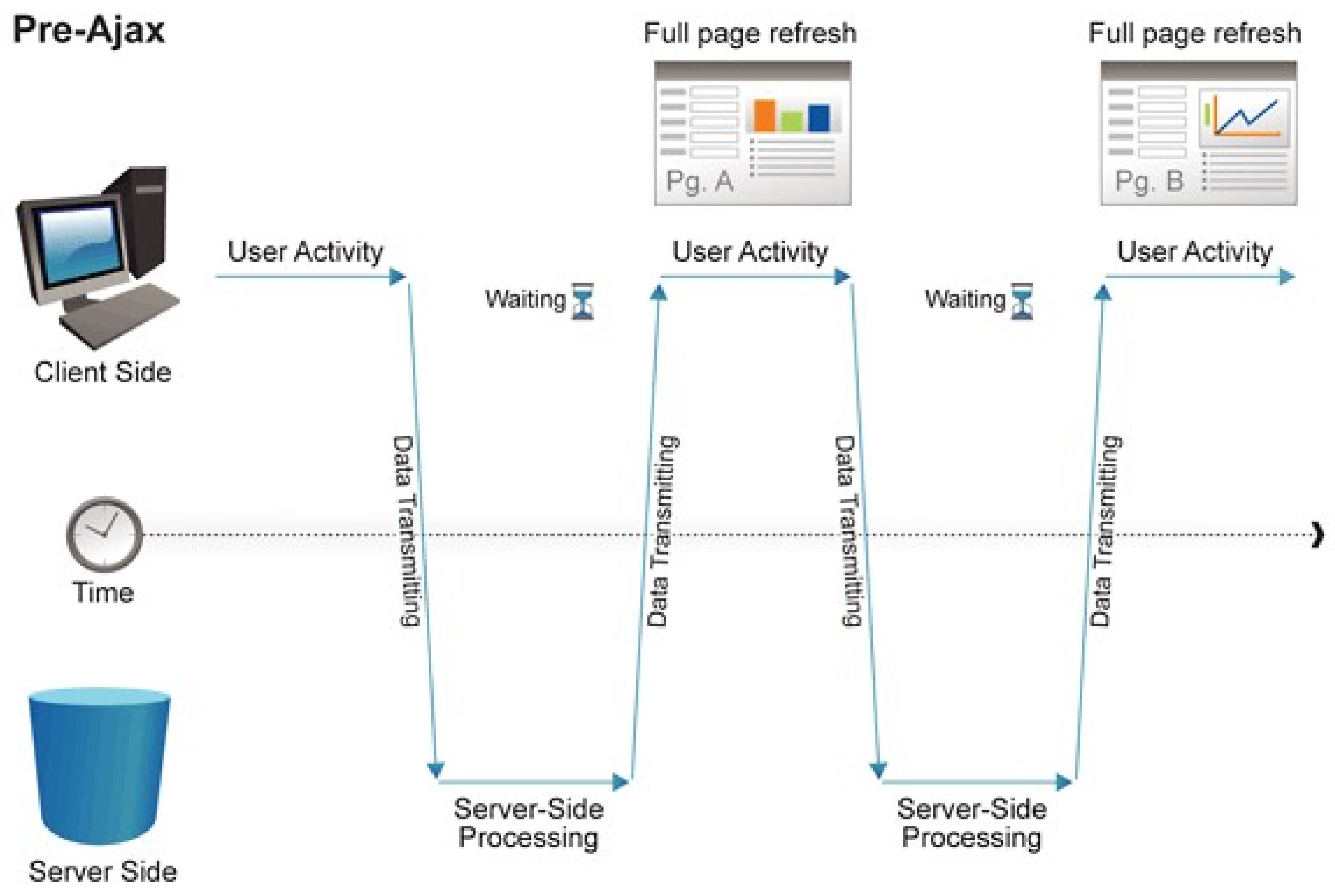
Introduction to AJAX

Introduction to AJAX

- The request and response model for HTTP that we have used so far building PHP web applications is fairly primitive.
- Each time a UI component is changed (e.g. typing a value into a textbox) the user has to submit the page or form back to the server for processing and wait for the whole page to be rendered back to their client machine.
- In most cases this requires unnecessary data and processing e.g. the header and menu structure is requested and rendered but probably hasn’t changed since the last page request so this is wasteful.
- The diagram on the next slide demonstrates the life cycle of subsequent page requests in a simple design showing that full page contents are refreshed for each request.

University of Salford MANCHESTER

Full page refresh method



http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.png

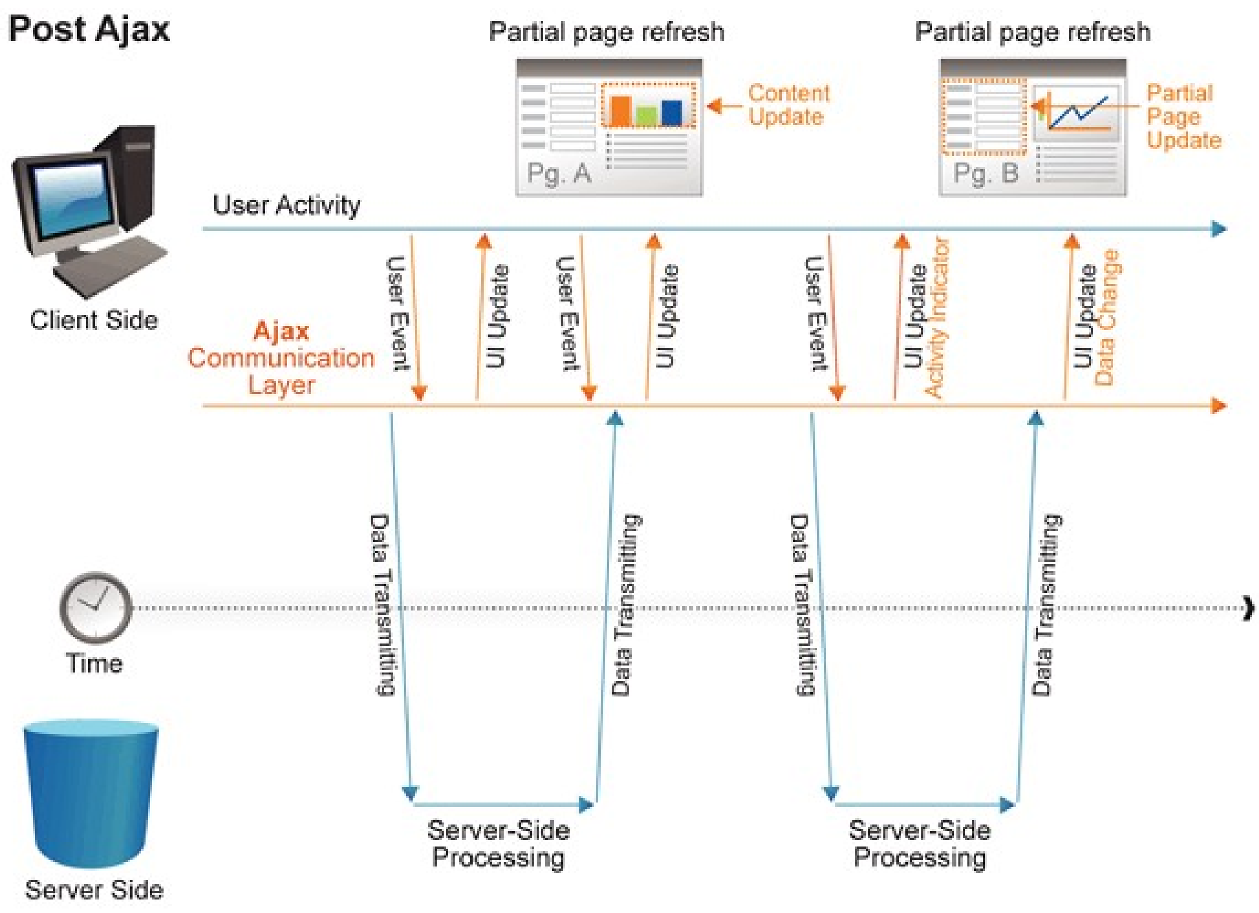
AJAX method

- AJAX stands for Asynchronous JavaScript and XML and is the general idea that data is delivered to a web page when it is requested by an event in that page (UI event, timer event etc.) without disrupting the user experience with whole page refreshes.

