Practical No. 01

Title:- Write Javascript program using document.write().

Theory:

* document.write():

Document.write() is a function in Javascript that used to display some strings in the output of HTML webpage. We know that in Javascript, a name followed by parantheses () is known as Function or Method. So document write() is a method. Document write() is a method use to display text in the browser window.

- Syntax:-

document.write (exp1 .... expN)

- Program:-

```
<html>
<head> <title> String Display </title> </head>
<body>
< script language = "Javascript" type = "text/Java
                                        script">
        document.write (" Hello World! ")
</script>
</ body> </html>
```

Conclusion:- Hence, we learnt a program of basic javascript using document.write() method

java.html

File Edit View

```
<html>
<head>
<title>basic pgm</title></head>
<body>
<script language="JavaScript" type="Text/JavaScript">
document.write("Hello World");
</script>
</body>
</html>
```

basic pgm

File C:/Users/GAyatri/...

Hello World

Practical No.02

Title:- Write a Javascript program to the sum of two numbers using var.

Theory:-

Variable Declaration:

Var:- This Keyword is used to declare variables globally. If we use this keyword to declare a variable then the variable can be accessed globally and changeable also.

Syntax:-

```
Var variablename = value.
```

let:- This keyword is used to declare variables locally. If we use this keyword to declare a variable then the variable can be accessed locally and it is changeable as well.

Syntax:-

```
let variablename = value name
OR
let variable name;
```

const:- This keyword is used to declare variable locally. The variable can be used only inside the block. The variables declared using const values can't be reassigned. So we should assign the value declaring the variable.

Syntax:-
```
const variablename = "Value";
```

* Program:-

Q. Write a program to find the sum of two numbers using var.

→
```
<html>
<head><title>Sum of No.s</title></head>
<body>
<script language="Javascript" type="text/Javascript">
var a;
var b; var sum;
a prompt("Enter the value of a:");
b prompt("Enter the value of b:"); a:to;
b:to; sum = a+b;
alert("The sum of the numbers is:"+sum);
</script>
</body>
</html>
```

Conclusion :- Hence we learnt program using if else to do sum of two numbers

```
<!DOCTYPE html>
<html>
<head>
    <title>Sum of 2 Numbers</title>
</head>
<body>
    <script type="text/JavaScript">
        var a;
        var b;
        var sum;

        a = prompt("Enter the value of a=");
        b = prompt("Enter the value of b=");


        a = +a;
        b = +b;


        sum = a + b;

        alert("The sum of a & b is = " + sum);
    </script>
</body>
</html>
```

Taking input from user

Output.

- document.links :- return collection of all <a> elements in the document that have href attributes
  e.g. console.log (document.links);

- Common Methods of 'document' object

- document.getElementById (id);
  → returns element with the specified 'id'.

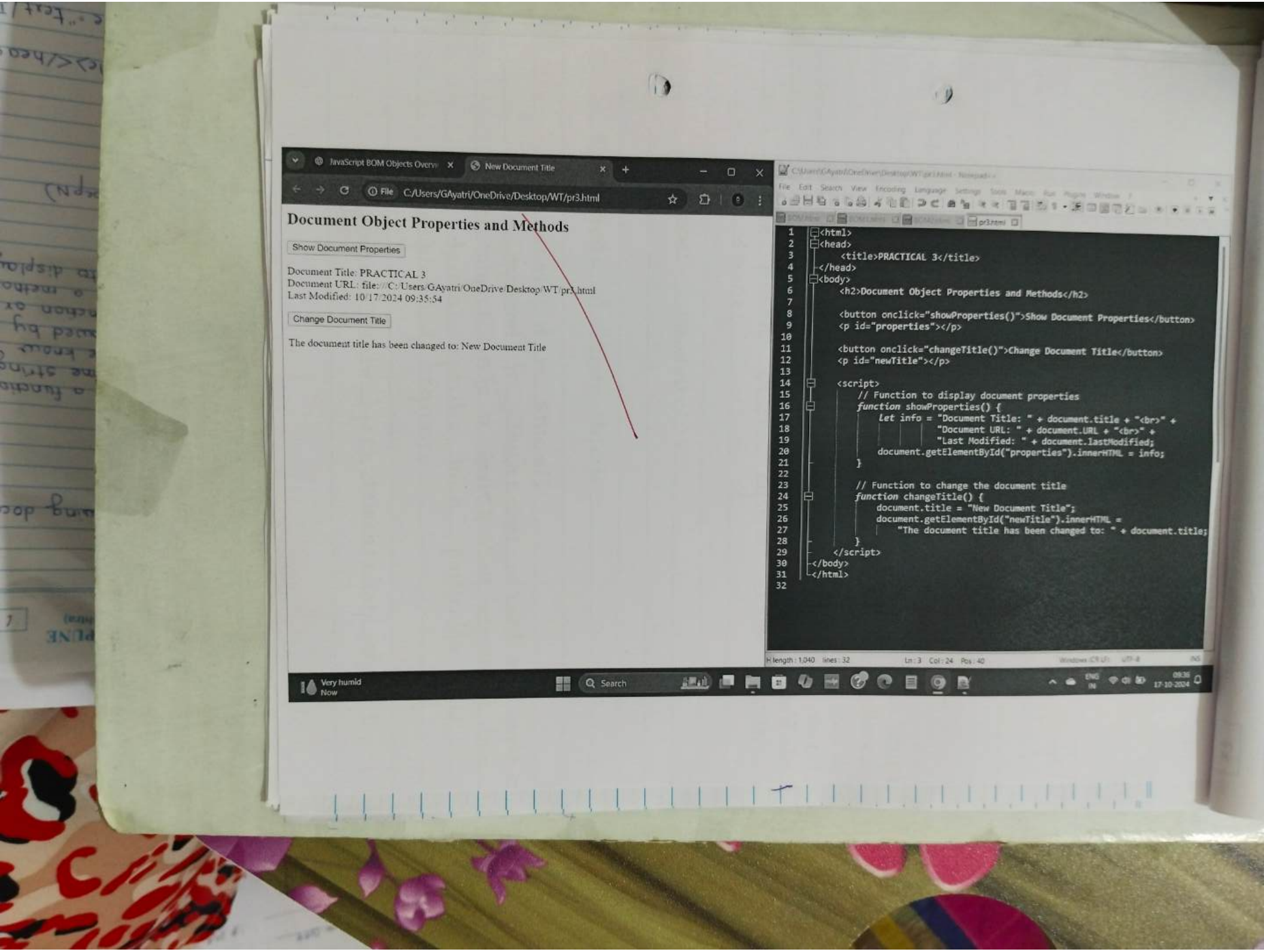
- document.getElementByClassName ("className");
  → returns a collection of all elements with the specified class name.

- document.getElementByTagName ("tagName");
  → returns a collection of all elements with the specified tagname.

- document.write ("text");
  → write HTML expressions or JavaScript code to the documents.

Conclusion :- Hence we learnt properties & methods of document object in this practical.

Sign


JavaScript BOM Objects Overvi
New Document Title
File C:/Users/GAyatri/OneDrive/Desktop/WT/pr3.html
Document Object Properties and Methods
Show Document Properties
Document Title: PRACTICAL 3
Document URL: file:///C:/Users/GAyatri/OneDrive/Desktop/WT/pr3.html
Last Modified: 10/17/2024 09:35:54
Change Document Title
The document title has been changed to: New Document Title
pr3.html
1 <html>
2 <head>
3 <title>PRACTICAL 3</title>
4 </head>
5 <body>
6 <h2>Document Object Properties and Methods</h2>
7
8 <button onclick="showProperties()">Show Document Properties</button>
9
10
11 <button onclick="changeTitle()">Change Document Title</button>
12
13
14 <script>
15 // Function to display document properties
16 function showProperties() {
17 let info = "Document Title: " + document.title + "
" +
18 "Document URL: " + document.URL + "
" +
19 "Last Modified: " + document.lastModified;
20 document.getElementById("properties").innerHTML = info;
21 }
22
23 // Function to change the document title
24 function changeTitle() {
25 document.title = "New Document Title";
26 document.getElementById("newTitle").innerHTML =
27 "The document title has been changed to: " + document.title;
28 }
29 </script>
30 </body>
31 </html>
32
Very humid
Now
Search

Practical No. 04

Title:- Write a JavaScript program that displays todays date & time.

Theory:

1. Date object (newdate()):-

A built in JavaScript object that represents a single moment in time.

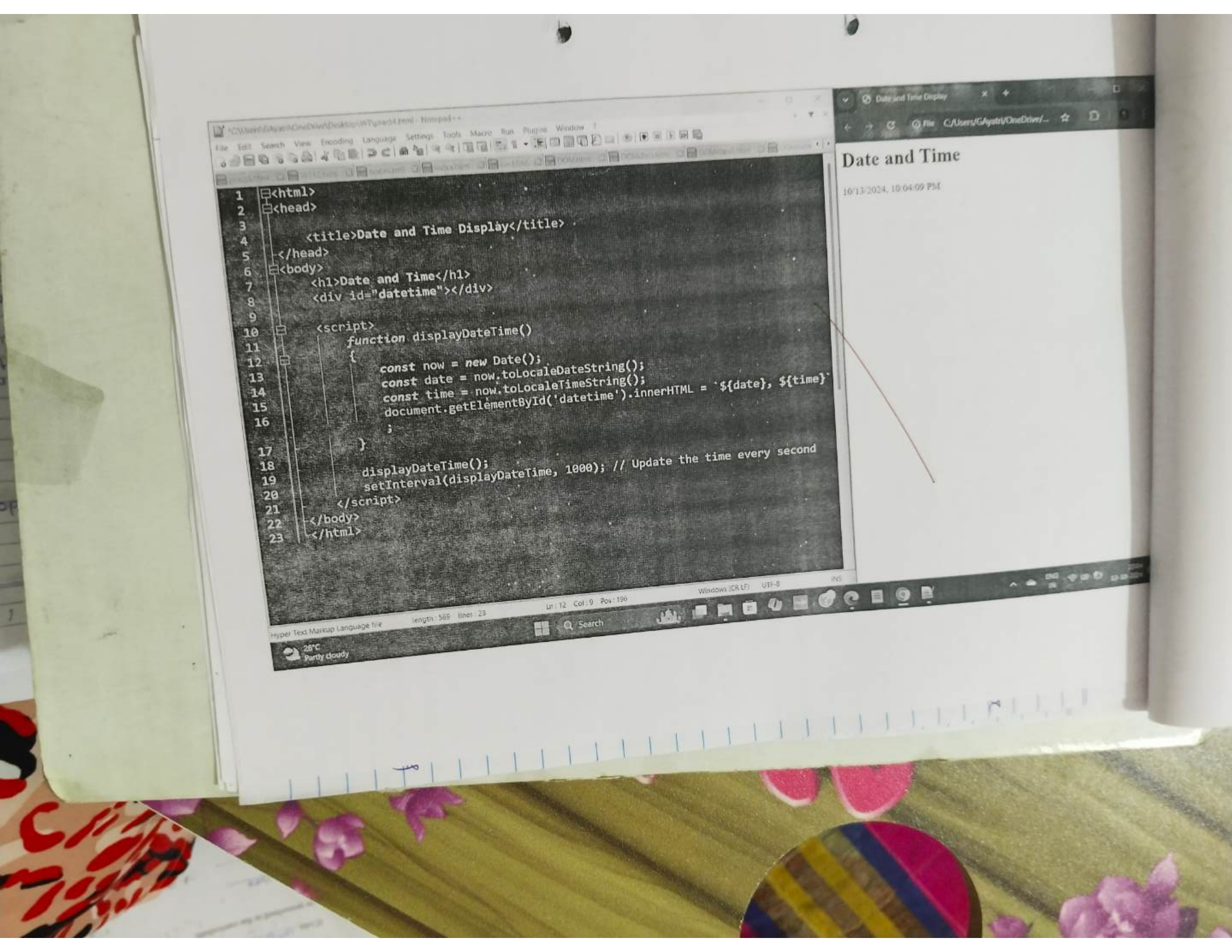
It contains methods to retrive the current object using newdate(), it returns the current date & time based on the system clock of the user device.

