Q1. How can you create an object in JavaScript ?

Ans.
JavaScript is a flexible object-oriented language when it comes to syntax. we will see the different ways to instantiate objects in JavaScript.
Before we proceed it is important to note that JavaScript is an object-based language based on prototypes, rather than being class-based. Because of this different basis, it can be less apparent how JavaScript allows you to create hierarchies of objects and to have an inheritance of properties and their values.
We can create an object in javascript 4 Different Ways

1) Creating object with a constructor : -

One of the easiest way to instantiate an object in JavaScript. Constructor is nothing but a function and with help of new keyword, constructor function allows to create multiple objects of same flavor as shown below:-

```
//simple function
function vehicle(name,maker,engine){
      this.name = name;
      this.maker = maker;
      this.engine = engine;
}
//new keyword to create an object
let car = new vehicle('GT','BMW','1998cc');
//property accessors
console.log(car.name);
console.log(car.maker);
console.log(car['engine']);
```

2) Using object literals:-

Literals are smaller and simpler ways to define objects.We simple define the property and values inside curly braces as shown below:

```
// Create object with object literal
let car1 = {
name1: "GT",
marker1 : "BMW",
engine1 : "1888cc"
};
car1.color = "Red"; // Add new property color
// Property accessors
console.log("Create an object with object literal");
console.log(car1.name1); // access using dot opearator(.) notation
console.log(car1['engine1']); // access using bracket [] notation
console.log(car1)
```

3) Creating object with Object.create() method:-

The Object.create() method creates a new object, using an existing object as the prototype of the newly created object.

```
// Create object using Object.create method
const info = {
college : " ",
printinfo : function(){
console.log(`My name is ${this.name} . I am from ${this.place}. I am studing $
{this.college}`);
}
};
const obj = Object.create(info);
obj.name = "Ravindra kumar kushwaha";
obj.college = "IET-DAVV";
obj.place = "Chhatarpur";
console.log("Create object using Object.create method");
obj.printinfo();
```

4) Using es6 classes:-

ES6 supports class concept like any other Statically typed or object oriented language. So, object can be created out of a class in javascript as well as shown below :-

```
// Create object using es6 classes
class Employee{
constructor(name,email,age){
this.name = name;
this.email = email;
this.age = age;
}
}
let obj1 = new Employee("Ravindra","ravindrakushwahanwg@gmail.com",21);
console.log("Create object using es6 classes");
console.log(obj1.name);
console.log(obj1);
```

Q2. How can you create an array in javaScript ?

In JavaScript, array is a single variable that is used to store different elements. It is often used when we want to store list of elements and access them by a single variable. Unlike most languages where array is a reference to the multiple variable, in JavaScript array is a single variable that stores multiple elements.

Declaration of an Array
There are basically two ways to declare an array.

```
Var array = []
var array = new array();
<script>
// create array without new kayword
function arr(){
var array = ["Ravindra","Sumit","Sonu"];
console.log(array);
}
// Create array with new keyboard
function arr1(){
var city = new Array(1,2,3,4,5);
console.log(city);
}
</script>
```

Q3 How to create cookie using javaScript ?

A cookie is an amount of information that persists between a server-side and a client-side. A web browser stores this information at the time of browsing.
A cookie contains the information as a string generally in the form of a name-value pair separated by semi-colons. It maintains the state of a user and remembers the user's information among all the web pages.
How Cookies Works?
When a user sends a request to the server, then each of that request is treated as a new request sent by the different user.
So, to recognize the old user, we need to add the cookie with the response from the server.
browser at the client-side.
Now, whenever a user sends a request to the server, the cookie is added with that request automatically. Due to the cookie, the server recognizes the users.

```
<!DOCTYPE html>
<html>
<head>
<script type = "text/javascript">
function WriteCookie() {
if( document.myform.customer.value == "" ) {
alert("Enter some value!");
return;
}
cookievalue = escape(document.myform.customer.value) + ";";
document.cookie = "name=" + cookievalue;
document.write ("Setting Cookies : " + "name=" + cookievalue );
}
</script>
</head>
<body>
<form name = "myform" action = "">
Enter name: <input type = "text" name = "customer"/>
<input type = "button" value = "Set Cookie" onclick = "WriteCookie();"/>
</form>
</body>
</html>
```

Q5. What is difference between local and session storage with example?

LocalStorage:-

Web storage can be viewed simplistically as an improvement on cookies, providing much greater storage capacity. Available size is 5MB which considerably more space to work with than a typical 4KB cookie.
The data is not sent back to the server for every HTTP request (HTML, images, JavaScript, CSS, etc) - reducing the amount of traffic between client and server.
The data stored in localStorage persists until explicitly deleted. Changes made are saved and available for all current and future visits to the site.

sessionStorage:

It is similar to localStorage.
Changes are only available per window (or tab in browsers like Chrome and Firefox). Changes made are saved and available for the current page, as well as future visits to the site on the same window. Once the window is closed, the storage is deleted
The data is available only inside the window/tab in which it was set.

