Question 1 : Explore and explain the various methods in console function Explain them Ex. console.log() console.warn(). etc...

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

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
```

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.

Ex:

```
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.

Ex:

```
console.warn('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.

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).

Ex:

6. **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);
}
```

7. **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.

Ex:

```
// 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');
```

Question 2 : Write the difference between var, let and const with code examples.

```
var a=10;
let b=20;
const PI=3.14;
```

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”.

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”.

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. However it CAN be mutated.

Question 3 : Write a brief intro on available data types in Javascript.

There are six basic data types in JavaScript which can be divided into three main categories: primitive (or primary), composite (or reference), and special data types. String, Number, and Boolean are primitive data types. Object, Array, and Function (which are all types of objects) are composite data types. Whereas Undefined and Null are special data types.

Primitive data types can hold only one value at a time, whereas composite data types can hold collections of values and more complex entities.

The String Data Type

The string data type is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters

Example

```
var a = 'Hi there!';  // using single quotes
var b = "Hi there!";  // using double quotes
```

The Number Data Type

The number data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation

Example

```
var a = 25;         // integer
var b = 80.5;       // floating-point number
var c = 4.25e+6;    // exponential notation, same as 4.25e6 or 4250000
var d = 4.25e-6;    // exponential notation, same as 0.00000425
```

The Boolean Data Type

The Boolean data type can hold only two values: `true` or `false`. It is typically used to store values like yes (`true`) or no (`false`), on (`true`) or off (`false`), etc. as demonstrated below:

Example

```
var isReading = true;   // yes, I'm reading
var isSleeping = false; // no, I'm not sleeping
```

The Undefined Data Type

The undefined data type can only have one value-the special value `undefined`. If a variable has been declared, but has not been assigned a value, has the value `undefined`.

Example

```
var a;
var b = "Hello World!"

alert(a) // Output: undefined
alert(b) // Output: Hello World!
```

The Null Data Type

This is another special data type that can have only one value-the `null` value. A `null` value means that there is no value. It is not equivalent to an empty string (`""`) or 0, it is simply nothing.

A variable can be explicitly emptied of its current contents by assigning it the `null` value.

Example

```
var a = null;
alert(a); // Output: null

var b = "Hello World!"
alert(b); // Output: Hello World!

b = null;
alert(b) // Output: null
```

The Object Data Type

The `object` is a complex data type that allows you to store collections of data.

An object contains properties, defined as a key-value pair. A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects. You'll learn more about objects in upcoming chapters.

The following example will show you the simplest way to create an object in JavaScript.

Example

```
var emptyObject = {};
var person = {"name": "Clark", "surname": "Kent", "age": "36"};

// For better reading
var car = {
```

```
    "modal": "BMW X3",
    "color": "white",
    "doors": 5
}
```

The Array Data Type

An array is a type of object used for storing multiple values in single variable. Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays. The array index starts from 0, so that the first array element is `arr[0]` not `arr[1]`.

The simplest way to create an array is by specifying the array elements as a comma-separated list enclosed by square brackets, as shown in the example below:

Example

```
var colors = ["Red", "Yellow", "Green", "Orange"];
var cities = ["London", "Paris", "New York"];

alert(colors[0]);   // Output: Red
alert(cities[2]);   // Output: New York
```

The Function Data Type

The function is callable object that executes a block of code. Since functions are objects, so it is possible to assign them to variables, as shown in the example below:

Example

```
var greeting = function(){
    return "Hello World!";
}

// Check the type of greeting variable
alert(typeof greeting) // Output: function
alert(greeting());     // Output: Hello World!
```