Syntax:- NewDate();

2. Methods of Date objects:-

- To locate date string();
- This method in JavaScript is used to convert a date object into a string that represents date in format specific to a particular local (language & region).
-

Syntax:- dateobject.tolocaleDate string()


Date and Time Display
Date and Time
10/13/2024, 10:04:09 PM
<html>
<head>
<title>Date and Time Display</title>
</head>
<body>
<h1>Date and Time</h1>

<script>
function displayDateTime()
{
const now = new Date();
const date = now.toLocaleDateString();
const time = now.toLocaleTimeString();
document.getElementById('datetime').innerHTML = `${date}, ${time}`
;
}
displayDateTime();
setInterval(displayDateTime, 1000); // Update the time every second
</script>
</body>
</html>
Hyper Text Markup Language file
length : 569 lines : 23
Ln : 12 Col : 9 Pos : 196
Windows (CR LF)
UTF-8
INS
Search
28°C
Partly cloudy

Practical No. 05

Title: Write a Javascript program to computer the day of the week while you, input the date within the prompt dialog box.

Theory:

1. Date Object Initialization:
When you create a new 'date' object using 'NewDate(userinput)'. JS tries to presue the 'userInput' string into a Data.

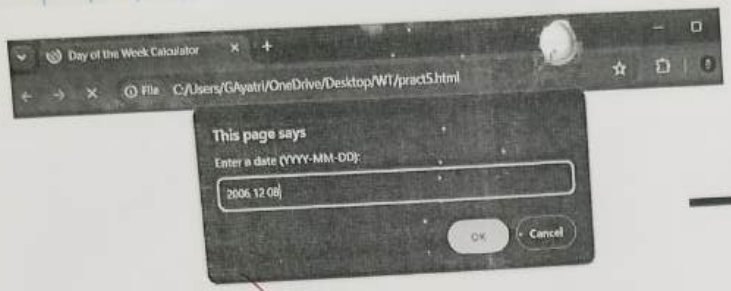
2. Validating Dates:-
To ensure the data is Valid you can check if 'date.gettime()' return 'Now (Not a No.) indicates that the date object could't be created from input.

3. Creating Day of the week:-
Getplay() method of the date object returns an integer between 0 & 6, where 0= sunday & 1-6 is Monday to Saturday.

4. Mapping Days to Names:

an Array day of week map these integer values to human-readable day names.

Conclusion: Olence are performed practical to compute day of week in Javascript.




```html
<html>
<head>
    <title>Day of the Week Calculator</title>
</head>
<body>
    <h1>Day of the Week Calculator</h1>

    <script>
        function getDayOfWeek()
    {
        const dateInput = prompt("Enter a date (YYYY-MM-DD):");
        if (dateInput)
            {
            const date = new Date(dateInput);
                if (!isNaN(date.getTime()))
                {
                    const daysOfWeek = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];
                    const dayIndex = date.getDay();
                    const dayName = daysOfWeek[dayIndex];
                    alert(`The day of the week for ${dateInput} is ${dayName}.`);
                }
                else
                {
                    alert("Invalid date format. Please use YYYY-MM-DD.");
                }
            }
        else
        {
            alert("No date entered.");
        }
    }
        getDayOfWeek();
    </script>
</body>
</html>
```

Practical No. 06

Title:- Write a JavaScript program to generate the table of numbers using for loop statement.

Theory:-

Javascript loops are essential for efficiently handling repetitive tasks. They execute a block of code repeatendly as long as specified condition remains true.

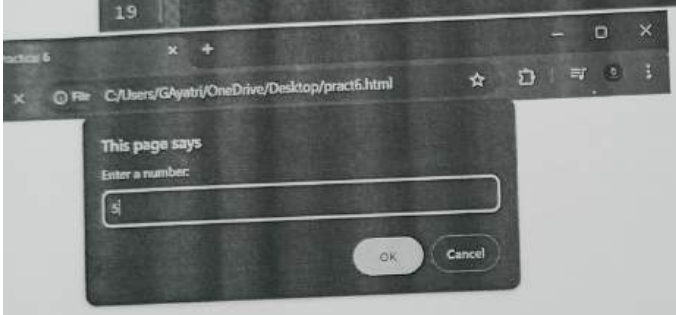
For loop:-

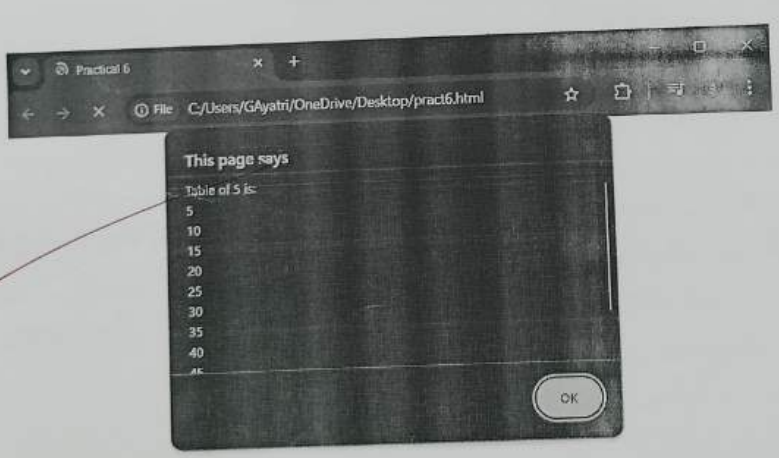
The Javascript for loops provides a concise way of writting the loop structures. The for loop contains initialization, condition & increement/ decrement in one line thereby providing a shorter easy to debug structure of looping.

```
for (initialization; condition; increament/decreament)
{
    statement;
}
```

Conclusion: Hence we performed this practical and generated the table of numbers using for loop statement.

```html
<html>
<head><title>Practical 6</title></head>
<body>
<script>
var a;
var table = "";

a = prompt("Enter a number:");

for (let i = 1; i <= 10; i++)
{
  table += a * i + "\n";
}

alert("Table of " + a + " is:\n" + table);
</script>
</body>
</html>
```





Practical No. 07.

Title:- Write a Javascript program to generate the first 10 fibonacci numbers.

Theory:- We can implement fibonacci series using for loop, while loop, do-while loop & by using recursion.

In this practical we are using 'for loop' for fibonacci series.

The for loop approach calculates the fibonacci series by iteratively summing the previous two numbers, starting from 0 & 1. This method efficiently generates each fibonacci number upto desired term.

Fibonacci series is a series that generates subsequent series of number by the addition of the previous numbers.

$$fn = (fn-1) + (fn-2)$$

Where, fn represents the addition of previous terms (fn-1) & (fn-2). Here fn-1 is the first term & fn-2 is the second term of fibonacci series.

Example:-

- first term :- 0
- Second term:- 1
- Third term:- (0+1) = 1.

- fourth term :- (1+1) = 2
- fifth term :- (1+2) = 3

generated fibonacci series:- 0, 1, 1, 2, 3...
Similary we can find next terms.

When user gives input of the limit of fibonacci series, firstly the complixer prints 0 & 1 as they are num1 & num2. And then the program get the nextterm which is num1 + num2. After it the program goes into the loop, nextterm gets calculated & also program's compiler prints it. After terminating of the loop the program stops.

Conclusion:- Hence we performed this practical & learned that how to generate first 10 fibonacci numbers.

Practical No. 08

Title:- Create a HTML document that illustrate the function defination & calling of welcome() function.

Theory:-

A Javascript function is a block of code designed to perform a particular task. It gets executed when someone invokes it (calls it). It is defined with the function keyword, followed by a name & parantheses()

syntax:- function functionname()
{ // statements }

- Function Invocation: function code you have written will be executed whenever it is called. Triggered by an event (e.g. button click by user) when explicitly called from javascriplt code.
Automatically executed, such as inself invoking function.

- Funtion Defination: A function declaration or function statement. 1. Every function should begin with keyword function 2. A user defined function name should be unique. 3. A list of parameters enclosed within parantheses & separated by comma. 4. list of statements composing the body of function enclosed within cuelly braces {}.

- Function Expression:- It is similar to a function declaration without function name. function expression can be stored in a variable argument.

```
let var = function (parameters)
{ // set of statements
```

- Calling function:- We can call a function by using function name separately by the value of parameters endosed between parantheses & semicolon at end.

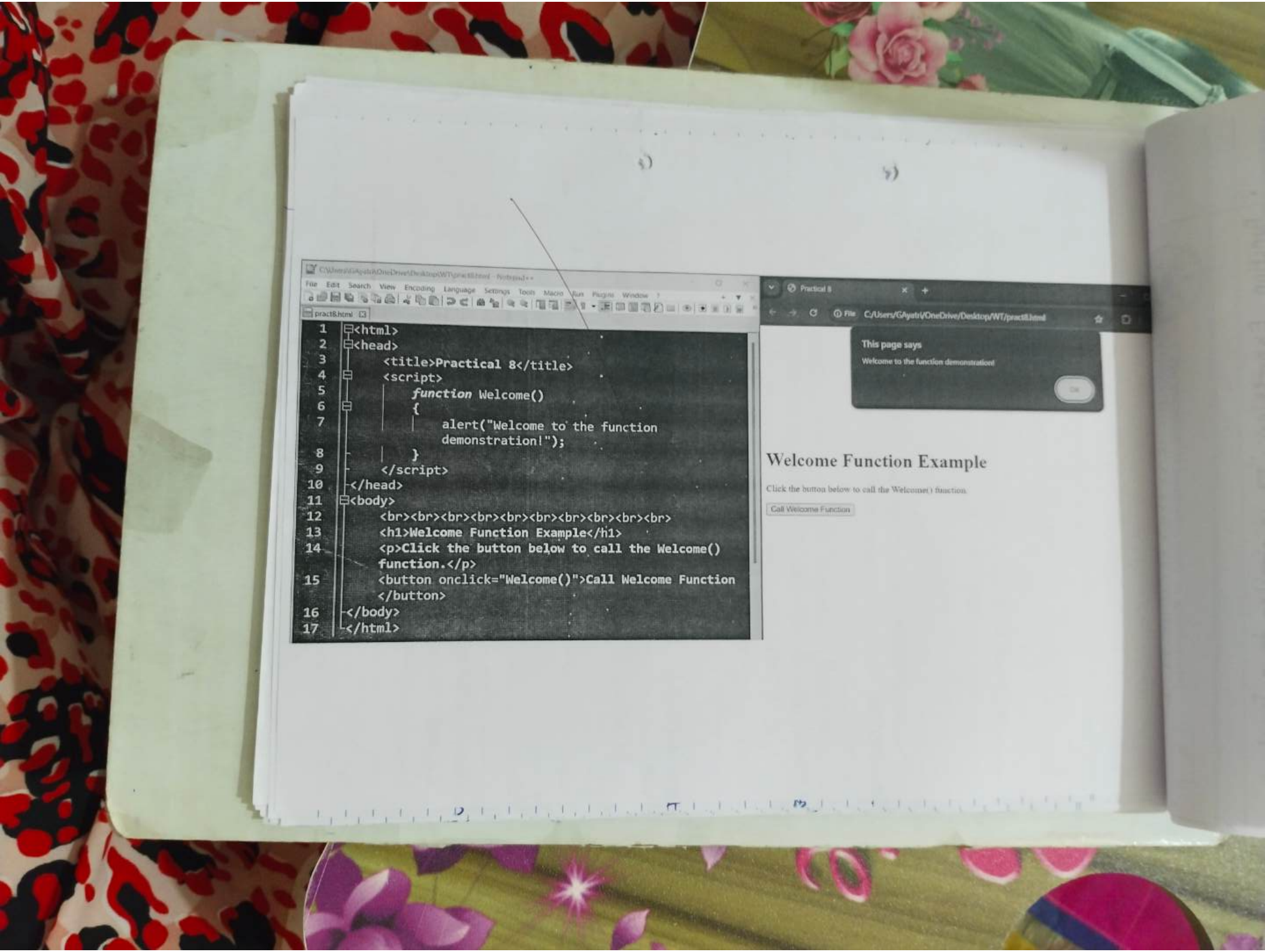
```
functionName(parameter list);
```

- Return Statement:- An optional & most of the time the last statement in a javascript pgm. To return same values from a function after performing same operations:

```
return returnvalue;
```

Conclusion:- Hence, we performed this practical & learned to illustrate the function defination & calling of welcome() function.

