Assignment Day 1 | 14th July 2020

Question 1 :

Explore and explain the various methods in console function Explain them.?

Ans:

- log()
- error()
- warn()
- clear()
- time()
- timeEnd()
- table()
- count()
- group()
- groupEnd()

1.Console.log()

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

// console.log() method

console.log('abc');

console.log(1);

console.log(true);

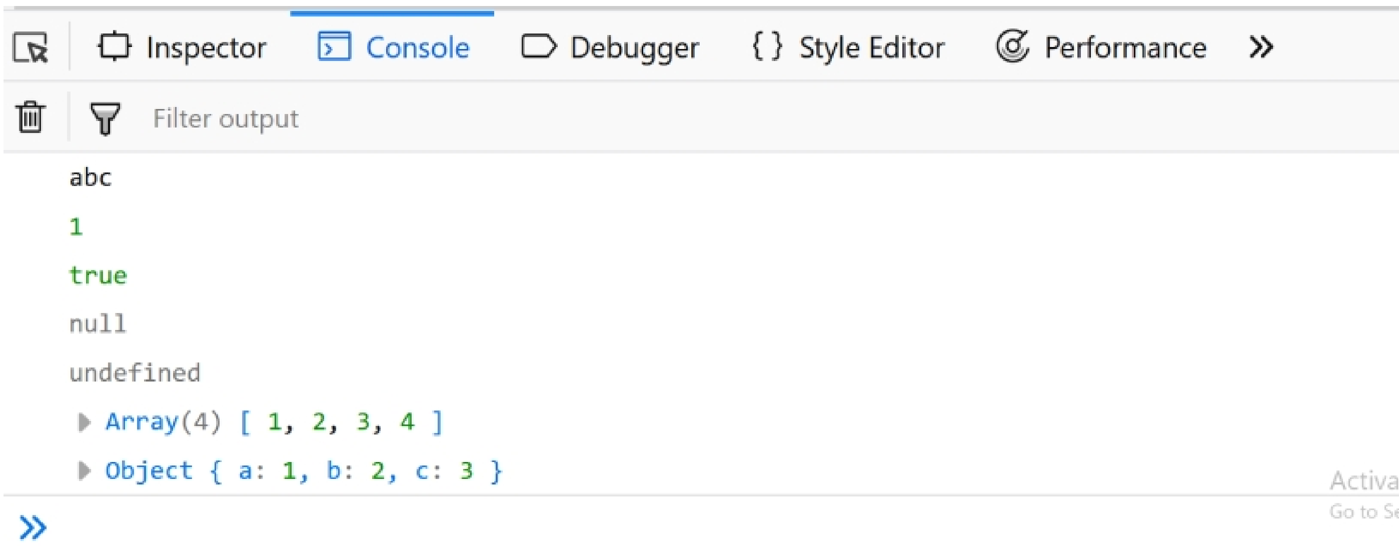
console.log(null);

console.log(undefined);

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

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

Output:



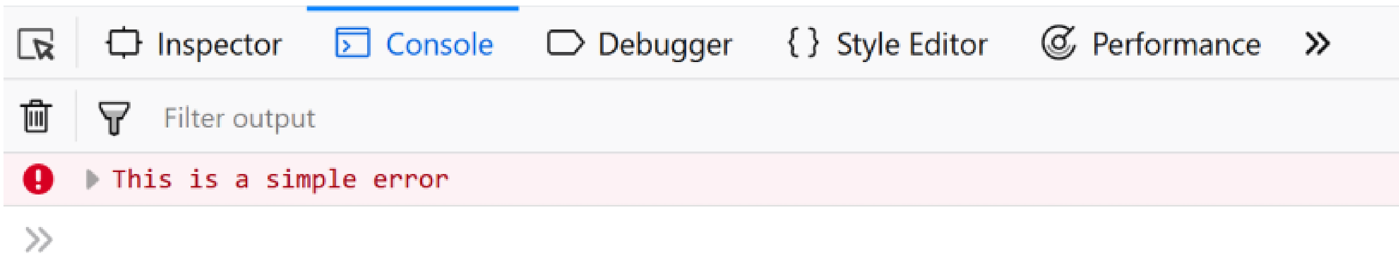
```
abc
1
true
null
undefined
▶ Array(4) [ 1, 2, 3, 4 ]
▶ Object { a: 1, b: 2, c: 3 }
```

2.Console.error()

Used to log error message to the console. Useful in testing of code. By default the error message will be highlighted with red color.// console.error() method

