JavaScript Booleans

[❮ Previous](https://www.w3schools.com/js/js_random.asp)[Next ❯](https://www.w3schools.com/js/js_comparisons.asp)

A JavaScript Boolean represents one of two values: **true** or **false**.

Boolean Values

Very often, in programming, you will need a data type that can only have one of two values, like

* YES / NO
* ON / OFF
* TRUE / FALSE

For this, JavaScript has a **Boolean** data type. It can only take the values **true** or **false**.

The Boolean() Function

You can use the Boolean() function to find out if an expression (or a variable) is true:

Example

Boolean(10 > 9)

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_boolean_expression1)

Or even easier:

Example

(10 > 9)  
10 > 9

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_boolean_expression2)

Comparisons and Conditions

The chapter JS Comparisons gives a full overview of comparison operators.

The chapter JS Conditions gives a full overview of conditional statements.

Here are some examples:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | equal to | if (day == "Monday") |
| > | greater than | if (salary > 9000) |
| < | less than | if (age < 18) |

The Boolean value of an expression is the basis for all JavaScript comparisons and conditions.

ADVERTISEMENT

Everything With a "Value" is True

Examples

100  
  
3.14  
  
-15  
  
"Hello"  
  
"false"  
  
7 + 1 + 3.14

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

Everything Without a "Value" is False

The Boolean value of **0** (zero) is **false**:

let x = 0;  
Boolean(x);

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

The Boolean value of **-0** (minus zero) is **false**:

let x = -0;  
Boolean(x);

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

The Boolean value of **""**(empty string) is **false**:

let x = "";  
Boolean(x);

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

The Boolean value of **undefined** is **false**:

let x;  
Boolean(x);

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

The Boolean value of **null** is **false**:

let x = null;  
Boolean(x);

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

The Boolean value of **false** is (you guessed it) **false**:

let x = false;  
Boolean(x);

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

The Boolean value of **NaN** is **false**:

let x = 10 / "Hallo";  
Boolean(x);

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

JavaScript Booleans as Objects

Normally JavaScript booleans are primitive values created from literals:

let x = false;

But booleans can also be defined as objects with the keyword new:

let y = new Boolean(false);

Example

let x = false;  
let y = new Boolean(false);  
  
// typeof x returns boolean  
// typeof y returns object

[Try it yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_booleans_object)

Do not create Boolean objects.

The new keyword complicates the code and slows down execution speed.

Boolean objects can produce unexpected results:

When using the == operator, x and y are **equal**:

let x = false;  
let y = new Boolean(false);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_booleans_object1)

When using the === operator, x and y are **not equal**:

let x = false;  
let y = new Boolean(false);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_booleans_object2)

Note the difference between (x==y) and (x===y).

(x == y) true of false?

let x = new Boolean(false);  
let y = new Boolean(false);

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_booleans_object3)

(x === y) true of false?

let x = new Boolean(false);  
let y = new Boolean(false);

# JavaScript Comparison and Logical Operators

[❮ Previous](https://www.w3schools.com/js/js_booleans.asp)[Next ❯](https://www.w3schools.com/js/js_if_else.asp)

Comparison and Logical operators are used to test for true or false.

## 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** | **Returns** | **Try it** |
| == | equal to | x == 8 | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison1) |
| x == 5 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison2) |
| x == "5" | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison12) |
| === | equal value and equal type | x === 5 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison4) |
| x === "5" | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison3) |
| != | not equal | x != 8 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison5) |
| !== | not equal value or not equal type | x !== 5 | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison7) |
| x !== "5" | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison6) |
| x !== 8 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison13) |
| > | greater than | x > 8 | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison8) |
| < | less than | x < 8 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison9) |
| >= | greater than or equal to | x >= 8 | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison10) |
| <= | less than or equal to | x <= 8 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison11) |

ADVERTISEMENT

## How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

if (age < 18) text = "Too young to buy alcohol";

You will learn more about the use of conditional statements in the next chapter of this tutorial.

## 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/js/tryit.asp?filename=tryjs_comparison_and) |
| || | or | (x == 5 || y == 5) is false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_or) |
| ! | not | !(x == y) is true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_not) |

## Conditional (Ternary) Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

### Syntax

variablename = (condition) ? value1:value2

### Example

let voteable = (age < 18) ? "Too young":"Old enough";

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

If the variable age is a value below 18, the value of the variable voteable will be "Too young", otherwise the value of voteable will be "Old enough".

## Comparing Different Types

Comparing data of different types may give unexpected results.

When comparing a string with a number, JavaScript will convert the string to a number when doing the comparison. An empty string converts to 0. A non-numeric string converts to NaN which is always false.

|  |  |  |
| --- | --- | --- |
| **Case** | **Value** | **Try** |
| 2 < 12 | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_20) |
| 2 < "12" | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_21) |
| 2 < "John" | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_23) |
| 2 > "John" | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_24) |
| 2 == "John" | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_25) |
| "2" < "12" | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_26) |
| "2" > "12" | true | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_27) |
| "2" == "12" | false | [Try it »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_comparison_28) |

When comparing two strings, "2" will be greater than "12", because (alphabetically) 1 is less than 2.

To secure a proper result, variables should be converted to the proper type before comparison:

age = Number(age);  
if (isNaN(age)) {  
  voteable = "Input is not a number";  
} else {  
  voteable = (age < 18) ? "Too young" : "Old enough";  
}