Sign


pract8.html
1 <html>
2 <head>
3 <title>Practical 8</title>
4 <script>
5 function Welcome()
6 {
7 alert("Welcome to the function demonstration!");
8 }
9 </script>
10 </head>
11 <body>
12

13 <h1>Welcome Function Example</h1>
14 Click the button below to call the Welcome() function.
15 <button onclick="Welcome()">Call Welcome Function</button>
16 </body>
17 </html>
Practical 8
This page says
Welcome to the function demonstration!
OK
Welcome Function Example
Click the button below to call the Welcome() function.
Call Welcome Function

8:42

rohan_date_rd 1h

Beast Inside Beats • Vibes

Added🥰🥂

Send message

Practical No. 09

Title:- Write javascript programs based on decision making statements.

Theory:-

The conditional statement evaluates a condition before execution of instruction.

- If statement:- One of the simplest decision making statement which is use to decide whether a block of javascript code will execute if a certain condition is true.

```
if (condition) { // block of code}
```

- If ... else statement:- This contains 2 blocks that if block & else block else statement is use for specifying the execution of block of code if condition is false.

```
if ( condition)
{ // code }
else { code}
```

- if ... else if:- It's use to test multiple condition if statement can have multiple or zero else if statements before using else statement.

```
if (condition) { // code }
else if (condition) { // code}...
else { // code}
```

- Nested if:- An if statement inside an if statement.

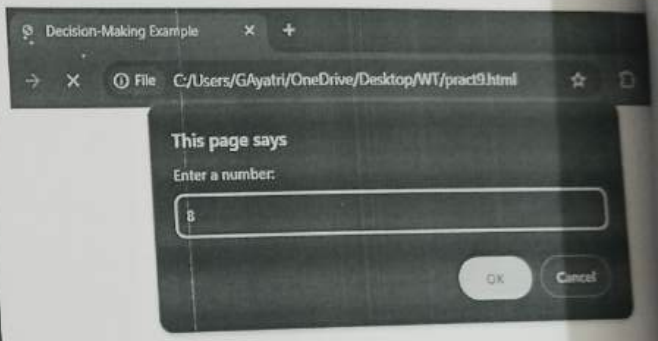
```
if (condition 1)
{ // code }
if (condition 2)
{ // code }
else { // code } }
```

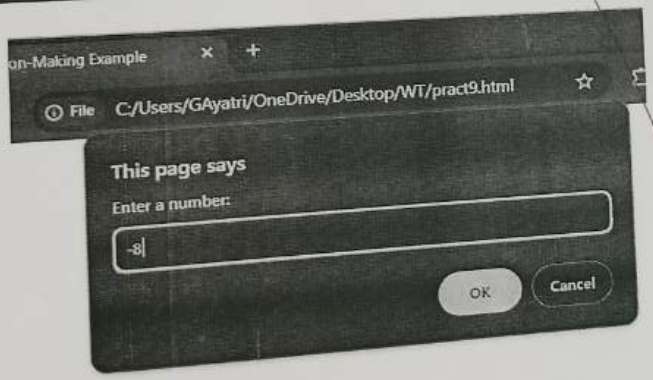
- Switch:- Multiway branch statement use when all branches depend upon the value of single variable. It wes break & default keyword.

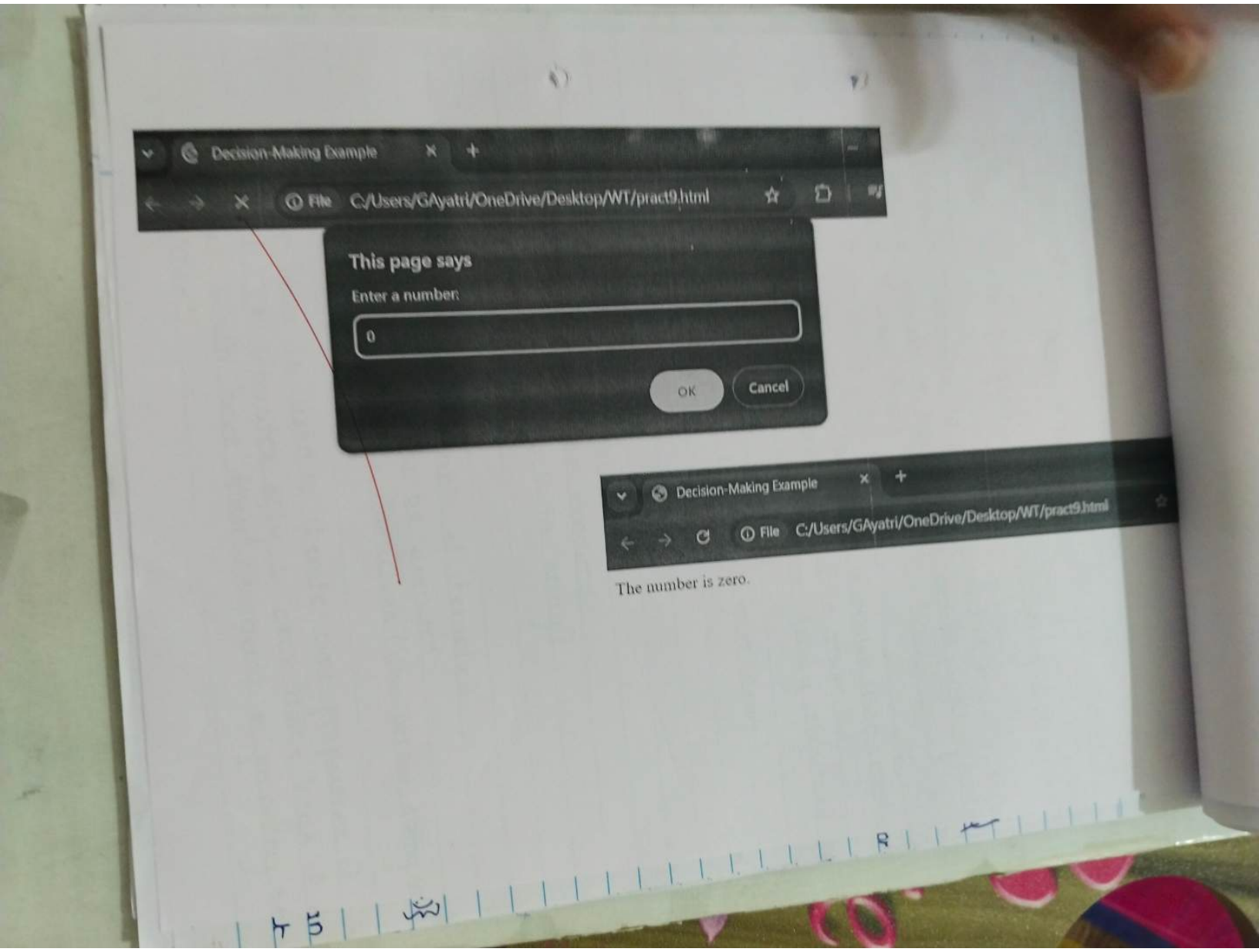
```
switch (expression)
{ case 'value' : // code
  break; ...
  default : // code }
```

Conclusion:- Olence, we performed this practical & learned to make programs based on decision making statements.

```
<head><title>Decision-Making Example</title></head>
<body>
<script>
var number = prompt("Enter a number:");

if (number > 0) {
    document.write("The number is positive.");
} else if (number < 0) {
    document.write("The number is negative.");
} else if (number == 0) {
    document.write("The number is zero.");
} else {
    document.write("Invalid input. Please enter a
    valid number.");
}
</script>
</body>
</html>
```






Decision-Making Example
File C:/Users/GAyatri/OneDrive/Desktop/WT/pract9.html
This page says
Enter a number:
0
OK
Cancel
Decision-Making Example
File C:/Users/GAyatri/OneDrive/Desktop/WT/pract9.html
The number is zero.

Practical No. 10

Title:- Write program based on looping statement.

Theory:- Javascript loops are essential for efficiently handling repetitive task. they repeat until specified condition become false.

- for loop:- for loop contains initialization, condition & encrement / decrement in 1 line there by providing a shortest, easy to debug structure of looping.

```
for (initialization; condition; incre / decre)
{ // code }
```

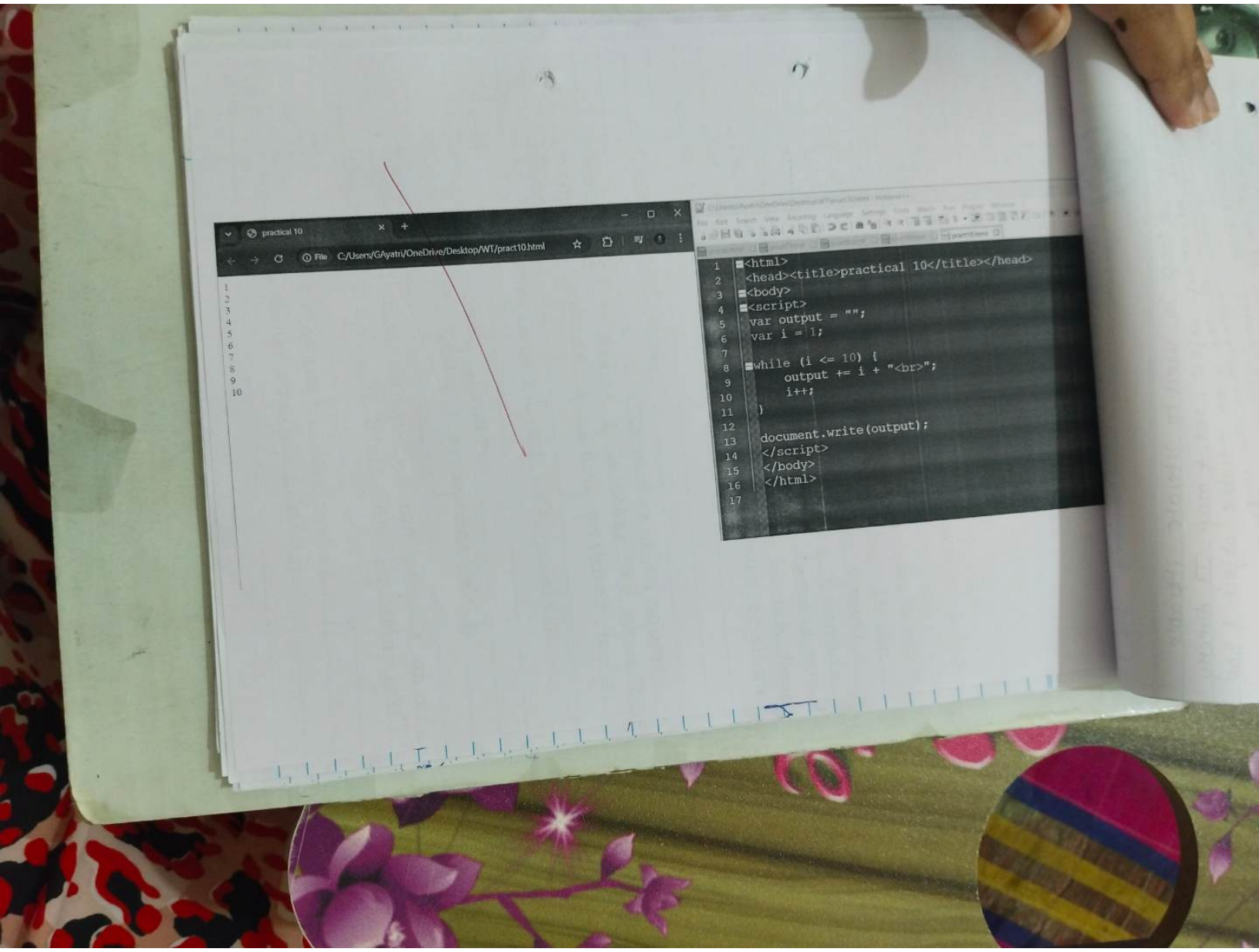
- for in loop:-

```
for (key in object)
{ // code to be executed;
  // also work with array }
```

