# Основы JavaScript

## Роль клиентского JavaScript

1. Пользовательский интерфейс:

* Создание визуальных эффектов.
* Выполнение несложных вычислений.
* Проверка пользовательских данных после ввода.
* Манипуляция данными, которые пользователь ввел в формы.
* Поиск по данным, встроенным в загружаемую страницу.
* Сохранение данных в cookies.

1. Динамическое изменение содержимого страницы (DHTML).
2. Прямое взаимодействие с сервером без перезагрузки всей страницы (AJAX).

Для интеграции JavaScript в страницу необходимо использовать тег <script>…</script>. Тег рекомендуется разместить в теге <head>…</head>. Внутри элемента <script> размещается JavaScript, код которого будет выполняться при загрузке страницы.

<!DOCTYPE html>

<html>

<head>

<title>Интеграция JavaScript</title>

<!--Тег script-->

<script type ="text/javascript">

alert("Hello world!");

document.Write("Hello world again!");

</script>

<!--Завершение тега script-->

</head>

<body>

</body>

</html>

Лексическая структура JavaScript:

* Чувствителен к регистру.
* Инструкции должны заканчиваться точкой с запятой и начинаться с новой строки.
* Игнорирует пробелы и табуляции.

JavaScript – динамически типизированный, а не строго типизированный язык программирования. Основным ключевым словом при объявлении переменной здесь является **var**.

NaN – не число.

Number.MAX\_VALUE – максимально допустимое значение в JavaScript.

Number.MIN\_VALUE – минимально допустимое значение в JavaScript.

Number.NaN – не число.

Number.POSITIVE\_INFINITY - + бесконечность.

Number.NEGATIVE\_INFINITY - - бесконечность.

var string1 = "Hello"; эквивалентно var string1 = 'Hello';

var string1 = "10";

var string2 = 2;

var string3 = string1 + string2 – результатом будет отображение с автоприведением string2 в строковый тип, то есть: 10 + 2.

При вычитании, умножении и делении первый аргумент переводится в целочисленный тип и выводится результат операции с целочисленными типами.

</br> - вывод с новой строки.

</hr> - выводит линию.

var a = undefined – переменная не инициализирована.

Любые данные, выводимые в document превращаются в строковый тип.

Тег <pre></pre> - сохраняет форматирование, заложенное разработчиком.

<!DOCTYPE html>

<!--

Метод prompt(1, 2) - принимает два аргумента:

1 - выводится в качестве простой строки в модальном окне.

2 - значение по умолчанию в текстовом поле для ввода.

Метод prompt(1, 2) - выводит сообщение в окне с текстовым полем и двумя кнопками: "ОК" и "ОТМЕНА".

Метод prompt(1, 2) - возвращает введенное значение или null, если пользователь нажал на кнопку "ОТМЕНА".

Как и в alert, prompt - модальное окно

-->

<html>

<head>

<title>Метод prompt()</title>

<meta charset="utf-8" />

<script>

var a = prompt("Введите какое либо значение", "сюда"); // блокирует выполнение скрипта.

alert(a);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

confirm() - действует похоже на alert, за исключением того, что добавляет кнопку "Отмена" в диалоговое окно.

Возвращает true, если нажата кнопка ОК, или false, если нажата кнопка Cancel.

-->

<html>

<head>

<title>Метод confirm()</title>

<meta charset="utf-8" />

<script>

var conf = confirm("Вы уверены?");

document.write(conf);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Пример применения методов</title>

<meta charset="utf-8" />

<script>

var x = prompt("Input x value", "0"); // x будет string типа

var y = prompt("Input y value", "0"); // y будет string типа

var result = x + y;

document.write(result + "<hr/>");

// Конвертация строкового значения в числовое:

// parseFloat() конвертация в вещественное значение.

// parseInt() конвертация в целочисленное значение.

// Данные функции возвращают значение NaN, если конвертация не удалась.

var integerX = parseInt(x); // paseInt находит в переданном значении цифры и выполняет с ними математические операции,

var integerY = parseInt(y); // только если цифры идут вначале.

result = integerX + integerY;

document.write(result);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Арифметические операторы</title>

<meta charset="utf-8" />

<script>

var a = 9;

var b = 5;

var result;

// Сложение

result = a + b; // в данном случае + - арифметическое сложение.

document.write("a + b = " + result + "<br/>");

// Вычитание

result = a - b; // в случае если а или b нечисловые, оператор пытается преобразовать их в числа.

document.write("a - b = " + result + "<br/>");

// Умножение

result = a \* b;

document.write("a \* b = " + result + "<br/>");

// Деление

result = a / b;

document.write("a / b = " + result + "<br/>"); // деление на 0 дает +/- бесконечность. 0/0 = NaN.

