1)What is JavaScript

Ans:

⇒ JavaScript is a versatile programming language primarily used for web development.

It allows you to add dynamic content, interactivity, and various functionalities to websites. JavaScript is commonly used alongside HTML and CSS to create dynamic and responsive web pages.

It can run in web browsers, making it an essential part of front-end development. Additionally, JavaScript has expanded its reach to server-side development (Node.js) and mobile app development.

2) What is the use of isNaN function?

Ans:

⇒ The **NaN** function in JavaScript stands for "is Not a Number." It is used to determine whether a given value is not a valid number. The function returns **true** if the value is NaN (not a number) and **false** if the value is a valid number or can be converted to one.

3) What is negative Infinity?

Ans:

⇒ In JavaScript, **Infinity** is a special numeric value representing positive infinity. On the other hand, **-Infinity** represents negative infinity. These values are used to denote mathematical concepts of positive and negative infinity in computations.

4) Which company developed JavaScript?

Ans:

⇒ JavaScript was developed by Netscape Communications Corporation. Brendan Eich, a programmer working at Netscape, created JavaScript in 1995.

Initially, it was named "Mocha" and later "LiveScript" before being officially named JavaScript. It was designed to add interactivity and dynamic content to web pages in the Netscape Navigator browser.

JavaScript has since become one of the most widely used programming languages for web development.

5) What are undeclared and undefined variables?

Ans :

⇒ **Undeclared Variables:**

- An undeclared variable is a variable that has been used in code without being declared using **var**, **let**, or **const**.
- Using an undeclared variable was allowed in older versions of JavaScript, but in modern JavaScript, it's considered bad practice. It can lead to issues and is generally avoided by declaring variables before using them.

  ```
  x = 10;  // undeclared variable
  ```

Undefined Variables:

- An undefined variable is a variable that has been declared but not assigned a value, or it is used before any value has been assigned to it.
- In JavaScript, when you declare a variable without assigning a value to it, the variable is automatically initialized with the value **undefined**.

```
var y;
console.log(y);  // Output: undefined
```

6) Write the code for adding new elements dynamically?

Ans :

⇒ Certainly! Adding new elements dynamically in JavaScript is commonly done using the Document Object Model (DOM)

```
var newParagraph = document.createElement("p");
```

7) What is the difference between ViewState and SessionState?

Ans :

⇒ 1.ViewState:-

• Scope: ViewState is used to store state information that is specific to a single web page.

• Storage: The information stored in ViewState is maintained on the client side, usually in a hidden field. It's included in the page's HTML and sent back and forth between the client and the server with each request/response.

• Lifetime: ViewState is short-lived and is only available during the lifespan of a single page. Once the user navigates away from the page, the ViewState is lost.

• Usage: ViewState is often used to persist state information between postbacks (round-trips between the client and server) for a specific page. It helps in maintaining the state of controls on the page across postbacks.

2.SessionState:-

• Scope: SessionState is used to store state information that needs to be shared across multiple pages during a user's session.

• Storage: The information stored in SessionState is maintained on the server. It can be stored in-memory, in a separate process, or in a database, depending on the configuration.

• Lifetime: SessionState persists throughout the user's session, which starts when the user accesses the website and ends when they close the browser or their session times out.

• Usage: SessionState is commonly used to store user-specific information, such as user preferences, authentication details, or shopping cart contents, across multiple pages.

8) What is === operator?

Ans :

⇒ The **===** operator in JavaScript is the strict equality operator. It is used to compare two values for equality without performing type coercion.

This means that both the value and the data type must be the same for the **===** operator to return **true**.

Example :

```
var x = 5;

var y = "5";


console.log(x === y);  // Output: false
```

9) How can the style/class of an element be changed?

Ans :

⇒ To change the style or class of an element in HTML using JavaScript, you can use the **style** property for inline styles or the **classList** property for managing classes. Here are examples for both scenarios.

Example:

```
<script>
  // Get the element by its ID
  var element = document.getElementById("myElement");

  // Change the style properties
  element.style.color = "red";
</script>
```

10) How to read and write a file using JavaScript?

Ans :

⇒ On the client side, you can't read or write files in JavaScript browsers. The fs module in Node.js

may be used to accomplish this on the server-side. It has methods for reading and writing files

on the file system that are both synchronous and asynchronous. Let's demonstrate some

examples of reading and writing files with the node.js fs module.

