JavaScript Operators

Agenda: Mastering JavaScript Operators

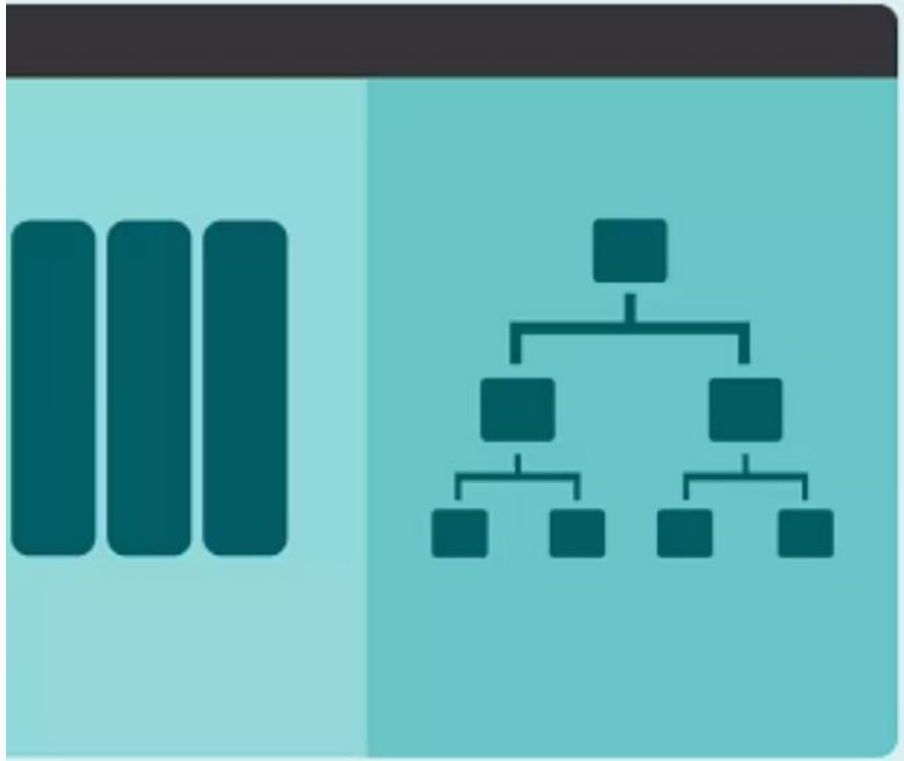
1. Arithmetic Operators
2. Comparison & Equality
3. Logical Operators
4. Optional Chaining (`?.`)
5. Interview Pro Tips
    - Top questions (+ solutions).
    - Best practices to stand out.
6. Q&A + Practice Challenge

Arithmetic Operators

- `+` (Addition)
- `-` (Subtraction)
- `*` (Multiplication)
- `/` (Division)
- `%` (Modulus/Remainder)

Arithmetic Operators

```
let a = 10, b = 3;
console.log(a + b);  // 13 (Addition)
console.log(a - b);  // 7 (Subtraction)
console.log(a * b);  // 30 (Multiplication)
console.log(a / b);  // 3.33 (Division)
console.log(a % b);  // 1 (Modulus)
```



Comparison Operators

- `==` (Loose Equality)
- `===` (Strict Equality)
- `!=` / `!==` (Inequality)
- `>` (Greater Than)
- `<` (Less Than)

Comparison Operators

```
let x = 5, y = "5";
console.log(x == y);   // true (Loose Equality)
console.log(x === y);  // false (Strict Equality)
console.log(x !== y);  // true (Strict Not Equal)
console.log(x > 2);    // true (Greater Than)
```

Logical Operators

- `&&` (AND)
- `||` (OR)
- `??` (Nullish Coalescing)

3. Logical Operators

```
let isLoggedIn = true, hasPermission = false;
console.log(isLoggedIn && hasPermission); // false (AND)
console.log(isLoggedIn || hasPermission); // true (OR)

let userName = null;
console.log(userName ?? "Guest"); // "Guest" (Nullish Coalescing)
```

4. Optional Chaining (`?.`)

```
const user = { profile: { name: "Alice" } };
console.log(user?.profile?.name); // "Alice"
console.log(user?.address?.city); // undefined (No error)
```

Pro Tips for Interviews (Core Concepts to Master)

- **Truthy/Falsy Values:**
  - Falsy: `false`, `0`, `""`, `null`, `undefined`, `NaN`.
  - **Trap Question:**

```
if ("0") { console.log("Runs!"); } // Truthy (non-empty string).
```

Nullish Coalescing (`??`) vs OR (`||`):

- `||` checks for *any* falsy value.
- `??` only checks for `null`/`undefined`.

```
console.log(0 || 100);    // 100 (falsy)
console.log(0 ?? 100);    // 0 (nullish)
```

- `==` **vs** `===`:
  - `==` checks value (with type coercion), e.g., `5 == "5"` → `true`.
  - `===` checks value + type (no coercion), e.g., `5 === "5"` → `false`.
  - **Pro Tip**: Always use `===` unless you *need* coercion (rare).

Common Interview Questions

Q1: Why does `[] == ![]` return `true`?

- `![]` → `false` → `false` coerced to `0`.
- `[]` coerced to `""` → then to `0`.
- Final: `0 == 0` → `true`.

Q2: How does `?.` differ from `&&` for property access?

```
user && user.address && user.address.city;  // Pre-optional chaining
user?.address?.city;                        // Modern equivalent
```

Quiz Time!

```
Bonus: Interviewer’s Hidden Agenda
They want to see if you:-
:) Understand implicit vs explicit coercion.
:) Know when to use ?? vs ||.
:) Can explain why a quirky result happens (e.g., "b" + "a" + +"a" + "a" → "baNaNa").
```

Resources

- Free eBook: "JavaScript Quick Reference"

W3Schools.com

https://www.geeksforgeeks.org/javascript/

- Follow for daily JS tips
- GitHub repo with code examples