// Деление по модулю

result = a % b;

document.write("a % b = " + result + "<br/>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Деление по модулю</title>

<meta charset="utf-8" />

<script>

var a = -8;

var string = "10";

// Унарный минус (-)

var result = -a; // Операция смены знака на противоположный.

document.write("-a = " + result + "<br/>");

// Унарный плюс (+)

// не изменяет знака при применении к числовому типу (ничего не происходит).

result = +a;

document.write("+a = " + result + "<br/>");

// Унарный плюс (+)

// при применении к строковому типу преобразовывает его в числовой.

result = +string + a;

document.write("+string +a = " + result + "<br/>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Инкремент</title>

<meta charset="utf-8" />

<script>

var a = 9;

a++;

document.write(a + "<br/>"); // выведет 10

document.write(a++ + "<br/>"); // выведет 10

document.write(a + "<br/>"); // выведет 11

document.write("<hr/>");

document.write(a + "<br/>"); // выведет 11

++a;

document.write(a + "<br/>"); // выведет 12

document.write(++a + "<br/>"); // выведет 13

document.write(a + "<br/>"); // выведет 13

document.write("<hr/>");

document.write(a + "<br/>"); // выведет 13

a--;

document.write(a + "<br/>"); // выведет 12

document.write(a-- + "<br/>"); // выведет 12

document.write(a + "<br/>"); // выведет 11

document.write("<hr/>");

document.write(a + "<br/>"); // выведет 11

--a;

document.write(a + "<br/>"); // выведет 10

document.write(--a + "<br/>"); // выведет 9

document.write(a + "<br/>"); // выведет 9

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Оператор равенства "==" предназначен для сравнения значений двух переменных. При этом необходимо помнить

о следующих правилах:

1. Если одно значение null, а другое undefined - значения равны.

2. Если одно значение число, а другое строка, то строка преобразовывается в число и сравнивается.

3. Если одно значение true, оно преобразовывается в 1. Если false - в 0.

-->

<html>

<head>

<title>Оператор сравнения</title>

<meta charset="utf-8" />

<script>

var a = 1;

var b = "1";

var c = "some";

var d = "som";

var e = true;

var res;

res = a == b;

document.write("a равно b : " + res + "<br/>");

res = c == d;

document.write("c равно d : " + res + "<br/>");

res = a == e;

document.write("a равно e : " + res + "<br/>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Идентичность - требует совпадения типа и значения переменной, когда для равенства достаточно

только совпадающих значений.

Оператор идентичности "===" предназначен для проверки значений двух переменных на идентичность.

При этом нужно помнить о следующих правилах:

1. Если значения имеют разные типы, они не идентичны.

2. Если значения являются числами, имеют одинаковые значения и не NaN - они идентичны.

3. Если две строки имеют одинаковую длину и содержат одни и те же символы - они идентичны.

4. Два одинаковых логических значения идентичны.

-->

<html>

<head>

<title>Операторы равенства</title>

<meta charset="utf-8" />

<script>

var a = 1;

var b = "1";

var c = "some";

var d = "some";

var e = true;

var f = 1;

var g = true;

var res;

res = a == b;

document.write("a равно b : " + res + "<br/>");

res = a === b;

document.write("a идентично b : " + res + "<br/>");

document.write("<hr/>");

res = a === f;

document.write("a идентично f : " + res + "<br/>");

document.write("<hr/>");

res = c === d;

document.write("c идентично d : " + res + "<br/>");

document.write("<hr/>");

res = f === g;

document.write("f идентично g : " + res + "<br/>");

document.write("<hr/>");

res = a != b;

document.write("a не равно b : " + res + "<br/>");

res = a !== b;

document.write("a не идентично b : " + res + "<br/>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Операторы</title>

<meta charset="utf-8" />

<script>

var a = +prompt("Введите первое число:", "сюда"); // +prompt преобразует а и b в число.

var b = +prompt("Введите второе число:", "сюда");

var res;

// Меньше

res = a < b;

document.write("a < b" + res + "<br/>");

// Больше

res = a > b;

document.write("a > b" + res + "<br/>");

// Меньше или равно

res = a <= b;

document.write("a <= b" + res + "<br/>");

// Больше или равно

res = a >= b;

document.write("a >= b" + res + "<br/>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Сравнение строк производится посимвольно для числовых значений каждого символа из кодировки Unicode.

Символ в верхнем регистре считается меньше символа в нижнем регистре.

Например: A - code (41 Hex)

a - code (61 Hex)

-->

<html>

<head>

<title>Сравнение строк</title>

<meta charset="utf-8" />

<script>

var res = "A" < "a";

document.write("A < a = " + res + "<br/><hr/>");

var c = "hello";

var d = "world";

res = c < d;

document.write("hello < world = " + res + "<br/><hr/>");

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Логические операторы</title>

<meta charset="utf-8" />

<script>

var a = true;

var b = false;

var res = a && b;

document.write("a && b = " + res + "<br/>");

</script>

</head>

<body>

</body>

</html>

## Условные и циклические конструкции

При использовании **if** во всех случаях необходимо ставить не одинарное =, а двойное (==).

<!DOCTYPE html>

<html>

<head>

<title>Coffee Maker</title>

<meta charset="utf-8" />

<script>

var coffeesize = prompt("Размеры стаканчиков: 1 = Маленький, 2 = Средний, 3 = Большой.");

var cost = 0;

switch (coffeesize) {

case "1":

{

cost += 25;

document.write("<img src='coffee.jpg' width='100' height='100'/>");

}

break;

case "2":

{

cost += 50;

document.write("<img src='coffee.jpg' width='150' height='150'/>");

}

break;

case "3":

{

cost += 75;

document.write("<img src='coffee.jpg' width='256' height='256'/>");

}

break;

}

if (cost != 0) {

document.write("<p>Внесите " + cost + " копеек.<p>Спасибо!");

}

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Циклическая конструкция Do While</title>

<meta charset="utf-8" />

<script>

// Ctrl + K, C Ctrl + K,U - убрать/снять комментарий.

var input;

do {

input = prompt("Введите значение.");

document.write("<p>" + input);

}

while (input != null);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Циклическая конструкция for</title>

