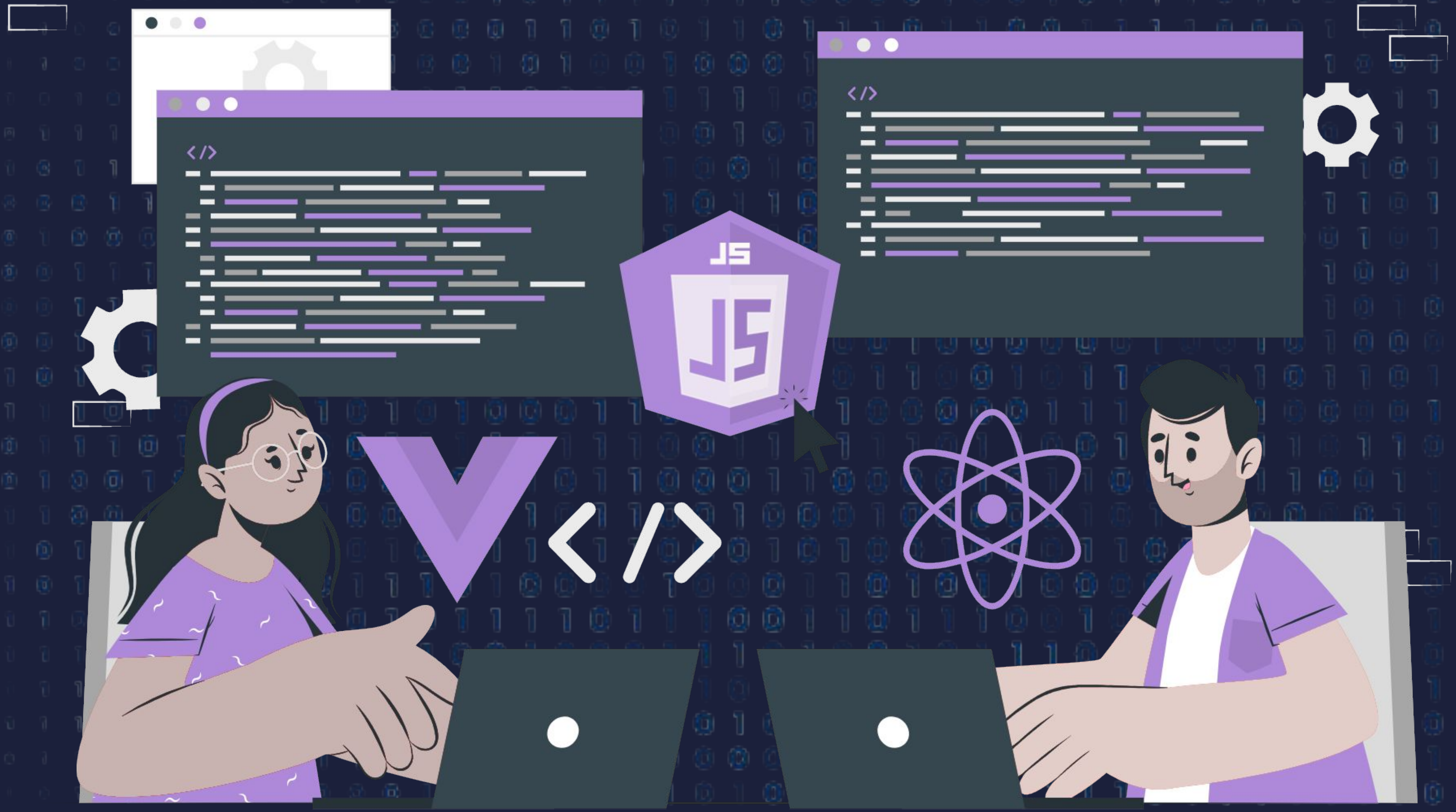
SKILLS

Scope / Lexical Scope / Block Scope



Lecture CheckList

1. Introduction to Scope.
2. Global Scope.
3. Local Scope.
4. Lexical Scope.
5. Block Scope.

Introduction to Scope

Scope in JavaScript refers to the area of a program where a particular variable or function can be accessed. In the programs we write, we declare several variables and functions. The scope determines where in your code a variable or function can be used, and which parts of the program can access it.

In JavaScript, there are two main types of scope: global scope and local scope. A variable with global scope is available throughout the entire program, while a variable with a local scope is only available within the block of code in which it was defined.

Let’s look at them one by one in this lecture.

Global Scope

Global scope is like a public space that anyone can access. A variable declared in global scope is available to all parts of the program, including functions and other blocks of code.

Variables with a global scope are usually declared outside of any functions or blocks of code, making them available to the entire program. This can be convenient when you need to use a variable in multiple parts of your program, but it can also make your code harder to manage and prone to conflicts.

One important thing to keep in mind when using global variables is that they can be accessed and modified by any part of the program, which can make it difficult to track down errors. For this reason, it's generally a good practice to minimize the use of global variables and to use local variables whenever possible.

Local Scope

Local scope, on the other hand, is like a private space that is only available within a specific block of code, such as a function. Variables declared within a function have local scope and can only be accessed within that function. This helps to prevent naming conflicts and makes your code easier to manage, but it can also make it more difficult to reuse code across different parts of your program.

Variables with the local scope are usually declared inside a function or block of code. This makes them available only to that specific function or block of code, and not to the rest of the program.

Another benefit of local scope is that it can help to prevent bugs and errors in your code. By keeping variables and functions within a specific block of code, you can help to ensure that they're used only in the way they were intended.

Lexical Scope

Global and local scopes are useful, but they can become difficult to manage in larger programs with many nested functions. To solve this, the lexical scope was introduced to help manage the complexity of larger JavaScript programs.

With the lexical scope, a function can access variables defined in its own scope, as well as variables defined in its parent scope. This means that you can create nested functions the child function or the nested functions gets access to the scope of their parent functions, making it easier to manage variables and functions in a large program.

Imagine you have a program with several nested functions. Each function has its own local scope, which can make it difficult to keep track of which variables are available where. With the lexical scope, you can define variables in the parent function's scope and make them available to all the child functions.

Block Scope

A block is a set of statements or codes enclosed within a pair of curly braces {}. A block can contain zero or more statements and can be used to group statements together and execute them as a single unit.

Block scope in JavaScript refers to the visibility or accessibility of variables or functions that are declared within a pair of curly braces, typically used to enclose statements or code blocks.

A block can be used in many places where a single statement is expected, such as in an if statement, a for loop, or a function declaration. A variable or function declared within a block can only be accessed within that block or within a nested block.

SKILLS

THANK YOU