Java script – Variable declaration:

In JavaScript, var, let, and const are used to declare variables, but they have different behaviors in terms of scope, hoisting, and reassignability. Here's a breakdown of their differences with examples:

**1. var:**

* **Scope**: var has **function scope** or **global scope** (if declared outside a function). It is not block-scoped, meaning it is accessible outside the block where it is defined.
* **Hoisting**: var is **hoisted** to the top of its scope, but only the declaration, not the initialization.
* **Reassignable**: Variables declared with var can be reassigned.
* **Redeclarable**: var allows redeclaration within the same scope, which can lead to bugs.

**Example:**

console.log(a); // undefined (hoisted)

var a = 10;

console.log(a); // 10

* **Scope**:

if (true) {

var x = 20;

}

console.log(x); // 20 (x is accessible outside the block)

**2. let:**

* **Scope**: let has **block scope** (the variable is only available within the block it is declared).
* **Hoisting**: let is also **hoisted**, but it is not initialized until the code execution reaches the declaration. This leads to a "temporal dead zone" where the variable cannot be accessed before its declaration.
* **Reassignable**: Variables declared with let can be reassigned.
* **Redeclarable**: You cannot redeclare a let variable within the same block scope.

**Example:**

function exampleLet() {

// console.log(b); // ReferenceError: Cannot access 'b' before initialization

let b = 20;

console.log(b); // 20

}

* **Scope**:

if (true) {

let y = 30;

}

console.log(y); // ReferenceError: y is not defined (y is only available inside the block)

**3. const:**

* **Scope**: const also has **block scope** (like let).
* **Hoisting**: const is **hoisted** similarly to let
* **Reassignable**: Variables declared with const **cannot be reassigned** after initialization. This makes it useful for defining constants.
* **Redeclarable**: You cannot redeclare a const variable within the same block scope.

**Example:**

const c = 30;

console.log(c); // 30

// c = 40; // TypeError: Assignment to constant variable.

* **Scope**:

if (true) {

const z = 40;

}

console.log(z); // ReferenceError: z is not defined (z is only available inside the block)

**Summary of Key Differences:**

| **Feature** | **var** | **let** | **const** |
| --- | --- | --- | --- |
| **Scope** | Function or global scope - Not having the block scope | Block scope - you can not use let outside of the {} | Block scope – you can not used const out side of the {} |
| **Hoisting** | Hoisted (initialized as undefined) | Hoisted (not initialized)- reference error | Hoisted (not initialized)-reference error |
| **Re-assignable** | Yes | Yes | No |
| **Re-declarable** | Yes (within the same scope) | No | No |

Key Note:

**Use let for variables that will change, and const for variables whose values should remain constant (especially objects/arrays). Avoid using var in modern JavaScript due to its unpredictable behavior.**