

# JavaScript- библиотеки.

---

**jQuery**

# jQuery

---

- Разные Web-браузеры могут по-разному выполнять код программы. По этой причине при написании приложений приходится учитывать особенности каждого Web-браузера. Проблема заключается в том, что установить все версии каждого Web-браузера на один компьютер сложно, а значит, обеспечить полную кросс-браузерность самостоятельно не получится.

# jQuery

---

- При использовании библиотек любой программист может сообщить о проблеме в каком-либо Web-браузере, а разработчик библиотеки, имеет возможность обработать ошибку. После исправления ошибки достаточно сменить версию библиотеки. Таким образом, используя возможности какой-либо библиотеки можно забыть о проблеме с кроссбраузерностью приложения.

# jQuery

---

- Наиболее часто используются следующие JavaScript-библиотеки:
- jQuery — <http://jquery.com/>;
- Prototype — <http://www.prototypejs.org/>;
- ExtJS — <http://www.extjs.com/>;
- MooTools — <http://mootools.net/>;
- Yahoo! UI Library (YUI) — <http://developer.yahoo.com/yui/>.

# jQuery

---

- Особо выделим библиотеку jQuery, предоставляющую функциональность, которую может использовать практически любой разработчик, даже не владея основами JavaScript. Она обеспечивает кросс-браузерную поддержку приложений (работает в Internet Explorer 6.0+, Mozilla Firefox 2+, Safari 3.0+, Opera 9.0+ и Chrome), имеет небольшой размер.

# jQuery

---

- Большой популярности jQuery способствовали дополнительные модули (более 1500), реализующие готовые компоненты или добавляющие новую функциональность. Например, библиотека jQuery UI добавляет возможность перемещения и изменения размеров любых элементов с помощью мыши, позволяет сортировать и выделять элементы, а также предоставляет готовые компоненты ("Аккордеон", панель с вкладками, диалоговые окна, календарь и др.).

# jQuery

---

- Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими.
- Автор библиотеки, Джон Резиг, в 2004 – 2006 годах создал множество различных JavaScript-приложений, прежде чем закончить создание jQuery.

# jQuery

---

- Основной целью создания jQuery Резиг видел возможность закодировать многократно повторяющиеся куски кода, которые позволят упростить JavaScript и использовать их так, чтобы не беспокоиться о кросс-браузерных вопросах. Библиотека была представлена обществу на компьютерной конференции «BarCamp» в Нью-Йорке в 2006 году.

# jQuery

---

- jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл, скачанный с сайта jQuery *http://jquery.com*:
- **`<script type="text/javascript" src=" jquery-1.7.2.js"></script>`**
- Если желания скачивать нет, то можно указать ссылку прямо на последнюю версию библиотеки, расположенную на сервере разработчика:

# jQuery

---

- `<script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"> </script>`
- Ссылаться на последнюю версию хорошо, поскольку в ней постоянно устраняют какие-то ошибки и оптимизируют работу отдельных ее функций, но в новой версии могут появиться изменения, которые приведут к изменениям в работе проекта.

# Синтаксис jQuery

---

- Синтаксис запросов jQuery достаточно прост. Именно запросов, поскольку программирование на JavaScript с использованием jQuery можно отнести к *декларативному программированию*, которое отличается от классического процедурного программирование тем, что вы описывает “что нужно сделать”, а не “как сделать”.

