# Intro

## Start

### Include library

<!--

http://jquery.com/ - официальный сайт библеотеки JQuery

http://api.jquery.com/ - документация

-->

<!DOCTYPE html>

<html>

<head>

<title>Знакомство с библиотекой JQuery. Подключение удаленной библиотеки.</title>

<!-- Подключение удаленной библиотеки (CDN) -->

<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>

<!-- Подключение локальной библиотеки -->

<!-- <script src="jquery-1.10.2.js"></script> -->

<script>

jQuery(); // Главная функция для использования всех возможностей jQuery.

$(); // Псевдоним функции jQuery() с тем же набором функционала.

</script>

</head>

<body>

<p>JQuery - write less, do more.</p>

<p><a href="http://jquery.com/" target="\_blank">Official site</a></p>

<p><a href="http://api.jquery.com/" target="\_blank">Documenatation</a></p>

</body>

</html>

### Ready

<head>

<title>Метод ready().</title>

<script src="jquery-1.10.2.js"></script>

<script>

/\*

Событие ready происходит в момент готовности DOM-дерева, что происходит раньше начала загрузки "тяжелых" мультимедийных файлов.

Это оптимальный момент, когда можно приступить к установке обработчиков различных событий и выполнять другой подготовительный javascript-код. ВАЖНО!

Если подготовительные действия требуют наличия таких элементов как картинки,

правильным будет полагаться на событие load. Однако, имейте ввиду, что одновременное использование .ready() и <body onload="..."> приведет к ошибке.

Если вам необходимо использовать оба события, используйте метод библиотеки jQuery .load(), вместо <body onload="...">.

\*/

jQuery(document).ready(function () {

alert('Структура страницы сформирована, можно приступать!');

jQuery("p").text("Hello World (jQuery)");

});

// $() - псевдоним функции jQuery()

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

// alert('Структура страницы сформирована, можно приступать!');

// $("p").text("Hello World (jQuery)");

//});

</script>

</head>

<body>

<p></p>

</body>

</html>

## Selectors

### SelectAll

<title>Селектор \*</title>

<script src="jquery-1.10.2.js"></script>

<script>

/\*

$("\*") - выбор всех элементов на странице.

css("a","b") - метод jQuery, позволяющий добавлять CSS-стили елементам.

a - имя CSS-свойства.

b - значение.

\*/

$(document).ready(function () {

$("\*").css("font-style", "italic");

});

</script>

</head>

<body>

<p>Paragraph in body</p>

<div>

<p>Paragraph in div</p>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

</ul>

</div>

</body>

### Select Tag Name

<title>Селектор имя тега</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E") - соответствует всем элементам с именем тега E.

// Выбор всех параграфов на странице и установка СSS свойству color значения red.

$("p").css("color", "red");

});

</script>

</head>

<body>

<p>Paragraph in body</p>

<div>

<p>Paragraph in div</p>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

</ul>

</div>

</body>

### Multiple select

<title>Группировка</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E,G,H") - соответствует всем элементам с именами тегов E, G, H.

// Выбор тегов "p" и "ul" на странице.

$("p,ul").css("color", "red");

});

</script>

</head>

### Select inherit

<title>Выбор вложенных элементов (потомков)</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E F") - соответствует всем элементам с именем тега F, вложенным в элемент с именем тега E.

// Выбор всех параграфов, вложенным в тег "div".

$("div p").css("color", "red");

});

</script>

</head>

### Neighborhood select

<head>

<title>Выбор соседних элементов.</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E + F") - соответствует всем элементам с именем тега F, которым непосредственно предшествует элемент E на том же уровне вложенности.

// Выбор всех элементов "div", которым непосредственно предшествует элемент "p" на том же уровне вложенности.

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

});

</script>

</head>

<body>

### Child select

<title>Выбор прямых потомков.</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E > F") - соответствует всем элементам с именем тега F, являющимися прямыми потомками элементов с именем тега E.

// Выбор всех элементов "span", являющимися прямыми потомками элементов "div".

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

});

</script>

</head>

### Has selector

<title>"Has"</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E:has(F)) - соответствует всем элементам с именем тега E, имеющим хотя бы один вложенный элемент с именем тега F.

// Выбор всех элементов с именем тега div, имеющим хотя бы один вложенный элемент с именем тега ul.

$("div:has(ul)").css("background", "red");

});

### Class select

<title>Выборка по имени класса</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E.C) - соответствует всем элементам E с именем класса C. При отсутствии E эквивалентен селектору \*.C

// Выбор всех элементов "p" с именем класса "red".

$("p.red").css("color", "red");

});

### Id select

<title>Выборка по идентификатору</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E#I) - соответствует всем элементам E с именем идентификатора I.

// Выбор всех элементов "p" с идентификатором "red".

$("p#red").css("color", "red");

});

### Sibling select

<title>Выборка "братьев".</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E ~ F") - соответствует всем элементам с именем тега F, которым предшествует элемент E на том же уровне вложенности.

// Выбор всех элементов "p", которым предшествует элемент "span" на том же уровне вложенности.

$("span ~ p").css("color", "red");

});

</script>

### Select attribute1

<title>Выборка по атрибутам.</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E[A]") - соответствует элементам E с атрибутом A, имеющим любое значение.

// Выбор всех элементов "a", с атрибутом "target", имеющим любое значение.

$("a[target]").css("color", "red");

});

### Select attribute2

<title>Выборка по атрибутам.</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

// $("E[A=V]") - соответствует элементам E с атрибутом A, который равен V.

// Выбор всех элементов "a", с атрибутом "target", имеющим значение text.

// Метод val() - метод, который позволяет получить значение атрибута "value" первого элемента в наборе соответствующих селектору элементов или установить новое значение (переданное в качестве аргумента) для всех елементов, подходящих под выборку.

$("input[type=text]").val("admin");

});

### Select attribute3

<title>Выборка по атрибутам.</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(document).ready(function () {

/\*

$("E[A^=V]") - соответствует элементам E с атрибутом A, который начинается с V.

$("E[A$=V]") - соответствует элементам E с атрибутом A, который заканчивается на V.

$("E[A\*=V]") - соответствует элементам E с атрибутом A, который содержит V.

\*/

$("a[href^='http://']").css("color", "red"); // Выбор всех элементов "a", с атрибутом "href", который начинается c "http://"

$("a[href$='.com']").css("color", "blue"); // Выбор всех элементов "a", с атрибутом "href", который заканчивается на ".com"

$("a[href\*='html']").css("color", "green"); // Выбор всех элементов "a", с атрибутом "href", который содержит "html"

});

</script>

<style>

span {

display: inline-block;

width: 80px;

}

</style>

</head>

<body>

<a href="http://edu.cbsystematics.com/video/ru">Видео обучение</a><br />

<a href="http://jquery.com">jQuery</a><br />

<a href="http://htmlbook.ru/">HTML-book</a>

</body>

</html>

## Samples

### Eq method

<title>Выборка элементов с помощью jQuery</title>

<script src="jquery-1.10.2.js"></script>

<script>

// eq() - метод, который позволяет выбрать нужный элемент из массива по индексу.

jQuery(document).ready(function () {

$('p').eq(0).text("$('p') - выборка по имени тега");

$('[name = "input\_background"]').css('background', 'yellow');

$('.bold\_paragraph').css('fontWeight', 'bold');

$('#red\_paragraph').css('color', 'red');

});

</script>

</head>

<body>

<p></p>

<input type="text" name="input\_background" />

<p class="bold\_paragraph"><u>$('.bold\_paragraph')</u> - выборка по имени класса.</p>

<p id="red\_paragraph"><u>$('#red\_paragraph')</u> - выборка по идентификатору</p>

</body>

</html

### Query selector method

<title>Выборка элементов с помощью методов querySelector() и querySelectorAll()</title>

<script>

window.addEventListener("load", function () {

// querySelector() - получение первого элемента на странице, который соответствует селектору.

// В данном случае возвращаем элемент p.

var paragraph = document.querySelector("p");

paragraph.style.fontSize = "30px";

// querySelectorAll() - возвращает массив элементов, которые соответствуют селектору.

// В данном случае возвращается массив элементов списка, находящихся в элементе ul с атрибутом class="list"

var list = document.querySelectorAll(".list > li");

for (var i = 0; i < list.length; i++) {

list[i].style.color = "blue";

}

});

</script>

</head>

<body>

<p>Метод querySelector()</p>

<p>Параграф без стилей</p>

<ul class="list">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

</ul>

</body>

</html>

### On method

ad>

<title>Добавление классов к елементам с помощью jQuery</title>

<script src="jquery-1.10.2.js"></script>

<script>

// on() - Добавление обработчика событий текущему выбранному элементу. Он обеспечивает все функциональные возможности, необходимые для присоединения обработчиков событий.

