Theory: Comparison operators

7 minutes 0 / 5 problems solved

In JavaScript, it is possible to compare different types of data. The execution of such operations always returns the value of the logical type: `true` or `false`.

Comparison may be one of the most difficult and confusing topics you encounter on your way to mastering JavaScript. Today we will try to defeat it once and for all.

§1. Comparison operators list

There are several specific comparison operators in JS, but you probably know most of them from school math:

- greater than `>`, less than `<`
- greater than or equal `>=`, less than or equal `<=`
- equal `==` and not equal `!=` (in mathematics it is marked by the symbol `≠`);
- strict equal `===` and strict not equal `!==`;

> Note that double and triple equality marks are used for comparison. A single sign of equality would mean an assignment.

Let's look at how you can work with comparison operators and different data types.

§2. Numbers

Comparison operations between numbers are quite intuitive:

```
console.log(5 > 3);   // true
console.log(4 < 2);   // false
console.log(7 == 6);  // false
console.log(7 != 6);  // true
```

They're unlikely to cause you much trouble.

§3. Strings

In Javascript, you can compare not only numbers, but also other types of data. For example, strings. Strings are compared to letters; the greater one is the string whose letter is compared later in the Unicode character list:

```
console.log("A" == "A");             // true
console.log("A" != "Z");             // true
console.log("Z" > "A");              // true
console.log("colorful" > "color");   // true
console.log("Dog" > "Bird");         // true
```

The comparison will continue until one of the strings is finished. If both lines end simultaneously, they are equal. Otherwise, a longer string is considered to be greater.

> The order in which the characters are lined up in the Unicode table is very similar to alphabetic, but with a slight difference: lower case letters in Unicode will be larger than upper case letters. For example, it's true that `"z" > "Z"`.

§4. Different types

Current topic:

Comparison operators

Topic depends on:

× Type conversion

Topic is required for:

Conditional operators

Anonymous function

Table of contents:

When you need to compare different types of data to a number, JavaScript reduces each of them to a number:

```
console.log("10" > 5);    // true
console.log(14 == "14");  // true
```

The logical value `true` in such cases becomes `1`, and `false` is considered `0`.

```
console.log(true == 1);   // true
console.log(false == 0);  // true
```

§5. Strict equal and strict not equal

In the end, we should consider strict equal `===` and strict not equal `!==`. They differ from common operators equal `==` and not equal `!=` in that they compare not only values but data types as well:

```
console.log("15" === 15);         // false
console.log("15" == 15);          // true
console.log(true === 1);          // false
console.log(true == 1);           // true
console.log(null === undefined);  // false
console.log(null == undefined);   // true
```

§6. null and undefined

In all the considered algorithms of operators' work, there are several exceptions. For example, `null` behaves strangely in terms of mathematics when compared to `0`:

```
console.log(null > 0);  // false
console.log(null == 0); // false
console.log(null >= 0); // true
console.log(null <= 0); // true
```

Also, any comparison of `undefined` with `0` is always `false`:

```
console.log(undefined > 0);  // false
console.log(undefined < 0);  // false
console.log(undefined <= 0); // false
```

JS has another strange rule: `undefined` can only equal `null` and nothing more:

```
console.log(undefined == null);  // true
console.log(undefined == 0);     // false
```

Now you know how comparison operators work. Try to be careful with all comparison operations where there are `null` or `undefined`, so that no errors appear in your scripts.

Report a typo

125 users liked this theory. **3** didn't like it. **What about you?**

Start practicing

Comments (8) Hints (0) Useful links (0) Show discussion