**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/).so, we will see the differences between the var, let, and const keywords. We will discuss the scope and other required concepts about each keyword.

**var keyword in JavaScript:** 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. The output shown is in the console.

* Javascript

|  |
| --- |
| <script>  **var** a = 10  **function** f(){              console.log(a)          }      f();      console.log(a);  </script> |

**Output:**

10

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. Users can declare the 2 variables with the same name using the *var* keyword. Also, the user can reassign the value into the *var* variable. The output shown in the console.

* Javascript

|  |
| --- |
| <script>  **function** f() {            // It can be accessible any          // where within this function  **var** a = 10;          console.log(a)      }      f();     // A cannot be accessible     // outside of function      console.log(a);  </script> |

**Output:**

10

ReferenceError: a is not defined

**Example 3:**User can re-declare variable using*var* and user can update *var* variable. The output is shown in the console.

* Javascript

|  |
| --- |
| <script>  **var** a = 10        // User can re-declare      // variable using var  **var** a = 8        // User can update var variable      a = 7  </script> |

**Output:**

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.

* Javascript

|  |
| --- |
| <script>      console.log(a);  **var** a = 10;  <script> |

**Output:**

undefined

[**let**](https://www.geeksforgeeks.org/javascript-let/)**keyword in JavaScript:** The *let*keyword is an improved version of the *var* keyword.

**Scope:**[**block scoped:**](https://www.geeksforgeeks.org/javascript-es2015-block-scoping/) The scope of a *let*variable is only block scoped. It can’t be accessible outside the particular block ({block}). Let’s see the below example.

**Example 1:**The output is shown in the console.

* Javascript

|  |
| --- |
| <script>      let a = 10;  **function** f() {          let b = 9          console.log(b);          console.log(a);      }      f();  </script> |

**Output:**

9

10

**Example 2:**The code returns an error because we are accessing the *let* variable outside the function block. The output is shown in the console.

* Javascript

|  |
| --- |
| <script>      let a = 10;  **function** f() {  **if** (**true**) {              let b = 9                // It prints 9              console.log(b);          }            // It gives error as it          // defined in if block      }         console.log(b);      f()        // It prints 10      console.log(a)  </script> |

**Output:**

9

ReferenceError: b is not defined

**Example 3:**Users cannot re-declare the variable defined with the *let*keyword but can update it.

* Javascript

|  |
| --- |
| <script>        let a = 10        // It is not allowed      let a = 10        // It is allowed      a = 10  </script> |

**Output:**

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.

* Javascript

|  |
| --- |
| <script>    let a = 10  **if** (**true**) {      let a=9      console.log(a) // It prints 9    }    console.log(a) // It prints 10  </script> |

**Output:**

9

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.

* Javascript

|  |
| --- |
| <script>      console.log(a);      let a = 10;  </script> |

**Output:**

Uncaught ReferenceError: Cannot access 'a' before initialization

[**const**](https://www.geeksforgeeks.org/javascript-const/)**keyword in JavaScript:** 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. The output is shown in the console.

* Javascript

|  |
| --- |
| <script>      const a = 10;  **function** f() {          a = 9          console.log(a)      }      f();  </script> |

**Output:**

TypeError:Assignment to constant variable.

**Differences between var, let, and const**

|  |  |  |
| --- | --- | --- |
| **Var** | **let** | **const** |
| The scope of a *var*variable is functional scope. | The scope of a*let* variable is block scope. | The scope of a *const* variable is block scope. |
| It can be updated and re-declared into the scope. | It can be updated but cannot be re-declared into the scope. | It cannot be updated or re-declared into the scope. |
| It can be declared without initialization. | It can be declared without initialization. | It cannot be declared without initialization. |
| It can be accessed without initialization as its default value is “undefined”. | It cannot be accessed without initialization, as it returns an error. | It cannot be accessed without initialization, as it cannot be declared without initialization. |

**Note:**Sometimes, users face the problem while working with the *var*variable as they change the value of it in the particular block. So, users should use the *let* and *const* keyword to declare a variable in JavaScript.