9. BLOCK SCOPE & Shadowing in JS

📘 Index (Table of Contents)

❓ Questions and Answers

1. What is Block Scope in JavaScript?

Q: What does "block scope" mean in JavaScript?

A: Block scope refers to the scope created by curly braces `{}`. Variables declared with `let` and `const` inside a block are only accessible within that block.

Analogy: Think of a block as a room. If you place an item inside the room, it's only accessible within that room, not outside.

---

2. How Does Variable Shadowing Work?

Q: What is variable shadowing in JavaScript?

A: Variable shadowing occurs when a variable declared in an inner scope has the same name as a variable in an outer scope, effectively "shadowing" the outer variable within the inner scope.

Example:

```
let a = 10;
{
  let a = 20;
```

```
  console.log(a); // Logs 20
}
console.log(a); // Logs 10
```

Explanation: The inner `a` shadows the outer `a` within the block, but the outer `a` remains unaffected outside the block.

---

3. How Do `let`, `const`, and `var` Differ in Scoping?

Q: What are the differences between `let`, `const`, and `var` in terms of scoping?

A: The key differences are:

- **`let` and `const`:** Block-scoped; they are confined to the block in which they are declared.
- **`var`:** Function-scoped; it is scoped to the nearest function block, not the nearest enclosing block.

Example:

```
if (true) {
  var x = 1;
  let y = 2;
  const z = 3;
}
console.log(x); // Logs 1
console.log(y); // ReferenceError: y is not defined
console.log(z); // ReferenceError: z is not defined
```

Explanation: `x` is accessible outside the block because `var` is function-scoped, while `y` and `z` are not accessible because `let` and `const` are block-scoped.

---

4. Can `let` and `const` Variables Be Redeclared?

Q: Can variables declared with `let` and `const` be redeclared within the same scope?

A: No, redeclaring variables declared with `let` or `const` within the same scope results in a syntax error.

Example:

```
let a = 10;
let a = 20; // SyntaxError: Identifier 'a' has already been declared
```

Explanation: JavaScript does not allow redeclaring variables declared with `let` or `const` in the same scope to prevent confusion and potential bugs.

🔑 Summary and Key Takeaways

- **Block Scope:** Variables declared with `let` and `const` are confined to the block in which they are declared, unlike `var`, which is function-scoped.
- **Variable Shadowing:** An inner variable can shadow an outer variable if they have the same name within their respective scopes.
- **Redeclaration Restrictions:** Variables declared with `let` and `const` cannot be redeclared within the same scope, unlike `var`.