MODULE: 4 (JavaScript Basic & DOM)

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.

Javascript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.

2. What is the use of isNaN function?

➔ The **isNaN()** function determines whether a value is NaN when converted to a number. Because coercion inside the isNaN() function can be surprising , you may alternatively want to use Number.isNaN().

Example:

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Global Methods</h1>
<h2>The isNaN() Method</h2>

<p>isNaN() returns true if a value is NaN:</p>

<p id="demo"></p>

<script>
let result =
"Is 123 NaN? " + isNaN(123) + "<br>" +
"Is -1.23 NaN? " + isNaN(-1.23) + "<br>" +
"Is 5-2 NaN? " + isNaN(5-2) + "<br>" +
"Is 0 NaN? " + isNaN(0);
document.getElementById("demo").innerHTML = result;
</script>

</body>
</html>
```

Output:

JavaScript Global Methods

The isNaN() Method

isNaN() returns true if a value is NaN:

Is 123 NaN? false
Is -1.23 NaN? false
Is 5-2 NaN? false
Is 0 NaN? false

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.

Example:

```
<body>
    <h1 style="color: green;">
        GeeksforGeeks
    </h1>
    <h3>
        What is negative infinity in JavaScript?
    </h3>
    <button onclick="geekPositiveInfinity()">
        Generate negative infinity
    </button>
    <p id="geek"></p>
    <script>
        function geekPositiveInfinity() {
            var n=(-Number.MAX_VALUE)*2;
            document.getElementById("geek").innerHTML=
              "Here the number generated is twice of negative of
Number.MAX_VALUE"+"<br>"+
              " which is lesser than lower limit"+"<br><br>"+n;
        }
    </script>
</body>
```

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 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.

6. Write the code for adding new elements dynamically?

The createElement() method in JavaScript can be used to create new items dynamically. The setAttribute() method is used to set the attributes of the newly generated element.

```
<html>

<title>

    Adding new elements dynamically

</title>



<body>

    <button id="button">Hit me to add elements
dynamically</button>
    <h3 id="heading_A"></h3>
    <h5 id="alert"></h5>
    <script>
        const button = document.getElementById('button');
        const text = document.getElementById('heading_A');
        const alrt = document.getElementById('alert');
        button.onclick = () => {
            const name = prompt('What is your name?');
            const course = prompt('Which Course we are learning
?');
            alert(`Hello ${name}, Welcome to our
group...!`+ "\n" + `We are learning ${course}`);
            text.textContent = `Welcome ${name}to our
group...!` + `We are learning ${course}`;
           alert(button.textContent);
           text.textContent = `Welcome ${name}to our
group...!` + `We are learning ${course}`;

      }
    </script>
</body>

</html>
```

7. What is the difference 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.

8. What is === operator?

➔ === (Triple equals) is a strict equality comparison operator in JavaScript, which returns false for the values which are not of a similar type. This operator performs type casting for equality. If we compare 2 with "2" using ===, then it will return a false value.

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

->

In this article, we will learn how we can change the style/class of an element. If you want to build a cool website or app then UI plays an important role. 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 common approaches that allow us to achieve this task.

- style.property
- Changing the class itself

Approach 1: Changing CSS with the help of the style property:
Syntax:

```
document.getElementById("id").style.property = new_style
```

10. How to read and write a file using 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.

11. What are all the looping structures in JavaScript?

->

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.

There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

1) JavaScript For loop

The **JavaScript for loop** *iterates the elements for the fixed number of times*. It should be used if number of iteration is known. The syntax of for loop is given below.

```
for (initialization; condition; increment)
{
    code to be executed
}
```

2) JavaScript while loop

The **JavaScript while loop** *iterates the elements for the infinite number of times*. It should be used if number of iteration is not known. The syntax of while loop is given below.

```
while (condition)
{
    code to be executed
}
```

3) JavaScript do while loop

The **JavaScript do while loop** *iterates the elements for the infinite number of times* like while loop. But, code is *executed at least* once whether condition is true or false. The syntax of do while loop is given below.

```
do{
    code to be executed
}while (condition);
```

4) JavaScript for in loop

The **JavaScript for in loop** is used *to iterate the properties of an object*. We will discuss about it later.

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

In this article, we will convert a string into 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**. JavaScript parseInt() function returns Nan(not a number) when the string doesn't contain a number. We can convert a string to javascript by the following methods:

- Using the parseInt() method
- Using the Number() method
- Using the Unary operator

Using the parseInt() method: JavaScript parseInt() Method is used to accept the string and radix parameter and convert it into an integer.
Syntax:
```
parseInt(Value, radix)
```

Using the Number() method: In Javascript, the **Number()** method is used to convert any primitive data type to a number, if it is not convertible it returns NAN.
Syntax:
```
Number(value)
```

Using the Unary Operator: In Javascript, the **Unary operator(+)** is used to convert a string, boolean, and non-string to a number.
Syntax:
```
+op;
```

13. What is the function of the delete operator?

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']
```

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

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:

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Alert</h2>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  alert("I am an alert box!");
}
</script>

</body>
</html>
```

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.

```
<!DOCTYPE html>

<html>

<body>



<h2>JavaScript Confirm Box</h2>




<button onclick="myFunction()">Try it</button>



<p id="demo"></p>



<script>

function myFunction() {

  var txt;
  if (confirm("Press a button!")) {
    txt = "You pressed OK!";
  } else {
    txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script>

</body>
</html>
```

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

->

void means nothing. In a programming language, void means return nothing. “javascript: void(0)” is similar to void.

javascript: void(0) means return undefined as a primitive value. We use this to prevent any negative effects on a webpage 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).

Let’s split javascript: void(0) with a colon. We get javascript and void(0).

javascript: is a pseudo URL. The JavaScript engine interprets this as some code after that colon and executes that code.

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

-> In JavaScript, we can use window.location object to force a page to load another page. We can use the location object to set the URL of a new page.

Window.location.replace

The first way is to use the **window.location.href** property. This property contains information about the current URL of the page, and it can be used to redirect the user to a new page.

Syntax

```
window.location.href = "new_url";
```

The user will be immediately redirected to the specified URL (new_url).

To redirect the user after a specified amount of time has passed, we may also specify the setTimout function which allows the user to redirect to the source URL after the time specified in the function.

```
setTimeout(function() {
  window.location.href = "https://www.tutorialspoint.com";
}, 3000);
```

The above example will redirect the user to the given URL after 3 seconds have passed.

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.
  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.

MODULE: 4 (JavaScript Basic & DOM)