AJAX

AJAX

- An Asynchronous JavaScript and XML
- Technique for creating fast and dynamic web pages
- It allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes
- This means that it is possible to update parts of a web page without reloading the whole page
- Classic web pages, (which do not use AJAX) must reload the entire page if the content should change
- Examples of applications using AJAX: Google Maps, Gmail, YouTube, and Facebook

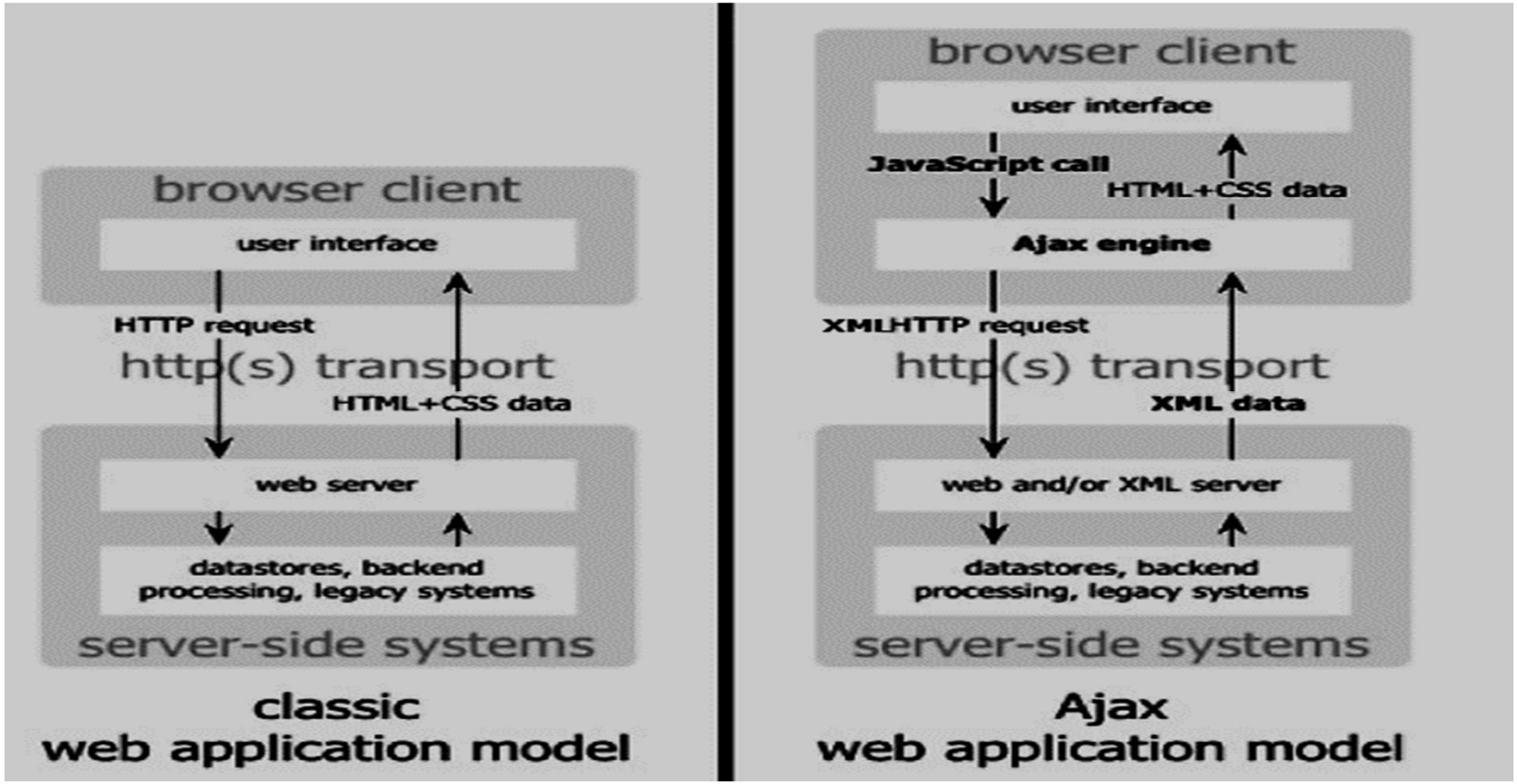
AJAX

- Sends and Receives data from Client and Server Asynchronously
- Different than Regular HTTP Requests
- Uses XmlHTTPRequest object
- Carries data in XML, Plain text and JSON Format
- Doesnot refresh the browser to receive data from server.

