This article discusses reasons why we should avoid using var to declare variable, also JavaScript 15 key words.

1. Reason why we should avoid using Var to declare a variable

Using the **var** keyword to declare variables in JavaScript is less common in modern coding practices, especially when compared to **let** and **const**. To address these issues, modern JavaScript introduced **let** and **const** declarations, which are block-scoped and do not exhibit hoisting behavior in the same way as **var**. These features provide better control and predictability when working with variables in JavaScript, making them preferred choices in most scenarios.

There are several reasons why you might want to avoid using **var**:

Function Scope: Variables declared with **var** are function-scoped, which means they are not block-scoped like **let** and **const**. This can lead to unintended variable hoisting and scope-related bugs. Variables declared with **var** are accessible throughout the entire function, even outside the block they are declared in.

Hoisting: Variables declared with **var** are hoisted to the top of their containing function or global scope. This means that even if you declare a variable inside a block of code, it gets moved to the top of the function during execution. This behavior can lead to confusion and unexpected results.

Lack of Block Scope: As mentioned earlier, variables declared with **var** do not have block scope. This can make it challenging to create well-encapsulated code because variables declared with **var** can leak their values outside of the intended block.

1. Here are 15 of the most commonly used keywords in JavaScript, These keywords are essential for defining variables, controlling program flow, and creating functions in JavaScript.

**var**: Used to declare variables. However, **var** has largely been replaced by **let** and **const** in modern JavaScript.

**let**: Declares a block-scoped variable that can be reassigned.

**const**: Declares a block-scoped variable that cannot be reassigned once a value is assigned to it.

**function**: Declares a function, a reusable block of code.

**if**: Starts a conditional statement that executes code based on a condition.

**else**: Specifies an alternative code block to execute if the condition in an **if** statement is false.

**for**: Initiates a loop that repeats code a specified number of times.

**while**: Creates a loop that continues executing code while a specified condition is true.

**switch**: Used for a multi-way branch, allowing you to execute different code blocks based on different conditions.

**return**: Exits a function and specifies a value to be returned to the calling code.

**break**: Terminates a loop or a **switch** statement prematurely.

**continue**: Skips the current iteration of a loop and proceeds to the next one.

**typeof**: Returns a string indicating the data type of a variable or expression.

**null**: Represents the intentional absence of any object value or no value at all.

**undefined**: Represents a variable that has been declared but hasn't been assigned a value.