- The specific technologies involved in implementing an AJAX approach differ and in fact have changed over the years. For example the X meaning XML data format has been replaced by other ways of representing data e.g. JSON (we'll come to that later). Technologies used include:
  - *HTML and CSS for presentation*
  - *The Document Object Model (DOM) for dynamic display & interaction with data*
  - *JSON or XML for the interchange of data from server to client*
  - *The XMLHttpRequest object for asynchronous communication*
  - *JavaScript to bring these technologies together*

- The diagram on the next slide demonstrates the life cycle of data requests in an AJAX style application design showing that only partial page contents are fetched or refreshed for each request event.

Async page refresh method



http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.png

AJAX method

- AJAX techniques are implemented in code on the client side primarily, and there are many libraries that offer simplified ways of doing this (e.g. jQuery)

- In this part of the module though it is important to see how AJAX is implemented at the lowest level using raw JavaScript.

- The basis of this process is the XMLHttpRequest object which was originally designed by **Microsoft**! around 1999 and later adopted by Mozilla, Apple, and Google. Since October 2014, it has been standardized (and will be superseded by *fetch()* in the future)

- Despite its name, **XML**HttpRequest can be used to retrieve any type of data, not just XML, and it supports protocols other than HTTP (including file and ftp).

AJAX Request

- 1-create XMLHttpRequest
- 2- open the request
- 3- send

```
var xhttp = new XMLHttpRequest();
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)

send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

More in open

GET Requests

In the example given, you may get a cached result. To avoid this, add a unique ID to the URL:

```
xhttp.open("GET", "demo_get.asp?t=" +
Math.random(), true);
xhttp.send();
```

If you want to have parameter with the GET method, add the information to the URL:

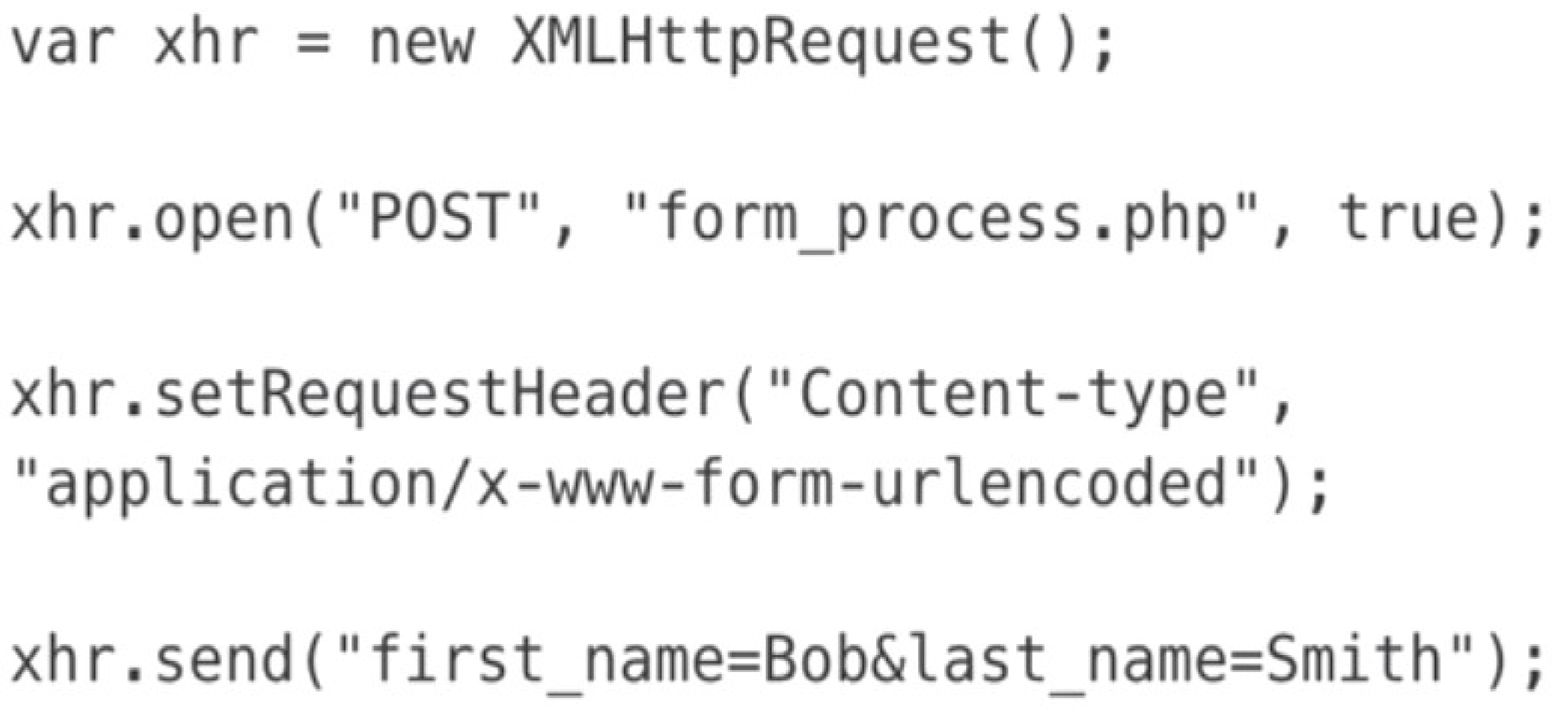
```
xhttp.open("GET", "demo_get2.asp?
fname=Henry&lname=Ford", true);
xhttp.send();
```

University of Salford MANCHESTER

Always get?

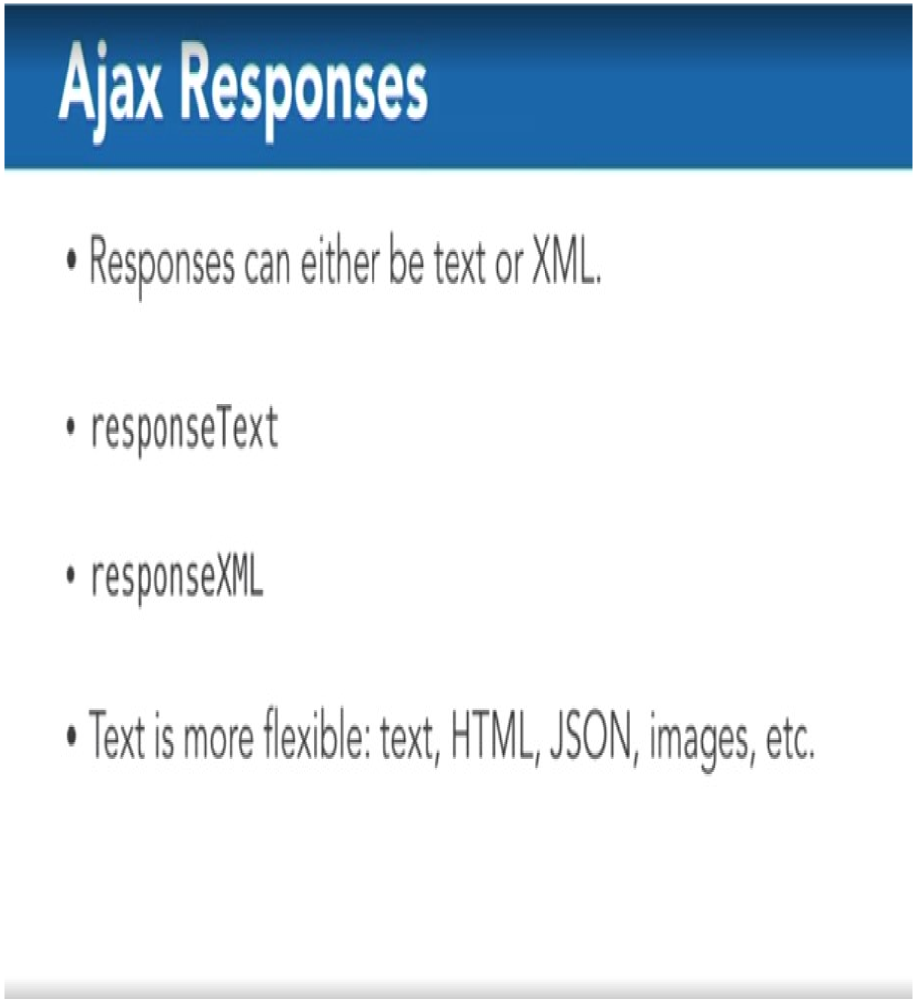
You can have post also when you send form

- Need to change header
- Send ajax with form data



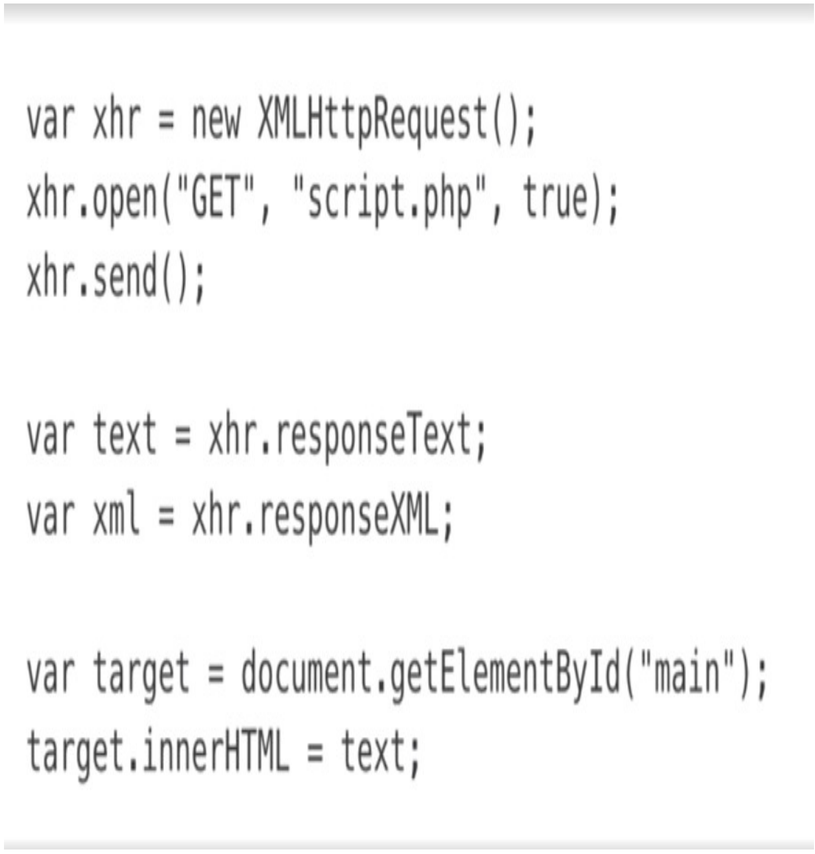
```
var xhr = new XMLHttpRequest();

