Switch statements and fall-through behavior

Sure! Let's explore `switch` **statements** in JavaScript and understand the concept of **fall-through behavior** in detail.

🔄 What is a `switch` Statement?

A `switch` statement is used to **evaluate an expression** and execute **one of many code blocks** based on the **matching case**.

It is often used as a cleaner alternative to multiple `if...else if...else` statements, especially when you're comparing the same variable to different values.

🧠 Basic Syntax:

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

✅ Example:

```
let day = 3;

switch (day) {
```

```
  case 1:
    console.log("Monday");
    break;
  case 2:
    console.log("Tuesday");
    break;
  case 3:
    console.log("Wednesday");
    break;
  default:
    console.log("Invalid day");
}
```

📌 Output: `Wednesday`

📌 Important Concepts

1. `break` Statement

- It **stops** the execution of more cases.
- If `break` is **not used**, **fall-through** occurs.

---

2. Fall-Through Behavior

If `break` is omitted, the program continues to execute the next case(s), **even if they don't match**.

⚠️ Example:

```
let color = "red";

switch (color) {
  case "red":
    console.log("Color is red");
  case "blue":
    console.log("Color is blue");
```

```
  case "green":
    console.log("Color is green");
  default:
    console.log("Unknown color");
}
```

📌 Output:

```
Color is red
Color is blue
Color is green
Unknown color
```

👉 This happens because there are **no** `break` **statements**, so it **falls through** all cases.

✅ Corrected with `break`:

```
switch (color) {
  case "red":
    console.log("Color is red");
    break;
  case "blue":
    console.log("Color is blue");
    break;
  case "green":
    console.log("Color is green");
    break;
  default:
    console.log("Unknown color");
}
```

📌 Output: `Color is red`

🌟 Default Case

- Acts like the `else` in `if...else`
- Runs when no case matches

🔁 Practical Example: Grading System

```
let grade = "B";

switch (grade) {
  case "A":
    console.log("Excellent!");
    break;
  case "B":
    console.log("Well done!");
    break;
  case "C":
    console.log("Good.");
    break;
  default:
    console.log("Grade not recognized.");
}
```

📌 Output: `Well done!`

🔂 Intentional Fall-Through (Advanced Use)

You can use fall-through **on purpose**, like grouping cases:

```
let fruit = "apple";

switch (fruit) {
  case "apple":
  case "banana":
  case "mango":
    console.log("Fruit is available.");
    break;
```

```
  default:
    console.log("Fruit is not available.");
}
```

📌 Output: `Fruit is available.`

Summary

Keyword	Purpose
`switch`	Starts the switch block
`case`	Checks if the expression matches the value
`break`	Exits the switch block
`default`	Runs if no case matches
Fall-through	When no `break`, control passes to next case

Would you like a **flowchart diagram**, a **comparison with if-else**, or some **practice questions**?