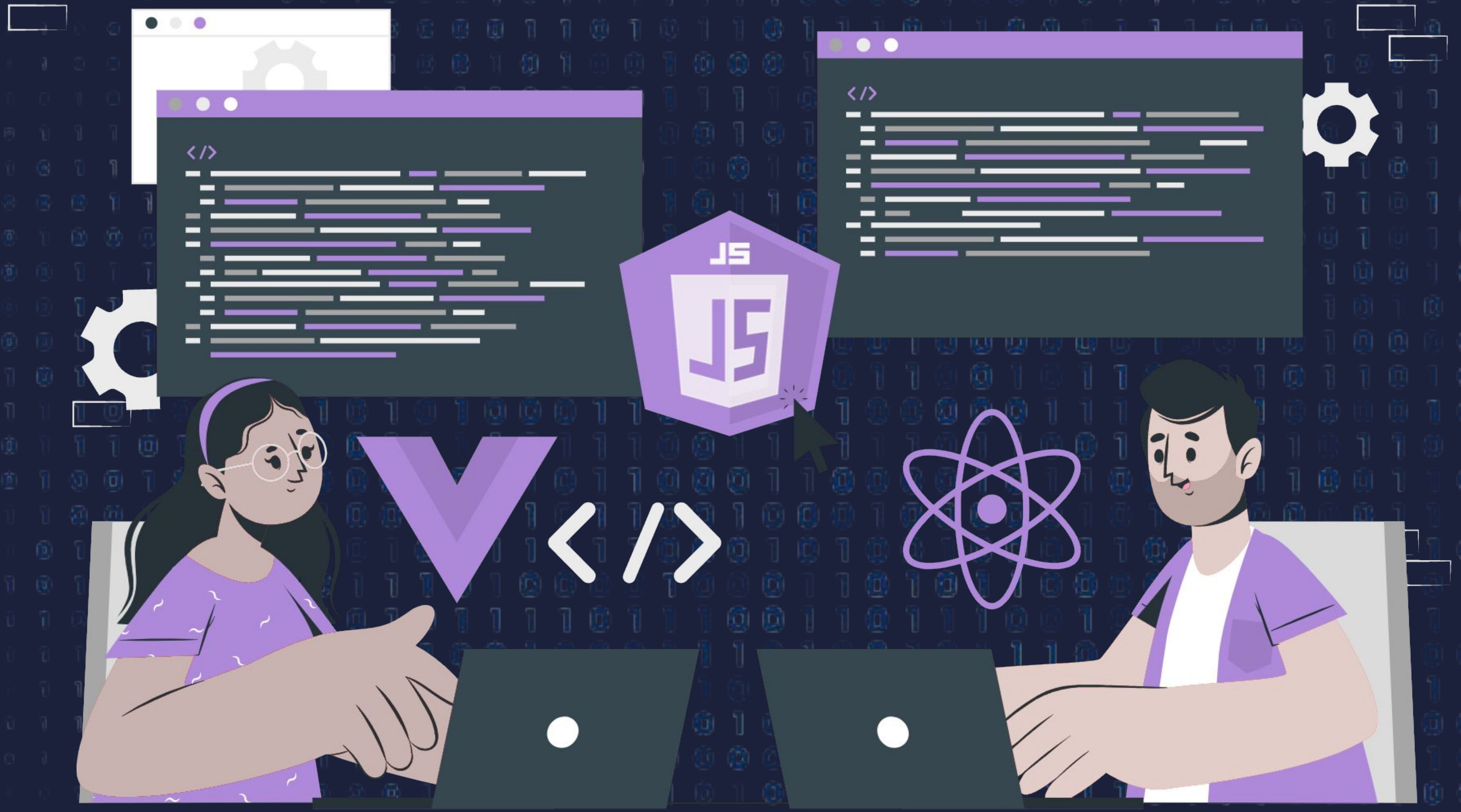
PW SKILLS

Hoisting



Lecture CheckList

1. Introduction.
2. What is Hoisting?
3. How Hoisting Works?
4. Undefined vs ReferenceError.
5. Variable hoisting.

Introduction

Have you ever written code in JavaScript and found that you can use a variable or function before it's actually declared? That's because of a nifty feature called hoisting! In simple terms, hoisting is when JavaScript moves all variable and function declarations to the top of their respective scopes before the code is executed.

SKILLS

What is Hoisting?

Before answering this question let’s look at some code samples and their outputs.

The javascript mechanism in which variables and function declarations are moved to the top of their scope before execution of the code is called Hoisting.

In simple words, Hoisting is a mechanism that makes it seem like variables and function declarations are moved to the top of the scope and lets us access variables and functions even before they are declared.

How Hoisting Works?

Javascript Engine executes code in two phases: The creation phase and the execution phase. In the creation phase, the variables are registered in the scope; this is what makes Hoisting possible, while the execution phase comes after the creation phase, where the JS engine executes the code line by line.

How Hoisting Works?

In the first phase, called the "creation" phase, the JavaScript engine scans the code and identifies all variable and function declarations. It then creates space in memory for these variables and functions but does not yet assign any values or execute any code. This is where hoisting occurs, as variable and function declarations are moved to the top of their respective scopes.

In the second phase, called the "execution" phase, the JavaScript engine assigns values to variables and executes the code. This is where the actual code is run and where variables and functions are used in calculations or other operations.

How Hoisting Works?

It's important to note that only the declarations themselves are hoisted, not the assignments or initializations. So if a variable is declared but not assigned a value until later in the code, its value will still be "undefined" until it is assigned a value.

Undefined vs ReferenceError

In JavaScript, "undefined" is a primitive data type that represents a variable that has been declared but has not been assigned a value. It is also used to indicate that a function has no return value.

In hoisting, the declarations themselves are hoisted, not the assignments or initializations. This means that if a variable is declared but not assigned a value until later in the code, its value will be "undefined" until it is assigned a value.

Undefined vs ReferenceError

On the other hand, a "ReferenceError" is a type of error that occurs when you try to reference a variable or function that has not been declared. This can happen when you misspell a variable name, or when you try to use a variable that is declared in a different scope. We have seen this in 3rd example.

Variable hoisting

Hoisting is applicable to variable and function declarations in JavaScript. In this lecture, we will be looking at variable hoisting and in the next lecture, we will look at the hoisting of function declarations.

Variable hoisting

Variable hoisting is a mechanism in JavaScript that moves variable declarations to the top of their respective scopes before the code is executed. This means that you can declare a variable anywhere in your code, and it will still be recognized as if it were declared at the top of the scope.

SKILLS

THANK YOU