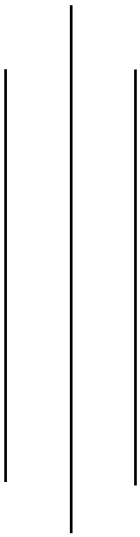
TRIBHUVAN UNIVERSITY
INSTITUE OF ENGINEERING
PULCHOWK CAMPUS

A LAB REPORT
on
Projects Using
HTML, CSS & JavaScript



SUBMITTED BY:

Name: Subesh Yadav
Roll: 080BCT084
Group: D
Lab No :2
Date: 1/18/2026

SUBMITTED TO:

Prakash Chandra Sir

Department of Electronics
and Computer Engineering

Assignment 1: Smart Calculator with Historv

Problem Statement Design a calculator that:
- *Performs basic arithmetic operations*
- *Stores last 5 calculations*
- *Displays calculation history dynamically*
- *Allows clearing history*

Additional Constraints
- *Handle invalid input (e.g., divide by zero)*
- *Use an array to store history*

1. ***Problem Understanding***
   The objective of this project is to design a smart calculator using JavaScript that can perform basic arithmetic operations such as addition, subtraction, multiplication, and division. In addition to calculation, the system must store and display the last five calculations dynamically. The calculator should also handle invalid inputs such as division by zero and allow users to clear both the current input and calculation history.
2. ***List of JavaScript Concepts Used***
   a. Variables
   b. Functions & Arrays
   c. Dom Manipulation
   d. Event Handling
   e. Built-in Functions like eval
   f. Error handling
3. ***Flow Diagram / Logic Steps***
   a. Users click on the button for calculation, input values get appended on display.
   b. When user clicks on '=', first of all, error handling is done to check any possible errors, after which it executes using eval command, if there is still errors, it shows on screen.
   c. The result is appended to history.
   d. Similarly users can clear the history, display screen as well with respective button.

4. ***Source Code***

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title>Calculator</title>
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
```

```
<div class="board" id='board'>
    <input type="text" name="" id="box" disabled>

    <div class="div" id="div">
    <button class="button" onclick="reset()">CE</button>
    <button class="button" onclick="removeOne()">X</button>
    <button class="button" onclick="Display('%')">%</button>
    <button class="button" onclick="Display('/')">/</button>
    <button class="button" onclick="Display('7')">7</button>
    <button class="button" onclick="Display('8')">8</button>
    <button class="button" onclick="Display('9')">9</button>
    <button class="button" onclick="Display('*')">*</button>
    <button class="button" onclick="Display('4')">4</button>
    <button class="button" onclick="Display('5')">5</button>
    <button class="button" onclick="Display('6')">6</button>
    <button class="button" onclick="Display('+')">+</button>
    <button class="button" onclick="Display('1')">1</button>
    <button class="button" onclick="Display('2')">2</button>
    <button class="button" onclick="Display('3')">3</button>
    <button class="button" onclick="Display('-')">-</button>
    <button class="button" onclick="Display('0')">0</button>
    <button class="button" onclick="Display('00')">00</button>
    <button class="button"
onclick="Display('.')"><strong>.</strong></button>

    <button class="button" onclick="cal()">=</button>
    </div>

</div>
<div class="history-board">
    <h3>History (Last 5)</h3>
    <ul id="historyList"></ul>
    <button onclick="clearHistory()">Clear History</button>
</div>


<script type="text/javascript" src="script.js">

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

Style.css

```css
* {
    padding: 0;
    margin: 0;
}

body {
    background-color: black;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
    gap:20px;
}

.board {
    margin-top: 50px;
    width: 214px;
    height: 351px;
    display: flex;
    flex-direction: column;
    background-color: rgb(242, 243, 248);
    border: 5px solid white;
    box-sizing: border-box;
    border-radius: 10px;
}

.div {
    display: grid;
    grid-template-columns: auto auto auto auto;
}

.button {
    font-size: 17px;
    margin: 3px;
    border-radius: 100%;
    box-shadow: 2px 1px 2px grey;
    width: 45px;
    height: 45px;
    border: none;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

```css
#box {
    border-radius: 10px;
    border: none;
    font-size: 17px;
    height: 100px;
    transition: 0.5s;
    position: relative;
}



.history-board {
    width: 214px;
    background-color: white;
    padding: 10px;
    border-radius: 8px;
    color: black;
}

.history-board ul {
    list-style: none;
    font-size: 14px;
    padding-left: 0;
}

