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.**

2. What is the use of is Nan function?

**NaN values are generated when arithmetic operations result is undefined or unrepresented values. Such value do not necessarily represent overflow conditions.**

**For example, dividing zero by zero results in a NaN — but dividing other numbers by zero does not.**

**Examples**

**Using NaN()**

**Number.isNaN(NaN); // true**

**Number.isNaN(Number.NaN); // true**

**Number.isNaN(0 / 0); // true**

**Number.isNaN(37); // false**

**Difference between Number.isNaN () and global is NaN ()**

**Number.isNaN() doesn't attempt to convert the parameter to a number, so non-numbers always return false. The following are all false:**

**Number.isNaN ("NaN");**

**Number.isNaN (undefined);**

**Number.isNaN({});**

**Number.isNaN ("blabla");**

**Number.isNaN (true);**

**Number.isNaN (null);**

**Number.isNaN ("37");**

**Number.isNaN ("37.37");**

**Number.isNaN ("");**

**Number.isNaN (" ");**

**The global isNaN () coerces its parameter to a number:**

**isNaN ("NaN"); // true**

**isNaN (undefined); // true**

**isNaN({}); // true**

**isNaN("blabla"); // true**

**isNaN(true); // false, this is coerced to 1**

**isNaN(null); // false, this is coerced to 0**

**isNaN("37"); // false, this is coerced to 37**

**isNaN("37.37"); // false, this is coerced to 37.37**

**isNaN(""); // false, this is coerced to 0**

**isNaN(" "); // false, this is coerced to 0**

3. What is negative Infinity?

**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.**

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.**

**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).**

5.What are undeclared and undefined variables?

**Undefined variable means a variable has been declared but does not have a value. Undeclared variable means that the variable does not exist in the program at all**.

**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.**

6.Write the code for adding new elements dynamically?

* **New elements can be dynamically created in JavaScript with the help of createElement() method. The attributes of the created element can be set using the setAttribute() method.**

7.What is the difference between ViewState and SessionState?

* **ViewState**
  + **Maintained at page level only.**
  + **View state can only be visible from a single page and not multiple pages.**
  + **It will retain values in the event of a postback operation occurring.**
  + **Information is stored on the client’s end only.**
  + **used to allow the persistence of page-instance-specific data.**
  + **ViewState values are lost/cleared when new page is loaded.**
* **SessionState**
  + **Maintained at session level.**
  + **Session state value availability is across all pages available in a user session.**
  + **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 server.**
  + **used for the persistence of user-specific data on the server’s end.**
  + **SessionState can be cleared by programmer or user or in case of timeouts.**

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.**

**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.**

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

**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";**

**Example : In this code change the class of the button from “default” to “changedClass” using the onclick event which in turn changes the background color of the button from RED to GREEN.**

<style>

    .default {

        background-color: red;

    }

    .changedClass {

        background-color: green;

    }

</style>

<body>

    <h1 style="color:green">

        for Example

    </h1>

    <h2> Change class name of element

    </h2>

    <button class="default" onclick="changeClass()" id="myButton"> Click

        Here!</button><br>

    <p id="myPara">

        Old class name: default

    </p>

    <script>

        function changeClass() {

            document.getElementById('myButton')

            .className = "changedClass";

            var button\_class = document.getElementById('myButton')

                .className;

            document.getElementById('myPara')

                .innerHTML = "New class name: " + button\_class;

        }

    </script>

</body>

10.How to read and write a file using JavaScript?

**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.**

**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.**

**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, Typed Array, 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.**

**EXAMPLE:**

**var** fs = require("fs");

console.log(" Writing into an file ");

// Sample.txt is an empty file

fs.writeFile(

"sample.txt",

"Let's write a few sentences in the file",

function (err) {

if (err) {

**return** console.error(err);

}

// If no error the remaining code executes

console.log(" Finished writing ");

console.log("Reading the data that's written");

// Reading the file

fs.readFile("sample.txt", **function** (err, data) {

if (err) {

**return** console.error(err);

}

console.log("Data read : " + data.toString());

});

}

);

11.What are all the looping structures in JavaScript?

* **Loops can execute a block of code a number of times**.
* **JavaScript Loops**
* **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:**
* **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**
* **for/of - loops through the values of an iterable 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**

Example:



**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 the base which is specified in the second argument of the parseInt() function.**

**Example 1: In this example, we will convert a string value to an integer using the parseInt() function with no second parameter.**

* javascript

|  |
| --- |
| **function** convertStoI() {  var a = "100";  var b = parseInt(a);  console.log("Integer value is" + b);  var d = parseInt("3 11 43");  console.log('Integer value is ' + d);  }  convertStoI(); |

**Output:**

Integer value is 100

Integer value is 3

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

**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**.

**EXAMPLE:**

****

**14.What are all the types of Pop up boxes available in JavaScript?**

**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.

**Example** : alert("I am an alert box!");

**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.

**Example**

if (confirm("Press a button!")) {

txt = "You pressed OK!";

}

else {

txt = "You pressed Cancel!";

}

**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.

**Example**

let person = prompt("Please enter your name", "Harry Potter");

let text;

if (person == null || person == "") {

text = "User cancelled the prompt."; }

else { text = "Hello " + person + "! How are you today?"; }



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

**If inserting an expression into a web page results in an unwanted effect, then use JavaScript void to remove it. Adding “JavaScript: void (0)”, returns the undefined primitive value.**

Graphical user interface, text, application

Description automatically generated

**16.How can a page be forced to load another page inJavaScript ?**

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**

<!DOCTYPE html>

<html lang="en">

<head>

<**meta** charset="UTF-8">

<**meta** http-equiv="X-UA-Compatible"

content="IE=edge">

<**meta** name="viewport" content=

"width=device-width, initial-scale=1.0">

<**title**> New Page </**title**>

</head>

<body>

<**h3**>This is the new loaded page</**h3**>

</body>

</html>

**17.What are the disadvantages of using innerHTML 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.**

**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.**

**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.**