MODULE: 4 (JavaScript Basic & DOM)

(1) What is JavaScript?

Ans -

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc.

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

(2) What is the use of isNaN function?

Ans -

In JavaScript NaN is short for "Not a Number".

Since NaN always compares unequal to any number, including NaN, it is usually used to indicate an error condition for a function that should return a valid number. When a string or something else is being converted into a number and that cannot be done, then we get to see NaN.

(3) What is negative Infinity?

Ans -

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.

Number.NEGATIVE_INFINITY is a special numeric value that is returned when an arithmetic operation or mathematical function generates a negative value greater than the largest representable number in JavaScript (i.e., more negative than -Number.MAX_VALUE).

JavaScript displays the NEGATIVE_INFINITY value as -Infinity. This value behaves mathematically like infinity; for example, anything multiplied by infinity is infinity, and anything divided by infinity is zero.

(4) Which company developed JavaScript?

Ans -

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.

In 1996, Netscape and Brendan Eich took JavaScript to the ECMA international standards organization, and a technical committee (TC39) was created to develop the language.

(5) What are undeclared and undefined variables?

Ans -

Undefined: It occurs when a variable has been declared but has not been assigned any value. Undefined is not a keyword.

Undeclared: It occurs when we try to access any variable that is not initialized or declared earlier using the var or const keyword.

(6) What is === operator?

Ans -

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 === operator compares the values as well as the data types of the operands.

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

Ans -

To change the style of an HTML element, use this syntax:

document.getElementById(id).style.property = new style

(8) Write the code for adding new elements dynamically?

Ans -

Create a <p> element and append it to the document:

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

<h1>The Document Object</h1>
<h2>The createElement() Method</h2>

<p>Create a p element with some text:</p>

<script>
// Create element:
const para = document.createElement("p");
para.innerText = "This is a paragraph.";

// Append to body:
document.body.appendChild(para);
</script>

</body>
</html>
```

(9) What is the difference between ViewState and SessionState?

Ans -

Differences 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.
ViewState can be used to store information that you wish to access from same web page.	SessionState can be used to store information that you wish to access on different web pages.

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

Ans -

In English, void means nothing. 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.

(11) What are all the looping structures in JavaScript?

Ans -

The for Loop:

```
for (expression 1; expression 2; expression 3) {
  // code block to be executed
}
```

The for In Loop:

```
for (key in object) {
  // code block to be executed
}
```

The for of Loop:

```
for (variable of iterable) {
  // code block to be executed
}
```

The while Loop:

```
while (condition) {
  // code block to be executed
}
```

The do while Loop:

```
do {
  // code block to be executed
}
while (condition);
```

(12) How to read and write a file using JavaScript?

Ans -

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.

Syntax:

fs.readFile(file_name, encoding, callback_function)

fs.writeFile(file_name, data, options, callback)

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

Ans -

Convert a string into a number is to use the parseInt() function.

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.

Syntax:

parseInt(string, radix)

(14) What is the function of the delete operator?

Ans -

The delete operator removes a property from an object. If the property's value is an object and there are no more references to the object, the object held by that property is eventually released automatically.

Syntax:
delete object
// or
delete object.property
// or
delete object['property']

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

Ans -

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>
```

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

Ans -

In JavaScript, Popup Boxes are used for displaying or showing the information to the user.

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Box:

An alert box is often used if you want to make sure information comes through to the user.
When an alert box pops up, the user will have to click "OK" to proceed.
Syntax:
window.alert("sometext");

Confirm Box:

A confirm box is often used if you want the user to verify or accept something.
When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.
Syntax
window.confirm("sometext");

Prompt Box:

A prompt box is often used if you want the user to input a value before entering a page.
When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
Syntax
window.prompt("sometext","defaultText");

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

Ans -

The use of innerHTML very slow:
The process of using innerHTML 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 innerHTML 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 innerHTML, all contents is replaced, also all the DOM nodes inside that element are reparsed and recreated.

Appending to innerHTML is not supported:
Usually, += is used for appending in JavaScript. But on appending to an Html tag using innerHTML, 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 innerHTML. Therefore, working with innerHTML 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 break the document:
There is no proper validation provided by innerHTML, 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.

Can also be used for Cross-site Scripting(XSS):
The fact that innerHTML 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.