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

Output:



```
This is a simple error
```

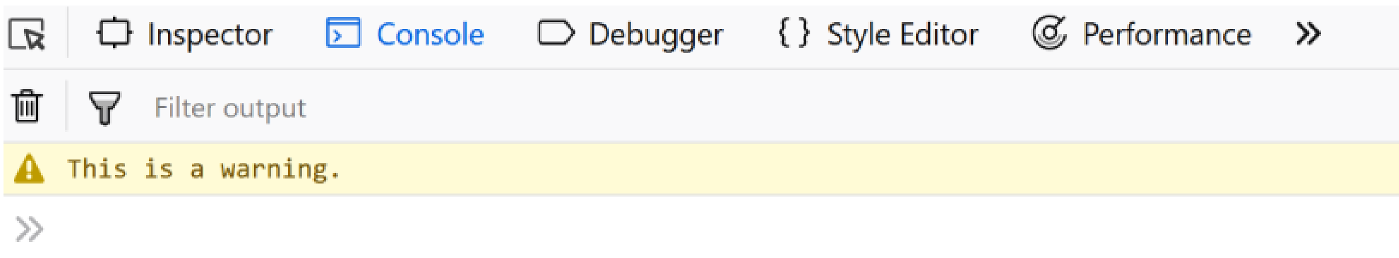
3.Console.warn()

Used to log warning message to the console. By default the warning message will be highlighted with yellow color.

// console.warn() method

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

Output:



```
This is a warning.
```

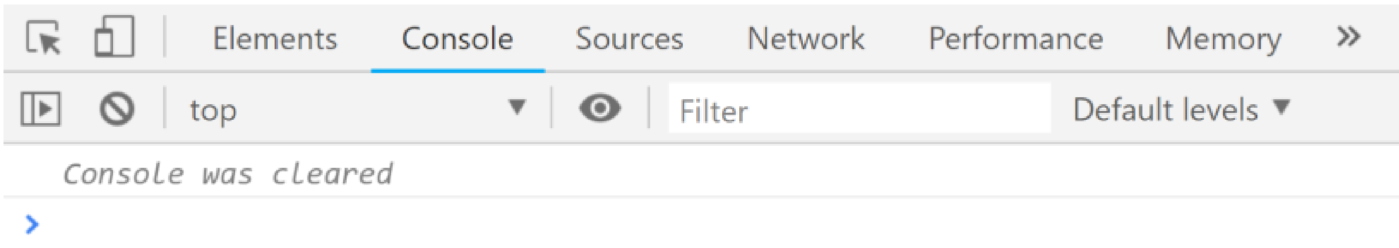
4.console.clear()

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 is returned.

// console.clear() method

console.clear();

Output:



```
Console was cleared
>
```

5.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).

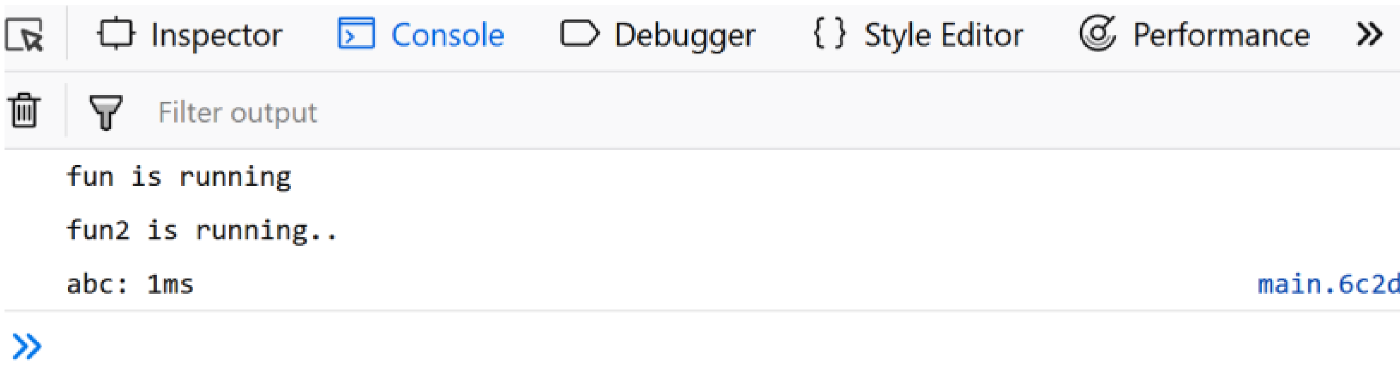
// console.time() and console.timeEnd() method

