ПОГРУЖЕНИЕ В ПРОСТЫЕ ТИПЫ

ЧИСЛО (NUMBER)

Числовые литералы

```
10
1.2
.2
1e10
0xe9a2
0123    // восьмеричное число (octal number)
```

Конструктор Number

```
Number.MIN_SAFE_INTEGER // -(2^53 - 1) || -9007199254740991
Number.MAX_SAFE_INTEGER //   2^53 - 1  ||  9007199254740991
```

```
Number.MIN_VALUE // 5e-324
Number.MAX_VALUE // 1.7976931348623157e308
```

```
Number.NEGATIVE_INFINITY // -Infinity
Number.POSITIVE_INFINITY //  Infinity
```

```
Number.NaN
```

```
Number.isNaN(NaN);             // true
Number.isFinite(Infinity);     // false
Number.isInteger(.9);          // false
Number.parseFloat("1.03");     // 1.03
Number.parseInt("1.03");       // 1
+"1.03"                        // 1.03
```

Методы экземпляров Number

```
var n = 200;
n.toFixed(2);         // -> "200.00"
2 .toFixed(3);        // -> "2.000"
n.toExponential(3); // -> "2.000e2"
n.toPrecision(5);    // -> "200.00"
n.toPrecision(22);  // Uncaught RangeError:
                      // toPrecision() argument must be between 1 and 2
n.toString();         // "200"
```

АРИФМЕТИЧЕСКИЕ ОПЕРАТОРЫ

```
let i = 1;
--i; ++i;
i--; i++;
i + 1;
i - 1;
i * 2;
i / 2;
i % 2;
```

ПРИСВАИВАНИЕ С ОПЕРАЦИЕЙ

```
let i = 10;
i += 1;
i -= 1;
i *= 1;
i /= 2;
i %= 2;
```

ОПЕРАТОРЫ ОТНОШЕНИЯ

```
let i = 10
i < 5          // false
i > 3          // true
i <= 10        // true
i >= 20        // false
i === 10       // true
i !== 20       // true
NaN === NaN    // false
```

```
// сравнение с приведением типов
i == 10     // true
i == "10"   // true
0 != ""     // false
// сравнение без приведения типов
0 !== ""    // true
```

ОБЪЕКТ MATH

```
// Свойства
Math.PI               // approx. 3.14159

// Методы
Math.cbrt(27)         // 3
Math.cos(Math.PI)     // -1
Math.floor(5.946)     // 5
Math.pow(2, 3)        // 8
Math.random()         // ??
                      // 0 <= result && result <= 1
Math.round(1.5)       // 2
Math.max(1, 2, 5)     // 5
Math.min(1, 2, 5)     // 1
Math.sign(-5)         // -1
Math.sqrt(9)          // 3
```

СТРОКА (STRING)

Строковые литералы

```
'Это строка';
"И это тоже строка";
`И "даже" 'это' строка ${typeof ''}`;
```

Экранирование и управляющие последовательности

```
"Это строка.\
Ее можно написать \n
на нескольких строках.";

'Отступ \t(или табуляция)';

"А вот \"кавычки\" и обратный слеш\\";
```

```
`Но можно испльзовать
"teplate strings"
    и получать тот же результат,
но писать меньше символов)`;

`<div>
   <a href="/">Хотя обратный слеш \\
     все равно надо экранировать((
   </a>
</div>`;
```

