Assignment-1

Q1.Difference between let and var.

- *Var and let are both used for variable declaration in javascript.*
- *But the Difference between them is that **Var is function scoped** and **let is block scoped.***
- *The key difference between var and let is not in the syntax,but it differs in the **semantics.***

SN	var	let
1.	*The var keyword was introduced with JavaScript.*	*The let keyword was added in ES6 (ES 2015) version of JavaScript.*
2.	*It has global scope.*	*It is limited to block scope.*
3.	*It can be declared globally and can be accessed globally.*	*It can be declared globally but cannot be accessed globally.*
4.	*Variable declared with var keyword can be re-declared and updated in the same scope.* ***Example:*** *Function varname(){* *var a = 10;* *var a = 20; //a is replaced* *console.log(a);* *}* *varname();*	*Variable declared with let keyword can be updated but not re-declared.* ***Example:*** *function varname(){* *let a = 10;* *let a = 20; //SyntaxError:* *//Identifier 'a' has already been declared* *console.log(a);* *}* *varname();*
5.	*It is hoisted.* ***Example:*** *{* *console.log(c); // undefined.* *//Due to hoisting* *var c = 2;* *}*	*It is not hoisted.* ***Example:*** *{* *console.log(b); // ReferenceError:* *//b is not defined* *let b = 3;* *}*

Assignment-1

Q2.Why are null,array printed as object type in console.

Null

- *JavaScript null is a primitive type that contains a special value null.*
- *JavaScript uses the null value to represent the **intentional absence of any object value**.*
- *If you find a variable or a function that returns null, it means that the expected object couldn't be created.*

```
typeof null // "object"
```

- *The **null** value is technically a primitive, the way "object" or "number" are primitives. This would typically mean that the type of null should also be "null". However, this is not the case because of a peculiarity with the way JavaScript was first defined.*

- *In the first implementation of JavaScript, values were represented in two parts - a type tag and the actual value. There were 5 type tags that could be used, and the tag for referencing an object was **0**. The **null** value, however, was represented as the **NULL** pointer, which was **0x00** for most platforms. As a result of this similarity, null has the **0** type tag, which corresponds to an object.*

Array

```
const arr = [2,4,6,8const obj = { type: 'serviceBot', valid: true }
console.log(typeof arr)
console.log(typeof obj)
```

The result is

object
object

- *Apparently there seems to be something wrong because the array is recognized as an object and seems to be no real difference between object and array.*
- *This because in javascript **all derived data type is always a type object**.*
- *Included functions and array.*
- *In case you need to check if it's an array you can use **isArray** method of **Array**.*

Assignment-1

Undefined

- *Undefined is also a primitive value in JavaScript.*
- *A variable or an object has an undefined value when no value is assigned before using it.*
- *So we can say that undefined means lack of value or unknown value.*
- *Example: undefined*
- *var myVar;*

- *alert(myVar); // undefined*
- *In the above example, we have not assigned any value to a variable named 'myVar'. A variable 'myVar' lacks a value. So it is undefined.*

- *Undefined is a token. typeof undefined will return undefined not an object.*

By U.Naga Nandhini.