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');
```

In the above code sample, we can see that the label is 'abc' which is same for both the time() and the timeEnd() method. If we increase the amount of code inside the block defined by these methods, then the time will increase. It is also worth remembering that the time returned to the console will be in milliseconds and might be different each time we refresh the page.

Output:



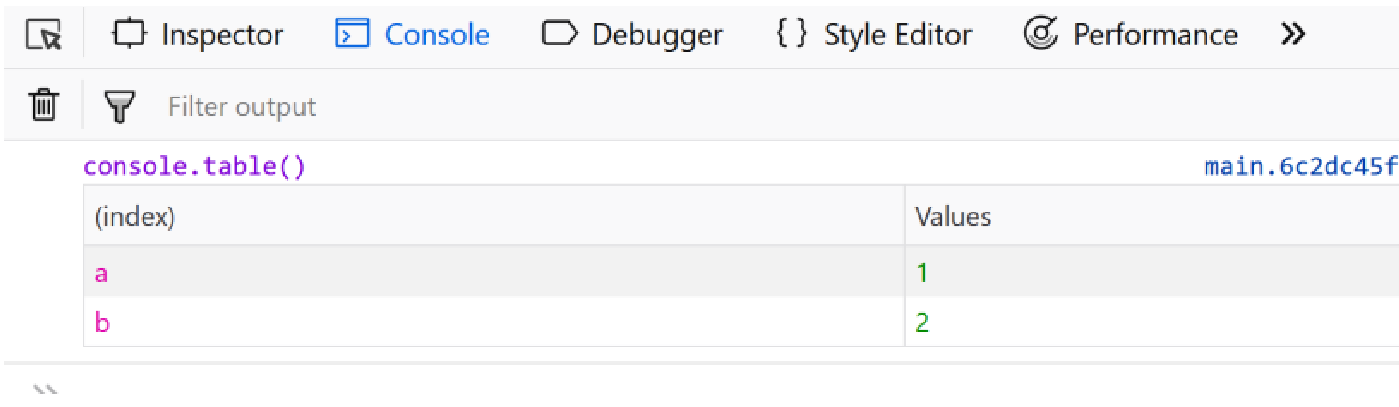
```
fun is running
fun2 is running..
abc: 1ms                                                      main.6c2d
```

6.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.

```
// console.table() method

console.table({'a':1, 'b':2});
```

Output:



```
console.table()                                              main.6c2dc45f
```

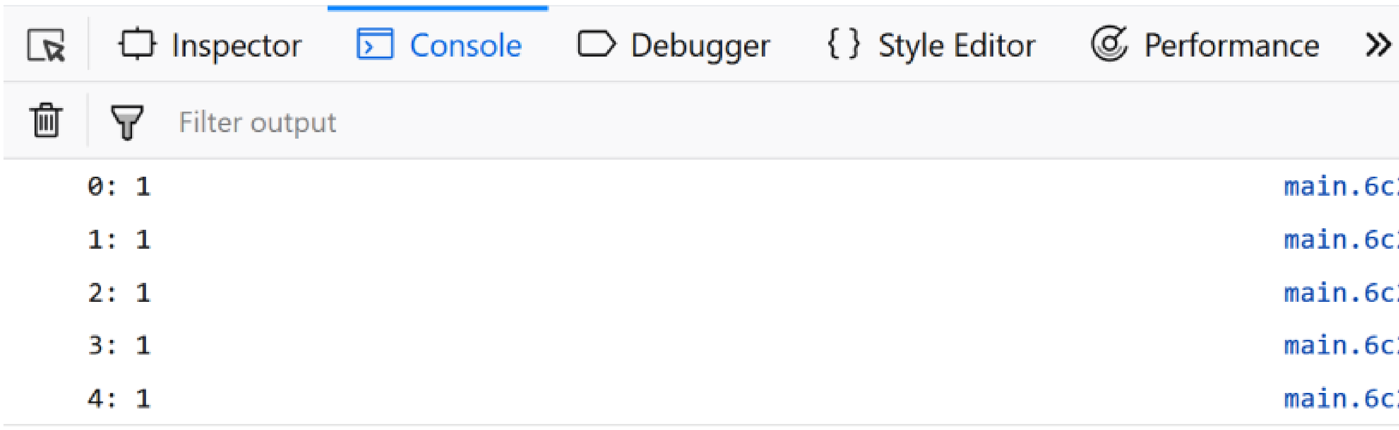
(index)	Values
a	1
b	2

7.console.count()

This method is used to count the number that the function hit by this counting method.

