LEARN JAVASCRIPT

🔍 Unraveling the Mysteries of JavaScript Operators: 10 Essential Coding Questions! 🔍

Coding Questions and explanations!

Question: What is the result of 10 + 20 + "30" in JavaScript?

Answer: The result is "3030".

Explanation: JavaScript performs the operation left to right. 10 + 20 is evaluated first, resulting in 30. Then 30 + "30" performs string concatenation, resulting in "3030".

Question: How does the === operator differ from == in JavaScript?

Answer: === is the strict equality operator, while == is the loose equality operator.
Explanation: === compares both the value and type of the operands, whereas == converts the operands to the same type before making the comparison. For example, 0 == '0' is true but 0 === '0' is false.

Question: What is the result of !true && (!true || 100)?

Answer: The result is false.
Explanation: The not operator (!) flips true to false. So, the expression becomes false && (false || 100). Since false || 100 evaluates to 100 (truthy), the entire expression becomes false && true, which results in false.

Question: What does the ?? operator do in JavaScript?

Answer: It's the nullish coalescing operator.

Explanation: ?? returns the right-hand operand when the left-hand operand is null or undefined, otherwise it returns the left-hand operand. For example, null ?? 'default' results in 'default'.

Question: What will be the output of 3 > 2 > 1?

Answer: The output is false.

Explanation: This is due to the left-to-right evaluation of operators. 3 > 2 is true, but true is treated as 1 in the next comparison, so it becomes 1 > 1, which is false.

Question: What is the purpose of the typeof operator?

Answer: It returns a string indicating the type of the operand.

Explanation: For example, typeof "Hello" returns "string", and typeof 5 returns "number".

Question: How does the post-increment operator work? Provide an example.

Answer: It increments the variable but returns the value before incrementing.

Explanation:

```
let x = 5;
let y = x++;
// x is now 6, but y is 5
```

Here,

y is assigned the value of x before x is incremented, so y becomes 5, and x becomes 6.

Question: What will 4 | 3 return in JavaScript?

Answer: It returns 7.

Explanation: The | operator performs a bitwise OR. In binary, 4 is 0100 and 3 is 0011. The OR operation on each bit gives 0111, which is 7 in decimal.

Question: What is the result of the expression 2 ** 3?

Answer: The result is 8.

Explanation: The ** operator is used for exponentiation. 2 ** 3 means 2 raised to the power of 3, which equals 8.

Question: How does the comma operator (,) work in JavaScript? Provide an example.

Answer: It evaluates each of its operands and returns the value of the last operand.

Explanation:

```
let a = (1, 2, 3);
```

Here, 1, 2, and 3 are evaluated, but a is assigned the value 3, which is the last operand.