

ОСНОВЫ JS

Лексическая структура

**Простые типы данных и операции над
НИМИ**

Лексическая структура

- Кодировка UTF-16
- Case-sensitive
- Комментарии
- Идентификаторы
- Ключевые слова
- Точка с запятой

```
/* Это  
комментарий */  
Abc = 5;  
abc = '5';  
ABC = { x: 1, y: 2 };  
// Abc, abc и ABC - разные переменные!  
  
i = "I'm a variable too!";
```

Типы данных

Простые

- Number
- String
- Boolean
- Symbol
- null
- undefined

Составные

- Object
- Array
- Function
- RegExp
- Date
- Error
- Set/Weak Set
- Map/Weak Map

Числа (Number)

- 8 байт, число с плавающей точкой ($-2^{53} .. 2^{53}$)
- Арифметические операции: +, -, /, *, %, **, ++, --
- Битовые операции: &, |, ^, ~, <<, >>
- Операции сравнения: ===, <, >, <=, >=, !==
- Объект Math

```
15
0xFFED
-45e13
+56.14
5 + 15;           // 20
.3 - .2 === .1;  // false
1 === 1.0;        // true
Math.abs(.3 - .2 - .1) < Number.EPSILON; // true
Math.random();    // 0.8042966571754915
```

Специальные значения (NaN, +-Infinity, -0)

```
5 / 0;           // Infinity
-5 / 0;          // -Infinity
Infinity / 1000; // Infinity
5 / -0;          // -Infinity
Infinity * 0;     // NaN
0 / 0;           // NaN
NaN === NaN;     // false
Number.isNaN(NaN); // true
typeof NaN;      // "number"
```

Класс `Number` - статические свойства

Имя	Значение
EPSILON	2^{-52}
MAX_VALUE	1.79E+308
MIN_VALUE	5E-324
MAX_SAFE_INTEGER	$2^{53} - 1$
MIN_SAFE_INTEGER	$-(2^{53} - 1)$
POSITIVE_INFINITY	Infinity
NEGATIVE_INFINITY	-Infinity
NaN	NaN

Класс Number - методы

- Number
 - isFinite()
 - isInteger()
 - isNaN()
 - parseInt()
 - parseFloat()
- Number.prototype
 - toFixed()
 - toPrecision()
 - toExponential()

```
Number.isFinite(12);           // true
Number.isFinite(Infinity);     // false

Number.isInteger(1);           // true
Number.isInteger(1.0);         // true
Number.isInteger(1.1);         // false

Number.isNaN(NaN);             // true
Number.isNaN('abc');           // false
Number.isNaN(15);              // false

Number.parseInt(' F', 16);      // 15
Number.parseInt('15e2');         // 15
Number.parseInt('H', 8);         // NaN
Number.parseFloat('15e2');       // 1500
Number.parseFloat('15eg');       // 15

(12345.6789).toFixed(2);        // '12345.68'
(-12).toFixed(3);               // '-12.000'
(NaN).toFixed(2);               // 'NaN'

(77.123).toExponential(3);      // '7.712e+1'
(5.156).toPrecision(2);         // '5.2'
(0.000015).toPrecision(1);      // '0.00002'
(0.000015).toPrecision(3);      // '0.0000150'
```


Строки (String)

- Упорядоченная последовательность 16-битных значений
- UTF-16, нет последовательной поддержки multibyte chars (code points)
- Нет отдельного типа для символов: это строки с длиной 1
- Операторы: + (конкатенация), [] (доступ по индексу)

```
'I\'m a string!'      // I'm a string!  
"I\"m a string too!"  // I"m a string too!
```

```
'abcd'.length;        // 4  
'abcd'[2];           // 'c'  
'abcd'.charCodeAt(2); // 99
```

Литералы строк

- single quotes / double quotes / backticks (template strings)

```
'my name is "Patches O'hoolihan"'
"or with double quotes - \"Patches O'hoolihan\" and escape sequences: \n

'And you can use utf codes: \uD834\uDF06' // "And you can use utf codes
"Or use code points: \u{1D306}"           // "Or use code points: ≡"

`Interpolation is best: ${20 / 4}`        // "Interpolation is best: 5"

"Multiline " +
"string" // "Multiline string"
"Multiline \
string" // "Multiline string"
"Multiline \n\
string" // "Multiline
// string"
`Multiline
string` // "Multiline
// string"
```