.history-board li {
    padding: 4px 0;
    border-bottom: 1px solid #ccc;
}

.history-board button {
    margin-top: 8px;
    width: 100%;
    padding: 5px;
}
```

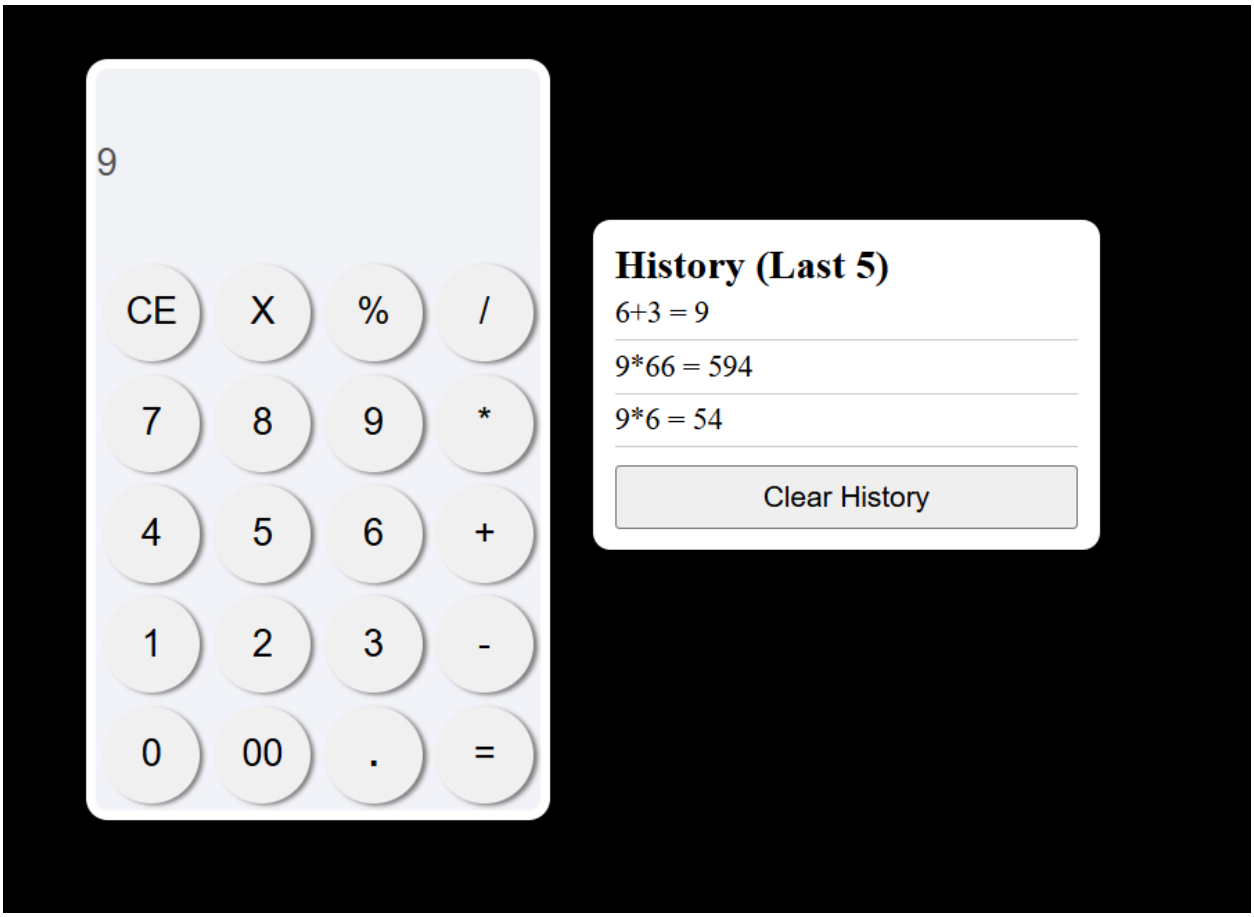
Script.js

```js
const box = document.getElementById('box');
const buttons = document.querySelectorAll(".button");
```

```
let calculation = '';
let history = [];
let lastCalculation=false;

function Display(value){
    if(lastCalculation){
        calculation='';
        lastCalculation=false;
    }
    calculation += value;
    box.value = calculation;
}

function cal(){
    try {
        if (calculation === "") return;

        if (calculation.includes("/0")) {
            throw new Error("Division by zero");
        }

        if (calculation.includes("**") || calculation.includes("//"))
{
            throw new Error("Invalid operator usage");
        }


        let result = eval(calculation);

        if (!isFinite(result)) {
            throw new Error("Invalid calculation");
        }

        addToHistory(calculation + " = " + result);

        box.value = result;
        calculation = result.toString();
        lastCalculation=true;
    } catch (err) {
        alert("Invalid input: " + err.message);
        reset();
    }
}
```

