****

**Assignment No: 02**

**Difference between var, let and const**

**Submitted By: Shehzadi Kiran**

**Reg no: sp18-mcs-007**

**Submitted To: Sir Kamran**

**Date: 11-10-2019**

# Difference between let, const and var:

## Differences Between let and var:

A var variable can be redeclared and updated. A let variable can be updated but not redeclared.

### Example:

<script>

Let number = 60;

Let number = 70;

Console.log(number); // uncaught syntax error, identifier number has already declared.

</script>

Let can be updated.

### Example:

Let number = 60;

number = 70;

Console.log(number); //70

### Scope of let:

If a let variable is declared at the global scope, then redeclare it within a block (curly brackets) as in the example below, then there will be no error in the console, but it will not actually redeclare let.

## Example:

Let number = 60;

Let topper = false;

If(number > 40){

Let topper = true;

}

// if call:

Topper

// it returns the first value

False

This is because of let topper = false and let topper = true, are actually two separate variables because they are scoped differently, even though they have the same name*.*

In the above example if we change both let variables to var, then call winner in the console it returns true because it is not inside a function.

### Example:

let number = 60;  
var topper = false;

if(points > 40) {  
var winner = true;  
}

// If I call  
winner

// It now returns  
true

## The Differences Between let and const:

1. const variables cannot be updated but let variables can be updated.

### Example:

// define the const variable  
const model = 'xyz123';

// Then try to redeclare it  
model = 'xyz1234'

// error  
Uncaught TypeError: Assignment to constant variable.

1. Create a const variable that is an object, the attributes of that object can be updated.

### Example:

// Creating my person object:  
const person = {  
 name: 'Kiran',  
 age: 23  
}

// Calling person in the console:  
person

// It returns  
{name: "Kiran", age: 23}

// redeclare the age attribute:  
person.age = 24

// call  
person

// It returns  
{name: "Kiran", **age: 24**}

1. If I then try to change an attribute of the person object in my console, it does not take the changes.

### Example:

// call:  
person

// It returns:  
{name: "Kiran", age: 23}

// However, if the object has been 'frozen' and try to change the age attribute  
person.age = 24

// It will immediately return  
24

// call  
person.age

// It returns the original age:  
23

## The Differences Between var and const:

### Scope of var:

Scope essentially means where these variables are available for use. var declarations are globally scoped or function/locally scoped. It is globally scoped when a var variable is declared outside a function. This means that any variable that is declared with var outside a function block is available for use in the whole window. var is function scoped when it is declared within a function. This means that it is available and can be accessed only within that function.

### Example:

Var model = 'xyz123';

Function newFunction(){

Var vehicle = “car”;

}

Here model is globally scoped because it exists outside a function while vehicle is locally scoped. So we can’t access the variable vehicle outside the function.

Now,

Var model = 'xyz123';

Function newFunction(){

Var vehicle = “car”;

}

Console.log(vehicle); //error, travel is not defined.

We'll get an error because travel is not available outside the function.

1. **var variables can be re-declared and updated:**

This means that we can do this within the same scope and will not get an error.

### Example:

Var model = 'xyz123';

Var model = 'abc456';

We can also write like this

Var model = 'xyz123';

model = 'abc456';

1. **Hoisting of var:**

Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.

### Example:

Console.log(vehicle)

Var vehicle = “car”;

It will generate an error if we write as:

Var vehicle;

Console.log(vehicle); // travel is undefined

vehicle = “car”;

So var variables are hoisted to the top of its scope and initialized with a value of undefined.

## CONST:

Variables declared with the const maintain constant values. const declarations share some similarities with let declarations.

1. **const declarations are block scoped:**

Like let declarations, const declarations can only be accessed within the block it was declared.

1. **const cannot be updated or re-declared:**

This means that the value of a variable declared with const remains the same within its scope. It cannot be updated or re-declared. So if we declare a variable with const, we can neither do this.

### Example:

const model = “xyz123”;

model= “abc456”; // error, Assignment to constant variable

nor this

### Example:

const model = “xyz123”;

model= “abc456”; // error, identifier city is already declared

Every const declaration therefore, must be initialized at the time of declaration.

This behavior is somehow different when it comes to objects declared with const. While a const object cannot be updated, the properties of this objects can be updated.

### Example:

Const vehicle = {

car: “BMW”,

quantity : 4

}

While we cannot do this

### Example:

Const vehicle = {

Word : “car”,

Number : “Five”

} // error, Assignment to constant variable

we can do this

### Example:

Vehicle.car= “BMW”;

This will update the value of Vehicle.car  without returning errors.

1. **Hoisting of const:**

Just like let, const declarations are hoisted to the top but are not initialized.