Variables and data types (var / let / const)

**Note:**for a quick JS code execution you may use [JSBin](https://jsbin.com/?js,console,output" \o "https://jsbin.com/?js,console,output" \t "_blank)

**Variables:**

var one = 1;

let second = 2;

**Exercise:** try to invoke the variables before and after the declaration using [console.log(one)](https://developer.mozilla.org/ru/docs/Web/API/Console/log) command;

**References:**

* [Hoisting](https://developer.mozilla.org/en-US/docs/Glossary/Hoisting) (declarations processed before any code was executed)
* [Data types](https://learn.javascript.ru/types)

const pi = 3.14

pi = 4; // throws an Error

**Data types:**

**Primitives**

var number = 4;

var string = 'Hello!';

var sym = Symbol();

var boolean = true;

null;

undefined;

**Object**

var obj = { key: 'value' };

var arr = ['a', 'b', 'c'];

[**String**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String)

* Text values
* Literals representation: 'Text', "Text", `Text`
* Convert to string: *""+23, String(23), toString()*

const first = 'Hello';

const second = "World";

let result = first + " " + second; // result??

result = `${first} ${second}` // String literals. the same result is expected

**See more at the String and methods section**

const first = 123;

const second = 0b11;

[**Number**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number)

* Numerical values
* Integer and float literals: 1, -1.2, 1e2, e2
* Convert to Number: *+"123", Number("123"), parseInt(), parseFloat()*

[**Undefined**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/undefined)

* variables without any assigned value
* type and value at the same type

const first;

const second = undefined;

[**Null**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Null)

* pointer to a non existing or invalid object
* type and value at the same type

const first = null;

[**Boolean**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/boolean)

* logical entity
* true / false
* falsy values in JS: *false, null, undefined, 0, NaN, '', ""*
* Not falsy:*{}, [];*
* Convert to boolean:*!!value; Boolean(value)*

const first = true;

const second = false;

[**Object**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/object)

* key-value pair
* Object literals: *{}, {name: "Pavel"}*
* Special object*: Date;*wrappers*: Number, Map, JSON, Array*

const obj = {name: "Pavel"};

const name = obj.name; // "Pavel"

Operators

**References:** <https://www.w3schools.com/jsref/jsref_operators.asp>

**JavaScript Arithmetic Operators**

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y = 5**, the table below explains the arithmetic operators:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Result in y** | **Result in x** | **Try it** |
| + | Addition | x = y + 2 | y = 5 | x = 7 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_add) |
| - | Subtraction | x = y - 2 | y = 5 | x = 3 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_sub) |
| \* | Multiplication | x = y \* 2 | y = 5 | x = 10 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_mult) |
| / | Division | x = y / 2 | y = 5 | x = 2.5 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_div) |
| % | Modulus (division remainder) | x = y % 2 | y = 5 | x = 1 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_mod) |
| ++ | Increment | x = ++y | y = 6 | x = 6 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_incr) |
| x = y++ | y = 6 | x = 5 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_incr2) |
| -- | Decrement | x = --y | y = 4 | x = 4 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_decr) |
| x = y-- | y = 4 | x = 5 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_decr2) |

**JavaScript Assignment Operators**

Assignment operators are used to assign values to JavaScript variables.

Given that **x = 10** and **y = 5**, the table below explains the assignment operators:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Example** | **Same As** | **Result in x** | **Try it** |
| = | x = y | x = y | x = 5 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_equal) |
| += | x += y | x = x + y | x = 15 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_plusequal) |
| -= | x -= y | x = x - y | x = 5 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_minequal) |
| \*= | x \*= y | x = x \* y | x = 50 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_multequal) |
| /= | x /= y | x = x / y | x = 2 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_divequal) |
| %= | x %= y | x = x % y | x = 0 | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_modequal) |

**JavaScript String Operators**

The + operator, and the += operator can also be used to concatenate (add) strings.