- for of loop:-

```
for (variable of iterable)
{ // code to be executed;
  // iterable structure (like array, string etc) }
```

- for in loop is used to iterate over properties of object. It literates only 1 over those keys of an object which have their enumerable property set to "true".


practical 10
File C:/Users/GAyatri/OneDrive/Desktop/WT/pract10.html
<html>
<head><title>practical 10</title></head>
<body>
<script>
var output = "";
var i = 1;
while (i <= 10) {
output += i + "
";
i++;
}
document.write(output);
</script>
</body>
</html>

Practical No. 11

Title:- Write JavaScript program based on arrays.

Theory:- An array is a data structure use to store multiple values in a single variable. It can hold various datatypes & allow for dynamic resizing. We can create JS array using 2 methods. using the array constructor or the shorthand array literal syntax, which is just square blackets. Arrays are flexible in size, so they can growe or shrink as you add or remove elements.

- Declaration of Array:-
  It involves using square brackets [] to define & initialize the array. The method is concise & widely preffered for its simplicity.

```
let arrayName = [val1, val2, ... valn];
```

- Accessing array elements:-

```
let value = arrayName[index];
```

- Initialization of array:- we can initialize an array by directly giving it values into squarebracket[].

```
let arrayName = [value1, value2, -- ];
let value = arrayName[index];
```

- Types of array:-

1. Numeric Array:- An array that contains only the numeric values as elements, that are stored at different memory locations but in a consecutive manner.

```
let num = [1,2,3,4,5];
```

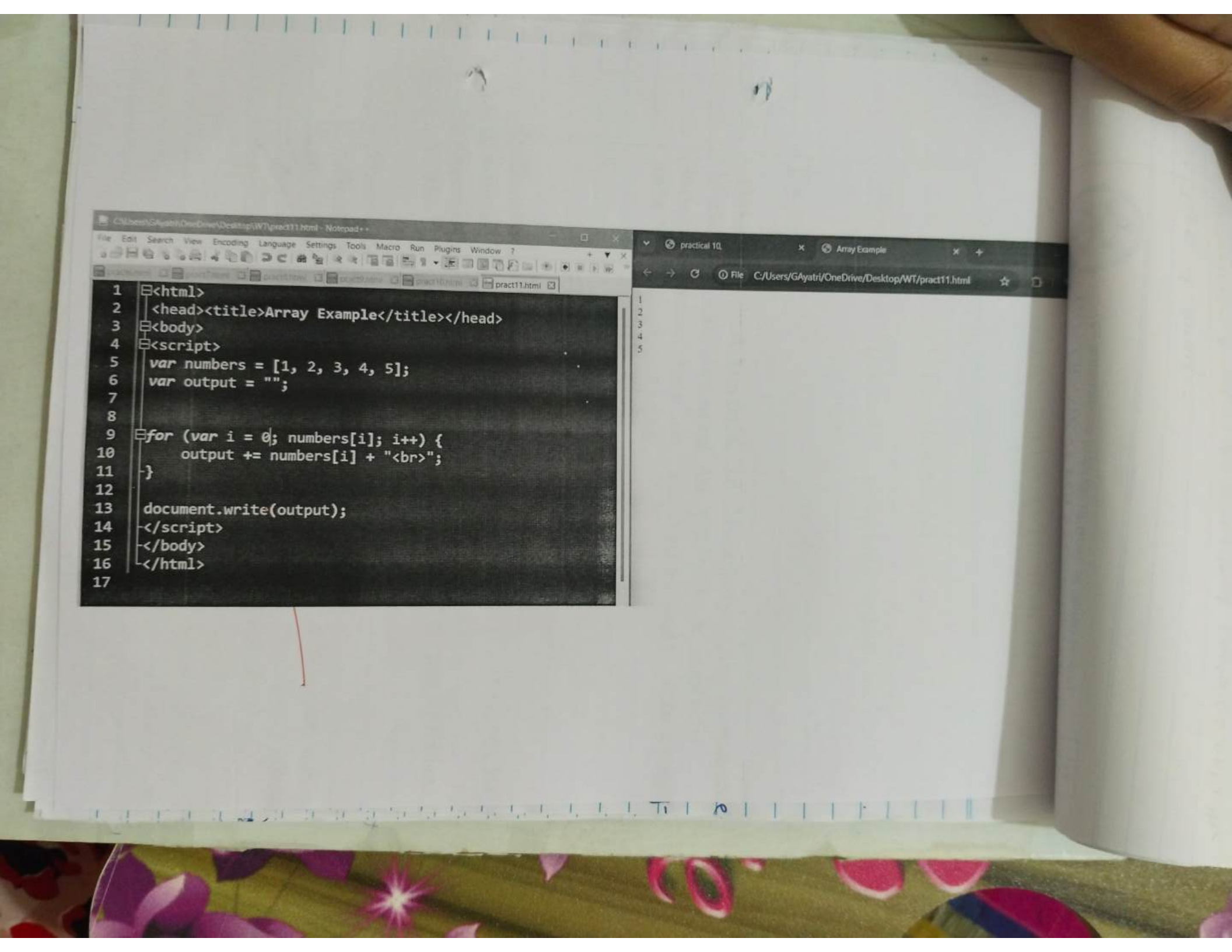
2. String Array:- It has all the memory block filled only with string values, which are characters or words only. We use ' ' for initialize it in square brackets.

```
let words= ['Greek','O','K'];
```

3. Array of Arrays:- 2D array contains the multiple arrays as its elements that are stored at different memory locations. It can be used to create nested array.

```
let arrayofarray= [[1,2,3],['English']];
```

Conclusion:- Hence, we learnt in this practical about how to write programs based on arrays.


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
pract11.html
practical 10
Array Example
File C:/Users/GAyatri/OneDrive/Desktop/WT/pract11.html
<html>
<head><title>Array Example</title></head>
<body>
<script>
var numbers = [1, 2, 3, 4, 5];
var output = "";
for (var i = 0; numbers[i]; i++) {
output += numbers[i] + "
";
}
document.write(output);
</script>
</body>
</html>

Practical No. 12

Title:- Write Javascript programs based on functions.

Theory:-

- Defination: A function in JavaScript is a reusable block of code designed to perform a specific task. functions help in orgdnizing code into manageable & logical sections.

Syntax:-

```
function func-name();
```

- Invocation:- • A function is executed or "called" by using its name followed by parantheses ().
Arguments can be passed within these parantheses if the function requires parameters.

- Parameters & Arguments:-
  - Parameters: they are placeholders in a function defination that specify what kind of values the func expects.
  - Arguments:- They are actual values passed to the function when it is called.

- Function Defination:-
  Before using a user-defined function in Javascript we have to create one. We can use the above syntax to create a function in Javascriptt.
  - Every function should begin with the keyword function followed by.
  - A user-defined function name that should be unique.

- Function Declaration:- It declares a function with a function keyword. The function declaration must have a function name.

Syntax:

```
function function_name(para1, para2){
    // set of statements
}
```

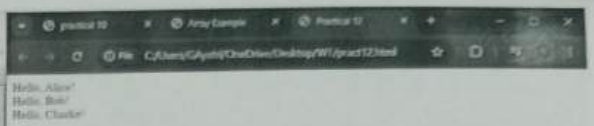
- Function Expression:- It is similar to a function declaration without the function name. Function expressions can be stored in a variable assignment.

Syntax:

```
let function_name = function(para1, para2){
    // set of statements
}
```

Conclusion:- Hence, we learnt how to declare function & how to use it in Javascript.

```html
<html>
<head><title>Practical 12</title></head>
<body>
<script>
function greet(name) {
    document.write("Hello, " + name + "!<br>");
}

greet("Alice");
greet("Bob");
greet("Charlie");
</script>
</body>
</html>
```




```html
<script>
function square(num) {
    return num * num;
}

var number1 = 4;
var number2 = 7;
var number3 = 10;

document.write("The square of " + number1 + " is " +
square(number1) + "<br>");
document.write("The square of " + number2 + " is " +
square(number2) + "<br>");
document.write("The square of " + number3 + " is " +
square(number3) + "<br>");
</script>
</body>
</html>
```

Practical No. 13

Title:- Write a Javascript Program on string

Theory:-

Strings in Javascript are sequences of characters used to represent text. They are enclosed in single quotes (' '), double quotes (" "), or backticks (` ` for template literals).
Strings can be created by assigning text to a variable or directly using string literals.

- **Basic Operations:-**
  - Concatenation: Combining two or more string into one using the + operator or concat() method.
  - Interpolation: Embedding expressions within strings using template literals (backticks).

- **Accessing Characters:**
  - Characters in a string can be accessed using bracket notation with an index (e.g. str[0] for the first character).

- Manipulation Methods:

- toUpperCase() and toLowerCase(): Convert the string to uppercase or lowercase.

- substring(), slice(), & substr(): Extract parts of the string.

- replace(): Replace occurrences of a substring with another string.

- trim(): remove whitespace from both ends of the string.

- Splitting & Joining:
- split(): Divide a string into an array of substrings based on a delimiter.
- join(): Combine elements of an array into a single string with a specified separator.

Conclusion:- Hence we learnt what is string in JS & how to handle & manipulate it.

```html
<html>
<head><title>practical 13</title></head>
<body>
<script>
var str = "Hello, World!";


var upperStr = str.toUpperCase();


var lowerStr = str.toLowerCase();

document.write("Original String: " + str + "<br>");
document.write("Uppercase: " + upperStr + "<br>");
document.write("Lowercase: " + lowerStr);
</script>
</body>
</html>

```

```
Original String: Hello, World!
Uppercase: HELLO, WORLD!
Lowercase: hello, world!
```

```html
<html>
<head><title>Practical 13</title></head>
<body>
<script>
var str = "Hello, World!";
var reversedStr = "";


for (var i = str.length - 1; i >= 0; i--) {
    reversedStr += str[i];
}

document.write("Original String: " + str + "<br>");
document.write("Reversed String: " + reversedStr);
</script>
</body>
</html>

```

```
Original String: Hello, World!
Reversed String: !dlroW ,olleH
```

Practical No. 14

Title:- Write Javascript programs on JS objects functions, classes.

