Q1} Explore and explain the various methods in console function Explain them.

Ex. console.log()console.warn().etc...

In JS, console is an object to deal with concole methods.Console is used for debugging. By using ctr+shift+k we can open console on a browser. Methods of a console

object are as follows:

1}Log

Log is a method which is used to print different string,object or boolean values. Log can print the value on console.

Example:

- console.log({a:1,b:2,c:3});
- console.log(5);
- console.log(a);
- console.log('abc');

2} Error

Error method is use to print error on console. 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 colour.

Example:

- console.error('This is error')

3}Warn

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

Example:

- console.warn(‘This is warning’);

4} 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.

Example:

- console.clear();

5} time and 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).

Example:

- 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');
```

6} 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.

Example:

- console.table({‘a’:1,’b’:2});

7} count

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

Example:

```
for(i=0;i<5;i++) {

console.count(i);

}
```

8} group and 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.

Example:

- console.group('simple');
- console.warn('warning!');
- console.error('error here');
- console.log('heyyyyyy');
- console.groupEnd('simple');
- console.log('new section');

Q2}Explore and explain difference between var let and const

with example.

1} VAR

i)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.

ii) Var is function scoped.

iii) Value can be changed.

Example:

```
function varscope() {
    var a = 'iamvar';
    console.log(a);
}
```

Output:

```
iamvar
```

2} LET

i) The scope of a variable defined with the keyword “let” or “const” is limited to the “block” defined by curly braces i.e. {} .

ii) Let is block scoped.

iii) Value can be changed.

Example:

```
function letscope(){
var a=’iamvar’;
{
    let a=’iamlet’;
    console.log(a);
}
console.log(“out of curly brackets”,a);
}
```

Output:

iamlet

iamvar

3} CONST

i) 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.

ii) Const cannot be re-assigned to a new value.

iii) Const are block scoped.

Example:

- function constuse(){

  const a=’constcantbechanged’;

  console.log(a);

  }

Output:

constcantbechanged

Q3} Write a brief intro on available datatypes in javascript.

JavaScript Data Types

JavaScript variables can hold many data types: numbers, strings, objects and more

```
var length= 16;                                    //Number
var lastname=”Bhosale”;                            //String
var x={firstname:”Aditya”, lastname=”Bhosale”};     //Object
```

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

Example:

- var x= 15 + “Perfect”;

Output: 15Perfect

- var x=“Perfect” + 15;

Output: Perfect15

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

1. var x= 15 + 9 + “Perfect”;

   Output: 24Perfect

2. var x=“Perfect”+ 15 + 9;

   Output: Perfect159

In the first example, JavaScript treats 15 and 9 as numbers, until it reaches “Perfect”.

In the second example, since the first operand is a string, all operands are treated as strings.

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

var = x; //x is undefiened.

x = 5; //x is Number.

x = “Perfect” //x is String.

JavaScript Strings

A string (or a text string) is a series of characters like "Aditya Bhosale".

Strings are written with quotes. You can use single or double quotes:

Example

- var carname1=”Creta” //Using Double Quotes.
  Output: Creta

- var carname1=’Creta’ //Using Single Quotes.
  Output:Creta

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

Example

- var answer1=”It’s alright” //Single quote inside.
  Output: It’s alright

- var answer2=”He is called ‘Perfect’” // Single quote inside.
  Output: He is called ‘Perfect’

- var answer3='He is called "Perfect"' // Double quote inside.
  Output: He is called "Perfect"

JavaScript Numbers

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals:

Example

- var x1=12.00; //Written with decimal.
  Output: 12

- var x2=12; //Written without decimal.
  Output: 12

- var x3=1.2; //Written with decimal.
  Output: 1.2

JavaScript Booleans

Booleans can only have two values: true or false

Examples:

```
var x = 5;

var y = 5;

var z = 4;

(x == y)        // Return true
(x == z)        //Return false
```

Booleans are often used in conditional testing.

JavaScript Array

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

Examples:

- var car = [“Volvo” , “BMW” , “Creta”];
  document.getElementById("demo").innerHTML = cars[0];

  Output:Volvo

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

JavaScript Objects

JavaScript objects are written with curly braces {}

Object properties are written as name:value pairs, separated by commas.

Example:

- var person = {firstname:”Aditya”, lastname:”Bhosale”,age:20};
  document.getElementById("demo").innerHTML =person.firstName + " is " + person.age + " years old.";

  Output: Aditya is 50 years old.

  The object (person) in the example above has 3 properties: firstName, lastName and age.

The typeof Operator

You can use the JavaScript `typeof` operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

Example:

- typeof “” //returns “string”
- typeof”Aditya” //returns “string”
- typeof 0 //returns “ number”
- typeof 3.14 //returns “ number”
- typeof (3+4) //returns “ number”

Undefined

In JavaScript, a variable without a value, has the value `undefined`. The type is also `undefined`.

Example:

- var car; //Value is undefined, type is undefined

  Any variable can be emptied, by setting the value to `undefined`. The type will also be `undefined`.

Empty Values

An empty value has nothing to do with `undefined`.

An empty string has both a legal value and a type.

Example:

- var car = "";   // The value is "", the typeof is "string"

Null

In JavaScript `null` is "nothing". It is supposed to be something that doesn't exist.

Unfortunately, in JavaScript, the data type of `null` is an object.

You can consider it a bug in JavaScript that `typeof null` is an object. It should be `null`.

You can empty an object by setting it to `null`:

Example

- var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
  person = null;   // Now value is null, but type is still an object

  document.getElementById("demo").innerHTML = typeof person;

  output:object

You can also empty an object by setting it to `undefined`:

Example

- var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
  person = undefined;   // Now both value and type is undefined

  document.getElementById("demo").innerHTML = typeof person;

  output:undefined

Difference Between Undefined and Null

`undefined` and `null` are equal in value but different in type:

```
typeof undefined            // undefined
typeof null                 // object

null === undefined          // false
null == undefined           // true
```

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`

Example

- typeof "John" // Returns "string"
- typeof 3.14 // Returns "number"
- typeof true // Returns "Boolean"
- typeof false // Returns "boolean"
- typeof x // Returns "undefined"(if x has no value)

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.

Example:

- typeof {name:'John', age:34} // Returns "object"
- typeof [1,2,3,4] // Returns "object" (not "array", see note below)
- typeof null // Returns "object"
- typeof function myFunc(){} // Returns "function"

The typeof operator returns "object" for arrays because in JavaScript arrays are objects.