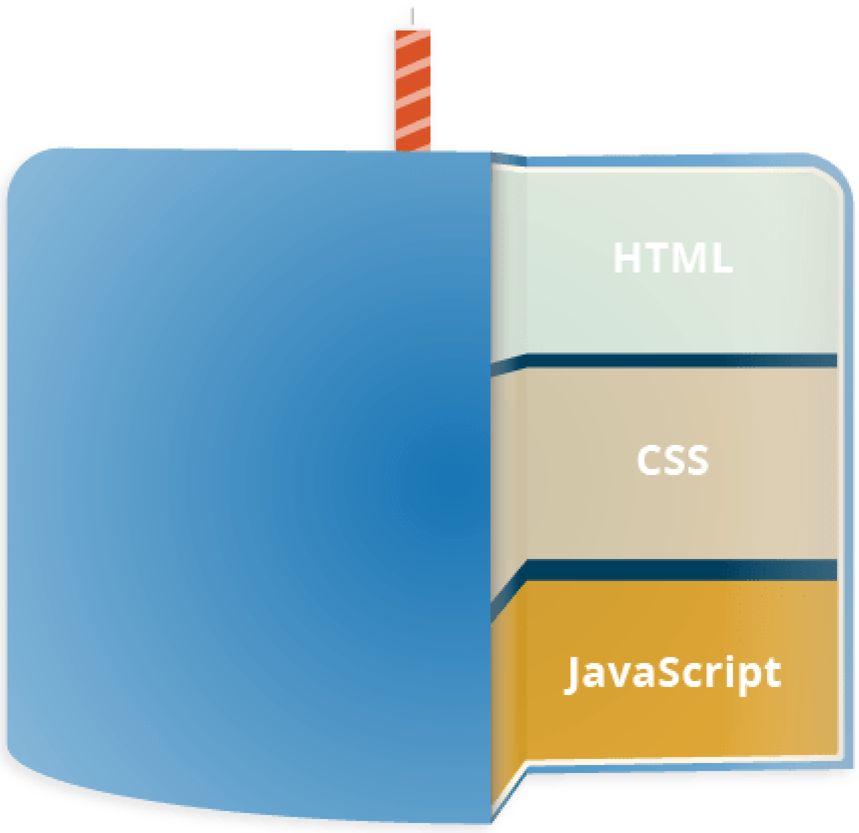
MODULE: 4 (JAVASCRIPT BASIC & DOM)

TO: ARPIT KANSARA
FROM:SHIVANI GOHIL

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. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS) we have covered in much more detail in other parts of the Learning Area.



https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/javascript%20example.html

2.What is the use of isNaN function?

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

The isNaN() method returns true if a value is NaN.

The isNaN() method converts the value to a number before testing it.

- 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.

Syntax:

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 isNaN() function.

Return Value: It returns a Boolean value i.e., returns true if the value is NaN else returns false.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/isNaN()%20Function.html

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.
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.

- NEGATIVE_INFINITY is a property of the JavaScript Number object.
- You can only use it as Number.NEGATIVE_INFINITY.
- Using x.NEGATIVE_INFINITY, where x is a variable, will return undefined:

Syntax:

Number.NEGATIVE_INFINITY

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Negative%20infinity.html

4. Which company developed JavaScript?

Ans. JavaScript / ECMAScript

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).

Internet Explorer (IE4) was the first browser to support ECMA-262 Edition 1 (ES1).

Year	ECMA	Browser
1995		JavaScript was invented by Brendan Eich
1996		Netscape 2 was released with JavaScript 1.0
1997		JavaScript became an ECMA standard (ECMA-2

1997	ES1	ECMAScript 1 was released
1997	ES1	IE 4 was the first browser to support ES1
1998	ES2	ECMAScript 2 was released
1998		Netscape 42 was released with JavaScript 1.3
1999	ES2	IE 5 was the first browser to support ES2
1999	ES3	ECMAScript 3 was released
2000	ES3	IE 5.5 was the first browser to support ES3
2000		Netscape 62 was released with JavaScript 1.5
2000		Firefox 1 was released with JavaScript 1.5
2008	ES4	ECMAScript 4 was abandoned
2009	ES5	ECMAScript 5 was released

2011	ES5	IE 9 was the first browser to support ES5 *
2011	ES5	Firefox 4 was released with JavaScript 1.8.5
2012	ES5	Full support for ES5 in Safari 6
2012	ES5	Full support for ES5 in IE 10
2012	ES5	Full support for ES5 in Chrome 23
2013	ES5	Full support for ES5 in Firefox 21
2013	ES5	Full support for ES5 in Opera 15
2014	ES5	Full support for ES5 in all browsers
2015	ES6	ECMAScript 6 was released
2016	ES6	Full support for ES6 in Chrome 51
2016	ES6	Full support for ES6 in Opera 38

2016	ES6	Full support for ES6 in Edge 14
2016	ES6	Full support for ES6 in Safari 10
2015	ES6	Full support for ES6 in Firefox 52
2018	ES6	Full support for ES6 in all browsers **

5. What are undeclared and undefined variables?

Ans. Undeclared − It occurs when a variable which hasn’t been declared using var, let or const is being tried to access.

