1. Hoisting
Hoisting is JavaScript's default behavior of **moving declarations to the top** of the current scope (either global or function scope) **during the compilation phase**, before code execution.

This means:

- **Variable declarations** (var, let, const) and
- **Function declarations**

...are **processed before any code is executed**.

🧠 **Example with var:**

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

var a = 10;
```

Why?
This is interpreted like:

```
var a;

console.log(a); // undefined

a = 10;
```

- Only the **declaration** is hoisted (var a;), not the assignment.

✅ **Function hoisting:**

```
greet(); // "Hello!"


function greet() {
  console.log("Hello!");
}
```

- Entire function declarations are hoisted **with their body**, so calling it before the declaration works.

❌ **But Not with let and const:**

```
console.log(b); // ❌ ReferenceError

let b = 20;
```

Here comes the concept of the...

---

2. Temporal Dead Zone (TDZ)
The Temporal Dead Zone is the time between the **hoisting of a variable (with let or const)** and its **actual declaration/initialization** in the code, during which the variable **cannot be accessed**.

Even though let and const are hoisted, they are **not initialized**. Accessing them before the declaration results in a ReferenceError.

🧠 **Example:**

console.log(x); // ❌ ReferenceError

let x = 5;

- x is hoisted but remains **uninitialized** in the TDZ until let x = 5; is reached.

✅ **After initialization:**

let y = 10;

console.log(y); // 10

Key Takeaways

- **Hoisting** moves declarations to the top, but **only var and functions** are initialized early.
- **let and const** are hoisted but trapped in the **temporal dead zone** until their line is reached.
- Avoid using variables before declaring them — even if they're technically hoisted.