Experiment 9 : To study AJAX

Name of Student	**Gunjan Chandnani**
Class Roll No	**D15A_06**
D.O.P.	
D.O.S.	
Sign and Grade	

Aim: To study AJAX

Theory:-

1. How do Synchronous and Asynchronous Requests differ?

Synchronous Requests:

- The browser waits for the server to respond before executing the rest of the JavaScript code.
- UI becomes unresponsive or freezes during the request.
- Leads to a poor user experience if the server is slow.

Example:

If the browser is waiting for data, the user cannot interact with the page (e.g., clicking, typing) until the response arrives.

Asynchronous Requests:

- The browser does not wait for the server response.
- JavaScript continues executing, and the UI remains responsive.
- A callback or event is used to handle the response once it's received.

Example:

In live search suggestions, users keep typing while results are fetched and displayed dynamically without reloading the page.

Key Differences:

Feature	Synchronous	Asynchronous
Browser Behavior	Waits for response	Does not wait
User Experience	Freezes UI	UI stays responsive
Implementation	Simple but blocking	Needs callbacks/promises
Modern Usage	Rare	Preferred

2. Describe various properties and methods used in XMLHttpRequest Object

The XMLHttpRequest object is used in JavaScript to exchange data with a server **without reloading** the webpage – a key component of AJAX.

- **Common Properties:**

Property	Description
`readyState`	Indicates the state of the request (0–4).
`status`	HTTP status code of the response (e.g., 200).
`statusText`	Text version of the status (e.g., "OK").
`responseText`	Response data as a string.
`responseXML`	Response data as an XML document (if any).

🔹 **readyState Values:**

Value	State	Description
0	UNSENT	Request not initialized
1	OPENED	`open()` has been called
2	HEADERS_RECEIVED	`send()` has been called
3	LOADING	Downloading response
4	DONE	Request finished and response ready

Problem Statement Summary

You are asked to:

- Create a registration page with:
  - Name
  - College (with auto-suggest)
  - Username
  - Password (entered twice)
- Validate the following:
  - Username is **not** already taken.
  - Name field is **not empty**.
  - Passwords **match**.
- Show "**Successfully Registered**" on the same page.
- Perform all form validation and message updates using **Asynchronous (XMLHttpRequest)** logic.

CODE:-

Intex.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Registration Page</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background: #f0f4f8;
      margin: 0;
      padding: 20px;
    }

    .container {
      background: #ffffff;
      max-width: 500px;
      margin: 40px auto;
      padding: 35px;
      border-radius: 12px;
      box-shadow: 0 10px 25px rgba(0,0,0,0.1);
    }

    h2 {
      text-align: center;
      color: #0d47a1;
      margin-bottom: 25px;
    }

    label {
      display: block;
      margin-top: 18px;
      font-weight: 600;
      color: #333;
    }

    input[type="text"],
    input[type="password"],
    input[list] {
      width: 100%;
      padding: 12px;
      margin-top: 6px;
      border: 1px solid #ccc;
      border-radius: 6px;
```

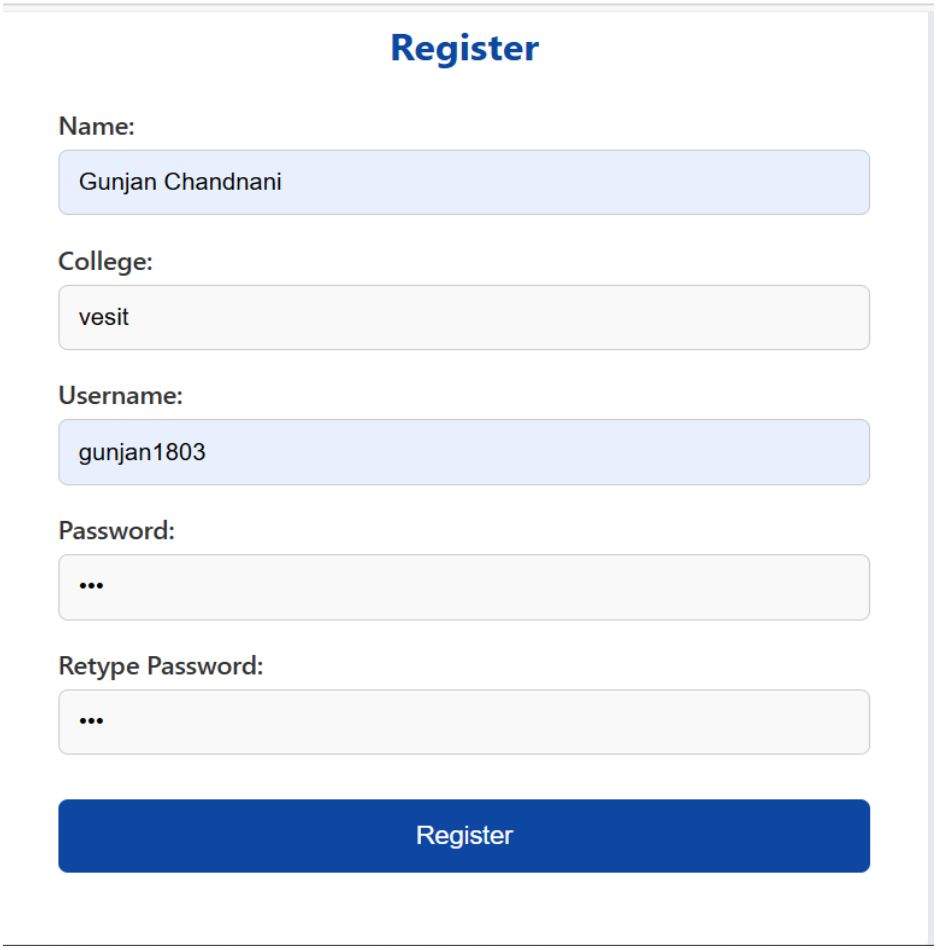
```
      box-sizing: border-box;
      font-size: 15px;
      background-color: #f9f9f9;
    }

