# Hoisting Done for var, let, and const?

## ✅ Short Answer:

Yes — **all declarations (**var**,** let**, and** const**) are hoisted**.  
But they behave **differently after hoisting** due to their **initialization phase and scope**.

## 📚 What is Hoisting?

Hoisting is JavaScript’s default behavior of **moving declarations to the top of their scope** during the **memory creation phase** of execution context.

But:

| Declaration | Hoisted? | Initialized to | Temporal Dead Zone (TDZ)? | Scope |
| --- | --- | --- | --- | --- |
| var | ✅ Yes | undefined | ❌ No TDZ | Function |
| let | ✅ Yes | ❌ Not initialized | ✅ Yes | Block |
| const | ✅ Yes | ❌ Not initialized | ✅ Yes | Block |

## 🔬 Dry Run Example

console.log(a); // undefined

var a = 10;

console.log(b); // ❌ ReferenceError

let b = 20;

### 🧠 What happens behind the scenes?

### Memory Phase (before code runs):

var a = undefined;

let b = uninitialized (in TDZ);

### Execution Phase:

* console.log(a) → prints undefined ✅
* console.log(b) → ❌ throws ReferenceError: Cannot access 'b' before initialization

## ⚠️ What is Temporal Dead Zone (TDZ)?

TDZ is the time between the hoisting of a variable (let or const) and its actual declaration/initialization.

If you try to access the variable **before it's initialized**, you get a **ReferenceError**.

## ✅ Visual Summary

{

// TDZ begins

console.log(myName); // ❌ ReferenceError

let myName = "Rajeev";

// TDZ ends

}

### But with var:

{

console.log(myName); // ✅ undefined

var myName = "Rajeev";

}

## 🎯 Summry:

| Step | Approach |
| --- | --- |
| 1️⃣ | **all variables are hoisted**, but initialized differently |
| 2️⃣ | TDZ for let and const |
|  |  |
|  |  |

## What is Temporal Dead Zone (TDZ)?

TDZ is the time **between when a** let **or** const **variable is hoisted** and when it's **actually declared in the code**.

During this time, the variable **exists**, but **you are not allowed to touch it**.

## 🧒 Layman Explanation:

Imagine you enter a classroom (the execution context), and the teacher says:

🧑‍🏫

“I’ve kept your answer sheets ready on your desks. But you must **not open them until I say your name**.”

So the sheet **exists** on the desk — but if you try to touch it **before the teacher calls your name**, you’ll get scolded (⚠️ error).

That **untouchable time** = **Temporal Dead Zone**.

## 🧾 Code Example:

console.log(name); // ❌ Error: Cannot access 'name' before initialization

let name = "Rajeev";

Even though name is hoisted internally, it's in **TDZ** until the line let name = "Rajeev" runs.

## 🪜 Internal Steps – What’s Happening?

### Memory Phase (before code runs):

name = uninitialized (in TDZ)

### Execution Phase:

console.log(name); ❌ ReferenceError

Because it's in TDZ — **not safe to use yet**

## 📦 Compare with var (No TDZ):

console.log(city); // ✅ undefined

var city = "Delhi";

Here, city is also hoisted, but gets a **default value** undefined, so no error.

## 🧠 Why TDZ Exists?

* To **prevent bugs** from using variables **before they are clearly defined**
* To **enforce cleaner code** — no accidental usage