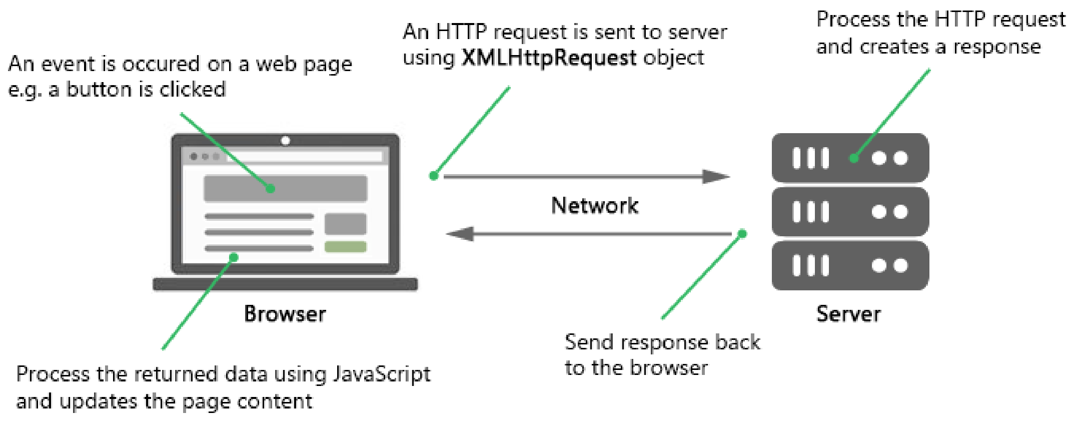
Research & Overview of AJAX Report

What is AJAX?

1. AJAX stands for **Asynchronous JavaScript And XML**.
2. Ajax is not a new technology, nor is it a new language. Instead, it is existing technologies used in a new way (a programming concept).
3. It send and retrieve data from a server asynchronously (in the background) without interfering with the display and behaviour of the existing page.
4. To change content dynamically without the need to reload the entire page.
5. It decoupling the data interchange layer from the presentation layer.

Working of AJAX

1. Some event happens on a webpage. For instance, the page loads, or a user clicks on a particular button.
2. JavaScript creates an XMLHttpRequest object.
3. This object sends a request to the corresponding web server.
4. The server processes the request and sends a response back to the browser.
5. JavaScript reads the response.
6. JavaScript performs the proper action, depending on the triggering event.


An event is occured on a web page e.g. a button is clicked
An HTTP request is sent to server using **XMLHttpRequest** object
Process the HTTP request and creates a response
Network
Browser
Server
Process the returned data using JavaScript and updates the page content
Send response back to the browser

XMLHttpRequest

It is an API in the form an object whose methods help in transfer of data between a web browser and a web server.

```
var xhttp = new XMLHttpRequest();
```

Properties of XMLHttpRequest object

`readystate` is a property of the XMLHttpRequest Object which holds the status of the XMLHttpRequest.

- 0: request not initialized
- 1: server connection established
- 2: request received
- 3: processing request
- 4: request finished and response is ready

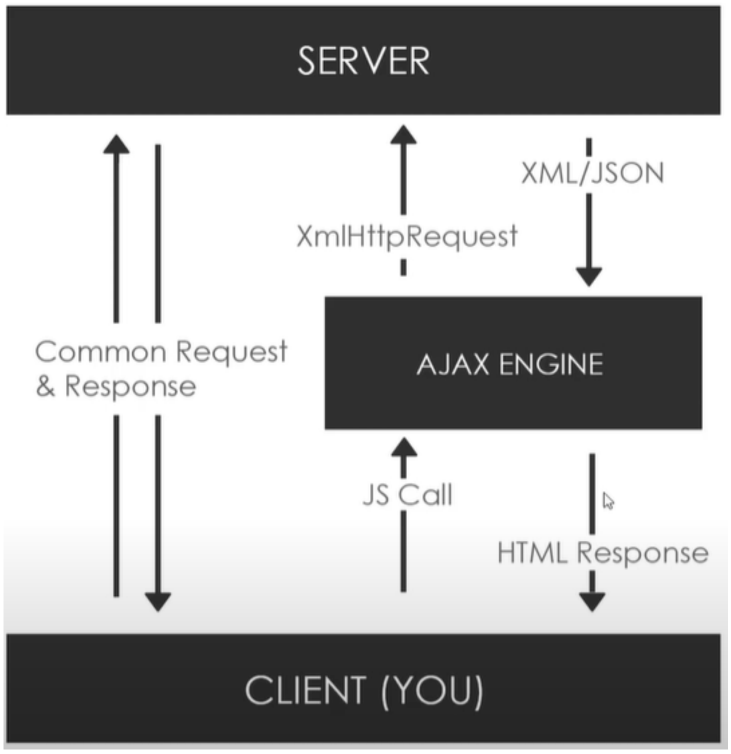
`onreadystatechange` is a property of the XMLHttpRequest Object which defines a function to be called when the readyState property changes.
`status` is a property of the XMLHttpRequest Object which returns the status-number of a request

