

Лекция 07 апреля 2020 года.
«Базовый JavaScript. Лексическая структура.
Типы данных, значения и переменные»

Основной теоретический материал находится в файле «[07 апреля 2020 г – WEB – Лекция.pdf](#)», составлен на основе книги *Флэнаган Дэвид. JavaScript. Подробное руководство*. 6-е изд. – Пер. с англ. – СПб: Символ-Плюс, 2012. – 1080с., ил.:

Дополнительно информацию по теме можно прочитать в книге *Закас Николас. JavaScript для профессиональных веб-разработчиков* / [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 960с.: ил. – (Серия «Для профессионалов»).

Синтаксис

Чувствительность к регистру.

test и Test различны.

Идентификатор:

- первый знак: буква, знак подчеркивания (_) или знак доллара (**\$**);
- все остальные знаки: буквы, знаки подчеркивания, знаки доллара или цифрами.

Буквы: из расширенного набора ASCII или из Юникода (например, À и Ё).

«Верблюжья» нотация: myName, learnJavaScript

Комментарии.

```
// однострочный комментарий
/*
* Это многострочный
* комментарий
*/
```

Строгий режим.

```
"use strict" // в ECMAScript 5
```

Эта директива, переводящая JavaScript в строгий режим. Такой синтаксис был выбран специально, чтобы исключить конфликты с ECMAScript 3. Строгий режим можно включить и для отдельной функции, добавив эту директиву в начало тела функции:

```
function doSomething( ) {
    "use strict " ;
    // тело функции
}
```

Инструкции.

```
var sum = a+b // верно даже без точки с запятой, но не рекомендуется
var diff = a - b; // правильно и рекомендуется
```

Пример.

```
var y=x+y
(a+b).toString()
```

```
var y=x+y(a+b).toString()
```

Оформление блоков.

```
if (test ) {  
    test = false;  
    alert (test ) ;  
}
```

Ключевые слова.

В ECMAScript 2015

arguments	delete	if	this
break	do	import	throw
case	else	in	try
class	export	instanceof	typeof
catch	extends	let	var
const	eval	new	void
continue	finally	return	while
debugger	for	super	with
default	function	switch	yield

Зарезервированные слова.

enum	interface	char	long
await	private	double	native
implements	public	final	short
package	abstract	float	synchronized
protected	boolean	goto	transient
static	byte	int	volatile

Переменные.

ECMAScript-переменные типизированы слабо.

Определение переменной

```
var message;  
var message = "hi" ;
```

«Изменение» типа

```
var message = "hi" ;  
message = 100; // допустимо , но не рекомендуется
```

Объявление локальной переменной

```
function test ( ) {  
    var message = " hi " ; // локальная переменная  
}  
test ( ) ;  
alert (message) ; // ошибка !
```

Объявление глобальной переменной

```
function test ( ) {  
    message = " hi " ; // глобальная переменная  
}  
test ( ) ;
```

```
alert (message ) ; // "hi"
```

Множественное определение переменных

```
var found = false,  
    age = 29 ;
```

Типы данных

Примитивные (primitive types):

- неопределенный (undefined),
- нулевой (null),
- логический (boolean),
- числовой (number),
- строковый (string).

Сложные типы данных (object, объектные, или ссылочные):

- Объекты.
- Массивы.
- Функции.

Оператор typeof.

typeof value

Undefined	"undefined"
Null	"object"
Boolean	"boolean"
Number или NaN	"number"
String	"string"
Function	"function"
Любой другой тип	"object"

Тип Undefined

```
var message;  
alert (message == undefined ) ; // true
```

```
var message = undefined ;  
alert (message == undefined ) ; // true
```

Пример 1. Объявлена ли переменная?

```
var message; // переменная объявляется, но имеет значение undefined  
// следующая переменная не объявляется  
// var age  
alert ( message ) ; // "undefined"  
alert ( age ) ; // ошибка
```

```
var message; // переменная объявляется, но имеет значение undefined  
// следующая переменная не объявляется
```

```
// var age
alert (typeof message ) ; // "undefined"
alert (typeof age ) ; // "undefined"
```

Пример 2. Undefined – не является зарезервированным словом

```
(function(){
    var undefined = 'foo';
    console.log(undefined, typeof undefined); })
();
// выводит 'foo string'
```

Tun Null

Пример. Объявление «нулевого» объекта

```
var car = null;
alert (typeof car) ; // "object"
if (car != null ) {
    // какие-то действия с car
}
```

Пример. Значение undefined является производным от null:

```
alert ( null == undefined ) ; // true
```

НО!!!

```
var value1 = undefined; // не рекомендуется
var value2 = null; // указатель на пустой объект
```

Tun Boolean

```
var found = true;
var lost = false;
var t = new Boolean(0); // false;
```

НО!!!

```
console.log(true == 1) // true
console.log(false == 0) // true
```

Функция приведения типов Boolean ():

```
var message = "Hello world ! " ;
var messageAsBoolean = Boolean (message) ;
```

Тип данных	Значение, преобразуемые в true	Значение, преобразуемые в false
boolean	true	false
string	Любая непустая строка	"" (пустая строка)
number	Любое ненулевое число (включая бесконечность)	0, NaN
object	Любой объект	null
undefined	-	undefined

Пример. «Автоматическое» преобразование в управляющих инструкциях

```
var message = "Hello world ! " ;
```

```
if (message) {
  console.log( "Value is true " ) ;
}
// ???
```

Числовой тип (Number).