// addClass() - Добавление указанного класса(ов) к каждому из набора соответствующих элементов.

// removeClass() - Удаляет классы в наборе соответствующих элементов.

$(document).ready(function () {

$('#first').on('click', firstButtonHandler);

$('#second').on('click', secondButtonHandler);

var div = $('#testDiv');

function firstButtonHandler() {

div.removeClass();

div.addClass("class1");

}

function secondButtonHandler() {

div.removeClass();

div.addClass("class2");

}

});

</script>

<style>

.class1 {

font-family: Tahoma;

color: Blue;

}

.class2 {

font-family: Courier New;

color: Red;

}

</style>

</head>

<body>

<input id="first" type="button" value="First" />

<input id="second" type="button" value="Second" />

<div id="testDiv">

Some text

</div>

</body>

# Filters

## Basic filters

### EvenOdd

<title>Выбор четных и нечетных элементов</title>

<script src="jquery-1.10.2.js"></script>

<script>

// $("E:even"); - выбор всех четных по значению индекса элементов.

// $("E:odd"); - выбор всех нечетных по значению элементов.

$(function () {

$("div p:even").css("background-color", "red");

//$("p:odd").css("background-color", "red");

});

</script>

<style>

body {

background-color: #44e3ff;

color: #fff;

}

p {

padding: 10px;

font-weight: bold;

}

</style>

</head>

<body>

<div>

<p>Параграф 0</p>

<p>Параграф 1</p>

<p>Параграф 2</p>

<p>Параграф 3</p>

<p>Параграф 4</p>

<p>Параграф 5</p>

<p>Параграф 6</p>

<p>Параграф 7</p>

<p>Параграф 8</p>

</div>

</body>

### FirstLast

<title>Фильтры :first и :last</title>

<script src="jquery-1.10.2.js"></script>

<script>

// $("E:first") - выбор первого совпадения на странице.

// $("E:last") - выбор последнего совпадения на странице.

$(function () {

$("li:first").css("color", "blue");

$("li:last").css("color", "red");

});

</script

### FirstChildLastChild

<title>Фильтры :first-child и :last-child</title>

<script src="jquery-1.10.2.js"></script>

<script>

// $("E:first-child") - выбор первого дочернего элемента.

// $("E:last-child") - выбор последнего дочернего элемента.

$(function () {

$("li:first-child").css("color", "blue");

$("li:last-child").css("color", "red");

});

</script>

</head>

<body>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

<li>Item 5</li>

</ul>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

<li>Item 5</li>

</ul>

</body>

### Only child

<title>Фильтр :only-child</title>

<script src="jquery-1.10.2.js"></script>

<script>

// $("E:only-child") - выбор единственных дочерних элементов.

$(function () {

$("div p:only-child").css("color", "blue");

});

</script>

<style>

div {

border: 1px solid;

margin-bottom: 10px;

padding: 10px;

}

</style>

</head>

<body>

<div>

<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.</p>

</div>

<div>

<p>Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem. Investigationes demonstraverunt lectores legere me lius quod ii legunt saepius.</p>

<p>Claritas est etiam processus dynamicus, qui sequitur mutationem consuetudium lectorum. Mirum est notare quam littera gothica, quam nunc putamus parum claram, anteposuerit litterarum formas humanitatis per seacula quarta decima et quinta decima. Eodem modo typi, qui nunc nobis videntur parum clari, fiant sollemnes in futurum.</p>

</div>

</body>

### NthChild

<script>

/\*

Фильтр :nth-child используется для добавления стиля к элементам на основе нумерации в дереве элементов.

:nth-child(n) - Выбирает n-ый дочерний элемент

:nth-child(even) - Выбирает все четные дочерние элементы

:nth-child(odd) - Выбирает все нечетные дочерние элементы

:nth-child(an+b) - В круглых скобках выражение, где a и b целые числа, а n — счетчик, который автоматически принимает значение 0,1,2...

\*/

$(function () {

$(".one tr:nth-child(7)").css("background-color", "lightblue");

$(".two tr:nth-child(even)").css("background-color", "lightblue");

$(".three tr:nth-child(odd)").css("background-color", "lightblue");

$(".four tr:nth-child(3n+1)").css("background-color", "lightblue");

});

</script>

<style>

.one, .two, .three, .four {

border: 1px solid;

border-collapse: collapse;

width: 300px;

margin-bottom: 30px;

text-align: center;

}

td, th {

border: 1px solid;

}

</style>

</head>

### Select less then index or more

<title>Фильтры eq(), gt(), lt()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// :eq(индекс) - выбор элемента по индексу.

// :gt(индекс) - все элементы, индекс которых больше указаного.

// :lt(индекс) - все элементы, индекс которых меньше указаного.

$(function () {

$("p:eq(3)").css("background-color", "#4dea4d"); // (equal) выделит параграф с индексом 3

//$("p:gt(4)").css("background-color", "#4dea4d"); // (greater than) выделит все параграфы от элемента с индексом 4 (4 > ...).

//$("p:lt(4)").css("background-color", "#4dea4d"); // (less than) выделит все параграфы до элемента с индексом 4 (4 < ...).

});

</script>

</head

### Eq example

// eq(n) - выбирает элемент с индексом n из набора. Индекс начинается с 0.

var i = 0;

$(function () {

// установим интервал в одну секунду

var timer = setInterval(choose, 1000);

});

function choose() {

i++;

// выделит все элементы списка до элемента с индексом i

$("li:eq(" + (i - 1) + ")")

.css("background-color", "white");

// выделит элементы списка по индексу i

$("li:eq(" + i + ")")

.css("background-color", "yellow");

}

</script>

<style>

ol li:first-child {

background-color: yellow;

}

</style>

</head>

<body>

<ol>

<li>Some item</li>

<li>Some item</li>

<li>Some item</li>

<li>Some item</li>

<li>Some item</li>

</ol>

### Not(n)

<script>

// :not(n) - выберет все элементы из массива выборки кроме тех, которые соответствуют "n".

$(function () {

// выберет все элементы из массива выборки кроме элемента с индексом 3.

$("li:not(li:eq(3))").css("background-color", "yellow");

});

</script>

</head>

### Header

<script>

// :header - получение всех заголовков.

$(function () {

//$("h1, h2, h3, h4, h5, h6").css("background-color", "#4dea4d");

$(":header").css("background-color", "#4dea4d");

});

</script>

</head>

<body>

<h1>Header 1</h1>

<h2>Header 2</h2>

<h3>Header 3</h3>

<h4>Header 4</h4>

<h5>Header 5</h5>

<h6>Header 6</h6>

</body>

### Animated

<script>

// $(":animated") - возвращает элементы, в которых происходит анимация.

// метод hide(n) - скрывает выбранные элементы за n секунд.

$(function () {

$(".last").hide(3000);

$(":animated").css("background", "red");

});

</script>

<style>

div {

width: 200px;

height: 200px;

border: 1px solid;

float: left;

margin-right: 30px;

}

</style>

</head>

<body>

<div></div>

<div></div>

<div class="last"></div>

</body>

## Content Filters

### Contains

<title>Фильтр :contains('text')</title>

<script src="jquery-1.10.2.js"></script>

<script>

//:contains('text') - фильтр, который выбирает все элементы, в которых встречается указанный текст.

$(function () {

$("p:contains('JavaScript')").css("background-color", "red");

//$("p:contains('HTML')").css("background-color", "red");

});

</script>

</head>

<body>

<p>JavaScript — прототипно-ориентированный сценарный язык программирования. Является диалектом языка ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.</p>

<p>HTML (от англ. HyperText Markup Language — «язык разметки гипертекста»;) — стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме. HTML является приложением («частным случаем») SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879. XHTML же является приложением XML.</p>

</body>

### Empty

<script src="jquery-1.10.2.js"></script>

<script>

//:empty - выбирает все элементы в которых отсутствует содержимое

$(function () {

$("div:empty").css("background-color", "Red");

});

</script>

<style>

div {

width: 200px;

height: 100px;

background-color: azure;

border: 1px solid gray;

margin-bottom: 10px;

}

</style>

</head>

<body>

<div>Some content</div>

<div></div>

<div>Some content</div>

## Attribute filters

### Attribute name equals value

<title>Фильтр атрибутов</title>

<script src="jquery-1.10.2.js"></script>

<script>

// $("E[AttrName~='value']"); - выбор всех элементов E с атрибутом AttrName содержащим value в списке значений, разделенных пробелом

$(function () {

$("img[alt~='new']").css("border", "4px solid #005cf2");

});

</script>

</head>

<body>

<img src="images/cat.jpg" alt="new\_cat" />

<img src="images/cat.jpg" alt="new cat" />

<img src="images/cat.jpg" alt="new-cat" />

</body>

<script src="jquery-1.10.2.js"></script>

<script>

// $("E[AttrName!='value']"); - выбор всех элементов E с атрибутом AttrName не равного value

$(function () {

$("a[target=\_blank]").css("background-color", "#9bff70");

//$("a[target!=\_blank]").css("background-color", "#9bff70");

});

</script>

</head>

<body>

<a href="http://edu.cbsystematics.com/" target="\_blank">Live education</a><br />

<a href="http://edu.cbsystematics.com/video/ru">Video education</a>

</body>

</html>

## Form filters

### Button filter

<title>Фильтр :button</title>

<script src="jquery-1.10.2.js"></script>

<script>

// :button - фильтр, который выбирает все кнопки на странице.

