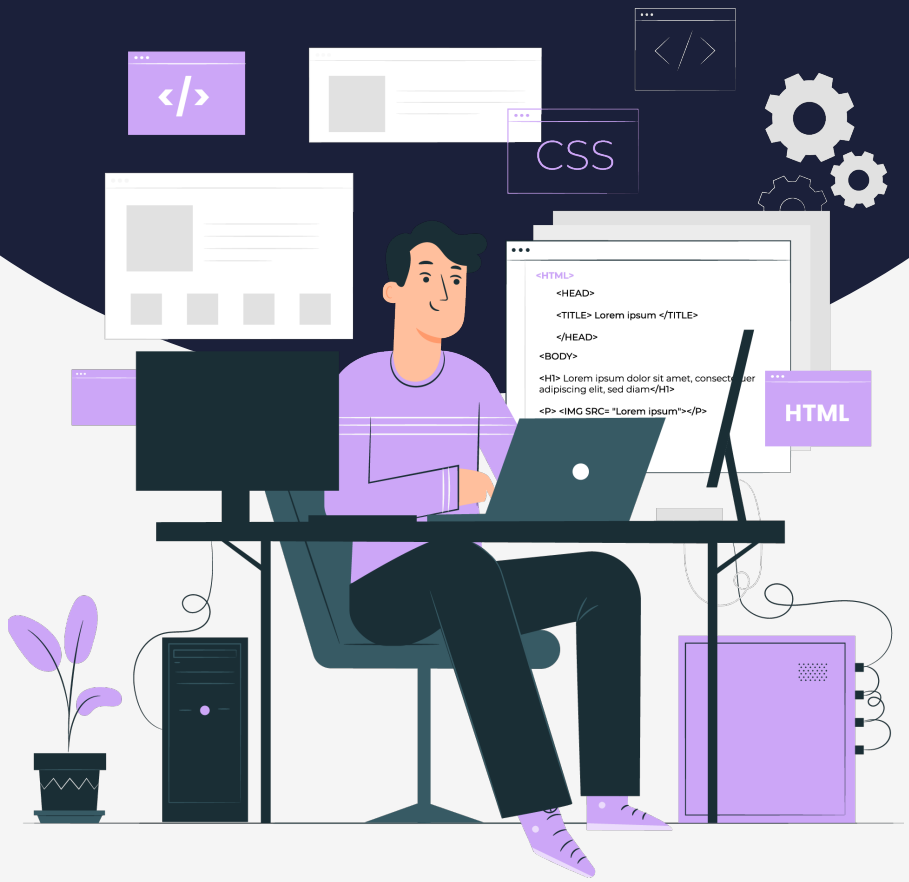
Lesson:

Scope



PW SKILLS

Topics Covered

1. What is scope?
2. Different scopes in JS
   - Global
   - Local
     -Function
     -Block
   - Module

What is Scope?

Scope refers to the accessibility of variables, functions in a particular section of code during runtime. It determines the portion of code where a variable or function can be accessed and manipulated.

Functions we will study in detail in later lessons, but to understand all scopes, just keep in mind that function is nothing but a block of statements.

Different scopes in JS

There are various scopes in JavaScript, let's study them one by one

1. Global Scope

The global scope is the outermost scope in JavaScript. Variables and functions declared in the global scope are accessible throughout the entire codebase. They are accessible from any part of the code.

Example:

```Javascript
// variable declared outside of all functions
var globalVariable = 10;

function xyz() {
  console.log(globalVaribles); // 10
}

console.log(globalVaribles); //10
```

2. Local Scope

Local scope is created when functions and variables are only accessible with any function or block, hence we have two subtypes of Scopes

- Function Scope
- Block Scope

a. Function Scope:

Function scope is created when a variable is defined inside a function. Variables declared within a function scope are accessible only within that function and not outside of it.

Each function creates its own scope, and variables declared within that function are local to that scope.

Example

```Javascript
function myFunction() {
  var localVariable = 20;
```

```
  console.log(localVariable);
}

myFunction(); // Output: I am a local variable
console.log(localVariable); //ReferenceError: localVariable is not defined
```

b. Block Scope:

Block scope was introduced in JavaScript with the introduction of the **let and const** keywords in ECMAScript 6 (ES6).

A block scope is created within any pair of curly braces {} (e.g., if statements, loops, functions). Variables declared with **let or const** are limited to the block scope and are not accessible outside of curly braces.

Example:

```Javascript
if (true) {
  var x = 10; // var has function scope
  let y = 20; // let has block scope
  const z = 30; /// const has block scope
  console.log(x, y, z); // Output: 10 20 30
}

console.log(x); // Output: 10
console.log(y); // Error: y is not defined
console.log(z); // Error: z is not defined
```

3. Module Scope:

Module scope is introduced with the advent of JavaScript modules, which allow you to encapsulate code within individual files.

Each module has its own scope, and variables and functions defined within a module are only accessible within that module by default.

Example:

```Javascript
// module1.js
var moduleVariable = 12;

function moduleFunction() {
  console.log('I am a module function');
}

export { moduleVariable, moduleFunction };
```

```Javascript
// module2.js
import { moduleVariable, moduleFunction } from './module1.js';

console.log(moduleVariable); // Output: I am a module variable
moduleFunction(); // Output: I am a module function
```

In the above example, **moduleVariable** and **moduleFunction** are only accessible within their respective module files.

These different scopes in JavaScript help in organizing and managing variables and functions, preventing naming conflicts, and controlling the visibility of identifiers within different sections of your code.