Difference between var, let and const keywords in JavaScript

In [JavaScript](https://www.geeksforgeeks.org/javascript-tutorial/), users can declare a variable using 3 keywords that are [var, let, and const](https://www.geeksforgeeks.org/how-to-declare-variables-in-different-ways-in-javascript/).

**var keyword :**  The *var* is the oldest keyword to declare a variable in JavaScript.

**Scope:**Global scoped or function scoped.

The scope of the*var* keyword is the global or function scope. It means variables defined outside the function can be accessed globally, and variables defined inside a particular function can be accessed within the function.

**Example 1:**Variable ‘a’ is declared globally. So, the scope of the variable ‘a’ is global, and it can be accessible everywhere in the program.

 var a = 10

 function func(){

      console.log(a) // 10

 }

 func();

 console.log(a); // 10

**Example 2:**The variable ‘a’ is declared inside the function. If the user tries to access it outside the function, it will display the error.

   function func() {

        // It can be accessible any

        // where within this function

        var a = 10;

        console.log(a) // 10

    }

    func();

    // A cannot be accessible

    // outside of function

    console.log(a); // ReferenceError: a is not defined

**Example 3:**User can re-declare variable using*var* and user can update *var* variable.

  var a = 10

    // User can re-declare

    // variable using var

    var a = 8

    // User can update var variable

    a = 7

console.log(a) // 7

**Example 4:**If users use the var variable before the declaration, it initializes with the *undefined*value. The output is shown in the console.

console.log(a); // undefined

    var a = 10;

|  |
| --- |
| [**let**](https://www.geeksforgeeks.org/javascript-let/)**keyword :** The *let*keyword is an improved version of the *var* keyword.  The scope of a *let*variable is only block scoped. It can’t be accessible outside  the particular block { }.  **Example 1:**    let a = 10;      function func() {          let b = 9          console.log(b); // 9          console.log(a); // 10      }      func();  **Example 2:**The code returns an error because we are accessing the *let* variable outside the function  block.   let a = 10;      function func() {          if (true) {              let b = 9               console.log(b); // 9          }            console.log(b); // ReferenceError: b is not defined      }      func()        console.log(a) // 10  **Example 3:**Users cannot re-declare the variable defined with the *let*keyword but  can update it.   let a = 10        // It is not allowed      let a = 18  console.log(a) // Uncaught SyntaxError: Identifier 'a' has already been declared  **Example 4:**Users can declare the variable with the same name in different blocks  using the *let* keyword.   let a = 10    if (true) {      let a=9      console.log(a) // 9    }  console.log(a) // 10  **Example 5:**If users use the *let* variable before the declaration, it does not initialize  with *undefined* just like a*var* variable and return an error.  console.log(a); //Uncaught ReferenceError: Cannot access 'a' before initialization      let a = 10;  [**const**](https://www.geeksforgeeks.org/javascript-const/)**keyword:** The *const* keyword has all the properties that are the same as the *let*  keyword, except the user cannot update it.  **Scope:**[block scoped:](https://www.geeksforgeeks.org/javascript-es2015-block-scoping/) When users declare a *const* variable, they need to initialize it,  otherwise, it returns an error. The user cannot update the *const*variable once it is  declared.  **Example 1:**We are changing the value of the const variable so that it returns an error.  const a = 10;      function func() {          a = 9          console.log(a) // TypeError:Assignment to constant variable.      }      func();  **Example 2:**Users cannot change the properties of the*const*object, but they can  change the value of properties of the *const*object.   const a = {          prop1: 10,          prop2: 9      }        // It is allowed      a.prop1 = 3 //{prop1: 3,prop2: 9}        // It is not allowed      a = {          b: 10,          prop2: 9      }  **Differences between var, let, and const** |
|  |
|  |
|  |