AJAX

Web Technology - B.Sc. CSIT 5th Semester

Binilraj Adhikari

Ajax stands for "Asynchronous JavaScript and XML."

Ajax enables web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes.

Web pages can make asynchronous requests to a web server. This means updates can occur without reloading the entire page.

Ajax is commonly used to create responsive and interactive user interfaces, making it a fundamental part of many single-page applications (SPAs).

Ajax plays a crucial role in the development of dynamic and real-time web experiences by facilitating seamless communication between the client and server.

Advantages of AJAX

Asynchronous Updates: Ajax allows for asynchronous updates, meaning that parts of a web page can be updated independently, without requiring a full page reload. This leads to a more responsive user interface.

Improved User Experience: With Ajax, users experience faster interactions as only specific portions of the page are updated, reducing the perceived loading time and enhancing the overall user experience.

Reduced Server Load: Since only the necessary data is exchanged with the server, Ajax reduces the amount of data transferred, resulting in lower server load and improved performance.

Interactive Interfaces: Ajax facilitates the creation of interactive and dynamic user interfaces, allowing developers to build web applications that feel more like desktop applications.

Bandwidth Efficiency: By updating only the required portions of a page, Ajax applications can be more bandwidth-efficient compared to traditional page reloads.

Increased Interactivity: Ajax enables the development of real-time and interactive features, such as live search suggestions, auto-complete fields, and interactive maps.

Enhanced Usability: With asynchronous updates, users can continue interacting with the application while data is being fetched or processed in the background, improving overall usability.

Reduced Latency: As Ajax allows for asynchronous communication, it helps reduce latency since users do not have to wait for the entire page to reload.

Disadvantages of AJAX

Back Button Issues: As Ajax may not trigger a full page reload, handling the browser's back button can be challenging, potentially leading to navigation issues for users.

Search Engine Optimization (SEO) Challenges: Search engines may have difficulty indexing content loaded asynchronously, which can impact search engine rankings and discoverability.

Security Concerns: Ajax applications can be vulnerable to security issues, such as Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF). Proper security measures must be implemented to mitigate these risks.

Complexity for Developers: Implementing Ajax functionality can introduce complexity to the code, making it harder to maintain and debug, especially for larger applications.

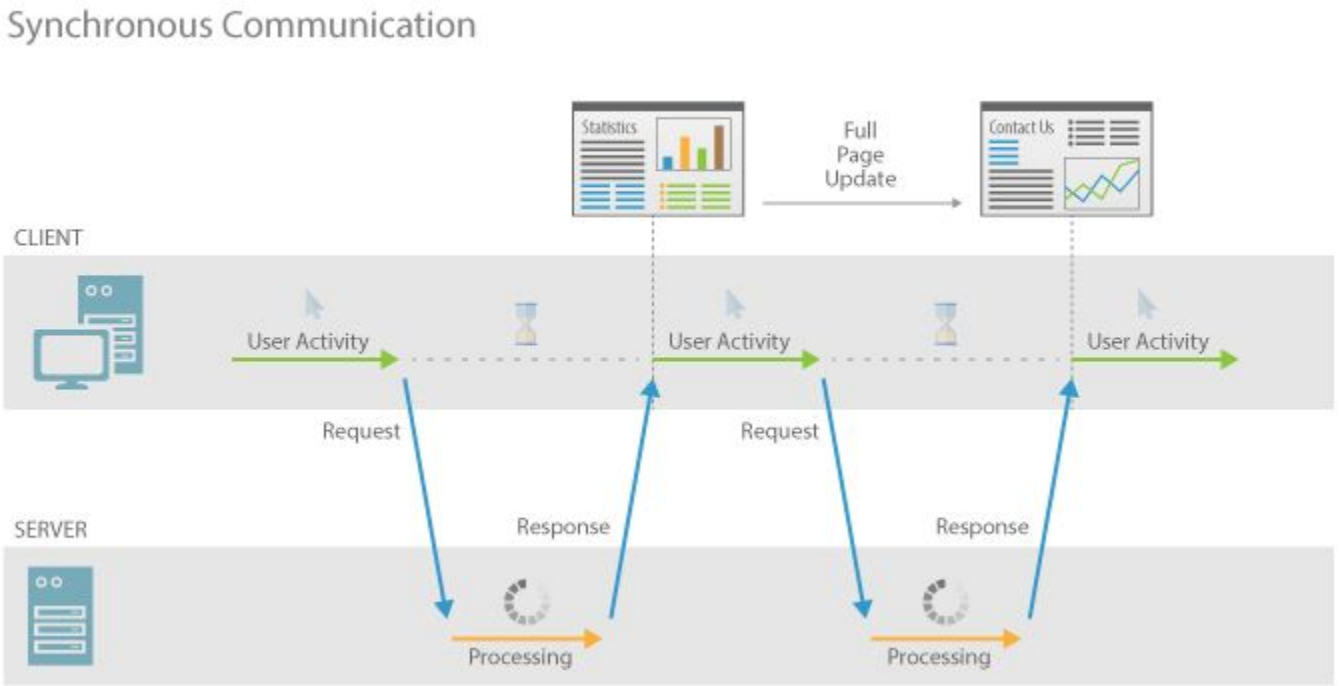
Browser Compatibility: While modern browsers generally support Ajax, there can be variations in how different browsers handle certain aspects of Ajax requests, leading to compatibility issues.

