To master JavaScript operators, we’ll start from the basics and work up to advanced usage. This "Operators from Zero to Hero" guide will cover these topics:

1. **Basic Operators**
   * Arithmetic Operators
   * Assignment Operators
   * Comparison Operators
   * Logical Operators
2. **Bitwise Operators**
3. **Advanced Operators**
   * Ternary (Conditional) Operator
   * Nullish Coalescing Operator (??)
   * Optional Chaining (?.)
4. **Operator Precedence**

Let’s dive in!

**1. Basic Operators**

**Arithmetic Operators**

These are used for basic math operations.

* + Addition
* - Subtraction
* \* Multiplication
* / Division
* % Modulus (remainder)
* \*\* Exponentiation

Example:

let x = 10;

let y = 3;

console.log(x + y); // 13

console.log(x - y); // 7

console.log(x \* y); // 30

console.log(x / y); // 3.33

console.log(x % y); // 1

console.log(x \*\* y); // 1000

**Assignment Operators**

Assignment operators assign values to variables.

* = Basic assignment
* += Adds to current value
* -= Subtracts from current value
* \*= Multiplies with current value
* /= Divides by current value

Example:

let a = 5;

a += 3; // a = a + 3 -> a = 8

console.log(a);

a \*= 2; // a = a \* 2 -> a = 16

console.log(a);

**Comparison Operators**

Used to compare two values, returning true or false.

* == Equal to
* === Strict equal to (checks value and type)
* != Not equal
* !== Strict not equal
* > Greater than
* < Less than
* >= Greater than or equal to
* <= Less than or equal to

Example:

console.log(5 == '5'); // true (checks only value)

console.log(5 === '5'); // false (checks value and type)

console.log(10 > 6); // true

**Logical Operators**

Used for combining multiple conditions.

* && Logical AND
* || Logical OR
* ! Logical NOT

Example:

let age = 20;

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

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

console.log(!(age === 20)); // false

**2. Bitwise Operators**

These work on binary representations of numbers. Useful in low-level programming.

* & Bitwise AND
* | Bitwise OR
* ^ Bitwise XOR
* ~ Bitwise NOT
* << Left shift
* >> Right shift

Example:

let n1 = 5; // binary: 0101

let n2 = 3; // binary: 0011

console.log(n1 & n2); // 1 (binary: 0001)

console.log(n1 | n2); // 7 (binary: 0111)

**3. Advanced Operators**

**Ternary (Conditional) Operator**

Short-hand for if...else conditions.

let age = 18;

let message = age >= 18 ? "Adult" : "Minor";

console.log(message); // "Adult"

**Nullish Coalescing Operator (??)**

Returns the right-side value if the left is null or undefined.

let name = null;

let defaultName = name ?? "Guest";

console.log(defaultName); // "Guest"

**Optional Chaining (?.)**

Access properties safely, even if they don’t exist.

let user = { name: "Alice", address: { city: "Paris" }};

console.log(user.address?.city); // "Paris"

console.log(user.contact?.phone); // undefined, without error

**4. Operator Precedence**

JavaScript evaluates expressions based on operator precedence. For example, \* has higher precedence than +, so 2 + 3 \* 4 equals 14 (multiplication happens first).

To control precedence, use parentheses:

let result = (2 + 3) \* 4; // 20

By practicing these operators and gradually using them in projects, you’ll move from zero to hero in JavaScript!