JavaScript to declare variables

1. `var`

- **What is it?**
  `var` was used in older versions of JavaScript to declare variables. It works in the function scope or global scope.
- **How does it work?**
  If you declare a variable using `var` and then redeclare it inside a block (like an `if` statement or a loop), the value gets overwritten.
- **Example**:

```
3    var a = 5;
4    console.log(a);
5
6    if (true) {
7        var a = 10;
8    }
9    console.log(a);
```

Output:

```
[Running] node "c:\Users\SONY\Desktop\damipro
5
10

[Done] exited with code=0 in 0.432 seconds
```

2. `let`

- **What is it?**
  `let` is used to declare block-scoped variables in JavaScript. It means the variable is only accessible inside the block where it is declared (like inside `if` statements or loops).
- **How does it work?**
  When you declare a variable using `let`, it is limited to the block in which it is declared. It does not affect the same variable outside the block.
- **Example**:

```
3    let b = 20;
4    console.log(b);
5
6    if (true) {
7        let b = 30;
8        console.log(b);
9    }
10   console.log(b);
```

Output:

```
[Running] node "c:\Users\SONY\Desktop\damipro
20
30
20

[Done] exited with code=0 in 0.387 seconds
```

3. `const`

- **What is it?**
  `const` is used to declare variables whose values cannot be changed after initialization. It creates a constant.
- **How does it work?**
  Once a variable is declared with `const`, you cannot change its value. If you try to reassign it, you'll get an error.
- **Example**:

```
1   const c = 40;
2   console.log(c);  // Output: 40
3
4   // Trying to change the value will throw an error
5   c = 50;  // Error: Assignment to constant variable
```

Now, let's look at the differences between them in a table:

Feature	var	let	const
Scope	Works in function scope or globally	Works only within the block	Works only within the block
Re-declaration	Can be redeclared in the same scope	Cannot be redeclared in the same scope	Cannot be reassigned or redeclared
Hoisting	Yes, but with undefined value	Yes, but not initialized immediately	Yes, but not initialized immediately