# Синтаксис jQuery

---

- В названии библиотеки отражена ее суть – язык запросов в контексте JavaScript. Все запросы на jQuery начинаются вызовом функции \$, аргументом которой является сам запрос. Итак, обобщенный синтаксис jQuery запроса выглядит примерно так:
- **`$(запрос)[{.фильтр(критерий)}].действие(аргументы){ .действие(аргументы)}`**

# Синтаксис jQuery

---

- **Запрос** – это селектор или список селекторов, разделенных запятой. Синтаксис запроса совпадает с синтаксисом селекторов CSS стилей. Таким образом, запрос jQuery может начинаться примерно так:
- **\$("#div")** - выбор всех элементов **div**;
- **\$("#div, p")** – выбор всех элементов **div** и элементов **p**;

# Синтаксис jQuery

---

- `$(".class_name")` – выбор всех элементов класса `class_name`;
- `$("#element_id")` – выбор элемента с идентификатором `element_id`;
- `$(*)` – выбор вообще всех элементов;
- `$(".class_name span")` – выбор всех элементов `span` в рамках всех элементов класса `class_name`;

# Синтаксис jQuery

---

- `$("#element_id > div")` – выбор всех элементов `div`, являющихся прямыми потомками (дочерними элементами) элемента с идентификатором `element_id`;
- `$("#element_id .class_name + div")` – выбор всех элементов, которые следуют сразу за элементами класса `class_name` в рамках элемента `element_id`;

# Синтаксис jQuery

---

- `$("#ul_id > li:first")` – выбор первого элемента `li` списка `ul_id`. Также, можно выбрать последний `last` элемент либо все четные `even` или нечетные `odd` элементы. Есть и другие варианты.
- `$("a[href^='http://ru.wikipedia.org']")` – выбор всех ссылок со значением атрибута `href`, начинающегося на `"http://ru.wikipedia.org"`.

# Синтаксис jQuery

---

- **Фильтр** – способ отфильтровать набор выбранных элементов по дополнительным критериям. Критерии – те же селекторы. Есть две противоположные друг другу операции: **filter** и **not**. Если использовать операцию **filter(критерий)**, то выбранными останутся только те элементы, которые удовлетворяют аргументу - критерию, а если **not**, то останутся только те, что не удовлетворяют.

# Синтаксис jQuery

---

- Обычно, смысл имеет использовать операцию **not**, поскольку критерий операции **filter** обычно легко объединить с основным селектором функции **\$()**. Вот пара примеров использования этих операций:
- **\$("div").not(".class\_name")** – исключаем все **div** класса **class\_name**;
- **\$("td, th, div").filter(".class\_name")** – отбираем **td, th, div** класса **class\_name**.

# Синтаксис jQuery

---

- **Действие** – некоторая операция над множеством выбранных элементов. Каждая операция возвращает результат: либо то же самое множество, если операция не связана с созданием новых или удалением выбранных элементов, либо уже отредактированное множество, либо набор вновь созданных элементов. Можно указать сразу несколько операций, разделенных точками.

# Синтаксис jQuery

---

- Приведенный ниже пример выполнит отбор всех элементов **span**, не относящихся к классу **class\_name** и скопирует их в элемент с идентификатором **target**:
- `$("#span").not(".class_name").clone().appendTo("#target");`
- Результат каждой операции – это *массив элементов HTML* *Element*.

# Синтаксис jQuery

---

- Размер массива можно узнать с использованием свойства **length** или функции **size()**. Аргументами операций могут быть не только строковые параметры, но и результаты других запросов:
- `$("#span").not(".class_name").clone().appendTo($("#span.class_name"))`

# Синтаксис jQuery

---

- В качестве аргумента можно использовать не весь результат запроса, а отдельный его элемент:
- `$("#span.class_name").append($("#span").not(".class_name")[0]);`
- Таким образом, аргументом операции может быть либо селектор в виде строки, либо результат другого запроса, либо любой экземпляр *HTMLElement*.

# Операции jQuery

---

- **Свойства**
- Возвращают или устанавливают значения различных свойств выбранных элементов. Если необходимо определить новое значение свойства, то его следует передать в качестве аргумента, например, `$("#element").text(text_value)` или `$("#element").html(html_value)`.
- `text()`: Возвращает текст.

