JavaScript Control Flow

Mastering Decision Making & Loops in JS

Agenda

1. Introduction to Control Flow
2. Conditional Statements (`if`, `else`, `switch`)
3. Loops (`for`, `while`, `do-while`, `for...of`, `for...in`)
4. Control Flow Keywords (`break`, `continue`, `return`, `throw`)
5. Error Handling (`try-catch-finally`)
6. Best Practices
7. Interview Pro Tips
8. Q&A + Practice Challenge
9. Quiz Time

Introduction to Control Flow

- **Definition:** The order in which statements are executed.
- **Why it matters?**
  - Makes programs dynamic (decision-making).
  - Handles repetitive tasks efficiently.
- **Types:**
  - **Conditional** (`if-else`, `switch`)
  - **Loops** (`for`, `while`)
  - **Error Handling** (`try-catch`)

Conditional Statements

1. `if-else`

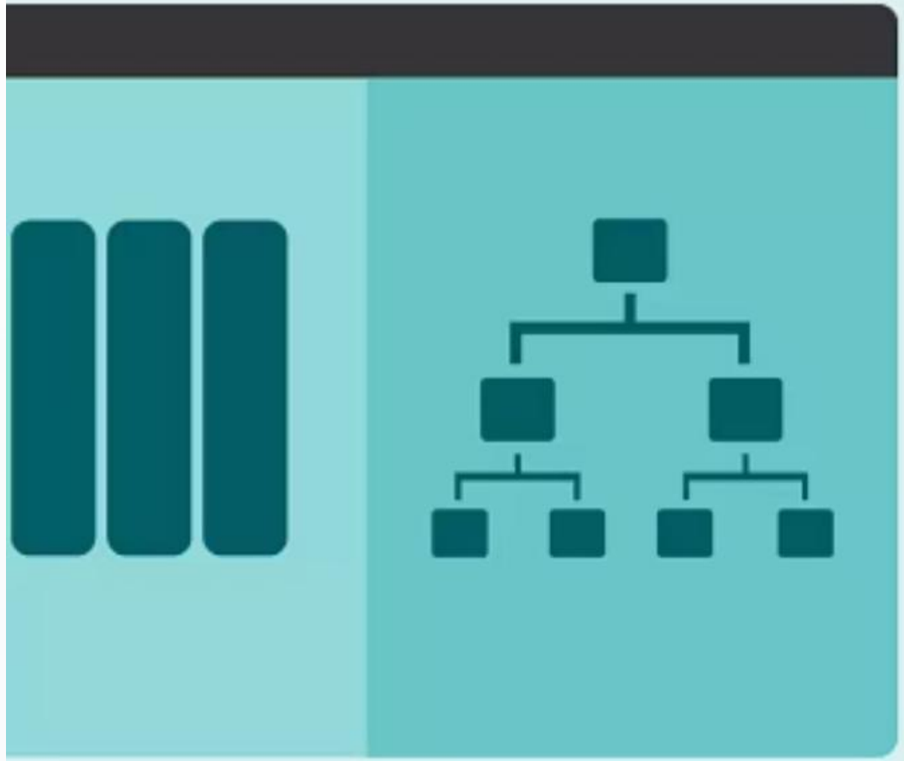
```
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```

2. Ternary Operator

