Function scope and local variables

Let's explore **Function Scope** and **Local Variables** in JavaScript — critical concepts to understand how variable access works inside functions.

🔷 What Is Scope?

Scope determines where variables are accessible in your code.

There are mainly **3 types** of scope in JavaScript:

1. **Global Scope**
2. **Function Scope**
3. **Block Scope** (added with `let` and `const` in ES6)

> For now, let's focus on Function Scope and how it affects local variables.

✅ Function Scope

Variables declared **inside a function** using `var`, `let`, or `const` are **only accessible inside that function**.

📌 Example:

```
function sayHello() {
  let message = "Hello, Abhi!";
  console.log(message);  // ✅ Accessible here
}

sayHello();
console.log(message);    // ❌ ReferenceError: message is not defined
```

- `message` is a **local variable**, scoped to the function `sayHello()`.

🔷 Local Variables

A **local variable** is a variable declared **within a function** and is only available inside that function.

```
function greetUser(name) {
  let greeting = `Hello, ${name}`;
  return greeting;
}

console.log(greetUser("Abhi")); // Hello, Abhi
console.log(greeting);        // ❌ Error: greeting is not defined
```

> greeting is a local variable, not accessible outside the function.

🧠 Function Scope vs Global Scope

```
let globalVar = "I am global";

function checkScope() {
  let localVar = "I am local";
  console.log(globalVar); // ✅ Accessible
  console.log(localVar);  // ✅ Accessible
}

checkScope();

console.log(globalVar); // ✅ Accessible
console.log(localVar);  // ❌ Error: localVar is not defined
```

- `globalVar` is accessible anywhere.
- `localVar` exists only inside the function.

⚠️ Variable Shadowing

You can **declare a local variable with the same name** as a global one — it will **shadow** the global variable inside the function.

```
let name = "Global Abhi";

function displayName() {
  let name = "Local Abhi";
  console.log(name); // Local Abhi
}

displayName();
console.log(name);   // Global Abhi
```

🧾 var vs let/const in Functions

All three (`var`, `let`, `const`) are function-scoped inside a function:

```
function demo() {
  var a = 1;
  let b = 2;
  const c = 3;
  console.log(a, b, c); // 1 2 3
}

demo();
console.log(a); // ❌ Error
```

> Inside a function, all variable declarations are scoped locally — but var has some quirks in block scope.

🔄 Re-declaring Local Variables

You can safely declare a variable with the same name in different functions:

```
function a() {
  let num = 1;
  console.log(num);
}

function b() {
  let num = 2;
  console.log(num);
}

a(); // 1
b(); // 2
```

They don’t interfere with each other.

✅ Summary

Concept	Description
Function Scope	Variables declared inside a function are only accessible there
Local Variables	Created within functions and destroyed when function ends
Global Variables	Declared outside all functions; accessible anywhere
Shadowing	Local variable with the same name hides the global one
var/let/const	All are function-scoped when declared inside a function

---

Would you like to start the next episode section — maybe on **Episode 26–30: Function Advanced Topics** or continue with something else?