The fs.readFile() and rs.writeFile() methods are used to read and write of a file using javascript.

The file is read using the fs.readFile() function, which is an inbuilt method. This technique reads

the full file into memory and stores it in a buffer.

11) What are all the looping structures in JavaScript?

Ans :

⇒ JavaScript provides several looping structures for iterating through a set of instructions or a collection of data. The main looping structures are:

1.for Loop:

The **for** loop is a common looping structure that repeats a block of code a specified number of times.

for (initialization; condition; update) { // code to be repeated }

2.while Loop:

The **while** loop repeats a block of code as long as a specified condition is true.

while (condition) { // code to be repeated }

3.do...while Loop:

Similar to the **while** loop, but it always executes the block of code at least once, even if the condition is initially false.

do { // code to be repeated } while (condition);

4.for...in Loop:

Iterates over the properties of an object.

for (variable in object) { // code to be repeated }

5.for...of Loop:

Introduced in ECMAScript 6 (ES6), it iterates over iterable objects like arrays, strings, etc.

for (variable of iterable) { // code to be repeated }

6.**forEach Method:**

A method available for arrays, which allows you to iterate through each element of the array.

array.forEach(function(element) { // code to be executed for each element });

12) How can you convert the string of any base to an integer in JavaScript?

Ans :

⇒ In JavaScript, you can use the **parseInt** function to convert a string representation of a number from any base to an integer.

The **parseInt** function takes two arguments: the string to be converted and the base of the numeral system.

Example:

var a = "1101";

var Number =Number. parseInt(a);

13) What is the function of the delete operator?

Ans :

⇒ The **delete** operator in JavaScript is used to delete a property from an object or an element from an array. It can also be used to delete variables declared with the **var**, **let**, or **const**

var myObject = { key1: "value1", key2: "value2" };

// Deleting a property from an object

delete myObject.key1;

14) What are all the types of Pop up boxes available in JavaScript?

Ans :

⇒ In JavaScript, there are three main types of pop-up boxes commonly used for user interaction:

Alert Box : alert("This is an alert box!");

Confirm Box : var result = confirm("Do you want to proceed?");

console.log(result); // Result will be true or false

Prompt Box: var userInput = prompt("Enter something:");

console.log(userInput); // Result will be the entered value or null

15) What is the use of Void (0)?

Ans :

⇒ The use of `void(0)` in JavaScript is often seen in the context of using it as the href attribute in an anchor (``) tag to create a "void" or "no-operation" link. This is typically done to prevent the page from navigating to a new URL when the link is clicked.

16) How can a page be forced to load another page in JavaScript?

Ans:

⇒ In JavaScript, you can force a page to load another page by setting the **window.location** property to the desired URL. Here's an example:

```
// Redirect to another page
window.location.href = "https://www.example.com";
```

17) What are the disadvantages of using innerHTML in JavaScript?

Ans :

⇒ While the **innerHTML** property in JavaScript is a convenient way to manipulate the content of HTML elements, it comes with some potential disadvantages:

1. **Security Risks:**
    - Using **innerHTML** to insert or update content with user-generated input can expose your application to cross-site scripting (XSS) attacks if the input is not properly sanitized.
2. **Performance Concerns:**
    - Manipulating **innerHTML** can be computationally expensive, especially when dealing with large or complex DOM structures. Repeatedly updating **innerHTML** within loops or frequently modifying large

portions of the DOM can lead to performance issues. In such cases, alternative methods like creating and appending elements directly might be more efficient.

3. **Potential for Overwriting Event Listeners:**
    - If you use **innerHTML** to replace or update the content of an element, any existing event listeners on the replaced elements might be lost. This can lead to unexpected behavior or the need to reattach event listeners after using **innerHTML**.
4. **Limited Support for XML Documents:**
    - While **innerHTML** is widely supported for HTML documents, it may not work as expected when dealing with XML documents. In XML, the parsing rules are stricter, and certain tags and attributes may not be supported or behave differently.
5. **Accessibility Concerns:**
    - Manipulating content with **innerHTML** might not be as accessible as using other methods, especially when it comes to screen readers. Creating and appending new elements or using ARIA attributes may provide more accessible solutions in some cases.