$(function () {

$(":button").css("font-weight","bold");

});

</script>

</head>

<body>

<form>

Login <input type="text" value="" /><br />

Password <input type="text" value="" /><br />

<input type="submit" value="Send" /><br />

<input type="button" value="Show" />

</form>

</body>

### Checkbox filters

<script src="jquery-1.10.2.js"></script>

<script>

// :checkbox - фильтр для выбора всех элементов "checkbox".

$(function () {

var count = $(":checkbox").length;

$(".output").text("Количество элементов \"checkbox\" на странице - " + count);

});

</script>

</head>

### Checked

<script>

// Свойство length - количество элементов в массиве.

// :checked фильтр выбирает только отмеченные флажки или переключатели

$(function () {

var countChecked = function () {

var n = $("input:checked").length;

$(".output").text("Пунктов отмечено - " + n);

};

countChecked();

$("input[type=checkbox]").on("click", countChecked);

});

</script>

</head>

<body>

<form>

<p><strong>Выбор пунктов:</strong></p>

Пункт 1

<input type="checkbox" value="p1" /><br />

Пункт 2

<input type="checkbox" value="p2" /><br />

Пункт 3

<input type="checkbox" value="p3" /><br />

Пункт 4

<input type="checkbox" value="p4" /><br />

<p class="output"></p>

</form>

### Disabled

<script>

// :disabled фильтр выбирает только элементы форм, находящиеся в неактивном состоянии

$(function () {

$("input:disabled").val("disabled");

});

</script>

</head>

<body>

<form>

<input type="text" disabled="disabled" />

<input type="text" />

</form>

</body>

### Enabled

<title>Фильтр :enabled</title>

<script src="jquery-1.10.2.js"></script>

<script>

// :enabled - фильтр выбирает только элементы форм, находящиеся в активном состоянии

$(function () {

$("input:enabled").val("enabled");

});

</script>

</head>

<body>

<form>

<input type="text" disabled="disabled" />

<input type="text" />

</form>

</body>

### File filter

<title>Фильтр :file</title>

<script src="jquery-1.10.2.js"></script>

<script>

// :file - выбирает все элементы типа "file"

$(function () {

$(":file").css("border", "1px solid red");

});

</script>

<style>

input {

display: block;

}

</style>

</head>

<body>

<form>

Name <input type="text" />

Surname <input type="text" /><br />

<input type="file" />

<input type="submit" />

</form>

</body>

# Work with attributes, properties and styles

## Attributes

### Attr

<title>Получение значения атрибутов</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод attr() позволяет получить доступ к HTML атрибуту первого тэга на странице, который был выбран селектором.

// Если элемент не имеет такого атрибута, то возвращается значение undefined.

// Также есть возможность добавить новый атрибут с помощью даного метода.

$(function () {

// Получение значения атрибута style у первого элемента, который найден селектором "p"

var style = $("p").attr("style");

var p = $("#output");

p.text("<b>Значение атрибута \"style\"</b> - " + style);

});

</script>

</head>

<body>

<p style="color: blue; font-weight: bold;">

Test string

</p>

<div id="output">

</div>

</body>

### Atrr key value

<script src="jquery-1.10.2.js"></script>

<script>

// Добавление/изменение атрибута с помощью метода attr().

// attr("имя\_атрибута","значение");

$(function () {

// Установка значеня атрибута "class" у всех параграфов

$('input').on("click", changeColor);

});

function changeColor()

{

$("p").attr("class", "testClass");

}

</script>

<style>

.testClass {

color: Red;

}

</style>

</head>

<body>

<input type="button" value="Ok ">

<p>

Test string 1

</p>

<p class="testClass">

Test string 2

</p>

<p>

Test string 3

</p>

</body>

</html>

### Attr Json

<script src="jquery-1.10.2.js"></script>

<script>

// В методы JQuery можно передавать JSON-объекты. С ними намного проще работать.

$(function () {

// Передача в метод attr() JSON-объекта

$("p").attr({

style: "color:red; font-size:22pt",

title: "JQuery Attr Test"

});

});

</script>

</head>

<body>

<p>

Test string

</p>

<p>

Test string 2

</p>

</body>

### Attr function

<script src="jquery-1.10.2.js"></script>

<script>

// Установка значения атрибута style через вызов функции у первого тэга! <p>

$(function () {

$("p:first").attr("style", CalculateValue());

});

function CalculateValue() {

return "color:red";

}

</script>

</head>

<body>

<p>

Test string 1

</p>

<p>

Test string 2

</p>

<p>

Test string 3

</p>

</body>

### Atrr remove

<script src="jquery-1.10.2.js"></script>

<script>

// Метод removeAttr() - удаляет указаный атрибут элемента.

$(function () {

// Удаление атрибута style у всех нечетных элементов <p>

$("p:odd").removeAttr("style");

});

</script>

</head>

<body>

<p style="color: Blue;">

Test string 1

</p>

<p style="color: Blue;">

Test string 2

</p>

<p style="color: Blue;">

Test string 3

</p>

<p style="color: Blue;">

Test string 4

</p>

</body>

### Prop

<script src="jquery-1.10.2.js"></script>

<script>

// http://javascript.ru/tutorial/dom/attributes - рекомендовано к прочтению

// Метод prop() - возвращает или изменяет значение свойств выбранных элементов страницы.

// .prop(propName, value) - во всех выбранных элементах, свойство propName примет значение value.

// .prop({propName1:value1, propName2:value2, ...}) - во всех выбранных элементах изменит значения группы свойств propName1, propName2, ... сделав их равными value1, value2, ... соответственно.

// .prop(propName, function(index, value)) - свойству propName будет присвоено значение, возвращенное пользовательской функцией (если она ничего не вернет, то свойство просто не изменит своего значения). Функция вызывается отдельно, для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, value — текущее значение свойства propName у элемента.

// Метод html() - получает HTML-содержимое первого элемента в наборе соответствующих элементов или устанавливает HTML-разметку для каждого совпавшего элемента.

$(function () {

$("input").change(function () {

var $input = $(this);

$("p").html(

".attr( \"checked\" ): <b>" + $input.attr("checked") + "</b><br>" +

".prop( \"checked\" ): <b>" + $input.prop("checked") + "</b><br>") + "</b>";

}).change();

});

</script>

<style>

p {

margin: 20px 0 0;

}

b {

color: cadetblue;

}

</style>

</head>

<body>

<label for="check1">Сохранить пароль? </label>

<input id="check1" type="checkbox" checked="checked" title="Сохранить/не сохранять пароль">

<p></p>

### RemoveProp

<title>Метод removeProp()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод removeProp() - удаляет свойства у выбранных элементах страницы.

// Замечание: используйте этот метод для удаления искусственно введенных свойств, не стоит удалять нативные свойства элементов (т.е. первоначальные, родные свойства элементов), это может вызвать ошибки работы браузера. В частности, не используйте .removeProp() для удаления таких атрибутов как checked и disabled. Когда эти атрибуты отсутствуют в input-элементах, их одноименные свойства не перестают существовать, они просто принимают значение false.

// Метод append() - добавляет контент внутрь каждого элемента набора. Добавляемый контент следует за уже существующим. Данный метод подобен применению appendChild.

$(function () {

var output = $("p");

output.prop("secretCode", 1234);

output.append("Секретный код: ", String(output.prop("secretCode")), ". ");

output.removeProp("secretCode");

output.append("Теперь: ", String(output.prop("secretCode")), ". ");

});

</script>

</head>

<body>

<p></p>

</body>

### Val

<title>Метод val()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод val() - позволяет получать и изменять значения элементов форм.

// .val() - возвращает значение атрибута value у выбранного элемента формы. Если выбрано несколько элементов, то значение будет взято у первого. В случае элемента формы <select multiple="multiple"> возвращается массив всех выбранных значений.