Свойство length

```
"a".length              // -> 1 (16 bits || 2 bytes)
'Привет'.length         // -> 6
`ab`.length             // -> 2
'吉'.length             // -> 2 (32 bits || 4 bytes)
```

```
"Hello"[1]              // "e"
```

```
"A" > "a"               // -> true
"Z" < "a"               // -> false
"b" < "c"               // -> true
"ab" === "ab"           // -> true
```

https://ru.wikipedia.org/wiki/Юникод

Конкатенация (сложение) строк

```
"Some string" + " another string";  // "Some string another string"
"Another" + ` string ${1 + 1}`      // "Another string 2"
```

```
"" + 200                            // "200"
"" + true                           // "true"
"" + null                           // "null"
"" + undefined                      // "undefined"
```

```
"" + function() {}           // "function() {}"
"" + [1]                     // "1"
"" + [1, [2]]                // "1,2"
"" + {key: 1}                // "[object Object]"
{key: 1} + ""                // 0
{} + {}                      // "[object Object][object Object]"
[] + {}                      // "[object Object]"
{} + []                      // 0
[] + []                      // ""

var sym = Symbol();
"" + sym                     // Uncaught TypeError:
                             // Cannot convert a Symbol value to a str
```

Методы

```
let someString = `Это строка-шаблон`,
    anotherString = " Some long, long, long string ";

someString.charAt(4);              // "с"
someString.indexOf("Э");           // 0
someString.indexOf("Ч");           // -1
someString.lastIndexOf("а");       // 12
someString.toUpperCase();          // "ЭТО СТРОКА-ШАБЛОН"
someString.toLowerCase();          // "это строка-шаблон"
anotherString.trim();              // "Some long, long, long string"
anotherString.trimRight();         // " Some long, long, long string"
anotherString.trimLeft();          // "Some long, long, long string "
```

```
anotherString.includes("long");         // true
anotherString.endsWith("String");       // false
anotherString.startsWith("Some");       // true
```

```
someString.slice(11);                   // "шаблон"
someString.slice(4, 10);                // "строка"
someString.slice(4, -7);                // "строка"
someString.substr(4, 6);                // "строка"
someString.substr(-13);                 // "строка-шаблон"
anotherString.substring(6, 17);         // "long, long,"  (positive onl
```

```
"da-".repeat(5);                        // "da-da-da-da-da-"
anotherString.substring(6, 12).repeat(10);
anotherString.split(",");               // [
                                        //   " Some long",
                                        //   " long",
                                        //   " long string "
                                        // ]

someString.split(" ");
anotherString.split("");
```

ЛОГИЧЕСКИЙ ТИП (BOOLEAN)

```
true
false
```

```
"a" === "a"                  // true
5 > 10                       // false
```

преобразование к логическому значению

```
// Коструктор Boolean
Boolean(1)                          // true
Boolean("false")                    // true
```

```
// всегда приводятся к ложному значению
Boolean(0)
Boolean("")
Boolean(undefined)
Boolean(null)
Boolean(NaN)
```

```
// приведение логического типа к числовому
+true                          // 1
+false                         // 0
```

```
let a = "ok",
    b = 0;
if(a) console.log("ok");
if(b) console.log(0);
```

Логические операторы

```
// Логическое И
true && true
false && true
true && false

// Логическое ИЛИ
true || false
false || true
true || "ok"

// Логическое отрицание
!true                   // false
!false                  // true
!""                     // true
!"ok"                   // false
```

```
let a = "ok",
    b = 0,
    c = a || "not ok",
    d = b || "default",
    e, f;

c && (e = 5);
f = e && 10;

!!true                  // true
!!false                 // false
!!0                     // false
!!1                     // true
```

УСЛОВНЫЙ (ТЕРНАРНЫЙ) ОПЕРАТОР

```
// выражение ? выражение : выражение
let a = 5,
    b = 10,
    c;

c = a < b ? 10 : 0;
c ? console.log("c is 10") : console.log("c is 0");
```

```
c < a ?
  "c is less than 5" :
  (c + 1) < 10 ?
    "c is less than 10" :
    "c is 10 or greater"
```

NULL И UNDEFINED

```
var a,
    b = {},
    c = [0, 1],
    d = function() { 1 + 1; },
    e = function(arg) {
          return "this is " + arg;
        };

a;                          // undefined
b.prop;                     // undefined
c[2];                       // undefined
d();                        // undefined
e("argument");              // "this is argument"
e();                        // "this is undefined"
```