<meta charset="utf-8" />

<script>

for (var i = 0; i < 3; i++) {

document.write("<p>Counter = " + i);

}

// i доступно за пределами цикла

document.write("<p>" + i);

</script>

</head>

<body>

</body>

</html>

## Массивы

<!DOCTYPE html>

<html>

<head>

<title>Массивы</title>

<meta charset="utf-8" />

<script>

var a = [1, 2, 3, 4, 5];

document.write(a); // Вывод элементов массива, при этом массив преобразуется в строку

var b = ["Jan", "Feb", "Mar", "Apr", "May"];

document.write("<hr/>");

document.write(a);

var c = [, , , ];

document.write("<hr/>");

document.write(c);

var d = ["Hello", 34, true];

document.write("<hr/>");

document.write(d);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

1. Конструктор без аргументов создает пустой массив.

2. Конструктор с одним целочисленным аргументом создаст массив размерностью в количество элементов.

3. Конструктор с двумя и более аргументами - создает массив из передаваемых аргументов, при этом все аргументы

будут расположены в массиве, так как заданы в аргументах.

Такой подход использования массива не рекомендуется!

-->

<html>

<head>

<title>Конструктор Array</title>

<meta charset="utf-8" />

<script>

var a = Array(); // empty array

var b = Array(1, 2, 3, 4); // массив размерностью четыре элемента

document.write(b);

document.write("<hr/>");

var c = Array(10); // массив размерностью в десять элементов

c[0] = 1;

c[1] = 2;

c[2] = 3;

c[3] = 4;

c[4] = 5;

c[5] = 6;

c[6] = 7;

c[7] = 8;

c[8] = 9;

c[9] = 10;

c[15] = "Десять"; // происходит изменение размера массива

document.write(c + " " + c.length);

c.length = 20; // изменяется длина массива

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Ассоциативный массив - абстрактный тип данных, позволяющий хранить пары вида "ключ-значение". От простых массивов

отличается тем, что доступ к элементам происходит по ключу.

-->

<html>

<head>

<title>Ассоциативные массивы</title>

<meta charset="utf-8" />

<script>

// Объект, имеющий три свойства

var laptop = { cpu: "Core i7", ram: "4GB", screen: "15" };

// Запрос на введение имени свойства

var key = prompt("Введите интересующий параметр (cpu, ram, screen)", "");

// Обращение к массиву по ключу

var value = laptop[key];

document.write(value + "<hr/>");

// Обращение к массиву по ключу

// Данные выражения имеют одинаковые значения:

document.write("<p>" + laptop.cpu);

document.write("<p>" + laptop["cpu"]);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Массивы бывают многомерными. Для этого в каждом элементе массива следует расположить еще один массив.

Таким образом, возможно реализовать n-мерный массив.

-->

<html>

<head>

<title>Двухмерная матрица</title>

<meta charset="utf-8" />

<script>

// Создание многомерного массива.

var table = new Array(10);

for (var i = 0; i < table.length; i++) { // в таблице 10 строк

table[i] = new Array(10); // в каждой строке 10 столбцов

}

// Инициализация массива

for (var row = 0; row < table.length; row++) {

for (var col = 0; col < table[row].length; col++) {

if ((row + col) % 2 == 1)

table[row][col] = "<img src = 'images/parquet1.jpg' width = '50' height = '50'/>";

else

table[row][col] = "<img src = 'images/parquet2.jpg' width = '50' height = '50'/>";

}

}

table[8][1] = "<img src = 'images/parquet3.jpg' width = '50' height = '50'/>";

for (var row = 0; row < table.length; row++) {

for (var col = 0; col < table.length; row++) {

document.write(table[row][col]);

}

document.write("<br/>");

}

var square = table[8][1];

document.write("<hr/>" + square);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Метод Join - преобразует все элементы массива в строки и объединяет их.

Если метод принимает строковый аргумент, то аргумент станет разделителем вместо запятых.

-->

<html>

<head>

<title>Метод Join()</title>

<meta charset="utf-8" />

<script>

var arr = ["Sep", "Oct", "Nov", "Dec", "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug"];

document.write(arr);

document.write("<p><b>Значение, которое возвращает метод - join(' | '); </b>");

var res = arr.join(" | ");

document.write("<p> + res");

document.write("<p><b>Значение, которое возвращает метод - join(<br/>); </b>");

res = arr.join("<br/>");

document.write("<p>" + res);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

concat - дополняет элементы массива элементами, которые указаны в параметрах.

-->

<html>

<head>

<title></title>

<meta charset="utf-8" />

<script>

var a = [1, 2, 3];

a = a.concat(4, 5);

document.write("<p>" + a); // 1,2,3,4,5

a = a.concat(6, 7, [8, 9]);

document.write("<p>" + a); // 1,2,3,4,5,6,7,8,9

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

reverse() - меняет порядок следования элементов массива на противоположный.

-->

<html>

<head>

<title>Метод reverse()</title>

<meta charset="utf-8" />

<script>

var a = new Array(1, 2, 3, 4, 5);

a.reverse(); // 5,4,3,2,1

document.write(a.join("<br/>"));

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

slice() - возвращает фрагмент или подмассив указанного массива.

