JavaScript learning path

Perfect 👍 Let's **redo the `switch...case` deep dive**, but this time I'll make it **structured**:
👉 Each subtopic will have **syntax → code → output → explanation**.
This way you see **how it behaves under the hood**.

🚦 `switch...case` in JavaScript — Detailed with Examples

---

1. 🔹 Basic Syntax

```
js

let fruit = "apple";

switch (fruit) {
  case "apple":
    console.log("It's an Apple");
    break;
  case "banana":
    console.log("It's a Banana");
    break;
  default:
    console.log("Unknown fruit");
}
```

Output:

```
It's an Apple
```

✅ `fruit` is `"apple"`, matches first case.
✅ `break` stops further execution.

---

2. 🔹 Fallthrough (When `break` is Missing)

```js
let fruit = "apple";

switch (fruit) {
  case "apple":
    console.log("Apple matched");
  case "banana":
    console.log("Banana matched");
  default:
    console.log("Default block");
}
```

Output:

```scss
Apple matched
Banana matched
Default block
```

⚠ Without `break`, control **flows down** into the next cases.

---

3. 🔹 Grouping Multiple Cases

```js
let day = 6;

switch (day) {
  case 1: case 2: case 3: case 4: case 5:
    console.log("Weekday");
    break;
  case 6: case 7:
    console.log("Weekend");
    break;
  default:
    console.log("Invalid day");
}
```

Output:

```
nginx

Weekend
```

✅ Multiple cases grouped into one block.

---

4. 🔹 `switch` vs `if-else` (Discrete vs Ranges)

```
js

let score = 85;

switch (score) {
  case 100:
    console.log("Perfect score");
    break;
  case 90:
    console.log("Excellent");
    break;
  default:
    console.log("Other score");
}
```

Output:

```
nginx

Other score
```

👉 `switch` checks **discrete values** only.
👉 For ranges, `if-else` is better.

---

5. 🔹 Trick: Range Checks with `switch(true)`

```
js

let age = 20;
```

```
switch (true) {
  case (age < 13):
    console.log("Child");
    break;
  case (age >= 13 && age < 20):
    console.log("Teenager");
    break;
  case (age >= 20 && age < 60):
    console.log("Adult");
    break;
  default:
    console.log("Senior");
}
```

Output:

```
nginx

Adult
```

✅ `switch(true)` allows **conditional checks** like `if-else`.

---

6. 🔹 Strict Equality (===) in Case Matching

```
js

let value = "5";

switch (value) {
  case 5:
    console.log("Number 5");
    break;
  case "5":
    console.log("String 5");
    break;
}
```

Output:

```
arduino
```

```
String 5
```

👉 `"5"` does **not** equal `5` because `switch` uses **strict equality (===)**.

---

7. 🔹 Duplicate Case Pitfall

```js
let num = 2;

switch (num) {
  case 2:
    console.log("First case");
    break;
  case 2: // ignored silently
    console.log("Second case");
    break;
}
```

Output:

```sql
First case
```

⚠ The second `case 2` is **unreachable**.

---

8. 🔹 Objects as Case Values

```js
let obj = {};

switch (obj) {
  case {}:
    console.log("Matched");
    break;
  default:
```

```
  console.log("Not matched");
}
```

Output:

```
mathematica

Not matched
```

👉 Even though both look `{}`, they are **different references** → no match.

---

9. 🔹 `default` Behavior

```
js

let color = "yellow";

switch (color) {
  case "red":
    console.log("Stop");
    break;
  case "green":
    console.log("Go");
    break;
  default:
    console.log("Unknown color");
}
```

Output:

```
css

Unknown color
```

✅ `default` executes when no match is found.

---

10. 🔹 Internal Optimization

```
js
```

```
let num = 3;

switch (num) {
  case 1:
    console.log("Case 1");
    break;
  case 2:
    console.log("Case 2");
    break;
  case 3:
    console.log("Case 3");
    break;
}
```

Output:

```
sql

Case 3
```

💡 If cases are **dense integers** (`1, 2, 3, ...`), engines like V8 may build a **jump table** internally → faster than multiple `if-else`.

⚠ Common Pitfalls

1. Forgetting `break` → unexpected fallthrough.
2. Assuming `"5"` == `5` → nope, strict equality.
3. Using objects as case values → always false (different refs).
4. Duplicate cases → silently ignored.

🧠 Interview Traps

Q1. Can `switch` handle ranges directly?

👉 No. Only equality checks. Trick is `switch(true)`.

Q2. What happens if you omit `break`?

👉 Fallthrough → runs next cases until `break` or end.

Q3. Why doesn't `{}` match `{}` in `switch`?

👉 Because objects are compared by **reference**, not structure.

Q4. When is `switch` faster than `if-else`?

👉 When cases are **dense numbers**, engine may optimize with jump tables.

✅ Now you have **examples + outputs + internals** for every case.

Would you like me to next **compare `switch` vs object-lookup (map of functions)** with real-world examples? This is an **interview favorite** where you replace `switch` with cleaner code.