

SWE 363: Web Engineering & Development

Module 7-4

Asynchronous JavaScript and XML



Objectives

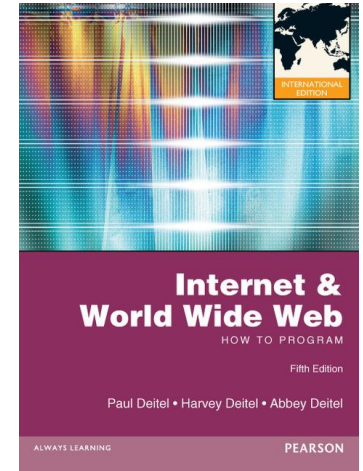
- ❑ To learn the role of AJAX technology
- ❑ To learn how to utilize AJAX's capability

- ❑ Rich-User Experience (UX)
- ❑ Issues of Conventional Web Application !
- ❑ What is AJAX?
- ❑ AJAX vs. Classic Web App.
- ❑ Technologies Used In AJAX
- ❑ XMLHttpRequest Object
- ❑ Ajax Typical Application Steps
- ❑ Using jQuery Ajax
 - load() method
 - .ajax() method

References

- ❑ “Internet & World Wide Web: How to Program 5th editions”

© Pearson Education



Some materials and examples of this part are taken from:

https://www.w3schools.com/Php/php_ajax_intro.asp

w3schools.com

If you want to learn more about AJAX, visit [AJAX tutorial](#).

Rich-User Experience (UX)

- ❑ UX focuses on having a deep understanding of users, what they need, what they value, their abilities, and also their limitations
- ❑ UX best practices promote improving the quality of the user's interaction and perceptions
- ❑ Consider a typical desktop application (e.g. spreadsheet app, etc.)
 - The program responds intuitively and quickly
 - The program gives a user meaningful feedback's instantly, e.g.
 - A cell in a spreadsheet changes color when you hover your mouse over it
 - Icons light up as mouse hovers them
- ❑ Things happen naturally

Quality Factors that Influence UX

- ❑ At the core of UX is ensuring that **users find value in what you are providing to them**. Peter Morville represents this through his [User Experience Honeycomb](#).
 - **Usable**: Site must be easy to use
 - **Useful**: Content should be original and fulfill a need
 - **Desirable**: Image, identity, brand, and other design elements are used to evoke emotion and appreciation
 - **Findable**: Content needs to be navigable and locatable onsite and offsite
 - **Accessible**: Content needs to be accessible to people with disabilities
 - **Credible**: Users must trust and believe what you tell them



User Experience Honeycomb

A key element of an effective UX is a **speedy response** to user input

With a standard website, users should wait for HTTP requests to the server resulting in a complete page load for every change (with every mouse click to request a new page, larger image, or more information, the DOM is completely thrown out and reloaded into the browser)

— Ineffective when loading pages on the same site which typically have similar content

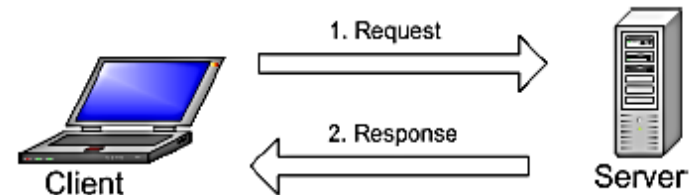
http://semanticstudios.com/user_experience_design/

<https://www.usability.gov/what-and-why/user-experience.html>

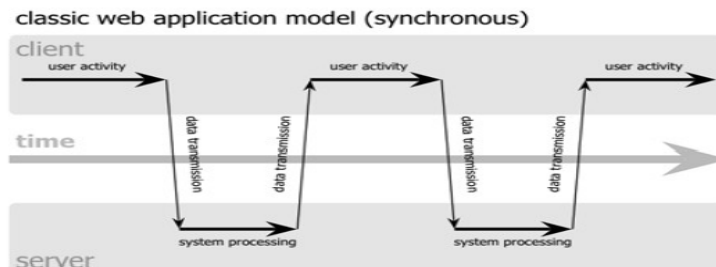
Characteristics of Conventional Web Apps

- ❑ “Click, wait, and refresh” user interaction
 - Page refreshes from the server needed for all events, data submissions, and navigation

- ❑ Page-driven: Workflow is based on pages



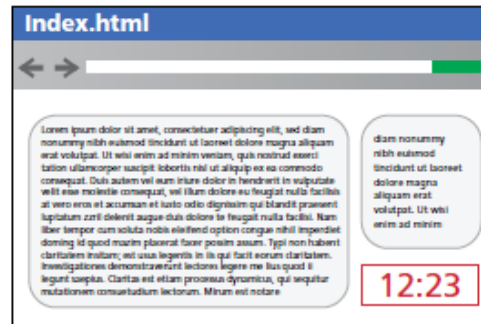
- ❑ Synchronous “request/response” communication model
 - The user has to wait for the response



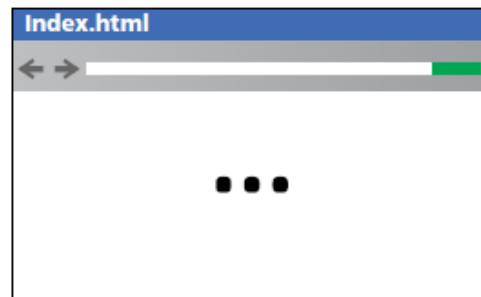
Interrupted user operation while the data is being fetched

Issues of Conventional Web Application

- ❑ **Interruption** of user operation
 - Users cannot perform any operation while waiting for a response
- ❑ No **instant feedback's** to user activities
 - A user has to wait for the next page
- ❑ Loss of **operational context** during refresh
 - Loss of information on the screen
 - Loss of scrolled position

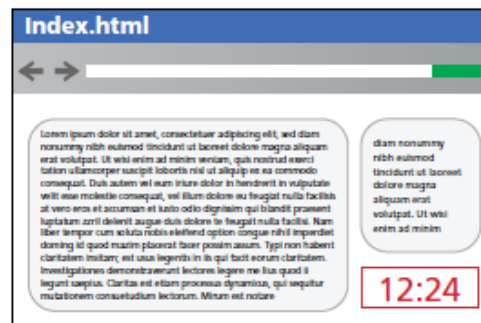


- 1 The page loads and shows the current server time as a small part of a larger page.



- 2 A **synchronous** JavaScript call makes an HTTP request for the “freshest” version of the page.

While waiting for the response, the browser goes into its waiting state.



- 3 The response arrives, so the browser can render the new version of the page, and the functionality in the browser is restored.

```
<html>
  <head>
  ...
</head>
<body>
  ...
  <div id='serverTime'>
    12.24
  </div>
  ...
</body>
</html>
```


What is AJAX?



Asynchronous JavaScript And XML

AJAX comes with the solution to **balance the load between the client and the server** by allowing them to **communicate in the background** while the user is working on the page.

AJAX is a means of using **JavaScript to communicate with a web server** **without** submitting a form or loading a new page.

With **AJAX**, a JavaScript can communicate directly with the server, with the **XMLHttpRequest** object. With this object, a JavaScript can trade data with a web server, **without reloading the page**.

What is AJAX?



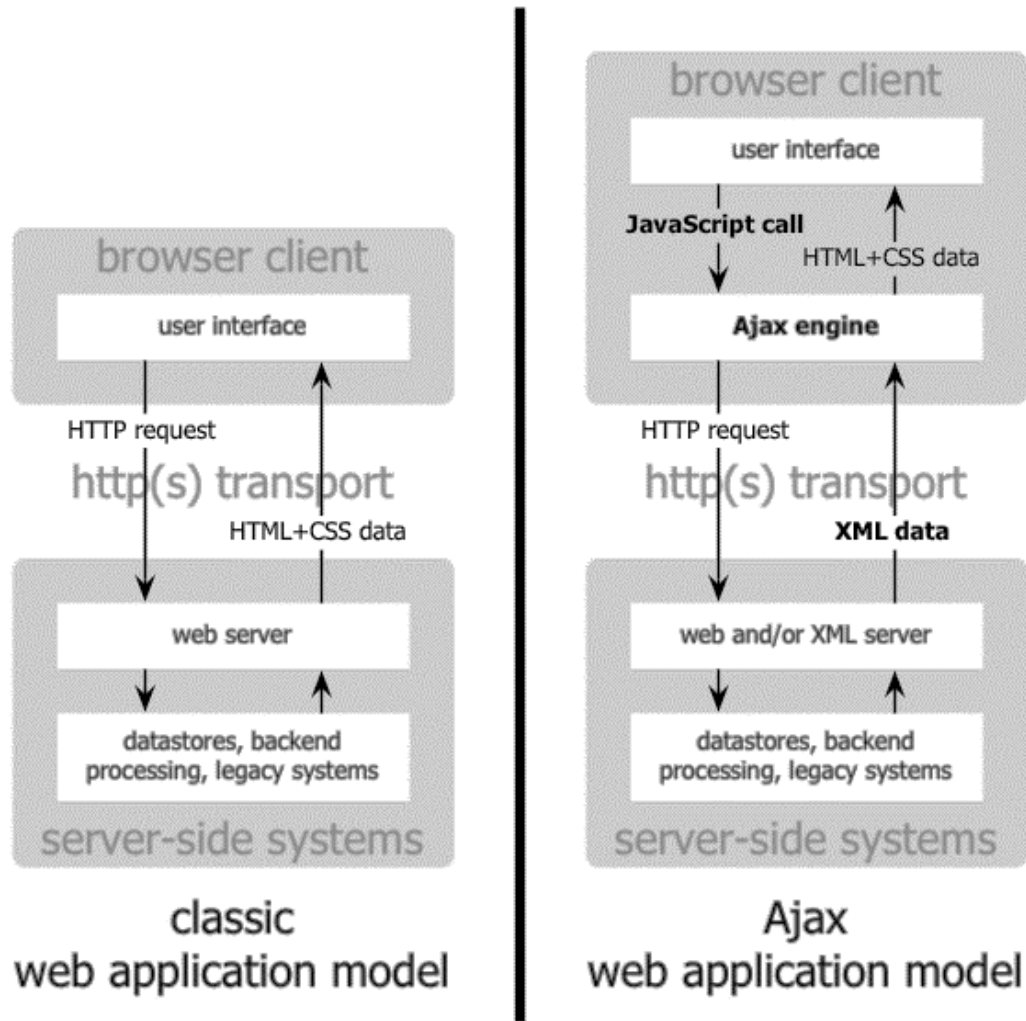
Asynchronous JavaScript And XML

AJAX is not a programming language, but simply a **development technique** for creating interactive web applications.

AJAX technique makes web pages more **responsive** by **exchanging data with a server behind the scenes**, instead of reloading an entire web page each time a user makes a change.

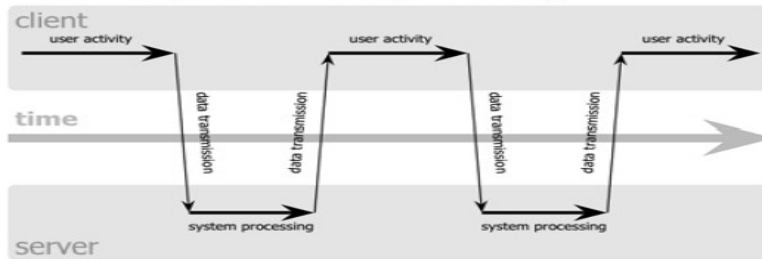
AJAX apps are **browser- and platform-independent**. With **AJAX** no need to download code & no plug-in required.

AJAX vs. Classic Web App.



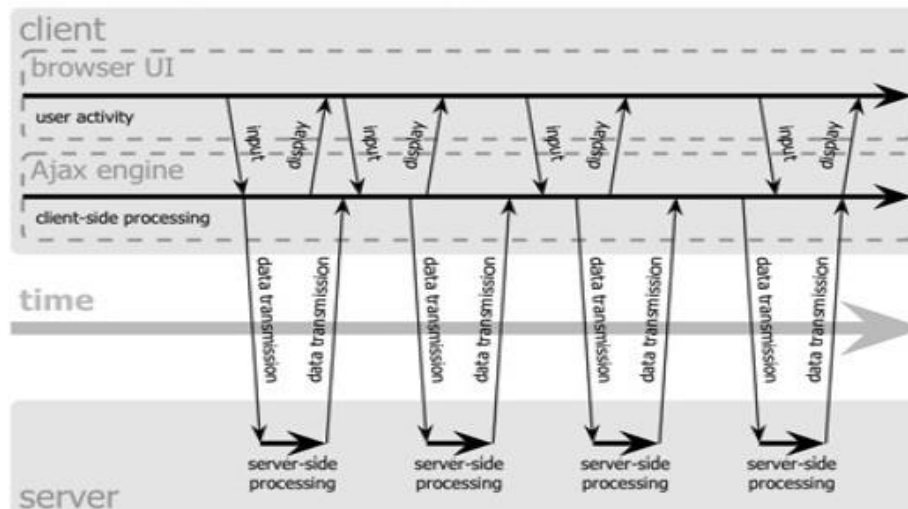
Synchronous vs. Asynchronous

classic web application model (synchronous)



Interrupted user operation while the data is being fetched

Ajax web application model (asynchronous)



Uninterrupted user operation while data is being fetched

Real-Life Examples of AJAX Apps

❑ Google Suggest

- helps you with your **Google** searches. The functionality is pretty spectacular; check it out at <http://www.google.com> . Similar functionality is offered by **Yahoo! Instant Search**, accessible at <https://search.yahoo.com>

❑ GMail

- <http://www.gmail.com>
- Other web-based email services such as **Yahoo! Mail** and **Hotmail** have followed the trend and offer AJAX-based functionality.

❑ Google Maps

- <http://maps.google.com>

❑ Facebook

- <http://www.facebook.com>

❑ Youtube

- www.youtube.com

AJAX Application Example

❑ Online test

- Many multiple choice questions
- All questions are displayed on one page
- After the user answers one question, the correct answer and explanation about why the user answer is wrong is shown on the page
- For all already-answered questions, their correct answers and explanations are always shown on the page

❑ Pure sever-side solution using conventional web application

- For each question answer submission, the whole page with most of repeated data sent to the browser

❑ Pure client-side solution using conventional JavaScript

- The user can read JavaScript source code to view what is correct answer
- Large amount of explanation data will be carried by the JavaScript code

❑ AJAX solution

- After the user answers a question, use [XmlHttpRequest](#) to ask the server to send the correct answer and explanation.
- Display the correct answer and explanation received from the server

Potential benefits

- ❑ **AJAX** brings you the following potential benefits when building a new web application:
 - It makes it possible to create **better and more responsive websites** and web applications.
 - It makes web applications more **faster, more interactive, and more user friendly**
 - Because of its popularity, **it encourages the development of patterns** that help developers avoid reinventing the wheel when performing common tasks.
 - **Features of AJAX integrate perfectly with existing functionality** provided by web browsers
 - The technologies AJAX is made of **are already implemented in all modern web browsers**
 - By using AJAX, you can:
 - build Update a web page without reloading the page
 - Request/ Receive data from a server - after the page has loaded
 - Send data to a server - in the background

Potential issues

- ❑ Because the page address doesn't change while working, you can't easily bookmark AJAX-enabled pages.
- ❑ Search engines may not be able to index all portions of your AJAX application site.
- ❑ The Back button in browsers, doesn't produce the same result as with classic web applications, because all actions happen inside the same page.
- ❑ JavaScript can be disabled at the client side, which makes the AJAX application non-functional, so it's good to have another plan in your site, whenever possible, to avoid losing visitors.

Technologies Used In AJAX

- **AJAX** is based on internet standards, and uses a combination of:
 - **HTML/XHTML** (for presentation)
 - **CSS** (to style the data)
 - **DOM** (for dynamic display of and interaction with data)
 - **XML** (often used as the format for transferring data) and **XSLT** for its manipulation
 - **JSON** is a more recent alternative for data interchange
 - **XMLHttpRequest** object (to exchange data asynchronously with a server)
 - **JavaScript**: the glue for the whole operation of these technologies together

XMLHttpRequest Object

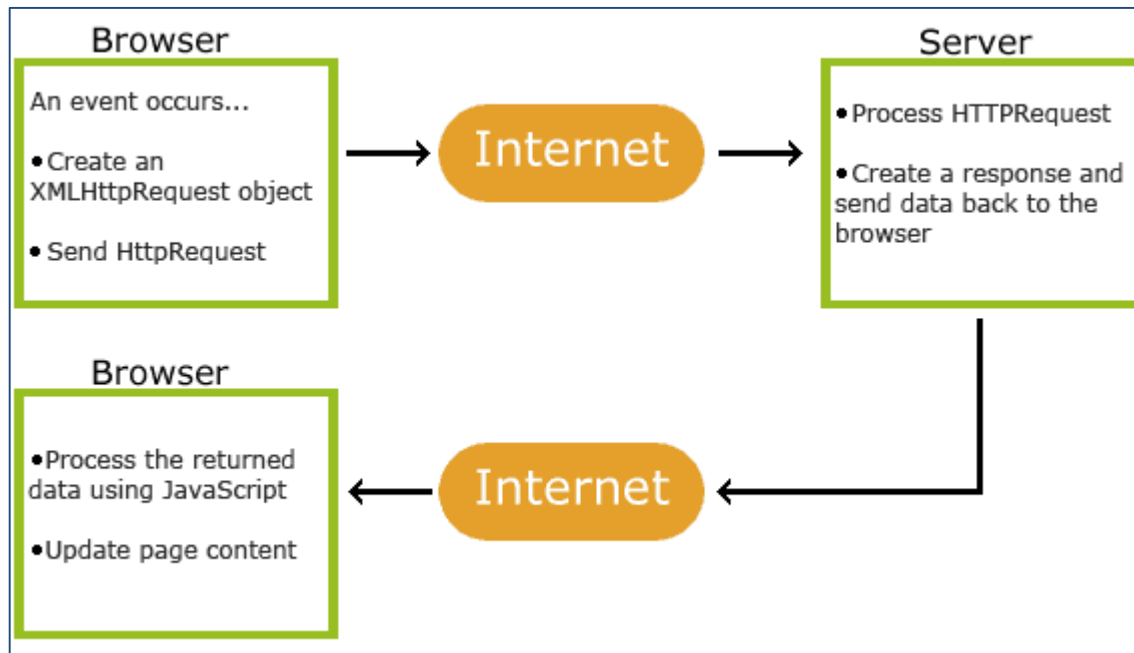
- ❑ JavaScript object
- ❑ The keystone of AJAX
- ❑ Communicates with a server via standard HTTP GET/POST
- ❑ Supported by all modern browsers.
 - Chrome, Firefox, IE7+, Edge, Safari Opera have a built-in XMLHttpRequest object
- ❑ It used to exchange data with a server behind the scenes (it is possible to update parts of a web page, without reloading the whole page)
- ❑ Works in the background for performing asynchronous communication with backend server (Does not interrupt user operation)
- ❑ Syntax for creating an XMLHttpRequest object:

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

XMLHttpRequest Properties

- ❑ `readyState` – current status of request
 - 0 = request not initialized, 1 = server connection established, 2 = request received, 3 = processing request, 4 = request finished and response is ready
- ❑ `Onreadystatechange` - Set with a JavaScript event handler that fires at each state change
- ❑ `Status` - HTTP Status returned from server, e.g. 200 = OK
- ❑ `statusText` - Status text returned from server
- ❑ `responseText` - String version of data returned from the server
- ❑ `responseXML` - XML document of data returned from the server

AJAX Workflow



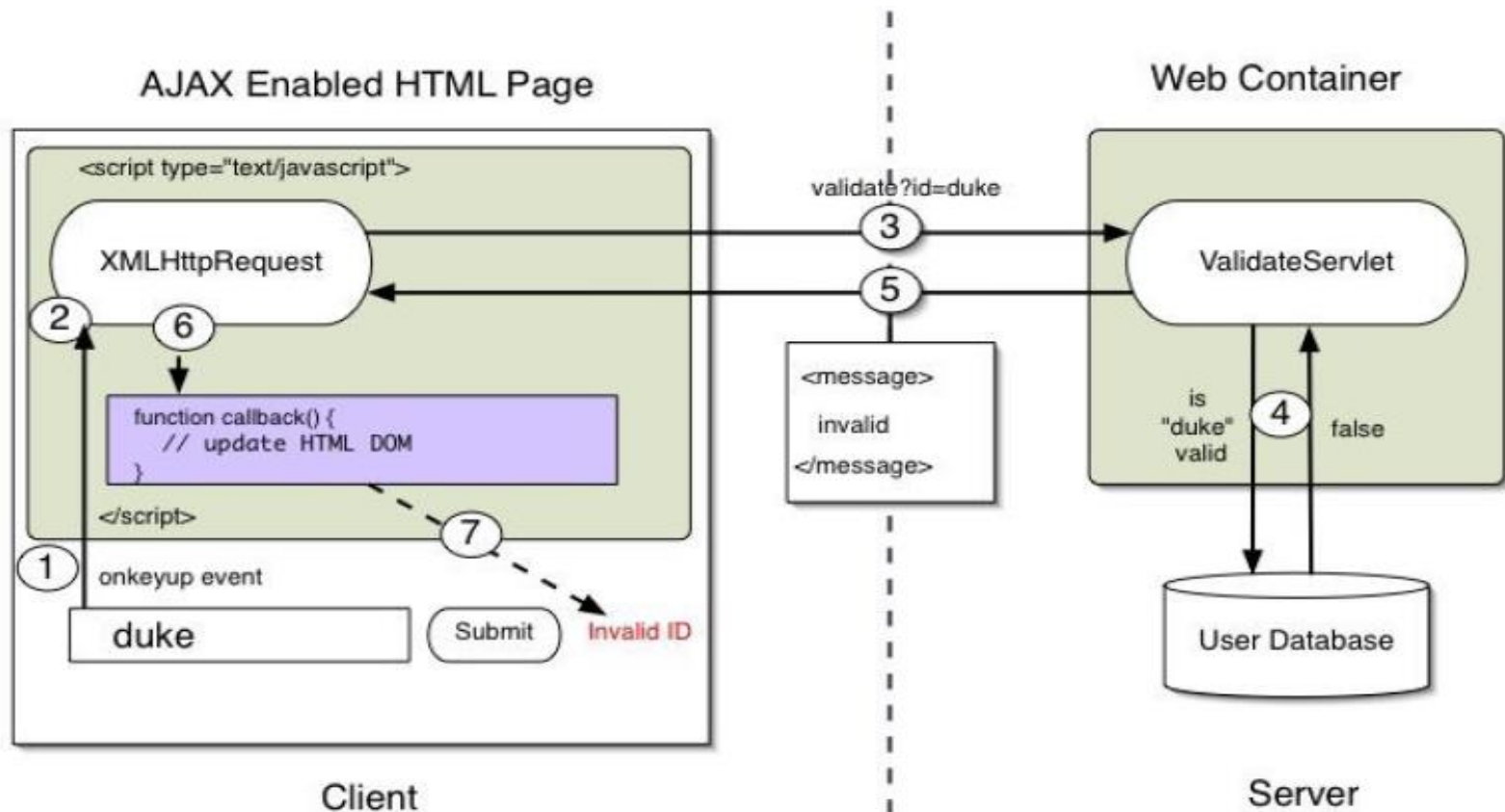
Server programming model remains the same

- It receives standard HTTP GETs/POSTs
- Can use Servlet, JSP, ASP, PHP...

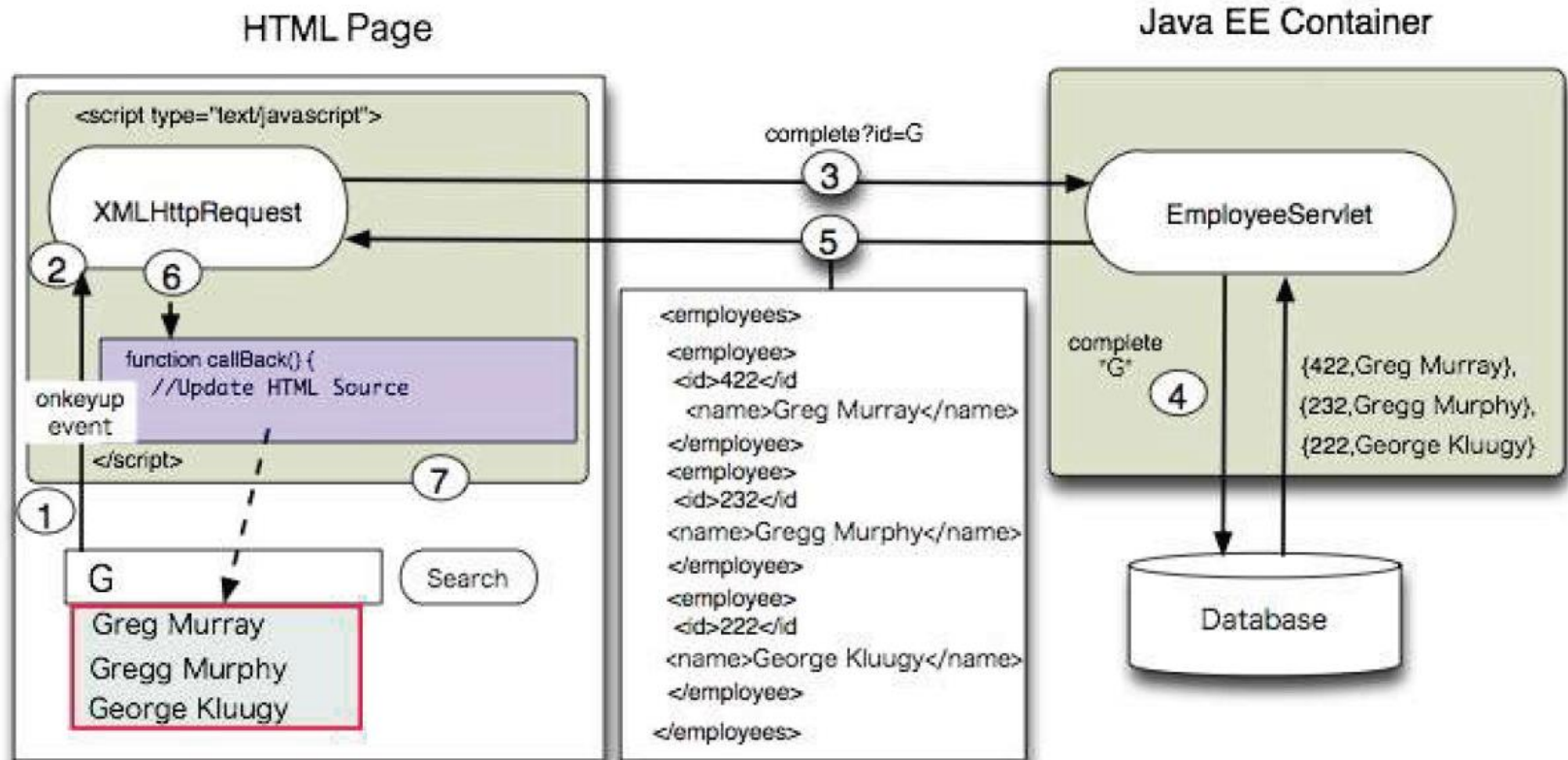
Response content type can be

- text/xml
- text/plain
- text/javascript
- text/json (JavaScript Object Notation)

AJAX: Sample App



AJAX: Another App



Ajax Typical Application Steps

1. Create an XMLHttpRequest object
2. Send a request to the server
3. Server response
4. Update HTML

1. Create an XMLHttpRequest Object

- ❑ Syntax for creating an XMLHttpRequest object:

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

- ❑ Old versions of IE (5 and 6) uses an ActiveX Object:

```
variable=new ActiveXObject("Microsoft.XMLHTTP");
```

- ❑ To handle all modern browsers, including IE5 and IE6, check if the browser supports the XMLHttpRequest object

```
var xmlhttp;  
if (window.XMLHttpRequest)  
    {  
        // code for IE7+, Firefox, Chrome, Opera, Safari  
        xmlhttp=new XMLHttpRequest();  
    }  
else  
    {  
        // code for IE6, IE5  
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}
```


2. Send a Request To a Server

- ❑ To send a request to a server, we use the `open()` and `send()` methods of the `XMLHttpRequest` object:

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

- `open(method,url,async)`: Specifies the type of request, the URL, and if the request should be handled asynchronously or not
- `method`: the type of request GET or POST
- `url`: the location of the file on the server (can be php/aspx/txt file)
 - For security reasons, modern browsers do not allow access across domains (same origin policy)
- `async`: true (**asynchronous**) or false (synchronous)

3. Server Response

- ❑ To get the response from a server, use the `responseText` or `responseXML` property of the `XMLHttpRequest` object.

The `responseText` Property:

If the response from the server is not XML, use the `responseText` property. The `responseText` property returns the response as a string, and you can use it accordingly:

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

The `responseXML` Property

If the response from the server is XML, and you want to parse it as an XML object, use the `responseXML` property

```
xmlDoc=xmlhttp.responseXML;  
txt="";  
x=xmlDoc.getElementsByTagName("BOOK");  
for (i=0;i<x.length;i++)  
{  
    txt=txt + x[i].childNodes[0].nodeValue + "<br />";  
}  
document.getElementById("myDiv").innerHTML=txt;
```

4. The onreadystatechange Event

- ❑ When a request to a server is sent, we want to perform some actions based on the response.
- ❑ Three **important properties** of the XMLHttpRequest object:

onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized Open() not called yet 1: server connection established Open() has been called 2: request received sent() has been called 3: processing request Downloading; responseText holds partial data. 4: request finished and response is ready The operation is complete.
status	Returns the status-number of HTTP request 200: "OK" 404: Page not found

Example: Ajax and PHP (DB Access)

Example

Select a person: ▼

Person info will be listed here...

The database table we use in the example above looks like this:

id	FirstName	LastName	Age	Hometown	Job
1	Peter	Griffin	41	Quahog	Brewery
2	Lois	Griffin	40	Newport	Piano Teacher
3	Joseph	Swanson	39	Quahog	Police Officer
4	Glenn	Quagmire	41	Quahog	Pilot

Output:

Joseph Swanson ▾

Firstname	Lastname	Age	Hometown	Job
Joseph	Swanson	39	Quahog	Police Officer

Example: HTML Code

```
<html><head><script type="text/javascript">
function showUser(str)
{
if (str=="")
{ document.getElementById("txtHint").innerHTML=""; return; }
if (window.XMLHttpRequest)
  // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
} else{// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
  document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","getuser.php?q="+str,true);
xmlhttp.send();
</script>
</head>
```

```
<form>
<select name="users"
  onchange="showUser(this.value)">
  <option value="">Select a person:</option>
  <option value="1">Peter Griffin</option>
  <option value="2">Lois Griffin</option>
  <option value="3">Glenn Quagmire</option>
  <option value="4">Joseph Swanson</option>
</select>
</form>
<br />
<div id="txtHint">
  <b>Person info will be listed here.</b></div>

</body>
</html>
```

The showUser() function does the following:

1. Check if a person is selected
2. Create an XMLHttpRequest object
3. Create the function to be executed when the server response is ready
4. Send the request off to a file on the server
5. Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

Example: PHP code

```
<?php
$q=$_GET["q"];

$con = mysql_connect('localhost',
'peter', 'abc123');
if (!$con)
{
    die('Could not connect: ' .
mysql_error());
}

mysql_select_db("ajax_demo", $con);

$sql="SELECT * FROM user WHERE id =
'".$q."'";

$result = mysql_query($sql);

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
<th>Hometown</th>
<th>Job</th>
</tr>";
```

```
while($row = mysql_fetch_array($result))
{
    echo "<tr>";
    echo "<td>" . $row['FirstName'] .
"</td>";
    echo "<td>" . $row['LastName'] .
"</td>";
    echo "<td>" . $row['Age'] . "</td>";
    echo "<td>" . $row['Hometown'] .
"</td>";
    echo "<td>" . $row['Job'] . "</td>";
    echo "</tr>";
}
echo "</table>";

mysql_close($con);
?>
```

When the query is sent from the JavaScript to the PHP file, the following happens:

1. PHP opens a connection to a MySQL server
2. The correct person is found
3. An HTML table is created, filled with data, and sent back to the "txtHint" placeholder

jQuery - AJAX load() Method

- ❑ JQuery provides a family of methods to simplify asynchronous requests.
- ❑ The jQuery `load()` method is a simple and powerful AJAX method.
 - It loads data from a server and puts the returned data into the selected element.

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

- `URL`: specifies the URL you wish to load (**required**).
 - `data`: specifies a set of query string key/value pairs to send along with the request.
 - `callback`: a function to be executed after the `load()` method is completed.
- ❑ The callback function can have different parameters:
 - `responseTxt` - contains the resulting content if the call succeeds
 - `statusTxt` - contains the status of the call
 - `xhr` - contains the XMLHttpRequest object

jQuery - AJAX load() Method

Example1

Info.txt

```
<h2>Introduction to use AJAX</h2>
<p> This is to show how much it is useful.</p>
```

```
<html> <head>
<script src="jquery.min.js"> </script>

<script>
    $(document).ready(function() {
        $("button").click(
            function() {
                $(".info").load("info.txt");
            });
    });
</script>
</head>
<body>
    <button> Click </button>
    <div class="info"> Click to show more info.. </div>
</body>
</html>
```

Click

Click to show more info..

Click

Introduction to use AJAX

This is to show how much it is useful

jQuery - AJAX load() Method

Example2

https://www.w3schools.com/jquery/jquery_ajax_load.asp

```
<html> <head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function() {
    $("button").click(function() {
        $(".info").load("info.txt",function(responseTxt, statusTxt, xhr){
            alert(responseTxt);
            alert(statusTxt);
            alert(xhr.status);});
    });
</script> </head>
<body>
    <button> Click </button>
    <div class="info"> Click to show more info.. </div>

</body>
</html>
```

Info.txt

```
<h2>Introduction to use AJAX</h2>
<p> This is to show how much it is useful.</p>
```

Click

Click to show more info..

Click

Introduction to use AJAX

This is to show how much it is useful

localhost says:

```
<h1> Introduction to use AJAX </h1>
<p> This is to show how much it is useful </p>
```

OK

localhost says:

success

OK

localhost says:

200

OK

jQuery - AJAX load() Method

https://www.w3schools.com/jquery/jquery_ajax_load.asp

Info.txt

Example2

```
<h2>Introduction to use AJAX</h2>
<p> This is to show how much it is useful.</p>
```

```
<!DOCTYPE html>
<html><head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function() {
  $("#button").click(function() {
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr) {
      if(statusTxt == "success")
        alert("External content loaded successfully!");
      if(statusTxt == "error")
        alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
  });
});
</script></head>
<body>

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

<button>Get External Content</button>

</body>
</html>
```

Let jQuery AJAX Change This Text

Get External Content

An embedded page on this page says
External content loaded successfully!

OK

jQuery and AJAX is FUN!

This is some text in a paragraph.

Get External Content

jQuery ajax() Method

- ❑ There are many jQuery AJAX methods → to see more about jQuery AJAX Methods: https://www.w3schools.com/jquery/jquery_ref_ajax.asp

- ❑ `$.ajax()` – is used to perform an asynchronous HTTP (ajax) request.
 - All jQuery AJAX methods use the `ajax()` method

- ❑ `$.ajax(url, [settings])`
 - `url` : a string containing the url - optional
 - `settings` : key-value pairs

- ❑ Example:

```
$.ajax({  
  type: "POST",  
  url: "some.php",  
  data: { name: "John", location: "Boston" }  
});
```

Example1

```
<html><head>
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function() {
    $("button").click(function() {
        $.ajax({
            url: "info.txt",
            success: function(result) {
                $("#div1").html(result);
            }
        });
    });
});
</script> </head>
<body>
<button>Get External Content</button>
<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

</body> </html>
```

Get External Content

Let jQuery AJAX Change This Text

Get External Content

Introduction to use AJAX

This is to show how much it is useful

```
$("#selector").load("anotherpage.html")
```

```
url: "anotherpage.html",
```

Example2

```
<script src="jquery-3.2.1.min.js"></script>
<script>
$(document).ready(function() {
    $("form").submit(function() {
        $.ajax({
            url: "file.php",
            type: "POST",
            data: $("form").serialize(),

            success: function(result, status) {
                alert(status);
                $("#div1").html(result);
            } });
        return false;
    });
});
</script>
```

The serialize() method creates a URL encoded text string by serializing form values.

First name:

Last name:

FirstName=Ahmed&LastName=Khalid

```
<form>
    <input type="text" name="username"/>
    <input type="text" name="major"/>
    <input type="submit"/>
</form>
<div id="div1"> The given data are.. </div>
```

First name:

Last name:

```
<?php
$name= $_POST['username'];
$major= $_POST['major'];
echo "You name : ". $name.
    " </br> Your major is : " . $major;
?>
```

Form page

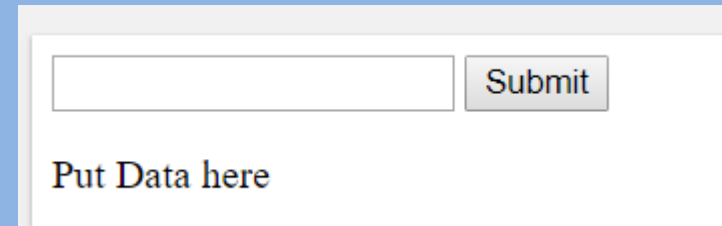
File.php

Example3

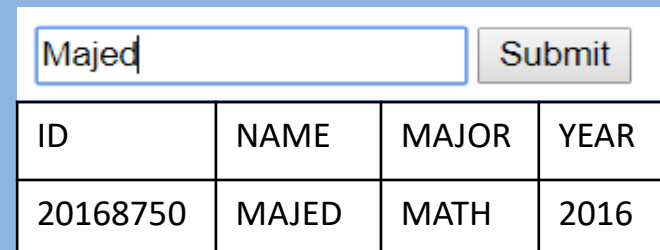
Form page

```
<html> <body>
<form>
    <input type="text" name="name"/>
    <input type="submit"/>
</form>
<div class="dataDiv"> Put Data here </div>

<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $("form").submit(function() {
            $.ajax({
                url: "fetch.php",
                type: "POST",
                data: $("form").serialize(),
                success: function(result) {
                    $(".dataDiv").html(result);
                }
            });
            return false;
        });
    });
</script>
</body></html>
```



A screenshot of a web form. It features a single-line text input field and a 'Submit' button. Below the input field, the text 'Put Data here' is displayed, indicating the initial state of the dataDiv before any data is submitted.



A screenshot of the same web form after a submission. The text input field now contains the name 'Majed'. Below the input field, a table is displayed, showing the data received from the server.

ID	NAME	MAJOR	YEAR
20168750	MAJED	MATH	2016

Example3...

fetch.php

```
<?php
$name=$_POST['name'];
$link = mysqli_connect("localhost", "root", "", "training");
// Attempt select query execution
$sql = "SELECT * FROM stud_info WHERE NAME LIKE '" . $name . "%'";
if($result = mysqli_query($link, $sql)){
    if(mysqli_num_rows($result) > 0){
        echo "<table>"; echo "<tr>";
        echo "<th>ID</th>"; echo "<th>NAME</th>";
        echo "<th>MAJOR</th>";echo "<th>YEAR</th>"; echo "</tr>";
        while($row = mysqli_fetch_array($result)){
            echo "<tr>"; echo "<td>" . $row['ID'] . "</td>";
            echo "<td>" . $row['NAME'] . "</td>";
            echo "<td>" . $row['MAJOR'] . "</td>";
            echo "<td>" . $row['YEAR'] . "</td>";
            echo "</tr>";
        }
        echo "</table>";
        mysqli_free_result($result); // Free result set
    } else{
        echo "No records matching your query were found.";
    }
} else{
    echo "ERROR: Could not able to execute $sql. " . mysqli_error($link);
}
mysqli_close($link); // Close connection
?>
```

Explore more examples

- ❑ https://www.w3schools.com/xml/tryit.asp?filename=tryajax_suggest_php
- ❑ https://www.w3schools.com/xml/tryit.asp?filename=tryajax_database
- ❑ https://www.w3schools.com/xml/ajax_examples.asp