    button {
      margin-top: 25px;
      width: 100%;
      padding: 14px;
      background-color: #0d47a1;
      color: white;
      border: none;
      border-radius: 6px;
      font-size: 16px;
      cursor: pointer;
      transition: background 0.3s;
    }

    button:hover {
      background-color: #08306b;
    }

    .message {
      margin-top: 20px;
      font-size: 15px;
      color: #d32f2f;
      text-align: center;
    }

    .success {
      color: #2e7d32;
    }

    .input-feedback {
      font-size: 13px;
      color: #d32f2f;
      margin-top: 4px;
    }
  </style>
</head>
<body>

<div class="container">
  <h2>Register</h2>
  <form id="regForm" onsubmit="return false;">
    <label for="name">Name:</label>
```

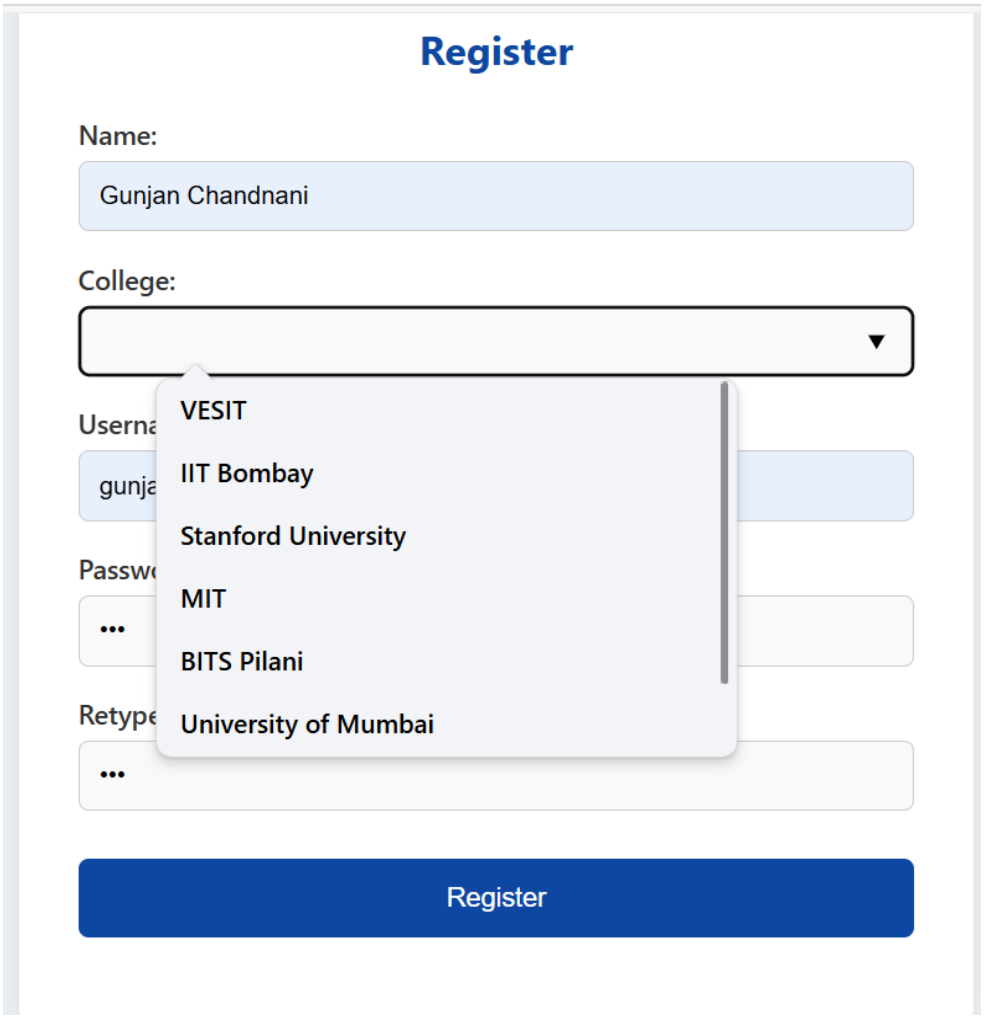
```
      <input type="text" id="name">
      <div class="input-feedback" id="nameFeedback"></div>