xhr.open("POST", "form_process.php", true);

xhr.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");

xhr.send("first_name=Bob&last_name=Smith");
```

Ajax response



Ajax Responses

- Responses can either be text or XML.
- responseText
- responseXML
- Text is more flexible: text, HTML, JSON, images, etc.



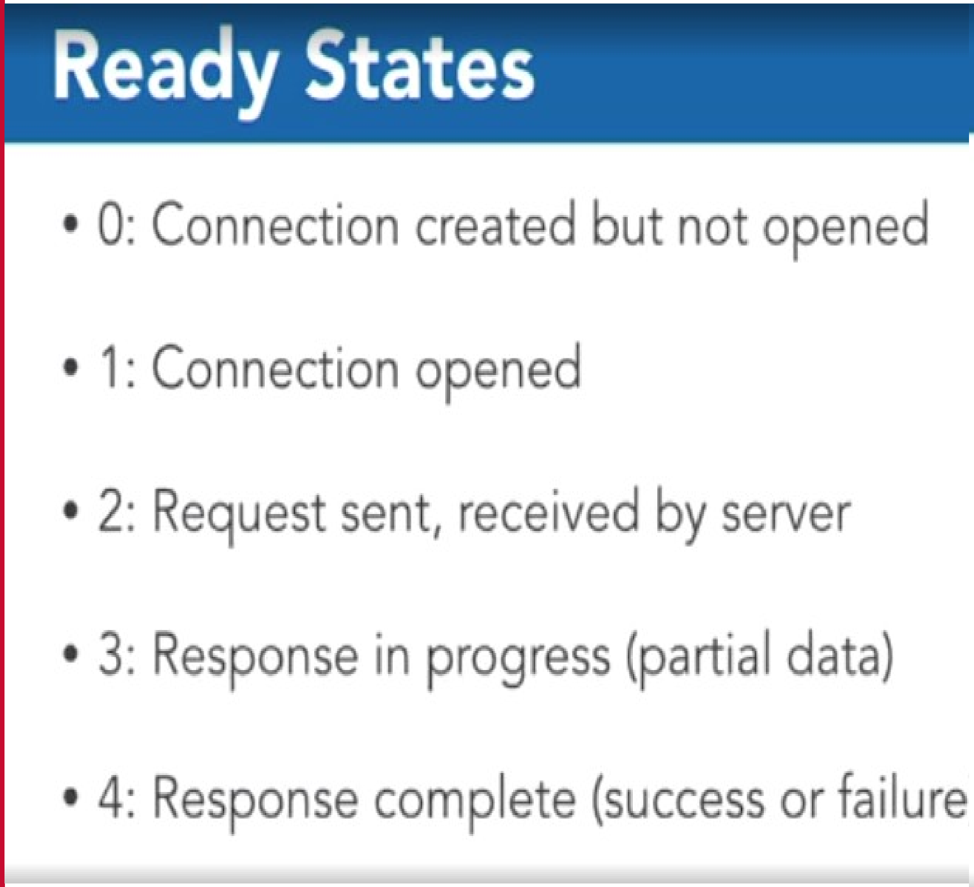
```
var xhr = new XMLHttpRequest();
xhr.open("GET", "script.php", true);
xhr.send();

var text = xhr.responseText;
var xml = xhr.responseXML;

var target = document.getElementById("main");
target.innerHTML = text;
```

Response state and Event

How does the client know the response is ready?

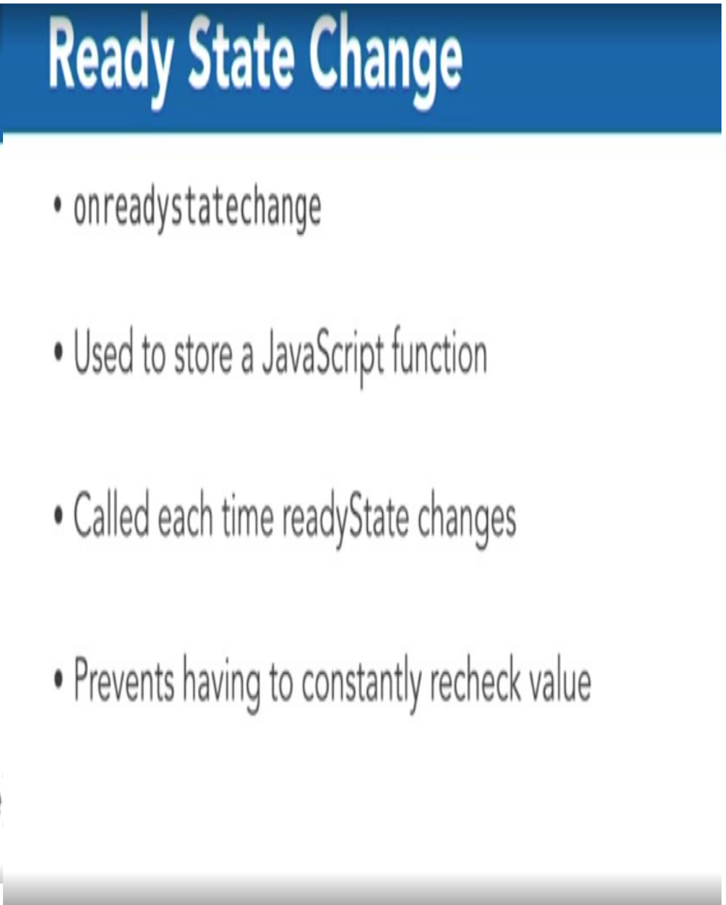
The **readyState** property holds the status of the XMLHttpRequest.

The **onreadystatechange** property defines a function to be executed when the readyState changes.



Ready States

- 0: Connection created but not opened
- 1: Connection opened
- 2: Request sent, received by server
- 3: Response in progress (partial data)
- 4: Response complete (success or failure



Ready State Change

- onreadystatechange
- Used to store a JavaScript function
- Called each time readyState changes
- Prevents having to constantly recheck value

Put all together /review

- The **`XMLHttpRequest()`** method is the core of the process to establish a communication channel to a server application.