# Операции jQuery

---

- **offset():** Смещение {left, top}. Пример: Игра 'Арканоид', функция перемещения шарика `ball.move()`:
- `$("#ball").offset({left:$("#ball").offset().left + this.dx,top:$("#ball").offset().top + this.dy});`.
- **width(), height():** Ширина и высота.
- **html():** html-разметка внутри выбранных элементов.

# Операции jQuery

---

- **Индикаторы**
- Возвращают **true** или **false**. Их можно использовать в логических выражениях условных операторов и циклов:
- **hasClass(class)**: Имеют ли выбранные элементы указанный класс.
- **is(expression)**: Проверяет истинность выражения для выбранных элементов, например: **is(":first-child")**.

# Операции jQuery

---

- **Методы.**
- Выполняют определенные действия с выбранными элементами:
- **css(parameter, value):** Определяет значение параметра CSS стиля, например:  
`$("#div_id").css("box-shadow", "inset 0px 0px 20px #fff;");`

# Операции jQuery

---

- **addClass(class), removeClass(class), toggleClass(class):** Добавляет, удаляет и переключает для выбранных элементов указанный класс CSS стиля, например:  
`$("#div").addClass("divClass");`
- **append(source), prepend(source):** Добавляет во все выбранные элементы (в конец - **append**, в начало - **prepend**) указанную разметку или результат другого запроса, например:  
`$("#target").append($("#span:last"));`

# Операции jQuery

---

- Более полный пример: Игра 'Арканоид', формирование массива 'кирпичей', функция `pa.initBricks()`:
- `for (var i = 0; i < _bricks_lines; i++) for (var j = 0; j < _bricks_line; j++) $("#playing-area").prepend("<div id='brick_"+i+"_"+j+... + "</div>");`

# Операции jQuery

---

- **appendTo(target), prependTo(target):** Добавляет выбранные элементы в указанный в качестве аргумента элемент, например:  
`$("#span:first").appendTo($("#span:last"));`
- **after(source), before(source):** Добавление указанного в качестве аргумента содержимого после или до выбранных элементов.

# Операции jQuery

---

- **insertAfter(target), insertBefore(target):** Добавление выбранных элементов до или после указанного в качестве аргумента содержимого.
- **wrap(wrapper), wrapAll(wrapper), wrapInner(wrapper):** Обертывание каждого, всех или только внутреннего содержания выбранных элементов указанной оберткой, например: `$("span").wrapAll("<P>");`

# Операции jQuery

---

- **replaceWith(new), replaceAll(old):** Операции замещения: замещение выбранных элементов новой разметкой или замещение указанной в качестве аргумента разметки выбранными элементами.
- **empty():** Удаляет у выбранных элементов все содержимое.
- **remove():** Удаляет выбранные элементы.

# Операции jQuery

---

- **clone()**: Копирует выбранные элементы (в память). Далее, созданные копии можно использовать в качестве аргументов других операций, например:  
`$("#span").not(".class_name").clone().appendTo("#target");`
- **removeAttr(attribute)**: Удаляет указанный атрибут у всех выбранных элементов.

# Операции jQuery

---

- `ready(handler)`, `mousemove(handler)`,  
`mousedown(handler)`, `click(handler)`, . . . :  
Добавляют обработчики соответствующих  
наименованию метода событий (`onload`,  
`onmousemove`, `onmousedown`, `onclick`),  
например:  
`$(document).mousemove(function(event){...  
})`.

# Операции jQuery

---

- Пример - Игра 'Арканоид', инициализация страницы:
- `$(document).ready(function() {`
- `//Создание объектов ...`  
`$(document).mousemove(function(event) {`
- `//Перемещение платформы ...`  
`}) $(document).mousedown(function(event)`  
`{ //Запуск шарика ...`  
`}})`

# Операции jQuery

---

- **bind(event, handler):** Добавляет обработчик по идентификатору события, например:  
`$("#button").bind("click", function(){...})`
- **hover(over, out):** Добавляет обработчики событий `mouseover` и `mouseout`.
- **hide(delay, callback), show(delay, callback), toggle(delay, callback):** Управляют видимостью элементов и обработчик завершения переключения.