// .val(newVal) - атрибуту value будет присвоено значение newVal, у всех выбранных элементов.

// .val(function(index, newVal) - атрибуту value будет присвоено значение, возвращенное пользовательской функцией. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове, ей передаются следующие параметры: index — позиция элемента в наборе, newVal — текущее значение атрибута value у элемента.

$(function () {

//Нажатие на первую кнопку

$("#button1").click(function () {

$("#edit1").val("Hello world"); // Запись в поле ввода.

});

//Нажатие на вторую кнопку

$("#button2").click(function () {

alert($("#edit1").val()); // Чтение из поля ввода.

});

});

</script>

</head>

<body>

<input type="text" id="edit1" />

<input type="button" id="button1" value="Push me!" />

<input type="button" id="button2" value="Get Text" />

</body>

### Add Class

<script src="jquery-1.10.2.js"></script>

<script>

// Метод addClass() - добавляет класс(ы) к выбранным элементам страницы и не заменяет уже существующие.

// .addClass("clName1 clName2 ...") - добавляет классы clName1, clName2, ... всем выбранным элементам.

// .addClass(function(index, class)) - добавляет классы, список которых возвращает заданная пользователем функция. Возвращаемое ей значение должно представлять из себя строку, с перечислением добавляемых классов через пробел. Функция вызывается отдельно для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, class — текущий класс(ы) элемента.

$(function () {

// Добавление класса к элементам с идентификаторами First, Second и Third.

$("#First").addClass("testClassGreen");

$("#Second").addClass("testClassRed");

$("#Third").addClass("testClassBlue");

});

</script>

<style>

.testClassGreen {

color: Green;

}

.testClassRed {

color: Red;

}

.testClassBlue {

color: Blue;

}

</style>

</head>

<body>

<p id="First">

Test string 1

</p>

<p id="Second">

Test string 2

</p>

<p id="Third">

Test string 3

</p>

</body>

### Remove Class

<title>Метод removeClass()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод removeClass() - удаляет заданные классы у элементов на странице.

// .removeClass() - удаляет все классы у выбранных элементов.

// .removeClass("class1 class2 ...") - удаляет классы class1, class2, ... у выбранных элементов.

// .removeClass(function(index, class)) - удаляет классы, список которых возвращает заданная пользователем функция. Возвращаемое ей значение должно представлять из себя строку с перечислением классов через пробел. Функция вызывается отдельно для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, class — текущий класс(ы) элемента.

$(function () {

$(".class1").removeClass("class1");

});

</script>

### Toggle class

<script>

// Метод toggleClass() - добавляет или удаляет заданный класс(ы) по принципу переключателя (добавляет, если элемент не содержит класса, и удаляет, если класс есть).

// .toggleClass("class1 class2 ...") - соответствует поведению описанному выше.

// .toggleClass("class1 class2 ...", switch ) - только добавляет или только удаляет классы в зависимости от параметра switch. Параметр имеет тип boolean. true — в случае добавления класса(ов), false — иначе.

// .toggleClass( function(index, class) ,[switch] ) - "переключает" классы, список которых возвращает заданная пользователем функция. Возвращаемое ей значение должно представлять из себя строку, с перечислением классов через пробел. Функция вызывается для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, class — текущий класс(ы) элемента. Параметр switch играет такую же роль, как и в предыдущем случае.

$(function () {

$("#toggle").on("click", toggle);

function toggle() {

$("p").toggleClass("class1");

}

});

</script>

### Has Class

<script src="jquery-1.10.2.js"></script>

<script>

// hasClass(className) - проверяет наличие класса "className" у выбранных элементов страницы. Если хотя бы один из элементов содержит этот класс, то функция вернет true, иначе вернет false.

$(function () {

var className = "class1";

if ($("p:first").hasClass(className)) {

alert("Тэг содержит указанный CSS-класс - " + className);

}

else {

alert("Тэг не содержит указанный CSS-класс");

}

});

</script>

## Css

### Css()

<title>Метод css()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод css() - возвращает или изменяет значения css-величин у выбранных элементов страницы.

// .css(styleName) - возвращает значение css-свойства styleName у выбранного элемента. Если выбрано несколько элементов, то значение будет взято у первого.

// .css(styleName, value) - css-свойству styleName будет присвоено значение value, у всех выбранных элементов.

// .css({styleName1: value1, styleName2: value2, ...}) - группе css-свойств styleName1, styleName2, ... будут присвоены значения value1, value2, ..., у всех выбранных элементов.

// .css(styleName, function(index, value)) - css-свойству styleName будет присвоено значение, возвращенное пользовательской функцией. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове, ей передаются следующие параметры: index — позиция элемента в наборе, value — текущее значение css-свойства styleName у элемента.

$(function () {

$("p").css("color", "green");

});

</script>

</head>

<body>

<p>

Test string 1

</p>

<p>

Test string 2

</p>

<p>

Test string 3

</p>

// .css({styleName1: value1, styleName2: value2, ...}) - группе css-свойств styleName1, styleName2, ... будут присвоены значения value1, value2, ..., у всех выбранных элементов.

// .css(styleName, function(index, value)) - css-свойству styleName будет присвоено значение, возвращенное пользовательской функцией. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове, ей передаются следующие параметры: index — позиция элемента в наборе, value — текущее значение css-свойства styleName у элемента.

$(function () {

$("p").css(

{

color: "green",

backgroundColor: "lightgray"

});

});

</script>

### Height

<script src="jquery-1.10.2.js"></script>

<script>

/\* Методы .height() .innerHeight() .outerHeight([includeMargin]) - все три функции возвращают высоту первого из выбранных элементов страницы.

height() — высота элемента без учета внутренних отступов и толщины рамки.

innerHeight() — высота элемента с учетом размера внутренних отступов (padding).

outerHeight(includeMargin) — высота элемента с учетом внутренних отступов, рамки (border-width) и при необходимости внешних отступов (includeMargin == true) \*/

// .height(value) - устанавливает новое значение высоты равное value, для всех выбранных элементов.

// .height(function(index, value)) - устанавливает новое значение высоты элементов, равное значению, которое вернет пользовательская функция. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, value — текущее значение высоты элемента.

$(function () {

var a = "";

a += "heigh() - " + $("div").height() + "<br />";

a += "innerHeight() - " + $("div").innerHeight() + "<br />";

a += "outerHeight() - " + $("div").outerHeight() + "<br />";

a += "outerHeight(true) - " + $("div").outerHeight(true) + "<br />";

$("p").html(a);

//$("div").height(400);

});

</script>

<style>

div {

width: 300px;

background: green;

padding: 10px;

margin: 15px;

color: white;

border: 5px solid yellow;

text-align: justify;

}

</style>

### Width

<title>Методы .width() .innerWidth() .outerWidth([includeMargin])</title>

<script src="jquery-1.10.2.js"></script>

<script>

/\* Методы .width() .innerWidth() .outerWidth([includeMargin]) - все три функции возвращают ширину первого из выбранных элементов страницы. Отличия у функций следующие:

width() — ширина элемента без учета отступов и толщины рамки.

innerWidth() — ширина элемента с учетом размера внутренних отступов (padding).

outerWidth(includeMargin) — ширина элемента с учетом внутренних отступов, рамки (border-width) и при необходимости внешних отступов (includeMargin == true) \*/

// .width(value) - устанавливает новое значение ширины равное value, для всех выбранных элементов.

// .width(function(index, value)) - устанавливает новое значение ширины элементов, равное значению, которое вернет пользовательская функция. Функция вызывается отдельно, для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, value — текущее значение ширины элемента.

$(function () {

var a = "";

a += "width() - " + $("div").width() + "<br />";

a += "innerWidth() - " + $("div").innerWidth() + "<br />";

a += "outerWidth() - " + $("div").outerWidth() + "<br />";

a += "outerWidth(true) - " + $("div").outerWidth(true) + "<br />";

$("p").html(a);

//$("div").width(400);

});

</script>

<style>

div {

width: 300px;

background: green;

padding: 10px;

margin: 15px;

color: white;

border: 5px solid yellow;

text-align: justify;

}

</style>

### Offset

<title>Метод offset()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод offset() - возвращает координаты выбранного элемента относительно начала страницы. Также с помощью offset(), можно изменить координаты элемента.

// offset(value) - изменяет координаты всех выбранных элементов делая их равными value. Значение value должено быть объектом с двумя свойствами — {top:newTop, left:newLeft}.

// offset(function(index, value)) - устанавливает новое значение координат элементов, которое возвращает пользовательская функция. Функция вызывается отдельно для каждого из выбранных элементов. При вызове ей передаются следующие параметры: index — позиция элемента в наборе, value — текущие координаты элемента.