```
// This is the client-side script.
// Initialize the HTTP request.

var xhr = new XMLHttpRequest();
xhr.open('GET', 'send-ajax-data.php');
```

The object is created and the open() method is used to prepare a GET request (in this instance) to a PHP script. Once sent (see next slide), the PHP script will return some data which can then be processed by the client with more JavaScript.

- The **`XMLHttpRequest()`** method uses a send() method to initiate the request but needs to be told how to handle any data returned. Thus an event handler function needs to be added to the client script:

```
// Track the state changes of the request.
xhr.onreadystatechange = function () {
       var DONE = 4; // readyState 4 means the request is done.
       var OK = 200; // status 200 is a successful return.
       if (xhr.readyState === DONE)
       {
              if (xhr.status === OK) {
                  console.log(xhr.responseText);  //  'This  is  the
output.'
              }
              else
              {
              console.log('Error: ' + xhr.status);
              // An error occurred during the request.
              }
       }
 };
```

PHP script to service the req

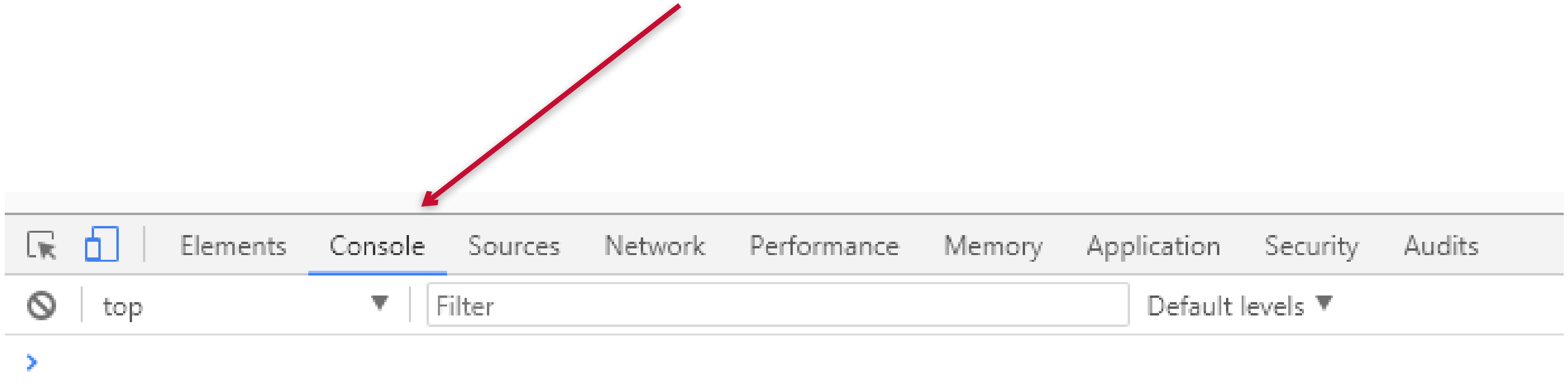
- The `send-ajax-data.php` script sends data back from the server. This example just sends a string back.

