- **What is JavaScript?**

➢ JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
➢ JavaScript can update and change both HTML and CSS.
➢ JavaScript can calculate, manipulate and validate data.

- **What is the use of isNaN function?**

➤ The JavaScript isNaN() Function is used to check whether a given value is an illegal number or not. It returns true if the value is a NaN else returns false. It is different from the Number isNaN() Method.

➤ **<u>Syntax</u>**:
   isNaN(value)

➤ Parameter Values: This method accepts a single parameter as mentioned above and described below:
➤ Value: It is a required value passed in the is NaN() function.
➤ Return Value: It returns a Boolean value i.e. returns true if the value is NaN else returns false.

- **What is negative Infinity?**

➤ The negative infinity in JavaScript is a constant value that is used to represent a value that is the lowest available. This means that no other number is lesser than this value. It can be generated using a self-made function or by an arithmetic operation.

❖ Negative infinity is different from mathematical infinity in the following ways:
  ✓ Negative infinity results in -0(different from 0) when divided by any other number.
  ✓ When divided by itself or positive infinity, negative infinity return NaN
  ✓ Negative infinity, when divided by any positive number (apart from positive infinity) is negative infinity.
  ✓ Negative infinity, divided by any negative number (apart from negative infinity) is positive infinity.
  ✓ If we multiply negative infinity with NaN, we will get NaN as a result.
  ✓ The product of 0 and negative infinity is Nan.
  ✓ The product of two negative infinities is always a positive infinity.
  ✓ The product of both positive and negative infinity is always negative infinity.

- **Which company developed JavaScript?**

➢ JavaScript was invented by Brendan Eich in 1995. It was developed for Netscape 2, and became the ECMA-262 standard in 1997. After Netscape handed JavaScript over to ECMA, the Mozilla foundation continued to develop JavaScript for the Firefox browser. Mozilla's latest version was 1.8.

- **What are undeclared and undefined variables?**

➢ **<u>Undefined</u>:** It occurs when a variable has been declared but has not been assigned with any value. Undefined is not a keyword.

➢ **<u>Undeclared</u>:** It occurs when we try to access any variable that is not initialized or declared earlier using *var* or *const keyword*. If we use *'typeof'* operator to get the value of an undeclared variable, we will face the *runtime error* with return value as **"undefined"**. The scope of the undeclared variables is always global.

- **Write the code for adding new elements dynamically?**

➤ JavaScript is a very important language when it comes to learning how the browser works. Often there are times we would like to add dynamic elements/content to our web pages. This post deals with all of that.

➤ **Creation of new element:** New elements can be created in JS by using the **createElement()** method.

➤ **Syntax:**

document.createElement("*<tagName>*");

// newDiv element has been created
For Eg: **let newDiv = document.createElement("div");**

- **What is the difference between ViewState and SessionState?**

ViewState	SessionState
Maintained at page level only.	Maintained at session level.
View state can only be visible from a single page and not multiple pages.	Session state value availability is across all pages available in a user session.
It will retain values in the event of a postback operation occurring.	In session state, user data remains in the server. Data is available to user until the browser is closed or there is session expiration.
Information is stored on the client's end only.	Information is stored on the server.
Used to allow the persistence of page-instance-specific data.	Used for the persistence of user-specific data on the server's end.
ViewState values are lost/cleared when new page is loaded.	SessionState can be cleared by programmer or user or in case of timeouts.

- **What is === operator?**

➢ The strict equality (===) operator checks whether its two operands are equal, returning a Boolean result. Unlike the equality operator, the strict equality operator always considers operands of different types to be different.

❖ The strict equality operators (=== and !==) provide the IsStrictlyEqual semantic.

✓ If the operands are of different types, return false.
✓ If both operands are objects, return true only if they refer to the same object.
✓ If both operands are null or both operands are undefined, return true.
✓ If either operand is NaN, return false.
✓ Otherwise, compare the two operand's values:
  ▪ Numbers must have the same numeric values. +0 and -0 are considered to be the same value.
  ▪ Strings must have the same characters in the same order.
  ▪ Booleans must be both true or both false.

- **How can the style/class of an element be changed?**

➢ The class name is used as a selector in HTML which helps to give some value to the element attributes.The document.getElementById() method is used to return the element in the document with the "id" attribute and the "className" attribute can be used to change/append the class of the element.

➢ **<u>Syntax</u> :**

  document.getElementById('myElement').className = "myclass";

- **How to read and write a file using JavaScript?**

❖ <u>**Write operation on a file**</u>

➢ After the File System file is imported then, the writeFile() operation is called. The writeFile() method is used to write into the file in JavaScript. The syntax of this method is as follows

➢ <u>**Syntax**</u>:

   writeFile(path,inputData,callBackFunction)

➢ The writeFile() function accepts three parameters :

1.) **Path** − The first parameter is the path of the file or the name of the file into which the input data is to be written.
If there is a file already, then the contents in the file are deleted and the input which is given by the user will get updated or if the file is not present, then the file with that will be created in the given path and the input information is written into it.

