Task 1: Variable Declaration and Assignment

Differences between var, let, and const

Feature	var	let	const
Scope	Function-scoped	Block-scoped	Block-scoped
Re-declaration	Allowed	Not allowed in same scope	Not allowed in same scope
Re-assignment	Allowed	Allowed	Not allowed
Hoisting	Hoisted (undefined)	Hoisted (TDZ)	Hoisted (TDZ)
Best Use	Legacy code	Changing values	Constants

Examples

#### var
```javascript
var x = 10;
console.log(x); // 10
var x = 20;
console.log(x); // 20
if (true) {
var y = 30;
}
console.log(y); // 30
```

#### let
```javascript
let a = 5;
console.log(a); // 5
a = 15;
console.log(a); // 15
if (true) {
let b = 25;
console.log(b);
}
// console.log(b); // Error
```

#### const
```javascript
const pi = 3.14;
console.log(pi);
// pi = 3.14159; // Error
const arr = [1,2,3];
arr.push(4);
console.log(arr); // [1,2,3,4]
```

Task 2: Hoisting

Definition

Hoisting is JavaScript’s behavior of moving variable and function declarations to the top before execution.

Behavior

var

```javascript
console.log(x); // undefined
var x = 10;
```

let

```javascript
console.log(a); // ReferenceError
let a = 5;
```

### *const*

```javascript
console.log(b); // ReferenceError
const b = 20;
```

Task 3: Scopes in JavaScript

Global Scope

```javascript
var globalVar = "I am global";
function test() {
console.log(globalVar);
}
test();
console.log(globalVar);
```

Function Scope

```javascript
function example() {
var funcVar = "inside function";
console.log(funcVar);
}
// console.log(funcVar); // Error
```

Block Scope

```javascript
if (true) {
let blockVar = "inside block";
const constVar = "also inside block";
console.log(blockVar);
}
// console.log(blockVar); // Error
```

Comparison

```javascript
if (true) {
var v = "var is function scoped";
let l = "let is block scoped";
const c = "const is block scoped";
}
console.log(v); // Works
// console.log(l); // Error
// console.log(c); // Error
```