

Name : Atharva Prabhu

Class : D15-A

Roll no. : 42

Experiment – 9: AJAX

1) **Aim:** To study AJAX

2) **Theory:**

A. How do Synchronous and Asynchronous Requests differ?

Synchronous and asynchronous requests differ in how they interact with the main thread of execution. In a synchronous request, the code execution waits for the response from the server before moving on to the next line of code. This means the browser will be unresponsive until the request completes. On the other hand, an asynchronous request allows the rest of the code to continue running while waiting for the server response. This makes the application more responsive and improves the user experience, as it does not block other operations. Asynchronous requests are commonly used in web applications to update content dynamically without reloading the entire page.

B. Describe various properties and methods used in XMLHttpRequest Object

The XMLHttpRequest object is used to send and receive data asynchronously in web applications. It has several important properties and methods. Key properties include `readyState`, which shows the state of the request, `status` and `statusText` for response status, and `responseText` to access the returned data. The `onreadystatechange` property defines a function to be called whenever the `readyState` changes. Key methods include `open()` to initialize the request, `send()` to send it, `setRequestHeader()` to set HTTP headers, and `abort()` to cancel the request. These allow developers to create dynamic and interactive web pages without needing to reload the entire page.

3) **Problem Statement:**

Create a registration page having fields like Name, College, Username and Password (read password twice).

Validate the form by checking for

1. Username is not same as existing entries
2. Name field is not empty
3. Retyped password is matching with the earlier one. Prompt a message is

And also auto suggest college names.

Show the message "Successfully Registered" on the same page below the submit button, on Successfully registration. Let all the updations on the page be Asynchronously loaded. Implement the same using XMLHttpRequest Object.

4) Code:

Server.php:

```
<?php

// Simulate existing usernames

$existing_usernames = ["john123", "alice", "testuser"];

$name = $_POST['name'] ?? "";
$college = $_POST['college'] ?? "";
$username = $_POST['username'] ?? "";
$password = $_POST['password'] ?? "";

if (in_array(strtolower($username), $existing_usernames)) {
    echo "Username already taken. Please choose another.";
} else {
    // Simulate saving to DB...

    echo "Successfully Registered";
}

?>
```

script.js:

```
document.getElementById('regForm').addEventListener('submit', function (e) {  
    e.preventDefault(); // Prevent default form submit  
  
    let name = document.getElementById('name').value.trim();  
    let college = document.getElementById('college').value.trim();  
    let username = document.getElementById('username').value.trim();  
    let password = document.getElementById('password').value;  
    let repassword = document.getElementById('repassword').value;  
    let message = document.getElementById('message');  
  
    // Clear message area  
    message.textContent = "";  
    message.className = "";  
  
    // Basic validation  
    if (!name) {  
        message.textContent = "Name cannot be empty.";  
        message.className = 'error';  
        return;  
    }  
  
    if (password !== repassword) {
```

```
    message.textContent = "Passwords do not match.";

    message.className = 'error';

    return;
}
```

// Create AJAX Request

```
let xhr = new XMLHttpRequest();

xhr.open("POST", "server.php", true);

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onload = function () {

    if (this.status === 200) {

        let response = this.responseText;

        message.textContent = response;

        message.className = response.includes("Successfully") ? "message" : "error";

    }

};
```

```
let params =
`name=${name}&college=${college}&username=${username}&password=${password}`;

xhr.send(params);

});
```

```

// College auto-suggest

document.getElementById("college").addEventListener("input", function () {

    let query = this.value.trim();

    let suggestionBox = document.getElementById("collegeSuggestions");

    if (query.length < 1) {

        suggestionBox.innerHTML = "";

        return;

    }

    let colleges = ["MIT", "Harvard", "Stanford", "IIT Delhi", "IIT Bombay", "Oxford",
"Cambridge"];

    let filtered = colleges.filter(c => c.toLowerCase().startsWith(query.toLowerCase()));

    suggestionBox.innerHTML = filtered.map(c => `<div
onclick="selectCollege('${c}')">${c}</div>`).join("");

});

function selectCollege(name) {

    document.getElementById("college").value = name;

    document.getElementById("collegeSuggestions").innerHTML = "";

}

