Основы JavaScript

- Объявление переменных, объектов, функций и массивов на JavaScript
- Создание массивов JavaScript для хранения данных
- Задание объектов JavaScript как хранилищ пар “ключ-значение”
- Доступ к свойствам объекта
- Практика: Написание кода JavaScript для выполнения тестов в Jasmine

Версии JavaScript

ES6 = ECMAScript 2015

ES7 = ECMAScript 2016

ES8 = ECMAScript 2017

```
<script src="https://cdn.polyfill.io/v2/polyfill.min.js">
</script>
```

Объявление переменной

- Явно

```
var i=10;
var n, sum;
var message='Привет';
var i=0, j=0, k;          // undefined
```

- Неявно (глобальные)

```
j = "десять";
```

- Допускаются повторные объявления

```
var j=0;
var j=1;
```

Числовые литералы

- Десятичные целые

```
0
3
10000000
```

- Шестнадцатеричные целые

```
0xff  // 15*16 + 15 = 255
0xCAFE911
```

- Восьмеричные целые

```
0377  // 3*64 + 7*8 + 7 = 255
```

- С плавающей точкой

```
3.14
2345.789
.3333333333333333
6.02e23          // 6.02  x  10^23
1.4738223E-32    // 1.4738223  x  10^-32
```

Числовая арифметика

до 15 цифр

```
var y = 9999999999999999;  // 10000000000000000
```

до 17 десятичных знаков

```
var x = 0.2 + 0.1;          // 0.30000000000000004
```

```
var x = (0.2 * 10 + 0.1 * 10) / 10;        // 0.3
```

Специальные значения чисел

Infinity	бесконечность
NaN	не число
Number.MAX_VALUE	максимальное число
Number.MIN_VALUE	минимальное число (ближайшее к нулю)
Number.NaN	не число
Number.POSITIVE_INFINITY	+ бесконечность
Number.NEGATIVE_INFINITY	- бесконечность

Работа с числами

```
синус_x = Math.sin(x);

hypot = Math.sqrt(x*x + y*y);

var x = 33;
var y = x.toString(2);  // y is "100001"

var y = (257).toString(0x10);  // y is "101"

var q = 130/0; // Infinity
var s = 0/a;   // NaN
var t = 0/0;   // NaN
```

Символьные строки

Нет типа char

```
""  // пустая строка (0 символов)
'testing'
"3.14"
'name="myform"'
"Wouldn't you like breaks?"
"Это двустрочная\nстрока"
"П - отношение длины окружности к её диаметру"
```

При включении в HTML рекомендуется так:

```
<a href="" onclick="alert('Thank you')">
   Click Me
</a>
```

Символьные строки - экранирование

```
'You\'re right, it can\'t be a quote'
```

```
\0 -  NUL (\u0000)
\b -  Backspace (\u0008)
\t -  Horizontal tab (\u0009)
\n -  Newline (\u000A)
\v -  Vertical tab (\u000B)
\f -  Form feed (\u000C)
\r -  Carriage return (\u000D)
\" -  Double quote (\u0022)
\' -  Apostrophe or single quote (\u0027)
\\ -  Backslash (\u005C)
\xXX -  Символ Latin-1 с 16-ричным кодом XX
\uXXXX  -  Символ Unicode с 16-ричным кодом XXXX
```

Работа с символьными строками

Конкатенация

```
msg = "Hello, " + "world";   // "Hello, world"
greeting =
   "Welcome to my home page," +
   " " +
   name;
```

Методы

```
last_char = s.charAt(s.length - 1); // с нуля
sub = s.substring(1,4);
i = s.indexOf('a');
lastSymbol = s[s.length-1];  //ES5
```

строки – не объекты

Логический тип

true, false

```
let isPositive = b > 0;
if (isPositive)
  b = b - 1;
else
  b = b + 1;
```

Функции

Предопределённые

```
Math.sin()
```

Определённые в программе

```
function square(x){
  return x*x;
}
```

Функция - литерал

```
var square = function(x){ return x*x; }
```

Arrow functions (ES6)

```
var square = (x) => { return x*x; }
var square = x => x*x;
var hello = () => { document.write('Hello'); }
```

