JavaScript TUTORIAL

JavaScript boolean type

Summary: in this tutorial, you'll learn about the JavaScript boolean type that has two literal values `true` and `false` .

Introduction to the JavaScript boolean type

The JavaScript `boolean` primitive type has two literal values: `true` and `false` .

The following example declares two variables and initializes their values to `true` and `false` :

```
let completed = true;
let running = false;
```

The boolean's literal values are case-sensitive. This means that the `True` and `False` are valid identifiers but they're not `boolean` values.

JavaScript allows the values of other types to be cast to boolean values. To cast a non-Boolean value to a boolean value, you use the built-in `Boolean()` function. For example:

```
let error = 'An error occurred';
let hasError = Boolean(error);

console.log(hasError);
```

Output:

```
true
```

How it works.

First, declare a variable `error` that holds a literal string `'An error occurred'` :

```
let error = 'An error occurred';
```

Second, cast the `error` variable to a boolean value using the `Boolean()` function:

```
let hasError = Boolean(error);
```

Third, output the value of the `hasError` variable to the console:

```
console.log(hasError);
```

Since the `error` variable holds a non-empty string, the `Boolean()` function casts its value to `true`.

The following table shows how the `Boolean()` function casts the values of other types to boolean values:

Data Type	Values converted to true	Value Converted to false
string	Any non-empty string	"" (empty string)
number	Any Non-zero number	`0`, `NaN`
object	Any object	null
undefined	(not relevant)	undefined

This table is important because some statements automatically cast a non-boolean value to a boolean value using the `Boolean()` function.

For example, the `if` statement executes a block if a condition is `true`. If you use a non-boolean value, it'll use the `Boolean()` function to implicitly cast that value to a boolean value.

> Note that you'll learn about the `if` statement in the `if` tutorial.

See the following example:

```
let error = 'An error occurred';

if (error) {
  console.log(error);
}
```

Output:

```
An error occurred
```

In this example, since the `error` variable holds a non-empty string, the `if` statement evaluates its value to `true`. Therefore, it executes the `console.log(error)` to output the `error` to the console.

If you change the value of the `error` variable to an empty string (`""`), you won't see anything in the output because the `if` statement evaluates it as `false`:

```
let error = '';
if (error) {
  console.log(error);
}
```

Quiz

Summary

- JavaScript `boolean` type has two literal values `true` and `false`.
- Use the `Boolean()` function to cast a non-boolean value to a boolean value.
- Some statements implicitly cast a non-boolean value into a boolean value.