|  |
| --- |
| ASSIGNMENT |
| MODULE:4 |
| (JAVASCRIPT BASIC & DOM) |
|  |

|  |
| --- |
| AAKASH SATHWARA |

1. **What is JavaScript?**

**Ans.** JavaScript is a script and 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 details in other parts of the learning area.

1. **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 number before testing

It.

**Example:**

**https://github.com/Webakki/Practicle/blob/main/isNaN()**

1. **What is negative infinity?**

**Ans.** NEGATIVE\_INFINITY is a special numeric value that is returned when arithmetic operation or mathematical function generates a negative value greater than the largest represantable number in javascript. Javascript displays the NAGATIVE\_INFINTY value as –infinity. This value behaves mathematically like infinity; for example, anything multiplied by infinity is infinity, and anything divided by infinity is zero. In ECMAscript v1 and later, you can also use –infinity instead of number .NEGATIVE\_INFINITY.

**Example:**

**https://github.com/Webakki/Practicle/blob/main/Nagative\_Infinity**

1. **Which company developed javascript?**

**Ans.** Javascript was invented by Brenden Eich in 1997.

It was developed for Netscape 2, and became the ECAM-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).

1. **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.

**Example:**

**https://github.com/Webakki/Practicle/blob/main/Undefined%26Undeclared.html**

1. **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 Javascript by using the createElement() method.

**Example:**

**https://github.com/Webakki/Practicle/tree/main/Adding%20New%20Element%20in%20Dynamically**

1. **What is the difference between view state and session state?**

**Ans.**

|  |  |
| --- | --- |
| View state | Session state |
| Maintained at page level only. | Maintained as session level only. |
| 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 postbook 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 stored in the server. |
| Used to allowed the persistence of page-instance-specific data. | Used for the persistence of user-specific data on the server’s end. |
| View state values are lost/cleared when new page is loaded. | Session state can be cleared by programmer or user or in case of timeouts. |

**Usage:**

* **Session state**: it can be used to store information that you wish to access on different web pages.
* **View state**: it can be used to store information that you wish to access from same web pages.

1. **What is === operators?**

**Ans.** Javascript ‘===’ operators: also known as strict equality operators, it compare both the value and the type which is why the name “strict equality”

The strict equality (===) operators checks whether its two operands are equal, returning a boolean result. Unlike the equality operators, the strict equality operator, the strict equality operators always considers operands of different type to be different.

**Example:**

**https://github.com/Webakki/Practicle/blob/main/Operator.html**

1. **How can the style/class of an elements be changes?**

**Ans.** The class name is used as a selectors in HTML which helps to give some value to the elements attributes.

The Document.getElementsbyId() 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”;

**Example:** [**https://github.com/Webakki/Practicle/tree/main/Class%20of%20an%20Element%20be%20Changed**](https://github.com/Webakki/Practicle/tree/main/Class%20of%20an%20Element%20be%20Changed)

1. **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 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. ‘uff8’ is the default setting.
* **Callback function:** This is a function that is invoked after the using file has been read. It requires two inputs:
* **Err:** if there was an error.
* **Data:** the file’s content.
* **Return value:** it returns the containscontained 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 is already exists, is will be replaced.

**Syntax:**

fs.writeFile( file\_name data, options, callbacks )

**Parameters:**

* **Filename:** it’s a string , a buffer, a URL, or a file description intenger that specifies the location or 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, typed Array, or data view.
* **Options:** it’s a string or object that may be used to indicate optional output option. It includes three more parameters that may be selected.
* **Encoding:** it’s a string value that indicate the file’s encoding. ‘utf8’ is the default setting.
* **Mode:** the file mode is a specified by an integer number called mode. 0o666 is the default value.
* **Flag:**  this is the string 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.

1. **What are all the looping structure in javascript?**

**Ans.** Loops are handy, if you want to run the same code over and over again, each time with 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, and do while or for-in loops. It makes the code code compact. It is mostly used in array.

Javascript supports different kinds of loops:

**For :** Loops through a block of code a numbers of time

**For/in:** Loops through the properties of an a 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 nested loops will be one loop inside another.

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

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

**Example:**

**https://github.com/Webakki/Practicle/tree/main/Loops**

1. **How can convert the string of any base to an integer in javascript?**

**Ans.** In java script parselnt() 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 parselnt() function. Javascript parselnt() function returns NaN(not a number) when the string doesn’t contain a number.

**Syntax:**

Parselnt(Value, radix)

**Parameters:**

It accepts a string as a value and converts it to a specified radix system and return an integer.

**Example:**

**https://github.com/Webakki/Practicle/blob/main/Parseint.html**

1. **What is the function of the delete operator?**

**Ans:** The delete operators 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.

**Example:**

**https://github.com/Webakki/Practicle/blob/main/Delete\_Operator.html**

1. **What are all the types of popup 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.

1. **Confirm :** 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 return false.

1. **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 return null.

**Example:**

**https://github.com/Webakki/Practicle/tree/main/popup%20boxes**

1. **What is the use of void (0)?**

**Ans.** Void(0) means return undefined as a primitive value.

We use this to prevent any negative effects on web page when we insert some expression.

For Example:

In the case of URL hyperlinks. Hyperlinks open by reloading the page when the user clicks on the link.

When you need to run some other code in such cases, you can use javascript: void(0).

**Example:**

**https://github.com/Webakki/Practicle/blob/main/Void(0).html**

1. **How can a page be forced to load another Page in javascript?**

**Ans. Approach:** we can use window.location property in side the script tag to forcefully load another page in javascript. It is a reference to a location object that is represents the current location of the document.

We can change the URL of a window by accessing it

<script>

Window.location = <path/URL>

</script>

**Example:**

**https://github.com/Webakki/Practicle/blob/main/window.location.html**

1. **What are the disadvantages of using innerHTML in javascript?**

**Ans.** The innerHTML property is a part of a document object model (DOM) that is used to setor return the html content of element. Where the return value represents the text content of html element. It allows to 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 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 javscript:

* **The use of innerHTML very slow**: the process of using innerHTML is much slower as its contents as slowly built, also already parsed content 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 to 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, +=used for appending in javascript. But on appending to an html tag using innerHTML, the whole tag re-parsed.
* **Example:**

**https://github.com/Webakki/Practicle/blob/main/inner.html**

<p id=”Akki”>Akki</p>

Title = document.getElementById(‘#Akki’)

//The Whole “Akki” tahe is reparsed

Title.innerHTML +=’<p>forAkki</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. There for working with innerHTML becomes very slow. String concentration just does not scale when dynamic DOM elements need to be created as the plus’ and qoute openings and closing becomes difficult to track.
* **Can breck the document:** There is no proper validtion provided by innnerHTML, so any valid HTML code can be used, which may lead to unexpected problems.
* **Can also be used for cross-site javascript scripting(XSS):** The fact that innerHTML can added text and elements to the web page, can easilybe used by malicious user to manipulate and display undesirable or harmful elementswithin other HTML element tags. Cross-site scripting may also lead to loss, leak and change of sensitive information.