Объекты

Набор именованных значений (свойства, properties)

```
image.width
```

Значения свойств могут быть любого типа

```
image.width + 10                                    //число
document.myform.target                              //объект
order.items[2]                                      //массив
document.write()                                    //функция
```

Объект = ассоциативный массив

```
image.width === image['width']
image['1st layer']

var dim='width';
document.write(image[dim]);
```

Создание объектов

Функция-конструктор

```
var o = new Object();
var now = new Date();
var pattern = new RegExp("\\sjava\\s", "i");
```

Литерал

```
var image = {width: 100, height: 150};
var rectangle = {
   upperLeft: { x: 2, y: 2 },
   lowerRight: { x: 4, y: 4}
};
var square = {
   upperLeft:{ x:point.x, y:point.y },
   lowerRight:{ x:(point.x+side), y:(point.y+side) }
};
```

JSON = JavaScript Object Notation

Массивы

Набор пронумерованных значений. Индекс (ключ).

```
document.images[1].width        //объект
cells[1][2]                     //массив
```

Конструктор

```
var a = new Array( );
a[0] = 1.2;
a[1] = "JavaScript";
a[2] = true;
a[3] = { x:1, y:3 };

var a = new Array(1.2, "JavaScript", true, { x:1, y:3 });

var a = new Array(10);
```

Литерал (JSON)

```
var a = [1.2, "JavaScript", true, { x:1, y:3 }];
var sparseArray = [1,,,,5];
```

null и undefined

null = специальное значение "отсутствие значения"

```
var n = null;
```

undefined = неинициализированная переменная или свойство

```
var n;
myObject.prop2016
```

```
null == undefined  // true
null === undefined //false
```

Полезные классы

Date

```
var xmas = new Date(2017, 1, 7);
xmas.setFullYear (xmas.getFullYear+1);
var weekday = xmas.getDay( );
document.write("Today is: " + now.toLocaleString( ));
```

RegExp

```
var pattern = /s$/g;
var pattern = new RegExp("s$","g");
"JavaScript".search(/script/i);
```

Классы-обёртки

- String

```
var s = "hello world";                  // примитив
var S = new String("Hello World");      // объект String
S.valueOf;                              // примитив
```

- Boolean
- Number
- Symbol (ES6)

Преобразование типов

- Неявное преобразование

```
document.write(120);
```

	String	**Number**	**Boolean**	**Object**
undefined	"undefined"	NaN	false	Error
null	"null"	0	false	Error
непустая строка	-	Число или NaN	true	String
""	-	0	false	String
0	"0"	-	false	Number
NaN	"NaN"	-	false	Number
∞	"Infinity"	-	true	Number
-∞	"-Infinity"	-	true	Number
остальные числа	строка	-	true	Number
true	"true"	1	-	Boolean
false	"false"	0	-	Boolean
Object	toString()	valueOf() or toString() or NaN	true	-

Неявное преобразование

Массивы

```
> Number([]), Number([15]), Number([true]), Number([1,2,3])

0, 15, NaN, NaN
```

Объекты – только в число или строку

1. Если valueOf() даёт примитивный тип, преобразовать.
2. Если toString() даёт примитивный тип, преобразовать.
3. Ошибка (TypeError).

Алгоритм применяется для + < <= > >=

Явное преобразование типов

- parseInt, parseFloat

```
parseInt("3 blind mice");     // 3
parseFloat("3.14 meters");    // 3.14
parseInt("12.34");            // 12
parseInt("0xFF");             // 255
parseInt("eleven");           // NaN
parseFloat("$72.47");         // NaN
```

- toString, toFixed, toExponential, toPrecision

```
var n = 78;
binary_string = n.toString(2);         // "10001"
octal_string = "0" + n.toString(8);    // "021"
hex_string = "0x" + n.toString(16);    // "0x11"

var n = 123456.789;
n.toFixed(2);           // "123456.79"
n.toExponential(3);     // "1.235e+5"
n.toPrecision(4);       // "1.235e+5"
n.toPrecision(7);       // "123456.8"
```

Явное преобразование типов

- Конструкторы классов-обёрток Number, Boolean, String, Object

```
var x_as_string  = String(x);
var x_as_number  = Number(x);
var x_as_boolean = Boolean(x);
```

- синтаксический сахар – неявное преобразование

```
var x_as_string  = x + "";
var x_as_number  = +x;
var x_as_boolean = !!x;
```

Объект

Конструктор

```
let o = new Object( );
let now = new Date( );
var new_year_evening = new Date(2017, 11, 31);
```

Литерал (JSON)

```
let circle = {x:0, y:0, radius:2};
```

```
let person = {
   name: "Jack Daniel",
   age: 64,
   married: true,
   occupation: "businessman",
   email: "jack.daniel@google.com"
};
```

Свойства объекта

```
table.caption = "Employees";
table.body = new Object();
table.body.rows = [
   {name:"Иванов", salary:1000},
   {name:"Петров", salary:1700}
]
table.body.color = "gray";
```

```
let names = "";
for(let name in obj){
   names += name + "\n";
}
alert(names);
```

```
delete table.body;
alert(table.body);
```

Объект как массив

```
table['caption'] = "Employees";
table['body'].color = "gray";
```

```
var addr = "";
for(i = 0; i < 4; i++) {
   addr += customer["address" + i] + '\n';
}
```

```
let box = {l:10, w:8, h:3};
for (dim in box){
   box[dim] *= 1.2;
}
```

Стандартные свойства и методы

constructor

```
let c = new Complex(3,-4);
c.constructor == Complex;                    //true
```

toString(), toLocaleString(), valueOf()

```
alert('c=' + c);                    // c=5 (valueOf)
alert('c=' + c.toString());         // c={3,4}
```

hasOwnProperty()

```
Math.hasOwnProperty("cos");     // true
c.hasOwnProperty("toString");   // false
```

Стандартные свойства и методы

propertyIsEnumerable()

```
var o = { x:1 };
o.propertyIsEnumerable("x");          // true
o.propertyIsEnumerable("y");          // false
o.propertyIsEnumerable("valueOf");    // false
```

isPrototypeOf()

```
var o = new Object( );
Object.prototype.isPrototypeOf(o);        // true
```

Массив

Конструктор

```
var a = new Array();
var a = new Array(1,7,15,'элемент',true);
var a = new Array(10);
```

Литерал

```
var a = [1,7,15,'элемент',true];
var b = [[1,{x:1, y:2}], [2, {x:3, y:4}]];
```

Элементы массива

Доступ к элементам

```
value = a[9];
a[3] = 3.14;
a[i+1] = 'привет';
a[a[i]] = a[0];
me['salary'] *= 1.5;
```

Добавление элементов

```
a[1001] = 9;
```

```
var fruits = ['Apple','Orange','Pineapple'];
fruits.push('Kiwi');
```

```
var c = new Point(-1,1);
c[0] = 'это новый элемент';
```

Длина массива

```
var a = new Array( );    // a.length == 0
a = new Array(10);       // a.length == 10
a = new Array(1,2,3);    // a.length == 3
a = [4, 5];              // a.length == 2
a[5] = -1;               // a.length == 6
a[49] = 0;               // a.length == 50
```

```
for(var i = 0; i < fruits.length; i++){
   if (fruits[i] != undefined){
      alert(fruits[i]);
   }
}
```

```
a.length = 2;
a.toString();                    // 4,5
```

Методы массива

join

```
[1,2,3].join();                    // '1,2,3'.split(',')
[1,2,3].join('*');                 // '1*2*3'
```

reverse

```
[1,2,3].reverse();                 // [3,2,1]
```

sort

```
['Kiwi','Apple','Orange'].sort();
                  // ['Apple','Kiwi','Orange']
[33, 4, 1111, 222].sort()    // [1111,222,33,4]
a.sort( function(a,b){return a-b;} );
```

concat

```
[1,2,3].concat(4,5,6);
[1,2,3].concat([4,5],[6]);
```

Методы массива

slice, splice

```
a=[1,2,3,4,5];
a.slice(0,3);           // [1,2,3]
a.splice(1,2,11,12); // [2,3,4]; a:[1,11,12,4,5]
```

push, pop

```
a.pop();                // 5 ; a:[1,11,12,4]
a.push(7);              // 5 ; a:[1,11,12,4,7]
```

shift, unshift

```
a.shift();              // 1 ; a:[11,12,4,7]
a.unshift(51,52);    // 5 ; a:[51,52,11,12,4,7]
```

toString, toLocaleString

```
["a", "b", "c"].toString( )  // 'a,b,c'
[1, [2,'c']].toString( )     // '1,2,c'
```

Роль JS в Web

Клиентский JavaScript

Окружение в браузере:

- Глобальный объект window.
- Иерархия объектов. DOM – её часть.
- Управляемая событиями модель программирования.

