Scope

in JavaScript

/TamadaKrishna
/TamadaKrishna

Scope

- Scope refers to the visibility and accessibility of variables and functions within different parts of your code.
- It determines where variables and functions are accessible and how they are managed.
- There are several types of scope in JavaScript:

  ☞ Global Scope

  ☞ Local Scope

  ☞ Block Scope

  ☞ Function Scope

  ☞ Lexical Scope

/TamadaKrishna
/TamadaKrishna

1. Global Scope:

➥ Variables or functions declared outside of any function or block are in the global scope.
They can be accessed from anywhere in your code.

➥ Declaring a variable or function without var, let, or const implicitly creates a global variable.

Example:

Global.js

```
var globalVar = "I'm global";

function test() {
  console.log(globalVar); // Accessible here
}

test();
console.log(globalVar); // Accessible here too
```

/TamadaKrishna
/TamadaKrishna

2. Local Scope:

➥ Variables or functions declared inside a function are in the local scope of that function. They are only accessible within that function.

Example:

Local.js

```
function test() {
  var localVar = "I'm local";
  console.log(localVar); // Accessible here
}

test();
console.log(localVar); // ReferenceError: localVar is not defined
```

/TamadaKrishna
/TamadaKrishna

3. Block Scope:

➥ With the introduction of let and const in ES6, JavaScript now has block scope.

➥ Variables declared with let or const within a block (e.g., inside curly braces {}) are only accessible within that block.

Example:

Block.js

```
if (true) {
  let blockVar = "I'm block scoped";
  console.log(blockVar); // Accessible here
}

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

/TamadaKrishna
/TamadaKrishna

4. Function Scope:

➥ Before ES6, JavaScript only had function scope. Variables declared with var are function-scoped and not block-scoped.

➥ This means that a variable declared inside a block (like an if statement) is still accessible outside that block within the same function.

Example:

Function.js

```
function test() {
  if (true) {
    var functionScopedVar = "I'm function scoped";
  }
  console.log(functionScopedVar); // Accessible here
}

test();
```

/TamadaKrishna
/TamadaKrishna

5. Lexical Scope:

➥ JavaScript functions are lexically scoped, meaning they retain access to the variables from their scope when they were created.

Example:

Lexical.js

```
function outer() {
  var outerVar = "I'm from outer";

  function inner() {
    console.log(outerVar); // Accessible here
  }

  inner();
}

outer();
```

/TamadaKrishna
/TamadaKrishna