Module (JAVASCRIPT BASIC & DOM) – 4

1) What is JavaScript?

a. JavaScript is a programming language that is commonly used to create interactive effects and dynamic content on websites. It is a client-side language, meaning that it runs in the user's web browser and can be used to manipulate HTML, CSS, and other elements on the page.

2) What is the use of is NaN function?

a.

Definition and Usage

- In javaScript Nan is short for “Not-a-Number”.
- The is Nan() method returns true if a value is NaN.
- The is NaN() method converts the value to a number before testing it.

3) What is negative Infinity?

a. 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.

4) Which company developed JavaScript?

a. JavaScript was developed by Netscape Communications Corporation in 1995, and it was created by Brendan Eich.

5) What are undeclared and undefined variables?

a. An undeclared variable is a variable that has been used in the code but has not been declared using the var, let, or const keyword, while an undefined variable is a variable that has been declared but has not been assigned a value.

Ex. x = 5; // undeclared variable

var y; // declared but undefined variable

console.log(y); // outputs "undefined"

6) Write the code for adding new elements dynamically?

a.

```
<!DOCTYPE html>
<html>
        <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>Adding New Elements</title>
</head>
        <script type="text/javascript">
        function addNode() { var newP = document.createElement("p");
        var textNode = document.createTextNode(" This is a new text node");
        newP.appendChild(textNode);
document.getElementById("firstP").appendChild   (newP); }

        </script>

        </head>

        <body> <p id="firstP">firstP<p> </body>

</html>
```

7) What is the difference between View State and Session State?

a.

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?

a. The strict equality operator (===) checks whether its two operand 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?

a.

1: **Changing CSS with the help of the style property:**

Syntax:

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

Example: In this example, we have built a PAN number validator. First, we will take the input value and match it with a regex pattern. If it matches then using JavaScript add an inline style on the <p> tag. Otherwise, add a different style on the <p> tag.

```
<!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>document</title>
</head>

<body>
    <h1 style="color: green;">
        GeeksforGeeks
    </h1>

    <h2>
        How can the style/class of
        an element be changed?
    </h2>

    <b>Validate Pan Number</b>

    <input type="text" id="pan" />
    <p></p>
    <button id="submit">Validate</button>
    <script>
        const btn = document.getElementById("submit");
        btn.addEventListener("click", function () {
            const pan = document.getElementById("pan").value;
            const para = document.querySelector("p");

            let regex = /([A-Z]){5}([0-9]){4}([A-Z]){1}$/;
            if (regex.test(pan.toUpperCase())) {
                para.innerHTML = "Hurrey It's correct";

                // Inline style
                para.style.color = "green";
            } else {
                para.innerHTML = "OOps It's wrong!";

                // Inline style
                para.style.color = "red";
            }
        });
    </script>
</body>

</html>
```

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`

Example:

- HTML

```
<!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>document</title>

```
    <style>
        .colorBlue {
            color: blue;
        }

        .colorRed {
            color: red;
        }
    </style>
</head>

<body>
    <h1 style="color: green;">
        GeeksforGeeks
    </h1>

    <h2>
        How can the style/class of
        an element be changed?
    </h2>

    <h3>className Example</h3>

    <p class="colorBlue">
        GeeksforGeeks is a computer science portal
        for geeks.This platform has been designed
        for every geek wishing to expand their
        knowledge, share their knowledge and is
        ready to grab their dream job. GFG have
        millions of articles, live as well
        as online courses, thousands of tutorials
        and much more just for the geek inside you.
    </p>

    <button id="submit">Change Color</button>

    <script>
        const btn = document.getElementById("submit");
        const para = document.querySelector("p");
        btn.addEventListener("click", function () {
            para.className = "colorRed";
        });
    </script>
</body>

</html>
```

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

a.

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.

Example:

Following example tries to read the contents of the file populate in the previous example and print it −

```
const fs = require('fs')
fs.readFile('tp.txt', (err, inputD) => {
   if (err) throw err;
      console.log(inputD.toString());
})
```

Output

Following is the output of the above example −

```
You are reading the content from Tutorials Point
```

The text which is displayed in the console is the text which is in the given file.

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.

Example:

Following is an example of the write operation in files in JavaScript.

```
const fs = require('fs')
let fInput = "You are reading the content from Tutorials 
Point"
fs.writeFile('tp.txt', fInput, (err) => {
   if (err) throw err;
   else{
      console.log("The file is updated with the given data")
   }
})
```

11) What are all the looping structures in JavaScript?

a. In JavaScript, there are five main looping structures:

for loop

while loop

do...while loop

for...in loop

for...of loop

Each loop has its own specific use case, and can be used in different scenarios depending on the requirements of your code.

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

a.

. Given a string containing an integer value and along with that user passes a base value. We need to convert that string of any base value to an integer in JavaScript.

```
String          Integer
"1002"            1002
```

For performing the above-illustrated task, we would be using a method (or a function) provided by JavaScript called as **parseInt().**
This is a special method, provided by JavaScript, that takes an integer value (of any base which is either specified or not) and further converts the string into an integer value.

Syntax:

- Following is the syntax that a user may use to convert a string into an integer value (of any base)-

  ```
  parseInt(string_value, base)
  ```

- Alternatively, if we don't want to specify the base value and just want to convert our string value into an integer value itself, then we may use the following syntax also-

  ```
  parseInt(string_value)
  ```

Default value returned by base or radix of parseInt() method is **10.** In other words, if we don't specify any base or radix value then it by default converts the string value to an integer value by taking into regard the base or radix value as 10.
Let us visualize all of the above-illustrated facts with some of the following examples-

Example: In this example, we would be passing the string value in a method (which is explicitly declared for ease purpose) and further that string value is passed inside the parseInt() method which then further converts that string value in the corresponding integer value.

- JavaScript

```
<script>
    let stringConversion = (string_value) => {
      console.log("Initial Type: " + typeof string_value);
      let integer_value = parseInt(string_value);
      console.log("Final Type: " + typeof integer_value);
      console.log(integer_value);
    };

    stringConversion("512000");
    stringConversion("126410");
    stringConversion("0x8975");
</script>
```

Output:

```
Initial Type: string
```

```
Final Type: number
512000
Initial Type: string
Final Type: number
126410
Initial Type: string
Final Type: number
35189
```

13) What is the function of the delete operator?

a.

The `delete` **operator** in JavaScript is used to delete an object’s property.

If it is used to delete an object property that already exists, it returns `true` and removes the property from the object. However, deleting an object property that doesn’t exist will not affect the object, but will still return `true`.

The only time `false` will be returned is when the `delete` operator is used to delete a variable or a function.

Syntax

The syntax for using the `delete` operator is as follows:

```
delete object.property;

// OR

delete object["property"];
```

Parameters

`object`: This is the object whose property we want to delete.

`property`: This is the property to be deleted.

Return value

The `delete` operator returns `true` if the specified property is deleted, or `false` if the property is not deleted.

Code

In the code below, an object is created and the `delete` operator is used to delete some of its properties:

```
let human = {
    name: "John Doe",
    age: 15,
    country: "Nigeria"
}

let dog = {
    name: "Buddy",
    age: 2,
    country : "Germany"
}

// log retured values after delete
console.log(delete human["country"]) // same as human.country
console.log(delete dog.country) // same as dog["country"]

// log affected objects
console.log(human)  // "country" property deleted
console.log(dog)    // "country" property deleted
```

Output

```
true
true
{ name: 'John Doe', age: 15 }
 { name: 'Buddy', age: 2 }
```

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

a. JavaScript has three types of popup boxes: alert(), confirm(), and prompt(). They are used to display messages, confirm actions, and request input from the user, respectively.

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

a.

Using “javascript:void(0);” in anchor tag: Writing “javascript:void(0);” in anchor tag can prevent the page to reload and JavaScript functions can be called on single or double clicks easily. **Example:**

- html

```
<!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>JavaScript:void(0)</title>

</head>

<body>
    <center>
        <h1 style="color:green">GeeksforGeeks</h1>
        <h3>JavaScript:void(0)</h3>
        <a href="javascript:void(0);"
           ondblclick="alert('Welcome to Geeks for Geeks')">
Double click on me </a>
    </center>
</body>

</html>
```

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

a.

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.

- index.html

```
<!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">

</head>


<body>

    <h3>This is the original page</h3>

    <br>


    <button onclick="force_load_gfg()">

        Force Load GFG Page

    </button>
    <br><br>


    <button onclick="force_load_local()">

        Force Load Local HTML page

    </button>


    <script>

        function force_load_gfg() {

            window.location =

                "https://www.geeksforgeeks.org/"

        }


        function force_load_local() {

            window.location =

                "F:/gfg/PageRedirect/newPage.html"

        }

    </script>

</body>


</html>
```

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

- newPage.html

```
<!DOCTYPE html>

<html lang="en">


<head>
```

```html
    <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?

a. Using innerHTML in JavaScript has some disadvantages including potential security vulnerabilities, slower performance compared to other DOM manipulation techniques, loss of event listeners, and reduced code maintainability. While innerHTML can be useful in certain situations, it should be used judiciously, and its limitations and potential pitfalls should be carefully considered.