### **Hoisting in JavaScript**

Hoisting is a JavaScript mechanism where variables and function declarations are moved ("hoisted") to the top of their scope before code execution. This means you can use functions and var variables before they are declared in the code.

### **How Hoisting Works** Before the code executes, memory is allocated for variables and functions:

### **Function Definitions are Allocated Memory** Functions are stored in memory and can be called before they are defined.

### **var is Hoisted with undefined** Variables declared with var are moved to the top but not initialized. Instead, they are assigned undefined and can be accessed before initialization.

### **let and const are Hoisted but Not Initialized** These variables exist in a Temporal Dead Zone (TDZ) until their declaration is encountered. Unlike var, trying to access them before initialization throws an error. 🚀

**Examples**

#### **✅ Function Hoisting:**

sayHello(); // Works! Function is hoisted

function sayHello() {

console.log("Hello, world!");

}

#### **✅ var Hoisting:**

console.log(x); // Undefined (not an error)

var x = 5;

console.log(x); // 5

#### **❌ let and const Hoisting (TDZ Error):**

console.log(y); // ReferenceError: Cannot access 'y' before initialization

let y = 10;  
console.log(y); // 10

### **Hoisting in Local Scope:**

Variables are hoisted **only within the scope they are declared in**.

print();

console.log(name); // ReferenceError: name is not defined

function print() {

var name = "John"; // 'name' is hoisted but remains undefined inside the function scope

}

Here, we get a **ReferenceError: name is not defined**. Even though var is hoisted, it is only available **inside the function scope**, not outside.

Now, let’s check what happens inside the function:

print();

function print() {

console.log(name); // undefined

var name = "John";

}

Since name is declared **inside** print(), it is hoisted within that function, meaning it is **undefined** when we first try to log it.

### **Best Practices**

1. Always declare variables at the top of their scope.
2. Prefer let and const over var to avoid unintended hoisting issues.
3. Declare functions before calling them for better readability.