```

Index.html:

```
<!DOCTYPE html>
```

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

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <title>Registration Form</title>
```

```
  <style>
```

```
    input {
```

```
      margin: 5px 0;
```

```
      display: block;
```

```
    }
```

```
    .message {
```

```
      color: green;
```

```
      margin-top: 10px;
```

```
    }
```

```
    .error {
```

```
      color: red;
```

```
    }
```

```
  </style>
```

```
</head>
```

<body>

<h2>Registration Form</h2>

<form id="regForm">

Name: <input type="text" id="name" name="name" required>

College: <input type="text" id="college" name="college" autocomplete="off">

<div id="collegeSuggestions"></div>

Username: <input type="text" id="username" name="username" required>

Password: <input type="password" id="password" name="password" required>

Re-type Password: <input type="password" id="repassword" name="repassword" required>

<button type="submit">Register</button>

</form>

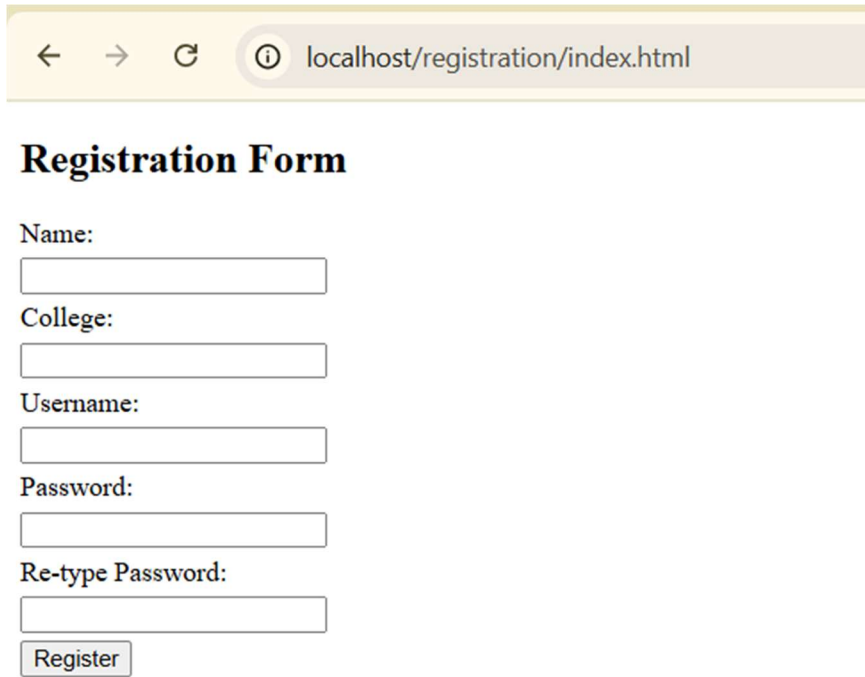
<div id="message" class="message"></div>

<script src="script.js"></script>

</body>

</html>

5) **Output:**



The screenshot shows a web browser window with a yellow address bar. The address bar contains navigation icons (back, forward, refresh) and an information icon, followed by the text "localhost/registration/index.html". Below the address bar, the page title "Registration Form" is displayed in a bold, black, serif font. The form itself consists of five labeled text input fields stacked vertically: "Name:", "College:", "Username:", "Password:", and "Re-type Password:". Each label is in a black, serif font. At the bottom of the form is a "Register" button with a light gray background and a thin black border.

localhost/registration/index.html

Registration Form

Name:

College:

Username:

Password:

Re-type Password:

Registration Form

Name:

College:

Username:

Password:

Re-type Password:

Successfully Registered

Conclusion:

Learned how to build a dynamic and user-friendly registration form using XMLHttpRequest for asynchronous data handling includes form validation, real-time username checks, and college name suggestions without reloading the page.