1) Числа ДТ по IEEE-754: ~~целые числа~~, \pm Infinity, NaN

2)

```
Number.MAX_VALUE; // 1.7976931348623157e+308
Number.MIN_VALUE; // 5e-324
```

С EcmaScript6:

```
Number.MAX_SAFE_INTEGER // 9007199254740991
Number.MIN_SAFE_INTEGER // -9007199254740991
Number.isSafeInteger()
```

Бесконечность

Number.NEGATIVE_INFINITY – отрицательная бесконечность (-Infinity),
Number.POSITIVE_INFINITY – положительная бесконечность (Infinity)

Infinity не число!!!.

```
var result = Number.MAX_VALUE + Number.MAX_VALUE;
isFinite(result); // false
```

```
4/+0; // Infinity
4/-0; // -Infinity
4/"a"; // NaN
```

3)

```
var intNum1 = new Number(55); // создаёт число 55
var intNum2 = 55 ; // тоже
```

4)

Пример. Представление чисел в разных СС:

```
var octalNum1 = 070; // 56 в восьмеричном формате
var octalNum2 = 079; // недопустимое восьмеричное значение,
// интерпретируется как 79
var octalNum3 = 08; // недопустимое восьмеричное значение,
// интерпретируется как 8
```

```
var hexNum1 = 0xA; // 10 в шестнадцатеричном формате
var hexNum2 = 0x1f; // 31 в шестнадцатеричном формате
```

```
var binNum1 = 0B01000; // 8 в двоичном виде
```

5)

Пример. Преобразование строк с числами в числа

```
Number('123') // 123
```

```

Number('') // 0
Number('0x11') // 17
Number('0b11') // 3
Number('0o11') // 9
Number('foo') // NaN
Number('100a') // NaN

```

Пример. Преобразование чисел в строки

```

var n = 123456.789;
n.toFixed(0); // "123457"
n.toFixed(2); // "123456.79"
n.toFixed(5); // "123456.78900"

n.toExponential(1); // "1.2e+5"
n.toExponential(3); // "1.235e+5"

n.toPrecision(4); // "1.235e+5"
n.toPrecision(7); // "123456.8"
n.toPrecision(10); // "123456.7890"

```

Пример. Опять строки в числа.

```

parseInt("3 blind mice") // => 3
parseFloat(" 3.14 meters") // => 3.14
parseInt("-12.34") // => -12
parseInt("0xFF") // => 255
parseInt("0xff") // => 255
parseInt("-0XFF") // => -255
parseFloat(".1") // => 0.1
parseInt("0.1") // => 0
parseInt(".1") // => NaN: целые числа не могут начинаться с "."
parseFloat("$72.47"); // => NaN: числа не могут начинаться с "$"

parseInt("11", 2); // => 3 (1*2 + 1)
parseInt("ff", 16); // => 255 (15*16 + 15)
parseInt("zz", 36); // => 1295 (35*36 + 35)
parseInt("077", 8); // => 63 (7*8 + 7)
parseInt("077", 10); // => 77 (7*10 + 7)

```

Значения с плавающей точкой

1)

```

var floatNum1 = 1.1;
var floatNum2 = 0.1;
var floatNum3 = .1; // допустимо, но не рекомендуется

```

Преобразование в целое число:

```

var floatNum1=1.; // нет разрядов после десятичной точки,
// интерпретируется как целое число 1
// Number.isInteger(1.) -> true
var floatNum2 = 10.0; // нет дробной части -

```

```
// интерпретируется как целое число 10
// Number.isInteger(10.0) -> true
```

2)

Экспоненциальный формат записи чисел:

```
var floatNum1 = 3.125e7; // 31250000
var floatNum2 = 3e-17; // 0.000000000000000003
```

НО! Автоматическое преобразование в экспоненциальный формат (> 6 знаков):
0.0000003 преобразуется в 3e-7

3)

Точность вычислений (17 десятичных разрядов):

```
if (0.1 + 0.2 == 0.3 ) { // не верно, 0.300000000000000004
    console.log("результат 0.3") ;
}
```

НО!

```
0.05 + 0.25 == 0.3 // верно
0.15 + 0.15 == 0.3 // верно
```

NaN

Number.NaN – специальное значение для представления «не числа».

Number.isNaN()

Определяет, является ли переданное значение значением NaN.

