✨ [Day 3] Hoisting and Temporal Dead Zone 🚀

📌 What is Hoisting in JavaScript?

Hoisting is JavaScript's default behavior of **moving declarations (not initializations)** to the top of their containing scope during the **compilation phase**, before any code is executed.

➡️ This means you can refer to variables/functions **before** they are declared — but behavior differs for `var`, `let`, and `const`.

🔄 Hoisting with Functions and Variables

1️⃣ **Variable Hoisting with `var`**

- Only the declaration is hoisted, not the assignment.

```
console.log(person); // 🔸 Output: undefined
var person = "John";
console.log(person); // ☑️ Output: John
```

💡 Explanation: Declaration is hoisted, so during execution `person` exists but is `undefined`.

2️⃣ **Function Hoisting**

- Function declarations are hoisted **completely**.

```
greet(); // ☑️ Works!

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

💡 Explanation: You can safely call functions before they are defined.

⚠️ Temporal Dead Zone (TDZ) with `let` and `const`

Variables declared with `let` and `const` are **hoisted**, but **not initialized** until the code reaches them — creating a **Temporal Dead Zone** (TDZ).

⏳ TDZ = Time between entering scope and the declaration being executed.

Example with `let`:

```
console.log(name); // ❌ ReferenceError
let name = "Alice";
console.log(name); // ☑ Output: Alice
```

Example with `const`:

```
console.log(age); // ❌ ReferenceError
const age = 30;
console.log(age); // ☑ Output: 30
```

🌐 Real-world Web Development Examples

🔄 Using `var` in a Web App (Hoisting)

```
function validateForm() {
  console.log(formName);  //  ◇ Output: undefined
  var formName = "Registration Form";
  console.log(formName);  // ☑ Output: Registration Form
}
```

🧠 Reason: `formName` is hoisted as `undefined`, then later assigned.

👆 Function Hoisting in a Web App

```
greetUser();  // ☑ Works due to hoisting

function greetUser() {
  console.log("👋 Welcome to the website!");
}
```

⏳ TDZ Scenario with `let`

```
function submitForm() {
  console.log(formStatus);  // ❌ ReferenceError
  let formStatus = "valid";
  console.log(formStatus);  // ☑ Output: valid
}
```

🧠 Reason: `formStatus` is in TDZ until declared.

🧠 Key Takeaways

- 🎯 **Hoisting** = Declarations moved to top of scope.
- ☑️ **Function declarations** are hoisted completely.
- 🩶 `var` is hoisted with `undefined`.
- 🔒 `let` & `const` are hoisted but remain in the **TDZ**.
- ⚠️ Accessing a variable in the TDZ throws a **ReferenceError**.
- ✔️ Prefer `let` and `const` for safer scoping and predictable behavior.

👨‍💻 Author

🚀 Darshan Vasani

💡 **Full-Stack Developer | Software Engineer | Mentor**

🔗 Connect with me! 🌍

🌐 **Portfolio:** dpvasani56.vercel.app
🐙 **GitHub:** github.com/dpvasani
💼 **LinkedIn:** linkedin.com/in/dpvasani56
🌳 **Linktree:** linktr.ee/dpvasani56
🎓 **Credly:** credly.com/users/dpvasani57
🐦 **Twitter:** x.com/vasanidarshan56
📢 **Topmate:** topmate.io/dpvasani56