Объект `String` - методы и свойства

- `String`
 - `length`
 - `fromCharCode()`
 - `fromCodePoint()`
- `String.prototype`
 - `charAt()`
 - `charCodeAt()`
 - `codePointAt()`
 - `includes()`
 - `replace()`
 - `trim()`
 - `substr()`
 - `search()`
 - `match()`

```
String.fromCharCode(65, 66, 67); // "ABC"
String.fromCodePoint(0x1D306, 0x61, 0x1D307); // "≡a≡"
"吉a".length; // 3
"吉a"[0]; // '?'
"吉a"[1]; // '?'
"吉a".charAt(2); // 'a'
"吉a".charCodeAt(0); // 55362
"吉a".charCodeAt(1); // 57271
"吉a".charCodeAt(2); // 97
"吉a".codePointAt(0); // 134071
"吉a".codePointAt(1); // 57271
"吉a".codePointAt(2); // 97

'To be or not to be'.includes('To be'); // true
'To be or not to be'.includes('TO BE'); // false

'To be or not to be'.replace('not ', ''); // 'To be or to be'
'To be or not to be'.replace(/be/g, 'do'); // 'To do or not to do'

' Hello \t'.trim(); // 'Hello'

'Hello'.substr(3); // 'lo'
'Hello'.substr(3, 1); // 'l'
```

Логические значения (Boolean)

- Два значения: `true` и `false`
- "falsy" значения: `false`, `0`, `-0`, `null`, `undefined`, `NaN`, `""`
- Операции: `&&`, `||`, `!`

`null` и `undefined`

- Отдельные типы данных, с единственным значением в каждом
- `null` означает отсутствие значения
- `undefined` означает что значение не определено/не инициализировано

```
let drWatsonName = {  
  first: "John",  
  last: "Watson",  
  middle: null // спойлер: после третьего сезона - "Hamish!"  
};
```

```
let undef;  
undef; // undefined
```

Объектные обертки над простыми типами данных

- Все простые типы кроме `null`, `undefined` имеют специальные объектные обертки
- `Number`, `String`, `Boolean`, `Symbol` - это функции-конструкторы
- Объект-обертка создается неявно при использовании простого типа в объектном контексте
- Никогда не нужно создавать обертки явно

```
let s = "test";
s.myProp = 4;           // (new String(s)).myProp = 4;
s.myProp;               // (new String(s)).myProp; => undefined
```

```
new Number(5) + new Number(5); // 10, примитивное значение
```

```
let result = new Boolean(false);
if (result) {
    // эта ветка выполнится (!)
}
```

Неизменяемые простые значения и ссылки на изменяемые объекты

- Простые типы данных неизменяемы и сравниваются по значению
- Объекты изменяемы и сравниваются по ссылке

```
let s = "hello";  
s.toUpperCase(); // HELLO  
s;              // "hello"
```

```
let o = {x: 1};  
o.x = 2, o.y = 3;  
o.x; // 2, изменения сохранились
```

```
let a = [1, 2, 3];  
let b = a; // b указывает на тот же массив что и a
```

```
b[0] = 0, b[3] = 4;  
a[0]; // 0  
a[3]; // 4
```

```
let c = [0, 2, 3, 4];  
a === c; // false, разные ссылки на массивы
```


Приведение типов

- Неявное приведение типов - Очень Плохая Вещь
- Явное приведение с помощью Number, String, Boolean, Object
- "Полуявное" приведение типов с помощью неявного

```
Number(true);    // 1
Number('15hj');  // NaN, строже чем parseInt
Number([]);      // 0
```

```
String(false);   // 'false'
String(15e4);     // '150000'
String({});       // '[object Object]'
String([1, 2]);   // '1,2'
```

```
Boolean(NaN);    // false
Boolean({});     // true
```

```
Object(5);        // new Number(5)
Object(null);     // {}
```

```
// "Полуявное" приведение типов
+ x;              // то же что Number(x)
x + '';           // то же что String(x)
!!x;              // то же что Boolean(x)
```

Глобальный объект

- Разные имена в зависимости от среды (window, global, etc)
- Поля этого объекта - глобальные переменные
- Содержит:
 - глобальные свойства (NaN, undefined, Infinity)
 - глобальные функции (isNaN(), parseInt(), etc.)
 - конструкторы стандартных классов (Number, Object, String, etc.)
 - глобальные объекты (Math, JSON)
 - объекты, специфичные для среды (document, console, setTimeout, etc.)