**Assignment-4**

**1)What is JavaScript?**

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.

Many browsers use JavaScript as a scripting language for doing dynamic things on the web. Any time you see a click-to-show dropdown menu, extra content added to a page, and dynamically changing element colors on a page, to name a few features, you're seeing the effects of JavaScript.

**2)What is the use of isNaN function?**

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

**Syntax:**

isNaN( value )

**Parameter Values:** This method accepts 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.

**Supported Browsers:**

Chrome 1 and above

Firefox 1 and above

Edge 12 and above

Opera 3 and above

Safari 1 and above

**3)What is negative Infinity?**

The negative infinity in JavaScript is a constant value which is used to represent a value which 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 when divided by any other number.

When divided by itself or positive infinity, negative infinity returns 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 NaN and negative infinity is 0.

The product of two negative infinities is always a positive infinity.

The product of both positive and negative infinity is always negative infinity.

**Syntax:**

Number.NEGATIVE\_INFINITY

**4)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.

**5)What are undeclared and undefined variables?**

**Undefined:** It occurs when a variable has been declared but has not been assigned with 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 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.

**For example:**

Undefined:

var geek;

undefined

console.log(geek)

Undeclared:

//ReferenceError: myVariable is not defined

console.log(myVariable)

**6)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>");

// Where <tagName> can be any HTML

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

// newDiv element has been created

For E.g.: 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.

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 E.g.: newDiv.setAttribute("class","container");

**7)What is the difference between ViewState and SessionState?**

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.

**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.

**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?**

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.

**Syntax:**

x === y

**Description:**

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.

The most notable difference between this operator and the equality (==) operator is that if the operands are of different types, the == operator attempts to convert them to the same type before comparing.

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

**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.

**2. The className Property:** This property is used to set the current class of the element to the specified class.

**Syntax:**

document.getElementById("id").className = class

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

The read and write operations in a file can be done by using some commands. But the module which is required to perform these operations is to be imported. The required module is ‘fs’ which is called as File System module in JavaScript.

Write operation on a file

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 −

writeFile(path,inputData,callBackFunction)

The writeFile() function accepts three parameters −

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.

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

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.

Reading from the file

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

**Syntax:**

The syntax to read from a file is as follows −

readFile(path, format, callBackFunc)

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

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.

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

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.

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

Looping in programming languages is a feature that facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true. For example, suppose we want to print “Hello World” 10 times. This can be done in two ways as shown below:

**Iterative Method:** The iterative method to do this is to write the document.write() statement 10 times.

<script type="text/javascript">

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

document.write("Hello World<br>");

</script>

**Using Loops:** In Loop, the statement needs to be written only once and the loop will be executed 10 times as shown below:

<script type="text/javascript">

for (let i = 0; i < 10; i++)

{

document.write("Hello World!<br>");

}

</script>

In computer programming, a loop is a sequence of instructions that is repeated until a certain condition is reached.

1- An operation is done, such as getting an item of data and changing it, and then some condition is checked such as whether a counter has reached a prescribed number.

2- Counter not Reached: If the counter has not reached the desired number, the next instruction in the sequence returns to the first instruction in the sequence and repeats it.

3- Counter reached: If the condition has been reached, the next instruction “falls through” to the next sequential instruction or branches outside the loop.

There are mainly two types of loops:

**1- Entry Controlled loops:** In these types of loops, the test condition is tested before entering the loop body. For Loops and While Loops are entry-controlled loops.

**2- Exit Controlled loops:** In these types of loops the test condition is tested or evaluated at the end of the loop body. Therefore, the loop body will execute at least once, irrespective of whether the test condition is true or false. The do-while loop is exit controlled loop.

JavaScript mainly provides three ways for executing the loops. While all the ways provide similar basic functionality, they differ in their syntax and condition checking time. Let us learn about each one of these in detail.

**1)while loop:** A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

**Syntax:**

while (boolean condition)

{

loop statements...

}

• While loop starts with checking the condition. If it is evaluated to be true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason, it is also called the Entry control loop

• Once the condition is evaluated to be true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.

• When the condition becomes false, the loop terminates which marks the end of its life cycle.

**2)for loop:** for loop provides a concise way of writing the loop structure. Unlike a while loop, a for statement consumes the initialization, condition, and increment/decrement in one line thereby providing a shorter, easy-to-debug structure of looping.

**Syntax:**

for (initialization condition; testing condition;

increment/decrement)

{

statement(s)

}

1- Initialization condition: Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.

2- Testing Condition: It is used for testing the exit condition for a loop. It must return a boolean value. It is also an Entry Control Loop as the condition is checked prior to the execution of the loop statements.

3- Statement execution: Once the condition is evaluated to be true, the statements in the loop body are executed.

4- Increment/ Decrement: It is used for updating the variable for the next iteration.

5- Loop termination: When the condition becomes false, the loop terminates marking the end of its life cycle.

**3)do-while:** The do-while loop is similar to the while loop with the only difference that it checks for the condition after executing the statements, and therefore is an example of an Exit Control Loop.

**Syntax:**

do

{

statements.

}

while (condition);

1- The do-while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.

2- After the execution of the statements, and update of the variable value, the condition is checked for a true or false value. If it is evaluated to be true, the next iteration of the loop starts.

3- When the condition becomes false, the loop terminates which marks the end of its life cycle.

4- It is important to note that the do-while loop will execute its statements at least once before any condition is checked, and therefore is an example of the exit control loop.

4)Infinite loop: One of the most common mistakes while implementing any sort of looping is that it may not ever exit, that is the loop runs for infinite times. This happens when the condition fails for some reason.

**12)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 base which is specified in second argument of parseInt() function. JavaScript parseInt() function returns Nan( not a number) when the string doesn’t contain number.

**Syntax:**

parseInt(Value, radix)

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

**13)What is the function of the delete operator?**

The delete operator deletes an object, an object's property, or an element from an array. The operator can also delete variables which are not declared with the var statement.

**Syntax:**

delete objectName

delete objectName.property

delete objectName[index]

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

**Parameters**

objectName:The name of an object.

property: The property is an existing property.

index: An integer representing the array index.

**14)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.

**1)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.

**Syntax:**

alert("your Alert here")

**2)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.

**Syntax:**

prompt("your Prompt here")

**3)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.

**Syntax:**

confirm ("your query here")

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

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.

**16)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>

**Example:**

<script>

window.location = "https://www.geeksforgeeks.org/"

</script>

So in the above example, we see that by changing the window.location Object inside Javascript we can change the URL of our window and thus successfully load any page forcibly from our Javascript without any href tag. We will build a small working sample to learn it practically.

Below is the step-by-step implementation:

**Step 1:** Create a file named index.html. Add a heading and two buttons to it. One button forcefully loads a page with a live URL and the other button loads a local HTML page. In the <script> tag we have two functions, one loads gfg home page, and the second loads a local HTML page using window.location property.

**Step 2:** Create a file named newPage.html. This is the local HTML page that would be loaded by Javascript.

**17)What are the disadvantages of 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.

• 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.