Lexical Scoping in JavaScript

@_codevalley

Zuhaib Asif

What is Lexical Scoping?

Lexical scoping in JavaScript means that variables are accessible within the block or function where they are defined, as well as any nested blocks or functions within it. However, they are not directly accessible in outer blocks or functions.

Let's understand lexical scoping with real-life scenario

Real-Life Example:

Imagine you're on a university campus, and there are multiple buildings with different levels of access. Each building represents a scope, and the rooms inside them represent variables.

```
function campus() {
  var universityName = "My University";  // Variable in the campus scope

function building() {
   var roomNumber = 205; // Variable in the building scope
    console.log(universityName);    // This works, building can access the campus name
  }

building();                 // Entering the building
console.log(roomNumber);  // This will throw an error, campus can't directly access building's rooms
}

campus();       // Entering the campus
```

Let's Understand Previous Code...

- The **campus function** represents the outermost scope, like the entire university campus.
- The **building function** represents a scope within the campus, like a specific building on campus.
- The **universityName variable** is accessible within the campus scope.
- The **roomNumber variable** is accessible only within the building scope.

Similar to lexical scoping in JavaScript:

- The **inner scope (building)** can access variables from its containing **(outer) scope (campus)**, such as accessing the **universityName variable**.
- However, the **outer scope (campus)** cannot directly access variables from within an **inner scope (building)**, resulting in an error when trying to access the roomNumber variable.

This analogy demonstrates how lexical scoping controls the visibility and accessibility of variables based on their location within the nested structure of functions and blocks in JavaScript.