Объект window

- Глобальный объект
- Свойства:
  - document
  - window, self
  - глобальные переменные

```
var answer = 42;
window.answer = 42;
```

- Методы
  - open, close

Внедрение JS в HTML

```
<html>
   <head>
      <title>Today's Date</title>
      <script>
         function print_todays_date( ) {
            var d = new Date( );
            document.write(d.toLocaleString( ));
         }
      </script>
   </head>
   <body>
      The date and time are:<br>
      <script>
         print_todays_date( );
      </script>
   </body>
</html>
```

Внедрение JS в HTML

```
<script
   language="JavaScript1.5"
   type="text/javascript"
>
   . . .   // JS
</script>
```

```
<script src="url"></script>
```

```
<script src="url" async></script>
<script src="url" defer></script>
```

Выполнение JS

- Скрипты в порядке следования
- document.write – после </script>
- Для внешних файлов – аналогично
- Нужно учитывать последовательность разбора HTML
- Синхронные скрипты должны быть быстрыми
- При загрузке внешнего файла разбор останавливается

- При перезагрузке страницы window возвращается в исходное состояние

Безопасность JS

Отсутствие возможностей

- Не поддерживает работу с локальными файлами
- Работа с сетью ограничена
  - Загрузка URL, отправка данных
  - Отправка почты
  - Нет сетевых примитивов
- Компоненты (плагины, ActiveX, Java) имеют больше возможностей
- Приватность (тип браузера, IP, история)

Основы JavaScript

- Объявление переменных, объектов, функций и массивов на JavaScript
- Создание массивов JavaScript для хранения данных
- Задание объектов JavaScript как хранилищ пар “ключ-значение”
- Доступ к свойствам объекта
- Практика: Написание кода JavaScript для выполнения тестов в Jasmine

Версии JavaScript

ES6 = ECMAScript 2015

ES7 = ECMAScript 2016

ES8 = ECMAScript 2017

```
<script src="https://cdn.polyfill.io/v2/polyfill.min.js">
</script>
```

Объявление переменной

- Явно

```
var i=10;
var n, sum;
var message='Привет';
var i=0, j=0, k;        // undefined
```

- Неявно (глобальные)

```
j = "десять";
```

- Допускаются повторные объявления

```
var j=0;
var j=1;
```

Числовые литералы

- Десятичные целые

```
0
3
10000000
```

- Шестнадцатеричные целые

```
0xff  // 15*16 + 15 = 255
0xCAFE911
```

- Восьмеричные целые

```
0377  // 3*64 + 7*8 + 7 = 255
```

- С плавающей точкой

```
3.14
2345.789
.333333333333333333
6.02e23         // 6.02  x  10^23
1.4738223E-32   // 1.4738223  x  10^-32
```

Числовая арифметика

до 15 цифр

```
var y = 9999999999999999;  // 10000000000000000
```

до 17 десятичных знаков

```
var x = 0.2 + 0.1;         // 0.30000000000000004
```

```
var x = (0.2 * 10 + 0.1 * 10) / 10;       // 0.3
```

Специальные значения чисел

Infinity	бесконечность
NaN	не число
Number.MAX_VALUE	максимальное число
Number.MIN_VALUE	минимальное число (ближайшее к нулю)
Number.NaN	не число
Number.POSITIVE_INFINITY	+ бесконечность
Number.NEGATIVE_INFINITY	- бесконечность

Работа с числами

```
синус_x = Math.sin(x);

hypot = Math.sqrt(x*x + y*y);

var x = 33;
var y = x.toString(2);  // y is "100001"

var y = (257).toString(0x10);  // y is "101"

var q = 130/0; // Infinity
var s = 0/a;   // NaN
var t = 0/0;   // NaN
```

Символьные строки

Нет типа char

```
""  // пустая строка (0 символов)
'testing'
"3.14"
'name="myform"'
"Wouldn't you like breaks?"
"Это двустрочная\nстрока"
"П – отношение длины окружности к её диаметру"
```