```php
<?php
// This is the server-side script.
// Set the content type.
header('Content-Type: text/plain');
// Send the data back.
echo "This is the output.";
?>
```

- **`xhr.responseText`** in the JavaSctipt holds the returned server string value and this can be inserted into the DOM of the page.
- The client UI via JavaScript can be made to trigger server calls like then when necessary to update part of the page without a complete page reload.

Workshop 15.1

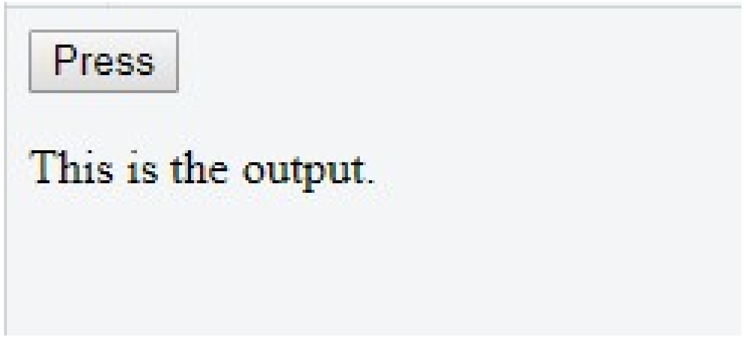
- Create an HTML page with the AJAX code on slides 186 and 187 in a <script> element and the PHP server code on slide 188 in a .php file in PHPStorm.
- Load the page in Chrome and check the **Console** via the inspector to see that output string `"This is the output."` appears



Workshop 15.2

- The simple example in 15.1 executes the XMLHttpRequest when the page is loaded but it is more realistic to execute the call on a UI event such as a button press or text input action.

- Add a text element (e.g. a <p>) and button to your webpage and modify the JavaScript code to wrap the `xhr.send(null);` call in a **function** that is called when the button is pressed – look back at your JavaScript workshops at event listeners. You can keep the xhr constructor and `.Onreadystatechange` code in a `<script>` element loaded when the page loads.



This is the output.

Ajax server-side programing

With security consideration

Ajax server-side programing

Detecting Ajax request by Server

- We can have two approaches:
  - Page will assume all requested is Ajax
  - Page detected if request was Ajax so it can handle them differently

we do this by use SetRequestheder in ajax and in php write cod for dealing with it as shown in next slide

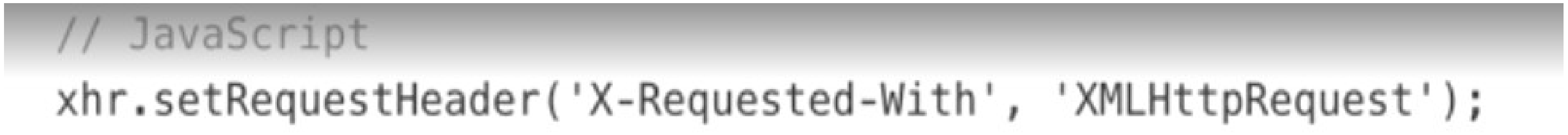
Security advise

http://tech.beacondeacon.com/ajax-referrer.html
https://thisinterestsme.com/detecting-ajax-requests-with-php/

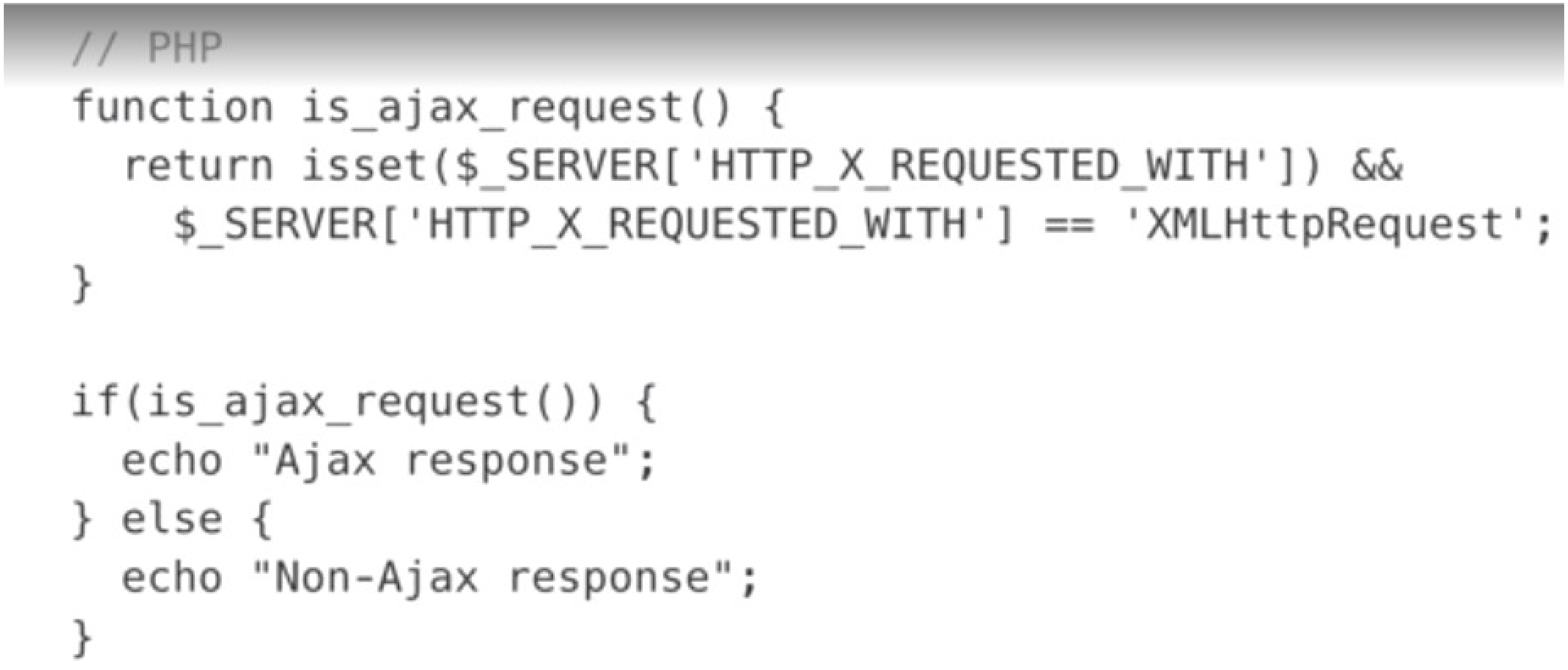
https://processwire.com/talk/topic/20605-easy-csrf-protection-for-ajax-requests-everywhere/

Ajax server-side programing

- Detect Ajax request

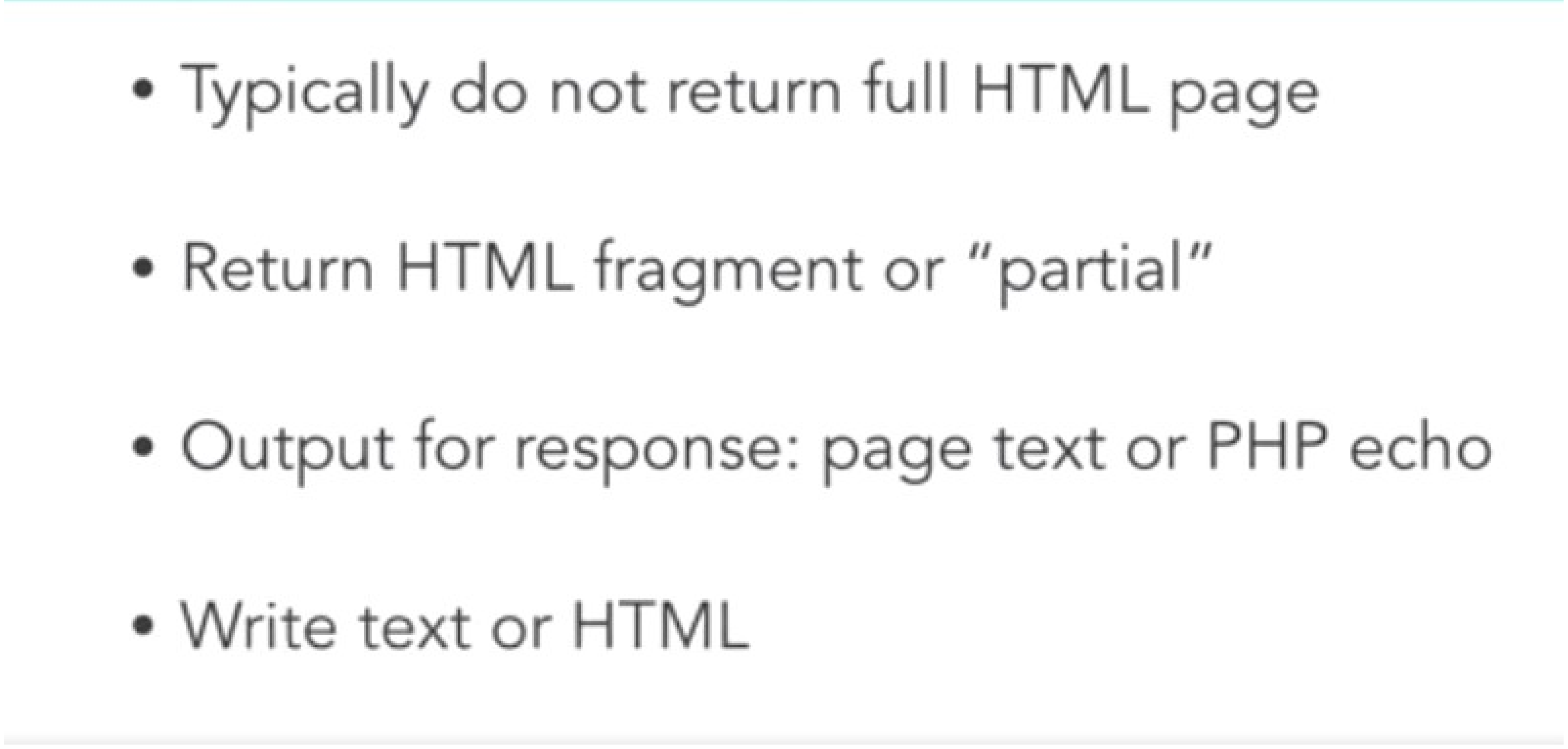


```
// JavaScript
xhr.setRequestHeader('X-Requested-With', 'XMLHttpRequest');
```



```
// PHP
function is_ajax_request() {
  return isset($_SERVER['HTTP_X_REQUESTED_WITH']) &&
    $_SERVER['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest';
}

if(is_ajax_request()) {
  echo "Ajax response";
} else {
  echo "Non-Ajax response";
}
```

Respond with HTML partials



- Typically do not return full HTML page
- Return HTML fragment or “partial”
- Output for response: page text or PHP echo
- Write text or HTML

RESPONSE

Ajax server-side programing

Ajax response is either a text or XML, with two different function

Response Text	get the response data as a string
responseXML	get the response data as XML data

text is more flexible, and it can includes JSon

Data and AJAX

- The example we did work with simple text.
- When the client received the string it had to process it in someway to organize the data ready to display in UI controls.
- Depending on how the data sent from the server is formed (e.g. string of text, string of CSV values) the client application probably has to split() it in someway and the developer needs to know the format structure to make this work.
- E.g. 2 “sets” of data could be sent in the form:

  “Bob,Jones,45,9783092\n“Dave,Smith,34,893209”

This is ok, but not very extensible or portable as a concept or particularly readable as you have to know what each field represents.

Data and AJAX

- Consider a more complete CSV example which also contains field names (in the first line) and with records seperated by a new line

```
Firstname,Lastname,Age,ID
Bob,Jones,45,9783092
Dave,Smith,34,893209
```

It is better but the programmer still has to look inside the data to see the format.

- The X in AJAX stands for XML – XML is a standardized, “readable”, self describing data format - eXtensible Markup Language.

- Using a standardised scheme for data formatting allows other applications to use the same data source more easily.

- It also allows the data to be converted into other formats more easily as libraries of functions have been created to do this efficiently.

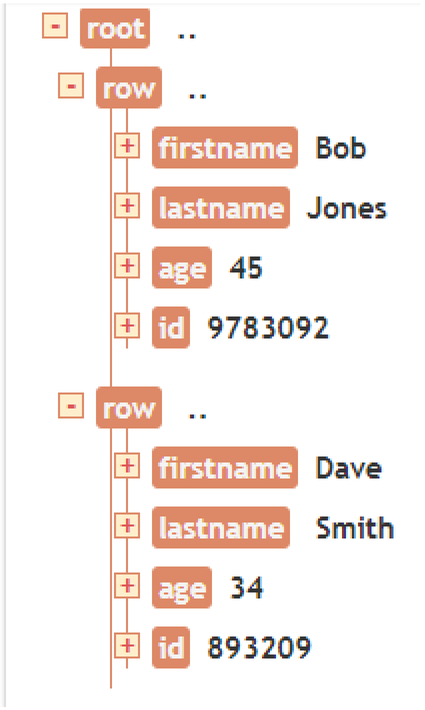
CSV to XML

- Using the previous CSV example (which in reality is just a stream of characters):