Network Dependency: Since Ajax relies on network communication, applications may experience issues or delays if there are network problems or if the user has a slow internet connection.

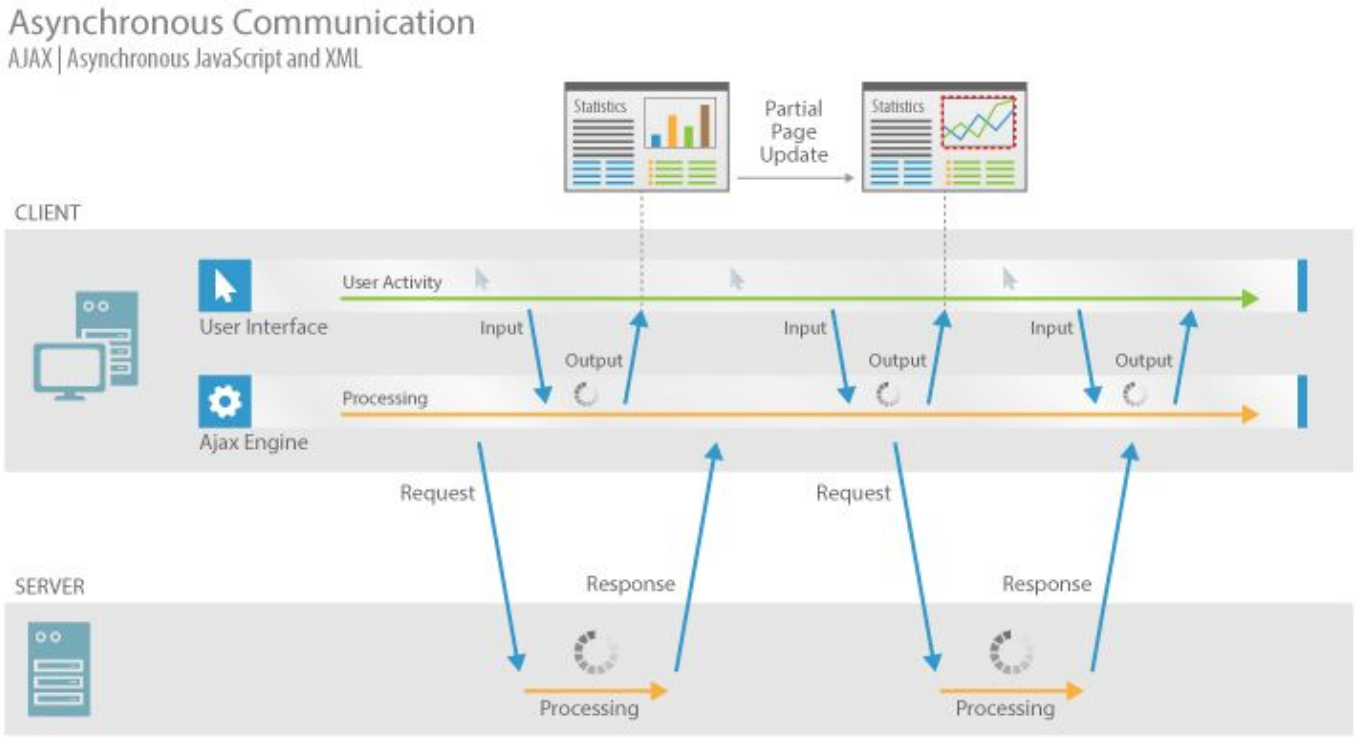
Graceful Degradation: Developers need to implement strategies for graceful degradation to ensure that the application still functions reasonably well for users who have disabled JavaScript or are using browsers that don't support Ajax.

