JavaScript Type Conversion

❮ Previous Next ❯

Number() converts to a Number, String() converts to a String, Boolean() converts to a Boolean.

JavaScript Data Types

In JavaScript there are 5 different data types that can contain values:

- string
- number
- boolean
- object
- function

There are 3 types of objects:

- Object
- Date
- Array

And 2 data types that cannot contain values:

- null
- undefined

The typeof Operator

You can use the **typeof** operator to find the data type of a JavaScript variable.

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" *
typeof null                   // Returns "object"
```

Try it Yourself »

Please observe:

- The data type of NaN is number
- The data type of an array is object
- The data type of a date is object
- The data type of null is object
- The data type of an undefined variable is **undefined** *
- The data type of a variable that has not been assigned a value is also **undefined** *

You cannot use **typeof** to determine if a JavaScript object is an array (or a date).

The Data Type of typeof

The typeof operator is not a variable. It is an operator. Operators (+ - * /) do not have any data type.

But, the typeof operator always **returns a string** (containing the type of the operand).

The constructor Property

The **constructor** property returns the constructor function for all JavaScript variables.

Example

```
"John".constructor                 // Returns function String()  {[native code]}
(3.14).constructor                 // Returns function Number()  {[native code]}
false.constructor                  // Returns function Boolean() {[native code]}
[1,2,3,4].constructor              // Returns function Array()   {[native code]}
{name:'John',age:34}.constructor   // Returns function Object()  {[native code]}
new Date().constructor             // Returns function Date()    {[native code]}
function () {}.constructor         // Returns function Function(){[native code]}
```

Try it Yourself »

You can check the constructor property to find out if an object is an Array (contains the word "Array"):

Example

```
function isArray(myArray) {
    return myArray.constructor.toString().indexOf("Array") > -1;
}
```

Try it Yourself »

Or even simpler, you can check if the object is an Array function:

Example

```
function isArray(myArray) {
    return myArray.constructor === Array;
}
```

Try it Yourself »

You can check the constructor property to find out if an object is a Date (contains the word "Date"):

Example

```
function isDate(myDate) {
    return myDate.constructor.toString().indexOf("Date") > -1;
}
```

Try it Yourself »

Or even simpler, you can check if the object is a Date function:

Example

```
function isDate(myDate) {
    return myDate.constructor === Date;
}
```

Try it Yourself »

JavaScript Type Conversion

JavaScript variables can be converted to a new variable and another data type:

- By the use of a JavaScript function
- **Automatically** by JavaScript itself

Converting Numbers to Strings

The global method **String()** can convert numbers to strings.

It can be used on any type of numbers, literals, variables, or expressions:

Example

```
String(x)         // returns a string from a number variable x
String(123)       // returns a string from a number literal 123
String(100 + 23)  // returns a string from a number from an expression
```

Try it Yourself »

The Number method **toString()** does the same.

Example

```
x.toString()
(123).toString()
(100 + 23).toString()
```

Try it Yourself »

In the chapter Number Methods, you will find more methods that can be used to convert numbers to strings:

Method	Description
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length

Converting Booleans to Strings

The global method **String()** can convert booleans to strings.

```
String(false)        // returns "false"
String(true)         // returns "true"
```

The Boolean method **toString()** does the same.

```
false.toString()     // returns "false"
true.toString()      // returns "true"
```

Converting Dates to Strings

The global method **String()** can convert dates to strings.

```
String(Date())      // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)"
```

The Date method **toString()** does the same.

Example

```
Date().toString()   // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)"
```

In the chapter Date Methods, you will find more methods that can be used to convert dates to strings:

Method	Description
getDate()	Get the day as a number (1-31)
getDay()	Get the weekday a number (0-6)
getFullYear()	Get the four digit year (yyyy)
getHours()	Get the hour (0-23)
getMilliseconds()	Get the milliseconds (0-999)
getMinutes()	Get the minutes (0-59)
getMonth()	Get the month (0-11)
getSeconds()	Get the seconds (0-59)
getTime()	Get the time (milliseconds since January 1, 1970)

Converting Strings to Numbers

The global method **Number()** can convert strings to numbers.

Strings containing numbers (like "3.14") convert to numbers (like 3.14).

Empty strings convert to 0.

Anything else converts to NaN (Not a number).