Undefined − It occurs when a variable has been declared using var, let or const but isn’t given a value.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/tree/main/Java%20script%20practicals

6. Write the code for adding new elements dynamically?

Ans. 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*>");
// Where <*tagName*> can be any HTML
// tagName like div, ul, button, etc.

// newDiv element has been created
For Eg: **let newDiv = document.createElement("div");**
Once the element has been created, let’s move on to the setting of attributes of the newly created element.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Adding_%20new_%20elements_dynamically.html

7. What is the difference between ViewState and SessionState?

Ans. The basic difference between these two is that the ViewState is to manage state at the client’s end, making state management easy for end-user while SessionState manages state at the server’s end, making it easy to manage content from this end too.

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.

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.

•

8. What is === operator?

Ans. JavaScript ‘===’ operator: Also known as strict equality operator, it compares both the value and the type which is why the name “strict equality”.

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.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Strict%20equality.html

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

Ans. 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.

Syntax:

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

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Class_of%20_element_can_be_change.html

10. 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)

Parameters:

- **filename:** It contains the filename to be read, or the whole path if the file is saved elsewhere.
- **encoding:** It stores the file's encoding. 'utf8' is the default setting.
- **callback function:** This is a function that is invoked after the file has been read. It requires two inputs:
- **err:** If there was an error.
- **data:** The file's content.
- **Return Value:** It returns the contents contained in the file, as well as any errors that may have occurred.

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)

Parameters:

- **file_name**: It's a string, a buffer, a URL, or a file description integer that specifies the location of the file to be written. When you use a file descriptor, it will function similarly to the fs. write() method.
- **data**: The data that will be sent to the file is a string, Buffer, TypedArray, or DataView.
- **options:** It's a string or object that may be used to indicate optional output options. It includes three more parameters that may be selected.
- **encoding**: It's a string value that indicates the file's encoding. 'utf8' is the default setting.
- **mode**: The file mode is specified by an integer number called mode. 0o666 is the default value.
- **flag**: This is a string that indicates the file-writing flag. 'w' is the default value.
- **callback**: This function gets invoked when the method is run.
- **err**: If the process fails, this is the error that will be thrown.

11. What are all the looping structures in JavaScript?

Ans. Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this is the case when working with arrays:

The JavaScript loops are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

❖ JavaScript supports different kinds of loops:

for - loops through a block of code a number of times

for/in - loops through the properties of an 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

nested loop-The most common type of nested loops will be one loop inside another.

The first loop is usually called the outer loop while the second loop is called the inner loop.

The outer loop always executes first, and the inner loop executes inside the outer loop each time the outer loop executes once.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/tree/main/Java%20script%20practicals/Types%20of%20loop

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

Ans. 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.

Syntax:

parseInt(Value, radix)

Parameters: It accepts a string as a value and converts it to a specified radix system (any desired numerical value passed by a user) and returns an integer (corresponding to the passed in numerical radix value).

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Parseint().html

13. 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 objectName

delete objectName.property

delete objectName[index]

delete property // The command acts only within a with statement.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Delete%20operator.html

14. 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.

1.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.

2.Conform 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**.

3.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.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/tree/main/Java%20script%20practicals/Popup%20Boxes

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

Ans.

The word void means "completely empty space" according to the dictionary. This term, when used in programming, refers to a return of "nothing" - an "empty value" so to speak JavaScript void 0 means returning undefined (void) as a primitive value. You might come across the term "JavaScript:void(0)" while going through HTML documents. It is used to prevent any side effects caused while inserting an expression in a web page. For instance, URLs or hyperlinks are the common examples of using JavaScript void 0. Suppose you insert a link and want to call some JavaScript through it. Usually, when you click on a link, the browser will either reload or open a new page. However, if you just want to call JavaScript through that link, you would not want the entire page to refresh. This is where the JavaScript:void(0) will come in handy.

When you use JavaScript void 0, it will return an undefined primitive value. This will prevent the browser from opening a new or reloading the web page and allowing you to call the JavaScript through it.

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Void(0).html

16. 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>
```

https://github.com/Shivani081/JavaScript-Practical-for-assignment/blob/main/Java%20script%20practicals/Window_location.html

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

Ans.

The innerHTML property is a part of the Document Object Model (DOM) that is used to set or return the HTML content of an element. Where the return value represents the text content of the HTML element. It allows JavaScript code to manipulate a website being displayed. More specifically, it sets or returns the HTML content (the inner HTML) of an element. The innerHTML property is widely used to modify the contents of a webpage as it is the easiest way of modifying DOM. But there are some disadvantages to using innerHTML in JavaScript.

Disadvantages of using innerHTML property in JavaScript:

- **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.

  Example:

```
<p id="geek">Geeks</p>

title = document.getElementById('#geek')


// The whole "geek" tag is reparsed

title.innerHTML += '<p> forGeeks </p>'
```

- **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.

THANK YOU