**Que no.1 Explore and explain the various methods in console function & Explain them.**

**Answer:**

Methods in Console Function :

**1.console.log()** :

The **console.log()** is a function in JavaScript which is used to print any kind of variables defined before in it or to just print any message that needs to be displayed to the user.

**Syntax:**

console.log(A);

**1) Passing a number as an argument:** If the number is passed to the function console.log() then the function will display it.

|  |
| --- |
| <script>  var a = 2;  console.log(a);  </script> |

**2) Passing a string as an argument:** If the string is passed to the function console.log(), then the function will display it.

|  |
| --- |
| <script>  var str = "WelCome";  console.log(str);  </script> |

**3) Passing a char as an argument:** If the char is passed to the function console.log(), then the function will display it.

|  |
| --- |
| <script>  var ch = '2';  console.log(ch);  </script> |

**4) Passing a message as an argument:** If the message is passed to the function console.log(), then the function will display the given message.

|  |
| --- |
| <script>  console.log("Hello");  </script> |

**5) Passing a function as an argument:** If the function is passed to the function console.log(), then the function will display the value of the passed function().

|  |
| --- |
| <script>  function func() { return (5 \* 19); }  console.log(func());  </script> |

**6) Passing a number with message as an argument:** If the number is passed to the function console.log(), then the function will display it along with the given message.

|  |
| --- |
| <script>  var a = 2;  console.log("The value of a is " + a);  </script> |

**7) Passing a string with message as an argument:** If the string is passed to the function console.log(), then the function will display it along with the given message.

|  |
| --- |
| <script>  var str = "Welcome";  console.log("The value of str is " + str);  </script> |

**8) Passing a char with message as an argument:** If the char is passed to the function console.log(), then the function will display it along with the given message.

|  |
| --- |
| <script>  var ch = '2';  console.log("The value of ch is " + ch);  </script> |

**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'); |

**3. console.warn()**

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

**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(); |

**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();   fun2();  console.timeEnd('abc'); |

**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}); |

**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);  } |

**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'); |

**Que 2.Write the Differnce between var,let,const with code example.**

**Answer:**

var, let and const keywords are used to declare variables in Javascript. While var is the oldest keyword for declaring variables from its inception, let and const are introduced in ES6.

All three variables differ in the following cases:

1. **Assignment**

* let and var can be reassigned to a new value while const cannot be reassigned.

var a = 10;

a = 20;

//output: 20

console.log(a);

let b = 'hello';

b = 'world';

//output: 'world'

console.log(b);

const c = 'hello'

//Error: Uncaught TypeError: Assignment to constant variable.

c = 'world'

This makes const the best option for declaring values that do not change in the program, preventing reassignment.

1. **Scope**

* var is function-scoped.
* let and const are block-scoped(any code within {} braces).

1. **Hoisting**

* var is always hoisted to the top of their respective scope.
* let and const is also hoisted but will throw an error if the variable is used before the declaration. It is a little complicated and we will discuss it in a separate article dedicated to this specific topic.

**Que no.3 Write a brief intro on available data types in javascript.**

**Answer :**

The data types in javascript can be further divided as **Primitive and Object**. There is a total of 6 primitive types defined by the latest ECMAScript standard:

1. boolean
2. null
3. undefined
4. number
5. string
6. symbol(new in ECMAScript 2015)
7. **Object Type**.

## Primitive Types

In Javascript, all values except Objects are defined as immutable values (values that cannot be changed or modified).

### Boolean

Boolean represents a logical entity that can only have two values, true or false. It is useful in controlling program flow and decision making using conditional statements.

var bool = true;

if ( bool ) {

//then some action;

}

### Null

null represents absense of any value ie empty or non-existent value. null explicitly means nothing. Content of variable can be erased without deleting it, by assigning it to null.

var num = 45;

num = null; //num contains no value now

### Undefined

Undefined means variable has been declared but is not yet assigned to any value.

var exp;

//Print Output

document.getElementById("output").innerHTML = exp;

### Number

According to the ECMAScript standard, there is only one Number Type in Javascript. It serves for both integer and floating-point. The number type has three special numeric values: +Infinity, -Infinity, and NaN(not-a-number).

**Infinity and -Infinity** are special values that represent mathematical infinity and are greater than or smaller to any number respectively.

var exp = 1 / 0;

//Print Output

document.getElementById("output").innerHTML = exp;

**NaN** represents a computational error. It is a result of failed or incorrect mathematical operations.

var exp = "Hello" / 45;

//Print Output

document.getElementById("output").innerHTML = exp;

Any further operations on NaN will give NaN

var exp1 = "Hello" / 2;

var exp2 = exp1 + 3;

//Print Output

document.getElementById("output").innerHTML = exp2;

### String

Strings are used to represent text. Javascript strings are immutable, that means once a string is created, it is not possible to modify it.

var exp = "Welcome to LetsupGrade";

//Print Output

document.getElementById("output").innerHTML = exp;

### Symbol

Symbols are new to Javascript. Symbols are unique and immutable primitive values. They are used to create unique identifiers for objects (more on it in another article, as it can be explained better after discussing a few topics first).

### Object Type

Object type refers to a compound value that can be seen as a collection of properties. Properties are key/value pairs. Keys are strings (or Symbols) and values can be of any type, including other objects.

var obj = {

name: 'Cheese Burger',

cost: 5,

extra\_cheese: true

};

You would have noticed that the key extracheese is written with '' rather than using a space. Space can be used as well but then the key should be quoted in "" (double quotes like "key"), example:

var obj = {

name: 'Cheese Burger',

cost: 5,

"extra cheese": true

};