-->

<html>

<head>

<title>Метод slice</title>

<meta charset="utf-8" />

<script>

var a = new Array(1, 2, 3, 4, 5);

// slice - начальный индекс (включительно), конечный индекс (не включительно)

var temp = a.slice(0, 3); // возвращает первые три элемента 1,2,3

document.write("<p>" + temp);

// slice - начальный индекс; возвращает подмассив от указанного индекса до конца массива.

temp = a.slice(3); // 4,5

document.write("<p>" + temp);

// Если аргумент отрицательный, он задает позицию относительно конца массива.

var temp = a.slice(1, -1); // 2,3,4

document.write("<p>" + temp);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Метод splice() - универсальный метод для вставки и удаления элементов из массива.

Метод возвращает массив из удаленных элементов.

arrayObj.splice(start, deleteCount, [elem1[, elem2[, ...[, elemN]]]])

start - индекс в массиве, с которого нужно начать удаление.

deleteCount - количество элементов для удаления начиная с индекса start.

elem1, elem2, ..., elemN - добавляемые в массив элементы. Добавление начинается с позиции start.

-->

<html>

<head>

<title>Метод splice()</title>

<meta charset="utf-8" />

<script>

var a = [1, 2, 3, 4, 5, 6, 7];

document.write("<p>Исходный массив: " + a);

// Удаление.

//splice (начальный индекс, сколько элементов удалить)

var d = a.splice(1, 3); // будут удалены: 2,3,4

document.write("<p>Удалены: " + d);

document.write("<p>Остались: " + a);

// Вставка.

//splice (начальный индекс = 1, сколько элементов удалить = 0, значения для вставки с начального индекса)

a.splice(1, 0, 2, 3, 4);

document.write("<p>После вставки: " + a);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

sort() - Сортирует элементы массива в алфавитном порядке

(при необходимости временно преобразуя элементы в строки для выполнения сравнения).

-->

<html>

<head>

<title>Метод сортировки массива sort()</title>

<meta charset="utf-8" />

<script>

var a = [1, 4, 11, 2, 9, 99, 5];

a.sort();

document.write(a.join("<p>"));

document.write("<hr/>");

var s = ["Jan", "Feb", "Mar", "Apr", "May"];

s.sort();

document.write(s.join("<p>"));

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

push() - добавление одного или нескольких элементов в конец массива.

(возвращаемое значение - количество элементов в новом массиве).

-->

<html>

<head>

<title>Метод push()</title>

<meta charset="utf-8" />

<script>

var stack = ["Jan", "Feb", "Mar"];

document.write("<p>Исходный: " + stack.join(" "));

document.write("<p>Размер массива: " + stack.length);

document.write("<hr/>");

var lenght = stack.push("Apr"); // Метод push() всегда возвращает размер нового массива!

document.write("<p>После добавления: " + stack.join(" "));

document.write("<p>Размер массива: " + lenght);

document.write("<hr/>");

lenght = stack.push("May", "Jun");

document.write("<p>После добавления: " + stack.join(" "));

document.write("<p>Размер массива: " + lenght);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

pop() - удаление последнего элемента массива.

(возвращаемое значение - удаленный элемент массива).

-->

<html>

<head>

<title>Метод pop()</title>

<meta charset="utf-8" />

<script>

var stack = ["Jan", "Feb", "Mar", "Apr", "May", "Jun"];

document.write("<p>Исходный: " + stack.join(" "));

document.write("<p>Размер массива: " + stack.length);

document.write("<hr/>");

var lenght = stack.pop();

document.write("<p>После pop: " + stack.join(" "));

document.write("<p>Удаленный элемент: " + lenght);

document.write("<p>Размер массива после удаления: " + stack.length);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

shift() - удаляет и возвращает первый элемент массива.

-->

<html>

<head>

<title>Метод shift()</title>

<meta charset="utf-8" />

<script>

var arr = [1, 2, 3];

document.write("<p>Исходный массив: " + arr.join(" "));

document.write("<hr/>");

var value = arr.shift();

document.write("<p>Массив после сдвига: " + arr.join(" "));

document.write("<p>Выдвинутое значение: " + value);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

unshift() - добавляет элемент в начало массива.

-->

<html>

<head>

<title>Метод unshift()</title>

<meta charset="utf-8" />

<script>

var arr = [1, 2, 3];

document.write("<p>Исходный массив: " + arr.join(" "));

document.write("<hr/>");

arr.ushift(0);

document.write("<p>Массив после сдвига: " + arr.join(" "));

document.write("<hr/>");

arr.unshift(-3, -2, -1);

document.write("<p>Массив после сдвига: " + arr.join(" "));

</script>

</head>

<body>

</body>

</html>

## Функции

<!DOCTYPE html>

<!--

Вызов функции происходит следующим образом:

имя\_функции(аргументы);

-->

<html>

<head>

<title>Функции</title>

<meta charset="utf-8" />

<script>

function MyFunction() {

document.write("<p>Hello world!");

}

MyFunction(); // вызов функции

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Аргументы функции</title>

<meta charset="utf-8" />

<script>

var a = 10;

var b = 20;

function Sum(x1, x2)

{

var sum = x1 + x2;

document.write("<p>sum: " + sum);

}

// Вызов функции

Sum(); // результат будет sum: NaN

Sum(a, b);

Sum(2, 10);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

В языке JS при вызове функции количество параметров не обязательно должно совпадать с количеством

параметров при объявлении. Параметры, которым при вызове не передали значения, будут равны

undefined.