# Операции jQuery

---

- Пример - Игра 'Арканоид', удаление 'кирпича' после попадания шарика, функция `ball.move()`:  
`$("#"+_brick.id).hide(200, function(){ $("#"+this.id).remove()});`
- `slideUp(delay, callback)`, `slideDown(delay, callback)`, `slideToggle(delay, callback)`:  
Позволяют сворачивать и разворачивать выбранные элементы и обработчик завершения сворачивания.

# Операции jQuery

---

- **fadeIn(delay, callback), fadeout(delay, callback), fadeTo(delay, target\_opacity, callback):** Позволяют плавно управлять прозрачностью элементов от 0 до 1. Метод **fadeTo** позволяет плавно установить прозрачность до определенного **target\_opacity** значения. Пример - Игра 'Арканоид', визуальный эффект при отскоке шарика от платформы, функция **ball.move()**:

# Операции jQuery

---

- `$("#platform").fadeTo(200, 0.5, function() {$("#platform").fadeTo(200,1)});`  
**animate(target\_style, delay):** Позволяет плавно изменить текущие параметры стиля выбранных элементов до установленных **target\_style** значений, например, переместить элемент:  
`$("#object").animate({left: new_left, top: new_top}, 1000);`

# Примеры с jQuery

---

- Рассмотрим несколько небольших, но законченных примеров.
- **1. Выдвижная панель**
- В начале создаем html-код:
- `<div id="panel">`
- `<!-- Содержимое спрятанного блока -->`
- `</div>`

# Примеры с jQuery

---

- `<p class="slide"><a href="#" class="btn-slide">Slide Panel</a></p>`
- Добавляем CSS-стили:
- `#panel {background: #754c24;height: 200px;display: none;}`
- `.slide {margin: 0;padding: 0;border-top: solid 4px #422410; background: url(slid.gif) no-repeat center top;}`

# Примеры с jQuery

---

- `.btn-slide { background: url(ar.gif) no-repeat right -50px; text-align: center; color: #fff; width: 144px; height: 31px; padding: 10px 10px 0 0; margin: 0 auto; display: block; font: bold 120%/100% Arial, Helvetica, sans-serif; text-decoration: none;}`
- `.active {background-position: right 12px;}`

# Примеры с jQuery

---

- Создаем оболочку для нашего кода:
- `$(document).ready(function(){`
- `});`
- Панелька будет выдвигать при клике на объект с классом `btn-slide`.
- `$(document).ready(function(){`
- `$(".btn-slide").click(function(){`
- `}); });`

# Примеры с jQuery

---

- Привязываем к объекту эффект **slideToggle** (скорость: "slow") и **toggleClass**:
- `$(document).ready(function(){`
- `$(".btn-slide").click(function(){`
- `$("#panel").slideToggle("slow");`
- `$(this).toggleClass("active");`
- `return false;`
- `}); });`

# Примеры с jQuery

---

- Если `#panel` имеет `display:none`, то объект плавно выползет вниз. Если же `display:block`, то объект плавно поднимется вверх и исчезнет.
- Строчка `$(this)` означает, что действие будет происходить с объектом на который кликаем `.btn-click`. А `toggleClass` добавляет или убирает класс `active`.
- [Пример использования этого скрипта.](#)

# Примеры с jQuery

---

- 2. Эффект исчезновения
- Html-оболочка:
- `<div class="pane">`
  - `<h3>Блок</h3>`
  - `<p>Текст внутри блока</p>`
- `<span class="delete">Удалить</span>`
- `</div>`

# Примеры с jQuery

---

- Когда кликаем по тексту `<span class="delete">Удалить</span>`, будет найден родительский элемент `<div class="pane">`, и его прозрачность будет изменяться от `opacity= 1.0` до `opacity=hide` со скоростью `slow`:
- `$(document).ready(function){`
- `$(".pane .delete").click(function){`
- `$(this).parents(".pane").fadeOut("slow");}); });`

# Полезные jQuery плагины

---

- **Галерея Highslide JS**
- Highslide JS – jQuery-галерея для показа фотографий и другого медиа-контента, например, flash-роликов, динамического контента, загружаемого с помощью AJAX, видео с youtube и многого другого. Этот jQuery-плагин обладает массой возможностей и настроек.