Theory:-

In Javascript, an object is a collection of properties, where each property is a key-value pair. Keys are usually strings, while values can be any data type. Objects are used to store, organize & manipulate data.

- Concepts:

1) Properties: key-value pairs that define the characteristics of an object.
   Ex. {name: "Alice", age: 25}

2) Methods: functions that are stored as object properties.
   Ex. { greet: function()
   { consde.log("Hello");}}

3) Accessing Values: Use dot notation (object.property) or bracket notation (object["Property"]).

- In JS, classes are a blueprint for creating objects that share the same properties & methods.

Key Points:

- Class Declaration: Defined using the class keyword.

  Ex. class Person { }

- Constructor: A special method [constructor()] that initializes object properties.
- Methods: functions defined within the class.
- Inheritance: Allows child classes to use properties & methods from a parent class.
- Static Methods: Belong to the class, not to instances.

Conclusion: Olence, we learned use of object, function in object & classes.

```
getDetails() {
        return `${this.brand} ${this.model}`;
    }
}

// Function to display car information
function displayCarInfo() {
    // Getting input values
    let brand = document.getElementById(
        "carBrand").value;
    let model = document.getElementById(
        "carModel").value;

    // Creating an instance of the Car class
    let myCar = new Car(brand, model);

    // Displaying the car information in the
    "carInfo" paragraph
    document.getElementById("carInfo").
    innerHTML = `${myCar.getDetails()}`;
}
</script>
</body>
</html>
```

Practical No. 15

Title:- Write Javascript programs on JS HTMS DOM.

Theory:-

The HTML DOM (Document Object Model) is an inheritance that represents an HTML document as a structured tree of objects, allowing programs to access and manipulate the content, structure & styles of a web page dynamically.

1. Structure:
   The DOM represents the document as a tree, with the document object as the root, containing nodes for elements, text, and attributes.

2. Node Tree:
   - Element Nodes
   - Attribute Nodes
   - Accessing the DOM

3. Accessing Elements:
   Use method like document.getElementById(), document.getElementsByClassName() & document.querySelector() to select HTML elements.

4. Manipulating content: Change the content of elements using textContent or innerHTML.

5. Adding & Removing Elements: Create new elements

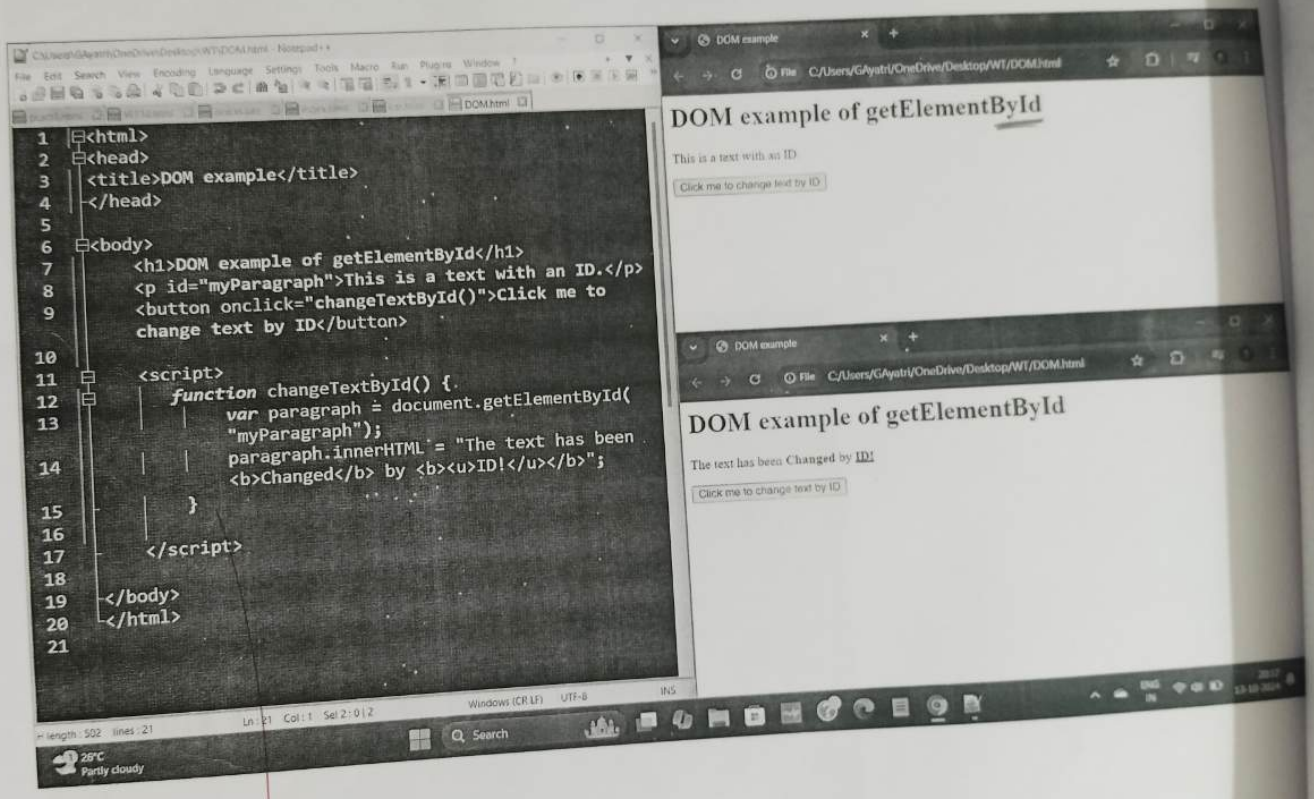
with document.createElement(), append then using appendChild(), & remove them with removeChild().

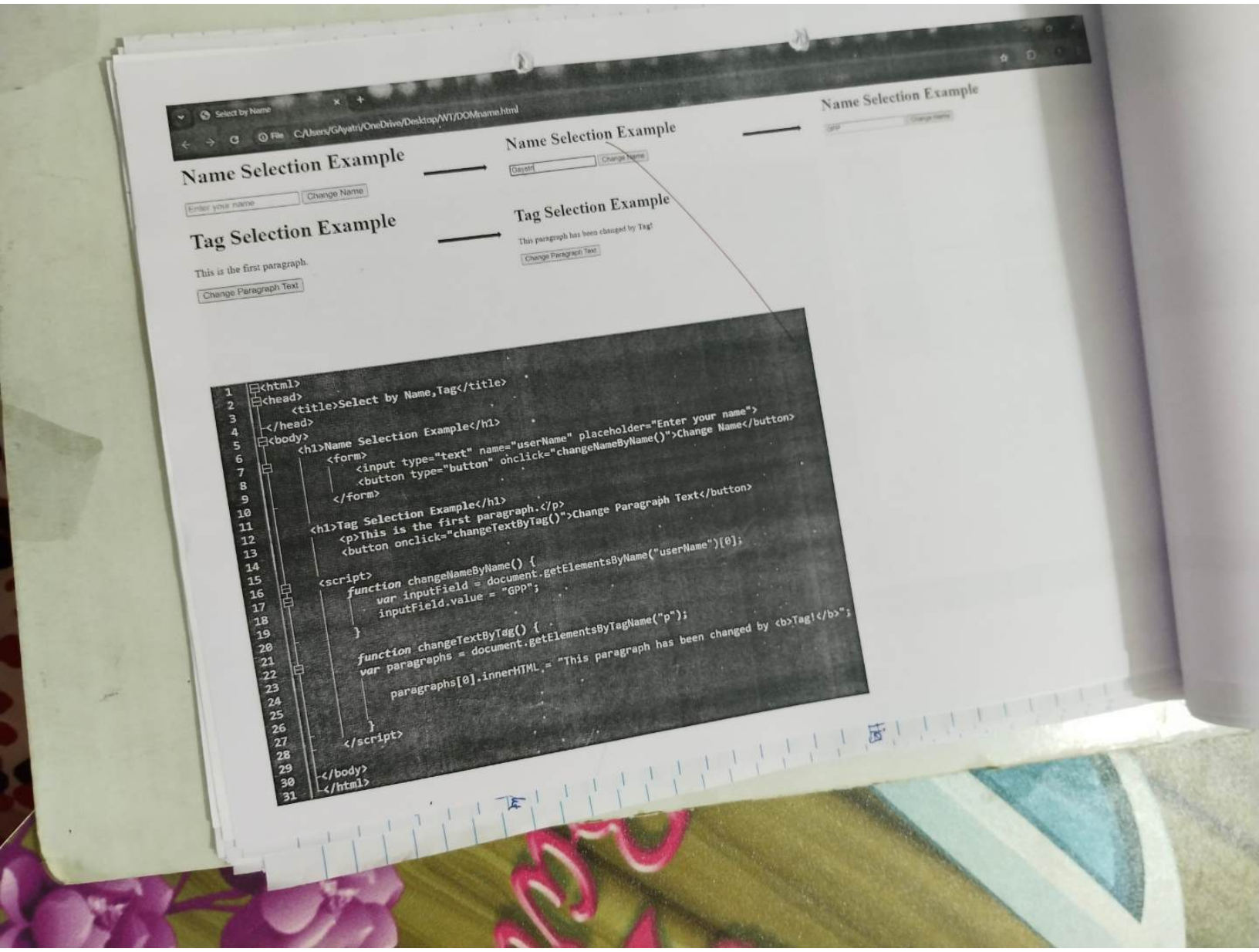
6. Changing Syles: Modify css styles directly through the style property.