-->

<html>

<head>

<title>Необязательные аргументы</title>

<meta charset="utf-8" />

<script>

function Print(msg, count /\* необязательный параметр count \*/)

{

if(count == undefined)

{

count = 3;

}

for (var i = 0; i < count; i++) {

document.write("<p>" + msg);

}

document.write("<hr/>");

}

// Вызов функции

Print("hello");

Print("world", 7);

</script>

</head>

<body>

</body>

</html>

Если в функции не указать **return**, она будет возвращать undefined.

<!DOCTYPE html>

<!--

Функции-литералы.

-->

<html>

<head>

<title>Функции-литералы</title>

<meta charset="utf-8" />

<script>

var f = function () {

document.write("<p>Hello world!");

};

f(); // Вызов функции.

f = function () {

document.write("<p>Bye world!");

};

f();

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Функции, передаваемые в качестве аргументов другой функции.

(CallBack Function - функции обратного вызова).

-->

<html>

<head>

<title>Функции обратного вызова</title>

<meta charset="utf-8" />

<script>

var add = function (a, b) {

return a + b;

}

var sub = function (a, b) {

return a - b;

}

function show(callBackFunction, a, b) {

var result = callBackFunction(a, b);

document.write("<p>" + result);

};

show(add, 10, 20);

show(sub, 50, 30);

</script>

</head>

<body>

</body>

</html>

## Объекты

<!DOCTYPE html>

<html>

<head>

<title>Объекты</title>

<meta charset="utf-8" />

<script>

// Создание объекта через блок инициализации

var instance = { propertyA: 222, propertyB: 444 };

// Вывод значений свойств

document.write(instance.propertyA);

document.write(instance.propertyB);

// Присвоение свойствам новых значений

instance.propertyA = 333; // 1 способ

instance["propertyB"] = 555; // 2 способ

document.write(instance.propertyA);

document.write(instance.propertyB);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Конструктор Object</title>

<meta charset="utf-8" />

<script>

var point = new Object();

// Если свойство отсутствует, оно создается автоматически и ему присваивается значение.

point.x = 10;

point.y = 15;

// Добавляем к объекту метод

point.f = function () {

document.write("<p>x = " + point.x + "<p>y = " + point.y);

};

point.f();

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Свойства в JS</title>

<meta charset="utf-8" />

<script>

var person = {

firstName: "Jim",

lastName: "Bim",

age: 30,

occupation: "Web Developer"

};

document.write("<p>" + person.firstName + " " + person.lastName + " " + person.occupation);

person.age = 25;

document.write("<p>" + person.age);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

В объекты можно вкладывать другие объекты. Для этого необходимо в качестве содержимого свойства поместить описание

еще одного объекта. При обращении к свойствам вложенных объектов необходимо полностью указывать путь, например:

human.occupation.company - обращаемся к свойству company объекта occupation, который вложен в объект human

-->

<html>

<head>

<title>Вложенные объекты</title>

<meta charset="utf-8" />

<script>

var human = {

firstname: "Alex",

lastname: "Morozov",

age: 27,

// Вложенный объект

occupation: {

company: "Microsoft",

position: "Developer"

}

};

document.write("<p>" + human.firstname);

document.write("<p>" + human.age);

document.write("<p>" + human.occupation.company);

document.write("<p>" + human.occupation.position);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Вложенный объект можно создать с помощью функции конструктора

-->

<html>

<head>

<title>Создание вложенных объектов через конструктор</title>

<meta charset="utf-8" />

<script>

var human = new Object();

human.firstname = "Alex";

human.lastname = "Bird";

human.age = 27;

// Создание вложенного объекта

human.occupation = new Object();

human.occupation.company = "Microsoft";

human.occupation.position = "Developer";

document.write("<p>" + human.firstname);

document.write("<p>" + human.age);

document.write("<p>" + human.occupation.company);

document.write("<p>" + human.occupation.position);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Метод - это функция, принадлежащая конкретному объекту, которую можно вызвать только на

этом объекте. При этом такая функция описывается как функция литерал присвоенная свойству.

method: function(){...}

-->

<html>

<head>

<title>Методы</title>

<meta charset="utf-8" />

<script>

var human = {

firstname: "Alex",

lastname: "Morozov",

age: 27,

// Метод show() в блоке инициализации объекта

show: function(){

document.write("<p>" + this.firstname);

document.write("<p>" + this.lastname);

document.write("<p>" + this.age);

},

// Создаем метод saygoodbye() в блоке инициализации объекта

chagework: function(age){

this.age = age;

}

}

// Вызов методов

human.show();

human.chagework(30);

human.show();

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Builder</title>

<meta charset="utf-8" />

<script>

var builder = new Object();

builder.createroof = function () {

document.write("<p>Build roof");

}

builder.createfloor = function () {

document.write("<p>Build floor");

}

builder.createbasement = function () {

document.write("<p>Build basement");

}

builder.createroof();

builder.createfloor();

builder.createbasement();

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Циклическая конструкция for in используется для перебора

элементов массива или перебора свойств объекта.

-->

<html>

<head>

<title>For in</title>

<meta charset="utf-8" />

<script>

var house = {

roof: "ROOF",

secondfloor: "SECOND FLOOR",

firstfloor: "FIRST FLOOR",

basement: "BASEMENT"

}

for (element in house) {

document.write(house[element]);

}

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

Конструкция if (имя\_свойства in объект) используется для проверки

наличия свойства у объекта

-->

<html>

<head>

<title>Проверка существования свойства</title>

<meta charset="utf-8" />

<script>

var house = {

roof: "<p>Roof",

// thirdfloor: отсутствует

secfloor: "<p>Second Floor",

firstfloor: "<p>First Floor",

basement: "<p>Basement",

address: "<p>Address"

}

// Строительство дома

document.write(house.roof);

if ("thirdfloor" in house) {

document.write(house["thirdfloor"]);

}

else {

document.write(house["secfloor"]);

}

document.write(house.secfloor);

document.write(house.firstfloor);

document.write(house.basement);

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<!--

delete - оператор, способный удалить свойство в объекте. Также удаляет элементы массивов.

