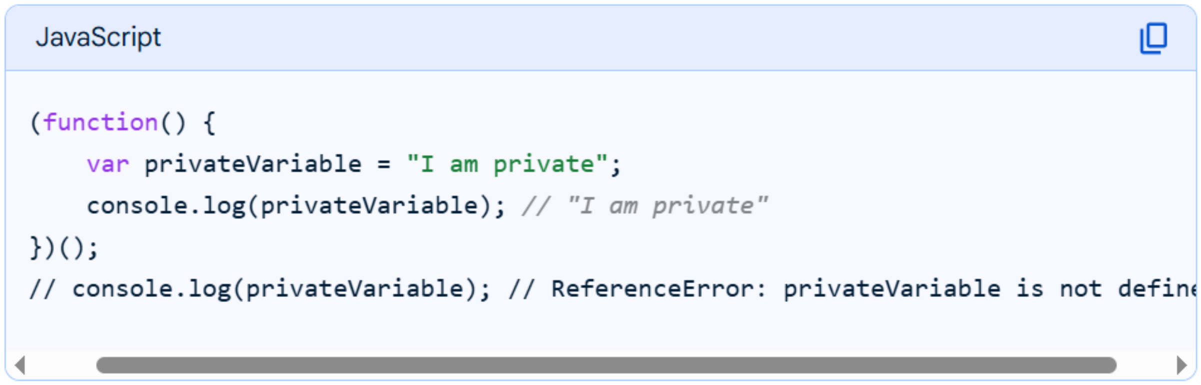
Immediately Invoked Function Expressions (IIFEs) and blocks in JavaScript

both offer ways to manage scope, but they differ in their primary purpose and behavior:

1. Immediately Invoked Function Expressions (IIFEs):

- An IIFE is a function that executes immediately after its definition.
- **Purpose:** Primarily used to create a private scope for variables and functions, preventing them from polluting the global scope. This is particularly useful for encapsulating code and avoiding naming conflicts in older JavaScript versions (pre-ES6 modules).
- **Mechanism:** It achieves scope encapsulation by creating a function execution context, where variables declared with `var`, `let`, or `const` are scoped to that function.

Syntax Example:



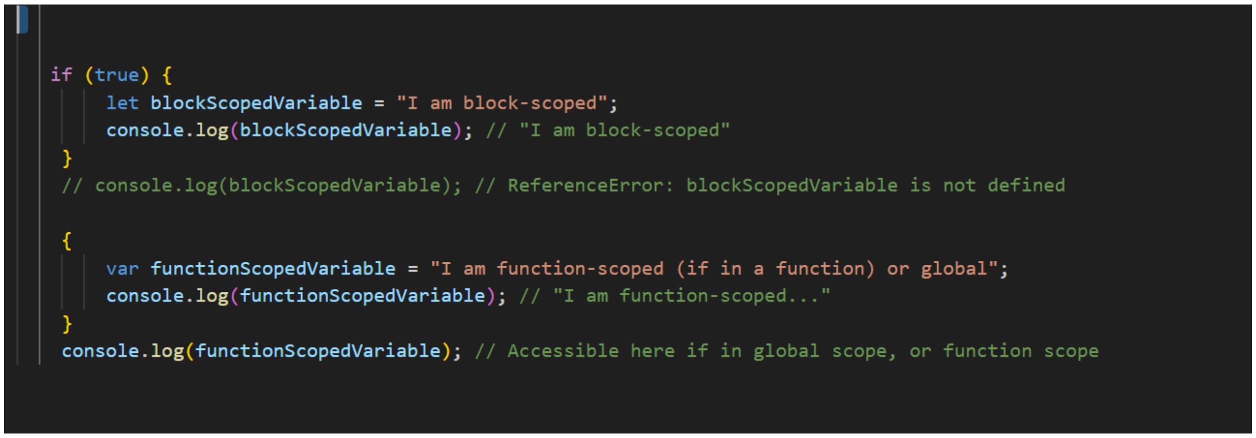
```javascript
(function() {
    var privateVariable = "I am private";
    console.log(privateVariable); // "I am private"
})();
// console.log(privateVariable); // ReferenceError: privateVariable is not defin
```

2. Blocks:

- A block is a group of zero or more statements enclosed in curly braces (`{}`).
- **Purpose:** Used to group statements together and, with the introduction of `let` and `const` in ES6, to create block-scoped variables.
- **Mechanism:** Variables declared with `let` and `const` within a block are scoped to that block, meaning they are only accessible within those curly braces. `var` declarations,

however, are not block-scoped and are hoisted to the nearest function scope or global scope.

- **Syntax Example:**



```
if (true) {
    let blockScopedVariable = "I am block-scoped";
    console.log(blockScopedVariable); // "I am block-scoped"
 }
 // console.log(blockScopedVariable); // ReferenceError: blockScopedVariable is not defined

 {
    var functionScopedVariable = "I am function-scoped (if in a function) or global";
    console.log(functionScopedVariable); // "I am function-scoped..."
 }
 console.log(functionScopedVariable); // Accessible here if in global scope, or function scope
```

Key Differences:

- **Execution:**

  IIFEs execute immediately; blocks simply group statements and define scope for `let`/`const` declarations.

- **Scope Encapsulation:**

  IIFEs provide function-level scope for `var` and block-level scope for `let`/`const`. Blocks primarily provide block-level scope for `let`/`const`.

- **Modern JavaScript:**
  With ES6 modules and `let`/`const`, the need for IIFEs for scope encapsulation has diminished, as modules provide a more structured way to manage scope and prevent global pollution. Blocks are still fundamental for control flow and defining localized variable scope.

END,,,