Understanding Decision-Making in JS

Conditionals in JavaScript (`if-else`, `switch-case`, Ternary Operator)

Agenda

1. Introduction to Conditionals
2. `if-else` Statement
3. `switch-case` Statement
4. Ternary Operator
5. Comparison & Use Cases
6. Interview Pro Tips
7. Best Practices
8. Q&A + Practice Challenge
9. Quiz Time

Introduction to Conditionals

Definition:
Conditionals allow JavaScript programs to execute different code blocks based on specified conditions.

Why Use Them?

- Control program flow dynamically
- Handle different scenarios efficiently
- Improve code logic and readability

`if-else` Statement

Key Features:

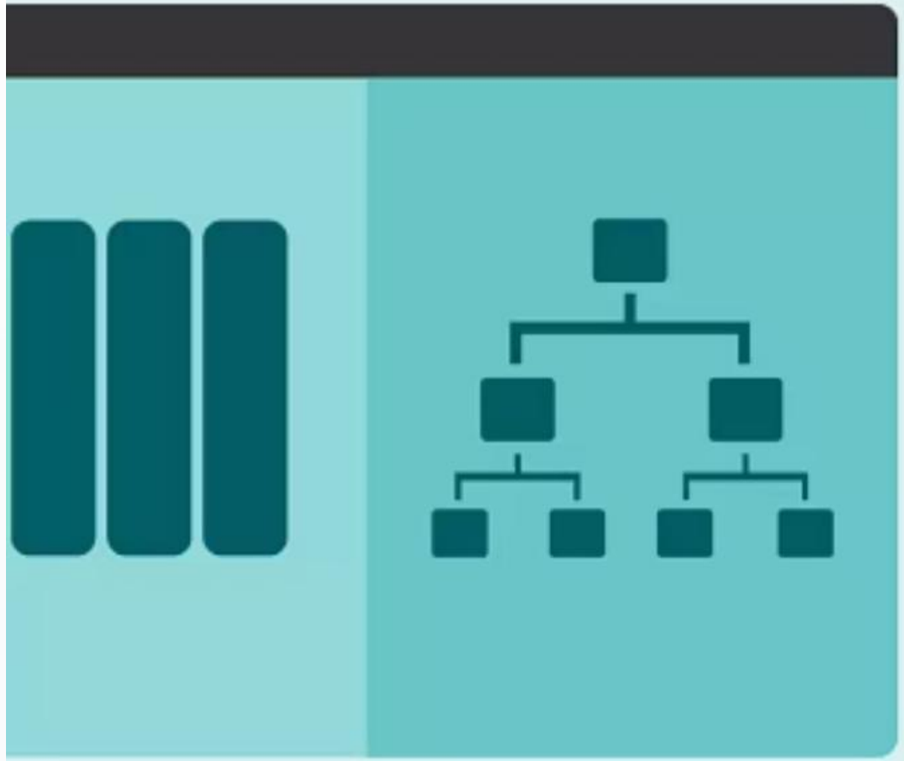
- Executes a block if a condition is `true`
- Optional `else if` and `else` for multiple conditions

Example:

```
if (age >= 18) {
    console.log("You can vote!");
} else {
    console.log("You cannot vote yet.");
}
```

Use Case:

- Simple true/false conditions
- Multiple conditions with `else if`



`switch-case` Statement

Key Features:

- Evaluates an expression against multiple cases
- More efficient than multiple `if-else` for fixed values

Example:

```
switch (day) {
    case "Monday":
        console.log("Week start!");
        break;
    case "Friday":
        console.log("Weekend soon!");
        break;
    default:
        console.log("Midweek");
}
```

Use Case:

- Checking against multiple possible values
- Cleaner alternative to long `if-else` chains

Ternary Operator

Key Features:

