https://jsmastery.pro

Oh JavaScript!
The Weird Parts

{JS} MASTERY JavaScript Mastery

1. Zero and Empty String

```
0 == false // true
'' == false // true
```

Why does **0 == false** in JavaScript? It's due to **implicit type coercion** which happens when the loose equality operator **(==)** is used.

In the above example, **false** is coerced to **0.** So, since **0 is equal to 0,** the comparison returns **true!**

An **empty string** (‘’) behaves similarly.

2. Null

```
null <= 0 // true
null >= 0 // true
```

In Javascript, when the comparison operators such as **‘>=, =<, >, <’** are used, **null** is treated as **zero.**

So, the above examples are true.

```
0 <= 0 // true
0 >= 0 // true
```

3. NaN

```
NaN === NaN // false
```

The NaN global property in JavaScript is a value representing **Not-A-Number.**

It represents the result of an operation that does not produce a **well-defined numerical result.**

It’s behavior can be confusing as it is **not equal to any other value in JavaScript, including itself.**

4. Objects

```
let obj1 = { name: "Sarah", age: 30 }
let obj2 = { name: "Sarah", age: 30 }
obj1 == obj2 // false
obj1 === obj2 // false
```

Even though **obj1** and **obj2** have the same **structure** and **content,** they are two separate objects **stored in different locations in memory.**

Primitive values like numbers, strings, and booleans are compared by **value,** whereas objects are compared by **reference.**

5. Subtraction of Strings

```
'ab' + 'cd' // 'abcd'
'ab' - 'cd' // NaN
```

In JavaScript, when we add two strings, they get **concatenated** to form a single string.

But when we try to subtract one string from another, JavaScript attempts to **convert those strings to numbers.**

If either string cannot be converted to a number, **the result is NaN.**

6. Logical Operators

```
true || 'hello' // true
true && 'hello' // 'hello'
```

Logical operators in JavaScript can be tricky 😉

The logical OR (||) operator returns the first truthy value it encounters **or the last value if none are truthy.**

The logical AND (&&) operator returns the first falsy value it encounters **or the last value if none are falsy.**

7. Arrays and Loose Equality

```
Array(3) == ",," // true
```

The above code may look a little strange, but let's break it down!

Array(3) in JavaScript creates a new array with a length of 3, but **without initializing the elements.**

This means it will look like this:

```
[undefined, undefined, undefined]
```

7. Arrays and Loose Equality

Now, we know that when **comparing values of different types using ==,** JavaScript attempts to convert the operands to a **common type** before making the comparison.

In this case, the array **[undefined, undefined, undefined]** is coerced to a **string** where the words 'undefined' are removed, leaving only the commas - **",,"** - which means that our comparison is indeed **true!**

Do you know of any other 'weird parts' of JavaScript? We'd love to hear them! 🔥