Module 6) WD - JAVASCRIPT BASIC & DOM

1. What is JavaScript?

Ans. JavaScript is the Programming Language for the Web.

- ➔ JavaScript can update and change both HTML and CSS.JavaScript can calculate, manipulate and validate data.
- ➔ JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.
- ➔ It is an interpreted programming language with object-oriented capabilities.

2. What is the use of isNaN function?

Ans. 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.

```
<!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>
```

3. What is negative Infinity?

Ans. 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.

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 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 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.

Example.

```
<!DOCTYPE html>

<html>


<body>

        <style>

                h1 {

                        color: green;

                }

        </style>
        <h1>
        What is negative infinity in JavaScript?
</h1>
        <button onclick="geekNegativeInfinity()">
        Generate negative infinite
</button>
        <p id="geek"></p>
        <script>
                function geekNegativeInfinity() {
                        //negative value greater than the
                        //largest representable number in JavaScript
                        var n = (-Number.MAX_VALUE) * 2;
                        document.getElementById("geek").innerHTML = n;
                }
        </script>
</body>
</html>
```

4. Which company developed JavaScript?

Ans. 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?

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.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
<title>Document</title>
<style>
  body {
    font-family: "Segoe UI", Tahoma, Geneva, Verdana, sans-serif;
  }
  .result {
    font-size: 18px;
    font-weight: 500;
    color: blueviolet;
  }
</style>
</head>
<body>
<h1>Undeclared vs Undefined</h1>
<div class="result"></div>
<div class="result"></div>
```

```
<button class="Btn">Click here</button>
<h3>Click on the above button to access undeclared and undefined variable
</h3>
<script>
   let BtnEle = document.querySelector(".Btn");
   let resEle = document.querySelectorAll(".result");
   let a;
   BtnEle.addEventListener("click", () => {
      resEle[0].innerHTML += "Accessing undefined variable = " + a;
      try {
         resEle[1].innerHTML = b;
      } catch (err) {
          resEle[1].innerHTML = "Accessing undeclared variable = " + err;
      }
   });
</script>
</body>
</html>
```

6. Write the code for adding new elements dynamically?

Ans. HTML documents can be easily accessed and manipulated using HTML DOM , which represents an HTML document as a tree-structure. When an HTML document is loaded in a browser window, it becomes a Document object. With document.createElement() method you can create a specified HTML element dynamically in JavaScript. After creation you can add attributes. If you want the element to show up in your document, you have to insert in into the DOM-tree of the document.

```
<html>
 <body>
  <button onclick="create()">Create Heading</button>
  <script>
   function create() {
    var h1 = document.createElement('h1');
    h1.textContent = "New Heading!!!";
    h1.setAttribute('class', 'note');
    document.body.appendChild(h1);
   }
  </script>
 </body>
</html>
```

So document.createElement is used with an HTML tag to create the element. The textContent is then modified and then the class attribute is modified using setAttribute . This could also be used to add a data attribute or any other kind of attribute, like you can in HTML. Finally the element is appended to the body using the body element's appendChild method.

Actually, It's essentially equivalent to < H1 class="note" > New Heading!!! < /h1 > .

The power of what we seen today is that we are creating and styling elements "on the fly". It's your decision to either add all the elements on the page while designing or you can create and insert HTML elements at runtime, that is dynamically using the createElement() method.

7. What is the difference between ViewState and SessionState?

Ans. The values of controls of a particular page of the client browser is persisted by ViewState at the time of post back operation is done. If the user requests another page, the data of previous page is no longer available.

- The data of a particular server persists in the server by SessionState. The availability of the user data is up to the completion of a session or closure of the browser.

A ViewState is a state of a page within a browser wherein the values of controls persist when post back operation is done.

When another page is loaded, the previous page data is no longer available.

- SessionState is the data of a user session and is maintained on the server side. This data available until user closes the browser or session time-outs.

8. What is === operator?

Ans. 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.

Example:-

