ScopeTreasure Courses

JAVASCRIPT COURSES · LESSON 3.5 · CODE SCOPE SERIES

If/Else & Switch

Code that decides: if, else, else-if ladders, nested doors, the ternary shortcut — and switch with its break, plus when each one wins.

Introduction

Until now, your code ran top to bottom, every line, every time. Real programs **decide**: a big bill gets a discount, a small one does not; a correct password opens the door, a wrong one does not. The tool for decisions is the **if statement** — run this block only when the condition is true.

In this lesson you will build decisions with `if` and `else`, climb an `else if` ladder, walk through nested doors, take the ternary shortcut, and meet `switch` — the specialist for one value with many fixed options. You will also see the famous `switch` mistake live: forgetting `break`.

(Roman Urdu: if matlab "agar ye sach hai to ye karo" — code ab faislay karna seekh jayega.)

Key points

- **`if` runs a block when the condition is true; `else` catches everything else.** Two paths, exactly one runs: bill big enough → discount; otherwise → full price.
- **The `else if` ladder checks in order — first match wins.** Grading A, then B, then C, then fail: JavaScript tests from the top, runs the **first** true branch, and skips the rest. Order matters: put the highest grade first.

  (Roman Urdu: else if ki seerhi upar se neeche check hoti hai — jo pehle match ho jaye, wahi chalta hai.)
- **Nested if = two doors:** an `if` inside an `if` — first door: logged in? second door: has a coupon? Both must open. Honest hint: often a single condition with `&&` does the same job more cleanly.
- **The ternary `?:` is a one-line if/else:** `condition ? valueIfTrue : valueIfFalse` — perfect for small choices like picking a label or a message.
- **`switch` matches ONE value against many fixed cases:** the value is compared with each `case` using strict equality, and the matching block runs.

  (Roman Urdu: switch aik hi value ko bohat se fixed options se match karta hai.)
- **`break` ends each case — forget it and you fall through:** without `break`, JavaScript keeps running the **next** cases too, even though they do not match. This fall-through is the classic switch bug.
- **When each one wins:** ranges, combos, and conditions → `if`. One value, many fixed options → `switch`.

Code examples

Example 1 — The else if ladder: grading

```
let marks = 72;
let grade;

if (marks >= 80) {
  grade = "A";
} else if (marks >= 60) {
  grade = "B";
} else if (marks >= 40) {
  grade = "C";
} else {
  grade = "Fail";
}

console.log(grade);
```

What it does: Marks are tested from the top. `72` is not `>= 80`, so the first branch is skipped. It **is** `>= 60`, so grade becomes `"B"` and the ladder stops — the C and Fail branches never run.

What you will see: one line in the Console: `B`. Change `marks` to `35` and run again — you will see `Fail`, because only the final `else` catches it.

Example 2 — switch, break, and the fall-through mistake

```
let day = 3;
let dayName;

switch (day) {
  case 1:
    dayName = "Monday";
    break;
  case 2:
    dayName = "Tuesday";
    break;
  case 3:
    dayName = "Wednesday";
    break;
  default:
    dayName = "Unknown day";
}

console.log(dayName);
```

What it does: The value `3` is matched strictly against each case. Case 3 matches, sets the name, and `break` exits the switch cleanly. The `default` case is the safety net for any value that matches nothing.

What you will see: one line in the Console: `Wednesday`. Now delete the `break` after case 3 and run again: the code **falls through** into `default`, and you will see `Unknown day` — that is the fall-through bug, live.

Example 3 — Two doors and the ternary shortcut

```
let isLoggedIn = true;
let hasCoupon = false;
let message;

if (isLoggedIn) {
  if (hasCoupon) {
    message = "Welcome! Coupon applied.";
  } else {
    message = "Welcome! No coupon found.";
  }
} else {
  message = "Please log in first.";
}

console.log(message);

let bill = 1200;
let label = bill >= 1000 ? "Big bill" : "Small bill";
console.log(label);
```

What it does: Nested doors: the outer `if` checks login, the inner `if` checks the coupon — and here the second door stays shut, so the middle message wins. Below, the ternary makes the same kind of two-way choice in one line: is the bill at least 1000?

What you will see: two lines in the Console: `Welcome! No coupon found.` and then `Big bill`.

Quick recap

- `if` / `else if` / `else`: checked in order, **first match wins** — built for ranges and combos.
- `switch`: one value, many fixed cases — and every case needs its `break`, or you fall through.
- Ternary `? :` = the one-line if/else for simple choices.

Common mistakes

1. **Wrong order in an else if ladder** — checking `marks >= 40` before `marks >= 80` means everyone passes into C and nobody reaches A. Fix: order conditions from highest to lowest; first match wins.

2. **Forgetting `break` in switch** — the code falls through and runs the following cases too, giving wrong results that look mysterious. Fix: end every case with `break` (unless you truly want fall-through).
3. **Using switch for ranges** — `switch` only matches exact values, so `case marks >= 80` does not do what beginners expect. Fix: ranges and combos belong to `if`; `switch` is for one value with fixed options.

Practice

1. Write a login check: if the username is `"admin"` and the password is `"chai123"`, print `Login successful` in green style text on a small page; otherwise print `Wrong details` — use an `if` with `&&` and set the message colour with `style.color`.
2. Build a `switch` that takes a number from 1 to 3 and prints a drink name for each (1 = Chai, 2 = Coffee, 3 = Lassi), with a `default` for other numbers. Then remove one `break`, run it, and write down what fall-through did.

Practice quiz

1. How does an else if ladder choose its branch?

- It runs every true branch
- It checks in order and runs the first true branch, skipping the rest
- It runs the last true branch
- It picks at random

Answer: It checks in order and runs the first true branch, skipping the rest

2. What happens in a switch case if you forget break?

- Nothing, break is optional decoration
- The code falls through and runs the next cases too, even though they do not match
- The switch stops with an error
- It jumps back to the first case

Answer: The code falls through and runs the next cases too, even though they do not match

3. When do you choose switch over if?

- For ranges and combinations
- For one value matched against many fixed options; ranges belong to if
- Never, switch is obsolete
- Only for numbers

Answer: For one value matched against many fixed options; ranges belong to if

---

Watch the video lesson: https://youtu.be/GEcDsHcelJ4

Read online: https://scopetreasure.com/notes/js-if-else-switch · More free tools and notes at https://scopetreasure.com