```
const status = age >= 18 ? "Adult" : "Minor";
```

3. `switch-case`

```
switch (day) {
  case "Monday": console.log("Work day"); break;
  default: console.log("Weekend");
}
```



Loops in JavaScript

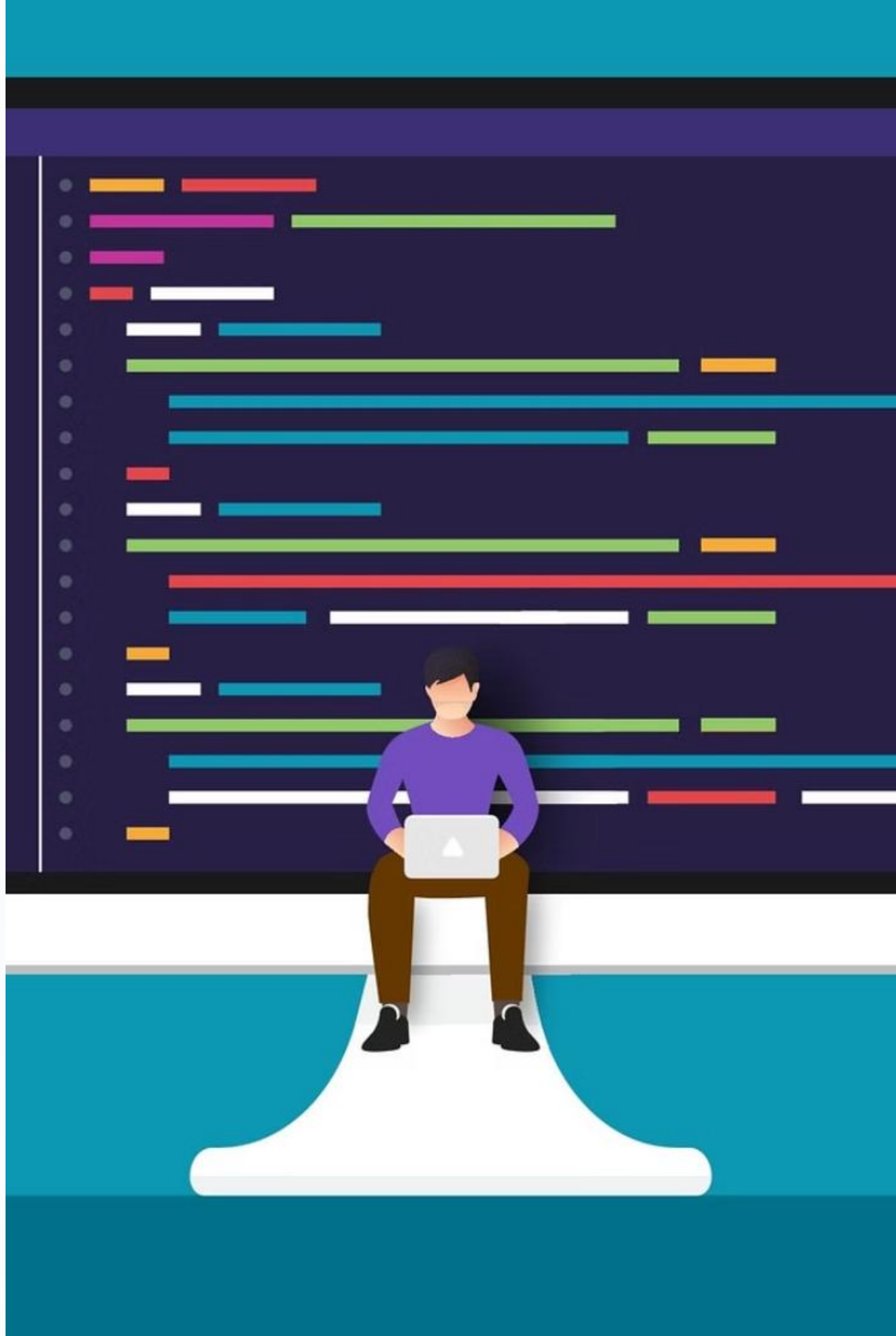
Loop Type	Use Case	Example
`for`	Known iterations	`for (let i = 0; i < 5; i++)`
`while`	Unknown iterations	`while (x < 10)`
`do-while`	Runs at least once	`do { ... } while (x < 10)`
`for...of`	Arrays/Strings	`for (let item of array)`
`for...in`	Object properties	`for (let key in obj)`

Control Flow Keywords

- `break` → Exit loop/switch
- `continue` → Skip to next iteration
- `return` → Exit function (with/without value)
- `throw` → Raise an error

Example:

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break; // Stops at 5
  if (i % 2 === 0) continue; // Skips even numbers
}
```



Error Handling (`try-catch-finally`)

```
try {
  riskyOperation();
} catch (error) {
  console.error("Error:", error.message);
} finally {
  console.log("Runs always");
}
```

Why use it? → Prevents crashes & handles exceptions gracefully.

Best Practices

✓ **Use `switch` for multiple fixed conditions** (better readability).
✓ **Prefer `for...of` over `for` for arrays** (cleaner syntax).
✓ **Avoid deep nesting** (use early returns/guard clauses).
✓ **Always handle errors** (`try-catch` for async operations).
❌ **Avoid `==` in conditions** (use strict equality `===`).

Interview Pro Tips

🔹 **Common Questions:**

1. *"Difference between* `for...of` *and* `for...in`*?"*
2. *"When to use* `while` *vs* `do-while`*?"*
3. *"Explain* `break` *vs* `continue`*."*
4. *"How does* `switch` *differ from* `if-else`*?"*

🔹 **Pro Tip:**

- Practice **nested loops & conditions** (common in coding tests).

Q&A + Practice Challenge

```
// Write a function that prints numbers 1-100, but:
// - For multiples of 3 → "Fizz"
// - For multiples of 5 → "Buzz"
// - For both → "FizzBuzz"
function fizzBuzz() { /* Your code */ }
```

Q&A Solution:-

// past solution here

Quiz Time

1. Which loop runs at least once? → `do-while`
2. What does `continue` do? → **Skips to next iteration**
3. How to exit a loop early? → `break`
4. Is `switch` strict (===) or loose (==)? → **Strict (===)**

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! 🎉

✅ **Learned:** Conditionals, Loops, Error Handling.

📌 **Key Takeaway:** Control flow makes JS dynamic.

🚀 **Next:** Functions & Scope in JS.