```
function removeOne(){
    calculation = calculation.slice(0, -1);
    box.value = calculation;
}

function reset(){
    calculation = '';
    box.value = '';
}

function addToHistory(entry){
    history.unshift(entry);

    if (history.length > 5) {
        history.pop();
    }

    renderHistory();
}

function renderHistory(){
    const historyList = document.getElementById("historyList");
    historyList.innerHTML = "";

    history.forEach(item => {
        const li = document.createElement("li");
        li.innerText = item;
        historyList.appendChild(li);
    });
}

function clearHistory(){
    history = [];
    renderHistory();
}
```

5. ***Output***


9
CE
X
%
/
7
8
9
*
4
5
6
+
1
2
3
-
0
00
.
=
History (Last 5)
6+3 = 9
9*66 = 594
9*6 = 54
Clear History

Assignment 2: Student Marks & Grade Analyzer

Problem Statement Create a system where:

- *Student details are stored as objects*
- *Marks for multiple subjects are entered*
- *Total, percentage, and grade are calculated*
- *Result is displayed in a table*
- *Highlight pass/fail rows using CSS via JS*

1. ***Problem Understanding***
   The objective of this project is to develop a student result management system using JavaScript. Student details are stored as objects, and marks for multiple subjects are entered for each student. The system calculates the total marks, percentage, and grade for every student and displays the result in a tabular format. Students are categorized as pass or fail, and the corresponding rows are highlighted using CSS applied through JavaScript.
2. ***List of JavaScript Concepts Used***
   a. Objects & Array
   b. Array Methods like forEach, reduce
   c. Functions
   d. DOM Manipulations
   e. Conditional Statements

3. ***Flow Diagram / Logic Steps***
   a. Store student details and marks
   b. Calculate total and percentage
   c. Assign grade and pass/fail status
   d. Display results in a table
   e. Highlight pass/fail rows

4. ***Source Code***

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Marks & Grade Analyzer</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>

<h2>Student Marks & Grade Analyzer</h2>

<table style="border: 1px solid black;" id="result-table">
    <tr>
        <th>Name</th>
```

```
        <th>Marks</th>
        <th>Total</th>
        <th>Percentage</th>
        <th>Grade</th>
        <th>Result</th>
    </tr>
</table>

<script src="script.js"></script>
</body>
</html>
```

Style.css

```
table {
    border-collapse: collapse;
    width: 80%;
}

th, td {
    padding: 8px;
    text-align: center;
}

.pass {
    background-color: #c8f7c5;
}

.fail {
    background-color: #f7c5c5;
}
```

Script.js

```


const table = document.getElementById("result-table");

const students = [
    {
        name: "Subesh",
        marks: [75, 80, 70]
```

```
    },
    {
        name: "Saroj",
        marks: [45, 35, 30]
    },
    {
        name: "Sangam",
        marks: [90, 67, 92]
    }
];

students.forEach(student => {
    let total = student.marks.reduce((a, b) => a + b, 0);
    let percentage = total / student.marks.length;
    let grade = getGrade(percentage);
    let result = percentage >= 40 ? "Pass" : "Fail";

    let row = table.insertRow();
    row.innerHTML = `
        <td>${student.name}</td>
        <td>${student.marks.join(", ")}</td>
        <td>${total}</td>
        <td>${percentage.toFixed(2)}%</td>
        <td>${grade}</td>
        <td>${result}</td>
    `;

    row.className = result === "Pass" ? "pass" : "fail";
});

function getGrade(percent) {
    if (percent >= 80) return "A";
    if (percent >= 60) return "B";
    if (percent >= 40) return "C";
    return "F";
}
```

5. Output

Student Marks & Grade Analyzer

Name	Marks	Total	Percentage	Grade	Result
Subesh	75, 80, 70	225	75.00%	B	Pass
Saroj	45, 35, 30	110	36.67%	F	Fail
Sangam	90, 67, 92	249	83.00%	A	Pass

Assignment 3: Dynamic To-Do List with Status Filter

Problem Statement Build a To-Do app that:

- *Adds, deletes, and marks tasks as completed*
- *Filters tasks (All / Completed / Pending)*
- *Displays count of completed tasks*

1. ***Problem Understanding***
   The objective of this project is to create a dynamic To-Do List application using JavaScript. The app allows users to add new tasks, delete existing tasks, and mark tasks as completed. It also provides a filtering feature to view all, completed, or pending tasks, and displays the count of completed tasks dynamically.
2. ***List of JavaScript Concepts Used***
   a. Objects & Array & Array Methods like forEach, reduce & Filter
   b. Functions
   c. DOM Manipulations & Event Handling
   d. Conditional Statements
   e. Template Literals
3. ***Flow Diagram / Logic Steps***
   a. Add task to the list
   b. Mark task as completed or pending
   c. Delete a task
   d. Filter tasks (All / Completed / Pending)
   e. Update and display count of completed tasks

4. ***Source Code***

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Dynamic To-Do List</title>
<link rel="stylesheet" href="style.css">
</head>
<body>

<h2>Dynamic To-Do List</h2>

<div class="todo-container">
    <input type="text" id="taskInput" placeholder="Enter a task">
```

```
    <button id="addTaskBtn">Add Task</button>
</div>

<div class="filters">
    <button data-filter="all">All</button>
    <button data-filter="completed">Completed</button>
    <button data-filter="pending">Pending</button>
</div>

<p>Completed Tasks: <span id="completedCount">0</span></p>

<ul id="taskList"></ul>

<script src="script.js"></script>
</body>
</html>
```

Style.css

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f4f7f9;
    padding: 40px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

h2 {
    color: #333;
    margin-bottom: 20px;
}

.todo-container {
    display: flex;
    justify-content: center;
    margin-bottom: 20px;
}

#taskInput {
    width: 250px;
    padding: 10px;
    border: 2px solid #ddd;
```

```css
    border-radius: 8px 0 0 8px;
    outline: none;
    font-size: 16px;
    transition: border-color 0.3s;
}

#taskInput:focus {
    border-color: #007BFF;
}

#addTaskBtn {
    padding: 10px 20px;
    background-color: #007BFF;
    border: none;
    color: white;
    font-weight: bold;
    cursor: pointer;
    border-radius: 0 8px 8px 0;
    transition: background-color 0.3s;
}

#addTaskBtn:hover {
    background-color: #0056b3;
}

.filters {
    margin-bottom: 15px;
}

.filters button {
    padding: 6px 15px;
    margin: 0 5px;
    border: none;
    border-radius: 5px;
    background-color: #ddd;
    cursor: pointer;
    transition: background-color 0.3s;
}

.filters button:hover {
    background-color: #bbb;
}
```

```css
ul {
    list-style-type: none;
    padding-left: 0;
    width: 350px;
}

li {
    background-color: white;
    padding: 12px 15px;
    margin-bottom: 8px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    transition: transform 0.2s;
}

li:hover {
    transform: translateY(-2px);
}

.completed span {
    text-decoration: line-through;
    color: #28a745;
    cursor: pointer;
}

.pending span {
    color: #dc3545;
    cursor: pointer;
}

li button {
    background-color: #dc3545;
    border: none;
    color: white;
    padding: 5px 10px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s;
}
```

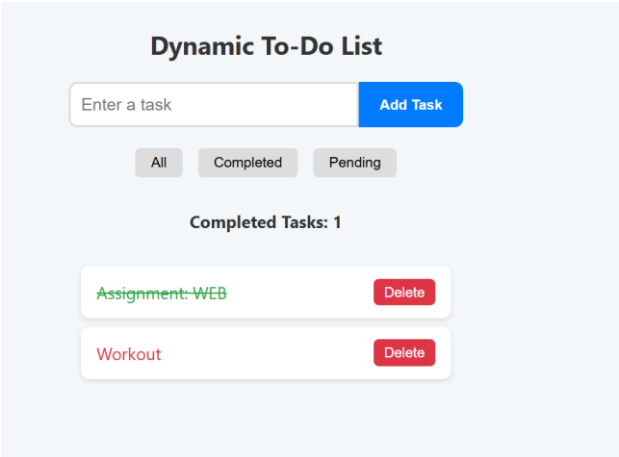
```css
li button:hover {
    background-color: #a71d2a;
}

p {
    font-weight: bold;
    color: #333;
}
```

Script.js

```js
const taskInput = document.getElementById("taskInput");
const addTaskBtn = document.getElementById("addTaskBtn");
const taskList = document.getElementById("taskList");
const completedCount = document.getElementById("completedCount");
const filterButtons = document.querySelectorAll(".filters button");

let tasks = [];

addTaskBtn.addEventListener("click", () => {
    const text = taskInput.value.trim();
    if (text === "") return;
    tasks.push({ id: Date.now(), text: text, completed: false });
    taskInput.value = "";
    renderTasks();
});

function toggleTask(id) {
    tasks = tasks.map(task => {
        if (task.id === id) task.completed = !task.completed;
        return task;
    });
    renderTasks();
}

function deleteTask(id) {
    tasks = tasks.filter(task => task.id !== id);
    renderTasks();
}

function renderTasks(filter = "all") {
    taskList.innerHTML = "";
```

```
    let filteredTasks = tasks;
    if (filter === "completed") filteredTasks = tasks.filter(t =>
t.completed);
    else if (filter === "pending") filteredTasks = tasks.filter(t =>
!t.completed);

    filteredTasks.forEach(task => {
        const li = document.createElement("li");
        li.className = task.completed ? "completed" : "pending";
        const span = document.createElement("span");
        span.innerText = task.text;
        span.onclick = () => toggleTask(task.id);
        const delBtn = document.createElement("button");
        delBtn.innerText = "Delete";
        delBtn.onclick = () => deleteTask(task.id);
        li.appendChild(span);
        li.appendChild(delBtn);
        taskList.appendChild(li);
    });
    completedCount.innerText = tasks.filter(t => t.completed).length;
}

filterButtons.forEach(btn => {
    btn.addEventListener("click", () =>
renderTasks(btn.dataset.filter));
});
```

6. **Output**



Assignment 4: Number Guessing Game with Score Tracking

Problem Statement Develop a game where:

- *System generates a random number (1–100)*
- *User guesses the number*
- *Feedback is shown (High / Low / Correct)*
- *Score decreases on wrong attempts*

1. ***Problem Understanding***
   The Number Guessing Game asks the computer to generate a random number between 1 and 100. The player tries to guess this number, and after each guess, the game provides feedback—telling the player if their guess is too high, too low, or correct. Each wrong guess reduces the player's score, and the player can reset the game to start over. The goal is to guess the correct number with the highest score possible.

2. ***List of JavaScript Concepts Used***
   a. Objects & Array
   b. Functions
   c. DOM Manipulations & Event Handling
   d. Conditional Statements
   e. Template Literals

3. ***Flow Diagram / Logic Steps***
   a. Generate a random number between 1–100 and set the initial score.
   b. User enters a guess and clicks submit.
   c. Check if the guess is valid; show a warning if not.
   d. Compare the guess to the random number: show "Too High", "Too Low", or "Correct".
   e. Reduce the score for wrong guesses and update the display.
   f. User can reset the game to start over with a new number and full score.

4. ***Source Code***

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Number Guessing Game with Score Tracking</title>
    <link rel="stylesheet" href="style.css">
</head>
```

```
<body>

    <div id="game">
        <h1 id="title">Guess the Number</h1>

        <div id="inputSection">
            <input type="number" id="userGuess" placeholder="Enter
your guess (1-100)" min="1" max="100">
            <button id="submitGuess">Submit Guess</button>
        </div>

        <div id="message">
            <h1 id="feedback-title">Feedback:</h1>
            <p id="feedback"></p>
        </div>
        <div>
            <button id="reset">Reset</button>
        </div>
    </div>
    <div id="score">
        <h2>Score: <span id="scoreValue">100</span></h2>
    </div>
    <script src="script.js"></script>
</body>

</html>
```

Style.css

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f4f7f9;
    padding: 40px;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
    position: relative;
}

#score {
    position: absolute;
    top: 10px;
    right: 10px;
```

```
    font-size: 1.2em;
    font-weight: bold;
    color: #333;

}

#title {

    color: #333;
    margin-bottom: 0;
    position: relative;
}

#title::after {
    content: 'Win Against The Computer';
    position: absolute;
    display: block;
    left: 100px;
    font-size: 0.4em;
    font-style: italic;
    color: #666;
    margin-top: -7px;
}


#game {
    display: flex;
    flex-direction: column;
    align-items: center;
    gap: 20px;
}

#inputSection {
    margin-top: 50px;
}

#userGuess {
    width: 200px;
    padding: 7px;
    border-radius: 5px;
}

#submitGuess {
```

```
    padding: 7px 15px;
    border-radius: 5px;
    border: none;
    background-color: #007BFF;
    color: white;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s;
}

#submitGuess:hover {
    background-color: #0056b3;
}


#message {
    color: #333;
    font-family: Arial, sans-serif;
    font-style: italic;
    padding: 5px;
    border-radius: 8px;
    background-color: #f0f0f0;
    max-width: 400px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.15);
    display: none;
}

#feedback-title {
    font-size: 0.7em;
    color: #2f3640;
}

#feedback {
    background-color: #a4b0b8;
    color: #fff;
    border-radius: 5px;
    margin: 0;
    padding: 2px;
}


#reset {
```

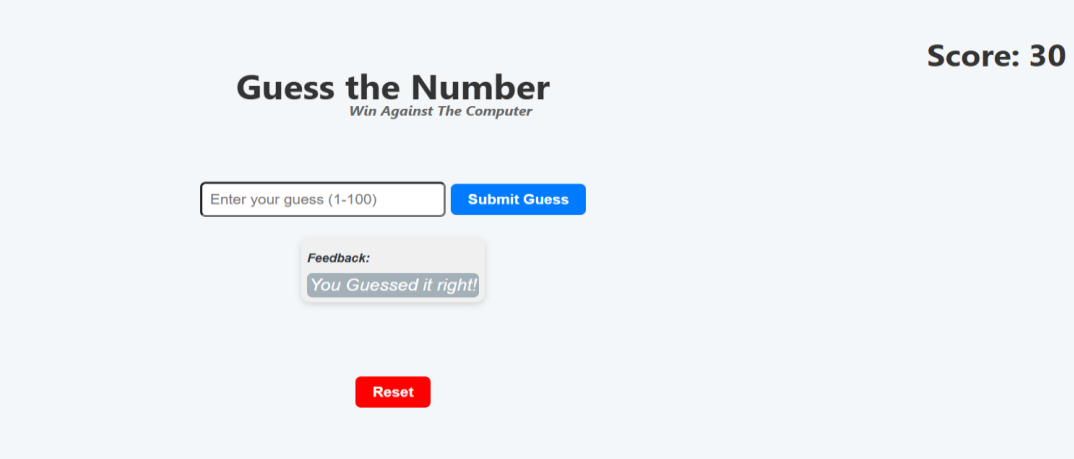
```
    padding: 7px 15px;
    border-radius: 5px;
    border: none;
    margin-top: 50px;
    background-color: red;
    color: white;
    font-weight: bold;
    cursor: pointer;
    transition: background-color 0.3s;
}
```

Script.js

```
const userGuess = document.getElementById('userGuess');
const submitGuess = document.getElementById('submitGuess');
const scoreValue = document.getElementById('scoreValue');
const feedback = document.getElementById('feedback');
const message = document.getElementById('message');
const reset = document.getElementById('reset');

let score = 100;
let computerRandom = generateComputerGuess();
handleScore(score);

submitGuess.addEventListener('click', handleGuess);
reset.addEventListener('click', handleReset);

function generateComputerGuess() {
    return Math.floor(Math.random() * 100) + 1;
}

function handleScore(score) {
    scoreValue.innerText = score;
}

function showMessage(text) {
    message.style.display = 'block';
    feedback.innerText = text;
}

function handleGuess() {
    const guess = Number(userGuess.value);
```

```
    if (guess < 1 || guess > 100) {
        showMessage("Invalid Guess: Guess Between 1 to 100");
        return;
    }

    if (guess === computerRandom) {
        showMessage("You Guessed it right!");
    } else {
        score = Math.max(score - 10, 0);
        handleScore(score);

        if (guess > computerRandom) {
            showMessage("Guess Smaller Number than " + guess);
        } else {
            showMessage("Guess Larger Number than " + guess);
        }
    }

    userGuess.value = '';
}

function handleReset() {
    score = 100;
    computerRandom = generateComputerGuess();
    handleScore(score);
    message.style.display = 'none';
    userGuess.value = '';
}
```

5. **Output**



Assignment 5: Interactive Quiz Application

Problem Statement: *Create a quiz app where:*
- *Questions are stored as objects*
- *One question is shown at a time*
- *User selects an answer*
- *Score is calculated and shown at the end*

1. ***Problem Statement***
   The Number Guessing Game asks the computer to generate a random number between 1 and 100. The player tries to guess this number, and after each guess, the game provides feedback—telling the player if their guess is too high, too low, or correct. Each wrong guess reduces the player's score, and the player can reset the game to start over. The goal is to guess the correct number with the highest score possible.

2. ***List of JavaScript Concepts Used***
   a. Objects & Array & Array Methods
   b. Functions
   c. DOM Manipulations & Event Handling
   d. Conditional Statements
   e. Template Literals
3. ***Flow Diagram / Logic Steps***
   a. Start quiz → Hide start button → Initialize question counter and user answers.
   b. Display the current question and its options.
   c. User clicks an option button → mark it as selected → store answer.
   d. User clicks Next: If no answer selected → show alert.
   e. If answer selected → move to next question.
   f. Repeat steps 2–4 until the last question.
   g. On last question → calculate score by comparing user answers with correct answers.
   h. Display the final score

4. ***Source Code***

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Interactive Quiz Application</title>
    <link rel="stylesheet" href="style.css">

</head>
<body>

    <div id="quiz">
```

```
        <h1>Interactive Quiz Application</h1>
        <button id="quiz-start" onclick="playQuiz()">Start Quiz</button>
        <div id="quiz-box">
         </div>
         <div id="result">
         </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

Style.css

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #f0f4f8, #d9e2ec);
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    height: 100vh;
}

#quiz {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    margin-top: 40px;
    background-color: #ffffff;
    padding: 30px 40px;
    border-radius: 15px;
    box-shadow: 0 8px 25px rgba(0,0,0,0.1);
    max-width: 500px;
    height: 80%;
}

#quiz h1 {
    margin-bottom: 20px;
    color: #1f2937;
    text-align: center;
}

#quiz-start {
    margin-top: 10px;
    background-color: #4fd1c5;
    color: #fff;
    border: none;
    border-radius: 8px;
```

```
    padding: 12px 20px;
    font-size: 1em;
    cursor: pointer;
    transition: 0.3s;
}

#quiz-start:hover {
    background-color: #38b2ac;
}

#quiz-box {
    margin-top: 20px;
    display: flex;
    flex-direction: column;
    width: 100%;
}

#quiz-box h3#title {
    font-size: 1.2em;
    color: #0f172a;
    margin-bottom: 20px;
}

.ansChoices {
    counter-reset: option;
    display: flex;
    flex-direction: column;
    gap: 12px;
}

.option-btn {
    counter-increment: option;
    padding: 12px 16px;
    text-align: left;
    font-size: 1em;
    cursor: pointer;
    border: 2px solid #cbd5e1;
    border-radius: 8px;
    background-color: #f8fafc;
    transition: 0.3s;
    position: relative;
}

.option-btn::before {
    content: counter(option, lower-alpha) ". ";
    font-weight: bold;
    margin-right: 6px;
}

.option-btn:hover {
```

```
    background-color: #e2e8f0;
}

.option-btn.selected {
    background-color: #38b2ac;
    color: white;
    border-color: #319795;
}

.next-btn {
    background-color: #4fd1c5;
    border: none;
    margin-top: 20px;
    align-self: flex-end;
    padding: 10px 18px;
    border-radius: 8px;
    font-size: 1em;
    letter-spacing: 1px;
    cursor: pointer;
    transition: 0.3s;
}

#next-btn:hover {
    background-color: #38b2ac;
}

#result h3 {
    margin-top: 20px;
    color: #1f2937;
    text-align: center;
}

#result button{
    margin: 20px;
    background-color: #4fd1c5;
    color: #fff;
    border: none;
    border-radius: 8px;
    padding: 12px 20px;
    font-size: 1em;
    cursor: pointer;
    transition: 0.3s;
}
```

Script.js

```
const quizBox=document.getElementById("quiz-box");
const quizStart=document.getElementById("quiz-start");
const result=document.getElementById('result')
```

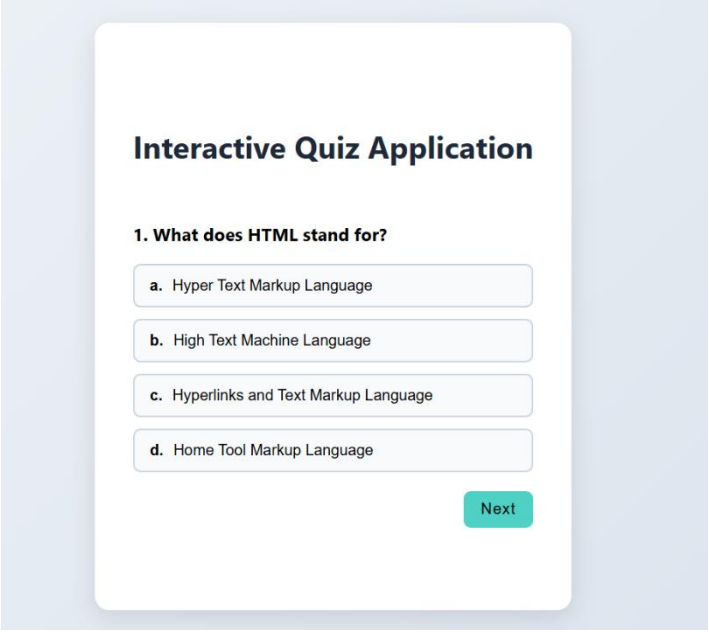
```
const quiz_questions = [
    {
        question: "What does HTML stand for?",
        options: [
            "Hyper Text Markup Language",
            "High Text Machine Language",
            "Hyperlinks and Text Markup Language",
            "Home Tool Markup Language"
        ],
        answer: "Hyper Text Markup Language"
    },
    {
        question: "Which HTML tag is used to create a hyperlink?",
        options: [
            "<link>",
            "<a>",
            "<href>",
            "<url>"
        ],
        answer: "<a>"
    },
    {
        question: "Which CSS property is used to change text color?",
        options: [
            "font-color",
            "text-color",
            "color",
            "foreground"
        ],
        answer: "color"
    },
    {
        question: "Which CSS unit is relative to the parent element?",
        options: [
            "px",
            "em",
            "cm",
            "mm"
        ],
        answer: "em"
    },
    {
        question: "Which keyword is used to declare a constant in JavaScript?",
        options: [
            "var",
            "let",
            "const",
            "static"
        ],
        answer: "const"
```

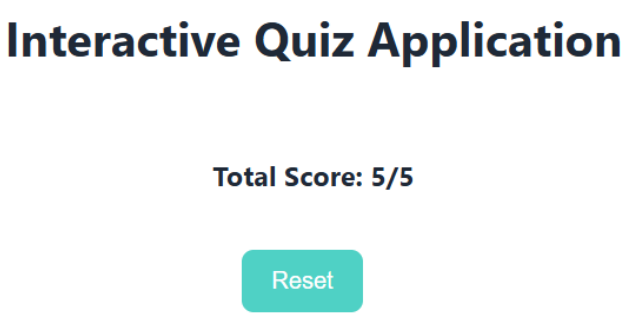
```
    }
];

let question_counter=0;
let isQuizRunning=false;
let userAnswers=[];


function playQuiz(){
    quizStart.style.display='none';
    question_counter=0;
    isQuizRunning=true;
    userAnswers=[]
    showQuizQuestions();
}

function selectOption(selected,questionNumber){
    document.querySelectorAll('.option-btn').forEach(btn=>{
        btn.classList.remove('selected')
    })

    selected.classList.add('selected');
    const
selectedQuestion=userAnswers.find(question=>(question.questionNumber==questionNumber
))
    if(selectedQuestion){
        selectedQuestion.answer=selected.textContent;
    }
    else{
        userAnswers.push({
            questionNumber:questionNumber,
            answer:selected.textContent
        })
    }

}

function showResult(){
    quizBox.innerHTML="";
    let userScore=0;
    userAnswers.map((quiz)=>{
        const correctAnswer = quiz_questions[quiz.questionNumber];
        if(correctAnswer.answer==quiz.answer){
            userScore+=1;
        }
    })
    const resultScore=document.createElement("h3");
    resultScore.textContent=`Total Score: ${userScore}/${quiz_questions.length}`;
    result.appendChild(resultScore);
```

```
    const resetBtn=document.createElement('button');
    resetBtn.textContent="Reset"
    result.style.display='block'
    result.appendChild(resetBtn)

    resetBtn.addEventListener("click",()=>{window.location.reload()})
}

function showQuizQuestions() {
    quizBox.innerHTML = "";
    let answerSelected = false;

    const question = quiz_questions[question_counter];

    const title = document.createElement("h3");
    title.textContent = `${question_counter + 1}. ${question.question}`;
    quizBox.appendChild(title);

    const ansChoices = document.createElement('div');
    ansChoices.classList.add('ansChoices');

    question.options.forEach(option => {
        const btn = document.createElement('button');
        btn.textContent = option;
        btn.classList.add('option-btn');
        btn.addEventListener('click', () => {
            selectOption(btn, question_counter);
            answerSelected = true;
        });
        ansChoices.appendChild(btn);
    });

    quizBox.appendChild(ansChoices);

    const nextButton = document.createElement("button");
    nextButton.classList.add('next-btn');
    nextButton.textContent = (question_counter === quiz_questions.length - 1) ?
"Result" : "Next";

    nextButton.addEventListener("click", () => {
        if (!answerSelected) {
            alert("Choose answer first before moving to next question!");
            return;
        }
        if (nextButton.textContent === 'Result') {
            showResult();
            return;
        }
        question_counter += 1;
        showQuizQuestions();
```

```
    });

    quizBox.appendChild(nextButton);
}
```

5. Output





Conclusion

In assignments 1 to 5, I learned how to use JavaScript to dynamically interact with the user and the webpage. I practiced performing calculations and storing history in arrays, managing student data and computing grades, creating and updating tasks in a to-do list, generating random numbers and tracking scores in a guessing game, and building an interactive quiz that tracks answers and calculates results. Through these tasks, I improved my skills in DOM manipulation, event handling, conditional logic, arrays, objects, and dynamically updating the UI to provide feedback based on user actions.