При включении в HTML рекомендуется так:

```
<a href="" onclick="alert('Thank you')">
  Click Me
</a>
```

Символьные строки - экранирование

```
'You\'re right, it can\'t be a quote'
```

```
\0 -  NUL (\u0000)
\b -  Backspace (\u0008)
\t -  Horizontal tab (\u0009)
\n -  Newline (\u000A)
\v -  Vertical tab (\u000B)
\f -  Form feed (\u000C)
\r -  Carriage return (\u000D)
\" -  Double quote (\u0022)
\' -  Apostrophe or single quote (\u0027)
\\ -  Backslash (\u005C)
\xXX -  Символ Latin-1 с 16-ричным кодом XX
\uXXXX  -  Символ Unicode с 16-ричным кодом XXXX
```

Работа с символьными строками

Конкатенация

```
msg = "Hello, " + "world";   // "Hello, world"
greeting =
   "Welcome to my home page," +
   " " +
   name;
```

Методы

```
last_char = s.charAt(s.length - 1); // с нуля
sub = s.substring(1,4);
i = s.indexOf('a');
lastSymbol = s[s.length-1];  //ES5
```

строки – не объекты

Логический тип

true, false

```
let isPositive = b > 0;
if (isPositive)
  b = b - 1;
else
  b = b + 1;
```

Функции

Предопределённые

```
Math.sin()
```

Определённые в программе

```
function square(x){
  return x*x;
}
```

Функция - литерал

```
var square = function(x){ return x*x; }
```

Arrow functions (ES6)

```
var square = (x) => { return x*x; }
var square = x => x*x;
var hello = () => { document.write('Hello'); }
```

Объекты

Набор именованных значений (свойства, properties)

```
image.width
```

Значения свойств могут быть любого типа

```
image.width + 10                        //число
document.myform.target                  //объект
order.items[2]                          //массив
document.write()                        //функция
```

Объект = ассоциативный массив

```
image.width === image['width']
image['1st layer']

var dim='width';
document.write(image[dim]);
```

Создание объектов

Функция-конструктор

```
var o = new Object();
var now = new Date();
var pattern = new RegExp("\\sjava\\s", "i");
```

Литерал

```
var image = {width: 100, height: 150};
var rectangle = {
   upperLeft: { x: 2, y: 2 },
   lowerRight: { x: 4, y: 4}
};
var square = {
   upperLeft:{ x:point.x, y:point.y },
   lowerRight:{ x:(point.x+side), y:(point.y+side) }
};
```

JSON = JavaScript Object Notation

Массивы

Набор пронумерованных значений. Индекс (ключ).

```
document.images[1].width        //объект
cells[1][2]                     //массив
```

Конструктор

```
var a = new Array( );
a[0] = 1.2;
a[1] = "JavaScript";
a[2] = true;
a[3] = { x:1, y:3 };

var a = new Array(1.2, "JavaScript", true, { x:1, y:3 });

var a = new Array(10);
```

Литерал (JSON)

```
var a = [1.2, "JavaScript", true, { x:1, y:3 }];
var sparseArray = [1,,,,5];
```

null и undefined

null = специальное значение "отсутствие значения"

```
var n = null;
```

undefined = неинициализированная переменная или свойство

```
var n;
myObject.prop2016
```

```
null == undefined  // true
null === undefined //false
```

Полезные классы

Date

```
var xmas = new Date(2017, 1, 7);
xmas.setFullYear (xmas.getFullYear+1);
var weekday = xmas.getDay( );
document.write("Today is: " + now.toLocaleString( ));
```

RegExp

```
var pattern = /s$/g;
var pattern = new RegExp("s$","g");
"JavaScript".search(/script/i);
```

Классы-обёртки

- String

```
var s = "hello world";                // примитив
var S = new String("Hello World");    // объект String
S.valueOf;                            // примитив
```

- Boolean
- Number
- Symbol (ES6)

Предпочтительнее использовать примитивы, а не классы-обёртки.

A symbol is a unique and immutable data type. It may be used as an identifier for object properties. The Symbol object is an implicit object wrapper for the symbol primitive data type.

```
var sym1 = Symbol();
var sym2 = Symbol("foo");
var sym3 = Symbol("foo");
```

создаёт 3 разных символа.

Преобразование типов

- Неявное преобразование

```
document.write(120);
```

	String	**Number**	**Boolean**	**Object**
undefined	"undefined"	NaN	false	Error
null	"null"	0	false	Error
непустая строка	-	Число или NaN	true	String
""	-	0	false	String
0	"0"	-	false	Number
NaN	"NaN"	-	false	Number
∞	"Infinity"	-	true	Number
−∞	"-Infinity"	-	true	Number
остальные числа	строка	-	true	Number
true	"true"	1	-	Boolean
false	"false"	0	-	Boolean
Object	toString()	valueOf() or toString() or NaN	true	-

Неявное преобразование

Массивы

```
> Number([]), Number([15]), Number([true]), Number([1,2,3])

0, 15, NaN, NaN
```

Объекты – только в число или строку

1. Если valueOf() даёт примитивный тип, преобразовать.
2. Если toString() даёт примитивный тип, преобразовать.
3. Ошибка (TypeError).

Алгоритм применяется для + < <= > >=

Исключение – Date

valueOf() - количество миллисекунд с 1970-01-01,
toString() - текстовое представление даты в локали.

Для операции + предполагается, что это конкатенация, и valueOf() не применяется. Для операторов сравнения предполагается сравнение дат как чисел (раньше/позже) и применяется valueOf().

Явное преобразование типов

- parseInt, parseFloat

```
parseInt("3 blind mice");     // 3
parseFloat("3.14 meters");    // 3.14
parseInt("12.34");            // 12
parseInt("0xFF");             // 255
parseInt("eleven");           // NaN
parseFloat("$72.47");         // NaN
```

- toString, toFixed, toExponential, toPrecision

```
var n = 78;
binary_string = n.toString(2);        // "10001"
octal_string = "0" + n.toString(8);   // "021"
hex_string = "0x" + n.toString(16);   // "0x11"

var n = 123456.789;
n.toFixed(2);          // "123456.79"
n.toExponential(3);    // "1.235e+5"
n.toPrecision(4);      // "1.235e+5"
n.toPrecision(7);      // "123456.8"
```

Явное преобразование типов

- Конструкторы классов-обёрток Number, Boolean, String, Object

```
var x_as_string  = String(x);
var x_as_number  = Number(x);
var x_as_boolean = Boolean(x);
```

- синтаксический сахар – неявное преобразование

```
var x_as_string  = x + "";
var x_as_number  = +x;
var x_as_boolean = !!x;
```

Объект

Конструктор

```
let o = new Object( );
let now = new Date( );
var new_year_evening = new Date(2017, 11, 31);
```

Литерал (JSON)

```
let circle = {x:0, y:0, radius:2};
```

```
let person = {
   name: "Jack Daniel",
   age: 64,
   married: true,
   occupation: "businessman",
   email: "jack.daniel@google.com"
};
```

Свойства объекта

```
table.caption = "Employees";
table.body = new Object();
table.body.rows = [
   {name:"Иванов", salary:1000},
   {name:"Петров", salary:1700}
]
table.body.color = "gray";
```

```
let names = "";
for(let name in obj){
   names += name + "\n";
}
alert(names);
```

```
delete table.body;
alert(table.body);
```

Объект как массив

```
table['caption'] = "Employees";
table['body'].color = "gray";
```

```
var addr = "";
for(i = 0; i < 4; i++) {
   addr += customer["address" + i] + '\n';
}
```

```
let box = {l:10, w:8, h:3};
for (dim in box){
   box[dim] *= 1.2;
}
```

Стандартные свойства и методы

constructor

```
let c = new Complex(3,-4);
c.constructor == Complex;                    //true
```

toString(), toLocaleString(), valueOf()

```
alert('c=' + c);                  // c=5 (valueOf)
alert('c=' + c.toString());       // c={3,4}
```

hasOwnProperty()

```
Math.hasOwnProperty("cos");     // true
c.hasOwnProperty("toString");   // false
```

Стандартные свойства и методы

propertyIsEnumerable()

```
var o = { x:1 };
o.propertyIsEnumerable("x");         // true
o.propertyIsEnumerable("y");         // false
o.propertyIsEnumerable("valueOf");   // false
```

isPrototypeOf()

```
var o = new Object( );
Object.prototype.isPrototypeOf(o);       // true
```

