**Module 6: JAVASCRIPT BASIC & DOM**

**1. What is JavaScript?**

**A:** JavaScript is a programming language that developers use to make interactive webpages. From refreshing social media feeds to displaying animations and interactive maps, JavaScript functions can improve a website's user experience. As a client-side scripting language, it is one of the core technologies of the World Wide Web. For example, when browsing the internet, anytime you see an image carousel, a click-to-show dropdown menu, or dynamically changing element colors on a webpage, you see the effects of JavaScript.

**2. What is the use of isNaN function?**

**A:** The JavaScript isNaN() Function is used to check whether a given value is an illegal number or not. It returns true if the value is a NaN else returns false. It is different from the Number.

**3. What is negative Infinity?**

**A:** NEGATIVE\_INFINITY is a special numeric value that is returned when an arithmetic operation or mathematical function generates a negative value greater than the largest representable number in JavaScript (i.e., more negative than -Number. MAX\_VALUE) . JavaScript displays the NEGATIVE\_INFINITY value as -Infinity .

**4. Which company developed JavaScript?**

**A:** 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?**

**A: 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.

<!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?**

A: <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <h2>JavaScript HTML DOM</h2>

<p>Add a new HTML Element.</p>

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

const para = document.createElement("p");

const node = document.createTextNode("This is new.");

para.appendChild(node);

const element = document.getElementById("div1");

element.appendChild(para);

</script>

</body>

</html>

**7. What is the difference between ViewState and SessionState?**

**A: Differences 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?**

**A:**In javascript === is strict equality operator. Used to compare two variable and check both value and datatype. If both datatypes and value matches of two variables it will return Boolean result (true or False).

**Example:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p id="demo"></p>

<script>

  var x = 10;

  document.getElementById("demo").innerHTML = (x === 10);

</script>

</body>

</html>

**Output:**

**True**

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

**A: You can use the below-mentioned methods to add classes, remove classes, and toggle between different classes respectively.**

1. **The add() method**: It adds one or more classes.
2. **The remove() method:** It removes one or more classes.
3. **The toggle() method:** If the class does not exist it adds it and returns true.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <style>

        .one{

            color: blue;

        }

        .colorblack{

            color: black;

        }

</style>

<p class="one">Lorem ipsum dolor sit amet consectetur adipisicing elit. Labore sapiente dolore sequi libero fuga ab similique

        explicabo fugiat eos, atque, obcaecati quam voluptas animi ipsa iusto accusantium velit enim alias.</p>

<button id="click">click</button>

<script>

            const button =document.getElementById("click");

            const para = document.querySelector("p");

            button.addEventListener("click",function () {

                para.className= "color black"

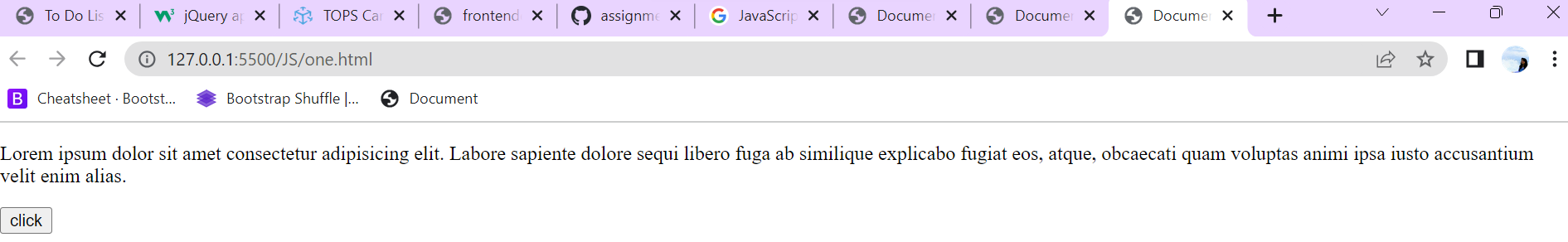
            })

</script>

</body>

</html>

**Output:**



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

**A:** 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 −

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: 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:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script>

        for(let i = 0; i <=5; i++){

            console.log("tops");

        }

        i=0

        while (i<5) {

            console.log("tops");

            i++

        }

        i=0

        do{

            console.log("tops");

            i++

        }while (i<5);

       </script>

</body>

</html>

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

**A:** To convert a string to an integer parseInt(), Number(), and Unary operator(+) 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.