```
Number("3.14")    // returns 3.14
```

```
Number(" ")       // returns 0
Number("")        // returns 0
Number("99 88")   // returns NaN
```

In the chapter Number Methods, you will find more methods that can be used to convert strings to numbers:

Method	Description
parseFloat()	Parses a string and returns a floating point number
parseInt()	Parses a string and returns an integer

The Unary + Operator

The **unary + operator** can be used to convert a variable to a number:

Example

```
var y = "5";      // y is a string
var x = + y;      // x is a number
```

Try it Yourself »

If the variable cannot be converted, it will still become a number, but with the value NaN (Not a number):

Example

```
var y = "John";   // y is a string
var x = + y;      // x is a number (NaN)
```

Try it Yourself »

Converting Booleans to Numbers

The global method **Number()** can also convert booleans to numbers.

```
Number(false)     // returns 0
Number(true)      // returns 1
```

Converting Dates to Numbers

The global method **Number()** can be used to convert dates to numbers.

```
d = new Date();
Number(d)          // returns 1404568027739
```

The date method **getTime()** does the same.

```
d = new Date();
d.getTime()        // returns 1404568027739
```

Automatic Type Conversion

When JavaScript tries to operate on a "wrong" data type, it will try to convert the value to a "right" type.

The result is not always what you expect:

```
5 + null    // returns 5         because null is converted to 0
"5" + null  // returns "5null"   because null is converted to "null"
"5" + 2     // returns "52"      because 2 is converted to "2"
"5" - 2     // returns 3         because "5" is converted to 5
"5" * "2"   // returns 10        because "5" and "2" are converted to 5 and 2
```

Try it Yourself »

Automatic String Conversion

JavaScript automatically calls the variable's toString() function when you try to "output" an object or a variable:

```
document.getElementById("demo").innerHTML = myVar;

// if myVar = {name:"Fjohn"}  // toString converts to "[object Object]"
// if myVar = [1,2,3,4]       // toString converts to "1,2,3,4"
// if myVar = new Date()      // toString converts to "Fri Jul 18 2014 09:08:55 GMT+0200"
```

Numbers and booleans are also converted, but this is not very visible:

```
// if myVar = 123             // toString converts to "123"
// if myVar = true            // toString converts to "true"
// if myVar = false           // toString converts to "false"
```

JavaScript Type Conversion Table

This table shows the result of converting different JavaScript values to Number, String, and Boolean:

Original Value	Converted to Number	Converted to String	Converted to Boolean	Try it
false	0	"false"	false	Try it »
true	1	"true"	true	Try it »
0	0	"0"	false	Try it »
1	1	"1"	true	Try it »
"0"	0	"0"	**true**	Try it »
"000"	0	"000"	**true**	Try it »
"1"	1	"1"	true	Try it »
NaN	NaN	"NaN"	false	Try it »
Infinity	Infinity	"Infinity"	true	Try it »
-Infinity	-Infinity	"-Infinity"	true	Try it »
""	**0**	""	**false**	Try it »
"20"	20	"20"	true	Try it »
"twenty"	NaN	"twenty"	true	Try it »
[ ]	**0**	""	true	Try it »
[20]	**20**	"20"	true	Try it »
[10,20]	NaN	"10,20"	true	Try it »
["twenty"]	NaN	"twenty"	true	Try it »
["ten","twenty"]	NaN	"ten,twenty"	true	Try it »
function(){}	NaN	"function(){}"	true	Try it »
{ }	NaN	"[object Object]"	true	Try it »

null	**0**	"null"	false	Try it »
undefined	NaN	"undefined"	false	Try it »

Values in quotes indicate string values.

Red values indicate values (some) programmers might not expect.

❮ Previous Next ❯