Learning Curve: For developers new to Ajax, there may be a learning curve associated with understanding and implementing asynchronous programming and handling various Ajax-related concepts.

Synchronous Web Application Model



Asynchronous Web Application Model



Steps of AJAX Operations

- A client event occurs.
- An XMLHttpRequest object is created.
- The XMLHttpRequest object is configured.
- The XMLHttpRequest object makes an asynchronous request to the Webserver.
- The Webserver returns the result containing XML document.
- The XMLHttpRequest object calls the callback() function and processes the result.
- The HTML DOM is updated.

Properties of XMLHttpRequest(XHR) Object

Property	Description
onload	Defines a function to be called when the request is received (loaded)
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	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
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">
    Change Content
</button>
</div>

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

</body>
</html>
```

```
function loadDoc() {
    const xhttp = new XMLHttpRequest();
    xhttp.onload = function() {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
    xhttp.open("GET", "ajax.txt");
    xhttp.send();
  }
```

Hello everyone I am Binilraj Adhikari.
I love amalgamating business and technology.

XML

Web Technology - B.Sc. CSIT 5th Semester

Binilraj Adhikari

Definition: XML stands for Extensible Markup Language.

Purpose: It is designed for storing and transporting data.

Format: XML uses a simple and flexible text-based format.

Derivation: Derived from SGML (Standard Generalized Markup Language).

Structure: XML represents data in a hierarchical and tree-like structure.

Tags: Information in XML is enclosed in tags denoted by angle brackets (e.g., <tag>).

Hierarchy: XML documents have a hierarchical structure with elements nested within other elements.

Attributes: Elements can have attributes that provide additional information.

Human-Readable: XML is designed to be both human-readable and machine-readable.

Interoperability: XML is platform-independent, facilitating data exchange between different systems.

Validation: XML documents can be validated against a Document Type Definition (DTD) or an XML Schema Definition (XSD).

Use Cases: XML is used in various contexts, including web development (e.g., RSS feeds, SOAP for web services), configuration files, and data interchange.

Foundation: It serves as a foundation for other data interchange formats, such as JSON.

Advantages of XML

Human-Readable and Self-Descriptive:XML is a plain text format with clear, human-readable tags, making it easy for humans to read and understand.

Platform-Independent:XML is platform-independent, meaning it can be used on any operating system or device.

Hierarchical Structure: The hierarchical structure of XML allows for the representation of complex data relationships in a tree-like format.

Standardization:XML is a widely accepted and standardized format for data interchange, promoting interoperability between different systems.

Data Separation from Presentation:XML enables the separation of data from presentation, allowing for greater flexibility in how data is displayed.

Extensibility:XML is extensible, allowing users to define their own tags and structures to meet specific requirements.

Data Validation:XML documents can be validated against a Document Type Definition (DTD) or an XML Schema Definition (XSD) for consistency and correctness.

Versatility:XML is used in a variety of domains, including web development, configuration files, data exchange, and representation of structured data.

Disadvantages of XML

Verbose and Redundant:XML documents can be verbose and contain redundant information, leading to larger file sizes.

Parsing Overhead: Parsing XML documents can be computationally expensive, especially for large and complex documents.

Complexity:XML's flexibility can lead to complex document structures, making it more challenging to manage and process.

No Built-in Support for Non-Text Data:XML is primarily designed for text-based data, and handling binary data can be less straightforward.

Less Efficient for Some Use Cases: For certain use cases, such as high-performance data storage or transmission, more efficient formats like binary formats or JSON might be preferred.

Security Concerns:XML can be vulnerable to certain security issues, such as external entity expansion attacks.

Limited Native Support in Some Technologies: While widely supported, XML might not be as natively supported in some modern technologies as alternatives like JSON.

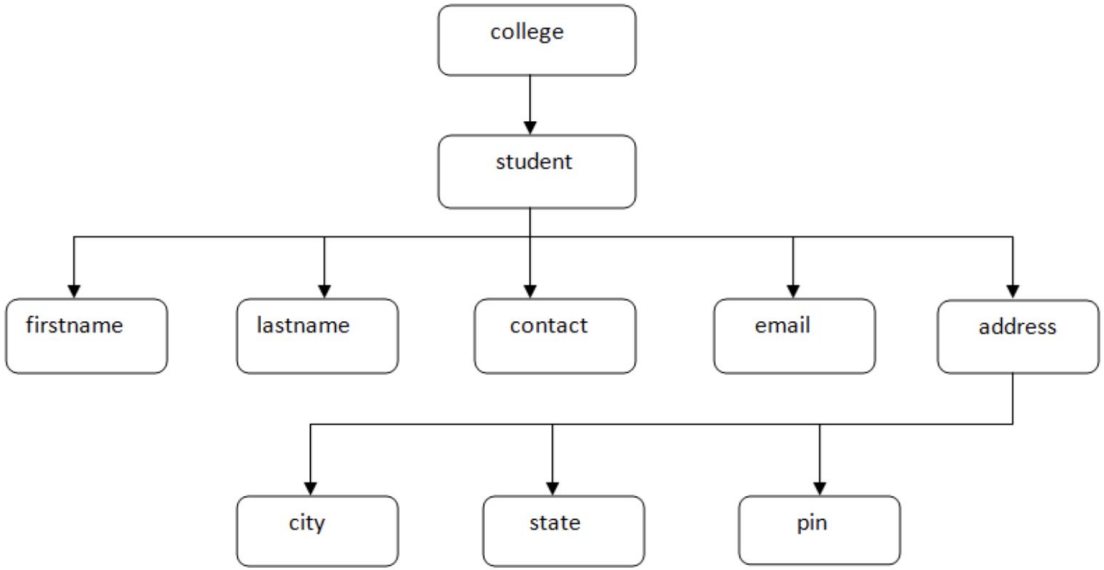
Readability vs. Efficiency Trade-Off: The human readability of XML comes at the cost of efficiency, as it can result in larger file sizes compared to more compact formats.

Syntax of XML

XML documents must contain one root element that is the parent of all other elements:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

XML Tree