# Полезные jQuery плагины

---

- Вот некоторые из них:
- высокая скорость работы и элегантный вид;
- не требует подключения сторонних плагинов, таких как Flash или Java;
- увеличение изображения производится одним кликом, при этом посетитель может прокрутить страницу, не закрывая окно с изображением;

# Полезные jQuery плагины

---

- если отключить все дополнительные возможности галереи, объем ее файлов составит всего 10 Кб;
- при отключенном javascript или при использовании устаревшего браузера, пользователь все равно увидит содержимое, только откроется оно в новом окне или новой вкладке;
- доступен исходный код;
- [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- **Галерея jQuery PrettyPhoto.**
- PrettyPhoto – полноценная lightbox галерея, которая кроме изображений поддерживает также видео файлы, Flash, YouTube и iFrame. Установка этой jQuery-галереи не составляет особого труда. Поддерживает все основные браузеры. Работает даже в IE6. Это абсолютно свободное ПО.
- [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- **jQuery - фотогалерея Lightbox2.**
- Lightbox2 – небольшой jQuery-плагин для показа изображений во всплывающем блоке поверх содержимого страницы с затемнением. Он прост в установке и работает во всех современных браузерах. При его использовании время загрузки страницы увеличивается незначительно.
- [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- **jQuery слайдер noUiSlider.**
- noUiSlider – небольшой, гибкий в настройке слайдер «без наворотов». Поддерживает следующие браузеры: Google Chrome, Firefox, Opera, Safari и Internet Explorer, начиная с 7 версии. Поддерживает возможности тачскрин. Это отличная альтернатива jQuery UI в случае, если кроме слайдера ничего больше не нужно. [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- **jQuery слайдер Dragdealer JS.**
- Dragdealer JS разработан для знающих frontend-разработчиков. Минимальный размер этого jQuery-плагина составляет всего 12 Кб. Взаимодействие с этим слайдером осуществляется с помощью мыши, также поддерживаются устройства с тачскрин.
- [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- **jQuery слайдер Nivo Slider.**
- Красивый и простой в использовании слайдер. Абсолютно бесплатен. Преимущества Nivo Slider в следующем:
- при смене изображений доступно 16 различных визуальных эффектов;
- простая и гибкая настройка;
- небольшой размер;
- [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- По функционалу clickCarousel не выделяется — есть возможность перехода от одного изображения к другому в обе стороны.
- [Демо](#). [Скачать](#)
- **jQuery карусель Modular**
- Позволяет галерею с эффектом карусели или слайдера. Гибко настраивается, вплоть до создания собственных элементов управления и эффектов смены изображений.

# Полезные jQuery плагины

---

- Modular поддерживает устройства с тачскрин. Область применения этой jQuery карусели широка: от онлайн-портфолио и поочередного показа нескольких рекламных баннеров до галереи изображений и отображения простого текстового содержимого.
- [Демо](#). [Скачать](#)

# Полезные jQuery плагины

---

- **cuSel.** Предназначен для улучшения интерфейса элементов. Для этих целей можно использовать плагин jQuery UI, но он большой, а иногда требуется изменить немного, например элемент `select`. Для этого подойдет cuSel. На странице плагина можно ознакомиться с настройками и скачать его.
- И в заключение - обзор Топ-40 полезных jQuery-плагинов 2013 г.