Functions and Returning Values

A function is like a small machine that does a specific task. You can give it input (data), and it will process that input and give you back a result.

Example 1: Function Without a Return

Imagine you tell someone to add 2 and 2, and they just say the answer out loud but don't write it down.

```
function addNumbers() {
  console.log(2 + 2);
}
```

If you run `addNumbers();`, it will **show** `4`, but you **can't use the result** elsewhere.

Example 2: Function With a Return

Now imagine they **write down** the answer on a paper and give it back to you. You can use it later.

```
function addNumbers() {
  return 2 + 2;
}

let result = addNumbers();  // result now holds 4
console.log(result);  // Output: 4
```

This is useful because now `result` can be used elsewhere in your program.

Global vs Local Variables

1. Global Variables (Available Everywhere)

A **global variable** is like a celebrity. Everyone knows them.

```
var globalVar = "I am global!";

function showGlobal() {
  console.log(globalVar);  // Can use it inside the function
}

showGlobal();
console.log(globalVar);  // Can use it outside the function too
```

2. Local Variables (Only Inside a Function)

A **local variable** is like a secret you tell inside a room. People outside the room won't know it.

```
function showLocal() {
  var localVar = "I am local!";
  console.log(localVar);  // Can use it inside the function
}

showLocal();
console.log(localVar);  // ✕ ERROR! localVar is not defined outside the function
```

3. Same Name, Different Scope

If you have a global variable and a local variable with the **same name**, the function only sees the local one.

```
var myVar = "I am global!";

function testScope() {
  var myVar = "I am local!";
  console.log(myVar);  // Local version is used inside the function
```

```
}

testScope();
console.log(myVar);  // Global version is used outside the function
```

Output:

```
I am local!
I am global!
```

Even though they have the same name, they are different variables.

Why Avoid Using Global Variables in Functions?

Using global variables in functions is **not a good practice** because:

1. It makes debugging harder.
2. If multiple functions change the same global variable, unexpected bugs can happen.
3. It's harder to reuse the function in another program.

Instead, always **pass values** to a function and **return** results.

```
function add(a, b) {
  return a + b;
}

let sum = add(5, 3); // 5 + 3 = 8
console.log(sum); // Output: 8
```

This way, the function doesn't rely on any outside variables. It just works with what you give it.

Final Takeaways

1. **Functions** process data and can return results.
2. **Return values** allow you to store and use results later.
3. **Global variables** are accessible everywhere but can cause problems.
4. **Local variables** exist only inside functions and are safer to use.
5. **Avoid using global variables inside functions**—instead, pass data as arguments.