```
console.log(1 === 1);
// expected output: true

console.log('hello' === 'hello');
// expected output: true

console.log('1' ===  1);
// expected output: false

console.log(0 === false);
// expected output: false

> true
> true
> false
> false
```

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?

Ans. 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">


<body>

        <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>

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

Ans.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.

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.

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

```
const fs = require('fs')

let fInput = "You are reading"

fs.writeFile('tp.txt', fInput, (err) => {
  if (err) throw err;
  else{
    console.log("The file is updated with the given data")
  }
})
```

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.

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());
})
```

11. What are all the looping structures in JavaScript?

Ans. 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:

Instead of writing:

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For Loop</h2>

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

<script>
const cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];
```

```
let text = "";
for (let i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

Output.

```
JavaScript For Loop
BMW
Volvo
Saab
Ford
Fiat
Audio
```

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

Ans. To convert a string to an integer parseInt() function is used in javascript. parseInt() function returns Nan(not a number) when the string doesn’t contain number. If a string with a number is sent then only that number will be returned as the output. This function won't accept spaces. If any particular number with spaces is sent then the part of the number that presents before space will be returned as the output.

syntax

parseInt(value);

This function takes a string and converts it into an integer. If there is no integer present in the string, NaN will be the output.

Example

In the following example, various cases of strings such as only strings, strings with numbers, etc have been taken and sent through the parseInt() function. Later on, their integer values, if present, have displayed as shown in the output.

```
<html>

<body>

<script>

  var a = "10";

  var b = parseInt(a);

  document.write("value is " + b);

  var c = parseInt("423-0-567");

  document.write("</br>");

  document.write('value is ' + c);

  document.write("</br>");

  var d = "string";

  var e = parseInt(d);

  document.write("value is " + e);

  document.write("</br>");

  var f = parseInt("2string");

  document.write("value is " + f);

</script>
```

```
</body>
</html>
```

Output

```
value is 10
value is 423
value is NaN
value is 2
```

13. What is the function of the delete operator?

Ans.

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.

```
const Employee = {
  firstname: 'John',
  lastname: 'Doe'
};

console.log(Employee.firstname);
// expected output: "John"

delete Employee.firstname;

console.log(Employee.firstname);
// expected output: undefined
```

Output.

```
> "John"
> undefined
```

Syntax

```
delete object.property
delete object[property]
```

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

Ans: JavaScript has three kind of popup boxes =>

- ➔ **Alert box :**

  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.

  Example

  ```
  <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>
```

➔ **Prompt box.**

A prompt box is often used if you want the user to input a value before entering a page.

```
<h2>JavaScript Prompt</h2>


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


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


<script>
function myFunction() {
  let text;
    let person = prompt("Please enter your name:", "Harry Potter");
  if (person == null || person == "") {
    text = "User cancelled the prompt.";
  } else {
    text = "Hello " + person + "! How are you today?";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>
```

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

Ans : JavaScript void 0 means returning undefined (void) as a primitive value. You might come across the term "JavaScript:void(0)" while going through HTML documents. It is used to prevent any side effects caused while inserting an expression in a web page.

```
<!DOCTYPE html>

<html>

<head>

        <title>without using JavaScript:void(0)</title>

</head>

<body align="center">

        <h2>This is without using JavaScript:void(0)</h2>

        <a href="#" ondblclick="alert('Task completed!')">Double Click
Me!</a>

</body>

</html>
```

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

Ans : 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.

Syntax:

```
<script>

   window.location = <Path / URL>

</script>
```

Example:

```
<script>
```

```
        window.location.href =” new_url”;
</script>
```

16. What are the disadvantages of using innerHTML in JavaScript?

Ans. Event handlers attached to any DOM element are preserved.

- ➔ Replacement is done everywhere.
- ➔ It is not possible to append innerHTML.
- ➔ Breaks the document.
- ➔ Used for Cross-site Scripting.