Experiment No. 4

Aim:

Javascript: Variables, Operators, Conditions, Loops, Functions, Events, Classes and Objects, Error handling, Validations, Arrays, String, Date.

A. Theory:

1. What is Java Script?

JavaScript (often abbreviated as JS) is a high-level, interpreted programming language primarily used for front-end web development. It is one of the core technologies for building interactive and dynamic websites. JavaScript allows developers to create interactive elements, handle user interactions, and dynamically modify the content of web pages.

2. Why Use Java Script?

1. Front-end Interactivity: JavaScript adds interactivity and dynamism to web pages, making them more engaging for users.

2. Client-Side Scripting: JavaScript runs directly in the user's web browser, reducing server load and improving performance.

3. Browser Compatibility: Supported by all modern browsers, making it cross-platform.

4. DOM Manipulation: JavaScript can interact with a web page's DOM, allowing real-time content updates.

5. Asynchronous Programming: Supports non-blocking tasks for efficient web applications.

6. Rich Ecosystem: Vast array of libraries and frameworks for rapid development.

7. Node.js: Enables server-side development with a single language.

8. Active Community: Large developer community with abundant resources and support.

9. Cross-Platform Development: Used for building mobile applications with frameworks like React Native and Ionic.

10. High Demand: High demand for JavaScript developers due to its widespread use in web development.

3. How to use Java Script in HTML?

1. To use JavaScript in HTML, you can include a <script> tag in our HTML file.
2. In the same directory as your HTML file, create a JavaScript file (e.g., script.js) with your JavaScript code:

B. Program:

HTML:

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="style.css">
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
    <div class="container mt-5">
        <h1 class="mb-4">Expense Tracker</h1>
        <div class="form-row">
            <div class="col-md-4">
                <label for="expenseInput">Expense Description:</label>
                <input type="text" class="form-control" id="expenseInput"
placeholder="Enter expense description...">
            </div>
            <div class="col-md-3">
                <label for="amountInput">Amount:</label>
                <input type="number" class="form-control" id="amountInput"
placeholder="Enter amount...">
            </div>
            <div class="col-md-3">
                <label for="dateInput">Date:</label>
                <input type="date" class="form-control" id="dateInput">
            </div>
            <div class="col-md-2">
                <button class="btn btn-primary btn-block mt-md-4"
onclick="addExpense()">Add Expense</button>
            </div>
        </div>
        <table class="table mt-4">
            <thead>
                <tr>
                </tr>
            </thead>
            <tbody id="expenseTable">
            </tbody>
        </table>
    </div>

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

JavaScript:

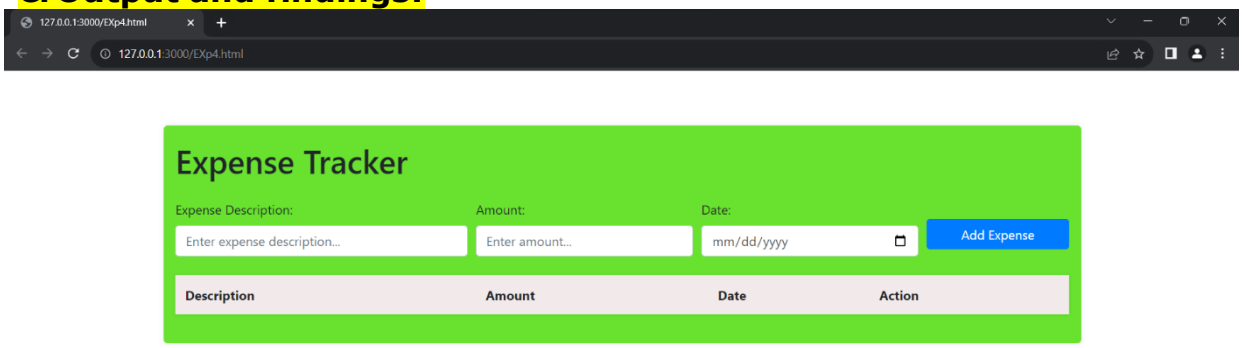
```
// Expense class to represent an expense
class Expense {
    constructor(description, amount, date) {
        this.description = description;
        this.amount = amount;
        this.date = date;
    }
}

// Array to store expenses
let expenses = [];

// Function to add a new expense
function addExpense() {
    const expenseDescription =
document.getElementById("expenseInput").value.trim();
    const expenseAmount =
parseFloat(document.getElementById("amountInput").value.trim());
    const expenseDate = document.getElementById("dateInput").value;

    // Validation: Check if any field is blank
    if (expenseDescription === "" || isNaN(expenseAmount) || expenseAmount <= 0
|| expenseDate === "") {
        alert("Please fill all fields before adding the expense.");
        return;
    }

    const newExpense = new Expense(expenseDescription, expenseAmount,
expenseDate);
    expenses.push(newExpense);

    displayExpenses();
    clearInputs();
}

// Function to display the list of expenses
function displayExpenses() {
    const expenseTable = document.getElementById("expenseTable");
    expenseTable.innerHTML = `
        <tr>
            <th>Description</th>
            <th>Amount</th>
            <th>Date</th>
            <th>Action</th>
        </tr>
    `;

    expenses.forEach((expense, index) => {
        const row = document.createElement("tr");
        row.innerHTML = `
            <td>${expense.description}</td>
            <td>${expense.amount.toFixed(2)}</td>
            <td>${expense.date}</td>
            <td><button onclick="deleteExpense(${index})">Delete</button></td>
        `;
```