// Замечание 1: при изменении координат с помощью функции offset, все выбранные элементы, у которых не задан тип позиционирования (то есть position = static), автоматически изменят позиционирование на относительное (relative).

// Замечание 2: важно отметить, что используя метод .offset() вы получите позицию только первого элемента из всех выбранных.

$(function () {

var element = $("div").offset();

var output = $("div");

output.html("Мои координаты относительно окна браузера:<br />" + "x: " + element.left + "<br />y: " + element.top);

});

</script>

<style>

\* {

margin: 0;

padding: 0;

font-family: Arial, 'DejaVu Sans', 'Liberation Sans', Freesans, sans-serif;

font-size: 12px;

}

div {

background-color: #7b91e5;

width: 270px;

height: 200px;

position: absolute;

top: 50px;

left: 200px;

padding: 5px;

}

</style>

</head>

<body>

### Position

<script src="jquery-1.10.2.js"></script>

<script>

// Метод position() - возвращают координаты выбранного элемента относительно ближайшего родителя, у которого задан тип позиционирования (css-свойство position равно relative или absolute или fixed).

$(function () {

var element = $("#child").position();

var output = $("#output");

output.html("Мои координаты относительно родителя:<br />" + "x: " + element.left + "<br />y: " + element.top);

});

</script>

<style>

\* {

margin: 0;

padding: 0;

font-family: Arial, 'DejaVu Sans', 'Liberation Sans', Freesans, sans-serif;

font-size: 12px;

}

/\*#main\_parent{

background-color: #f63737;

width: 700px;

height: 500px;

position: relative;

left: 300px;

top: 50px;

padding: 5px;

}\*/

#parent {

background-color: #c7cad6;

width: 500px;

height: 400px;

left: 500px; /\* Заменить на margin-left: 50px \*/

top: 100px; /\* Заменить на margin-top: 10px \*/

position: relative; /\* Закомментировать (Ctrl + k + c) \*/

padding: 5px;

}

#child {

background-color: #7b91e5;

width: 270px;

height: 200px;

position: absolute;

top: 50px;

left: 200px;

padding: 5px;

}

</style>

</head>

<body>

<!--<div id="main\_parent">-->

<div id="parent">

родитель

<div id="child">

<p id="output"></p>

</div>

</div>

<!--</div>-->

</body>

### Offset parent

<script src="jquery-1.10.2.js"></script>

<script>

// Метод offsetParent() - возвращает ближайшего предка (т.е. из числа: родитель, прародитель, и так далее, до начало дерева DOM), у которого задан тип позиционирования (position равен relative или absolute или fixed).

// Элементы с заданным позиционированием играют особую роль, поскольку именно относительно них происходит позиционирование всех элементов внутри.

$(function () {

$(".inner").offsetParent().css('background-color', 'yellow');

});

</script>

<style>

div {

border: 1px solid black;

}

.inner {

width: 30px;

height: 30px;

font-size: 20pt;

padding: 7px;

margin: 7px;

background-color: white;

}

.outer {

position: relative;

width: 100px;

height: 100px;

margin: 15px;

background-color: white;

}

.wrap {

width: 60px;

height: 60px;

margin: 18px;

background-color: white;

}

</style>

</head>

<body>

<div class="outer">

<div class="wrap">

<div class="inner">

B

</div>

</div>

</div>

</body>

# DOM

## Filters

### Eq()

<script src="jquery-1.10.2.js"></script>

<script>

// Метод eq(n) - возвращает элемент, идущий под индексом 'n' в наборе выбранных элементов.

$(function () {

$("div").eq(2).addClass("blue");

});

</script>

<style>

div {

width: 60px;

height: 60px;

margin: 10px;

float: left;

border: 2px solid blue;

}

.blue {

background: blue;

}

</style>

</head>

<body>

<div></div>

<div></div>

<div></div>

<div></div>

<div></div>

<div></div>

### Not()

title>Метод not()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод not(n) исключает из набора те элементы, которые соответствуют селектору 'n'.

$(function () {

$('li').not(':eq(2)').css("color","red");

});

</script>

</head>

<body>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

</ul>

</body>

</html>

### Is()

title>Метод is()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// is(n) - проверяет текущий набор элементов на соответствие указанному выражению 'n'. Возвращает true, если хотя бы один элемент соответствует выражению.

$(function () {

$("div").on("click", function () {

if ($(this).is(":first-child")) {

$("p").text("Это первый div.");

} else if ($(this).is(".blue,.red")) {

$("p").text("Это красный или синий div.");

} else if ($(this).is(":contains('Test')")) {

$("p").text("Содержит текст TEST!");

} else {

$("p").html("тут нет ничего особенного.");

}

});

});

</script>

<style>

div {

width: 60px;

height: 60px;

margin: 5px;

float: left;

border: 4px outset;

background: green;

text-align: center;

font-weight: bolder;

cursor: pointer;

}

.blue {

background: blue;

}

.red {

background: red;

}

span {

color: white;

font-size: 16px;

}

p {

color: red;

font-weight: bolder;

background: yellow;

margin: 3px;

clear: left;

}

</style>

</head>

<body>

<div></div>

<div class="blue"></div>

<div></div>

<div class="red"></div>

<div>

<br />

<span>Test</span>

</div>

<div class="blue"></div>

<p>&nbsp;</p>

### Filter

<script src="jquery-1.10.2.js"></script>

<script>

// Метод filter(selector) - в качестве параметра может быть передан селектор, используемый для получения соответствующих ему элементов.

$(function () {

var paragraphs = $("p");

paragraphs.filter(".test").css("background-color", "red");

paragraphs.filter(":odd").css("background-color", "green");

});

</script>

<style>

body {

background-color: Silver;

color: White;

}

p {

padding: 10px;

font-weight: bold;

background-color: Gray;

}

</style>

</head>

### Filter function

<title>Метод filter(function)</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод filter(function). Если передать методу filter() некоторую функцию, он вызовет ее для каждого элемента в наборе и исключит те элементы, для которых эта функция вернет значение false.

// Эта функция получает доступ к текущему элементу набора через $(this).

$(function () {

// В качестве параметра "index" функция будет получать порядковый номер обрабатываемого элемента.

// если функция возвращает значение true - элемент проходит проверку и попадает в результирующий набор данных

// $(this) - обращение к текущему элементу а функции

$("p").filter(function (index) {

if ($(this).attr("align") && index == 0) {

return true;

}

}).css("background-color", "red");

});

</script>

<style>

### Slice

<title>Метод slice(start,end) </title>

<script src="jquery-1.10.2.js"></script>

<script>

// slice(start,end) - выделяет подмножество из набора совпавших элементов.

// Ведет себя точно так же как и встроенный метод разделения массивов.

// Выделение подмножества начинается с индекса "start" (включительно) по индекс "end" (не включительно).

$(function () {

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

$("p").css("background-color", "green");

$("p")

.slice(0, 3)

.css("background-color", "red");

});

$("#btn2").click(function () {

$("p").css("background-color", "blue");

$("p")

.slice(2, 4)

.css("background-color", "red");

});

$("#btn3").click(function () {

$("p").css("background-color", "green");

$("p")

.slice(3)

.css("background-color", "red");

});

$("#btn4").click(function () {

$("p").css("background-color", "green");

$("p")

.slice(3, -3)

.css("background-color", "red");

});

});

</script>

## Traversing

### Add

<title>Метод add(selector)</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод add(selector) - добавляет в уже существующий набор дополнительные элементы, которые соответсвуют указанному выражению.

$(function () {

$("p") // получаем все параграфы

.add("span") // добавляем к выборке span

.css("background-color", "Red");

});

</script>

<style>

body {

background-color: Silver;

color: White;

}

p {

padding: 10px;

font-weight: bold;

background-color: Gray;

}

</style>

</head>

<body>

<p>Test paragraph 1</p>

<p>Test paragraph 2</p>

<p>Test paragraph 3</p>

<p>Test paragraph 4</p>

<span>First Span</span>

<span>Second Span</span>

### Find

<script src="jquery-1.10.2.js"></script>

<script>

// Метод find(selector) - отыскивает дочерние элементы, соответствующие указанному выражению.

$(function () {

// находим все параграфы и ищем в них элементы span

$("p").find("span").css('color', 'red');

});

</script>

</head>

<body>

<p>Тестовый параграф с элементами <span>span</span> и <span>span</span></p>

<span>Просто span</span>

### Contens

<script src="jquery-1.10.2.js"></script>

<script>

// Метод contents() - получает потомков каждого элемента в наборе соответствующих элементов, включая текстовые узлы и узлы комментариев.