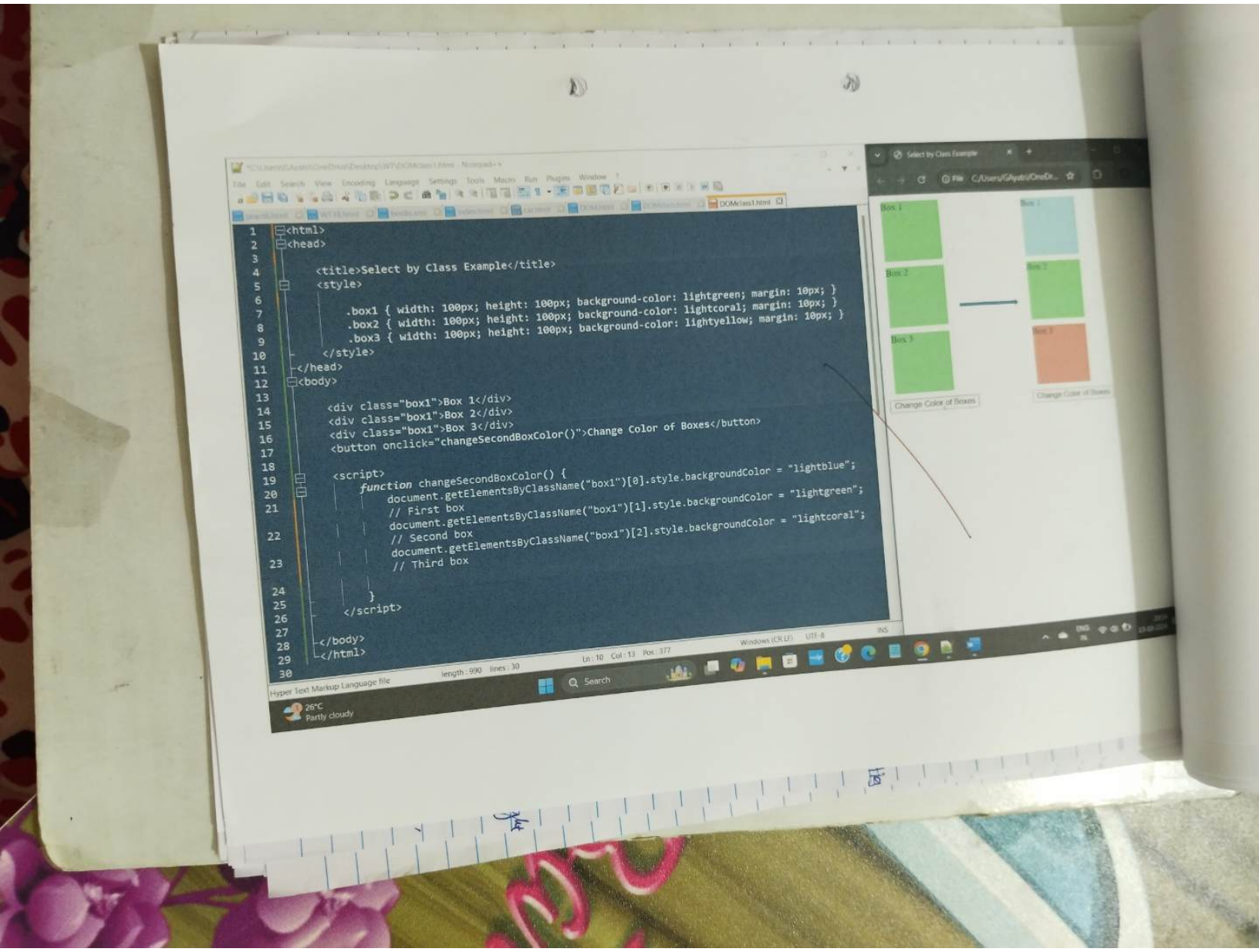
- DOM Manipulating:
- Selecting with id:- [#]
  document.getElementById ("id.name");
- Selecting with class: [.]
  document.getElementByClassName ("class_name");
- Selecting with tag:
  document.getElementsByTagName ("tag-name");
- Query selector: In accepts class, id & tags.
  1. document.querySelector("#id / class / tag");
- Get Attribute (attr) ⇒ To get attribute value
  set Attribute (attr, value) ⇒ To set attribute value.
- node.append (element) - adds at end of node [inside]
  node.prepend (el) - adds at start of node [inside]
  node.before (el) - adds before node [outside]
  node.remove () - removes node

- Data Manipulation Properties.
  - tagName: Returns tag for element nodes.
  - innerText: returns text content of element & all its children.
  - innerHTML: Returns plain text or html content in element.
  - textContent: returns textual content even for hidden element.

Conclusion: Hence we learnt Javascript programs on DS HTML DOM.


DOM.html
<html>
<head>
<title>DOM example</title>
</head>
<body>
<h1>DOM example of getElementById</h1>
This is a text with an ID.
<button onclick="changeTextById()">Click me to change text by ID</button>
<script>
function changeTextById() {
var paragraph = document.getElementById("myParagraph");
paragraph.innerHTML = "The text has been Changed by <u>ID!</u>";
}
</script>
</body>
</html>
DOM example
File C:/Users/GAyatri/OneDrive/Desktop/WT/DOM.html
DOM example of getElementById
This is a text with an ID
Click me to change text by ID
DOM example
File C:/Users/GAyatri/OneDrive/Desktop/WT/DOM.html
DOM example of getElementById
The text has been Changed by ID!
Click me to change text by ID
Search
28°C
Partly cloudy


Select by Name
File C:/Users/GAyatri/OneDrive/Desktop/W1/DOMname.html
Name Selection Example
Enter your name
Change Name
Tag Selection Example
This is the first paragraph.
Change Paragraph Text
Name Selection Example
GAyatri
Change Name
Tag Selection Example
This paragraph has been changed by Tag!
Change Paragraph Text
Name Selection Example
<html>
<head>
<title>Select by Name,Tag</title>
</head>
<body>
<h1>Name Selection Example</h1>
<form>
<input type="text" name="userName" placeholder="Enter your name">
<button type="button" onclick="changeNameByName()">Change Name</button>
</form>
<h1>Tag Selection Example</h1>
This is the first paragraph.
<button onclick="changeTextByTag()">Change Paragraph Text</button>
<script>
function changeNameByName() {
var inputField = document.getElementsByName("userName")[0];
inputField.value = "GPP";
}
function changeTextByTag() {
var paragraphs = document.getElementsByTagName("p");
paragraphs[0].innerHTML = "This paragraph has been changed by Tag!";
}
</script>
</body>
</html>


Select by Class Example
Box 1
Box 2
Box 3
Change Color of Boxes
Search
26°C
Partly cloudy

Practical No. 16.

Title: Write Javascript programs on JS Browser. BOM.

Theory:

Javascript Browser Object Model (BOM).

The Browser Object Model (BOM) allows Javascript to interact with the web browser providing functionality beyond just the HTML document.

1. Window Object: Represents the browser window Contains method like alert(), confirm(), prompt(), open() & close().

2. Navigator Object: Provides information about the browser, including properties like navigator.userAgent and navigator.language.

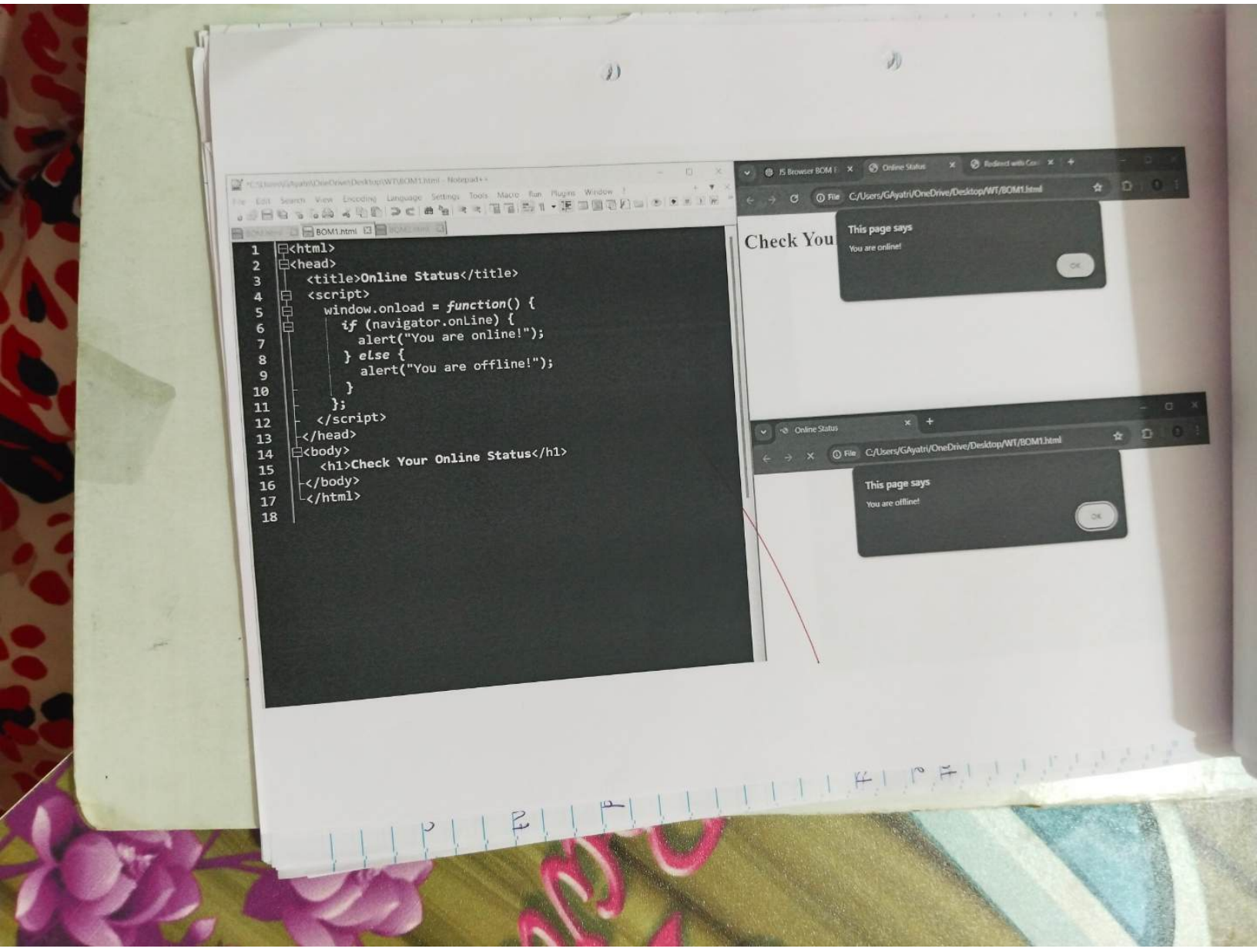
3. Screen Object: Contains information about the user's screen, such as screen width & screenheight

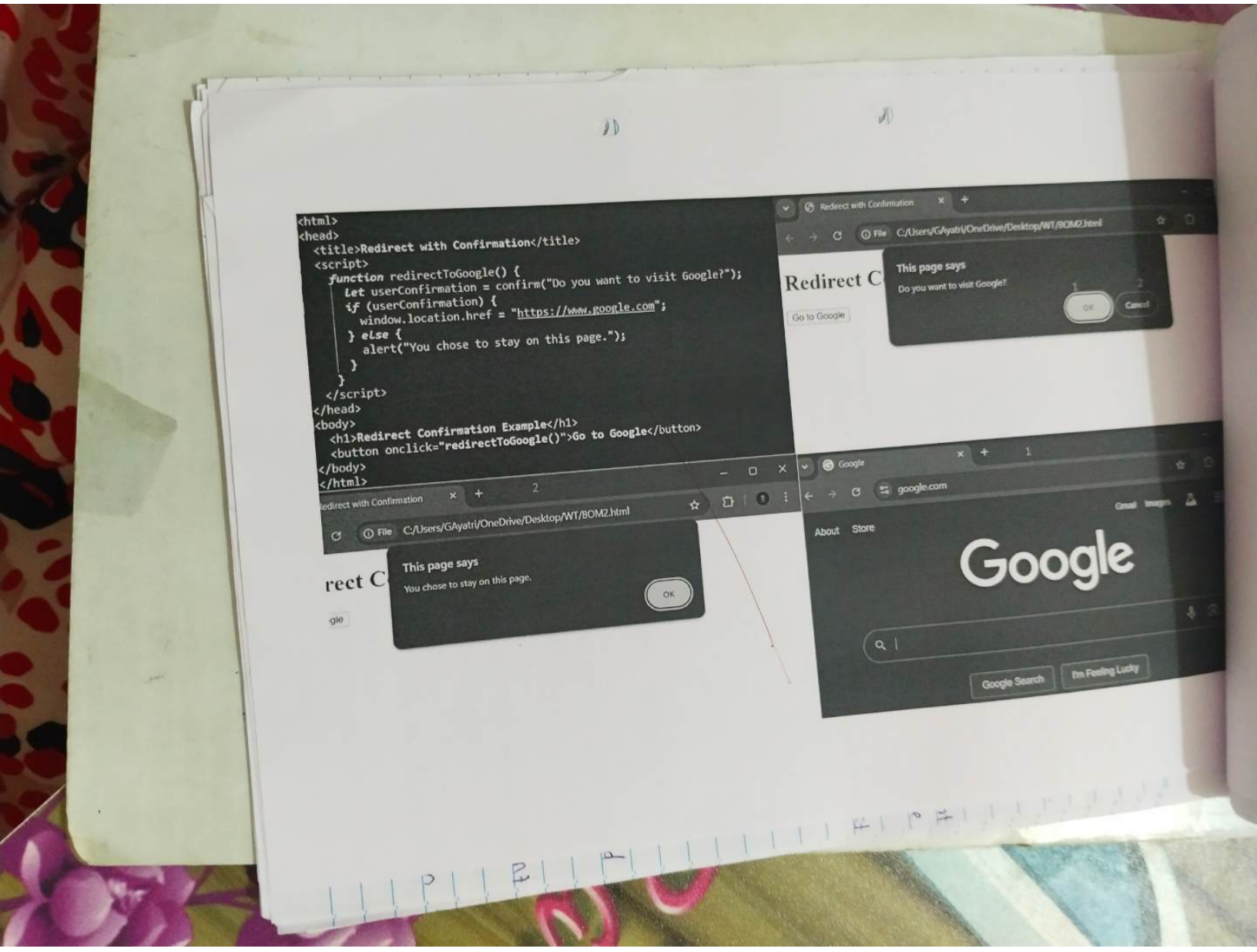
4. location Object: Represents the current URL methods include location.href, location.reload(), & location.assign(url).