Technologies

- HTML and CSS for presentation
- DOM for dynamic display and interaction with data
- XML, JSON, XSLT for the interchange and manipulation of data
- XMLHttpRequest for asynchronous communication
- Javascript brings these technologies together

Difference


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
XMLHTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model

AJAX Engine

- An Ajax application places an intermediary between the user and the server called AJAX Engine.(also known as the Javascript part of a web page).
- Instead of loading a webpage, at the start of the session, the browser loads an AJAX engine – written in Javascript and usually tucked away in a hidden frame.
- This engine is responsible for rendering the interface the user sees and communicating with the server on the user's behalf.
- Engine allows asynchronous interaction .

How Ajax works

- Every user action generates an HTTP request that takes the form of a javascript call to the Ajax Engine.

- Any response to a user action that doesn't require the server, is handled by Engine itself.

  Eg – simple data validation, editing data in memory, some navigation.

- If the engine needs something from the server for a user request, the engine makes those requests asynchronously using XML without stalling a user's interaction with the application.

  E.g. Submitting data for processing, retrieving new data etc.

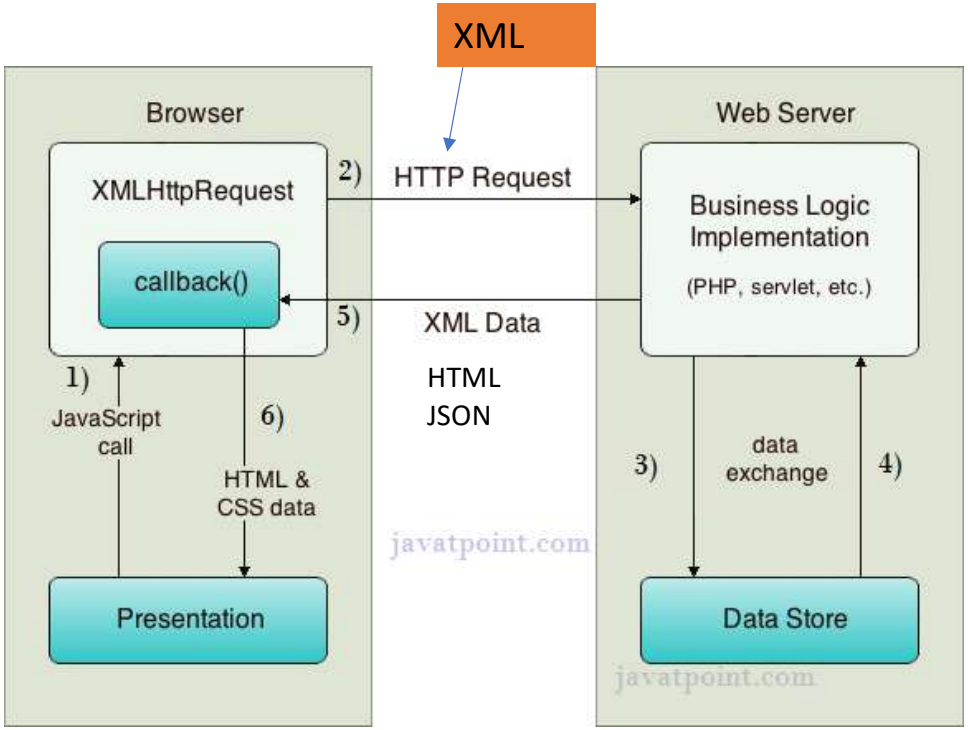
WAYS to perform AJAX

- **Regular AJAX Request – using javascript**
- Fetch API - Inbuilt in Browser
- Axios Library
- **Jquery Library**
- Node JS HTTP Module
- Angular JS HTTP Module
- React JS framework uses Axios Library

