❖ Module (JAVASCRIPT BASIC & DOM) – 4

- What is JavaScript?

Ans:- JavaScript (JS) is the most popular lightweight, interpreted compiled programming language. It can be used for both Client-side as well as Server-side developments. JavaScript also known as a scripting language for web pages.

JavaScript can be added to your HTML file in two ways:

- External JavaScript
- Internal JavaScript

Internal JavaScript: We can add JS code directly to our HTML file by writing the code inside the <script> & </script>. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.

External JavaScript: We can create the file with a .js extension and paste the JS code inside of it. After creating the file, add this file in <script src=”file_name.js”> tag, and this <sctipt> can

import inside <head> or <body> tag of the HTML file.

- What is the use of isNaN function?

Ans:-

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.

Example: In this example, we will check various values for isNan() and the output will be in boolean format.

```
<script>
console.log(isNaN(12));
console.log(isNaN(0 / 0));
console.log(isNaN(12.3));
console.log(isNaN("Geeks"));
console.log(isNaN("13/12/2020"));
console.log(isNaN(-46));
console.log(isNaN(NaN));
```

</script>

- **Output:**

false
true
false
true
true
false
true

- 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.

Note: JavaScript shows the NEGATIVE_INFINITY value as -Infinity.

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?

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, the Mozilla foundation continued to develop JavaScript for the Firefox browser. Mozilla's latest version was 1.8.5. (Identical to ES5).

- The first popular web browser with a graphical user interface, Mosaic, was released in 1993. Accessible to non-technical people, it played a prominent role in the rapid growth of the nascent World Wide Web.[11] The lead developers of Mosaic then founded the Netscape corporation, which released a more

polished browser, Netscape Navigator, in 1994. This quickly became the most-used.

- 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. If we use ‘typeof’ operator to get the value of an undeclared variable, we will face the runtime error with the return value as “undefined”. The scope of the undeclared variables is always global.

- Write the code for adding new elements dynamically?

Ans:-

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

Syntax:

document.createElement("<tagName>");

// Where <tagName> can be any HTML

// tagName like div, ul, button, etc.

// newDiv element has been created

ForEg:letnewDiv= document.createElement("div");

Once the element has been created, let’s move on to the setting of attributes of the newly created element.

Setting the attributes of the created element: Attributes can be set using **setAttribute()** method.

The syntax and example are as follows:

Element.setAttribute(name, value);

// Where Element is the name of web element.

// Here, we have created newDiv.

// Where name is the attribute name and

// value is the value that needs to be set

For Eg: new Div.setAttribute("class","container");

Example: Elements can be created based on some event like a click. Here’s an example of how to create elements dynamically with an onclick event. This code can be further made into a to do-list!

- What is the difference between ViewState and SessionState?

  Ans:-

- **ViewState:** It is maintained at only one level that is page-level. Changes made on a single page is not visible on other pages. Information that is gathered in view state

is stored for the clients only and cannot be transferred to any other place. View state is synonymous with serializable data only.

ViewState has a tendency for the persistence of page-instance-specific data. When view state is used, the values posted of a particular page persist in the browse area that the client is using and post back only when the entire operation is done. The data of the previous page is no longer available when another page is loaded. Also, Data is not secure in this case because it is exposed to clients. Encryption can be used for data security.

- **SessionState:** It is maintained at session-level and data can be accessed across all pages in the web application. The information is stored within the server and can be accessed by any person that has access to the server where the information is stored.

SessionState has the tendency for the persistence of user-specific data and is maintained on the server-side. This data

remains available until the time that the session is completed or the browser is closed by the user. The session state is only valid for type objects.

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.

Usage:

- **SessionState:** It can be used to store information that you wish to access on different web pages.

- **ViewState** It can be used to store information that you wish to access from same web page.

- 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.

Eg:-

```
console.log(1 === 1);
// Expected output: true

console.log('hello' === 'hello');
// Expected output: true

console.log('1' ===  1);
// Expected output: false

console.log(0 === false);
// Expected output: false
```

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

Ans:-

We can change, add or remove any CSS property from an HTML element on the occurrence of any event with the help of JavaScript. There are two approaches that allow us to achieve this task.

Approach 1: Changing CSS with the help of the style property:
Syntax:
document.getElementById("id").style.property = new_style

Approach 2: Changing the class itself – We can use two properties that can be used to manipulate the classes.

1. The classList Property: The **classList** is a read-only property that returns the CSS class names of an element as a DOMTokenList object.
Syntax:
document.getElementById("id").classList

You can use the below-mentioned methods to add classes, remove classes, and toggle between different classes respectively.