5. History Object: Manages the browser session history with methods like history.back(), history.forward(), & history.go(n).

* Additionally, objects like Document, Console, Storage and Geolocation offer advanced capabilities for managing page content, debugging, local storage and even fetching the user's geographical position.

Conclusion: Hence, we performed & learned how to interact with browser using BOM in this practical.


BOM1.html
1 <html>
2 <head>
3 <title>Online Status</title>
4 <script>
5 window.onload = function() {
6 if (navigator.onLine) {
7 alert("You are online!");
8 } else {
9 alert("You are offline!");
10 }
11 };
12 </script>
13 </head>
14 <body>
15 <h1>Check Your Online Status</h1>
16 </body>
17 </html>
18
Online Status
File C:/Users/GAyatri/OneDrive/Desktop/WT/BOM1.html
This page says
You are online!
OK
Check You
Online Status
File C:/Users/GAyatri/OneDrive/Desktop/WT/BOM1.html
This page says
You are offline!
OK


<html>
<head>
<title>Redirect with Confirmation</title>
<script>
function redirectToGoogle() {
let userConfirmation = confirm("Do you want to visit Google?");
if (userConfirmation) {
window.location.href = "https://www.google.com";
} else {
alert("You chose to stay on this page.");
}
}
</script>
</head>
<body>
<h1>Redirect Confirmation Example</h1>
<button onclick="redirectToGoogle()">Go to Google</button>
</body>
</html>
Redirect with Confirmation
File C:/Users/GAyatri/OneDrive/Desktop/WT/BOM2.html
This page says
You chose to stay on this page.
OK
Google
google.com
About
Store
Google Search
I'm Feeling Lucky

Practical No. 03

Title:- Write Javascript program to illustrate the properties & methods of the document object.

Theory:-

The 'document' object in javascript is a key part of DOM:- Document Object Model. representing the HTML or XML document based in the browser. It provides various properties & methods to interact with & manipulate contents of the webpage.

- Document.title:- represent the title of a document <title>
  e.g. document.title = "New Page";

- Document.Body :- represents the <body> element of the document.
  e.g. document.body.style.background color = "lightblue";

- Document.head :- represents <hea> element of document
  e.g. console.log (document.head);

cument.form:- represents a collection of all the form in the document.
e.g. console.log (document.form[0]);

Practical No. 17

Title:- Write Javascript programs on JS web APIs

Theory:

Web APIs are interface provided by web browsers that enable developers to access manipulate browser features & resources beyond standard Javascript.

Categories:

1. DOM Manipulation APIs: Interact with HTML documents & elements.

2. Fetch API: Make network requests to retrive resources using promises.

3. Geolocation API: Access the user's geographical location.

4. Local Storage API: Store data persistently in the browser.

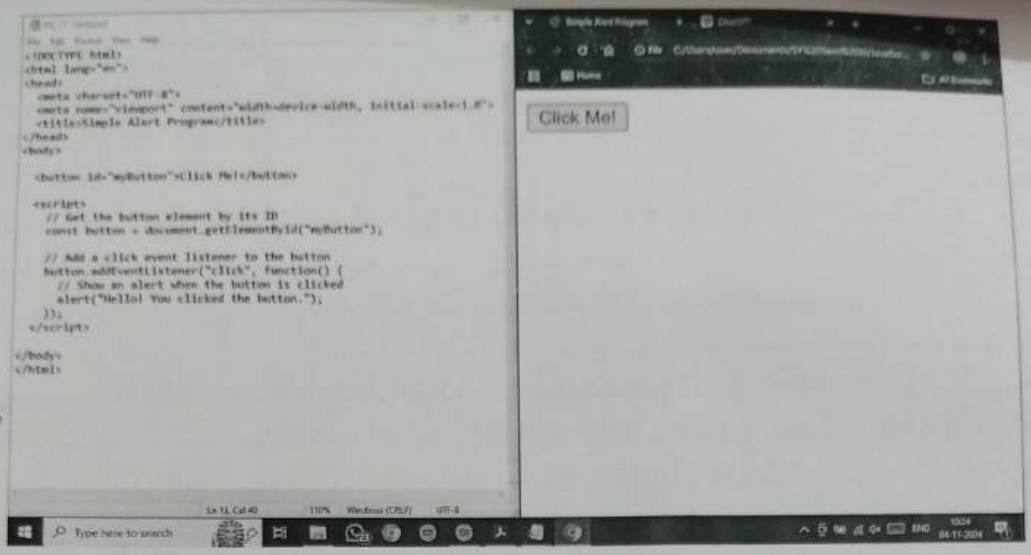
5. Session Storage API: Store data for the duration of a page session.

6. Web socket API : Enable real-time communication between the client & server.

7. Canvas API : Draw graphics & animations on the web

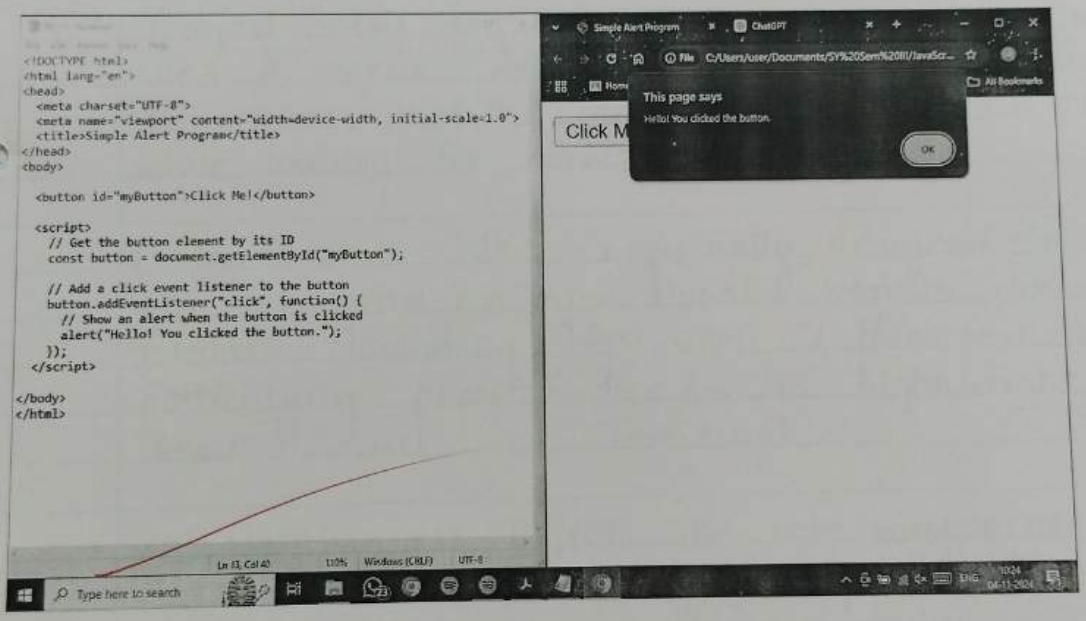
8. Audio & video APIs : Manipulate multimedia elements in the browser.

Conclusion: Hence we learned about the importance of API & its use.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Alert Program</title>
</head>
<body>

  <button id="myButton">Click Me!</button>

  <script>
    // Get the button element by its ID
    const button = document.getElementById("myButton");

    // Add a click event listener to the button
    button.addEventListener("click", function() {
      // Show an alert when the button is clicked
      alert("Hello! You clicked the button.");
    });
  </script>

</body>
</html>
```

Click Me!



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Alert Program</title>
</head>
<body>

  <button id="myButton">Click Me!</button>

  <script>
    // Get the button element by its ID
    const button = document.getElementById("myButton");

    // Add a click event listener to the button
    button.addEventListener("click", function() {
      // Show an alert when the button is clicked
      alert("Hello! You clicked the button.");
    });
  </script>

</body>
</html>
```

This page says

Hello! You clicked the button.

OK

GOVERNMENT POLYTE...
(An Autonomous Institute of Govt. of Maharashtra)

Practical No 18

Title:- Write Javascript programs on AJAX

Theory:-

AJAX (Asynchronous Javascript & XML):-
AJAX is a technique used web development to create asynchronous web applications. It allows web pages to be updated asynchronously by exchanging data with the server in the background, without requiring a full page refresh

- Asynchronous requests: AJAX enables web pages to send & recieve data from a server asynchronously This means that the web page can remain responsive while waiting for server responses.
- Data formats: While orginally designed to work with xml, AJAX can also handle various data formats, including JSON, HTML & plain text. JSON is particularly popular due to its lightweight structure & ease of use of Javascript.
- XMLHttpRequest Object: The core component of AJAX is the XMLHttpRequest object, which is used to make HTTP request to the server.

- The XMLHttpRequest Object
The XMLHttpRequest object is the faundation of AJAX. It allows you to send requests to a server & process the response.

1) Creating the Request: Initiatiate a new XMLHttpRequest object.