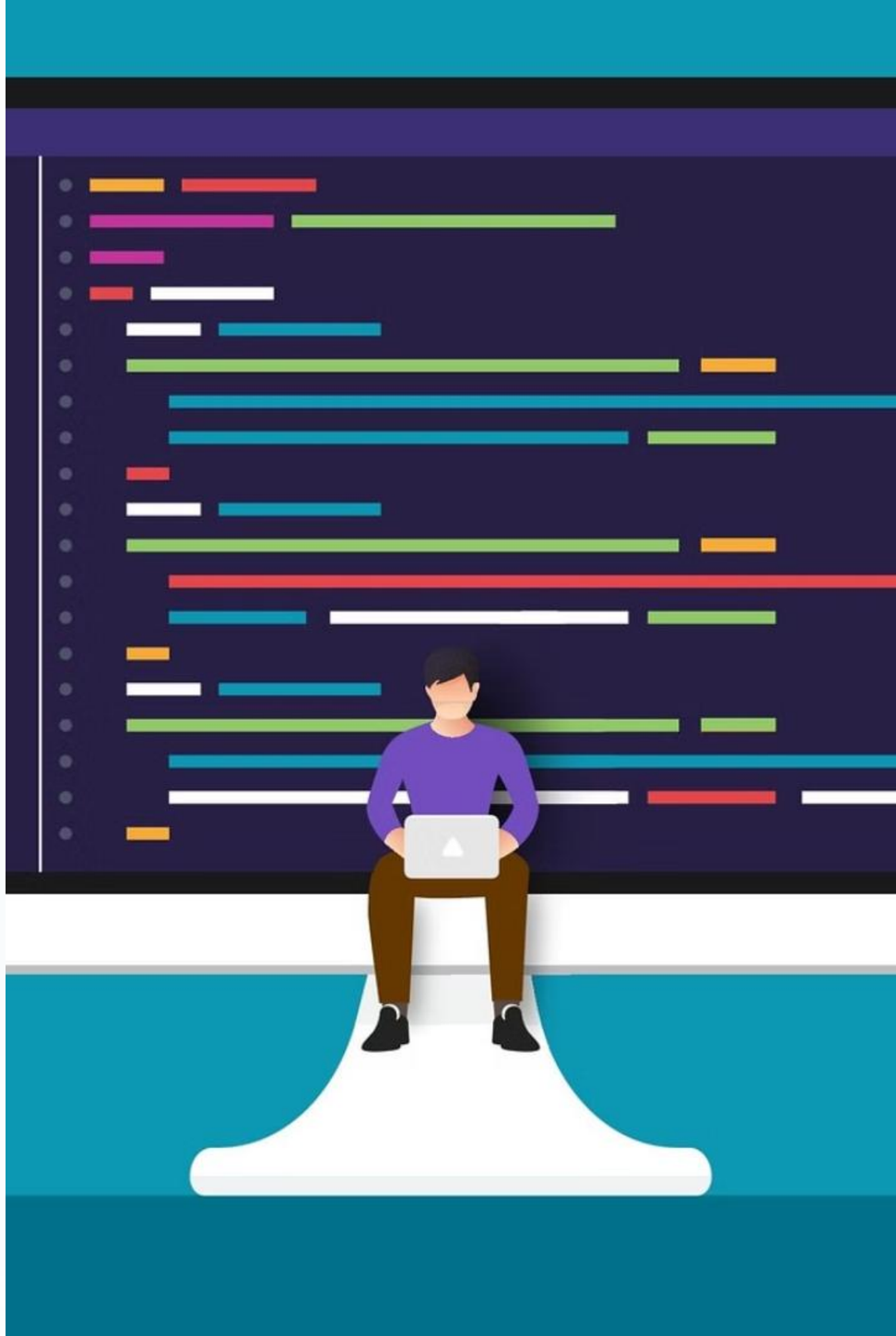
- Shorthand for `if-else`
- Returns a value based on a condition

Example:

```
const result = score >= 50 ? "Pass" : "Fail";
console.log(result);
```

Use Case:

- Quick conditional assignments
- Simple one-liner checks



Comparison & Use Cases Table

Conditional	Best For	Readability	Performance
`if-else`	Complex, range-based conditions	Moderate	Good
`switch-case`	Multiple fixed values	High	Faster (for many cases)
Ternary	Simple true/false assignments	High (if simple)	Best

Interview Pro Tips

✅ When to Use What?

- `switch-case` → Enums, fixed menu options, state machines.
- `if-else` → Range checks (e.g., `age > 18`), complex logic.
- **Ternary** → Short, readable assignments (e.g., `const access = isLoggedIn ? "Yes" : "No"`).

❌ Avoid Common Mistakes:

- Forgetting `break` in `switch-case` (causes fall-through).
- Overusing nested `if-else` (hard to debug).

Best Practices to Stand Out

✓ 1. Keep Conditions Simple

- Avoid deep nesting (use functions or early returns).
- **✓ 2. Always Include a `default` Case**
- Handles unexpected values gracefully.
- **✓ 3. Prefer Readability Over Cleverness**
- Use ternary only when it improves clarity.

Q&A + Practice Challenge

Q&A:

- Can `switch-case` check ranges (e.g., `score > 50`)?
- Is the ternary operator faster than `if-else`?

Challenge:

```
// Convert this `if-else` into a ternary:
let message;
if (isRaining) {
    message = "Take an umbrella!";
} else {
    message = "Enjoy the sun!";
}
```

Q&A Solution:-

Solution:

```
const message = isRaining ? "Take an umbrella!" : "Enjoy the sun!";
```

Quiz Time!

1️⃣ What happens if you omit `break` in `switch-case`?

- A) Syntax Error
- B) Fall-through execution ✅
- C) Nothing

2️⃣ Which is better for checking a variable against 10+ fixed strings?

- A) `if-else`
- B) `switch-case` ✅

3️⃣ Can a ternary operator return a function?

- A) Yes ✅
- B) No

Resources

- Free eBook: "JavaScript Quick Reference"

W3Schools.com

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

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

Thank You!

Recap:

- `if-else` → Best for flexibility
- `switch-case` → Best for multiple fixed cases
- **Ternary** → Best for concise assignments

Call to Action:

- Try coding exercises on CodeWars or LeetCode!