```
<?xml version="1.0"?>
<college>
 <student>
    <firstname>Binilraj</firstname>
    <lastname>Adhikari</lastname>
    <contact>9851335428</contact>
 <email>adhikaribinil@gmail.com</email>
    <address>
       <city>Biratnagar</city>
       <state>Koshi</state>
       <pin>56700</pin>
    </address>
 </student>
</college>
```

XML Elements and Attributes

Elements are the basic building blocks of an XML document.

They define the structure and content of the data.

An XML element is typically defined by an opening tag, content, and a closing tag.

```
<element_name>content</element_name>
```

Elements can be nested within other elements, creating a hierarchical structure in the XML document.

Elements can have attributes, providing additional information about the element.

```
<book category="fiction">...</book>
```

An **attribute** is defined within the opening tag and consists of a name and a value.

```
<element_name attribute_name="attribute_value">content</element_name>
```

Properties of Attribute

Purpose: Attributes are used to convey metadata or additional properties associated with an element.

Uniqueness: Attribute names must be unique within the scope of the containing element.

Values: Attribute values are typically enclosed in double or single quotes.

Multiple Attributes: An element can have multiple attributes.

Rules for writing XML

Well-Formed XML:An XML document must be well-formed, meaning it adheres to the basic syntax rules of XML.

Every opening tag must have a corresponding closing tag.

Tags are case-sensitive, so <book> and <Book> are considered different.

2. Root Element:An XML document must have a single root element that contains all other elements.

All other elements must be nested within the root element.

3. Nested Elements:Elements must be properly nested within each other, and they should not overlap.

4. Attributes:Attribute values must be enclosed in double or single quotes.

Attribute names must follow the same naming conventions as elements.

5. Character Data:Character data (text content) should be placed between the opening and closing tags.

Special characters like <, >, &, must be replaced with entities (<, >, &).

6. Empty Elements:Empty elements should be self-closing with a slash before the closing angle bracket.

Example: <emptyElement />

7. Comments:Comments in XML are enclosed in <!-- and -->.

8. Entity References:Entity references can be used to represent special characters.

Example: € represents the Euro symbol.

9. CDATA Sections:CDATA sections can be used to include character data that should not be treated as XML.

Example: <![CDATA[This is some bold text.]]>

10. Namespace Usage:When using namespaces, elements and attributes should be prefixed accordingly.

Create an XML file containing records of employee having elements of simple and complex types.