Массив

Конструктор

```
var a = new Array();
var a = new Array(1,7,15,'элемент',true);
var a = new Array(10);
```

Литерал

```
var a = [1,7,15,'элемент',true];
var b = [[1,{x:1, y:2}], [2, {x:3, y:4}]];
```

Элементы массива

Доступ к элементам

```
value = a[9];
a[3] = 3.14;
a[i+1] = 'привет';
a[a[i]] = a[0];
me['salary'] *= 1.5;
```

Добавление элементов

```
a[1001] = 9;
```

```
var fruits = ['Apple','Orange','Pineapple'];
fruits.push('Kiwi');
```

```
var c = new Point(-1,1);
c[0] = 'это новый элемент';
```

Длина массива

```
var a = new Array( );   // a.length == 0
a = new Array(10);      // a.length == 10
a = new Array(1,2,3);   // a.length == 3
a = [4, 5];             // a.length == 2
a[5] = -1;              // a.length == 6
a[49] = 0;              // a.length == 50
```

```
for(var i = 0; i < fruits.length; i++){
   if (fruits[i] != undefined){
      alert(fruits[i]);
   }
}
```

```
a.length = 2;
a.toString();                  // 4,5
```

Методы массива

join

```
[1,2,3].join();              // '1,2,3'.split(',')
[1,2,3].join('*');           // '1*2*3'
```

reverse

```
[1,2,3].reverse();           // [3,2,1]
```

sort

```
['Kiwi','Apple','Orange'].sort();
                  // ['Apple','Kiwi','Orange']
[33, 4, 1111, 222].sort()    // [1111,222,33,4]
a.sort( function(a,b){return a-b;} );
```

concat

```
[1,2,3].concat(4,5,6);
[1,2,3].concat([4,5],[6]);
```

Методы массива

slice, splice

```
a=[1,2,3,4,5];
a.slice(0,3);          // [1,2,3]
a.splice(1,2,11,12); // [2,3,4]; a:[1,11,12,4,5]
```

push, pop

```
a.pop();               // 5 ; a:[1,11,12,4]
a.push(7);             // 5 ; a:[1,11,12,4,7]
```

shift, unshift

```
a.shift();             // 1 ; a:[11,12,4,7]
a.unshift(51,52);      // 5 ; a:[51,52,11,12,4,7]
```

toString, toLocaleString

```
["a", "b", "c"].toString( )  // 'a,b,c'
[1, [2,'c']].toString( )     // '1,2,c'
```

Роль JS в Web

Клиентский JavaScript

Окружение в браузере:

- Глобальный объект window.
- Иерархия объектов. DOM – её часть.
- Управляемая событиями модель программирования.

Объект window

- Глобальный объект
- Свойства:
  - document
  - window, self
  - глобальные переменные

```
var answer = 42;
window.answer = 42;
```

- Методы
  - open, close

Внедрение JS в HTML

```
<html>
   <head>
      <title>Today's Date</title>
      <script>
         function print_todays_date( ) {
            var d = new Date( );
            document.write(d.toLocaleString( ));
         }
      </script>
   </head>
   <body>
      The date and time are:<br>
      <script>
         print_todays_date( );
      </script>
   </body>
</html>
```

Внедрение JS в HTML

```
<script
   language="JavaScript1.5"
   type="text/javascript"
>
   . . .   // JS
</script>
```

```
<script src="url"></script>
```

```
<script src="url" async></script>
<script src="url" defer></script>
```

Выполнение JS

- Скрипты в порядке следования
- document.write – после </script>
- Для внешних файлов – аналогично
- Нужно учитывать последовательность разбора HTML
- Синхронные скрипты должны быть быстрыми
- При загрузке внешнего файла разбор останавливается

- При перезагрузке страницы window возвращается в исходное состояние

Безопасность JS

Отсутствие возможностей

- Не поддерживает работу с локальными файлами
- Работа с сетью ограничена
  - Загрузка URL, отправка данных
  - Отправка почты
  - Нет сетевых примитивов
- Компоненты (плагины, ActiveX, Java) имеют больше возможностей
- Приватность (тип браузера, IP, история)