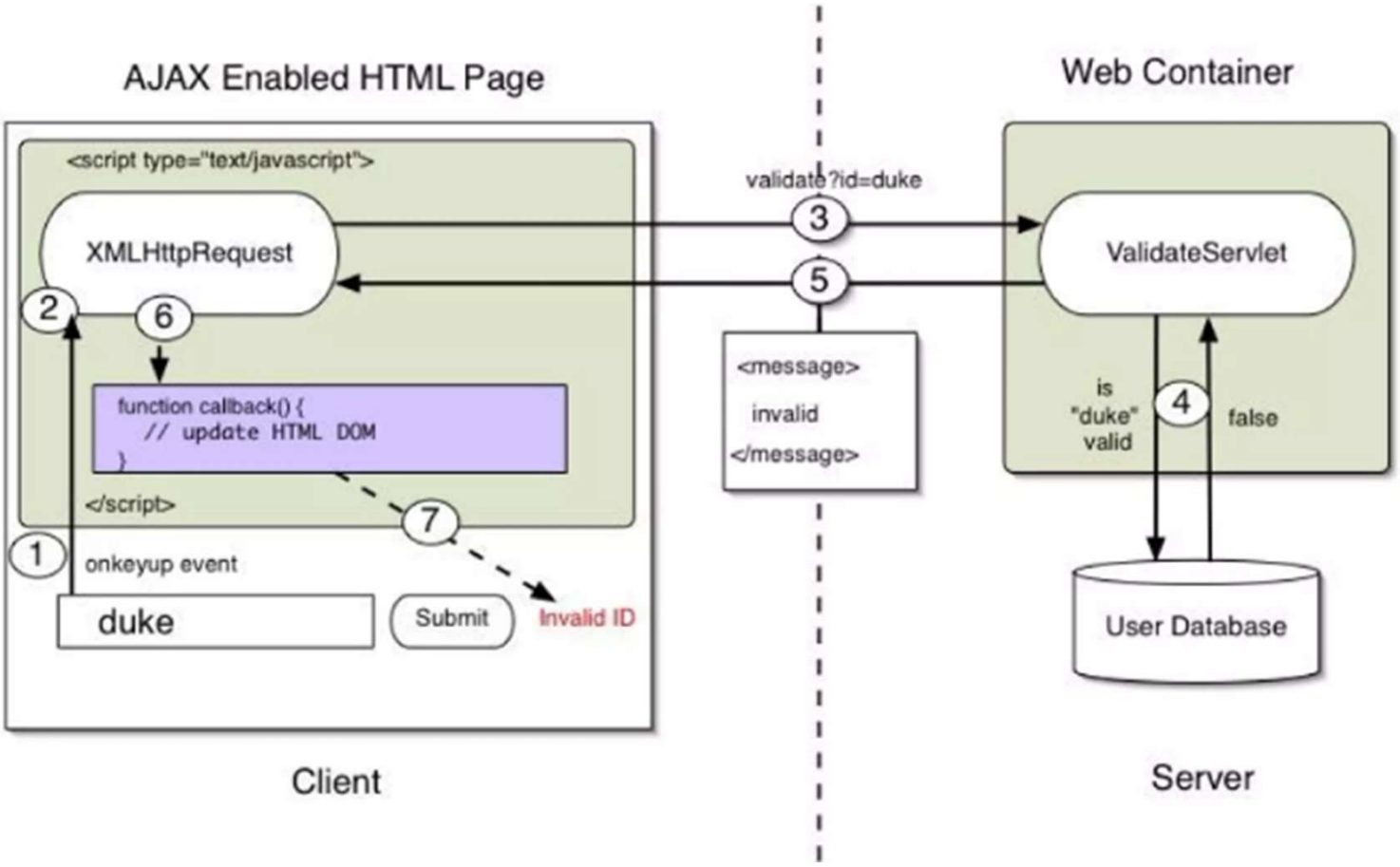
Steps of Ajax Operation

1. A client event occurs
2. An XMLHttpRequest object is created
3. The XMLHttpRequest object is configured
4. The XMLHttpRequest object makes an async. Request
5. The ValidateServlet returns an XML document containing the result
6. The XMLHttpRequest object calls the callback() function and processes the result
7. The HTML DOM is updated

How AJAX works



Steps of Ajax operation



Code for using Ajax... for lab session→

Starting from Browser …

- JavaScript has to handle events from the form, create an XMLHttpRequest object, and send it (via HTTP) to the server
- JavaScript has to create an XMLHttpRequest object
  - For most browsers, just do
    `var request = new XMLHttpRequest();`
- Once you have an XMLHttpRequest object, you have to prepare it with the open method
- ***request***.open(***method***, ***URL***, ***asynchronous***)
  - The ***method*** is usually 'GET' or 'POST'
  - The ***URL*** is where you are sending the data
    - If using a 'GET', data is appended to the URL
    - If using a 'POST', data is added in a later step
  - If ***asynchronous*** is true, the browser does not wait for a response (this is what you usually want)
- ***request***.open(***method***, ***URL***)
  - As above, with ***asynchronous*** defaulting to true