      <label for="college">College:</label>
      <input list="colleges" id="college">
      <datalist id="colleges">
        <option value="VESIT">
        <option value="IIT Bombay">
        <option value="Stanford University">
        <option value="MIT">
        <option value="BITS Pilani">
        <option value="University of Mumbai">
        <option value="Harvard University">
      </datalist>

      <label for="username">Username:</label>
      <input type="text" id="username" onblur="checkUsernameAsync()">
      <div class="input-feedback" id="usernameFeedback"></div>

      <label for="password">Password:</label>
      <input type="password" id="password">

      <label for="confirmPassword">Retype Password:</label>
      <input type="password" id="confirmPassword">
      <div class="input-feedback" id="passwordFeedback"></div>

      <button type="button" onclick="submitForm()">Register</button>
    </form>

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

  <script>
  const existingUsernames = ["sanket123", "john_doe", "admin", "guest"];

  function checkUsernameAvailable(username) {
    return new Promise((resolve) => {
      const taken = existingUsernames.includes(username.trim());
      const result = {
        available: !taken,
        message: taken ? "Username already exists." : "Username is available."
      };

      setTimeout(() => resolve(result), 500);
    });
  }
```

```
async function submitForm() {
  const name = document.getElementById("name").value.trim();
  const college = document.getElementById("college").value.trim();
  const username = document.getElementById("username").value.trim();
  const password = document.getElementById("password").value;
  const confirmPassword = document.getElementById("confirmPassword").value;

  const nameFeedback = document.getElementById("nameFeedback");
  const usernameFeedback = document.getElementById("usernameFeedback");
  const passwordFeedback = document.getElementById("passwordFeedback");
  const resultMessage = document.getElementById("resultMessage");

  nameFeedback.textContent = "";
  usernameFeedback.textContent = "";
  passwordFeedback.textContent = "";
  resultMessage.textContent = "";
  resultMessage.classList.remove("success");

  let valid = true;

  if (name === "") {
    nameFeedback.textContent = "Name cannot be empty.";
    valid = false;
  }

  if (password !== confirmPassword) {
    passwordFeedback.textContent = "Passwords do not match.";
    valid = false;
  }

  const usernameCheck = await checkUsernameAvailable(username);
  usernameFeedback.textContent = usernameCheck.message;
  usernameFeedback.style.color = usernameCheck.available ? "green" : "#d32f2f";

  if (!usernameCheck.available) {
    valid = false;
  }

  if (!valid) return;

  setTimeout(() => {
    resultMessage.textContent = "Successfully Registered";
    resultMessage.classList.add("success");
  }, 1000);
}
```

```
document.getElementById("username").addEventListener("input", () => {
   document.getElementById("usernameFeedback").textContent = "";
});
</script>

</body>
</html>
```

OUTPUT:-




Register
Name:
Gunjan Chandnani
College:
VESIT
IIT Bombay
Stanford University
MIT
BITS Pilani
University of Mumbai
Register


Username:
gunjan1803
Username is available.

Register

Name:

Gunjan Chandnani

College:

VESIT

Username:

gunjan

Username is available.

Password:

•••

Retype Password:

•••

Register

Successfully Registered