <input type='тип' />. Например :button выберет все <input type='button' /> элементы, :text выберет все <input type='text' /> элементы и т.д. |
| **$("[src='значение']")** | Будут выбраны все элементы со значениям атрибута src="значение". |
| **$("[src$='.gif']")** | Будут выбраны все элементы со значениями атрибута src заканчивающимися на .gif. |
| **$("div#wrap .par1")** | Будут выбраны все элементы с class=par1, которые находятся внутри элементов div с id=wrap. |

# Селекторы

## Выбор элементов по «id» либо «classNam» аналогично CSS

```
$("#content")      // выбираем элемент с id=content
$("div#content")   // выбираем div с id=content (хотя id и так однозначен)
$(".wrapper")      // выбираем элементы с class=wrapper
$("div.wrapper")   // выбираем div'ы с class=wrapper
$(".wrapper.box") // выбираем элементы с class=wrapper и box
$("h2")            // выбираем все теги h2
$("h1, h2")        // выбираем все теги h1 и h2
```

## Выбор элементов с учетом иерархической структуры

```
$("article h2")      // выбираем все теги h2 внутри тега article
$("div article h2") // выбираем все теги h2 внутри тега article
                     // внутри тега div

$("article").find("h2")                // аналогично примерам выше
$("div").find("article").find("h2")  //
```

# Селекторы

## Выбор соседних элементов

```
$("h1 + h2")         // выбор всех h2 элементов, перед которыми идут h1
                     // элементы (у нас только один такой)
$("#stick ~ article")  // выбор всех article элементов после элемента
                       // с id=stick
$("#stick").prev() // выбор предыдущего элемента от найденного
$("#stick").next() // выбор следующего элемента от найденного
```

## Выбор дочерних элементов

```
$("*")                  // выбор всех элементов
$("article > h2")       // выбираем все теги h2 которые являются
                        // непосредственными потомками тега article
$("article > *")        // выбор всех потомков элементов p
$("article").children()// --

$("p").parent()         // выбор всех прямых предков элементов p
$("p").parents()        // выбор всех предков элементов p (не понадобится)
$("p").parents("div")   // выбор всех предков элемента p которые есть div
                        // parents принимает в качестве параметра селектор
```

# События

**События** – это любые действия пользователя, будь то ввод данных с клавиатуры, проматывание страницы или передвижения мышки, или «клики».

**Обработчики событий** - это функции, код которых исполняется только после совершения определенных событий.

**$(селектор).обработчик_события(function(){код_обработчика_события});**

**Пример**

```
$(document).ready(function(){

    $("p").mouseover(function(){$("p").css("color","green")});
    $("p").mouseout(function(){$("p").css("color","black")});

});
```

# События

**Базовые**

| | |
|---|---|
| **.on()** | Универсальный метод для установки обработчиков событий на выбранные элементы страницы. |
| **.off()** | Удаляет обработчики, установленные с помощь .on(). |
| **.bind()** | Устанавливает обработчик события на выбранные элементы страницы. Обработчик не сработает на элементах, появившихся после его установки. |
| **.live()** | Устанавливает обработчик события на выбранные элементы страницы. Обработчик сработает и на элементах, появившихся после его установки. |
| **.delegate()** | Устанавливает обработчик события на выбранные элементы страницы. Элементы выбираются с помощью уточняющего селектора. Обработчик будет действовать и на элементах, появившихся после его установки. |
| **.one()** | Устанавливает обработчик события на выбранные элементы страницы, который сработает только по одному разу, на каждом из элементов. |
| **.unbind()** | Удаляет обработчик событий у выбранных элементов. |
| **.die()** | Удаляет обработчик событий, который был установлен с помощью live(). |

# События

## Базовые

| | |
|---|---|
| **.undelegate()** | Удаляет обработчик событий, который был установлен с помощью delegate(). |
| **.trigger()** | Выполняет указанное событие и запускает его обработчик. |
| **.triggerHandler()** | Запускает обработчик указанного события, без его выполнения. |
| **jQuery.proxy()** | По заданной функции, создает другую, внутри которой переменная this будет равна заданному значению. |
| **event** | Объект, содержащий данные о текущем событии. Передается всем обработчикам событий. |