Sending the XMLHttpRequest object

- Once the XMLHttpRequest object has been prepared, you have to send it
- ***request***.send(null);
  - This is the version you use with a GET request
- ***request***.send(***content***);
  - This is the version you use with a POST request
  - The content has the same syntax as the suffix to a GET request
  - POST requests are used less frequently than GET requests
  - Example:
    request.send('var1=' + ***value1*** + '&var2=' + ***value2***);

Some more methods of XMLHttpRequest object

- **abort()**
  - Terminates current request
- **getAllResponseHeaders()**
  - Returns headers (labels + values) as a string
- **getResponseHeader("header")**
  - Returns value of a given header
- **setRequestHeader("label","value")**
  - Sets Request Headers before sending

On the server side

- The server gets a completely standard HTTP request
- In a servlet, this would go to a doGet or doPost method
- The response is a completely standard HTTP response
- Instead of returning a complete HTML page as a response, the server returns an arbitrary text string (possibly XML, possibly something else)

Why use Ajax

- Client/Server Apps:
  - Dynamic data
  - Static forms, controls, code, etc
  - Efficient, but not flexible
- Traditional Web Apps:
  - Dynamic data
  - Dynamic forms, controls, code, etc
  - Flexible, but inefficient, and noticeably slow
- Ajax Apps:
  - Dynamic data
  - Static or dynamic forms, controls, code, etc
  - Best of both worlds

Why use AJAX

- Multithreaded data retrieval from Web servers
    - Pre-fetch data before needed
    - Progress indicators
    - Appearance of speed
    - Avoids need for setTimeout()
- Less bandwidth required; less server load
    - Reload partial page, not entire page
    - Load data only, not even partial page

Eg

- A web page uses AJAX to request data from a server.
- While waiting for the server's response, the user can continue interacting with the web page, by scrolling or clicking buttons.
- When the server responds, the data is retrieved and processed asynchronously. The web page can update its content without requiring a full page reload.

- Like buttons in youtube, facebook

EG

```
<html>
<head>      <title>        My AJAX page        </title>        </head>
<body>
<script>
function sample() {
  var x = new XMLHttpRequest();
  x.onreadystatechange = function() {
  document.getElementById("demo").innerHTML = this.responseText;
  };
  x.open("GET", "text.txt", true);
  x.send();
} </script>
<h1 id="demo">ABOUT AJAX</h1>
<button type="button" onclick="sample()">Change Content</button>
<p>AJAX = Asynchronous JavaScript and XML. AJAX is a technique for creating fast and dynamic web pages. AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.</p></body></html>
```

Summary

- AJAX provides functionality to create a robust web application.
- If an Ajax web application is coded properly it will run faster than and as secure as a non-Ajax program.
- Ajax also allows websites to reduce their overall bandwidth usage and server load by reducing the amount of full page loads

- https://www.geeksforgeeks.org/how-to-detect-ajax-request-to-normal-request-in-node-js/?ref=asr1

```
<head>
    <title>Page Title</title>
    <script>
    function sample() {
          var x = new XMLHttpRequest();
         x.onreadystatechange = function() {
          document.getElementById("demo").innerHTML = this.responsetext;
          };
          //x.open("POST", "text.txt", true);
           x.open("GET", "http://localhost:3000", true);
          x.send();

       }</script>
</head>
<body>
    <h1 style="color:green">AJAX CHECK</h1>
    <h3>
          How to detect AJAX request
          to normal request Node.js?
      </h3>
</body>
<script>
    const xhr = new XMLHttpRequest();
    xhr.open("GET", "http://localhost:3000", true);
    xhr.send();
</script>
<h1 id="demo">ABOUT AJAX</h1>
<button type="button" onclick="sample()">Change Content</button>
<p>AJAX = Asynchronous JavaScript and XML.
     AJAX is a technique for creating fast and dynamic web pages.
     AJAX allows web pages to be updated asynchronously by exchanging small
     amounts of data with the server behind the scenes.
     This means that it is possible to update parts of a web page,
       without reloading the whole page.  </p> </body></html>
```

Ajax.html

```
// app.js

const http = require('http');
const server = http.createServer((req, res) => {
    if (req.headers.host != req.headers.referer) {
        console.log("It is the AJAX Request!");
    }
});
server.listen(3000);
```