$(function () {

//Превратим текстовые узлы внутри #content в параграфы (текстовые узлы: nodeType = 3)

$("#content").contents().filter(function () {

return this.nodeType === 3;

}).wrap("<p></p>");

//Покрасим выше созданные параграфы внутри #content в красный цвет.

$("#content p").css("color", "red");

});

</script>

</head>

<body>

<div id="content">

lorem ipsum

<span>lorem ipsum in span</span>

</div>

### Children

<title>Метод children()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод children(selector) - получает набор элементов, содержащий всех прямых потомков для каждого совпавшего элемента.

$(function () {

$(".class1")

.children("p") // получить все дочерние элементы p для элемента с классом class1

.css("background-color", "Red");

});

</script>

</head>

### Hover

<script src="jquery-1.10.2.js"></script>

<script>

// parent() - возвращает ссылку на ближайшего родителя.

// hover(over,out) - метод, который имитирует эффект hovering (отслеживание попадания указателя мыши в пределы объекта). Параметры метода: over - функция при наведении, out - функция при отведении курсора мыши.

$(function () {

$("span").hover(

// функция срабатывает при наведении курсора на элемент

function () {

$(this).css("background-color", "Yellow")

.parent()

.css("background-color", "Red");

},

// функция срабатывает при выходе курсора за пределы элемента

function () {

$(this).css("background-color", "White")

.parent()

.css("background-color", "White");

});

});

</script>

</head>

<body>

<div>

<p>

Для изменения цвета этого параграфа нужно навести курсор мыши на этот <span>Span</span>

</p>

### Parents

<script src="jquery-1.10.2.js"></script>

<script>

/\* parents() - выбирает все родительские элементы (узлы) для ранее выбраных элементов. Принимает в качестве аргумента

дополнительный фильтр-селектор.\*/

$(function () {

$("span")

.parents("div")

.css("background-color", "Yellow");

//$("span")

// .parents()

// .css("background-color", "Yellow");

});

</script>

### Parents()

<script src="jquery-1.10.2.js"></script>

<script>

$(function () {

/\* parents() - выбирает все родительские элементы (узлы) для ранее выбраных элементов. Принимает в качестве аргумента дополнительный фильтр-селектор.\*/

function showParents() {

$("div").css("border-color", "white");

var len = $("span.selected")

.parents("div")

.css("border", "2px red solid")

.length;

$("b").text("Уникальных родителей (элементов div): " + len);

}

function customHover() {

$(this).toggleClass("selected");

showParents();

}

$("span").hover(customHover);

});

</script>

### Closet

script src="jquery-1.10.2.js"></script>

<script>

// Метод closest() - получает набор, содержащий ближайшие родительский элемент, которые соответствуют указанному селектору.

$(function () {

$("span").on("click", function () {

$(this).closest("div").toggleClass("hilight");

});

});

</script>

<style>

.outer {

padding: 20px;

background: red;

}

.inner {

padding: 20px;

background: yellow;

}

p {

padding: 5px;

background: green;

}

span {

background: pink;

}

.hilight {

background: blue;

}

</style>

</head>

<body>

<div class="outer">

<div class="inner">

<p>

<span>Show closest div</span>

</p>

</div>

</div>

### Next

<script src="jquery-1.10.2.js"></script>

<script>

// Метод next() - получает набор элементов, содержащий уникальные последующие родственные элементы для всех элементов существующего набора.

$(function () {

$("p:first").next().css("background", "yellow");

//$("p").next().css("background", "yellow");

});

</script>

</head>

<body>

<p>Paragraph 1</p>

<span>Span 1</span>

<p>Paragraph 2</p>

<span>Span 2</span>

<p>Paragraph 3</p>

<span>Span 3</span>

### NextAll

<script src="jquery-1.10.2.js"></script>

<script>

// Метод nextAll() - поиск всех родственных элементов после текущего элемента.

$(function () {

$("p.class1").nextAll().css("background-color", "Red");

//$("p.class1").nextAll("p").css("background-color", "Red");

});

</script>

<style>

body {

background-color: Silver;

color: White;

}

p {

padding: 10px;

font-weight: bold;

background-color: Gray;

}

</style>

</head>

<body>

<p>Paragraph 1</p>

<p>Paragraph 2</p>

<p class="class1">Paragraph 3</p>

<p>Paragraph 4</p>

<p>Paragraph 5</p>

<p>Paragraph 6</p>

<p>Paragraph 7</p>

<p>Paragraph 8</p>

<p>Paragraph 9</p>

<p>Paragraph 10</p>

<div>Div</div>

### NextUntil

<script src="jquery-1.10.2.js"></script>

<script>

// Метод nextUntil(selector) - Поиск всех родственных элементов после текущего элемента до указанного селектора.

$(function () {

//$(".class1").nextUntil().css("background-color", "Red");

$(".class1").nextUntil(".class2").css("background-color", "Red");

});

</script>

### Prev

<script src="jquery-1.10.2.js"></script>

<script>

// Метод prev() - получает набор элементов, содержащий уникальные предыдущие родственные элементы для всех элементов существующего набора.

$(function () {

$("p").prev().css("background", "yellow");

//$("p:last").prev().css("background", "yellow");

});

</script>

</head>

<body>

<span>Hello Again (span)</span>

<p>Hello (paragraph)</p>

<span>Hello Again (span)</span>

<p>And Again</p>

<span>Hello Again (span)</span>

<p>Hello (paragraph)</p>

### PrevAll

<script src="jquery-1.10.2.js"></script>

<script>

// Метод prevAll() - поиск всех родственных элементов перед текущим элементом

$(function () {

$("p.class1").prevAll().css("background-color", "Red");

//$("p.class1").prevAll("p").css("background-color", "Red");

});

</script>

### PrevUntil

<script src="jquery-1.10.2.js"></script>

<script>

// Метод prevUntil(selector) - поиск всех родственных элементов перед текущим элементом до указанного селектора.

$(function () {

//$(".class1").prevUntil().css("background-color", "Red");

$(".class2").prevUntil(".class1").css("background-color", "Red");

});

</script>

### Siblings

<script src="jquery-1.10.2.js"></script>

<script>

// Метод siblings() - получение всех родственных элементов.

$(function () {

var len = $(".highlight")

.siblings()

.css("color", "red")

.length;

$("b").text(len);

});

</script>

<style>

ul {

float: left;

margin: 5px;

font-size: 16px;

font-weight: bold;

}

p {

color: blue;

margin: 10px 20px;

font-size: 16px;

padding: 5px;

font-weight: bolder;

clear: both;

}

.highlight {

background: yellow;

}

</style>

</head>

<body>

<ul>

<li>One</li>

<li>Two</li>

<li class="highlight">Three</li>

<li>Four</li>

</ul>

### AdnSelf

<script src="jquery-1.10.2.js"></script>

<script>

// Метод .andSelf() добавляет предыдущий набор jQuery к текущему в пределах одной цепочки команд и возвращает объединенный набор.

$(function () {

$("div") // берем div'ы

.find("p") // находим дочерние параграфы

.andSelf() // добалвяем в выборку div'ы

.addClass("border"); // добавляем класс

//$("div").find("p").addClass("border");

});

</script>

### End

<script src="jquery-1.10.2.js"></script>

<script>

// end() - отменяет последнее деструктивное действие, возвращая тем самым набор элементов к его предыдущему состоянию (до деструктивной операции). Если никакой деструктивной операции не проводилось, то возвращается пустой набор. Под деструктивной операцией понимают операцию, которая изменяет набор элементов jQuery. То есть это любая из Traversing-функций, возвращающая объект jQuery: add, andSelf, children, filter, find, map, next, nextAll, not, parent, parents, prev, prevAll, siblings, и slice, а также clone, appendTo, prependTo, insertBefore, insertAfter, и replaceAll.

$(function () {

$("p") // находим все p

.add("div").css("border", "1px solid red") // добавляем к выборке все div и задаем рамку

.end().css("background-color", "green"); // убираем div, оставляя p и меняем фон

});

</script>

</head>

<body>

### Find example

script src="jquery-1.10.2.js"></script>

<script>

// find(selector) - отыскивает дочерние элементы, которые удовлетворяют указанному выражению.

$(function () {

// <span> When</span> <span>the</span> <span>day</span> <span>is</span> <span>short</span>

var newText = $("p") // находим все параграфы

.text() // извлекаем текст из параграфов

.split(" ") // разбиваем текст по пробелам

.join("</span> <span>"); // оборачиваем каждое слово в span

newText = "<span>" + newText + "</span>";

$("p").html(newText)

.find("span")

// на <span> устанавливаем обработчики на событие наведения/отведения курсора мыши

.hover(function () {

$(this).addClass("highlight");

},

function () {

$(this).removeClass("highlight");

})

// отменяем деструктивное действие метода find() и находим элементы, содержащие символ t. Добавляем им стили.

