Equality Operators

In this lesson, we'll see the significance and usage of the equality operators in JavaScript.
Let's begin!

WE'LL COVER THE FOLLOWING

Equality

Testing equality is an important operation in every programming language, just like in JavaScript.

Due to the loosely-typed nature of JavaScript, it defines **two kinds of equality**:

1. identically equal (`===`), and

2. equal (`==`)

Non-Equality

Implicitly, there are **two kinds of non-equality**:

1. not identically equal (`!==`), and
2. not equal (`!=`)

The equal and not equal operators use conversion to determine the equality of their operands.

Rules for conversions

During the conversion they follow these rules:

> 1. If an operand is a `Boolean` value, it is converted into a `Number` before checking for equality. A value of false is converted to 0, whereas a value of true is converted to 1.
> 2. If one operand is a string and the other is numeric, before testing, the string value is attempted to be converted into a `Number`.
> 3. If one of the operands is an object and the other is not, the `valueOf()` method is called on the object to retrieve a primitive value to compare according to the previous rules.

Rules for comparisons

When making the comparisons, operators follow these rules:

> 1. Values of null and undefined are equal.
> 2. Values of null and undefined cannot be converted into any other values for equality checking.
> 3. If either operand is NaN, the equal operator returns false and the not-equal operator returns true.
> 4. If both operands are objects, then they are compared to see if they

are the same object. If both operands point to the same object, then

the equal operator returns true; otherwise, false, because they are not equal.

Examples

Let’s look at a few examples:

index.js

```
console.log(null == undefined); // true
console.log(23 == "23");         // true
console.log(1 / 0 == 2 / 0);     // true
console.log(false == 0);         // true
console.log(true == 1);          // true
console.log(true == 2);          // false
console.log(NaN == NaN);         // false
```

NOTE: As it was already treated earlier, `NaN` is a special value that is not equal to itself.

The identically equal and not identically equal operators do not convert the operands before testing for equality. The two operands are identically equal if and only if their types and their values are equal.

These short examples demonstrate how they work:

index.js

```
console.log(23 == "23");        // true
console.log(23 === "23");       // false
console.log(23 === (12 + 11)); // true
console.log(23 != "23");        // false
console.log(23 !== "23");       // true
```

Quiz Time! :)

Question 1

Q

Given the following expression:

```
800 === "800"
```

What value will be returned?

Check Answers

Question 2

Q

Given the following expression:

```
800 == "800"
```

What value will be returned?

Check Answers

In the *next lesson*, we’ll see the capabilities of the relational and Boolean operators.

Stay tuned!