Делается это следующим образом: delete имя\_объекта.имя\_свойства;

При обращении к свойству впоследствии будет возвращаться undefined

-->

<html>

<head>

<title>Оператор delete</title>

<meta charset="utf-8" />

<script>

var house = {

roof: "<p>Roof",

// thirdfloor: отсутствует

secfloor: "<p>Second Floor",

firstfloor: "<p>First Floor",

basement: "<p>Basement",

address: "<p>Address"

}

// Строительство дома

if ("secfloor" in house) {

delete house.secfloor;

}

for (element in house) {

document.write(house[element]);

}

var array = new Array(0, 1, 2, 3, 4);

var index = prompt("Выберите индекс элемента для удаления: " + array.join(", "));

delete array[index];

</script>

</head>

<body>

</body>

</html>

# JQUERY

Функция jQuery() – основная функция в библиотеке, которая позволяет производить поиск элементов на веб-странице, создавать новые элементы, манипулировать ими и т.д. Возвращает объект jQuery.

Для упрощения написания кода используют сокращенную запись функции jQuery() – функцию $().

В качестве аргументов она принимает селекторы (элементы), над которыми следует выполнить какие-то действия.

$('p'); // выбор всех параграфов на веб-странице.

Прежде чем начинать выполнять JS-код необходимо быть уверенным в загрузке всех элементов страницы. В классическом JS коде чаще всего отслеживают полную загрузку страницы событием onload:

window.onload() = function(){

// тело функции

};

В JQ используют событие ready, которое происходит в момент готовности DOM-дерева:

$(document).ready(function(){

// тело функции

});

<!DOCTYPE html>

<html>

<head>

<title>Знакомство с JQ. Подключение удаленной библиотеки</title>

<meta charset="utf-8" />

<!-- 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

$(); // Псевдоним функции с одинаковым набором функционала

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Метод ready()</title>

<meta charset="utf-8" />

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

<script>

$(document).ready(function () {

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

$("p").text("Hello world jQuery");

});

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Селектор</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

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

// $("p").css("color", "red"); // сделает красным цветом то, что находится в теге <p>

// $("p + div").css("color", "red"); // сделает красным цветом все div, которые находятся

// непосредственно за тегом <p>

});

</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>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Выбор элементов-прямых потомков</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

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

});

</script>

</head>

<body>

<p>Paragraph in body</p>

<div>

<p>Paragraph in div with<span>SPAN</span></p>

<span>First single SPAN</span>

<ul>

<li><span>Item1 in SPAN</span></li>

<li>Item2</li>

<li>Item3</li>

<li>Item4</li>

</ul>

<span>Second single SPAN</span>

</div>

<div>

<span>Third single SPAN</span>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Has</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

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

});

</script>

</head>

<body>

<p>Paragraph in body</p>

<div>

<p>Paragraph in div1</p>

<ul>

<li>Item1</li>

<li>Item2</li>

<li>Item3</li>

</ul>

</div>

<div>

<table border="1">

<tr>

<td>1</td>

<td>2</td>

<td>3</td>

</tr>

</table>

<p>Paragraph in div2</p>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Выборка "братьев"</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

// Выбор всех элементов "р", которым предшествует "span" на том

// же уровне вложенности.

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

});

</script>

</head>

<body>

<p>Paragraph in body</p>

<div>

<p>Paragraph in div1</p>

<ul>

<li>Item1</li>

<li>Item2</li>

<li>Item3</li>

</ul>

</div>

<div>

<table border="1">

<tr>

<td>1</td>

<td>2</td>

<td>3</td>

</tr>

</table>

<p>Paragraph in div2</p>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Выборка по атрибутам</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

// Выборка всех элементов "а" с атрибутом "target", имеющим любое значение.

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

});

</script>

</head>

<body>

<a href="vk.com">Vkontakte</a><br/>

<a href="vk.com/friends" target="\_blank">Friends</a>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Выборка по атрибутам</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

// Выборка всех элементов "а" с атрибутом "target", имеющим значение text.

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

});

</script>

<style>

span{

display: inline-block;

width: 80px;

}

</style>

</head>

<body>

<span>Login:</span><input type="text"/><br/>

<span>Password:</span><input type="password"/>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Выборка по атрибутам</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(document).ready(function () {

// Выбор всех элементов "a" с атрибутом "href", начинающимся с "http://"

$("a[href^='http://']").css("color", "red");

// Выбор того же, где атрибут заканчивается на ".com"

$("a[href$='.com']").css("color", "blue");

// Выбор того же, где атрибут содержит "html"

$("a[href\*='html']").css("color", "blue");

});

</script>

<style>

span{

display: inline-block;

width: 80px;

}

</style>

</head>

<body>

<a href="http://edu.itvdn.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>

<!DOCTYPE html>

<html>

<head>

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

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

//eq() - метод для выбора нужного элемента 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('fontWight', '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>

<!DOCTYPE html>

<html>

<head>

<title>Добавление классов к элементам с помощью JQ</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

// on() - добавление обработчика событий текущему выбранному элементу.

