

# JS

javascript

```
> typeof NaN           > true==1
< "number"             < true

> 9999999999999999     > true===1
< 10000000000000000    < false

> 0.5+0.1==0.6         > (!+[[]+[]+![]).length
< true                 < 9

> 0.1+0.2==0.3         > 9+"1"
< false               < "91"

> Math.max()           > 91-"1"
< -Infinity           < 90

> Math.min()           > []==0
< Infinity            < true

> []+[]
< ""

> []+{}
< "[object Object]"

> {}+[]
< 0

> true+true+true===3
< true

> true-true
< 0
```



# Java vs JavaScript

объект-ориентированный язык

запускается на сервере

строго типизирован

мультипарадигменный язык

браузеры и серверы

не типизирован

JavaScript === ECMAScript

[Javascript на одной картинке](#)

# Переменные

# Переменные

var	устаревшее	
let	можно изменять	EcmaScript 2015 года (ES6)
const	нельзя изменять	EcmaScript 2015 года (ES6)

# Примитивные типы

# Числа

number       $-(2^{53} - 1)$  до  $(2^{53} - 1)$ , Infinity, -Infinity, NaN  
-9007199254740991    —    +9007199254740991

доби      ->    .toFixed()

infinity    ->    1e999

NaN        ->    .isNaN()

BigInt

```
const int = 4
```



# Строки

```
const double      = ""  
const single     = ''  
const literals   = `текст ${переменная}`
```

# Boolean

True / false

Бинарные (0) (1)

# undefined / null

undefined

неизвестное или неопределённое значение

null

отсутствие (намеренно) значения  
→ задается программистом

# Symbol

Symbol

создания скрытых свойств объектов

# Примитивы

Boolean

Null

Undefined

Number

BigInt

String

Symbol

**Всё остальное объекты**

# Преобразование типов

## Явные

`String(42)`

`Number("42")`

`Boolean(42)`

## Неявные

логические –

строка –

число –

&& и ||

+ и один операндов строка

<, <=, >, >=

/ \* - + (+ если не противоречит правилу строк)

`4+5+6+"7"`

# Строгое и нестрогое равенство

(==)                Пытается привести типы к одному

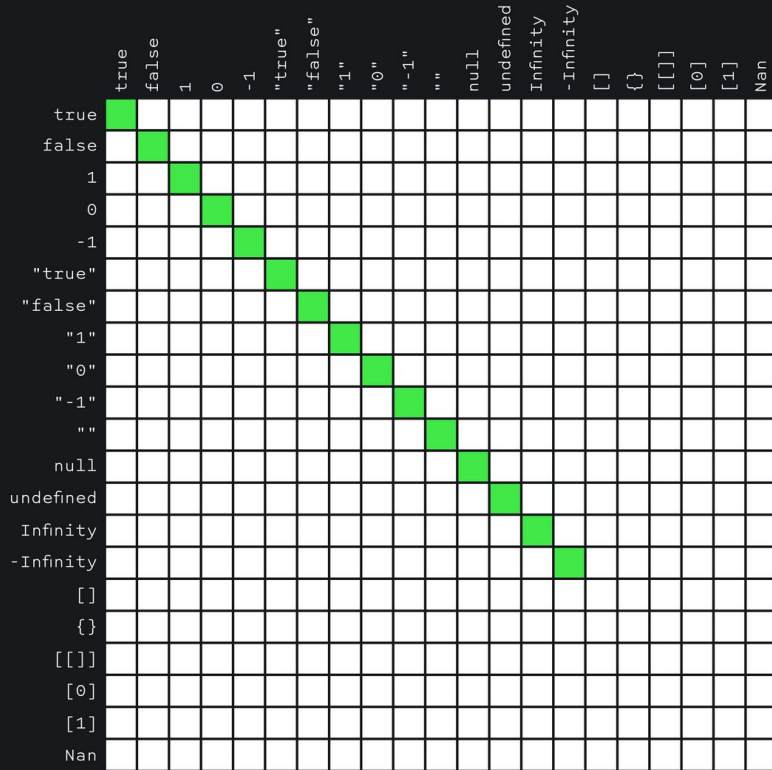
(===)             Нет изменений в типах

## Нестрогое (==)

[illegible]



## Строгое (===)



# Массив

## массив

## коллекция элементов

```
const array = []
```

```
const zeka = ['zeka', 30, 'developer']
```

push —> для добавления в конец массива

unshift —> для добавления в начало массива

indexOf → индексом элемента.

includes —> элемент есть в массиве:





# Функции

```
function (параметры) {  
    тело функции  
}
```

```
const myFunction = function() {  
    console.log('In')  
}
```

```
myFunction()
```

# Логические операторы

# Логические операторы

&& -

логический оператор И

||

логический оператор ИЛИ

!

логический оператор НЕ

# Циклы

# ЦИКЛЫ

if else  
switch  
while  
for  
try catch



## If ... else

```
if (условие) {  
  
} else {  
  
}
```

# switch

```
switch (имя_переменной_значение_которой_сравниваем) {  
    case значение:  
        // код  
        break  
}
```

# While

```
while (условие) {  
    // код  
}
```

# For

```
for (инициализация; условие; завершающая операция ) {  
    // тело цикла  
}
```