w3schools.com

THE WORLD'S LARGEST WEB DEVELOPER SITE

Learn How To Create CSS Alert Buttons

Success Info Warning Danger Default

JavaScript Number Methods

❮ Previous Next ❯

Number methods help you work with numbers.

Number Methods and Properties

Primitive values (like 3.14 or 2014), cannot have properties and methods (because they are not objects).

But with JavaScript, methods and properties are also available to primitive values, because JavaScript treats primitive values as objects when executing methods and properties.

The toString() Method

toString() returns a number as a string.

All number methods can be used on any type of numbers (literals, variables, or expressions):

Example

```
var x = 123;
x.toString();            // returns 123 from variable x
(123).toString();        // returns 123 from literal 123
(100 + 23).toString();   // returns 123 from expression 100 + 23
```

Try it Yourself »

The toExponential() Method

toExponential() returns a string, with a number rounded and written using exponential notation.

A parameter defines the number of characters behind the decimal point:

```
var x = 9.656;
x.toExponential(2);     // returns 9.66e+0
x.toExponential(4);     // returns 9.6560e+0
x.toExponential(6);     // returns 9.656000e+0
```

Try it yourself »

The parameter is optional. If you don't specify it, JavaScript will not round the number.

The toFixed() Method

toFixed() returns a string, with the number written with a specified number of decimals:

Example

```
var x = 9.656;
x.toFixed(0);           // returns 10
x.toFixed(2);           // returns 9.66
x.toFixed(4);           // returns 9.6560
x.toFixed(6);           // returns 9.656000
```

Try it yourself »

toFixed(2) is perfect for working with money.

The toPrecision() Method

toPrecision() returns a string, with a number written with a specified length:

```
var x = 9.656;
x.toPrecision();        // returns 9.656
x.toPrecision(2);       // returns 9.7
x.toPrecision(4);       // returns 9.656
x.toPrecision(6);       // returns 9.65600
```

Try it Yourself »

The valueOf() Method

valueOf() returns a number as a number.

Example

```
var x = 123;
x.valueOf();            // returns 123 from variable x
(123).valueOf();        // returns 123 from literal 123
(100 + 23).valueOf();   // returns 123 from expression 100 + 23
```

Try it Yourself »

In JavaScript, a number can be a primitive value (typeof = number) or an object (typeof = object).

The valueOf() method is used internally in JavaScript to convert Number objects to primitive values.

There is no reason to use it in your code.

All JavaScript data types have a valueOf() and a toString() method.

Converting Variables to Numbers

There are 3 JavaScript methods that can be used to convert variables to numbers:

- The Number() method
- The parseInt() method
- The parseFloat() method

These methods are not **number** methods, but **global** JavaScript methods.

Global Methods

JavaScript global methods can be used on all JavaScript data types.

These are the most relevant methods, when working with numbers:

Number()	Returns a number, converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns an integer

The Number() Method

Number() can be used to convert JavaScript variables to numbers:

Example

```
Number(true);          // returns 1
Number(false);         // returns 0
Number("10");          // returns 10
Number("  10");        // returns 10
Number("10  ");        // returns 10
Number("10 20");       // returns NaN
Number("John");        // returns NaN
```

Try it Yourself »

If the number cannot be converted, NaN (Not a Number) is returned.

The Number() Method Used on Dates

Number() can also convert a date to a number:

Example

```
Number(new Date("2017-09-30"));    // returns 1506729600000
```

Try it Yourself »

The Number() method above returns the number of milliseconds since 1.1.1970.

The parseInt() Method

parseInt() parses a string and returns a whole number. Spaces are allowed. Only the first number is returned:

```
parseInt("10");         // returns 10
parseInt("10.33");      // returns 10
parseInt("10 20 30");   // returns 10
parseInt("10 years");   // returns 10
parseInt("years 10");   // returns NaN
```

Try it yourself »

If the number cannot be converted, NaN (Not a Number) is returned.

The parseFloat() Method

parseFloat() parses a string and returns a number. Spaces are allowed. Only the first number is returned:

Example

```
parseFloat("10");        // returns 10
parseFloat("10.33");     // returns 10.33
parseFloat("10 20 30");  // returns 10
parseFloat("10 years");  // returns 10
parseFloat("years 10");  // returns NaN
```

Try it yourself »

If the number cannot be converted, NaN (Not a Number) is returned.

Number Properties

Property	**Description**
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value
POSITIVE_INFINITY	Represents infinity (returned on overflow)

Example

```
var x = Number.MAX_VALUE;
```

Number properties belongs to the JavaScript's number object wrapper called **Number**.

These properties can only be accessed as **Number**.MAX_VALUE.

Using *myNumber*.MAX_VALUE, where *myNumber* is a variable, expression, or value, will return undefined:

Example

```
var x = 6;
var y = x.MAX_VALUE;    // y becomes undefined
```

Try it yourself »

Complete JavaScript Number Reference

For a complete reference, go to our Complete JavaScript Number Reference.

The reference contains descriptions and examples of all Number properties and methods.

❮ Previous Next ❯