// Обеспечивает все функциональные возможности для присоединения обработчиков

// событий.

// adClass() - 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 firstButtonHandler() {

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">

Следите за актуальной экономической информацией с календарем событий от FXStreet. Релевантные новости по всем показателям с детальными пояснениями, данные о предстоящих публикациях по экономике США, Великобритании, Японии и других развитых стран, оценки ожидаемой волатильности и многое другое — экономический календарь станет незаменимым помощником для каждого трейдера. Вы можете настроить календарь событий и новостей экономики FXStreet под свои предпочтения, установив необходимые параметры.

Источник: «Экономический календарь от FXStreet»

</div>

</body>

</html>

## Фильтры в jQuery

$("table tr:even") – выбор четных элементов на странице

$("table tr:odd") – выбор нечетных элементов

$("table tr:eq(4)") – выбор элемента из массива по индексу

$("table tr:first") – выбор первого совпадения на странице

$("table tr:last") – выбор последнего совпадения на странице

<!DOCTYPE html>

<html>

<head>

<title>Фильтр only-child</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

// Выбор единственного дочернего элемента

// nth-child(n) - выбор n-ного дочернего элемента

// nth-child(even) - выбор четных дочерних элементов

// nth-child(odd) - выбор нечетных дочерних элементов

// nth-child(an+b) - в круглых скобках выражение, где a и b целые числа, а n - счетчик,

// автоматически принимающий значения 0, 1, 2...

// :eq(индекс) - выбор по индексу

// :gt(индекс) - (greater than) выбор всех элементов, индекс которых больше указанного

// :lt(индекс) - (less than) выбор всех элементов, индекс которых меньше указанного

// :not(n) - выбор всех элементов за исключением n

// :header - получение всех заголовков (вместо $("h1, h2, h3, h4, h5..."))

$(function () {

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

});

</script>

<style>

div{

border: 1px solid;

margin-bottom: 10px;

padding: 10px;

}

</style>

</head>

<body>

<div>

<p>Hello world!</p>

</div>

<div>

<p>Hello world with a new life!</p>

<p>Hellow people!</p>

</div>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Выбор элементов по индексу</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

var i = 0;

$(function () {

// установим интервал в одну секунду

var timer = setInterval(choose, 1500);

});

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>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Фильтр: animated</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<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>

</html>

Фильтры контента

$("p:contains('JavaScript')") – выбор всех элементов, содержащих текст JavaScript

$("div:empty") – выбор всех элементов, в которых отсутствует содержимое

$("div:has(p)") – выбор всех элементов, которые содержат указанный в параметрах элемент

<!DOCTYPE html>

<html>

<head>

<title>:contains()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(function () {

$('.search\_button').on('click', search);

function search() {

var word = $('.search\_item').val();

$("p:contains(" + word + ")").css("background-color", "yellow");

}

});

</script>

</head>

<body>

<input class="search\_item" type="text" value=""/>

<input class="search\_button" type="button" value="Найти" />

<h1>Header1</h1>

<p>Hello moto</p>

<p>Hello world</p>

<p>Hello people</p>

<p>Hello planet</p>

</body>

</html>

Дополнительные фильтры атрибутов:

$("img[alt~='new']") – выбор всех img с атрибутом alt, которые содержат значение new разделенное пробелом

$("a[target!='\_blank']") – выбор всех a с атрибутом target, который содержит значение не равное \_blank

Фильтры форм:  
$(":button") – выбор всех кнопок на странице

$(":checkbox") – выбор всех чекбоксов

$("input:checked") – выбор только отмеченных чекбоксов или переключателей

$("input:disabled") – выбор элементов форм, которые отключены

$("input:enabled") – выбор включенных элементов форм

$(":file") – выбор всех элементов типа file

<!DOCTYPE html>

<html>

<head>

<title>Фильтр :checkbox</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(function () {

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

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

});

</script>

</head>

<body>

<form>

<p><strong>Выбор пиццы</strong></p>

Большая

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

Средняя

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

Маленькая

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

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

</form>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Фильтр input:checkbox</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(function () {

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

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

count();

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

});

</script>

</head>

<body>

<form>

<p><strong>Выбор пиццы</strong></p>

Большая

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

Средняя

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

Маленькая

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

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

</form>

</body>

</html>

## Манипуляция свойствами, атрибутами, стилями

Функция **attr()** возвращает или изменяет значение атрибутов у выбранных элементов страницы.

Есть несколько вариантов использования данной функции:

* Чтение значения атрибута .attr(attrName). Возвращает значение атрибута attrName у выбранного элемента. Если выбрано несколько элементов, то значение будет взято у первого.
* Изменение значения атрибута .attr(attrName, value).
* Изменение или установка нескольких атрибутов .attr({attrName1:value1, attrName2:value2}). Также attr(attrName, function(index, value)).

Функция **removeAttr("attrName")** удаляет атрибут attrName у выбранных элементов на странице.

Если такого атрибута в списке нет, то ничего не произойдет.

Функции для манипуляции с классами:

* $("table tr:even").addClass("Class1"); - функция добавляет класс выбранным элементам страницы (не заменяя их).
* $("table tr:odd").removeClass("Class1"); - удаляет заданные классы у элементов на странице.
* If($("p:first").hasClass("Class1")) {…} – функция, которая проверяет наличие класса у элемента и возвращает true, если класс есть, и false, если его нет.
* $("table tr:first").toggleClass("class1"); - функция добавляет или удаляет заданный класс по принципу переключателя (если элемент есть, то он удалится, если нет - добавится). .tooggleClass("class1 class2 class3 …" );

<!DOCTYPE html>

<html>

<head>

<title>Метод prop()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

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

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

// .prop({propName1: value1, propName2: value2, ...})

// .prop(propName, function(index, value)) - свойству propName будет присвоено значение, возвращенное

// пользовательской функцией (если она ничего не вернет, то свойство просто не изменит значения).

