

Типы и Преобразование

8 ТИПОВ значений.

Тип — это встроенный набор характеристик, который однозначно идентифицирует поведение конкретного значения и отличает его от других значений, как для движка, так и для разработчика.

7 типов - примитивы

(значение передается как копия)

```
var a = 2;  
var b = a; // `b` всегда содержит копию  
значения из `a`  
b++;  
a; // 2  
b; // 3
```

1 тип - ссылки

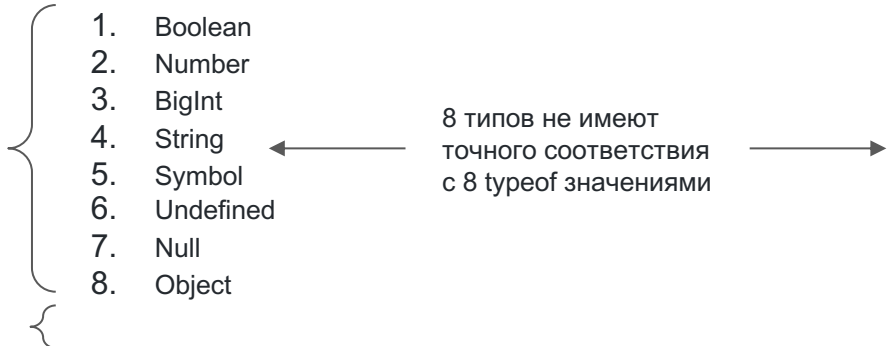
(значение передается как ссылка)

```
var c = [1,2,3];  
var d = c; // `d` - ссылка на общее значение  
[1,2,3]  
d.push( 4 );  
c; // [1,2,3,4]  
d; // [1,2,3,4]
```

Важно заметить, что значение [1, 2, 3] не "принадлежит" ни c, ни d -- это просто равноправные ссылки на значение.

function - это подтип объекта (технически, «вызываемый объект»), а не отдельный тип.

Массивы - это тоже подтип объекта с расширенным поведением.

- 
1. Boolean
 2. Number
 3. BigInt
 4. String
 5. Symbol
 6. Undefined
 7. Null
 8. Object

8 типов не имеют
точного соответствия
с 8 typeof значениями

1. 'boolean'
2. 'number'
3. 'bigint'
4. 'string'
5. 'symbol'
6. 'undefined'
7. 'function'
8. 'object'

typeof - оператор проверяет тип заданного значения и всегда возвращает одно из 8 **строковых** значений.

```
var qwe = null;  
typeof qwe === 'object' // true - известная бага в JS, по логике  
должно было бы быть typeof qwe === 'null' но нет.  
Поэтому null нужно сравнивать только с null.  
qwe === null // true  
typeof [] === 'object' // true
```

Встроенные объекты - являются ссылочными типами

<ul style="list-style-type: none">• Array• Date• Error• Function• Object• RegExp• Promise• Map• Set• ...	<p>Чтобы узнать тип объекта используй оператор <i>instanceof</i>:</p> <p>Чтобы протестировать что объект не данного типа:</p> <p>Для определения массива используй <i>Array.isArray()</i>:</p> <p>При передаче значения между разными контекстами (frames или windows), <i>instanceof</i> будет выдавать ошибку потому что глобальные объекты и функции конструкторы разные.</p>	<pre>{ a: 'qwe' } instanceof Object // true /^a\$/ instanceof RegExp // true !({ a: 'qwe' } instanceof Array) // true Array.isArray([8, 17, 37]) // true</pre>
---	--	---

Литерал - это упрощенный синтаксис для создания значения. Не нужно использовать оператор *new* и функцию конструктор.

<pre>const bool = true; Boolean(true) const num = 8; Number(8) const bigInt = 8n; const string = 'qwe'; String('qwe') const sym = Symbol(); const obj = {}; const arr = [8, 17, 37] const regexp = /^qwe\$/i const a = null; const b = undefined; function func() { return a * 2; } const func = function() { return a * 2; } const func = () => a * 2;</pre>	<pre>// new // new // BigInt(8) // new // Symbol() // new Object() // new Array(8, 17, 37) // new RegExp('^qwe\$', 'i') // - // - // new Function('a', 'return a * 2;')</pre>	<p><i>Literal:</i></p> <pre>var book = { "name": "The Principles of Object-Oriented JavaScript", "year": 2014 };</pre> <p><i>Use function constructor:</i></p> <pre>var book = new Object(); book.name = "The Principles of Object-Oriented JavaScript"; book.year = 2014;</pre>
--	---	--

Преобразование типов

1. false
2. null
3. undefined
4. +0, -0
5. Nan
6. ""



Только 6 false значение, все остальные дают true.

```
var a = "false";  
var b = "0";  
var c = "";  
var d = Boolean( a && b && c ); // true
```

// todo: describe == and ===

```
var a = [];  
var b = {};  
var c = function(){};  
var d = Boolean( a && b && c ); // true
```

// пустой массив
// пустой объект
// пустая функция

```
String(42); // '42'  
Number('3.14'); // 3.14  
Boolean( 0 | null | " ) // false
```

```
var a = 42;  
var b = a.toString(); // '42'
```

```
var c = "3.14";  
var d = +c; // 3.14
```