JavaScript Operators

An **operator** is a symbol that performs an operation on values or variables.
Example: `let a = 10 + 5;  // '+' is an operator`

◈ 1. Arithmetic Operators

Used for mathematical calculations.

Operator	Description	Example	Output
`+`	Addition	`10 + 5`	15
`-`	Subtraction	`10 - 5`	5
`*`	Multiplication	`10 * 5`	50
`/`	Division	`10 / 5`	2
`%`	Modulus (remainder)	`10 % 3`	1
`**`	Exponentiation	`2 ** 3`	8
`++`	Increment	`a++`	Adds 1
`--`	Decrement	`a--`	Subtracts 1

◈ 2. Assignment Operators

Used to assign values.

Operator	Example	Same As
`=`	`x = 5`	x = 5
`+=`	`x += 5`	x = x + 5
`-=`	`x -= 5`	x = x - 5
`*=`	`x *= 5`	x = x * 5
`/=`	`x /= 5`	x = x / 5
`%=`	`x %= 5`	x = x % 5

◈ 3. Comparison Operators

Used to compare two values (returns `true` or `false`).

Operator	Description	Example	Output
`==`	Equal to	`5 == "5"`	true
`===`	Strict equal (value + type)	`5 === "5"`	false
`!=`	Not equal	`5 != 6`	true
`!==`	Strict not equal	`5 !== "5"`	true
`>`	Greater than	`10 > 5`	true
`<`	Less than	`10 < 5`	false
`>=`	Greater or equal	`10 >= 10`	true
`<=`	Less or equal	`10 <= 5`	false

🔹 4. Logical Operators

Operator	Meaning	Example	Result
`&&`	AND	`(a > 0 && b > 0)`	true if both true
`||`	OR	`(a > 0 || b > 0)`	true if one true
`!`	NOT	`!(a > 0)`	reverses the result

🔹 5. String Operators

The plus sign (+) is used to concatenate (join) strings.

Operator	Description	Example	Result
`+`	Concatenation	`"Hello " + "World"`	`"Hello World"`
`+=`	Add & assign	`text += "JS"`	append text

🔹 6. Ternary Operator

Shortcut for `if...else`.

Example:

```
let age = 18;
let result = (age >= 18) ? "Adult" : "Minor";
console.log(result);  // Adult
```

Other miscellaneous operators

- **`typeof`**: Returns a string indicating the data type of an operand, e.g., `typeof 42` returns `"number"`.
- **`delete`**: Deletes a property from an object, e.g., `delete person.age`.
- **`in`**: Returns `true` if a property exists in a specified object.
- **`instanceof`**: Returns `true` if an object is an instance of a specified object type.
- **Comma (`,`)**: Evaluates multiple expressions and returns the value of the last one.
- **Spread (`...`)**: Expands an iterable into individual elements.

Template Literals

Template literals are an **easier way to create strings** in JavaScript.
They allow:

- Multi-line strings
- Embedding variables and expressions inside strings
- Using backticks (`` ` ``) instead of quotes (`'` or `"`)

Syntax:

```
`string text ${expression} string text`
```

- **Use backticks** (`` ` ``), not single or double quotes.
- Inside `${}`, you can write **variables** or even **expressions**.

Example 1 – Basic Variable Insertion

```
let name = "Raj";
let age = 24;

console.log(`My name is ${name} and I am ${age} years old.`);
```

🖨 Output:

```
My name is Raj and I am 24 years old.
```

Example 2 – Expression Inside `${}`

```
let a = 10;
let b = 5;

console.log(`The sum of ${a} and ${b} is ${a + b}.`);
```

🖨 Output:

```
The sum of 10 and 5 is 15.
```

Example 3 – Multi-line String

```
let message = `
Hello everyone,
This is a multi-line string
using template literals.
`;

console.log(message);
```

🖨 Output:

```
Hello everyone,
This is a multi-line string
using template literals.
```

Example 4 – Inside Object

```
let user = { name: "Aarav", country: "Nepal" };
console.log(`User ${user.name} is from ${user.country}.`);
```

🖨 Output:

```
User Aarav is from Nepal.
```

✅ Mini Assignment (Practice)

1. Create two variables `a` and `b` with any numbers.
2. Perform all arithmetic and comparison operations and print results.
3. Join two strings with `+`.
4. Use a ternary operator to check if a number is positive or negative.
5. Use **nested ternary** to grade marks: **(`marks >= 90 → A+, marks >= 80 → A, marks >= 70 → B,` otherwise → `Fail`)**