**QUE1 : How can you create an object in JavaScript?**

Ans : Creating a JavaScript Object :->

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

Define and create a single object, using an object literal.

Define and create a single object, with the keyword new.

Define an object constructor, and then create objects of the constructed type.

* Using an Object Literal

This is the easiest way to create a JavaScript Object.

Using an object literal, you both define and create an object in one statement.

An object literal is a list of name:value pairs (like age:50) inside curly braces {}.

The following example creates a new JavaScript object with four properties:

Example :

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

* Using the JavaScript Keyword new

The following example also creates a new JavaScript object with four properties:

Example :

var person = new Object();  
person.firstName = "John";  
person.lastName = "Doe";  
person.age = 50;  
person.eyeColor = "blue";

**QUE2 : How can you create an Array in JavaScript?**

**Ans :**

JavaScript arrays are used to store multiple values in a single variable.

var cars = ["Saab", "Volvo", "BMW"];

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

var array\_name = [item1, item2, ...];

**QUE3 : How to create a cookie using JavaScript?**

**Ans :**

Create a Cookie with JavaScript

JavaScript can create, read, and delete cookies with the document.cookie property.

With JavaScript, a cookie can be created like this:

document.cookie = "username=John Doe";

**QUE5 : What is the difference between Local storage & Session storage with example?**

**Ans :**

**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​

**Example :**

<script>

     sessionStorage.setItem("sessionData", "I am set in session storage.");

  localStorage.setItem("localData", "I am set in local storage.");

</script>

**QUE7 : What is the difference between Call & Apply with example?**

**Ans :**

* **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.

**Syntax:**

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.

**Example :**

var person = {  
  **fullName**: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}  
var person1 = {  
  firstName:"John",  
  lastName: "Doe"  
}  
var person2 = {  
  firstName:"Mary",  
  lastName: "Doe"  
}  
person.fullName.call(**person1**);  // Will return "John Doe"

* **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.

**Example :**

var person = {  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
}  
var person1 = {  
  firstName: "Mary",  
  lastName: "Doe"  
}  
person.fullName.apply(person1);  // Will return "Mary Doe"

**QUE8 : How to empty an Array in JavaScript?**

**Ans :**

There are multiple ways to clear/empty an array in JavaScript. You need to use them based on the context. Let us look at each of them. Assume we have an array defined as −

let arr = [1, 'test', {}, 123.43];

* Substituting with a new array −

arr = [];

This is the fastest way. This will set arr to a new array. This is perfect if you don't have any references from other places to the original arr. If you do, those references won't be updated and those places will continue to use the old array.

* Setting length prop to 0 −

arr.length = 0

This will clear the existing array by setting its length to 0. Fast solution, but this won't free up the objects in this array and may have some memory implications. In order to clean objects in array from memory, they need to be explicitly removed.

* Splice the whole array

arr.splice(0, arr.length)

This will remove all elements from the array and will actually clean the original array.