## События мыши

| | |
|---|---|
| **.click()** | обработчик "клика" мышью по элементу, либо, запускает это событие. |
| **.dblclick()** | обработчик двойного "клика" мышью по элементу, либо, запускает это событие. |
| **.hover()** | обработчик двух событий: появления/исчезновения курсора над элементом. |

# События

```
$(document).ready(function(){

   $("#but1").click(function(){alert("Вы нажали один раз на первую кнопку!");});
   $("#but2").dblclick(function(){alert("Вы нажали два раза на вторую кнопку!");});

});
```

- Код обработчика **click** будет выполнен после **одинарного щелчка мыши** на элементе.
- Код обработчика **dblclick** будет выполнен после **двойного щелчка мыши** на элементе.

```
$(document).ready(function(){

   $("#square").click(function(event){
      $("#coord").css("display","block");
      $("#x").html(event.pageX);
      $("#y").html(event.pageY);
   });

});
```

С помощью объекта **event** выводятся координаты, на которых находился курсор мыши во время того, как произошло событие.

# События

**События мыши**

| | |
|---|---|
| **.mousedown()** | нажатия кнопки мыши, либо, запускает это событие. |
| **.mouseup()** | обработчик поднятия кнопки мыши, либо, запускает это событие. |
| **.mouseenter()** | обработчик появления курсора в области элемента, либо, запускает это событие. Появление этого события, отработано лучше, чем стандартного mouseover. |
| **.mouseleave()** | обработчик выхода курсора из области элемента, либо, запускает это событие. Появление этого события, отработано лучше, чем стандартного mouseout. |
| **.mousemove()** | обработчик движения курсора в области элемента, либо, запускает это событие. |
| **.mouseout()** | обработчик выхода курсора из области элемента, либо, запускает это событие. |
| **.mouseover()** | обработчик появления курсора в области элемента, либо, запускает это событие. |
| **.toggle()** | поочередно выполняет одну из двух или более заданных функций, в ответ на "клик" по элементу. |

# События

```
$("#p1").mouseenter(function(){
  alert("You entered p1!");
});
```

Функция выполняется, когда **курсор появляется в области** HTML элемента

```
$("#p1").mousedown(function(){
alert("Mouse down over p1!");
});
```

Функция выполняется, при **нажатии левой кнопки мыши в нажатом положении, в то время курсор находится над** HTML элементом.

```
$("p").dblclick(function(){
$(this).hide();
});
```

Функция выполняется **при двойном "клике"** мышью по HTML элементу.

# События

**События формы**

| | |
|---|---|
| **.focus()** | обработчик получения фокуса, либо, запускает это событие. |
| **.blur()** | обработчик потери фокуса, либо, запускает это событие. |
| **.focusin()** | обработчик получения фокуса самим элементом или одним из его дочерних. |
| **.focusout()** | обработчик потери фокуса самим элементом или одним из его дочерних. |
| **.select()** | обработчик выделения текста, либо, запускает это событие. |
| **.submit()** | обработчик отправки формы, либо, запускает это событие. |
| **.change()** | обработчик изменения элемента формы, либо, запускает это событие. |

```
$(document).ready(function(){

    $("#el1").focus(function(){$(this).attr("value","")});
    $("#el1").blur(function(){$(this).attr("value","Введите ФИО")});
    $("#el2").change(function(){ alert("Содержимое данного элемента было изменено.") }

});
```

- Код обработчика **focus()** будет выполнен, когда **элемент станет активным**.
- Код обработчика **blur()** будет выполнен, когда **элемент перестанет быть активным**.
- Код обработчика **change()** будет выполнен, **при изменении содержимого элемента.**

# События

**События формы**

```
$("input").focus(function(){
    $(this).css("background-color", "#cccccc");
});
```

```
$("input").blur(function(){
    $(this).css("background-color", "#ffffff");
});
```

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

```
$("p").on("click", function(){
    $(this).hide();
});
```

# События

## События клавиатуры

| | |
|---|---|
| **.keydown()** | обработчик перехода клавиши клавиатуры в нажатое состояние, либо, запускает это событие. |
| **.keyup()** | обработчик возвращение клавиши клавиатуры в ненажатое состояние, либо, запускает это событие. |
| **.keypress()** | обработчик ввода символа с клавиатуры, либо, запускает это событие. |

