Temporal dead zone

Let's dive deep into the **Temporal Dead Zone (TDZ)** in JavaScript — a key concept introduced with `let` and `const` in ES6.

🧠 What Is the Temporal Dead Zone?

The **Temporal Dead Zone (TDZ)** is the time **between** the start of a block scope and the point where a `let` or `const` variable is **declared and initialized**.

Attempting to access the variable in this "zone" throws a **`ReferenceError`**.

📌 Example:

```
{
  // TDZ starts
  console.log(x); // ❌ ReferenceError
  let x = 10;     // TDZ ends
}
```

- The block `{ ... }` starts a new scope.
- `x` is in the TDZ from the start of the block until `let x = 10` is evaluated.

🛠️ Why Does TDZ Exist?

The TDZ helps catch **common bugs** and enforces **cleaner code**:

- Prevents the use of variables **before they are initialized**
- Encourages **explicit declarations before usage**
- Replaces the confusing `undefined` behavior of `var`

🔬 Behind the Scenes

When JavaScript runs:

1. It **hoists** the `let` / `const` declaration (not initialization).
2. It marks the variable as **uninitialized**.
3. Accessing it before initialization triggers a **ReferenceError**.

❌ Access Before Declaration

```
function demo() {
  console.log(msg); // ❌ ReferenceError
  let msg = "Hello";
}
```

✅ Access After Declaration

```
function demo() {
  let msg = "Hello";
  console.log(msg); // ✅ "Hello"
}
```

`let` and `const` in TDZ (vs `var`)

Feature	`var`	`let` / `const`
Hoisted?	✅ Yes	✅ Yes
Initialized at Hoist?	✅ Yes (`undefined`)	❌ No (TDZ exists)
Access before declaration	✅ (gets `undefined`)	❌ ReferenceError

❗ TDZ with Function Parameters

```
function greet(name = message) {
  let message = "Hi";
  return `${message}, ${name}`;
}
```

```
greet();
// ❌ ReferenceError: Cannot access 'message' before initialization
```

- `message` is in the TDZ during the evaluation of the default parameter.

🧼 Best Practices

- **Always declare variables at the top** of their scope.
- Use `const` where possible to avoid accidental reassignment.
- Understand the execution order when using `let` and `const`.

✅ Summary

- **TDZ exists only with `let` and `const`**, not `var`.
- It ensures **variables aren't accessed before initialization**.
- TDZ lasts from **entering scope** to **declaration line execution**.
- Access during TDZ = **ReferenceError**.

Would you like a visual flowchart or examples combining TDZ and closures next?