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

let a = 0,
    b = null;

a != null ?
  "любое значение, кроме null и undefined" :
  "значение переменной null или undefined";
```

ПРЕОБРАЗОВАНИЕ ТИПОВ

```
5 + "5"
2 * "10"
"10" - 2
"10" - "two"                    // NaN

"5" == 5
"0" == false
"1" == true
"2" == true
null == false
null == true
undefined == false
undefined == true
undefined == null
```

```
Number("1")
String(2)

2 .toString()
10 .toString(2)                 // "1010"
10 .toString(16)                // "a"
10 .toString(10)                // "10"
parseInt("15.655 deg", 10)      // 1
parseFloat("3.35 uah", 10)      // 3.35

Boolean("false")

+"1"
"" + 2
!!"false"
```

Объекты-обертки

```
let num = 5,
    str = "Hello",
    isNull = null,
    undef = undefined;

num.toString();
str.length;

num.something               // undefined
str.hello                   // undefined

isNull.val                  // Uncaught TypeError:
                            // Cannot read property 'val' of null
undef.val                   // Uncaught TypeError:
                            // Cannot read property 'val' of undefined
```

УСЛОВНЫЕ ИНСТРУКЦИИ

инструкция if

```
if(выражение) инструкция;
if(выражение) {
  инструкция;
  инструкция;
}

let t = true, f = false;

if(t) console.log("see! it's truthy");
if(f) console.log("don't see... it's falsy");

if(a !== b) {
  console.log("a and b are not equal");
  console.log("we can do something else");
}
```

Ключевое слово else

```
if(выражение)
  инструкция;          // выражение истинное
else
  инструкция;          // выражение ложное

let age = 17;

if(age > 18)
  console.log("Person is adult");
else
  console.log("Person is a child");
```

```
let num = 7;

if(num > 7) {
  console.log("'num' is greater than 7");
} else if(num < 7) {
  console.log("'num' is less than 7");
} else {
  console.log("'num' is 7");
}
```

Инструкция switch

```
switch(выражение) {
  case выражение: инструкции
    break;
  case выражение: инструкции
    break;
  default: инструкции
}
```

```
let lang = "javascript",
    est;

switch(lang) {
  case "C": est = 1972;
    break;
  case "python": est = 1991;
    break;
  case "javascript": est = 1995;
    break;
  default: est = 2000;
}
```

ЦИКЛЫ (LOOPS)

Цикл for

```
for(выражение; выражение; выражение) инструкция
for(инициализация; условие; инкремент) тело цикла

for(let i = 0, t = 10; i < 10; i++) {
  console.log(i);
}

let str = "Some string";
for(let i = 0, l = str.length; i < l; ++i) {
  console.log(i);
}

for(let i = 10; i--;) console.log(i);
```

Инструкции break, continue и label

```
for(let i = 5; i--;) {
  if(i === 3) break;
  console.log(i);
}

for(let i = 0; i < 5; i++) {
  if(i === 2) continue;
  console.log(i);
}

topLoop:
for(let i = 10; i--;) {
  for(let j = 5; j--;) if(i === j) break topLoop;
  console.log(i);
}
```

Цикл while

```
while(выражение) инструкция;

let i = 0;
while(i < 5) {
  console.log(i);
  i++;
}

let d = 10;
while(d--) console.log(d);
```

Цикл do while

```
do инструкция while(выражение)

let i = 20, j = 0;

do {
  console.log(i);
  i--;
} while(i)

do console.log(j--); while(j > 0)
```

ПОЛЕЗНЫЕ ССЫЛКИ

- Случайное число в заданных пределах
- Таблица поддержки новых стандартов ES
- Лаборатория Хакима
- You Don't Know JS: Up & Going
- You Don't Know JS: Types & Grammar
- Перевод книги "Выразительный Javascript" (Eloquent Javascript)