```
        "Firstname,Lastname,Age,ID\n1,Bob,Jones,45,9783092\
nDave,Smith,34,893209"
```

- The XML scheme version for this might be:

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <row>
    <Firstname>Bob</Firstname>
    <Lastname>Jones</Lastname>
    <Age>45</Age>
    <ID>9783092</ID>
  </row>
  <row>
    <Firstname>Dave</Firstname>
    <Lastname> Smith</Lastname>
    <Age>34</Age>
    <ID>893209</ID>
  </row>
</root>
```



This is a traversable tree structure whose data elements also contain their field descriptor

CSV to XML

CSV is also limited to a flat record structure. Consider the following XML example that is better expressed as objects and their fields:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<employees>
  <employee>
    <id>1</id>
    <firstName>Leonardo</firstName>
    <lastName>DiCaprio</lastName>
    <photo>http://photos.com/Leonardo+Dicaprio7.jpg</photo>
  </employee>
  <employee>
    <id>2</id>
    <firstName>Johnny</firstName>
    <lastName>Depp</lastName>
    <photo>http://photos.com/johnny-depp-pirates.jpg</photo>
  </employee>
  <employee>
    <id>3</id>
    <firstName>Hritik</firstName>
    <lastName>Roshan</lastName>
    <photo>http://photos.com/1411921557.jpg</photo>
  </employee>
</employees>
```

Data preparation Workshop 16.1

- There are plenty of tools for converting between CSV, XML and other data formats.

- Test the following tools on the employee CSV datafile on Blackboard. You need to make sure that newline characters are maintained. Load it into this tool to convert it from CSV to XML:

- http://www.convertcsv.com/csv-to-xml.htm

- Once converted to XML you can view the tree structure with this tool:

- https://codebeautify.org/xmlviewer#

Decoding XML Workshop 16.2

- We can work with an XML structure in JavaScript by creating a DOMDocument.

- If the source XML is plain text then the DOMparser class can used to convert serialises XML to a DOMDocument.

```
parser=new DOMParser();
xmlDoc=parser.parseFromString(xmlStr,"text/xml");
```

- Consider the XML used in the previous example as a text string:

-
  ```
  var xmlStr = "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>
  <employees><employee><id>1</id><firstName>Leonardo</firstName><lastName>Di
  Caprio</lastName><photo>http://photos.com/Leonardo+Dicaprio7.jpg</
  photo></employee><employee><id>2</id><firstName>Johnny</
  firstName><lastName>Depp</lastName><photo>http://photos.com/johnny-depp-
  pirates.jpg</photo></employee><employee><id>3</id><firstName>Hritik</
  firstName><lastName>Roshan</lastName><photo>http://photos.com/
  1411921557.jpg</photo></employee></employees>";
  ```

Decoding XML Workshop 16.2

- Creating a new tree of nodes from the document based on a given node tag value

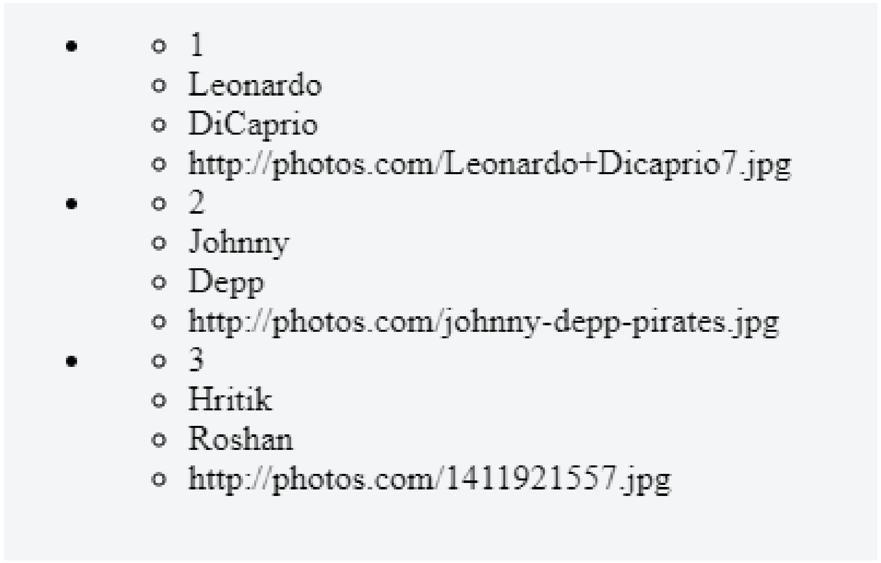
```
    x = xmlDoc.getElementsByTagName("employee"); // elements are extracted by root
tag
    for (var i = 0; i < x.length; i++) {
      //do something with these values...
            //each iteration gives details of one employee object
      newRecord = document.createElement('li'); // new an <li> element for this
employee
      newFields = document.createElement('ul'); // new a sublist of fields per
employee
      for (var j = 0; j < 4; j++) {   // extracting all the child nodes
        newField = document.createElement('li');
        newField.innerHTML = x[i].childNodes[j].firstChild.nodeValue;
        newFields.appendChild(newField)
      }
      newRecord.appendChild(newFields);

      document.getElementById("output").appendChild(newRecord);

   }
```

A named UI element to display the lists

Decoding XML Workshop 16.2

Adding a named UI element to display the list and sublist:

```
<div>
    <ul id="output">
   <!--- values are appended to this UL--->
    </ul>
</div>
```



```
•     ○ 1
      ○ Leonardo
      ○ DiCaprio
      ○ http://photos.com/Leonardo+Dicaprio7.jpg
•     ○ 2
      ○ Johnny
      ○ Depp
      ○ http://photos.com/johnny-depp-pirates.jpg
•     ○ 3
      ○ Hritik
      ○ Roshan
      ○ http://photos.com/1411921557.jpg
```

Exercise 16.2 is to put this HTML

And the preceding JavaScript together to parse and display the nodes in the <employees> XML data structure

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format which is easy for computers to parse and generate.

It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999.

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. It is a bit like a CSV file but more structured.

These properties make JSON an ideal data-interchange language for OO programing languages and where XML may be considered too bulky.

JSON - example

```
{
  "number":6,
  "people":[
    {
      "craft":"ISS",
      "name":"Alexander Misurkin"
    },
    {
      "craft":"ISS",
      "name":"Mark Vande Hei"
    },
    {
      "craft":"ISS",
      "name":"Joe Acaba"
    },
    {
      "craft":"ISS",
      "name":"Anton Shkaplerov"
    },
    {
      "craft":"ISS",
      "name":"Scott Tingle"
    },
    {
      "craft":"ISS",
      "name":"Norishige Kanai"
    }
  ],
  "message":"success"
}
```

JSON is used to construct object literals in JavaScript so you have already seen them.

In this example JSON dataset there is an anonymous root object which contains three fields: *number, people* and *message*

Number and message are simple scalar values of integer and string

People is an array of objects each one containing string fields *craft* and *name*

The dataset also contains the data values for each field.

JSON – example accessing the data to display it

JSON data string is assigned in the normal way in JavaScript using a var statement

```
 var jsondata = { "number": 6, "people": [{ "craft": "ISS", "name":
"Alexander Misurkin" }, { "craft": "ISS", "name": "Mark Vande Hei" },
{ "craft": "ISS", "name": "Joe Acaba" }, { "craft": "ISS", "name": "Anton
Shkaplerov" }, { "craft": "ISS", "name": "Scott Tingle" }, { "craft":
"ISS", "name": "Norishige Kanai" }], "message": "success" }
```

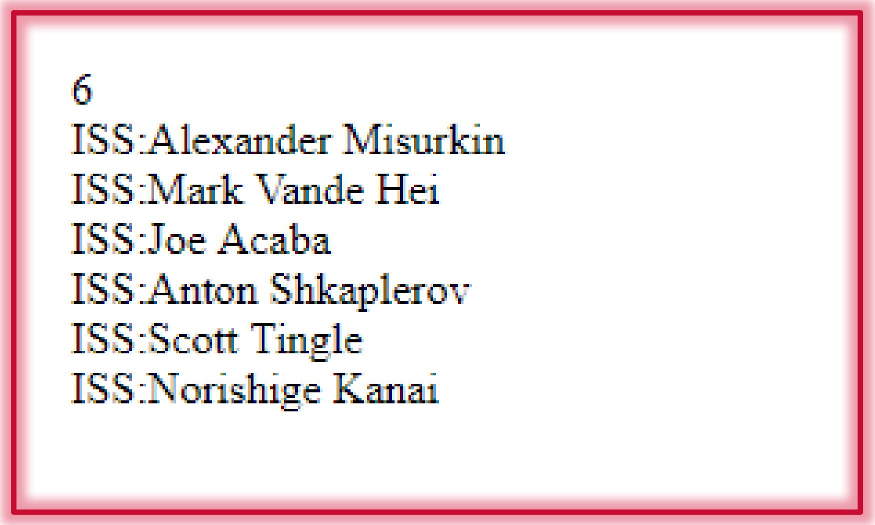
And can be read by accessing the member fields

