Experiment: Implementing AJAX for Web Page Login Process

Aim

To develop a web page that performs a user login operation using **AJAX**, ensuring asynchronous communication with the server without reloading the page.

Software & Hardware Requirements

- HTML5
- CSS3
- JavaScript
- AJAX (XMLHttpRequest / Fetch API)
- Web server (Tomcat / Node.js / XAMPP)
- Backend script (Servlet / PHP / Node.js)
- Browser (Chrome/Firefox)
- System with minimum 4GB RAM

Theory

AJAX (Asynchronous JavaScript and XML) allows web pages to retrieve data from a server asynchronously without refreshing the entire page. It improves user experience and makes applications faster and more interactive.

Key AJAX Concepts

- **XMLHttpRequest** / **Fetch API** is used to send requests.
- **Asynchronous Processing** allows page updates without reload.
- **JSON Response Handling** makes data parsing easy.

Algorithm

1. Create a login form with username and password fields.
2. Attach JavaScript validation for empty fields.
3. Use AJAX (Fetch API) to send login credentials to the server.
4. The backend script verifies the credentials from the database.
5. The backend returns a JSON response.
6. Display success or error message on the same page without reloading.

Program

1. HTML (login.html)

```html
<!DOCTYPE html>
<html>
<head>
    <title>AJAX Login</title>
    <style>
        body { font-family: Arial; margin: 50px; }
        .box { width: 300px; padding: 20px; border: 1px solid #333; border-
radius: 5px; }
        input { width: 100%; padding: 8px; margin-top: 10px; }
        #msg { margin-top: 15px; font-weight: bold; }
    </style>
</head>
<body>

<div class="box">
    <h3>Login</h3>
    <input type="text" id="username" placeholder="Enter Username">
    <input type="password" id="password" placeholder="Enter Password">
    <button onclick="login()">Login</button>
    <div id="msg"></div>
</div>

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

---

2. JavaScript (login.js)

```javascript
function login() {
    let user = document.getElementById("username").value;
    let pass = document.getElementById("password").value;
    let msgBox = document.getElementById("msg");

    if (user === "" || pass === "") {
        msgBox.style.color = "red";
        msgBox.innerHTML = "All fields are required!";
        return;
    }

    fetch("loginServlet", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ username: user, password: pass })
    })
    .then(response => response.json())
    .then(data => {
        if (data.status === "success") {
            msgBox.style.color = "green";
            msgBox.innerHTML = "Login Successful!";
        } else {
            msgBox.style.color = "red";
            msgBox.innerHTML = "Invalid Credentials!";
        }
    })
    .catch(error => console.error("Error:", error));
}
```

---

3. Backend (Servlet: LoginServlet.java)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import org.json.JSONObject;

public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        BufferedReader reader = request.getReader();
        StringBuilder sb = new StringBuilder();
        String line;

        while((line = reader.readLine()) != null) sb.append(line);

        JSONObject json = new JSONObject(sb.toString());
        String user = json.getString("username");
        String pass = json.getString("password");

        response.setContentType("application/json");
        JSONObject res = new JSONObject();

        if (user.equals("admin") && pass.equals("1234")) {
            res.put("status", "success");
        } else {
            res.put("status", "error");
        }

        PrintWriter out = response.getWriter();
        out.print(res);
    }
}
```

Output

- Displays login form.
- Validates inputs using JavaScript.
- Sends credentials to server asynchronously.
- Displays success/error message without page reload.

Result

Thus, an AJAX-based login system was successfully implemented using HTML, JavaScript, and Servlet backend, enabling asynchronous validation and dynamic message updates without reloading the page.