**Assignment Day-1 (14/07/2020)**

Ans -1) Various Methods in Console Function-

1. **console.log()** - Mainly used to print the output to the console. We can put any type inside the log(), be it a string, array, object, boolean etc.

Ex- console.log('abc');

console.log(1);

console.log(true);

console.log(null);

console.log(undefined);

console.log([1, 2, 3, 4]); // array inside log

console.log({a:1, b:2, c:3}); // object inside log

1. **console.error()** – It is used to log error message to the console. It is useful in testing of code. By default the error message will be highlighted with red color

Ex - console.error('This is a simple error'); .

1. **console.warn()** – It is used to log warning message to the console. By default the warning message will be highlighted with yellow color.

Ex - console.warn('This is a warning.');

1. **console.clear()** – It is used to clear the console. The console will be cleared, in case of chrome a simple overlayed text will be printed like : ‘Console was cleared’ while in firefox no message will be returned. Ex - console.clear();
2. **console.time() and console.timeEnd() -** Whenever we want to know the amount of time spend by a block or a function, we can make use of the time() and timeEnd() methods provided by the javascript console object. They take a label which must be same, and the code inside can be anything( function, object, simple console) Ex - console.time('abc');   let fun =  function(){ console.log('fun is running'); } let fun2 = function(){ console.log('fun2 is running..'); }  fun(); // calling fun(); fun2(); // calling fun2(); console.timeEnd('abc');
3. **console.table()** - This method allows us to generate a table inside a console. The input must be an array or an object which will be shown as a table. Ex - console.table({'a':1, 'b':2});
4. **console.count()** - This method is used to count the number that the function hit by this counting method. Ex - for(let i=0;i<5;i++){ console.count(i);}

Ans – 2) i) var - JavaScript variables are containers for storing data values.

Ex - <!DOCTYPE html>

<html>

<body>

<h2>JavaScript Variables</h2>

<p>In this example, x, y, and z are variables.</p>

<p id="demo"></p>

<script>

var x = 5;

var y = 6;

var z = x + y;

document.getElementById("demo").innerHTML =

"The value of z is: " + z;

</script>

</body>

</html>

ii) let - Variables declared with the let keyword can have Block Scope. Variables declared inside a block **{}** cannot be accessed from outside the block.

Ex – { let x = 2; }  
// x can NOT be used here

iii) const - Variables defined with const behave like let variables, except they cannot be reassigned.

Ex - <!DOCTYPE html>

<html>

<body>

<h2>JavaScript const</h2>

<p>You cannot change a primitive value.</p>

<p id="demo"></p>

<script>

try {

const PI = 3.141592653589793;

PI = 3.14;

}

catch (err) {

document.getElementById("demo").innerHTML = err;

}

</script>

</body>

</html>

Ans – 3) Data types in javascript-

## Primitive Data - A primitive data value is a single simple data value with no additional properties and methods. The typeof operator can return one of these primitive types:

* string
* number
* boolean
* undefined

Complex Data

The typeof operator can return one of two complex types:

* function
* object

The typeof operator returns "object" for objects, arrays, and null.

The typeof operator does not return "object" for functions.