```
// console.count() method
for(let i=0;i<5;i++){
   console.count(i);
}
```

Output:



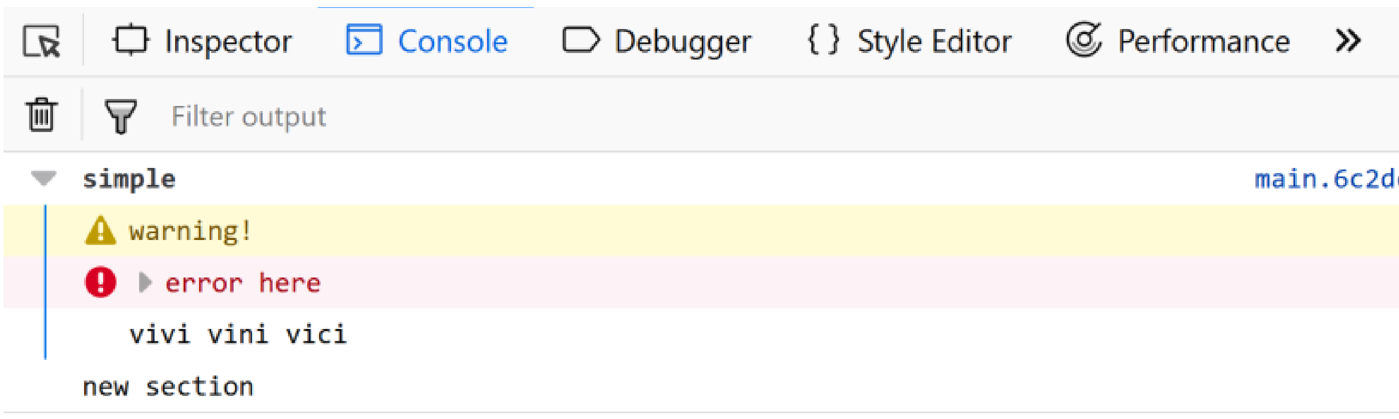
```
0: 1                                                         main.6c
1: 1                                                         main.6c
2: 1                                                         main.6c
3: 1                                                         main.6c
4: 1                                                         main.6c
```

8.console.group() and console.groupEnd()

group() and groupEnd() methods of the console object allows us to group contents in a separate block, which will be indented. Just like the time() and the timeEnd() they also accepts label, again of same value.

```
// console.group() and console.groupEnd() method

console.group('simple');

 console.warn('warning!');

console.error('error here');

  console.log('vivi vini vici');

console.groupEnd('simple');

console.log('new section');
```

Output:



```
▼ simple                                        main.6c2d
    warning!
    error here
    vivi vini vici
  new section
```

Question 2 :

Write the difference between var, let and const with code examples.?

Ans: In Javascript one can define variables using the keywords var, let or const. var a=10; let b=20; const PI=3.14;

VAR	LET	CONST

var: The scope of a variable defined with the keyword “var” is limited to the “function” within which it is defined. If it is defined outside any function, the scope of the variable is global. var is “function scoped”.
 	
Example Program:
{
  var a=10;
  console.log(a);
} //block 1
{
  a++;
  console.log(a);
} //block 2
/* Since we are using "var a=10", scope of "a" is limited to the function within which it is defined. In this case it is within the global function scope
*/
	let: The scope of a variable defined with the keyword “let” or “const” is limited to the “block” defined by curly braces i.e. {} . “let” and “const” are“block scoped”.
	
	Example Program:
	{
	 let a=10;
	 console.log(a);
	} //block 1
	{
	  a++;
	  console.log(a);
	} //block 2
	/* Since we are using "let a=10", scope of "a" is limited to block 1 and "a" is not recognized in block 2
	*/
		const: The scope of a variable defined with the keyword “const” is limited to the block defined by curly braces. However if a variable is defined with keyword const, it cannot be reassigned. “const” cannot be re-assigned to a new value.
		
		Example Program:
		{
		 const PI=3.14;
		 console.log(PI);
		} //block 1
		{
		  console.log(PI);
		} //block 2
		/* Since we are using "const PI=3.14", scope of "PI" is limited to block 1 and "PI" is not recognized in block 2
		*/

Question 3 :

Write a brief intro on available data types in Javascript.?

Ans:

Two Kinds of Data:

In JavaScript there are two different kinds of data: primitives, and objects. A primitive is simply a data type that is not an object, and has no methods.

In JS, there are six primitive data types:

1. Boolean
2. Number
3. String
4. Null
5. Undefined
6. Symbol

1.Boolean

A boolean represents only one of two values: true, or false. Think of a boolean as an on/off or a yes/no switch.

var boo1 = true;

var boo2 = false;

2.Number

There is only one type of Number in JavaScript. Numbers can be written with or without a decimal point. A number can also be +Infinity, -Infinity, and NaN (not a number).

var num1 = 32;

var num2 = +Infinity;

3.String

Strings are used for storing text. Strings must be inside of either double or single quotes. In JS, Strings are immutable (they cannot be changed).

var str1 = 'hello, it is me';

var str2 = "hello, it's me";

4.Null

Null has one value: null. It is explicitly nothing.

var nothing = null;

5.Undefined

A variable that has no value is undefined.

var testVar;

console.log(testVar); // undefined

6.Symbol

Symbols are new in ES6. A Symbol is an immutable primitive value that is unique. For the sake of brevity, that is the extent that this article will cover Symbols.

const mySymbol = Symbol('mySymbol');