// 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); // переменная input будет хранить наш текущий "input"

$("p").html(

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

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

}).change(); // вызываем событие change() принудительно

});

</script>

<style>

p{ margin: 20px 0 0; }

b{color: cadetblue; }

</style>

</head>

<body>

Сохранить пароль?

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

<p></p>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Метод removeprop()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

// Метод append() - добавляет контент внутрь каждого элемента набора

// Добавляемый контент следует за уже существующим

$(function () {

var output = $("p");

output.prop("secretCode", 1234);

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

output.removeProp("secretCode");

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

});

</script>

</head>

<body>

</body>

</html>

## Работа с CSS JQuery

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

$("div").css("width");

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

$("p").css({ backgroundcolor: "Red", fontSize: "18pt"}); - включение в параметры json-объекта

.css(styleName, function(index, value))

<!DOCTYPE html>

<html>

<head>

<title>css()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(function () {

$("p").css({

color: "green",

backgroundColor: "lightgray"

});

});

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Height()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

// height() - высота контента

// innerHeight() - высота контента + высота паддинга сверху и снизу

// outerHeight() - высота контента + высота паддинга сверху и снизу + высота рамки сверху и снизу

// outerHeight(true) - ко всему прочему добавляется значение margin сверху и снизу

$(function () {

var a = "";

a += "height() - " + $("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); самостоятельный выбор высоты

// width() аналогичен height(), только соответствует ширине

});

</script>

</head>

<body>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title>Offset()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

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

// offset можно изменять координаты элемента.

// offset(value) - изменяет координаты всех выбранных элементов, делая их равными value. Значение value

// должно быть объектом с двумя свойствами {top:newTop, left:newLeft}.

// offset(function(index, value)) - устанавливает новое значение координат, которое возвращает

// пользовательская функция. index - позиция в наборе, value - текущие координаты элемента.

// При помощи offset() можно получить позицию только первого элемента из выбранных

$(function () {

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

var output = $("div");

output.html("Мои координаты относительно документа:<br/>" + "x: " + element.left + "<br/>y: " +

element.top);

});

</script>

</head>

<body>

</body>

</html>

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

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

var output = $("#output");

output.html(element.left + element.top);

offsetParent() – возвращает ближайшего предка, у которого задан тип позиционирования.

$(function () {

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

});

## Работа с DOM в jQuery

.eq(index) – сокращает число совпавших элементов до одного. Аргументом является позиция элемента в наборе совпавших элементов.

.not(selector) – удаляет элементы, соответствующие указанному выражению из набора совпавших элементов.

.filter(function) – удаляет все элементы, которые не удовлетворяют требованиям функции из набора совпавших элементов.

.is(selector) – проверяет текущий набор элементов на соответствие указанному выражению и возвращает true, если хотя бы один элемент соответствует выражению.

.slice(index, index) – выделяет подмножество из набора совпавших элементов. Например, с 1 по 5 элемент.

.add(выражение) – добавляет в уже существующий набор дополнительные элементы, которые удовлетворяют указанному выражению.

.find(выражение) – отыскивает дочерние элементы, которые удовлетворяют указанному выражению.

.contents() – поиск всех дочерних узлов в наборе совпавших элементов (включая текстовые) или в содержимом документа, если он является фреймом.

.children() – получает набор элементов, содержащий всех непосредственных уникальных потомков для каждого совпавшего элемента.

.sibling() – получает набор элементов, содержащий все уникальные родственные элементы для набора совпавших элементов.

.andSelf() – добавление предыдущего набора к текущему набору.

.end() – отменяет последнее деструктивное действие.

<!DOCTYPE html>

<html>

<head>

<title>Contents()</title>

<script src="https://code.jquery.com/jquery-3.2.1.js"

integrity="sha256-DZAnKJ/6XZ9si04Hgrsxu/8s717jcIzLy3oi35EouyE="

crossorigin="anonymous"></script>

<meta charset="utf-8" />

<script>

$(function(){

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

return this === 3;

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

});

</script>

</head>

<body>

</body>

</html>

.parent(выражение) – получает прямого родителя элемента.

.parents(выражение) – получает набор элементов, содержащих уникальных родителей для совпавших элементов.

.closest() – получает набор, содержащий ближайшие родительские элементы, которые удовлетворяют указанному селектору, включая начальный элемент.

.offsetParent() – возвращает коллекцию jQuery с родителем по позиционированию первого совпавшего элемента.

.next(),

.prev() – получают набор элементов, содержащих уникальные последующие(предыдущие) родственные элементы для всех элементов существующего набора.

.nextAll(),

.prevAll() - поиск всех родственных элементов после текущего элемента (перед текущим элементом).

.nextUntil(селектор),

.prevUntil(селектор) – поиск всех родственных элементов после текущего элемента (перед текущим элементом) до указанного селектора.

**.hover(over, out)** – метод, который имитирует эффект hovering (отслеживание попадания указателя мыши в пределы объекта). Over – функция при наведении, out – при отведении курсора мыши.