## События загрузки страницы

| | |
|---|---|
| **.ready()** | обработчик готовности дерева DOM. |
| **.load()** | обработчик завершения загрузки элемента. |
| **.unload()** | обработчик ухода со страницы (при переходе по ссылке, закрытии браузера и.т.д.). |

## События браузера

| | |
|---|---|
| **.error()** | обработчик ошибки при загрузке элементов (например отсутствие необходимой картинки на сервере). |
| **.resize()** | обработчик изменения размеров окна браузера, либо, запускает это событие. |
| **.scroll()** | обработчик "прокрутки" элементов документа, либо, запускает это событие. |

# Атрибуты элементов и CSS

Метод css() возвращает текущее значение стиля, а не прописанное в CSS файле по указанному селектору.

```
css(property) — получение значения CSS свойства
css(property, value) — установка значения CSS свойства
css({key: value, key:value}) — установка нескольких значений
```

**Пример**

```
$("#my").css('color')          // получаем значение цвета шрифта
$("#my").css('color', 'red')   // устанавливаем значение цвета шрифта
```

# Атрибуты элементов и CSS

**Пример**

```
// установка нескольких значений
$("#my").css({
        'color':'red',
        'font-size':'14px',
        'margin-left':'10px'
})

// альтернативный способ
$("#my").css({
        color:'red',
        fontSize:'14px',
        marginLeft:'10px',
})
```

# Анимация

- jQuery имеет ряд функций, выполняющих **анимационные эффекты с элементами страницы**: скрытие, появление, перемещение элементов.
- Библиотека позволит **создавать сво**и, более сложные эффекты, основанные на изменении CSS свойств. Эти изменения могут происходить плавно или мгновенно, замедляться, ускоряться или выполняться равномерно.

**Управление анимацией**

| | |
|---|---|
| **.animate()** | Выполняет анимацию, которая была создана пользователем. |
| **.queue()** | Предоставляет/изменяет (в зависимости от параметров) очередь функций. |
| **.clearQueue()** | Очищает очередь функций. |
| **.dequeue()** | Начинает выполнение следующей функции в очереди. |
| **.stop()** | Останавливает выполнение текущей анимации. |
| **.delay()** | Приостанавливает выполнение следующих анимаций на заданное время. |
| **.toggle()** | Поочередно выполняет вызов одной из нескольких заданных функций. |
| **jQuery.fx.interval** | Содержит временной промежуток между кадрами анимации. |

# Анимация

**Стандартные анимации**

| | |
|---|---|
| **.hide()**<br>**.show()** | Скрывает/показывает элементы на странице (за счет плавного изменения его размера и прозрачности). |
| **.slideUp()**<br>**.slideDown()** | Разворачивает/сворачивает элементы на странице (за счет плавного изменения высоты элементов). |
| **.slideToggle()** | Поочередно разворачивает/сворачивает элементы на странице, как это делают .slideUp() и .slideDown(). |
| **.fadeIn()**<br>**.fadeOut()** | Скрывает/показывает элементы на странице за счет плавного изменения прозрачности. |
| **.fadeTo()** | Плавно изменяет прозрачность элементов. |
| **.fadeToggle()** | Поочередно скрывает/показывает элементы на странице, как это делают .fadeIn() и .fadeOut(). |

# Примеры кода

Поиск элементов по названию тэга

| Фреймворк | Код |
| --- | --- |
| Vanilla JS | `document.getElementsByTagName("span");` |
| Prototype JS | `Prototype.Selector.select('span', document);` |
| YUI | `YAHOO.util.Dom.getElementsBy(function(){return true;},'span')` |
| Ext JS | `Ext.query('span');` |
| jQuery | `$jq('span');` |
| Dojo | `dojo.query('span');` |
| MooTools | `Slick.search(document, 'span', new Elements);` |

# Рекомендованная литература

1. https://jquery.com/
2. http://plugins.jquery.com
3. http://www.w3schools.com/jquery/
4. http://jquery.page2page.ru
5. http://try.jquery.com
6. Бибо Б., Кац И. jQuery. Подробное руководство по продвинутому JavaScript (2009)
7. Самков Г. А. jQuery. Сборник рецептов (2-е издание, 2011)

# Q&A

# Thank You

Focused forward

Netcracker