```
jsondata.number; // accessing number member

var personObj = jsondata.people[i]; // accessing one of the people objects

personObj.craft;   // fields for that particular person
personObj.name;
```

JSON – example moving through the data

JSON data string is assigned in the normal way in JavaScript using a var statement

```
var outLabel = document.getElementById("output");

outLabel.innerHTML  += jsondata.number + "<br/>";;

for (var i = 0; i < jsondata.number; i++) {
    var personObj = jsondata.people[i];
    outLabel.innerHTML += personObj.craft + ":" + personObj.name + "<br/>";
}
```



```
6
ISS:Alexander Misurkin
ISS:Mark Vande Hei
ISS:Joe Acaba
ISS:Anton Shkaplerov
ISS:Scott Tingle
ISS:Norishige Kanai
```

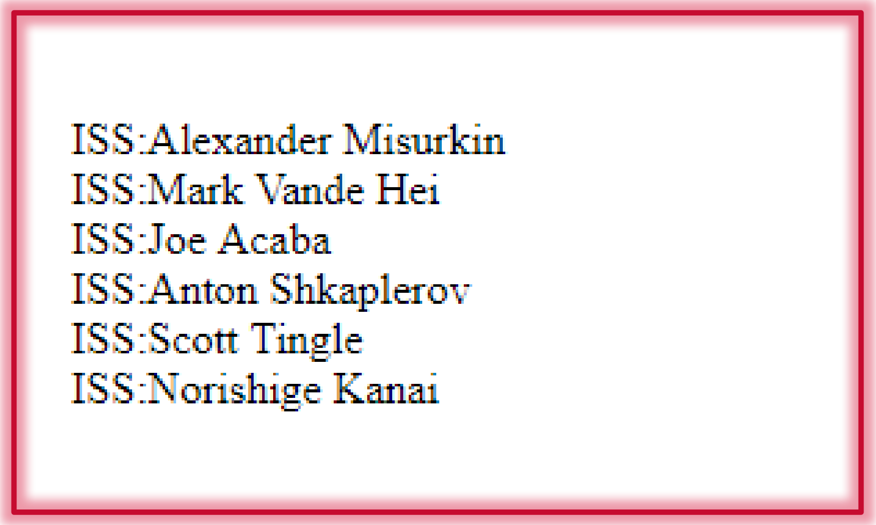
Include some HTML to display the output:

```
<p id="output">
</p>
```

JSON – example moving through the data

A more sophisticated method in JS to move through the people list with an anonymous handler function processing each object.

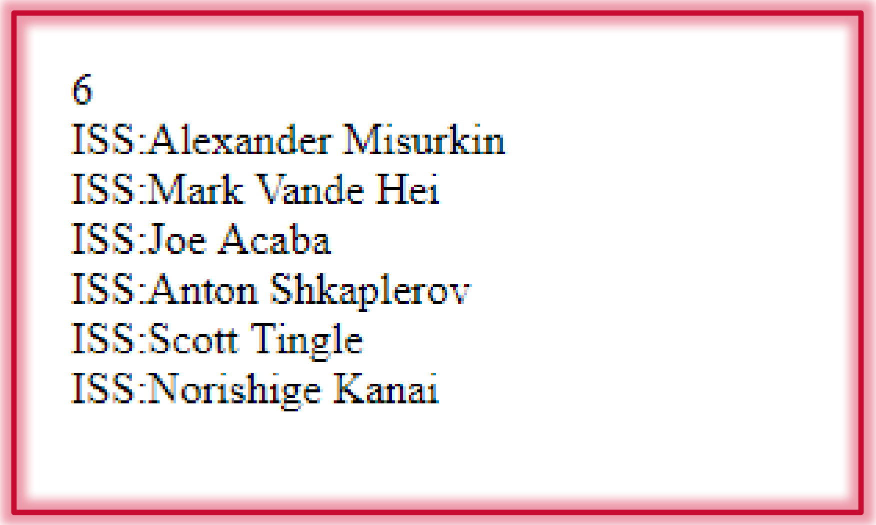
```
jsondata.people.forEach(function (obj) {
   outLabel.innerHTML += obj.craft + ": " + obj.name;
});
```



```
ISS:Alexander Misurkin
ISS:Mark Vande Hei
ISS:Joe Acaba
ISS:Anton Shkaplerov
ISS:Scott Tingle
ISS:Norishige Kanai
```

Exercise 16.3

Run through previous slide and make a test page (HTML or PHP) to hold the data and run the script to display it as shown.



```
6
ISS:Alexander Misurkin
ISS:Mark Vande Hei
ISS:Joe Acaba
ISS:Anton Shkaplerov
ISS:Scott Tingle
ISS:Norishige Kanai
```

– fetch()

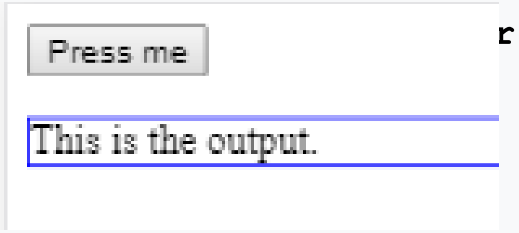
fetch() method : fetch is a new native JavaScript API. Although not yet supported by all browsers, it is gaining momentum as a more popular way to execute Ajax. According to Google documentation, "Fetch makes it easier to make web requests and handle responses than with the older XMLHttpRequest.“

```
fetch('send-ajax-data.php').then(function(response)
{
  return response.text();
}).then(function(data)
  {
    console.log(data);
  }).catch(function(error)
{ console.log('Error: ‘ + error); });
```

fetch() : 2

fetch() working with the DOM:

```
<script>
   function onclickHandler(event) {
      document.getElementById('output').innerText = 'fetching...';

      fetch('send-ajax-data.php').then(function (response) {
       return response.text(); }).then(function (data) {
                document.getElementById('output').innerText = data;
            }).catch(function (error) {
                document.getElementById('output').inner                      r;
            });
        }
</script>
```



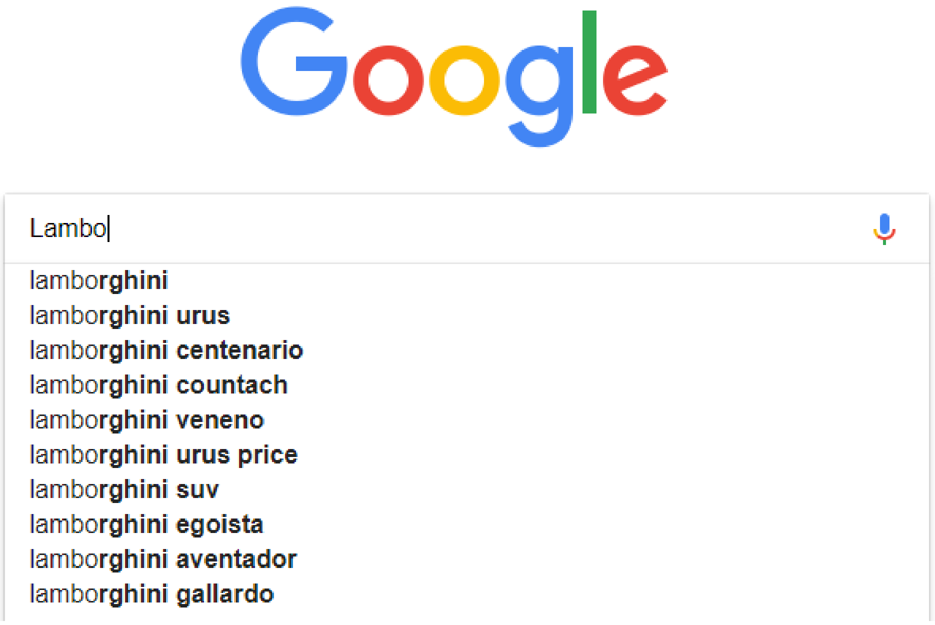
```
Press me

