

Working With the Boolean Type

In this lesson, we shall learn about the Boolean type and how to use it in JavaScript.
Let's begin!

WE'LL COVER THE FOLLOWING



- Some handy Boolean conversion rules
 - Some handy examples
- Quiz time! :)



Booleans in JavaScript



Boolean values are the basis of most flow-control statements because they represent either `true` or `false`; as you remember, these are Boolean literals and reserved words, too.

✓ **True**

■ **False**

You can declare Boolean variables with these literals, or expressions resulting in a Boolean value:

in a Boolean value.

```
var isValid = true;
var isTheGlobeFlat = false;
var flag = 3 < 4;
console.log(isValid);
console.log(isTheGlobeFlat);
console.log(flag);
```



With the help of the `Boolean()` casting function, you can convert any value to a Boolean:

```
var thisValue = Boolean("false");
console.log(thisValue);
```



You must be careful with Boolean conversion because it follows different rules than you may think.

For example, the `thisValue` variable will hold `true` after this conversion, instead of `false` suggested by the right side of the assignment statement.

To understand the above phenomenon, here are the conversion rules:

Some handy Boolean conversion rules


Boolean literals are converted to their appropriate Boolean values according to the following rules:

- ★ **Empty string** is converted to `false`
- ★ Any **nonempty strings** will be represented with `true`. This is why the `"false"` string results `true`.
- ★ Any **non-zero number** (including Infinity) results `true`

★ **0** and **NaN** (not-a-number) yield **false**.

★ The **null** object is converted to **false**, while any other objects represent **true**.

★ The **undefined** type always yields **false**.

 **NOTE:** Soon, you will learn about **NaN** and **Infinity**.

Some handy examples

Let's see a few examples:

```
var b1 = Boolean(undefined); // false
var b2 = Boolean(1/0);       // true
var b3 = Boolean(0);         // false
var b4 = Boolean(null);      // false
var b5 = Boolean(new Object()); // true

console.log(b1);
console.log(b2);
console.log(b3);
console.log(b4);
console.log(b5);
```



The **1/0** expression does not raise an error, it results **Infinity**.

Be aware that the **Boolean()** casting function is different from the **Boolean()** constructor.

In the following code snippet, **bool1** will be a Boolean value (**false**) while **bool2** is an object, wrapping a **true** Boolean value:

```
var bool1 = Boolean(false);
var bool2 = new Boolean(true);

console.log(bool1);
console.log(bool2);
```



Quiz time! :)

It's time to test how much we've learned in this lesson with a short quiz!

Check your Understanding

1

Given the following expression:

```
70 > 90
```

What value will be returned?

COMPLETED 0%

1 of 2



Now that we've become familiar with the Boolean type and its conversion rules, let's go on to meet the Number type in the *next lesson*.

See you there! :)