

Типы данных



- Числа
- Строки
- Логический тип
- Функции
- Объекты
- Массивы
- Значения `null` и `undefined`
- Объектные оболочки примитивных типов
- Конвертация типов данных

Числа



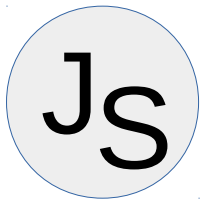
Один тип: 64 битные с плавающей точкой

$$52 + 11 + 1 = 64$$

$\pm 1.7976931348623157 \times 10^{308}$

$\pm 5 \times 10^{-324}$

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
.333333333333333333333333
6.02e23 // 6.02 x 1023
1.4738223E-32 // 1.4738223 x 10-32
```

Числовая арифметика



до 15 цифр

```
var y = 999999999999999999; // 10000000000000000000
```

до 17 десятичных знаков

```
var x = 0.2 + 0.1; // 0.3000000000000000004
```

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

Специальные значения чисел



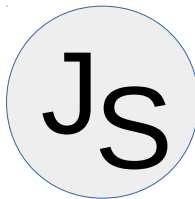
Infinity	бесконечность
NaN	не число
Number.MAX_VALUE	максимальное число
Number.MIN_VALUE	минимальное число (ближайшее к нулю)
Number.NaN	не число
Number.POSITIVE_INFINITY	+ бесконечность
Number.NEGATIVE_INFINITY	- бесконечность

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



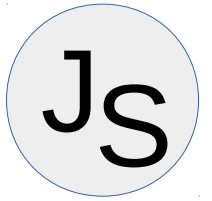
Нет типа char

```
" " // 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

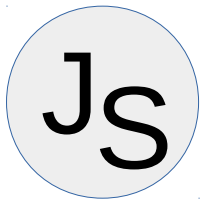
Логический тип



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;  
}
```

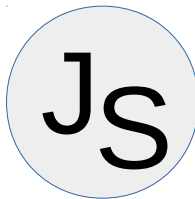
Функция - литерал

```
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

<code>image.width + 10</code>	//число
<code>document.myform.target</code>	//объект
<code>order.items[2]</code>	//массив
<code>document.write()</code>	//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"

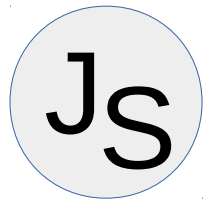
```
var n = null;
```

undefined = неинициализированная переменная или свойство

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

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

Полезные классы



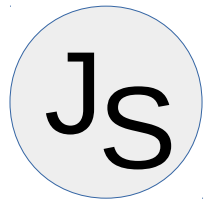
Date

```
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

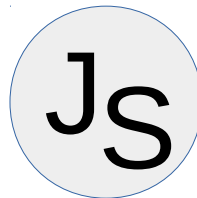
```
var s = "hello world";           // примитив
var S = new String("Hello World"); // объект String
S.valueOf();                     // примитив
```

- Boolean

- Number

- Symbol (ES6)

Преобразование типов

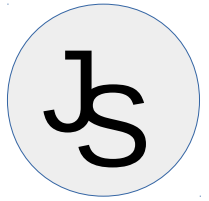


- Неявное преобразование

```
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

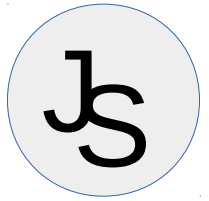
```
> Number([], Number([15]), Number([true]), Number([1,2,3]))  
0, 15, NaN, NaN
```

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

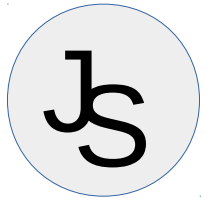
```
parseInt("3 blind mice");      // 3
parseFloat("3.14 meters");     // 3.14
parseInt("12.34");             // 12
parseInt("0xFF");              // 255
parseInt("eleven");            // NaN
parseFloat("$72.47");          // NaN
```

- 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;
```

Типы данных



- Числа
- Строки
- Логический тип
- Функции
- Объекты
- Массивы
- Значения null и undefined
- Объектные оболочки примитивных типов
- Конвертация типов данных

Числа

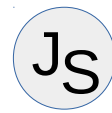


Один тип: 64 битные с плавающей точкой

$$52 + 11 + 1 = 64$$

$$\begin{aligned} &\pm 1.7976931348623157 \times 10^{308} \\ &\pm 5 \times 10^{-324} \end{aligned}$$

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 1023
1.4738223E-32 // 1.4738223 x 10-32
```


Числовая арифметика



до 15 цифр

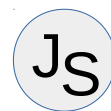
```
var y = 999999999999999; // 100000000000000000
```

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



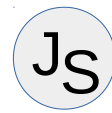
Infinity	бесконечность
NaN	не число
Number.MAX_VALUE	максимальное число
Number.MIN_VALUE	минимальное число (ближайшее к нулю)
Number.NaN	не число
Number.POSITIVE_INFINITY	+ бесконечность
Number.NEGATIVE_INFINITY	- бесконечность

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



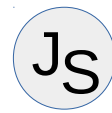
Нет типа char

```
" " // 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

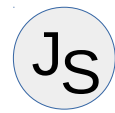
```
image.width + 10           //число
document.myform.target     //объект
order.items[2]             //массив
document.write()           //функция
```

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
```

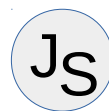
Конструктор

```
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"

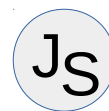
```
var n = null;
```

undefined = неинициализированная переменная или свойство

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

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

Полезные классы



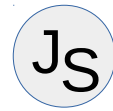
Date

```
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)

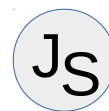
Предпочтительнее использовать примитивы, а не классы-обёртки.

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;
```