2.) **inputData** − The second parameter is the input data which contains the data to be written in the file that is opened.

3.) **callBackFuntion** − The third parameter is the function which is the call back function which takes the error as the parameter and shows the fault if the write operation fails.

❖ <u>Reading from the file</u>

➢ After the File System module is imported, the reading of the file in JavaScript can be done by using the readFile() function.

➢ **<u>Syntax</u>**:

readFile(path, format, callBackFunc)

➢ The readFile() function accepts three parameters including one optional parameter.

4.) **Path** − The first parameter is the path of the test file from which the contents are to read. If the current location or directory is the same directory where the file which is to be opened and read is located then, only the file name has to be given.

5.) **Format** − The second parameter is the optional parameter which is the format of the text file. The format can be ASCII, utf-8 etc.

6.) **CallBackFunc** − The third parameter is the call back function which takes the error as the parameter and displays the fault is any raised due to the error.

- **What are all the looping structures in JavaScript?**

➤ **For** - loops through a block of code a number of times.
➤ **For/in** - loops through the properties of an object.
➤ **For/of** - loops through the values of an iterable object.
➤ **While** - loops through a block of code while a specified condition is true.
➤ **Do/While** - also loops through a block of code while a specified condition is true.

- **How can you convert the string of any base to an integer in JavaScript?**

➤ In JavaScript parseInt() function (or a method) is used to convert the passed-in string parameter or value to an integer value itself. This function returns an integer of the base which is specified in the second argument of the parseInt() function. JavaScript parseInt() function returns Nan(not a number) when the string doesn't contain a number.

➤ **<u>Syntax</u>**:

       parseInt(Value, radix)

- **What is the function of the delete operator?**

➤ Delete is comparatively a lesser-known operator in JavaScript. This operator is more specifically used to delete JavaScript object properties.

➤ The JavaScript pop(), shift(), or splice() methods are available to delete an element from an array. But because of the key-value pair in an object, deleting is more complicated. Note that, the delete operator only works on objects and not on variables or functions.

➤ **Syntax** :

```
delete object
// or
delete object.property
// or
delete object['property']
```

- **What are all the types of Pop up boxes available in JavaScript?**

➤ In Javascript, popup boxes are used to display the message or notification to the user. There are three types of pop-up boxes in JavaScript namely Alert Box, Confirm Box and Prompt Box.

➤ **Alert Box** : It is used when a warning message is needed to be produced. When the alert box is displayed to the user, the user needs to press ok and proceed.

➤ **Prompt Box** : It is a type of pop up box which is used to get the user input for further use. After entering the required details user have to click ok to proceed next stage else by pressing the cancel button user returns the null value.

➤ **Confirm Box** : It is a type of pop-up box that is used to get authorization or permission from the user. The user has to press the ok or cancel button to proceed.

- **What is the use of Void (0)?**

  ➢ In a programming language, void means return nothing. "javascript: void(0)" is similar to void.
  ➢ javascript: void(0) means return undefined as a primitive value. We use this to prevent any negative effects on a webpage when we insert some expression.
  ➢ It is used to prevent any side effects caused while inserting an expression in a web page.

- **How can a page be forced to load another page in JavaScript?**

➢ **Approach** : We can use window.location property inside the script tag to forcefully load another page in Javascript. It is a reference to a Location object that is it represents the current location of the document. We can change the URL of a window by accessing it.

➢ **Syntax** :

```
<script>
    window.location = <Path / URL>
</script>
```

- **What are the disadvantages of using inner HTML in JavaScript?**

➤ **The use of inner HTML very slow**: The process of using inner HTML is much slower as its contents as slowly built, also already parsed contents and elements are also re-parsed which takes time.

➤ **Preserves event handlers attached to any DOM elements:** The event handlers do not get attached to the new elements created by setting inner HTML automatically. To do so one has to keep track of the event handlers and attach it to new elements manually. This may cause a memory leak on some browsers.

➤ **Content is replaced everywhere:** Either you add, append, delete or modify contents on a webpage using inner HTML, all contents is replaced, also all the DOM nodes inside that element are reparsed and recreated.

➤ **Appending to inner HTML is not supported**: Usually, += is used for appending in JavaScript. But on appending to an Html tag using inner HTML, the whole tag is re-parsed.

➤ **Old content replaced issue:** The old content is replaced even if object.innerHTML = object.innerHTML + 'html' is used instead of object.innerHTML += 'html'. There is no way of appending without reparsing the whole inner HTML. Therefore, working with inner HTML becomes very slow. String concatenation just does not scale when dynamic DOM elements need to be created as the plus' and quote openings and closings becomes difficult to track.

➤ **Can also be used for Cross-site Scripting(XSS):** The fact that inner HTML can add text and elements to the webpage, can easily be used by malicious users to manipulate and display undesirable or harmful elements within other HTML element tags. Cross-site Scripting may also lead to loss, leak and change of sensitive information.

➢ **<u>Can break the document</u>:** There is no proper validation provided by inner HTML, so any valid HTML code can be used. This may break the document of JavaScript. Even broken HTML can be used, which may lead to unexpected problems.