**Armen Melkumyan** • 1st
Technical / Solutions Architect
2mo •

🔥 From JavaScript Technical Interviews: Identify Outcomes of Tricky Coercion Cases

What is the result of the following operations?

```
console.log(1 + "2" + 3);  // "123"
console.log("5" - 2);
console.log("10" * "2");
console.log(false == 0);
console.log(false === 0);
console.log("123" == 123);
console.log(null == undefined);
console.log(null === undefined);
console.log([] + {});
console.log({} + []);
```

What's happening here?

1 + "2" + 3 → "123"

Numbers get converted to strings in a + operation with a string.

"5" - 2 → 3

String "5" is converted to number 5, then 5 - 2 = 3.

"10" * "2" → 20

Both strings convert to numbers, 10 * 2 = 20.

false == 0 → true

false becomes 0, so 0 == 0 is true.

false === 0 → false

Strict equality checks type as well; boolean ≠ number.

"123" == 123 → true

String "123" becomes number 123.

null == undefined → true

null and undefined are loosely equal.

null === undefined → false

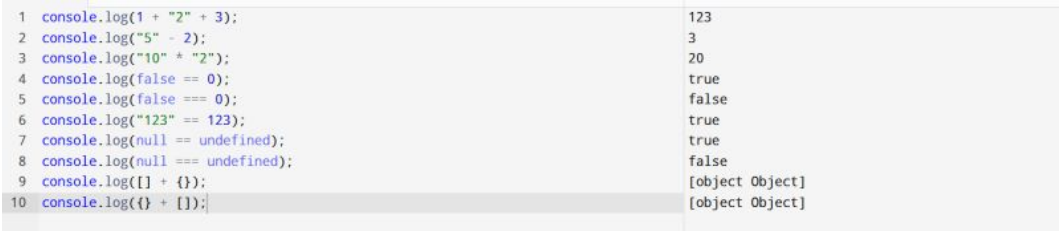
Strict equality fails since types differ.

[] + {} → "[object Object]"

An empty array [] converts to "", and {} converts to "[object Object]". The result is their concatenation.

{} + [] → 0 or "[object Object]" depends on environment type and JavaScript engine.

#JavaScript #Coding #TechInterview



```
console.log(1 + "2" + 3);
console.log("5" - 2);
console.log("10" * "2");
console.log(false == 0);
console.log(false === 0);
console.log("123" == 123);
console.log(null == undefined);
console.log(null === undefined);
console.log([] + {});
console.log({} + []);
```

```
123
3
20
true
false
true
true
false
[object Object]
[object Object]
```