- 200: "OK"
- 403: "Forbidden"
- 404: "Not Found"

XMLHttpRequest Object Methods :

To send a request to a Web Server, we use the open() and send() methods of the XMLHttpRequest object.

```
xhttp.open("GET", "content.txt", true);
xhttp.send();
```



JSON in AJAX

Initially, AJAX used XML as the data format for communication between the client and server. However, XML has a more complex structure and is heavier than JSON. With the advent of JSON, developers started to prefer JSON for its simplicity, readability, and ease of parsing in JavaScript. JSON became popular due to reasons below:

1. Simpler Parsing
2. Lightweight and Human-Readable

3. Easier Integration with JavaScript
4. Increased Popularity of RESTful APIs hyped JSON
5. JSONP for Cross-Domain Requests

Practical Implementation

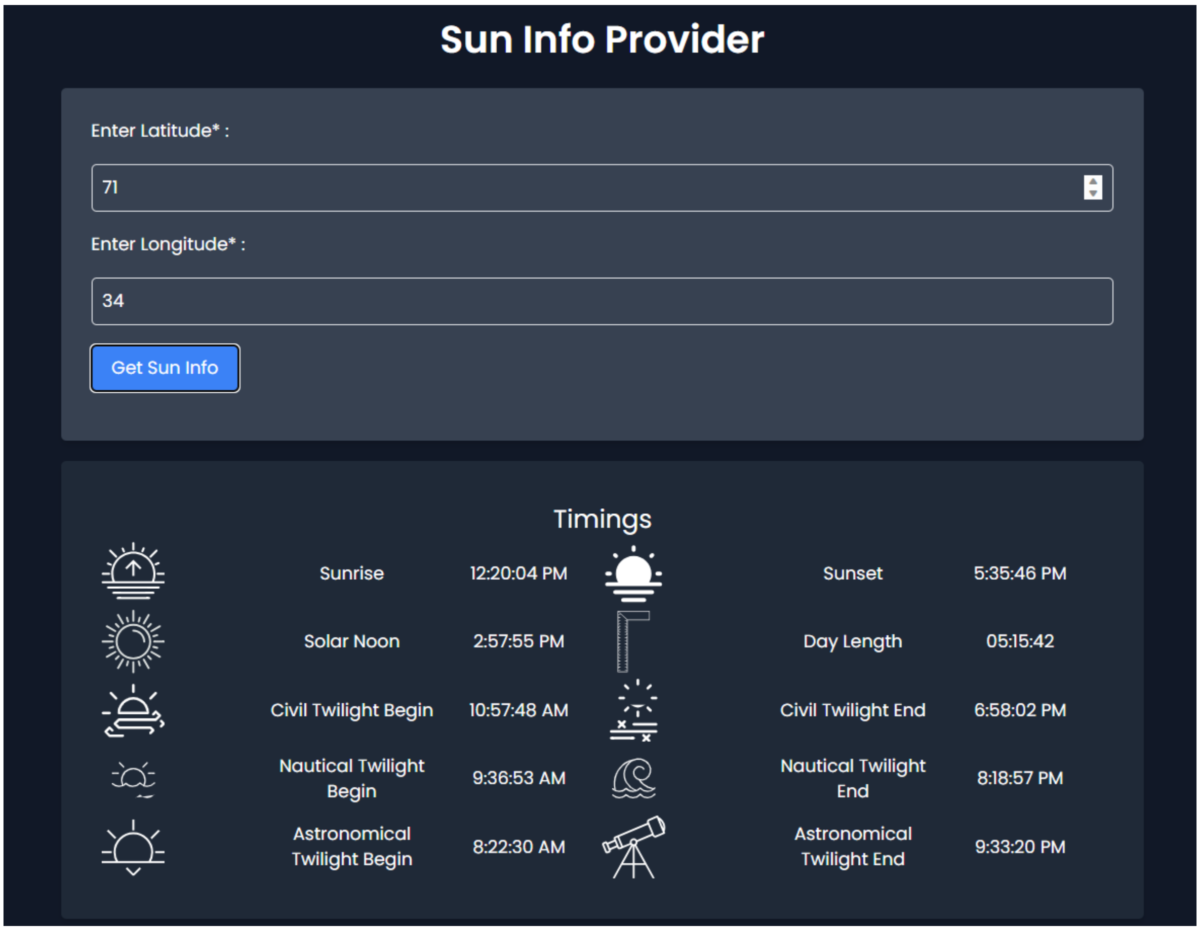
1.**API**:

I have used an Sunset-Sunrise API that provides information on the sunrise and sunset times for a given location's co-ordinates.

```
   const url = `https://api.sunrise-sunset.org/json?
lat=${latitudeInput}&lng=${longitudeInput}`;
```

2. **AJAx Call:**

```
function getSunInfo() {
  const latitudeInput = document.getElementById("latitudeInput").value;
  const longitudeInput = document.getElementById("longitudeInput").value;
  if (isValidLatitude(latitudeInput) && isValidLongitude(longitudeInput)) {
    const error = document.getElementById("error");
    error.textContent = "";

    const url = `https://api.sunrise-sunset.org/json?
lat=${latitudeInput}&lng=${longitudeInput}`;

    const xhr = new XMLHttpRequest();
    xhr.open("GET", url, true);

    xhr.onreadystatechange = function () {
      if (xhr.readyState === 4) {
        if (xhr.status === 200) {
          const data = JSON.parse(xhr.responseText);
          if (data) {
            setValues(data);
          } else {
            console.error("Error:", JSON.stringify(xhr.statusText));
          }
        } else {
          console.error("Error:", JSON.stringify(xhr.statusText));
        }
      }
    };

    xhr.onerror = function () {
      console.error("Request failed");
    };

    xhr.send();
  } else {
    const error = document.getElementById("error");
    error.textContent = "Error: Invalid latitude or longitude range";
    // console.error("Invalid latitude or longitude range");
  }
}
```


Sun Info Provider

Enter Latitude* : 71

Enter Longitude* : 34

Timings

Sunrise	12:20:04 PM	Sunset	5:35:46 PM
Solar Noon	2:57:55 PM	Day Length	05:15:42
Civil Twilight Begin	10:57:48 AM	Civil Twilight End	6:58:02 PM
Nautical Twilight Begin	9:36:53 AM	Nautical Twilight End	8:18:57 PM
Astronomical Twilight Begin	8:22:30 AM	Astronomical Twilight End	9:33:20 PM

Challenges of Using AJAX

1. **Cross-Origin Restrictions:**
2. **Security Concerns:**
   - AJAX requests can expose security vulnerabilities if not handled properly. Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks are risks associated with AJAX.
3. **Search Engine Optimization (SEO) Challenges:**
   - Search engines may have difficulties indexing content loaded asynchronously through AJAX. Traditional web crawlers often struggle to interpret content dynamically loaded after the initial page load. Solution
4. **Handling State and Back Button:**
   - Maintaining application state and handling browser navigation (back and forward buttons) can be tricky with AJAX-based applications.
5. **Concurrency and Race Conditions:**
   - In scenarios where multiple asynchronous requests are executed concurrently, race conditions can occur.

The Future of AJAX

The advancement in SPA's and modern `javascript` frameworks its importance and usage is reduce drastically.

1. **State Management:**

   Modern frameworks often employ sophisticated state management solutions (e.g., Redux in React) that coordinate data flow within the application. While AJAX may still be used for data fetching, the management of fetched data is integrated into the state management system.

2. **Data Fetching Frameworks:**

   In React, for instance, the `useEffect` hook can be utilized for data fetching, and libraries like Axios or the native `fetch` API handle AJAX requests.

3. **Real-Time Applications:**

   With the rise of real-time applications and the WebSocket protocol, AJAX is sometimes complemented or replaced by WebSocket

4. **GraphQL and Alternative Approaches:**

   The introduction of GraphQL has provided an alternative to traditional RESTful APIs, offering a more efficient way to query and manipulate data. In GraphQL-based applications, the need for multiple AJAX requests may be reduced.