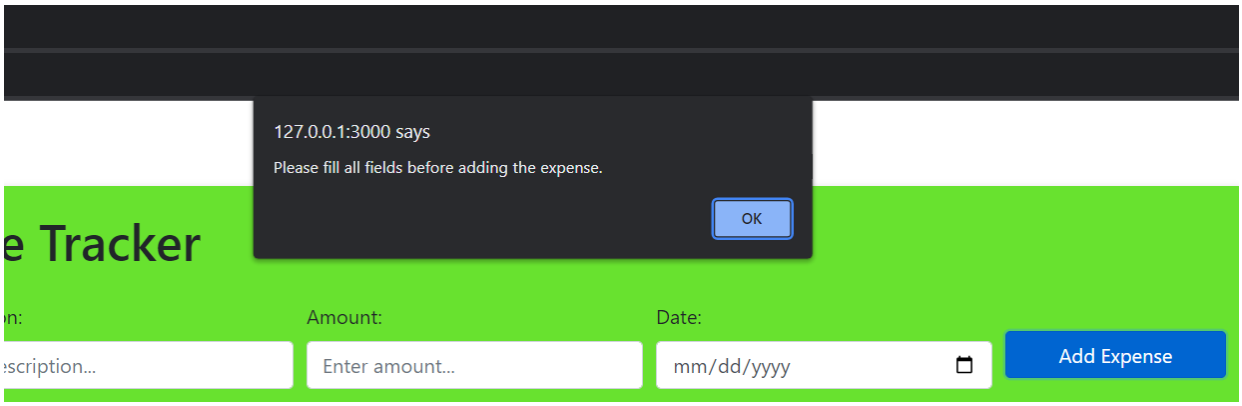
```
        expenseTable.appendChild(row);
    });
}

// Function to delete an expense
function deleteExpense(index) {
    expenses.splice(index, 1);
    displayExpenses();
}

// Function to clear input fields
function clearInputs() {
    document.getElementById("expenseInput").value = "";
    document.getElementById("amountInput").value = "";
    document.getElementById("dateInput").value = "";
}

// Initial display of expenses
displayExpenses();
```

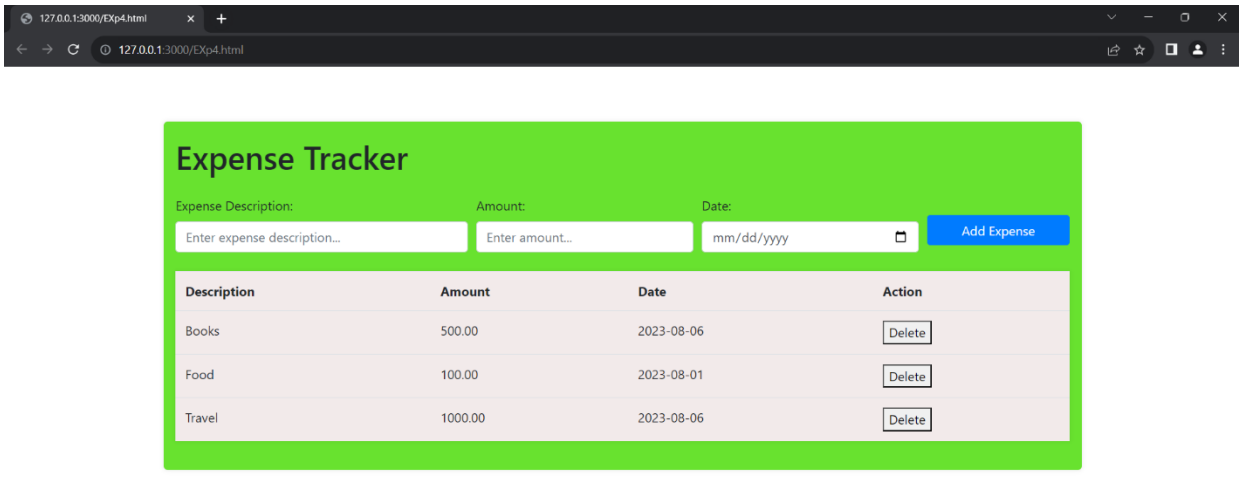
CSS:

```css
body {
    padding: 20px;
    background-color: #d98a8a;
}

.container {
    max-width: 800px;
    margin: 0 auto;
    background-color: #68e22f;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
}

.form-row {
    margin-bottom: 10px;
}

table {
    margin-top: 20px;
    background-color: #f2eaea;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
}

th, td {
    padding: 10px;
}

.btn {
    margin-top: 20px;
    padding: 10px 20px;

}
```

C. Output and findings:







D. Conclusion

The above project demonstrates an expense tracker application using JavaScript in HTML. JavaScript is effectively used to handle user interactions, validate and process input data, dynamically update the DOM to display expenses, and manage the state of the application. By utilizing JavaScript's event-driven nature and DOM manipulation capabilities, the application becomes interactive, allowing users to add, view, and delete expenses in real-time without page refreshes. Additionally, JavaScript's flexibility and ecosystem enable the project to be extended with additional features, improving the overall user experience and functionality of the expense tracker.

JavaScript Components Used:

1. Variables: Used to store input values, expense data, and the array of expenses.
2. Functions: Defined to add expenses, display expenses, delete expenses, and clear input fields.
3. Classes: The Expense class is used to create expense objects with description, amount, and date properties.
4. Arrays: The expenses array stores the list of expense objects.
5. DOM Manipulation: JavaScript interacts with the DOM to dynamically update the table with expenses.
6. Event Handling: Event listeners are attached to the "Add Expense" button and the "Delete" buttons for user interaction.
7. Error Handling: Validation checks ensure that no field is left blank before adding an expense.
By combining these JavaScript components, the project provides a functional and user-friendly expense tracker that enables users to manage and view their expenses in a simple and efficient manner.
8.Conditions: if statements are used in the project for validation. Let's update the conclusion to include the usage of conditions: