Rich Internet Application

Introduction

- Desktop is a very powerful s/w experience when talked about look & feel.
- To a user, it is easy to handle and work with any software that behaves in a manner similar to a users desktop.
- New ways are being adopted to develop interfaces that provide users with better powerful experiences.
- Applications developed to provide exciting rich interactivity features in software programs are known as Rich Internet Applications (RIA).

Introduction

- RIAs are developed as web applications behaving like desktop application in their look & feel.
- RIA can perform computation on both the client side and sever side.
- It is developed using various technologies that includes Java, Silverlight, JavaFX, Adobe Flash & Flex, AJAX, OpenLaszlo

Characteristics of RIA

1. Enhanced Accessibility
2. Direct Interaction
3. Improved features
4. Partial update of webpages.
5. Better feedback to users.
6. Collaborative Web Application Access
7. Consistency of look & feel
8. Offline use of the application
9. Impact on performance

Introduction to AJAX

Asynchronous **Ja**vaScript and **X**ML

Asynchronous JavaScript and XML

- AJAX is an acronym for Asynchronous JavaScript and XML.
- It is a group of inter-related technologies like JavaScript, DOM, XML, HTML/XHTML, CSS, XMLHttpRequest etc.
- AJAX allows to send and receive data asynchronously without reloading the web page.
- AJAX allows to send only important information to the server not the entire page.
- So only valuable data from the client side is routed to the server side.
- It makes application interactive and faster.
- There are too many web applications running on the web that are using ajax technology
  - gmail, facebook, twitter, google map, youtube etc.

Asynchronous JavaScript and XML

AJAX

- not a programming language.
- imply a development technique for creating interactive web applications.

JavaScript

- Client side scripting language.
- It is executed on the client side by the web browsers that support JavaScript.
- JavaScript code only works in browsers that have JavaScript enabled.

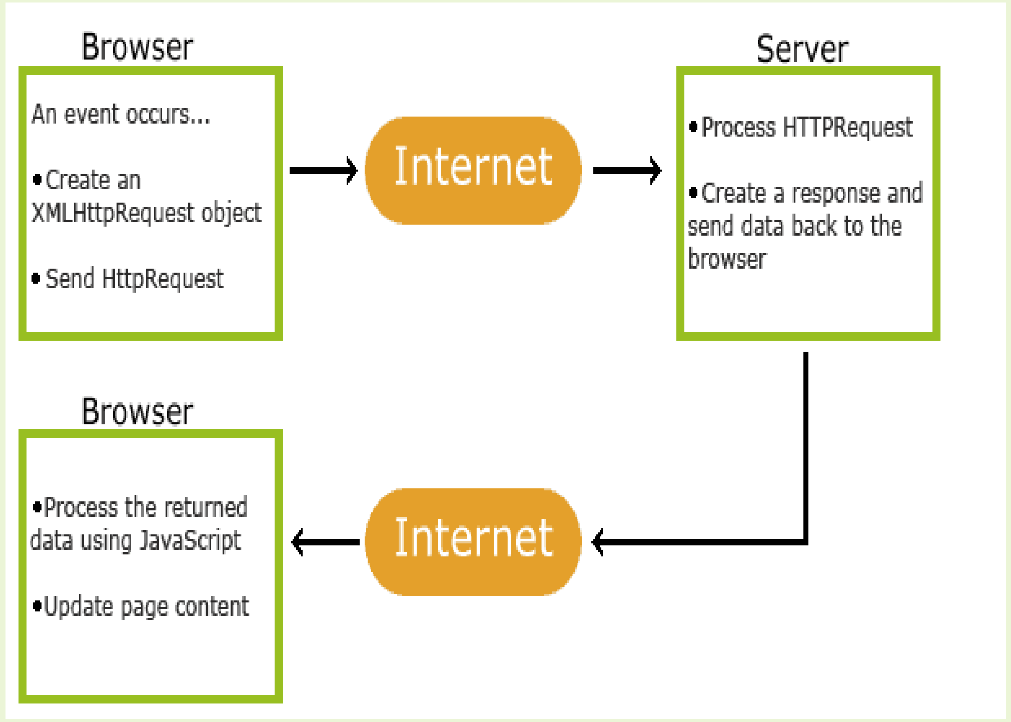
XML

- Acronym for Extensible Markup Language.
- It is used to encode messages in both human and machine readable formats.
- It’s like HTML but allows you to create your custom tags

Asynchronous JavaScript and XML

- AJAX uses an XMLHttpRequest object to send data to a web server
- XML is commonly used as the format for receiving server data, although any format including JSON, plain text can be used.

How it Works?


Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

- An event occurs in a web page (the page is loaded, a button is clicked)
- An XMLHttpRequest object is created by JavaScript
- The XMLHttpRequest object sends a request to a web server
- The server processes the request
- The server sends a response back to the web page
- The response is read by JavaScript
- Proper action (like page update) is performed by JavaScript

XMLHttpRequest Object

- The keystone of AJAX is the XMLHttpRequest object.
- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.

XMLHttpRequest Object

- An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

- Sends data from the client in the background
- Receives the data from the server
- Updates the webpage without reloading it.

XMLHttpRequest Object

Create an XMLHttpRequest Object

- All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

- Syntax for creating an XMLHttpRequest object:

```
        variable = new XMLHttpRequest();
```

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

Methods of XMLHttpRequest Object

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method, url, async, user, psw)	Specifies the request method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password

Methods of XMLHttpRequest Object

Method	Description
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

Properties of XMLHttpRequest Object

Property	Description
onload	Defines a function to be called when the request is recived (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

Properties of XMLHttpRequest Object

Property	Description
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")

The onload property Example

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

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

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>

</body>
</html>
```

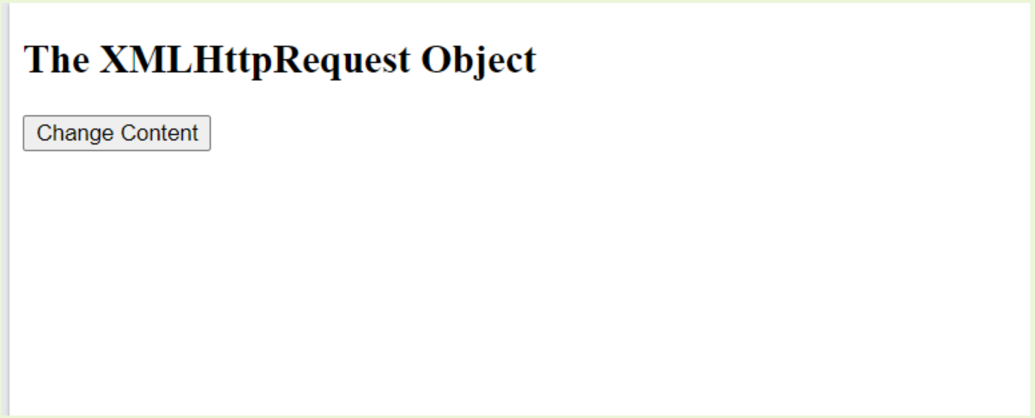
- The XMLHttpRequest object is used to exchange data with a server.
- To send a request to a server, use the open() and send() methods of the XMLHttpRequest object:

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

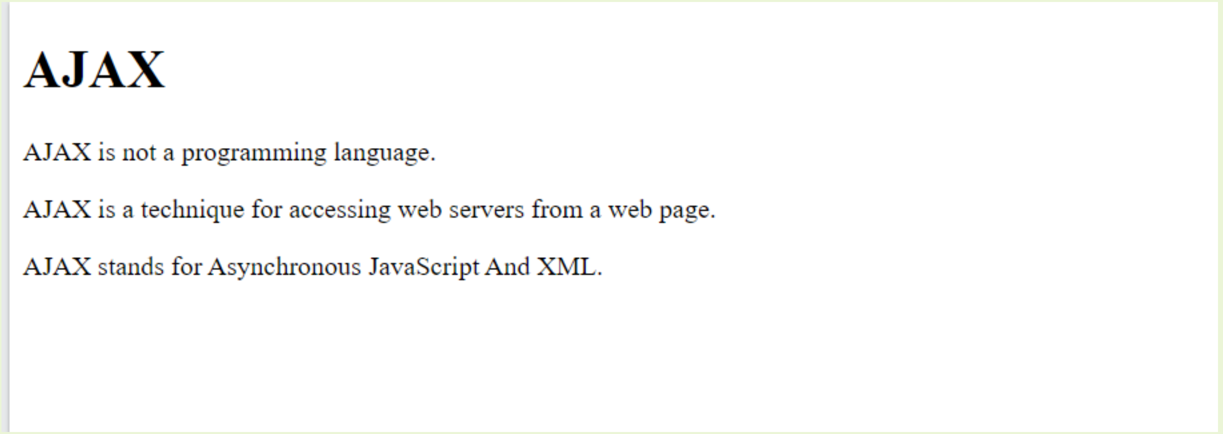
- For security reasons, modern browsers do not allow access across domains.
- This means that both the web page and the XML file it tries to load, must be located on the same server.

The onload property Example

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

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

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>

</body>
</html>
```



The XMLHttpRequest Object



AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

GET or POST method?

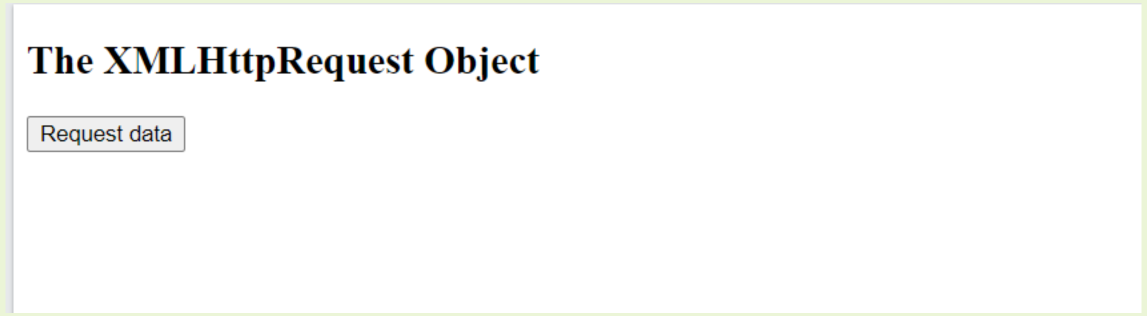
GET is simpler and faster than POST, and can be used in most cases.

Use POST requests when:

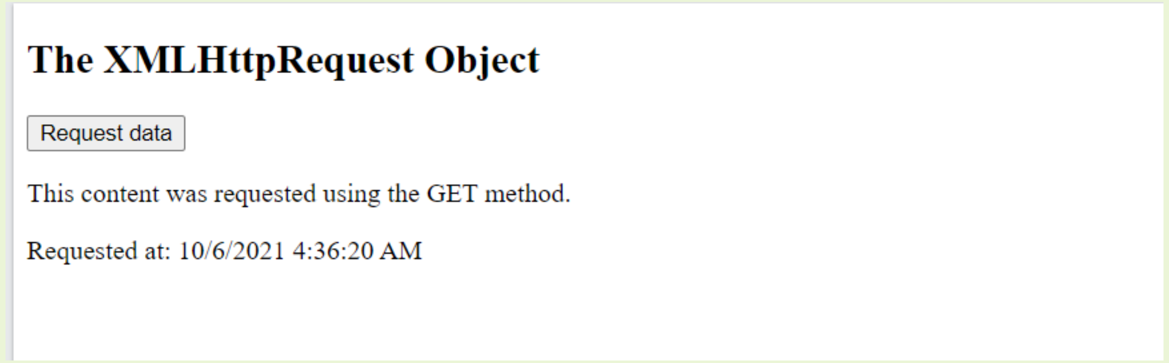
- A cached file is not an option (update a file or database on the server).
- Sending a large amount of data to the server (POST has no size limitations).
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

GET method Example

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

<h2>The XMLHttpRequest  Object</h2>
<button  type="button"  onclick="loadDoc()">Request data</button>
<p id="demo"></p>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML  = this.responseText;
  }
  xhttp.open("GET", "demo_get.asp");
  xhttp.send();
}
</script>

</body>
</html>
```



The XMLHttpRequest Object



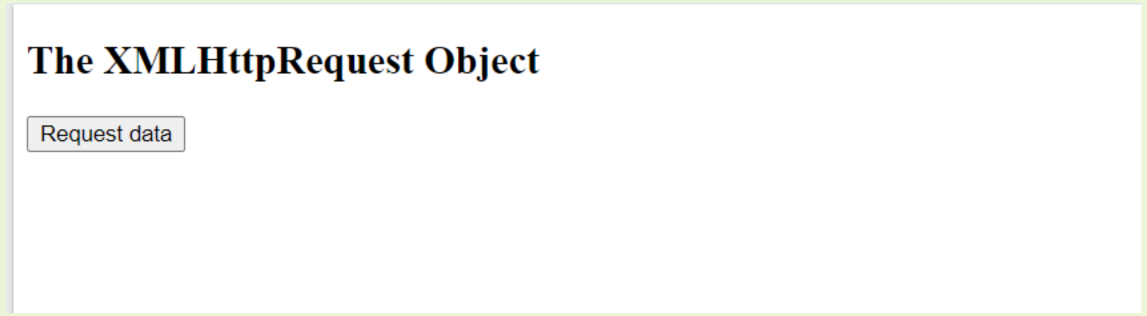
The XMLHttpRequest Object

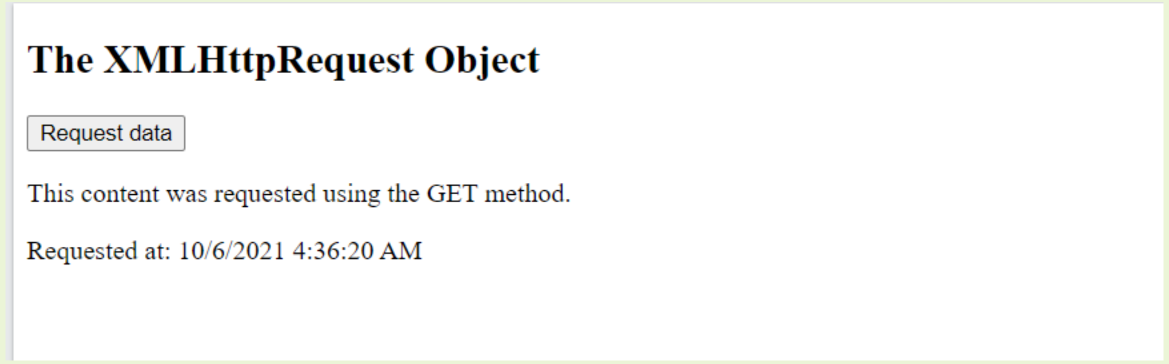
This content was requested using the GET method.

Requested at: 10/6/2021 4:36:20 AM

GET method Example (With math.random)

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

<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("GET", "demo_get.asp?t=" + Math.random());
  xhttp.send();
}
</script>

</body>
</html>
```


The XMLHttpRequest Object

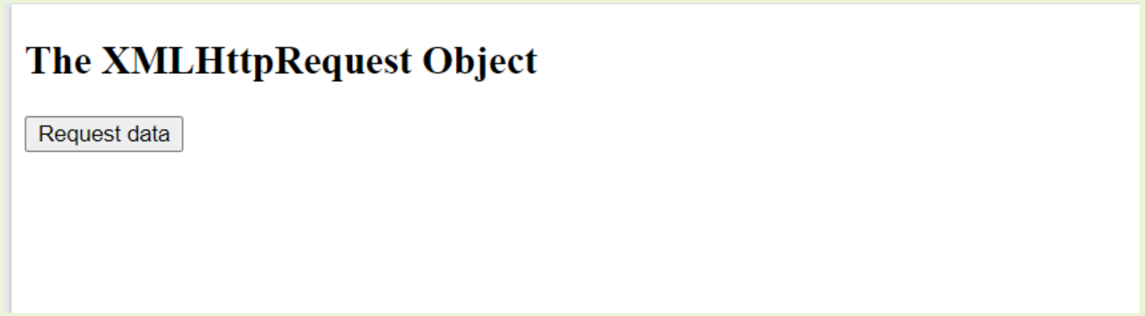

The XMLHttpRequest Object

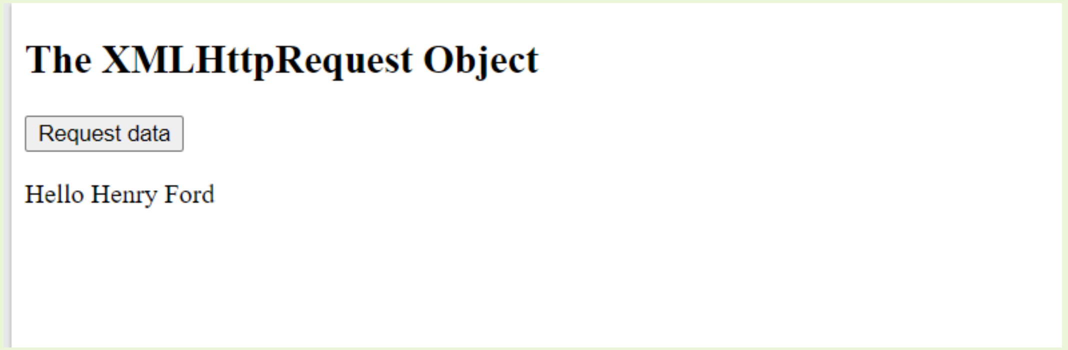
This content was requested using the GET method.

Requested at: 10/6/2021 4:36:20 AM

GET method Example (Adding information to URL)

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

<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford");
  xhttp.send();
}
</script>

</body>
</html>
```

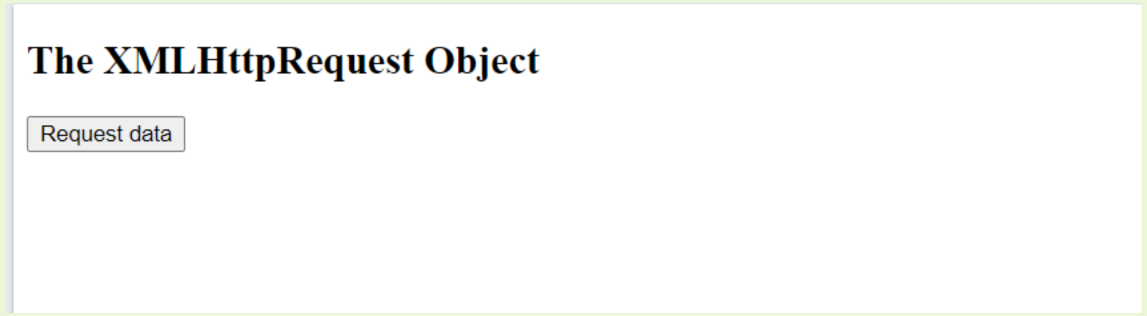

The XMLHttpRequest Object


The XMLHttpRequest Object

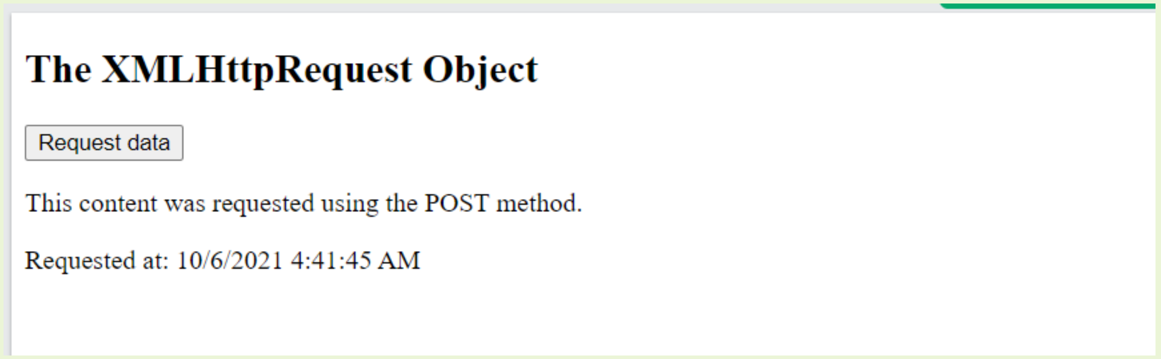
Hello Henry Ford

POST method Example

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

<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("POST", "demo_post.asp");
  xhttp.send();
}
</script>

</body>
</html>
```



The XMLHttpRequest Object



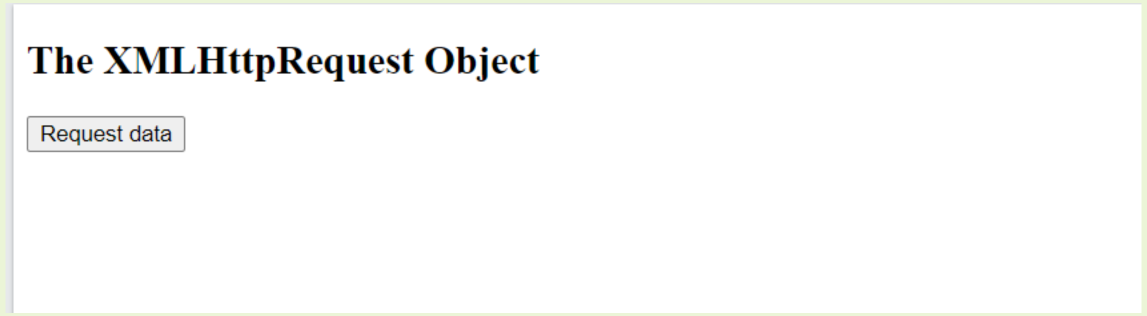
The XMLHttpRequest Object

This content was requested using the POST method.

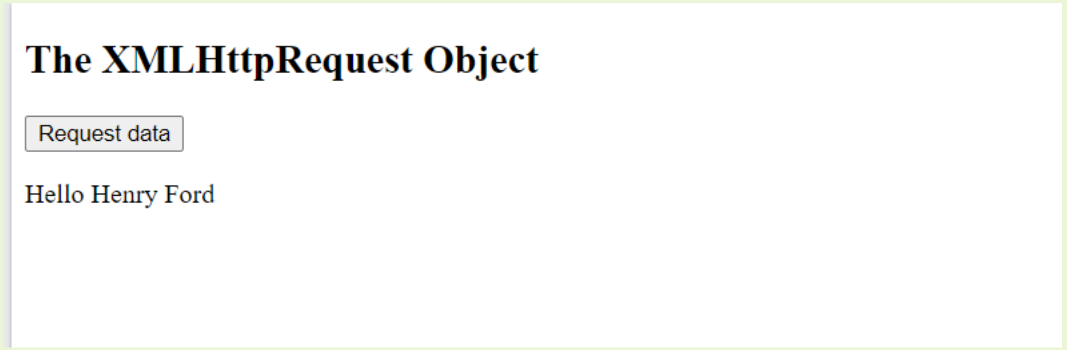
Requested at: 10/6/2021 4:41:45 AM

POST method Example (POST data like HTML form)

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

<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Request data</button>

<p id="demo"></p>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML = this.responseText;
  }
  xhttp.open("POST", "demo_post2.asp");
  xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xhttp.send("fname=Henry&lname=Ford");
}
</script>

</body>
</html>
```



The XMLHttpRequest Object

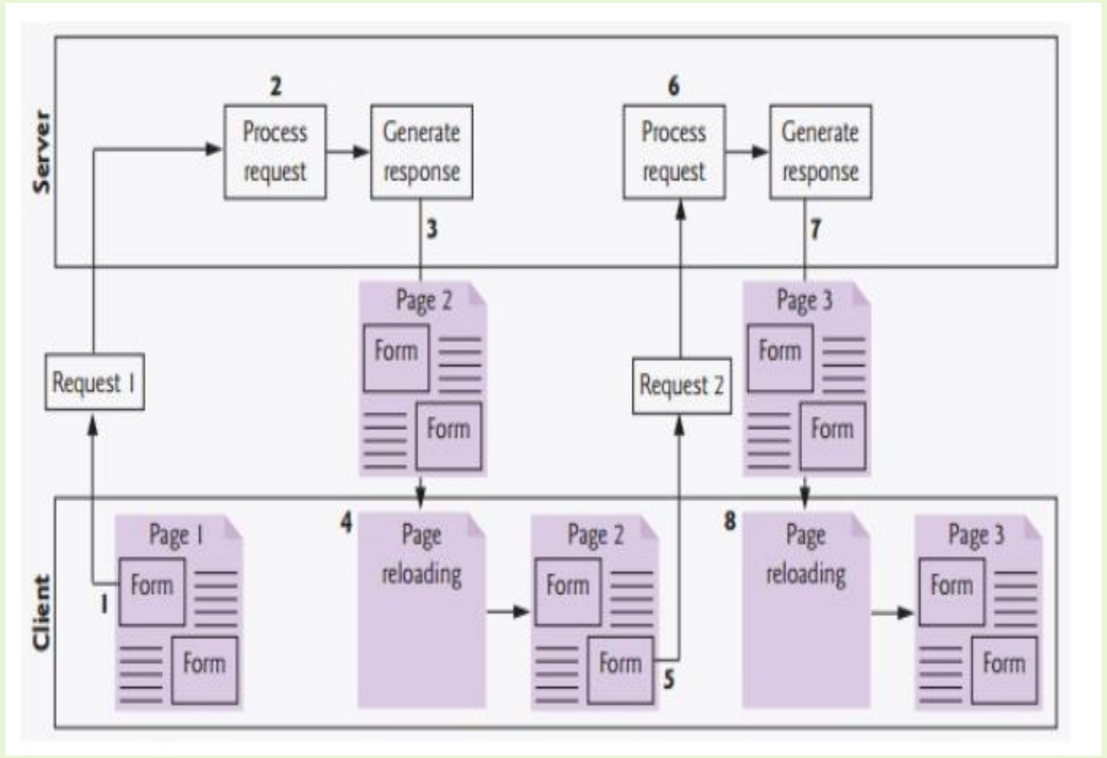


The XMLHttpRequest Object

Hello Henry Ford

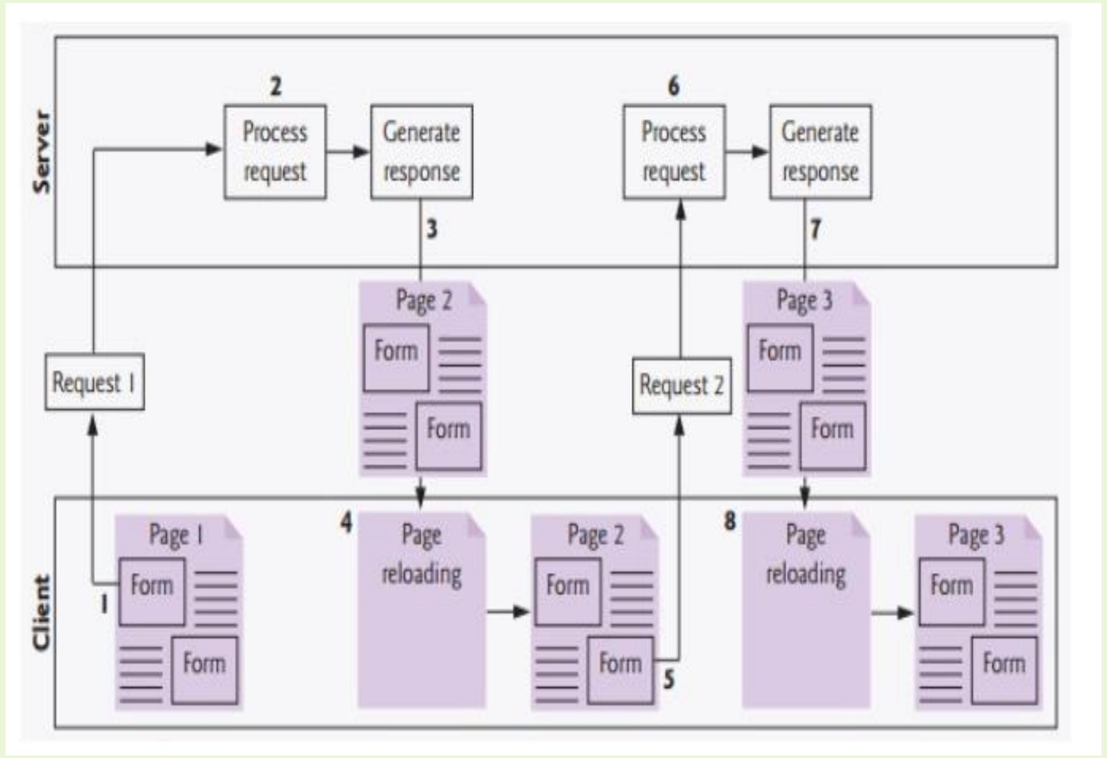
AJAX vs Traditional Approach

Traditional Approach



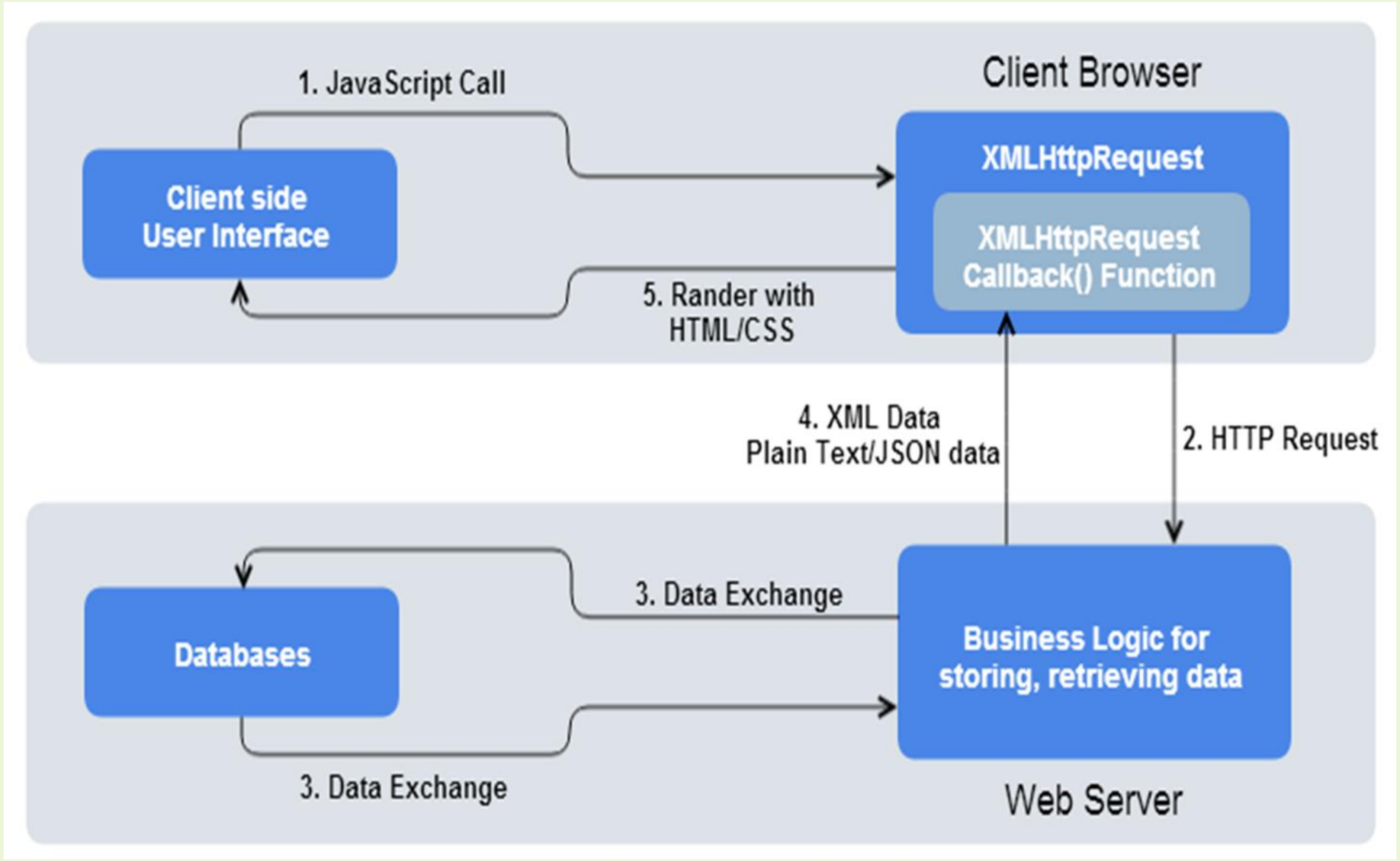
1. the user fills in the form’s fields, then submits the form (Fig. Step 1).
2. The browser generates a request to the server, which receives the request and processes it (Step 2).
3. The server generates and sends a response containing the exact page that the browser will render (Step 3), which causes the browser to load the new page (Step 4) and temporarily makes the browser window blank
4. Note that the client waits for the server to respond and reloads the entire page with the data from the response (Step4).
5. While such a synchronous request is being processed on the server, the user cannot interact with the client web page.

Traditional Approach

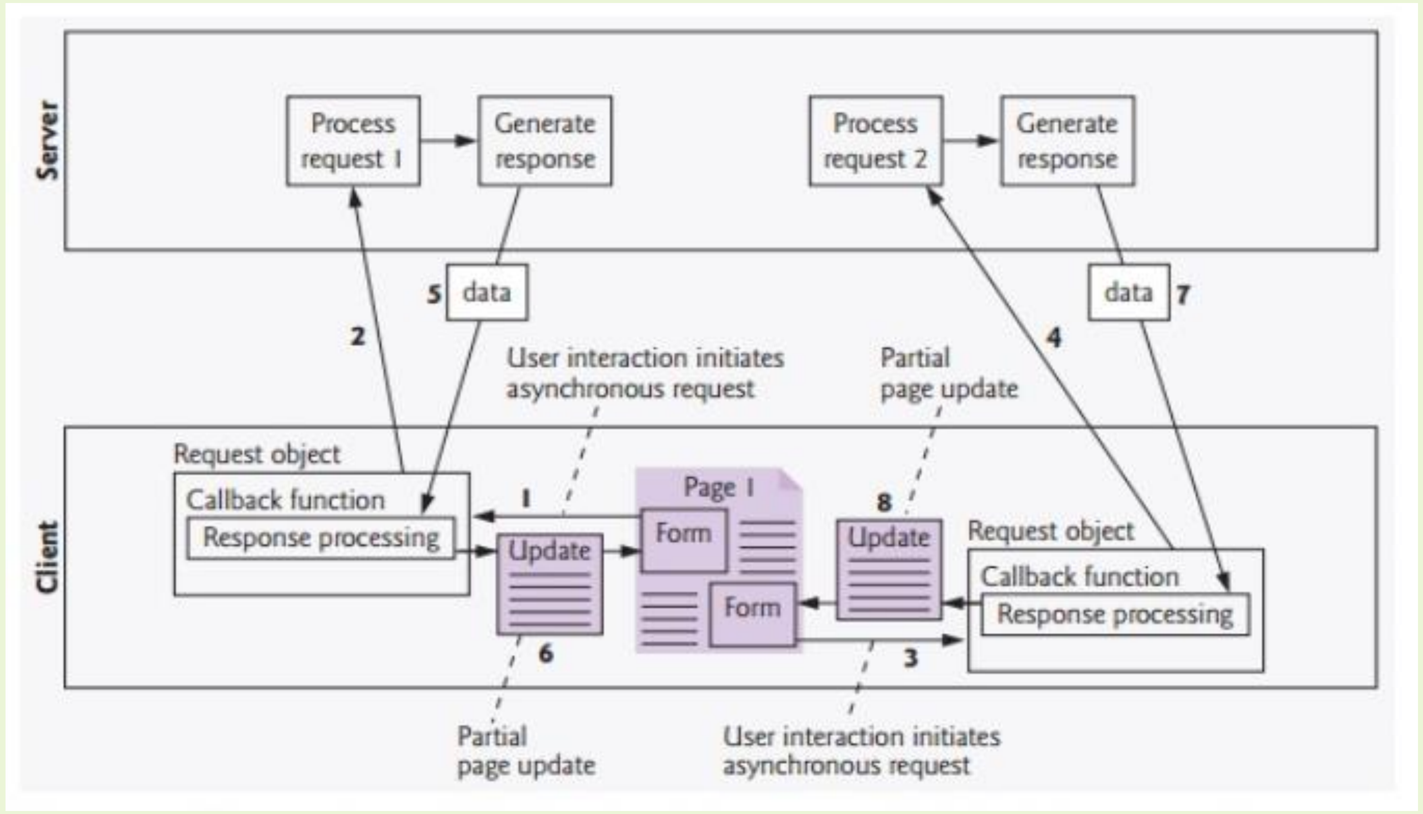


6. Frequent long periods of waiting, due perhaps to Internet congestion, have led some users to refer to the World Wide Web as the “World Wide Wait.”
7. If the user interacts with and submits another form, the process begins again (Steps 5–8).

AJAX Design Basics – Working of AJAX

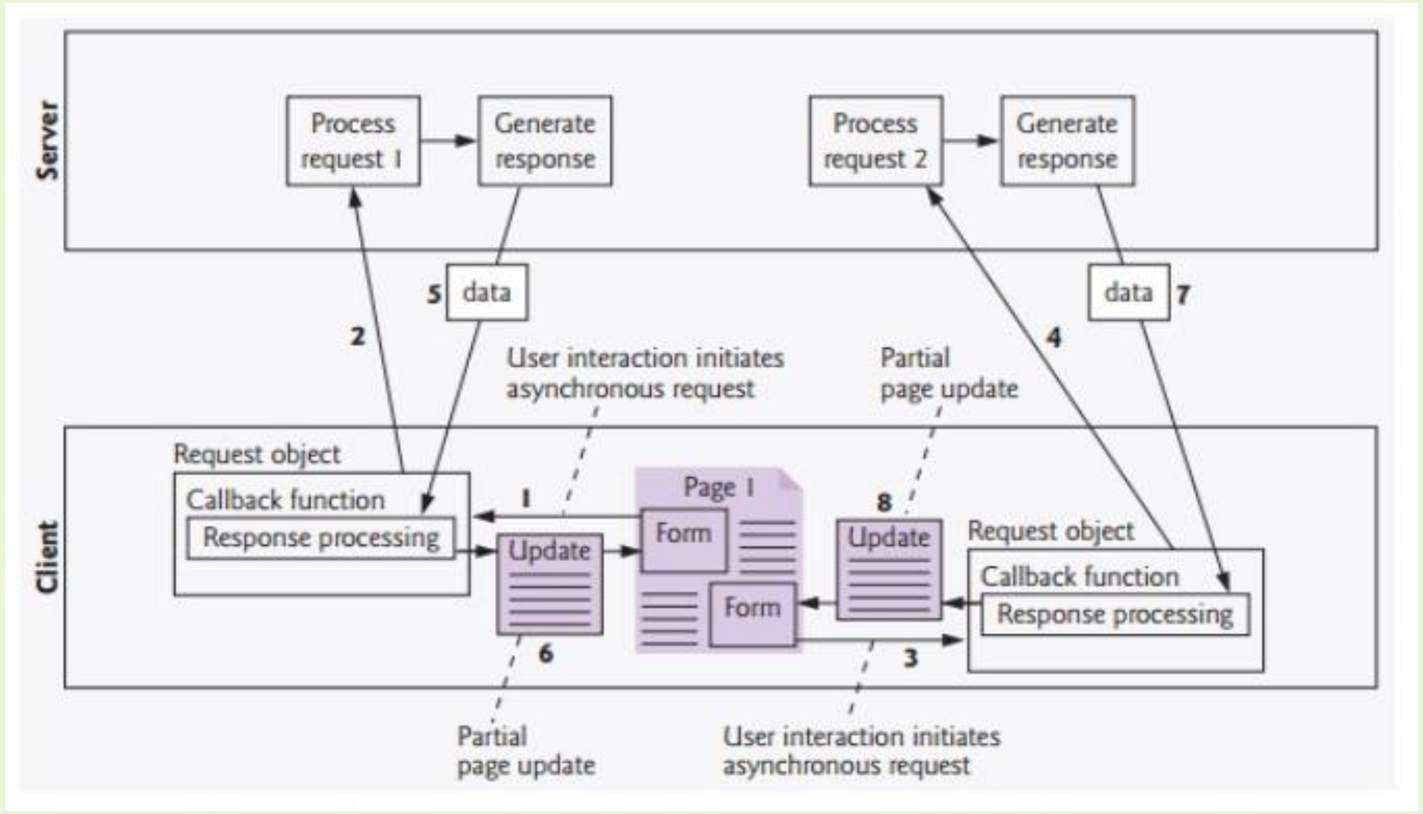

Client Browser
1. JavaScript Call
Client side User Interface
XMLHttpRequest
XMLHttpRequest Callback() Function
5. Rander with HTML/CSS
4. XML Data Plain Text/JSON data
2. HTTP Request
3. Data Exchange
Databases
Business Logic for storing, retrieving data
3. Data Exchange
Web Server

AJAX Web Application Model



1. Ajax applications add a layer between the client and the server to manage communication between the two
2. When the user interacts with the page, the client creates an XMLHttpRequest object to manage a request (Step 1).
3. The XMLHttpRequest object sends the request to the server (Step 2) and awaits the response.
4. The requests are asynchronous, so the user can continue interacting with the application on the client-side while the server processes the earlier request concurrently.
5. Other user interactions could result in additional requests to the server (Steps 3 and 4).
6. Once the server responds to the original request (Step 5), the XMLHttpRequest object that issued the request calls a client side function to process the data returned by the server.

AJAX Web Application Model



6. This function known as a callback function uses partial page updates (Step 6) to display the data in the existing web page without re-loading the entire page.
7. At the same time, the server may be responding to the second request (Step 7) and the client-side may be starting to do another partial page update (Step 8).
8. The callback function updates only a designated part of the page. Such partial page updates help make web applications more responsive, making them feel more like desktop applications.
9. The web application does not load a new page while the user interacts with it.

Key Points

- AJAX or Ajax is Asynchronous JavaScript and XML.
- It is used for exchanging data with a server and updating a portion of a webpage without reloading the whole webpage on the client side.
- The display and behavior of the existing webpage doesn’t get interfered at all while exchanging and updating the data.
- Ajax is also considered a group of technologies which has HTML, CSS, DOM, and JavaScript that are utilized to mark up, style, and allow the user to interact with the information on the webpage.
- Some basic knowledge of HTML, DOM, JavaScript, and a local Web server or remote Web Server is required.

JQuery

JQuery

- jQuery is open source, browser-independent, platform-independent.
- The most important reason behind developing jQuery is to simplify client side scripting (i.e. coding JavaScript).
- Using JavaScript, it is frustrating in some of the situations like selecting or targeting HTML elements, applying effects to them, adding and removing events, navigating and manipulating DOM tree, developing AJAX applications etc
- jQuery makes things like

  selecting or targeting elements ,

  adding styles,

  adding effects,

  creating animations,

  event handling,

  navigating and manipulating DOM tree,

  developing AJAX applications etc.
- It is much much simpler than JavaScript.

JQuery Syntax

The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).

Basic syntax is: **$(selector).action()**

where

- $ sign to define/access jQuery
- (selector) to "query (or find)" HTML elements
- jQuery action() to be performed on the element(s)

Examples:

- $(this).hide() - hides the current element.
- $("p").hide() - hides all <p> elements.
- $(".test").hide() - hides all elements with class="test".
- $("#test").hide() - hides the element with id="test".

The Document Ready Event

Here all jQuery methods in examples, are inside a document ready event:

```
$(document).ready(function(){
  // jQuery methods go here...
});
```

- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- It is good practice to wait for the document to be fully loaded and ready before working with it.
- This also allows you to have your JavaScript code before the body of your document, in the head section.

The Document Ready Event

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

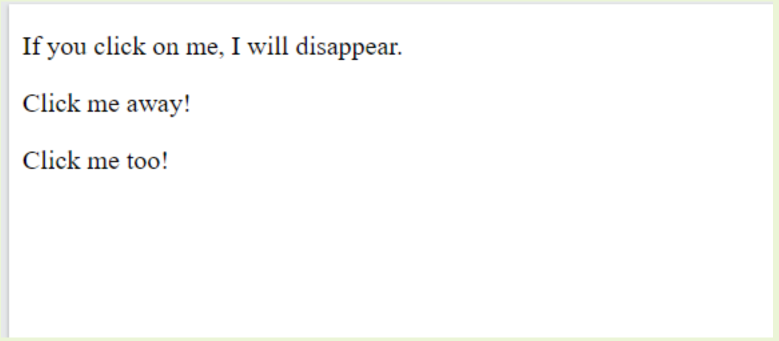
- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

The jQuery team has also created an even shorter method for the document ready event:

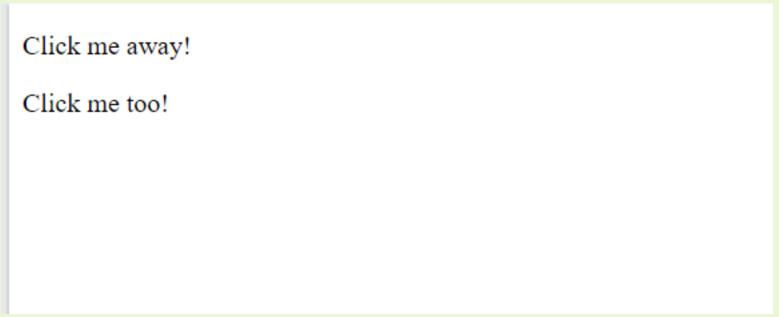
```
$(function(){
  // jQuery methods go here...
});
```

Jquery Example

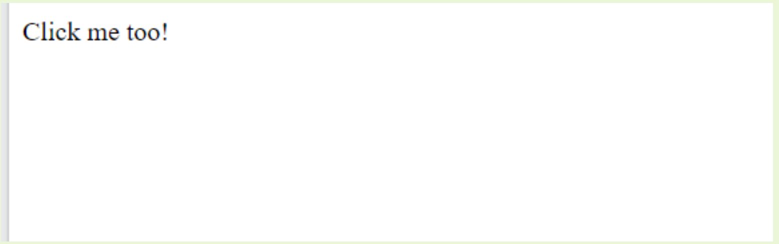
```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```



```
If you click on me, I will disappear.

Click me away!

Click me too!
```



```
Click me away!

Click me too!
```



```
Click me too!
```

JQuery and AJAX

- jQuery provides several methods for AJAX functionality.
- With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post –
- And you can load the external data directly into the selected HTML elements of your web page!
- Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation.
- This means that you will have to write extra code to test for different browsers.
- However, the jQuery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.

JQuery - AJAX load() Method

The jQuery load() method is a simple, but powerful AJAX method.

The load() method loads data from a server and puts the returned data into the selected element.

Syntax:

```
$(selector).load(URL, data, callback);
```

- The required URL parameter specifies the URL you wish to load.
- The optional data parameter specifies a set of querystring key/value pairs to send along with the request.
- The optional callback parameter is the name of a function to be executed after the load() method is completed.

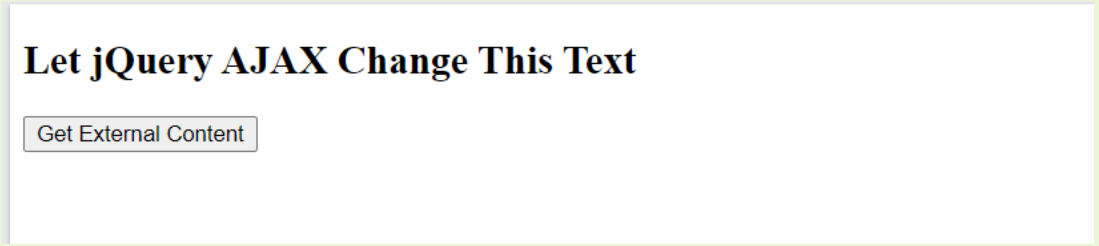
JQuery - AJAX load() Method

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
 $("button").click(function(){
  $("#div1").load("demo_test.txt");
 });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

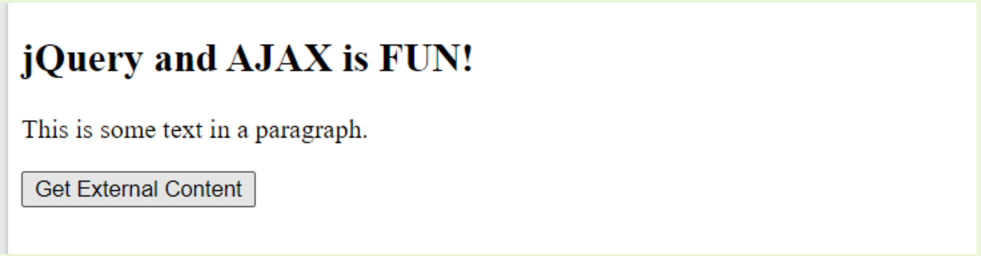
The following example loads the content of the file "demo_test.txt" into a specific <div> element

Here is the content of our example file: "demo_test.txt":

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```



Let jQuery AJAX Change This Text



jQuery and AJAX is FUN!

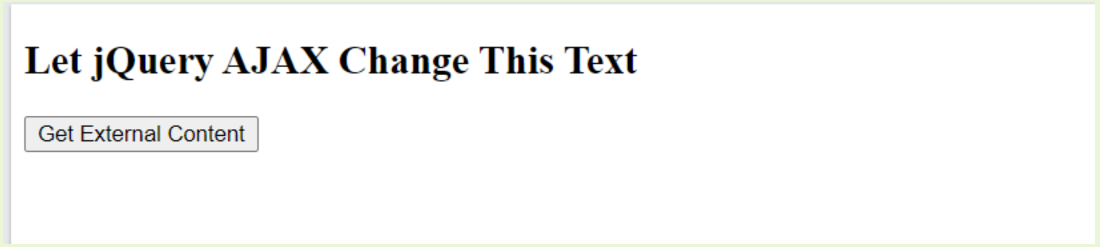
This is some text in a paragraph.

JQuery - AJAX load() Method

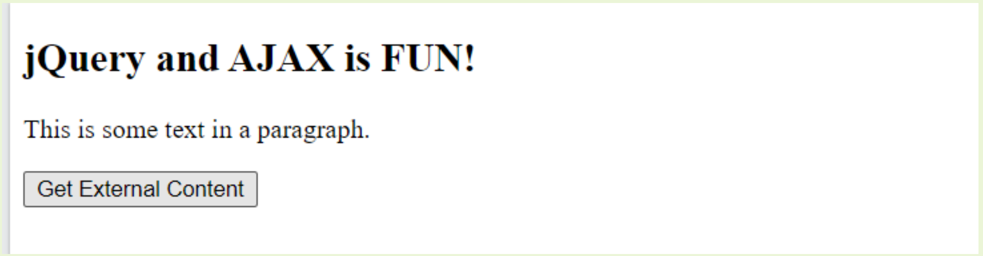
```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").load("demo_test.txt #p1");
  });
});
</script>
</head>
<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>
</html>
```

It is also possible to add a jQuery selector to the URL parameter.

The following example loads the content of the element with id="p1", inside the file "demo_test.txt", into a specific <div> element



Let jQuery AJAX Change This Text



jQuery and AJAX is FUN!

This is some text in a paragraph.

JQuery - AJAX get() Method

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("demo_test.asp", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>

</body>
</html>
```

jQuery $.get() Method

The $.get() method requests data from the server with an HTTP GET request.

Syntax:

$.get(URL,callback);

- The required URL parameter specifies the URL you wish to request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.
- The following example uses the $.get() method to retrieve data from a file on the server:

JQuery - AJAX get() Method

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("demo_test.asp", function(data, status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>

</body>
</html>
```

jQuery $.get() Method

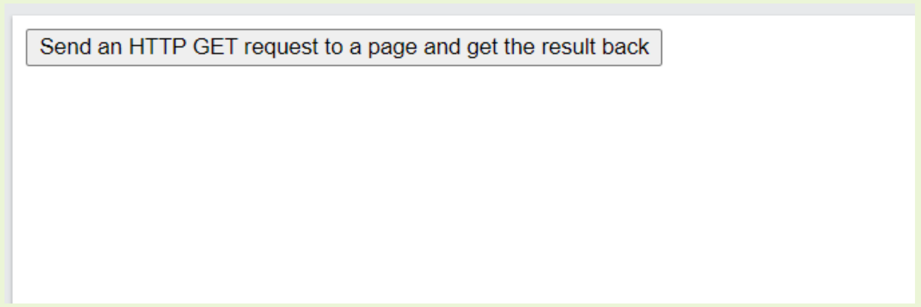
- The first parameter of $.get() is the URL we wish to request ("demo_test.asp").
- The second parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

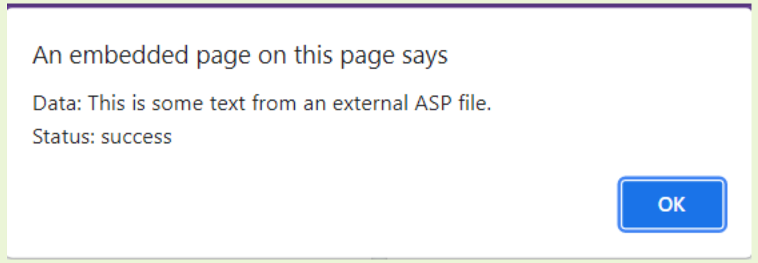
Here is how the ASP file looks like ("demo_test.asp"):

```
<%
response.write("This is some text from an external ASP file.")
%>
```

JQuery - AJAX get() Method

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
 $("button").click(function(){
  $.get("demo_test.asp", function(data, status){
   alert("Data: " + data + "\nStatus: " + status);
  });
 });
});
</script>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>

</body>
</html>
```




An embedded page on this page says

Data: This is some text from an external ASP file.
Status: success

JQuery – AJAX post() Method

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.post("demo_test_post.asp",
    {
      name: "Donald Duck",
      city: "Duckburg"
    },
    function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send  an HTTP POST request to a page and get the result back</button>

</body>
</html>
```

jQuery $.post() Method

The $.post() method requests data from the server using an HTTP POST request.

Syntax:

$.post(URL,data,callback);

- The required URL parameter specifies the URL you wish to request.
- The optional data parameter specifies some data to send along with the request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.

The following example uses the $.post() method to send some data along with the request

JQuery – AJAX post() Method

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.post("demo_test_post.asp",
    {
      name: "Donald Duck",
      city: "Duckburg"
    },
    function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send  an HTTP POST request to a page and get the result back</button>


</body>
</html>
```

jQuery $.post() Method

The first parameter of $.post() is the URL we wish to request ("demo_test_post.asp").

Then we pass in some data to send along with the request (name and city).

The ASP script in "demo_test_post.asp" reads the parameters, processes them, and returns a result.

The third parameter is a callback function.

The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

JQuery – AJAX post() Method

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.post("demo_test_post.asp",
    {
      name: "Donald Duck",
      city: "Duckburg"
    },
    function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
});
</script>
</head>
<body>

<button>Send  an HTTP POST request to a page and get the result back</button>

</body>
</html>
```

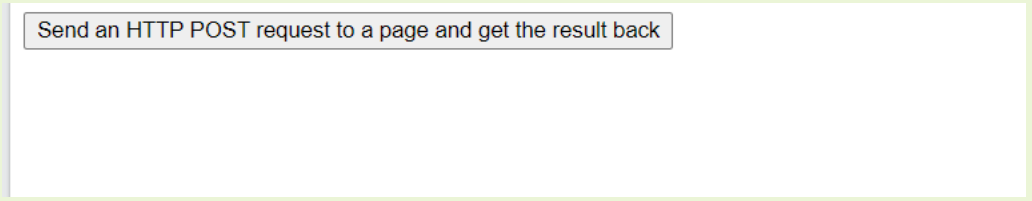
jQuery $.post() Method

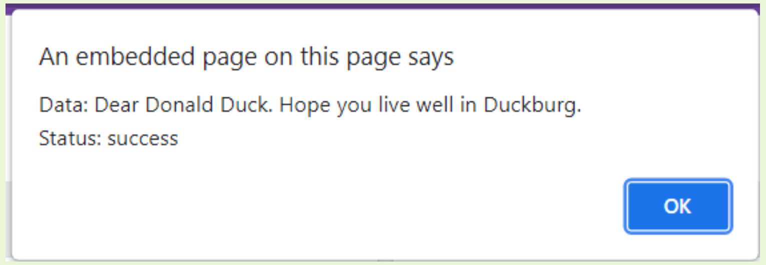
Here is how the ASP file looks like ("demo_test_post.asp"):

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city &
".")
%>
```

JQuery – AJAX post() Method

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
 $("button").click(function(){
  $.post("demo_test_post.asp",
  {
   name: "Donald Duck",
   city: "Duckburg"
  },
  function(data,status){
   alert("Data: " + data + "\nStatus: " + status);
  });
 });
});
</script>
</head>
<body>

<button>Send  an HTTP POST request to a page and get the result back</button>


</body>
</html>
```




An embedded page on this page says

Data: Dear Donald Duck. Hope you live well in Duckburg.
Status: success

Lesson 1: Content

- Add text here.
- To add a picture, chart, or other content in the right column, click the appropriate icon.
- To add a slide, click New Slide on the Insert menu, or press CTRL+M.

Lesson 1: Wrap-up

- Summarize important points.
- Allow time for questions.

Lesson 2: Objectives

- List the intended outcomes for this training session.
- Each objective should be concise, should contain a verb, and should have a measurable result.

Lesson 2: Content

- Add text here.
- To add a picture, chart, or other content in the right column, click the appropriate icon.
- To add a slide, click New Slide on the Insert menu, or press CTRL+M.

Lesson 2: Wrap-up

- Summarize important points.
- Allow time for questions.

Lesson 3: Objectives

- List the intended outcomes for this training session.
- Each objective should be concise, should contain a verb, and should have a measurable result.

Lesson 3: Content

- Add text here.
- To add a picture, chart, or other content in the right column, click the appropriate icon.
- To add a slide, click New Slide on the Insert menu, or press CTRL+M.

Lesson 3: Wrap-up

- Summarize important points.
- Allow time for questions.

Summary of Training

- List important points from each lesson.
- Provide resources for more information on subject.
  - List resources on this slide.
  - Provide handouts with additional resource material.

Assessment and Evaluation

- Prepare a quiz or challenge to assess how much information participants learned.
- Survey participants to see if they found the training beneficial.