JavaScript `Switch` Statement

Notes by: Pramod Sir

🔹 What is a `switch` Statement?

The `switch` statement is used to **perform different actions based on different conditions**. It is a cleaner alternative to multiple `if...else if` blocks — especially when you're checking **one variable against many values**.

🔸 Syntax of `switch`

```
switch (expression) {
  case value1:
    // code to execute if expression === value1
    break;
  case value2:
    // code to execute if expression === value2
    break;
  default:
    // code to execute if no match found
}
```

🔍 Key Points

- `expression` is **compared using strict equality (`===`)** with each `case` value.
- The `break` statement **stops execution** once a match is found. Without it, the next cases will also run (known as **fall-through**).

- `default` is **optional** and runs when no `case` matches.

✅ Basic Example: Weekday Checker

```
let day = 3;

switch (day) {
  case 1:
    console.log("Monday");
    break;
  case 2:
    console.log("Tuesday");
    break;
  case 3:
    console.log("Wednesday");
    break;
  case 4:
    console.log("Thursday");
    break;
  case 5:
    console.log("Friday");
    break;
  case 6:
    console.log("Saturday");
    break;
  case 7:
    console.log("Sunday");
    break;
  default:
    console.log("Invalid day");
}
```

🌐 Real-Life Example: Traffic Signal

```
let signal = prompt("Enter signal color:");

switch (signal.toLowerCase()) {
  case "red":
    alert("STOP");
    break;
  case "yellow":
    alert("GET READY");
    break;
  case "green":
    alert("GO");
    break;
  default:
    alert("Invalid color!");
}
```

📌 When to Use `switch`

Use `switch` when:

- You are **checking the same variable** against multiple values.
- You want **cleaner** and more **organized** code than multiple `else if`.

⚠️ Common Mistakes

Mistake	Explanation
Forgetting `break`	Causes **fall-through**, executing multiple cases
Using `switch` for ranges	Better handled using `if...else if`
Comparing different data types	`switch` uses **strict equality** (`===`)

🧠 Interview Perspective

❓ Q1: When should you prefer a `switch` over `if...else`?

Answer:
When you're checking **one variable** against **many possible exact values** (like `1`, `"admin"`, etc.), a `switch` is more readable and efficient than multiple `else if` blocks.

❓ Q2: What happens if `break` is missing in a `switch`?

Answer:
If `break` is missing, the code will **continue executing the next cases**, even if a match was already found — this is known as **fall-through** behavior.

❓ Q3: Can we use expressions or comparisons in `case`?

Answer:
No. Case values must be **constant expressions**. You can't use conditions like `case x > 5:` — use `if` for such cases.

Practice Task for Students

Try the following yourself:

```
let grade = prompt("Enter your grade (A, B, C, D, F):");

switch (grade.toUpperCase()) {
  case "A":
    alert("Excellent!");
    break;
  case "B":
```

```
    alert("Good job!");
    break;
  case "C":
    alert("Average");
    break;
  case "D":
    alert("Needs improvement");
    break;
  case "F":
    alert("Failed");
    break;
  default:
    alert("Invalid grade");
}
```

📌 Summary

Feature	switch	if...else
Use case	One variable, many fixed values	Multiple variables or range checking
Readability	High for fixed values	Better for complex conditions
Performance	Slightly better in large checks	Slower in large nested checks