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

**A:** Using the delete operator on an object deallocates its memory. A program that dereferences a pointer after the object is deleted can have unpredictable results or crash.

**Example:**

let emp = {

firstName: "parag",

lastName: "gorad",

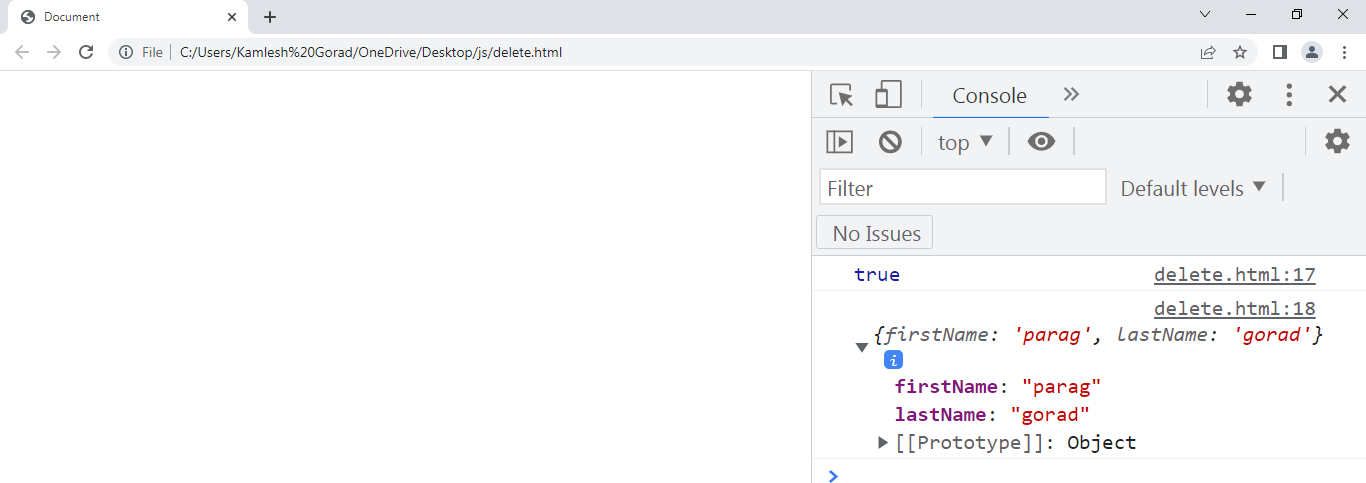
salary: 40000

}

console.log(delete emp.salary);

console.log(emp);

**output:**

****

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

**A:** JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

**Example:**

## **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.

<!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>

    <p id="one"></p>

    <button onclick="a()">jinal</button>

    <script>

        function a() {

            alert("jinal")

        }

    </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**.

<!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>

    <p id="one"></p>

    <button onclick="a()">jinal</button>

    <script>

        var txt

        function a() {

            if(confirm("jinal"))

            {

                txt="ok"

            }

            else{

                txt="no"

            }

            document.getElementById("one").innerHTML=txt

        }

    </script>

</body>

</html>

## **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.

<!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>

    <p id="one"></p>

    <button onclick="a()">jinal</button>

    <script>

        var txt

        function a() {

            let per = prompt("jinal")

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

                txt = "plz enter your name"

            }

            else{

                txt = "hiii" + per

            }

            document.getElementById("one").innerHTML = txt

        }

    </script>

</body>

</html>

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

**A: javascript void(0)**is an operator that executes an expression without reloading the web page and removes unwanted effects from the web page. It always returns an undefined primitive value.

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

**A:** 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.

**17. What are the disadvantages of using innerHTML in JavaScript?**

**A: 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.
* **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.
* **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.