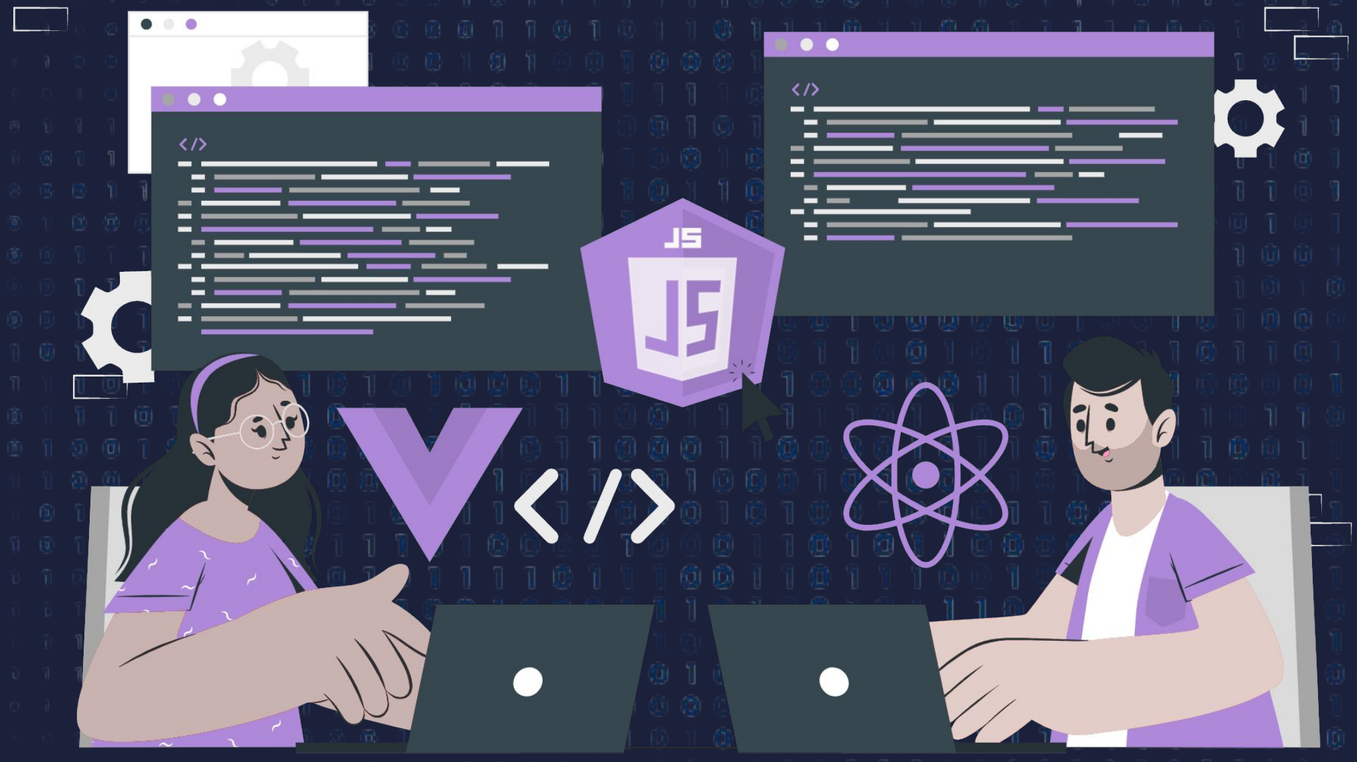
PW SKILLS

Let, var, and Const and Temporal Dead Zone



Lecture CheckList

1. Introduction.
2. Var keyword.
3. Let Keyword.
4. Const Keyword.
5. Temporal Dead Zone.

Introduction

In JavaScript, variables are used to store data values that can be accessed and manipulated throughout the program. There are three different ways to declare variables in JavaScript: var, let, and const. While var has been used traditionally to declare variables, the introduction of let and const in ES6 has added more flexibility and control over variable scope and behavior. However, when using let and const, there is a concept called the temporal dead zone.

Understanding the differences between var, let, and const, as well as the concept of the temporal dead zone, is essential to writing effective and efficient JavaScript code. In this lecture, we will be looking at var, let, and const along with temporal dead zone in depth.

var

In JavaScript, the var keyword is used to declare a variable. This keyword has been used since the early days of JavaScript and is still widely used today.

Variables declared with the var keyword have function scope, meaning that they are accessible only within the function in which they are declared or any nested functions within that function.

Variables declared with the var keyword are subject to hoisting. This means that the variable declarations are moved to the top of their scope, regardless of where the actual declaration appears in the code. However, the variable assignments are not hoisted, only the declarations.

var

Declaring the same variable with var multiple times within the same scope is allowed and does not result in a syntax error. However, this behavior can lead to unexpected results and should generally be avoided.

Function expressions declared with “var” are not hoisted to the top of their scope, which means that they are not accessible until they are assigned.

Arrow functions declared with var behave similarly to regular function expressions declared with var. They are not hoisted to the top of their scope and are not accessible until they are assigned.

Let

Let is a keyword used to declare block-scoped variables. Block-scoped variables are only accessible within the block they are declared in, which is defined by curly braces {}.

Unlike var, which allows redeclaration of variables within the same scope, let does not allow redeclaration of variables within the same scope. Attempting to declare a let variable with the same name as an existing let variable within the same scope will result in a SyntaxError.

Variables declared with let can be reassigned a new value after they have been initialized. This means that the value of the variable can change throughout the execution of the program.

Let

Unlike var variables, which are hoisted to the top of their scope and initialized with a value of undefined, let variables are not hoisted to the top of their scope. Instead, they are only accessible after they have been declared.

Like regular let variables, function expressions declared with let are also not hoisted to the top of their scope. This means that the function expression is only accessible after it has been declared.

Similar to regular let variables, arrow functions declared with let are also not hoisted to the top of their scope. This means that the arrow function is only accessible after it has been declared.

const

Const is another keyword used for declaring variables. Unlike let and var, variables declared with const cannot be reassigned to a new value after they have been initialized.

However, the value they point to can still be mutated if it is a mutable data type, such as an object or an array.

let, const does not allow redeclaration of variables within the same scope. Attempting to declare a const variable with the same name as an existing const variable within the same scope will result in a SyntaxError.

const

Like, let, const declarations also have block scope. This means that they are only accessible within the block in which they are defined.

In JavaScript, const variables are not hoisted to the top of their scope like var variables. This means that you cannot access a const variable before it has been declared.

const using function expressions are also not hoisted like var variables. This means that you cannot access a const variable that is declared using a function expression before it has been declared.

Variables declared with const using arrow functions are also not hoisted like var variables. This means that you cannot access a const variable that is declared using an arrow function before it has been declared.

Temporal Dead Zone

Temporal Dead Zone (TDZ) is a term used to describe the behavior of JavaScript's let and const declarations. This term came into existence in ES6 with the introduction of let and const to help developers write better code and make debugging easy, by telling them that they are accessing some data before defining it.

THANK YOU