This is the output.
```

```
<button id="goButton">Press me</button>
<p style="border : 1px solid blue; " id="output">Output</p>


<script>
    document.getElementById('goButton').addEventListener('click',
onclickHandler, false);
</script>
```

Supplementary material API

– live search

- Now we have seen the basics of loading data as required within a client web application we can put it to use. An example we have all seen below where Google's search engine returns results suggestions as the user types

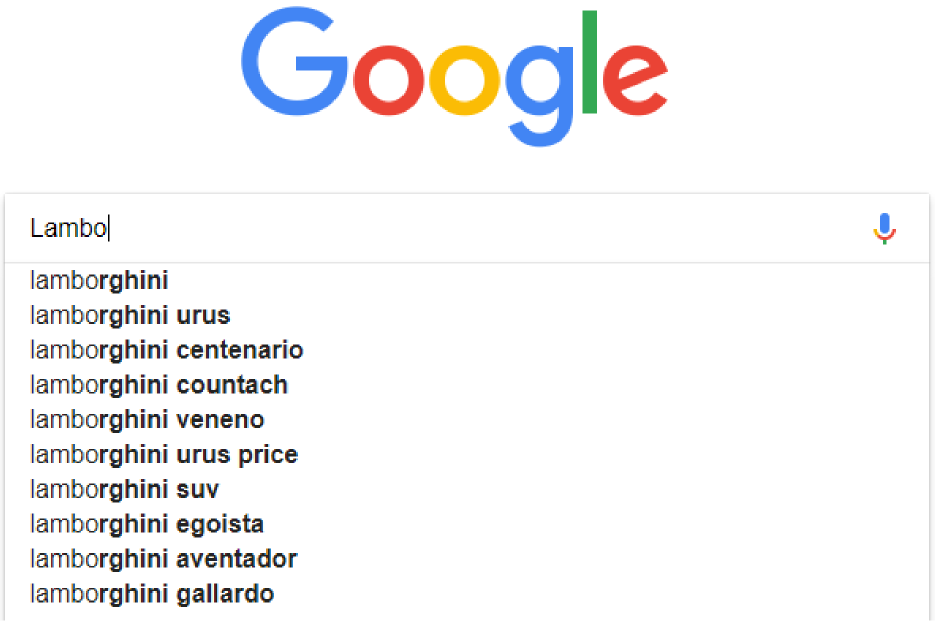


Lambo

lamborghini
lamborghini urus
lamborghini centenario
lamborghini countach
lamborghini veneno
lamborghini urus price
lamborghini suv
lamborghini egoista
lamborghini aventador
lamborghini gallardo

Workshop 15.5 – live search

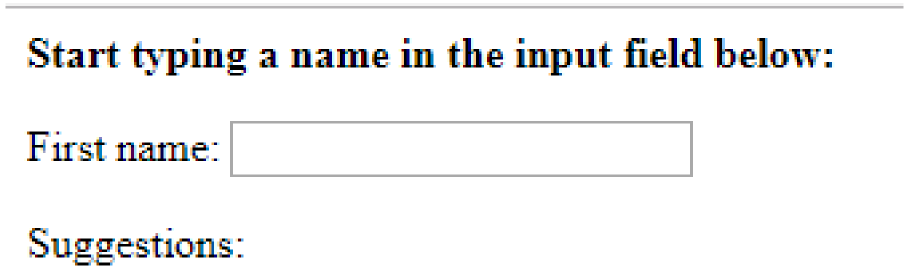
- The principle is straightforward:
  - UI text control which triggers events when changed
  - Event handler sends an AJAX type call to a server application with a text parameter in the query string
  - Server application carries out a search and returns data
  - Returned data is displayed in the UI without the need for a page refresh


Lambo
lamborghini
lamborghini urus
lamborghini centenario
lamborghini countach
lamborghini veneno
lamborghini urus price
lamborghini suv
lamborghini egoista
lamborghini aventador
lamborghini gallardo

Workshop 15.5 – live search

1) UI controls part

```
<body>

<p>Start typing a name in the input field below:</p>

<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>

<p>Suggestions: <span id="txtHint"></span></p>

</body>
```


Start typing a name in the input field below:

First name:

Suggestions:

Workshop 15.5 – live search

2) UI AJAX event handler part

```
<script>
function showHint(str) {
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  } else {
    var xmlhttp = new XMLHttpRequest();

    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        var uic = document.getElementById("txtHint")
        uic.innerHTML = this.responseText;
      }
    };
    xmlhttp.open("GET", "gethint.php?q=" + str, true);
    xmlhttp.send();
   }
 }
</script>
```

Workshop 15.5 – live search

3) Server script to carry out the search and return results

```
<?php
// Array with names
$a[] = "Anna";
$a[] = "Brittany";
$a[] = "Cinderella";
$a[] = "Diana";
$a[] = "Eva";
$a[] = "Fiona";
$a[] = "Gunda";
$a[] = "Hege";
$a[] = "Inga";
$a[] = "Johanna";
$a[] = "Kitty";
$a[] = "Linda";
$a[] = "Nina";
```

```
$a[] = "Ophelia"
$a[] = "Petunia"
$a[] = "Amanda"
$a[] = "Raquel"
$a[] = "Cindy"
$a[] = "Doris"
$a[] = "Eve"
$a[] = "Evita"
$a[] = "Sunniva"
$a[] = "Tove"
$a[] = "Unni"
$a[] = "Violet"
$a[] = "Liza"
$a[] = "Elizabeth"
$a[] = "Ellen"
$a[] = "Wenche"
$a[] = "Vicky“
. . .
```

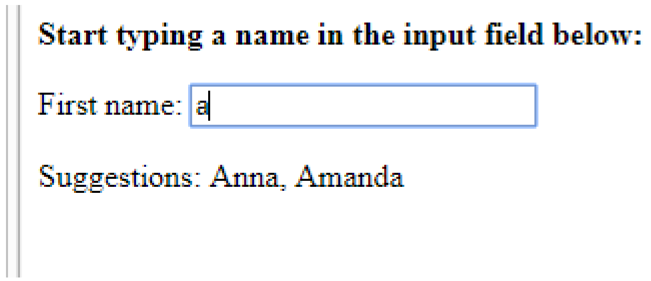
University of Salford MANCHESTER

Workshop 15.5 – live search

4) Server script to carry out the search and return results

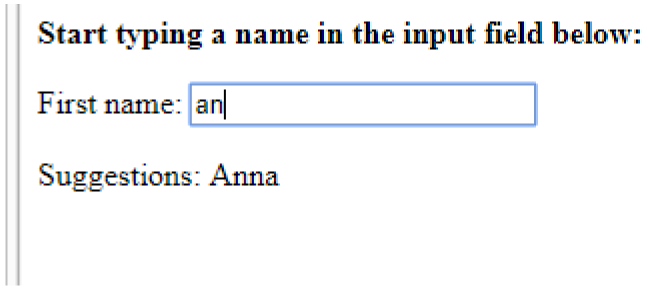
```
// get the q parameter, the text typed in, from URL
$q = $_REQUEST["q"];
$hint = "";
// lookup all hints from array if $q is different from ""
if ($q !== "") {
    $q = strtolower($q);
    $len=strlen($q);
    foreach($a as $name) {
        if (stristr($q, substr($name, 0, $len))) {
            if ($hint === "") {
                $hint = $name;
            } else {
                $hint .= ", $name";
            }
        }
    }
}
// Output "no suggestion" if no hint was found or output results
echo $hint === "" ? "no suggestion" : $hint;
```

Workshop 15.5 – live search


Start typing a name in the input field below:

First name: a

Suggestions: Anna, Amanda


Start typing a name in the input field below:

First name: an

Suggestions: Anna

- User starts typing, server application is called and returns data.
- As user types more characters the results return a more accurate match

Create this test application from the previous slides