

Lesson

Weekend

Introduction to Programming

(/introduction-to-programming)

/ JavaScript and Web Browsers

(/introduction-to-programming/javascript-and-web-browsers)

/ Assignment, Comparison, and Equality Operators

Text

Cheat sheet

JavaScript has many operators for many different situations. We've already learned about a two types of operators:

- Arithmetic operators: `+`, `-`, `*`, and `/`.
- The assignment operator: `=`

Let's now take the time to learn about more JavaScript operators used for assignment and comparison. For a list of all JavaScript operators, visit the MDN reference page on Expressions and Operators (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators>). We'll be linking to specific sections within the MDN "Expressions and Operators" page throughout this lesson.

Assignment Operators

We've already seen how we can use the `=` assignment operator to set a variable equal to a value.

```
> const favoriteNumber = 42;  
> favoriteNumber;  
42
```

The `=` is called an **assignment operator** because it assigns the value on the right of the operator to the variable on the left. In the above example, `favoriteNumber` is assigned the value 42.

Another assignment operator is the `+=` operator, because it also assigns a new value to the variable on the left based on the value to the right.

```
> let myNumber = 5;  
> myNumber += 1;  
> myNumber;  
6
```

There is an assignment operator for each of the mathematical functions: addition, subtraction, multiplication and division.

```
+=  
-=  
*=  
/=
```

When you use any of these assignment operators, the value of the variable on the left side is changed by the math operation and value on the right. Let's do one of each:

```
> let testNumber = 10;
> testNumber += 5;
> testNumber;
15
> testNumber -= 9;
> testNumber;
6
> testNumber *= 3;
> testNumber;
18
> testNumber /= 2
> testNumber;
9
```

Documentation on Assignment Operators

Visit this link for the MDN reference page on assignment operators:

-  **Assignment Operators** (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators#assignment_operators)

Comparison Operators

Assignment operators change the value of the variable on the left of the operator. **Comparison operators** do not change any values, but return the booleans `true` or `false` depending on whether the JavaScript expression evaluates as true or false.

```
> const myNumber = 5;
> myNumber < 10;
true
> myNumber > 10;
false
```

Comparison operators are also called **relational operators**, because they help find the relationship between two operands, asking questions like "is 10 bigger than 5?".

Greater Than Operator >

> means "greater-than (and not equal to)":

- `3 > 4` evaluates to `false`.
- `3 > 3` also evaluates to `false`, because 3 is equal to 3 and not greater.
- `3 > 2` evaluates to `true`.

Less Than Operator <

< is the opposite of >. It means "less-than (and not equal to)":

- `3 < 5` evaluates to `true`.
- `3 < 3` evaluates to `false` because they are equal.

Greater Than or Equal Operator >=

>= is the same as >, except it evaluates to `true` if the two sides are equal:

- `3 >= 3` evaluates to `true`
- `3 >= 2` also evaluates to `true`.

Less Than or Equal Operator <=

<= is the opposite of >=. It means "less-than-or-equal-to":


- `3 <= 3` evaluates to `true` because 3 is equal to 3.
- `3 <= 1` evaluates to `false`
- `3 <= 5` evaluates to `true`.

In the above examples, notice that the comparison operators return one of two values: `true` or `false`. Notice that there are no quotes around these values. `true` and `false` aren't strings — they're called

booleans. They simply represent being true or false.

Documentation on Comparison/Relational Operators

Visit this link for the MDN reference page on relational operators:

-  **Relational Operators** (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators#relational_operators)

Equality Operators

We can also compare the equality of two operands with **equality operators**. This means we can check to see whether or not two operands have the same value. Equality operators return a **boolean**; true or false.

```
> const myNumber = 5;
> myNumber === 10;
false
> myNumber === 5;
true
```

Notice the triple equal signs syntax: `===`. This is a type of **equality operator** called **strict equality**. We use 3 equal signs `===` when we are asking if two operands are equal to each other. When we're assigning a variable to a value, such as `const myNumber = 5` we use a single equal sign `=`. Mixing these up is one of the easiest syntax errors to make.

JavaScript also has an equal operator with 2 equal signs, `==`, but it is almost never used and you should generally avoid it. Try out this example in the DevTools Console:

```
> const myNumber = 5;
> myNumber === 5;
true
> myNumber === "5"
false
> myNumber == "5"
true
```

The double equals operator returns `true` when comparing `5 == "5"`, indicating that a number and a string are the same. Whereas the strict equality operator with 3 equal signs returns `false` when comparing `5 === "5"`, because a number and a string are not the same data type. With the double equals operator, JavaScript will make the assumption that you want the two different data types to be evaluated the same. As a developer you may not be expecting that assumption and this can lead to confusing bugs in your code.

Take some time to play around more with the double equals operator and the strict equality operator by visiting the MDN documentation:

- Strict Equality Operator (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Strict_equality)
- Double Equals Operator (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Equality>)

We can also check the opposite of equality — not being equal — with the **inequality operator** `!==`.

```
> const myNumber = 5;
> myNumber !== 10;
true
> myNumber !== 5;
false
```

Equality operators work for strings as well.

```
> const greeting = "hello world";  
> greeting === "hello world";  
true  
> greeting === "goodbye";  
false  
> greeting;  
"hello world"
```

Notice that if you type `greeting` after using the equality operators, you will see that the variable `greeting` still contains the string `"hello world"`. **Comparison and equality operators do not change the value of the variable.** Let's look at another example to illustrate that important difference between assignment operators and comparison/equality operators.

```
> let myNumber = 5;  
> myNumber === 5;  
true  
> myNumber === 10;  
false  
> myNumber = 10;  
> myNumber === 10;  
true  
> myNumber === 5;  
false
```

Notice that we use `let` instead of `const` here because we reassign the value of `myNumber` to 10. We wouldn't be able to do that if `myNumber` were a constant variable declared with `const`.

Here are some more examples of equality operators.

Equality Operator ===

=== means "equal-to".

- `5 === 5` or `"cat" === "cat"` evaluate to `true`

- `3 === 5` or `"cat" === "dog"` evaluate to `false`.

Inequality Operator `!==`

`!==` means "not-equal-to". It is the opposite of `===`.

- `"cat" !== "dog"` evaluates to `true`
- `5 !== 5` evaluates to `false`, because saying that 5 is not equal to 5 is not true.

Documentation on Equality Operators

Visit this link for the MDN reference page on equality operators:

- [🔗 Equality Operators \(https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators#equality_operators\)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators#equality_operators)

[Previous \(/introduction-to-programming/javascript-and-web-browsers/practice-using-mdn-documentation-javascript\)](#)

[Next \(/introduction-to-programming/javascript-and-web-browsers/practice-assignment-comparison-and-equality-operators\)](#)

Lesson 18 of 75

Last updated March 24, 2023

[disable dark mode](#)



© 2023 Epicodus (<http://www.epicodus.com/>), Inc.