- **The add() method:** It adds one or more classes.
- **The remove() method:** It removes one or more classes.
  - **The toggle() method:** If the class does not exist it adds it and returns true. It removes the class and

returns false. The second boolean argument forces the class to be added or removed.

- How to read and write a file using JavaScript?

Ans:-

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.

Syntax:

fs.readFile(file_name, encoding, callback_function)

The fs.writeFile() function is used to write data to a file in an asynchronous manner. If the file already exists, it will be replaced.

Syntax:

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

- What are all the looping structures in JavaScript?

Ans:-

Loops offer a quick and easy way to do something repeatedly. This chapter of the JavaScript Guide introduces the different iteration statements available to JavaScript.

You can think of a loop as a computerized version of the game where you tell someone to take X steps in one direction, then Y steps in another. For example, the idea "Go five steps to the east" could be expressed this way as a loop:

```
for (let step = 0; step < 5; step++) {
   // Runs 5 times, with values of step 0 through 4.
  console.log("Walking east one step");
}
```

There are many different kinds of loops, but they all essentially do the same thing: they repeat an action some number of times. (Note that it's possible that number could be zero!)

The various loop mechanisms offer different ways to determine the start and end points of the loop. There are various situations that are more easily served by one type of loop over the others.

The statements for loops provided in JavaScript are:

for statement

do...while statement

for...in statement

for...of statement

while statement

labeled statement

break statement

continue statement

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

Ans:-

Given a string containing an integer value and along with that user passes a base value. We need to convert that string of any base value to an integer in JavaScript.

String	Integer
"1002"	1002

For performing the above-illustrated task, we would be using a method (or a function) provided by JavaScript called as **parseInt().**

This is a special method, provided by JavaScript, that takes an integer value (of any base which is either specified or not) and further converts the string into an integer value.

Syntax:

- Following is the syntax that a user may use to convert a string into an integer value (of any base)-

  parseInt(string_value, base)
- Alternatively, if we don’t want to specify the base value and just want to convert our string value into an integer value itself, then we may use the following syntax also-

  parseInt(string_value)

Default value returned by base or radix of parseInt() method is **10.** In other words, if we don’t specify any base or radix value then it by default converts the string value to an integer value by taking into regard the base or radix value as 10.

Example 1: In this example, we would be passing the string value in a method (which is explicitly declared for ease purpose) and further that string value is passed inside the parseInt() method which then further converts that string value in the corresponding integer value.

- JavaScript

```
let stringConversion = (string_value) => {

  console.log("Initial Type: " + typeofstring_value);

  let integer_value = parseInt(string_value);

  console.log("Final Type: " + typeofinteger_value);

  console.log(integer_value);

};



stringConversion("512000");
```

```
stringConversion("126410");

stringConversion("0x8975");
```

Output:

Initial Type: string

Final Type: number

512000

Initial Type: string

Final Type: number

126410

Initial Type: string

Final Type: number 35189

- 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.

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']

Parameter: It does not take any parameter.

Return type: This operator returns *true* if it removes a property. While deleting an object property that doesn't exist will return a *true* but it will not affect the object. Though while trying to delete a variable or a function will return a *false*

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

Ans:- 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*");

The window.alert() method can be written without the window prefix.

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*");

The window.confirm() method can be written without the window prefix.

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*");

The window.prompt() method can be written without the window prefix.

- What is the use of Void (0)?

Ans:- You might have occasionally came across “javascript:void(0)” in an HTML Document. It is often used when inserting an expression in a web page might produce some unwanted effect. To remove this effect, *“javascript:void(0)”* is used. This expression returns undefined primitive value. This is often used with hyperlinks. Sometimes, you will decide to call some JavaScript from inside a link. Normally, when you click a link, the browser loads a brand new page or refreshes the same page (depending on the URL specified). But you most likely don’t desire this to happen if you have hooked up some JavaScript

thereto link. To prevent the page from refreshing, you could use void(0).
Using "#" in anchor tag: When writing the following code in the editor, the web page is refreshed after the alert message is shown.

Using "javascript:void(0);" in anchor tag: Writing "javascript:void(0);" in anchor tag can prevent the page to reload and JavaScript functions can be called on single or double clicks easily.

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

Ans:-

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 innerHTML in JavaScript?

Ans:-

Disadvantages of innerHTML:

- Event handlers attached to any DOM element are preserved.
- Replacement is done everywhere.
- It is not possible to append innerHTML.

- Breaks the document.
- Used for Cross-site Scripting.