.end()

.find(":contains('t')")

.css(

{

"font-style": "italic",

"text-decoration": "underline"

});

});

</script>

<style>

p {

font-size: 20px;

width: 200px;

cursor: default;

color: blue;

font-weight: bold;

margin: 0 10px;

}

.highlight {

background: yellow;

}

</style>

</head>

<body>

<p>When the day is short find that which matters to you or stop believing</p>

</body>

</html>

## Insertion

### Html

<script src="jquery-1.10.2.js"></script>

<script>

//Метод html() - получает HTML-содержимое первого элемента в наборе соответствующих элементов или устанавливает HTML-разметку для каждого совпавшего элемента.

$(function () {

$("p").click(function () {

// $(this) - ссылка на объект, в котором произошло событие.

alert($(this).html());

});

});

</script>

<style>

p {

background-color: #bdb1c9;

color: White;

padding: 4px;

font-family: Verdana;

cursor: pointer;

}

</style>

</head>

<body>

<p>

Test string 1

</p>

<p>

<b>Test string 2</b>

</p>

<p>

Test string 3

</p>

</body>

### Html(val)

<title>Метод html(val)</title>

<script src="jquery-1.10.2.js"></script>

<script>

//Метод html() - получает HTML-содержимое первого элемента в наборе соответствующих элементов или устанавливает HTML-разметку для каждого совпавшего элемента.

$(function () {

$("p").html("<u>Hello world</u>");

});

</script>

<style>

p {

background-color: #bdb1c9;

color: White;

padding: 4px;

font-family: Verdana;

cursor: pointer;

}

</style>

</head>

<body>

<p>

Test string 1

</p>

<p>

<b>Test string 2</b>

</p>

<p>

Test string 3

</p>

### Text

<script src="jquery-1.10.2.js"></script>

<script>

//Метод text() - получает содержимое текста каждого элемента в наборе соответствующих элементов, в том числе их потомков, или устанавливает текстовое содержимое элементов набора.

$(function () {

$("p").click(function () {

alert($(this).text());

//alert($(this).html());

});

});

</script>

### Text(val)

<title>text()</title>

<script src="jquery-1.10.2.js"></script>

<script>

//Метод text() - получает содержимое текста каждого элемента в наборе соответствующих элементов, в том числе их потомков, или устанавливает текстовое содержимое элементов набора.

$(function () {

$("p").click(function () {

$(this).text("<a href=\"#\">Link</a>");

//$(this).html("<a href=\"#\">Link</a>");

});

});

</script>

### Append

<title>Методы append() и appendTo()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод append() - добавляет контент внутрь каждого элемента набора. Добавляемый контент следует за уже существующим. Данный метод подобен применению appendChild.

// Метод appendTo() - добавляет все элементы набора к другому указанному набору элементов. Добавляемые элементы следуют после уже существующих.

$(function () {

$("p").append("new text"); // Добавление контента в элемент "p".

$("table").append("<tr><td>3</td><td>4</td></tr>");

$("<b>Hello world</b>").appendTo(".test"); // Добавление контента в элемент ".test".

});

</script>

### Prepend

<title>Методы prepend() и prependTo()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод prepend() - добавляет контент внутрь каждого элемента набора. Добавляемый контент следует перед уже существующим.

// Метод prependTo() - добавляет все элементы набора к другому указанному набору элементов. Добавляемые элементы следуют перед уже существующими.

$(function () {

$("p").prepend("new text"); // Добавление контента в элемент "p".

$("table").prepend("<tr><td>3</td><td>4</td></tr>");

$("<b>Hello world</b>").prependTo(".test"); // Добавление контента в элемент ".test".

});

</script>

### Wrap/Empty/Remove

<title>Методы wrap(), empty(), remove().</title>

<script src="jquery-1.10.2.js"></script>

<script>

// wrap() - вставляет каждый совпавший элемент набора в указанную конструкцию HTML кода. Этот процесс наиболее полезен для встраивания дополнительной структуры в документ без необходимости разрушения его первоначальных семантических свойств.

// empty() - удаляет все содержимое из каждого элемента в наборе совпавших элементов.

// remove() - удаляет все совпавшие элементы из DOM. Эта функция НЕ удаляет элементы из объекта jQuery, позволяя Вам воспользоваться ими позже. Обратите внимание, начиная с версии 1.2.2 эта функция также удаляет все обработчики событий и внутренние закешированные данные.

$(function () {

$("span").wrap("<div>");

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

$("p").empty();

});

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

$("span").remove();

});

});

</script>

<style>

div {

background-color: Blue;

padding: 10px;

}

span {

background-color: White;

}

</style>

</head>

<body>

<span>Paragraph</span>

<p>Hello world</p>

<input type="button" value="Clear" id="clearBtn" />

<input type="button" value="Remove" id="removeBtn" />

</body>

</html>

# Events and Event handlers

## Events

### Ready event

<script>

// ready - определяет функцию, которая будет запущена после полной загрузки DOM

// Первый вариант.

$(document).ready(function () {

alert("Page Loaded I");

});

// Второй вариант.

$(document).ready(after\_ready);

function after\_ready() {

alert("Page Loaded II");

}

// Третий вариант вариант. (Сокращенная запись)

$(function () {

alert("Page Loaded III");

});

</script>

### Click

<title>Событие click</title>

<script src="jquery-1.10.2.js"></script>

<script>

// click() - добавляет обработчик на JavaScript событие "click"

// click(function) - добавляет функцию к событию click для каждого элемента набора.

// click() рассматривается как mousedown и mouseup, следующие друг за другом на одном и том же элементе.

$(function () {

$("p").click(function () { alert("clicked"); });

$("#button").click(function () { $("p").click() });

});

</script>

</head>

<body>

<p>Параграф №1</p>

<p>Параграф №2</p>

<p>Параграф №3</p>

<input type="button" id="button" value="push me" />

### Focus/blur

<title>События focus & blur</title>

<script src="jquery-1.10.2.js"></script>

<script>

$(function() {

// focus - событие происходит, когда элемент получает фокус

$("input").focus(function() {

$(this).val("focus");

});

// blur - событие происходит, когда элемент теряет фокус

$("input").blur(function() {

$(this).val("blur");

});

});

</script>

</head>

<body>

<input type="text" />

<br />

<br />

### KeyDown

<title>Событие keydown</title>

<script src="jquery-1.10.2.js"></script>

<script>

// keydown() - устанавливает обработчик события keydown для каждого элемента набора.

// event - объект, определяющий информацию о событии.

$(window).keydown(function (event) {

switch (event.keyCode) { // event.keyCode - код нажатой клавиши

case 37: {

alert("left pushed");

}

break;

case 38: {

alert("up pushed");

}

break;

case 39: {

alert("right pushed");

}

break;

case 40: {

alert("down pushed");

}

break;

}

});

</script>

### KeyUp

<title>Событие keyup</title>

<script src="jquery-1.10.2.js"></script>

<script>

// keyup() - создает обработчик на событие keyup для каждого элемента набора.

$(window).keyup(function (event) {

switch (event.keyCode) {

case 37: {

alert("left poped");

}

break;

case 38: {

alert("up poped");

}

break;

case 39: {

alert("right poped");

}

break;

case 40: {

alert("down poped");

}

break;

}

});

</script>

### KeyPress

<title>Событие keypress</title>

<script src="jquery-1.10.2.js"></script>

<script>

// keypress - событие, которое возникает при нажатии символьной клавиши клавиатуры (не срабатывает для функциональных клавиш (Shift, Ctrl, Alt и т.д.)

$(window).keypress(function (event) {

alert(String.fromCharCode(event.charCode) + "(" + event.charCode + ")");

});

</script>

### MouseOver/MouseOut

<title>События mouseover & mouseout</title>

<script src="jquery-1.10.2.js"></script>

<script>

// mouseover(f) - функция, которая принимает и назначает функцию "f" как обработчик на событие mouseover для каждого элемента набора.

// событие происходит при входе курсора в область элемента.

// mouseout(f) - функция, которая принимает и назначает функцию "f" как обработчик на событие mouseout для каждого элемента набора.

// событие происходит при выходе курсора в область элемента.

// Создаем счетчики для подсчета количества входов и выходов мыши за пределы элемента.

var overCount = 0;

var outCount = 0;

$(function () {

// Добавляем обработчики на события

$("div:first").mouseover(function () { overCount++; Update(); }) // курсор попадает в область элемента

$("div:last").mouseout(function () { outCount++; Update(); }) // курсор выходит за область элемента

});

function Update() {

$("#over").text("MouseOver " + overCount);

$("#out").text("MouseOut " + outCount);

}

</script>

</head>

<body>

<div style="padding: 20px; background-color: yellow; margin-bottom: 30px;">

<span id="over">MouseOver</span>

</div>

<div style="padding: 20px; background-color: lightsteelblue">

<span id="out">MouseOut</span>

</div>

</body>

### Hover

<title>Функция hover()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// hover(in, out) - функция, которая отвечает за привязку обработчиков на события mouseover и mouseout.

