****

**Assignment No: 02**

**Difference between var, let and const**

**Submitted By: Amna Tabassum**

**Reg no: sp18-mcs-001**

**Submitted To: Sir Kamran**

**Date: 11-10-2019**

**Question:**

**What are the differences between Var, let and const?**

**Var:**

Var declarations are globally scoped or locally scoped. It is globally scoped once a volt**-**ampere variable is asserted outside an operate. This means that any variable that is declared with var outside a function block is available for use in the whole window. Var is operate scoped once it's declared among an operate.

This means that it is available and can be accessed only within that function.

**Example:**

Var city = “Islamabad”;

Function newFunction(){

Var travel = “Traveling to Islamabad”;

}

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

Now,

Var city = “Islamabad”;

Function newFunction(){

Var travel = “Traveling to Islamabad”;

}

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

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

**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.

Var city = “Islamabad”;

Var city =”I like Islamabad”;

We can also write like this:

Var city = “Islamabad”;

city =”I like Islamabad”;

**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(travel);

Var travel = “Traveling to Islamabad”;

It will generate an error if we write as:

Var travel;

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

Travel = “Traveling to Islamabad”;

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

**LET:**

let is preferred for variable declaration now. It's no surprise as it comes as an improvement to the var declarations.

**let is block scoped:**

A block is chunk of code bounded by {}. A block lives in curly braces. Anything within curly braces is a block. So a variable declared in a block with the let is only available for use within that block. Let me explain this with an example.

Let city = “Islamabad”;

Let travlelingHours = 4;

If (travlelingHours > 3){

Let travel = “Traveling to Islamabad”;

Console.log(travel); //Traveling to Islamabad

}

Console.log(travel); //travel is not defined

We see that using hello outside its block(the curly braces where it was defined) returns an error. This is because let variables are block scoped.

**let can be updated but not re-declared:**

let can be updated within its scope but it cannot be re-declared within its scope.

**Example:**

Let city = “Islamabad”;

City = “Peshawar”;

But it cannot be re-declared.

**Example:**

Let city = “Islamabad”;

Let city = “Peshawar”; //error, identifier city already declared

 If the same variable is defined in different scopes, there will be no error.

Let city = “Islamabad”;

If(true)

{

Let city = “Peshawar”;

Console.log(city); //“Peshawar”

}

Console.log(city); //“Islamabad”

There is no error because both instances are treated as different variables since they have different scopes.

**Hoisting of let**  
Just like var, let declarations are hoisted to the top. Unlike var which is initialized as undefined, the let keyword is not initialized. So if you try to use a let variable before declaration, you'll get a Reference Error.

**CONST:**

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

**const declarations are block scoped:**

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

**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 city = “Islamabad”;

city = “Peshawar”; // error, Assignment to constant variable

nor this

**Example:**

const city = “Islamabad”;

const city = “Peshawar”; // 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 travel = {

Area : “Islamabad”,

Times : 4

}

While we cannot do this

Example:

Const city = {

Word : “Islamabad”,

Number : “Five”

} // error, Assignment to constant variable

we can do this

**Example:**

City.area = “Peshawar”;

This will update the value of City.area without returning errors.

**Hoisting of const:**

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

There are some more differences:

1. var declarations are globally scoped or function scoped while let and const are block scoped.
2. var variables can be updated and re-declared within its scope; let variables can be updated but not re-declared; const variables can neither be updated nor re-declared.
3. They are all hoisted to the top of their scope but while var variables are initialized with undefined, let and const variables are not initialized.
4. While var and let can be declared without being initialized, const must be initialized during declaration.