Примеры.

```
7/0 == NaN; // true
NaN/10 == NaN; // true
NaN === NaN; // false
Number.NaN === NaN; // false
isNaN(NaN); // true
isNaN(Number.NaN); // true
isNaN("10"); // false - может быть преобразовано в число 10
isNaN("green"); // true - не может быть преобразовано в число
isNaN(true); // false - может быть преобразовано в число 1
```

Строки (или String)

Строки – последовательности 16-разрядных знаков Юникода

```
var firstName = "Nicholas";
var lastName = 'Zakas';
"" // пустая строка
'name="Anna"'
"n=3.14"

console.log(String('лето')); // "лето", аналог s="лето"
console.log(new String("зима")); // String [ "з", "и", "м", "а" ]
```

Esc-последовательности – символные литералы:

<code>\0</code>	нулевой символ (символ NUL)	
<code>\'</code>	одинарная кавычка	<code>\u0027</code>
<code>\"</code>	двойная кавычка	<code>\u0022</code>
<code>\\</code>	обратный слэш	<code>\u005C</code>
<code>\n</code>	новая строка	<code>\u000A</code>
<code>\r</code>	возврат каретки	<code>\u000D</code>
<code>\v</code>	вертикальная табуляция	<code>\u000B</code>
<code>\t</code>	табуляция	<code>\u0009</code>
<code>\b</code>	забой	<code>\u0008</code>
<code>\f</code>	подача страницы	<code>\u000C</code>
<code>\uXXXX</code>	кодировка Юникода	<code>\u03a3 = Σ</code>
<code>\xXX</code>	символ из кодировки Latin-1	<code>\xA9 = ©</code>

Пример. Внедрение на страницу.

```
"Две строки\nна экране"
"Символ сигма: \u03a3."
```

Особенности работы со строками:

1. Строки являются *неизменяемыми*

```
var lang = "Java";
lang = lang + "Script";
```

2. *Доступ* к символам.

```
console.log('весна'.charAt(1)); // вернёт "е"
console.log('весна'[1]); // вернёт "е", с ES5
```

!!! Числовое свойство – *незаписываемыми и ненастраиваемыми.*

```
s='весна'; // "весна"
s[1]; // "е"
s[1]="2"; // "2"
s; // "весна"
```

3. *Сравнение* строк

на C – strcmp().

в JavaScript – операторы меньше и больше:

```
var a = 'a';
var b = 'b';
if (a < b) { // true
    print(a + ' меньше чем ' + b);
} else if (a > b) {
```

```

print(a + ' больше чем ' + b);
} else {
print(a + ' и ' + b + ' равны.');
```

ИЛИ

Метод String.localeCompare():

```

"1".localeCompare("2"); // -1
"1".localeCompare("1"); // 0
"1".localeCompare("0"); // 1
```

4. *Методы* работы со строками:

charAt()	normalize()	toLocaleLowerCase()
charCodeAt()	quote()	toLocaleUpperCase()
codePointAt()	repeat()	toLowerCase()
concat()	replace()	toSource()
includes()	search()	toString()
endsWith()	slice()	toUpperCase()
indexOf()	split()	trim()
lastIndexOf()	startsWith()	trimLeft()
localeCompare()	substr()	trimRight()
match()	substring()	valueOf()

Неизменяемые простые значения и ссылки на изменяемые объекты.

Простые значения:

1. являются *неизменяемыми*

```

s='весна' // "весна"
s.len=7 // 7: работа с временным объектом-обёрткой,
// кроме null и undefined
var t=s.len // undefined
```

```

s.toUpperCase() // "ВЕСНА"
s // "весна"
```

НО

```

"лето"===new String("лето") // true
"море"===new String("море") // false
```

2. *сравниваются* по значению

Объектные значения:

1. являются *изменяемыми*

```

var o = { x:1 }; // Начальное значение
o.x = 2; // изменили значение свойства
```

```

o.y = 3;           // добавили новое свойство
var a = [1,2,3]    // Массивы - изменяемыми объектами
a[0] = 0;         // изменил элемента массива
a[3] = 4;         // добавили элемента массива

```

!!!

```

var s=new String('весна');
s.len=7          // 7
var t=s.len      //
t                // 7, но не undefined

```

2. не сравниваются по значению

```

var o = {x:1},
    p = {x:1}; // Два объекта с одинаковыми свойствами
o === p       // => false
var a = [],
    b = [];   // Два различных пустых массива
a === b      // =>false

```

Пример. Создать новую копию объекта или массива

```

var a = ['a', 'b', 'c']; // Копируемый массив
var b = [];              // Массив, куда выполняется копирование
for(var i = 0; i < a.length; i++) {
    b[i] = a[i];         // Скопировать элемент a[] в b[]
}

```

Пример. Сравнение массивов.

```

function equalArrays(a,b) {
    if (a.length != b.length) return false;
    for(var i = 0; i < a.length; i++)
        if (a[i] !== b[i]) return false;
    return true;
}

```