$(function () {

$("div").hover(onIn, onOut);

});

// Функция которая отработает при наведении курсора на элемент

function onIn() {

$("div").html("IN");

}

// Функция которая отработает при выходе курсора за элемент

function onOut() {

$("div").html("OUT");

}

</script>

<style>

div {

background: green;

width: 100px;

height: 100px;

text-align: center;

color: white;

}

</style>

</head>

<body>

<div>

Test div

</div>

### MouseMove

<title>Событие mousemove</title>

<script src="jquery-1.10.2.js"></script>

<script>

// mousemove() - функция, устанавливающая обработчик на событие движения курсора по поверхности элемента.

$(function () {

$("div").mousemove(function (e) {

// парметр "e" - объект с информацией о событии.

// pageX, pageY - координаты относительно документа, учитывая прокрутку.

var pageCoord = "( " + e.pageX + ", " + e.pageY + " )";

$("span").text("Page coords " + pageCoord);

});

});

</script>

<style>

div {

background-color: Yellow;

width: 300px;

height: 300px

}

body {

height: 2000px;

}

</style>

</head>

<body>

<span>Coords</span>

<div></div>

<span>Coords</span>

</body>

## EventHadlers

### AddEventListener

<title>Метод addEventListener()</title>

<script>

window.addEventListener('load', function () { // Установка обработичка на событие load.

function show\_msg() {

alert('Hello world');

}

var div = document.getElementById('test');

div.addEventListener('mouseover', show\_msg, false); // Установка обработичка на событие mouseover для элемента c id test.

}, false);

</script>

### RemoveEventListener

<script>

window.addEventListener('load', function () { // Установка обработичка на событие load.

var div = document.getElementById('test');

var del = document.getElementById('del');

function show\_msg() {

alert('Hello world');

}

div.addEventListener('mouseover', show\_msg, false); // Установка обработичка на событие mouseover для дива.

del.addEventListener('click', function(){

div.removeEventListener('mouseover', show\_msg, false); // Удаление обработичка на событие mouseover для дива.

}, false);

}, false);

</script>

### EventRoutes

// В модели DOM Level 2 события распространяются по следующему принципу:

// На этапе ПЕРЕХВАТА событие распространяется от Document вниз до целевого элемента управления.

// Второй этап - СОБЫТИЕ В ЦЕЛЕВОМ УЗЛЕ.

// На третьем этапе событие ВСПЛЫВАЕТ от целевого элемента обратно вверх до корневого элемента.

// последний параметр в методе addEventListener -

// true - обработать событие на этапе ПЕРЕХВАТА

// false - обработать событие на этапе ВСПЛЫВАНИЕ

var isTunnel = true;

window.addEventListener('load', function () {

var e1 = document.getElementById("first");

var e2 = document.getElementById("second");

var e3 = document.getElementById("third");

e1.addEventListener("click",

function () {

this.style.backgroundColor = "green";

alert("Обработчик DIV 1");

},

isTunnel);

e2.addEventListener("click",

function () {

this.style.backgroundColor = "yellow";

alert("Обработчик DIV 2");

},

isTunnel);

e3.addEventListener("click",

function () {

this.style.backgroundColor = "red";

alert("Обработчик DIV 3");

},

isTunnel);

}, false);

</script>

<style>

div {

border: 3px solid gray;

padding: 50px;

background-color: white;

}

</style>

</head>

<body>

Первый

<div id="first">

Второй

<div id="second">

Третий

<div id="third">

</div>

</div>

</div>

### PreventDefault

<title>Метод preventDefault()</title>

<script>

window.addEventListener("load", function () {

document.forms[0].addEventListener("submit", function (e) {

if (document.getElementById("nameInput").value.length == 0)

{

// Если поле ввода пустое - отменить действие по умолчанию, которое связанно с событием.

e.preventDefault();

alert("Введите имя");

}

}, false);

}, false);

</script>

</head>

<body>

<form action="/test.html" method="post">

Имя <input id="nameInput" type="text" name="name"/>

<input type="submit" value="Отправить" />

</form>

### StopPropagation

<title>Метод stopPropogation()</title>

<script>

var isTunnel = false;

window.onload = function () {

var e1 = document.getElementById("first");

var e2 = document.getElementById("second");

var e3 = document.getElementById("third");

e1.addEventListener("click",

// e - параметр хранит объект с информацией о событии.

function (e) {

this.style.backgroundColor = "green";

alert("Обработчик DIV 1");

// прекращение дальнейшего маршрута события.

e.stopPropagation();

},

isTunnel);

e2.addEventListener("click",

function (e) {

this.style.backgroundColor = "yellow";

alert("Обработчик DIV 2");

e.stopPropagation();

},

isTunnel);

e3.addEventListener("click",

function (e) {

this.style.backgroundColor = "red";

alert("Обработчик DIV 3");

e.stopPropagation();

},

isTunnel);

}

</script>

<style>

div {

border: 3px solid gray;

padding: 50px;

background-color: white;

}

</style>

</head>

<body>

Первый

<div id="first">

Второй

<div id="second">

Третий

<div id="third">

</div>

</div>

</div>

### On()/Target/Делегированная обработка событий

<title>Метод on()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод on(событие, обработчик) используется для привязки обработчиков на события к объекту или объектам.

// Он принимает в качестве аругментов событие и функцию обработчик. Предусмотрено добавление дополнительных параметров. В отличии от addEventListener(), который может работать как на "перехвате", так и на всплытии, метод on() работает на "всплытие".

//

$(function () {

$("ul").on("click", function (event) {

$(event.target).css("background", "red");

});

});

</script>

</head>

<body>

<ul>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

<li>Текст</li>

</ul>

<img src="DelegateOn.gif" />

</body>

### On() with option

<script src="jquery-1.10.2.js"></script>

<script>

// Метод on(событие, обработчик) используется для привязки обработчиков на события к объекту или объектам.

// Также есть возможность передавать дополнительные данные в виде объектов.

$(function () {

function user\_info(event) {

$('#out').html("Login: " + event.data.login + "<br />" + "Password: " + event.data.password);

}

// Создание и передача дополнительных данных в виде объектов

var access = { login: 'admin', password: '12345' };

$('#button').on('click', access, user\_info);

});

</script>

</head>

<body>

<p id="out"></p>

<input type="button" id="button" value="click me" />

</body

### Off()

<script src="jquery-1.10.2.js"></script>

<script>

// Метод off(событие, обработчик) - удаляет с выбранных элементов страницы обработчики событий, установленные с помощью метода .on().

$(function () {

function show() {

alert('Hello world');

}

function remove() {

$("#first").off('click', show);

}

$("#first").on('click',show);

$("#second").on('click', remove);

});

</script>

</head>

<body>

<input type="button" value="on" id="first" />

<input type="button" value="off" id="second" />

</body>

### One()

<title>Метод one()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод one() - устанавливает обработчик события выбранным элементам страницы. Особенностью метода является то, что обработчик будет вызван не более одного раза, на каждом из элементов.

$(function () {

var count = 0;

function add\_text() {

count++;

$('#out').text('Lorem ipsum ' + count);

}

$('#out').one('mouseover', add\_text);

//$('#out').on('mouseover', add\_text);

});

</script>

<style>

div {

width: 100px;

height: 100px;

background: yellow;

}

</style>

</head>

<body>

<div id="out"></div>

</body>

### Trigger

<title>Метод trigger()</title>

<script src="jquery-1.10.2.js"></script>

<script>

// Метод trigger() - позволяет запустить событие другого элемента управления на странице

var button1;

var button2;

var button1Ctr = 0;

var button2Ctr = 0;

$(function () {

button1 = $("#first");

button2 = $("#second");

button1.click(onButton1Click);

button2.click(onButton2Click);

});

function onButton1Click() {

button1Ctr++;

$("p:first").text("Button 1 clicked " + button1Ctr);

}

function onButton2Click() {

button2Ctr++;

$("p:last").text("Button 2 clicked " + button2Ctr);

// При нажатии на вторую кнопку, также происходит нажатие на первую кнопку.

button1.trigger("click");

}

</script>

</head>

<body>

<input type="button" id="first" value="Button 1" />

<input type="button" id="second" value="Button 2" />

<p>

Button 1 clicked

</p>

<p>

Button 2 clicked

</p>

</body>