Given that **text1 = "Good "**, **text2 = "Morning"**, **and text3 = ""**, the table below explains the operators:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Example** | **text1** | **text2** | **text3** | **Try it** |
| + | text3 = text1 + text2 | "Good " | "Morning" | "Good Morning" | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_string1) |
| += | text1 += text2 | "Good Morning" | "Morning" | "" | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_string2) |

**Comparison Operators**

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that **x = 5**, the table below explains the comparison operators:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Comparing** |
| == | equal to | x == 8 |
| x == 5 |
| === | equal value and equal type | x === "5" |
| x === 5 |
| != | not equal | x != 8 |
| !== | not equal value or not equal type | x !== "5" |
| x !== 5 |
| > | greater than | x > 8 |
| < | less than | x < 8 |
| >= | greater than or equal to | x >= 8 |
| <= | less than or equal to | x <= 8 |

**Conditional (Ternary) Operator**

The conditional operator assigns a value to a variable based on a condition.

|  |  |
| --- | --- |
| **Syntax** | **Example** |
| *variablename*= (*condition*) ?*value1*:*value2* | voteable = (age < 18) ? "Too young":"Old enough"; |

**Logical Operators**

Logical operators are used to determine the logic between variables or values.

Given that **x = 6 and y = 3**, the table below explains the logical operators:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| && | and | (x < 10 && y > 1) is true | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_and) |
| || | or | (x === 5 || y === 5) is false | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_or) |
| ! | not | !(x === y) is true | [Try it »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_not) |

**JavaScript Bitwise Operators**

Bit operators work on 32 bits numbers. Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Same as** | **Result** | **Decimal** |
| & | AND | x = 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | x = 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | x = ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | x = 5 ^ 1 | 0101 ^ 0001 | 0100 | 4 |
| << | Left shift | x = 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | Right shift | x = 5 >> 1 | 0101 >> 1 | 0010 | 2 |

**The typeof Operator**

The **typeof** operator returns the type of a variable, object, function or expression:

Example

typeof "John"                 // Returns string

typeof 3.14                   // Returns number

typeof NaN                    // Returns number

typeof false                  // Returns boolean

typeof [1, 2, 3, 4]           // Returns object

typeof {name:'John', age:34}  // Returns object

typeof new Date()             // Returns object

typeof function () {}         // Returns function

typeof myCar                  // Returns undefined (if myCar is not declared)

typeof null                   // Returns object

[Try it Yourself »](https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_oper_typeof)

Conditions

[**If Statement**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/if...else)

**Example 1**

if (2 \* 4 == 8) {

    console.log('That is right!');

} else {

    console.log('Wrong!');

}

**Example 2**

let num = 50;

if (num < 49) {

  console.log('Wrong!');

} else if (num > 100) {

    console.log('Too much!');

} else {

    console.log('That is right!');

}

[**Ternary operator**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/%D0%A3%D1%81%D0%BB%D0%BE%D0%B2%D0%BD%D1%8B%D0%B9_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80)

**Example 3**

let num = 50;

(num == 50) ? console.log('That is right!') : console.log('Wrong!');

[**Switch / case**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/switch)

**Example 4**

let num = 50;

switch (num) {

  case num < 49:

    console.log('Wrong!');

    break;

  case num > 100:

    console.log('Too much!');

    break;

  case 50:

    console.log('That is right!');

    break;

  default:

    console.log('Default behaviour');

    break;

}

Cycles

[**While**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/while)

**Example 1**

let num = 50;

while (num < 55) {

  console.log(num);

  num++;

}

[**Do...while**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/do...while)

**Example 2**

do {

    console.log(num);

    num++;

  }

while (num < 45);

[**For loop**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/for)

**Example 3**

for (let i = 0; i < 8; i++) {

    console.log(i);

}

**Example 4 (breaking a loop)**

for (let i = 0; i < 8; i++) {

    if (i == 6) {

        break;

    }

    console.log(i);

}

**Example 5 (continue loop execution)**

for (let i = 0; i < 8; i++) {

    if (i == 6) {

        continue;

    }

    console.log(i);

}