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

Console function is an object used in Javascript that gives access to the browser debugging console.

To open console in web browser: Ctrl+Shift+K for windows
Command +Option+K for Mac.

The following are various methods of Console:

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

❖ console.log()

console.log(print) is the one which is mainly used console.log() function. It can be of any type such as string, array, object, boolean and so on.

Example: console.log(true);

❖ Console.error()

It is used to log error message and it is purpose is of testing code. Error message will be in red color by default.

Example: //console.error method()
Console.error('This is simple errro');

❖ Console.warn()

It is used as warning message and it is highlighted by light yellow color.

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

❖ Console.clear()

It is used to clear the console. In chrome it will be displayed as console was cleared but in firefox no message is returned.

Example: console.clear();

❖ Console.time() and console.timeEnd()

It is used to know the time spent by block or a function. The label must be the same but the code inside can be anything.

❖ Console.table()

It is used to generate a table inside a console. The input must be an array or object that will be shown as a table.

❖ Console.count()

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

❖ Console.group() and console.groupEnd()

It is used to group contents in a separate block which will be indented. It uses lables with same value.

2. Write the difference between var, let and const with code example.

In JavaScript there are two kinds of scope: function-scope and block-scope.

Function-scope

```
function myFn() {
   var foo = 'peekaboo!';

   console.log(foo); // 'peekaboo!'}
console.log(foo); // ReferenceError: foo is not defined
```

Using var, variables are function-scoped because their *visibility is limited to the function*. When you try to use it outside of the function, you'll get an error.

Block-scope

```
if (true) {
   var foo = 'peekaboo!';
   let bar = 'i see u';
   const baz = 'baby blue!';

   console.log(foo); // 'peekaboo!';
   console.log(bar); // 'i see u';
   console.log(baz); // 'baby blue!';}
console.log(foo); // 'peekaboo!';console.log(bar); // ReferenceError: bar is not definedconsole.log(baz); // ReferenceError: baz is not defined.
```

This concept of scope is the most prominent distinction between the old-fashioned var and modern let/const.

var

There's other nuances between var and let/const so let's explore a few code snippets to understand them better.

var outside of a for-loop

```
// for-loopfor (var i = 0; i < 3; i++) {
   console.log(i);}
console.log(i);
```

The variable i is accessible outside of the for-loop. This is expected since variables defined with var are only inaccessible outside of a function definition.

let

In many ways let is like a cousin of var. It has a lot of similarities but differentiates in ways that makes ES2016 a more modern-feeling language.

let outside of a for-loop

```
// for-loopfor (let i = 0; i < 3; i++) {
   console.log(i);}
console.log(i);
```

Redefining let

Using let you can reassign variables, but its syntax is more strict than var.

```
function myFn() {
   let foo = 1;
   foo = 30;
   // let foo = 101;        //
   foo = 101;

   console.log(foo);    }
myFn();
console.log(foo);
```

const

The keyword const is an abbreviation for *constant*. Similar to let, it's block-scoped, however, you can't reassigned it.

```
const myBoolean = true;
if (myBoolean) {
   const turtles = [
      'leonardo',
      'donatello',
      'michaelangelo',
      'raphael'
   ];
   // turtles = turtles.concat('Shredder');    //

   console.log(turtles);}
console.log(turtles);
```

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

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc.

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100

Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

avaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression