# Green University of Bangladesh
# Department of Computer Science and Engineering (CSE)

**Faculty of Sciences and Engineering**

**Semester: (Spring, Year:2024), B.Sc. in CSE (Day)**

**Lab Report NO: 03**

**Course Title: Web Programming Lab**

**Course Code: CSE 302 Section: 231_D3**

**Lab Experiment Name: Introduction to JavaScript (JS) & Advance**

**Student Details**

| | Name | ID |
|---|---|---|
| 1. | Promod Chandra Das | 231002005 |

**Lab Date: 04.05.2025**

**Submission Date: 06.05.2025**

**Course Teacher's Name: Mr.Mozdaher Abdul Quader**

**Lab Report Status**

**Marks: ……**

**Comments:.....**

**Signature:.....**

**Date:.....**

❖ **TITLE OF THE LAB REPORT EXPERIMEN :- Introduction to JavaScript (JS) and Advance**

❖ **OBJECTIVES/AIM :**

The aim of this topic is to provide learners with a comprehensive understanding of JavaScript, starting from its fundamental concepts to more advanced techniques. It focuses on building the ability to write dynamic and interactive scripts for web development. The objective is to familiarize students with the syntax, variables, control structures, and functions of JavaScript, and gradually introduce advanced features such as objects, arrays, events, DOM manipulation, ES6 features, and asynchronous programming using promises and async/await. By the end of the course, students should be able to apply JavaScript effectively in real-world web applications.

## ❖ Procedure :

To achieve these objectives, learners will first be introduced to the basic structure and syntax of JavaScript using hands-on coding examples embedded in HTML documents. They will then practice writing simple scripts to perform calculations, validate forms, and manipulate content dynamically. As the course progresses, students will work on more complex tasks, including using functions, loops, and conditional statements. Practical activities will include interacting with the Document Object Model (DOM), responding to events, and using modern JavaScript features such as arrow functions, template literals, and modules. The learning process will be supported by interactive tutorials, real-time coding exercises, and mini-projects that simulate real-world web development scenarios.

## ❖ Implementation

✓ Problem 1: Write a JavaScript function that checks whether a passed string palindrome or not? Try to incorporate the concepts of function, object, and event in your solution.Note: A palindrome is word, phrase, or sequence that reads the same backward as forward, e.g., madam.

## ➢ Answer to the Q No 01:

```
<!DOCTYPE html>

<html>

<head>

  <title>Palindrome Checker</title>

</head>

<body>
```

```
<h3>Palindrome Checker</h3>
<input type="text" id="inputText" placeholder="Enter a word or phrase">
<button onclick="palindromeChecker.check()">Check</button>
<p id="result"></p>

<script>
    const palindromeChecker = {
    input: document.getElementById("inputText"),
    result: document.getElementById("result"),



    check: function() {
      const str = this.input.value.toLowerCase().replace(/[^a-z0-9]/g, '');
      const reversed = str.split('').reverse().join('');
      if (str && str === reversed) {
        this.result.innerText = `"${this.input.value}" is a palindrome.`;
      } else {
        this.result.innerText = `"${this.input.value}" is not a palindrome.`;
      }
    }
  };
</script>
```

```
</body>

</html>
```

- **Output :**

When a user enters a string and clicks the "Check" button, the JavaScript function is triggered through an event. The function processes the input string by removing non-alphanumeric characters and comparing it to its reverse. If both match, a message such as "madam" is a palindrome. is displayed on the screen. Otherwise, the result will show that the input is not a palindrome — for example, "hello" is not a palindrome.

- ✓ **Problem 02:**

i.Write a JavaScript function to get the values of First and Last name of the following form.

Ii.Write a JavaScript program to count and display the items of a dropdown list, in an alert window.
Iii. Write a module that exports a class with a constructor and methods, utilizing fetch to retrieve data and update the DOM based on the data retrieved.

**Answer To the Q No: (i)**

```
<!DOCTYPE html>
<html>
<head>
  <title>Get Name Values</title>
</head>
<body>

  <form id="nameForm">
    First name: <input type="text" id="firstName"><br><br>
    Last name: <input type="text" id="lastName"><br><br>
    <button type="button" onclick="getName()">Submit</button>
  </form>

  <script>
    function getName() {
      let first = document.getElementById("firstName").value;
      let last = document.getElementById("lastName").value;
      alert("First Name: " + first + "\nLast Name: " + last);
    }
  </script>

</body>
</html>
```

OUTPUT:

First Name: Alice
Last Name: Smith

**Answer To the Q No: (ii)**

```
<!DOCTYPE html>
<html>
<head>
  <title>Dropdown Counter</title>
</head>
<body>

  <select id="mySelect">
    <option>JavaScript</option>
    <option>Python</option>
    <option>Java</option>
    <option>C++</option>
  </select>
  <br><br>
  <button onclick="countOptions()">Count Items</button>

  <script>
    function countOptions() {
      let select = document.getElementById("mySelect");
      let count = select.options.length;
      alert("Number of items in dropdown: " + count);
    }
  </script>

</body>
</html>
```

OUTPUT:

Number of items in dropdown: 4

Answer to the Q no(iii):

**module.js (Assuming you're using ES6 modules)**

```
javascript
Copy code
// module.js
export class DataFetcher {
  constructor(url, elementId) {
    this.url = url;
    this.elementId = elementId;
  }

  async fetchData() {
    try {
      const response = await fetch(this.url);
      const data = await response.json();
      this.updateDOM(data);
    } catch (error) {
      console.error("Fetch error:", error);
    }
  }

  updateDOM(data) {
    const element = document.getElementById(this.elementId);
    element.innerHTML = JSON.stringify(data, null, 2);
  }
}
```

**□ main.html (Using the module)**

```
html
Copy code
<!DOCTYPE html><html type="module"><head>
  <title>Fetch Data</title></head><body>

  <h3>Fetched Data:</h3>
  <pre id="output"></pre>

  <script type="module">
    import { DataFetcher } from './module.js';

    const fetcher = new DataFetcher('https://jsonplaceholder.typicode.com/posts/1',
'output');
```

```
    fetcher.fetchData();
  </script>
</body></html>
```

OutPut:


```
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
  "body": "quia et suscipit..."
}
```


**ANALYSIS AND DISCUSSION :**

Each of these JavaScript functions demonstrates key aspects of web development: working with forms, manipulating DOM elements, and handling asynchronous data. While the examples are simple, they lay the foundation for more complex interactions and data-driven web applications. Enhancing user feedback, handling errors, and optimizing code structure will be crucial for building more robust and scalable applications.

Would you like this content further elaborated or converted into a more detailed report format?