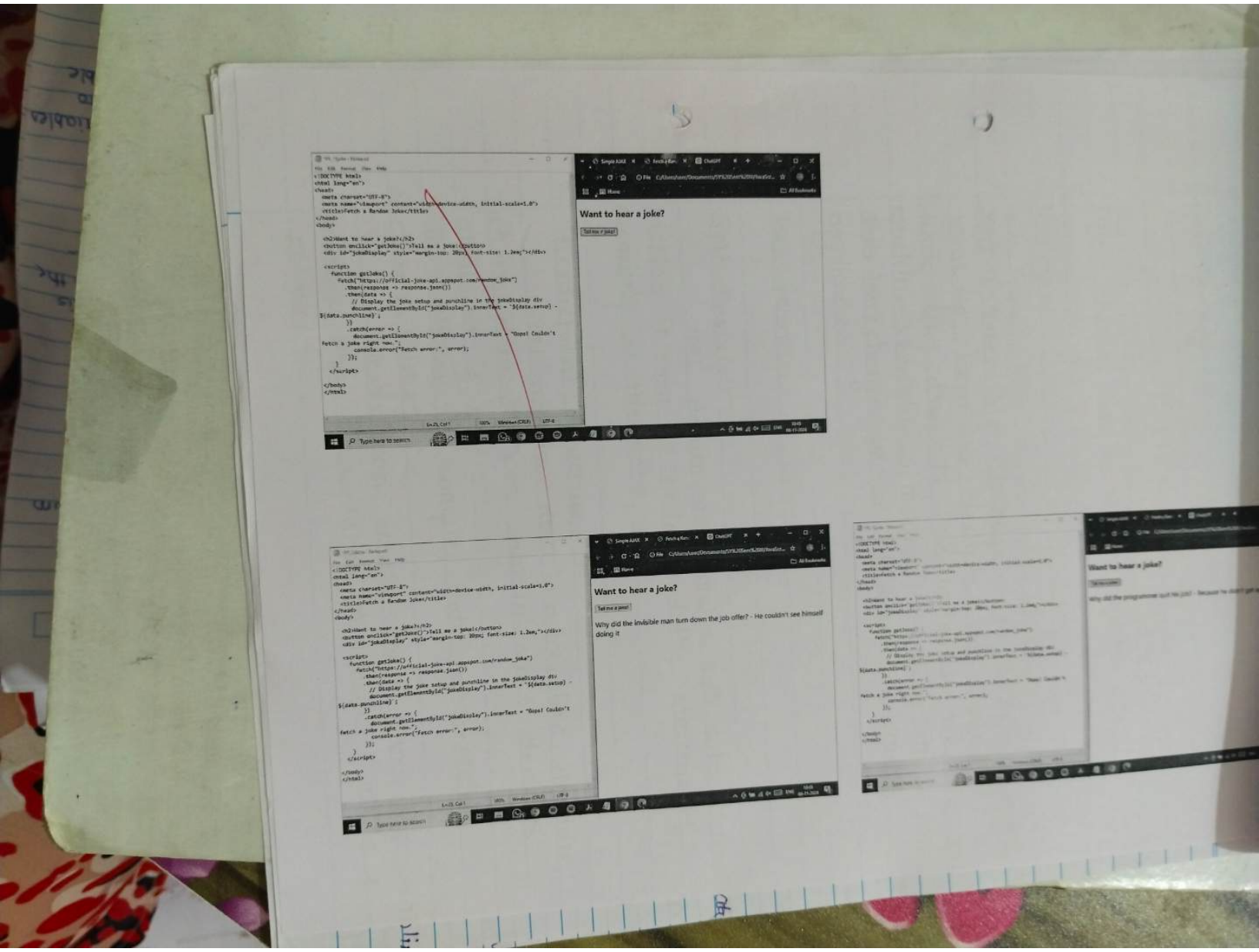
2) Configuring the request:
Use the open() method to specify the request type (GET or POST), URL & whether the request should be asynchraous Setting up callbacks:

3) Setting Up callbacks:
Define functions to handle the response, including success & error handling.

4) Sending the Request:
Call the send() method to dispatch the request to the server.

- Advantages of AJAX.
1) Improved user Experience
2) Reduced Server Load.
3) Enhanced performance.
4) Interactivity.

- Conclusion: Hence, we learnt & performed practical using Js program on AJAX.


Want to hear a joke?
Tell me a joke!
Why did the invisible man turn down the job offer? - He couldn't see himself doing it

4) Platform Independance:
XML is platform-agnostic, allowing data to be exchanged between different systems & programming language without compatibility issues.

5) Unicode Support:
XML supports Unicode, allowing it to handle a wide range of characters & symbols making it suitable for international applications.

- Basic Structure of XML
An XML document is composed of several essential components.

- Elements: Elements are the building blocks of XML, defined by opening & closing tags. Each element can contain text, attributes, or other nested elements.
- Attributes: Attributes provide additional information about elements & are defined within the opening tag.

- Advantages of XML:
1) XML facilitates seamless data exchange between heterogenous systems, making it ideal for integra tasks.

2) Human-readable Format: XML documents can be easily read & edited by humans, aiding in debug

Practical No. 20

- Title:- Write programs on XML AJAX.
- Theory:-

XML is a flexible markup language designed for data storage & transport. In context of AJAX, XML serves as a primary format for exchanging data btwn client & server, enabling dynamic & interactive web applications.

- Role of XML in AJAX

1) Data Format: XML provides a structured way to represent data. It allows developers to create complex hierarchical data formats that can be easily complex & understood by both humans & machines.
2) Asynchronous Communicate: AJAX enables web applications to send & recieve data. When a request is made, XML data can be sent to the server & a response can be recieved without reloading the entire web page, allowing for a seamless user experience.

- Key features of XML for AJAX:

1) Self-destructive Structure: XML uses tags to encapsulate data, making it easy to understand the data structure.
2) Hierarchical data Representation.

3) Cross Platform Compatibility
4) Support for MetaData.

Conclusion:-

Hence, we learnt & performed practical by using XML AJAX.

GOVERNMENT POLYTECHNIC, PUNE
(An Autonomous Institute of Govt. of Maharashtra)

41

Practical No. 21

- **Title:-** Write programs on XML DOM.
- **Theory:-**

XML DOM (Document Object Model) is a programming interface for XML documents. It represents the structure of an XML document as a tree of objects where each node corresponds to a part of the document (like elements, attributes & text).

- **Structure:**

1) Tree Representation: the XML document is parsed into a hierarchical tree structure, where:
   - Element Nodes: corresponds to XML elements.
   - Attribute Nodes: represent attributes of elements.
   - Text Nodes: contain the text within elements.
2) Text Node: contain the text within elements.

- **Methods & properties:**

1) Traversal: You can navigate the tree using methods to access parent, child & sibling nodes.
getElementById(), getElementByTagName(), querySelector(),

2) Manipulation: Nodes can be added, removed or modified.
- Methods like createElement(), appendChild(), removeChild(), & setAttribute().

3) Attributes: Access & modify attributes through getAttributes() & setAttribute() methods.

Conclusion:

DOM is a crucial for working with xml data programmatically, providing a structured way to access & manipulate XML documents in a tree-like format.

Practical No. 22

- Title:- Write Programs on XML Web Services

- Theory:-

XML Web Services: Facilitate data exchange between applications over the internet using XML, making them platform-independent.

- Key Components:

1) XML: Standard format for Data exchange.
2) SOAP (Simple Object Access Protocol):- Protocol for Sending XML messages.
3) WSDL (Web Services Description Language):- Describes the Web service, including operations & parameters
4) UDDI (Universal Descriptive, Discovery & Integral):- A registary for discovering available web Services

- Benifits:

XML Web services offer interoperability, scalability & ease of integration, making them widely used in enterprise applications & sectors requiring reliability such as healthcare & finance.

- Process Overview:

1) Service Request: A client sends a request to a provider.

2) Processing Service: The provider executes the request service.

3) Respons XML : The provider sends back results in XML.

- Security & Challenges:

Security is managed with WS-security & SSL/TLS, whole challenges include XML's data overhead & SOAP's complexity. Despite alternatives like REST, XML web services are valuable for high-security, structured applications.

Conclusion: ~~XML~~ web services are essential for building distributed, interopecable systems in modern enterprise environments.

Practical No. 23

Title:- A) Install & create Node Environement.
a) Extending Javascript b) v8
c) The process Object.

Theory:-

1) Installation:
- Download & install Node.js from node.js.org. which includes npm (Node Package Manager).
- Verify the installation with node -v & npm -v in the terminal to ensure Node & npm are properly installed.

2) Setting Up a Project:
- Create a Project folder & Navigate to it in the terminal.
- run npm in it to create a package.JSON file, which will store project details & manage dependancies configuations & scripts for the Node environment.

- Key Concepts in Node.Js.

a) Extending JavaScript:
Node Js enhances Javascripts capabilities, allowing it to run server-side by adding built-in modules and libraries (eg. fs for handling, http for servers)

Practical No. 24

Title:- 1] Writing programs on Streaming Data Across Node & clients.
a) Exporting Streams
b) creating an HTTP server
c) The request Object
d) Working with Headers.
e) Handling post Data.

- Theory:-

Streaming data involves the continous transmission of data between a server & clients, enabling real-time interactions.

a) Exporting Streams:-

Exporting streams allows clients to access data in real-time, useful for large datasets or continous data feeds. In python, this can be accomplished using flask's the Response object.

b) Creating an HTTP server.

An HTTP server listens for client requests & serves resources. frameworks like flask simplify the creation of HTTP servers.

c) The Request Object:

The request object contains details about the client's request, including method, URL, headers & form data. It helps the server understand the client's needs.

d. Working with Headers.

HTTP headers provide context about request & responses, such as content type & authentication Proper header management is essential for effecti communications.

e. Handling POST Data.

Post requests send data to a server for creat or updating resources, encapsulating the data in the request body.

Conclusion: Understanding streaming data is essential for building responsive web applications.

B] Write programs on file System Operations: Create, Read, Write files & directories.

File system operations involve managing files & directories within an operating system. These operation include creating directories, creating files, writing to files & reading from files.

a) Creating Directories:

Directories organize files in a hierarchical structure. They can be created using the os.makedirs() function in Python which checks for existance & creates necessary parent directories if they don't exist.

d) Creating files:-
Files are collections of data. To create a file, you can open it in append mode. If the file does not exist, it is created, if it does, no changes are made.

e) Writing Data to a file:-
Writing involves adding content to a file. Using write mode ('w') you can overwrite the existing content or create a new file.

e) Reading Data from:-
Reading retrives data stored in a file. Use read mode ('r') to access to files content.

Mode	Description
r	read mode (file must exist)
w	write mode (owerites file or creates new)
a	Append Mode (adds to end of file or creates new).
r+	Read/write mode same as r
w+	read/write mode same as w
a+	append/read mode.

- Error Handling:
file operations may rise errors, such as FileNotFoundError or PermissionError.

Conclusion:- Understanding the basic files systems operations is essential for tasks such as data storage, configuration management & logging in various applications

c) Write Programs on using Express.
a) Installing Express & making it Easier to Build web
b) Routes
c) Static file & Middleware
d) JSON.

Theory:-
a) Installing express:-
Express js is a minimal frameworks for building web applications in Node.js. It simplifies server setup & handling requests. You Install it via npm & create an Express app to manage routes.

b) Routes
Routing in Express determines how the application responds to specific URL endpoints & HTTP methods. You define routes using methods like app.get() & app.post() to manage different functionalities

c) Static file & Middleware.
Static file are served directly to clients (eg. HTML, css). Express provides middleware like express.static() to serve these files easily. Middleware functions can process requests &

GOVERNMENT POLYTECHNIC, PUNE
(An Autonomous Institute of Govt. of Maharashtra)

54

Practical No. 25

Title:- Write programs on DataBases.
a) Relational Databases & SQL
b) Node & MYSQL
c) NoSQL & Documents
d) MongoDB & Mongoose
e) CRUD operation using MongoDB.

Theory:-

a) Relational Database & SQL:-
Relational database use tables to store data with relationships defined through foreign keys. SQL is used to query & manipulate this data.

b) Node & MySQL:-
Node.js is a JavaScript runtime for building applications. It can interact with MySQL databases using packages like mysql or mysql12, allowing execution of SQL queries from Node.js.

c) NoSQL & Documents:-
NoSQL databases store unstructured data & offer flexible schemas. Document-oriented NOSQL database es save data in JSON-like documents, supporting nested structures.

GOVERNMENT POLYTECHNIC, PUNE
(An Autonomous Institute of Govt. of Maharashtra)

11

d) MongoDB & ~~MS~~ Mongoose :-

MongoDB uses its own language (MQL) for document-oriented operations.

1) db.collection.insertOne() :- adds a document
2) db.collection.find() :- retrives documents
3) db.collection.updateOne() :- updates document.

e) CRUD Operations In MongoDB :-

MongoDB's CRUD operations involve creating, reading, updating & deleting documents within collections.

Conclusion :- Hence, we learnt & performed practical using programs on Database.