The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session. The data is deleted when the user closes the browser window.
ex-

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
if(typeof(Storage) !== "undefined") {
if (sessionStorage.clickcount) {
sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
} else {
 sessionStorage.clickcount = 1;
}document.getElementById("result").innerHTML = "You have clicked the button " +
sessionStorage.clickcount + " time(s) in this session.";
} else {
document.getElementById("result").innerHTML = "Sorry, your browser does not
support web storage...";
 }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</
p>
</body>
</html>
```

With local storage, web applications can store data locally within the user's browser.
Before HTML5, application data had to be stored in cookies, included in every server request. Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
Local storage is per domain. All pages, from one domain, can store and access the same data.

Q6. What will be the output of the code below?

```
var Y = 1;
if (function F(){})
{
y += Typeof F;</span>
}
console.log(y);
```

Ans. Undefilne

Q7. what is difference between call and apply with example ?

call() Method:-

It calls the method, taking the owner object as argument. The keyword this refers to the ‘owner’ of the function or the object it belongs to. We can call a method which can be used on different objects.

Syntex :-

```
object.objectMethod.call( objectInstance, arguments )
```

Parameters:-

It accepts two parameters as mentioned above and described below:

- objectInstance:It holds the instance of an object.
- arguments:The call() method takes the comma separated arguments.

apply() Method:-

The apply() method is used to write methods, which can be used on different objects. It is different from the function call() because it takes arguments as an array.

Syntax:

```
object.objectMethod.apply(objectInstance, arrayOfArguments)
```

Parameters:-

It accepts two parameters as mentioned above and described below:

objectInstance: It holds the instance of an object.
ArrayOfArguments: The apply() method takes the array of arguments.
Difference between call() and apply() method: The only difference is call() method takes the arguments separated by comma while apply() method takes the array of arguments.

```
<!DOCTYPE html>
<html>
      <head>
            <title>call() method</title>
      </head>

      <body style = "text-align:center;">
            <h1 style = "color:green;" >
                  GeeksForGeeks
            </h1>

            <button onClick="fun()">
                  click
            </button>

            <p id="GFG"></p>

            <!-- Script to use call() method to call
                  function -->
            <script>
                  function fun() {
                  let p = {
                        fullName: function(addr1, addr2) {
                        return this.fName + " " + this.lName
                                    + ", " + addr1 + ", " + addr2;
                        }
                  }
                  let p1 = {
                        fName:"GFGfName",
                        lName: "GFGlName",
                  }
                  let x = p.fullName.call(p1, "India", "USA");
                  document.getElementById("GFG").innerHTML = x;
                  }
            </script>
      </body>
</html>
```

Q8. How to empty an Array in JavaScript?

```
<!DOCTYPE html>
<html>

<head>
<title>
JavaScript to set array empty
</title>
</head>
<body style = "text-align:center;">
<h1 style = "color:green;" >
Edunomics
</h1>
<div id = "up"></div> <br/><br/>
<button onclick="empty()">Click to Empty </button><br/><br/>
<div id = "down" style="color: green"></div>
<!-- Script to set size of array to zero -->
<script>
// Array Create and initialise
var array = [1, 2, 3, 4, 5,7,8,9,10];
var up = document.getElementById("up").innerHTML = array;
var down = document.getElementById("down").innerHTML = "length of Array =
"+ array.length;

// Empty array
function empty() {
array = [];
down = document.getElementById("down");
down.innerHTML = "length of Array = "
+ array.length;
}
</script>
</body>

</html>
```

Q9. What will be the output of the following code?

```
var X = { Foo : 1};
var Output = (function()
{
delete X.foo;
return X.foo;
}
)();
console.log(output);
```

Ans : cookie.html:24 Uncaught ReferenceError: output is not defined.

Q10. What will be the output of the following code?

```
var X = { Foo : 1};
var Output = (function()
{
delete X.foo;
return X.foo;
}
)();
console.log(output);
```

ans: Uncaught ReferenceError: output is not defined

Q11. What will be the output of the following code?

```
var Employee =
{
company: 'xyz'
}
var Emp1 = Object.create(employee);
delete Emp1.company Console.log(emp1.company);
```

Ans : Uncaught ReferenceError: employee is not defined

Q12: Name the types of function with reason?

```
function display()
{
document.writeln("Named Function");
}
display();
```

Ans : The function which has named at the time of definition is called a named function.

Q13. Name the types of function with reason?

```
var display=function()
{
document.writeln("Anonymous Function");
}
display();
```

Ans : This is the anonymous javascript function.
It is a function that has no name. These functions are declared dynamically at runtime using the function operator instead of the function declaration. The

function operator is more flexible than a function declaration. It can be easily used in the place of an expression.

Q14. If we want to return the character from a specific index which method is used, explain with example?

```
 <!DOCTYPE html>
<html>
<body>

<p>String "HELLO WORLD".</p>

<button onclick="fun()">Click</button>

<p id="demo"></p>

<script>
function fun() {
var str = "HELLO WORLD";
const i = prompt("Enter index");
if(i>= str.length){
alert("Index out of bound");
}else{
var res = str.charAt(i);
// document.getElementById("demo").innerHTML = res;
alert(res);
}
}
</script>

</body>
</html>
```