**JAVASCRIPT OPERATORS**

**Question 1: What are the different types of operators in JavaScript? Explain with examples. • Arithmetic operators • Assignment operators • Comparison operators • Logical operators**

**Different Types of Operators in JavaScript**

JavaScript operators are symbols or keywords used to perform operations on values or variables. They can manipulate data, compare values, and make logical decisions.

**1. Arithmetic Operators**

Arithmetic operators are used to perform basic mathematical calculations.

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| + | Addition | 5 + 3 | 8 |
| - | Subtraction | 5 - 3 | 2 |
| \* | Multiplication | 5 \* 3 | 15 |
| / | Division | 6 / 3 | 2 |
| % | Modulus (remainder) | 5 % 2 | 1 |
| ++ | Increment (adds 1) | let x = 5; x++ | 6 |
| -- | Decrement (subtracts 1) | let x = 5; x-- | 4 |

**Example:**

let a = 10, b = 3;

console.log(a + b); // Output: 13

console.log(a % b); // Output: 1

**2. Assignment Operators**

Assignment operators are used to assign values to variables.

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| = | Assign | x = 5 | x is 5 |
| += | Add and assign | x += 3 | x = x + 3 |
| -= | Subtract and assign | x -= 2 | x = x - 2 |
| \*= | Multiply and assign | x \*= 4 | x = x \* 4 |
| /= | Divide and assign | x /= 2 | x = x / 2 |
| %= | Modulus and assign | x %= 3 | x = x % 3 |

**Example:**

let x = 10;

x += 5; // Equivalent to x = x + 5

console.log(x); // Output: 15

**3. Comparison Operators**

Comparison operators are used to compare two values. They return a boolean (true or false).

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| == | Equal to (loose comparison) | 5 == "5" | true |
| === | Strict equal to | 5 === "5" | false |
| != | Not equal to (loose) | 5 != "5" | false |
| !== | Strict not equal to | 5 !== "5" | true |
| > | Greater than | 5 > 3 | true |
| < | Less than | 5 < 3 | false |
| >= | Greater than or equal to | 5 >= 5 | true |
| <= | Less than or equal to | 5 <= 3 | false |

**Example:**

console.log(10 > 5); // Output: true

console.log(10 === "10"); // Output: false

**4. Logical Operators**

Logical operators are used to perform logical operations, often in conditions.

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| && | Logical AND | true && false | false |
| ` |  | ` | Logical OR |
| ! | Logical NOT | !true | false |

**Example:**

let a = true, b = false;

console.log(a && b); // Output: false

console.log(a || b); // Output: true

console.log(!a); // Output: false

**Examples Combining Operators**

**Arithmetic + Assignment:**

let num = 10;

num += 5; // Equivalent to num = num + 5

console.log(num); // Output: 15

**Comparison + Logical:**

let age = 25;

console.log(age > 18 && age < 30); // Output: true

console.log(age < 18 || age > 30); // Output: false

Understanding these operators is crucial as they form the foundation for manipulating and evaluating data in JavaScript!

**Question 2: What is the difference between == and === in JavaScript?**

In JavaScript, the difference between == and === lies in how they compare values:

**1. == (Equality Operator)**

* Performs **type conversion** (also called coercion) if the operands are of different types.
* It compares the values **after converting them to the same type**.

**Examples:**

5 == '5' // true, because '5' (string) is converted to 5 (number)

0 == false // true, because false is converted to 0

null == undefined // true, because both are considered "empty" values

**2. === (Strict Equality Operator)**

* Does **not perform type conversion**.
* It checks for **both value and type equality**.

**Examples:**

5 === '5' // false, because the types (number and string) are different

0 === false // false, because the types (number and boolean) are different

null === undefined // false, because the types are different

**Key Takeaway:**

* Use === when you want **strict comparison** to avoid unexpected results caused by type coercion.
* Use == only if you intentionally want type conversion to happen (rare cases).

**Best Practice:**

Always prefer === over == unless you have a specific reason to use loose equality.