Intro to AJAX

Dr. Michael Whitney

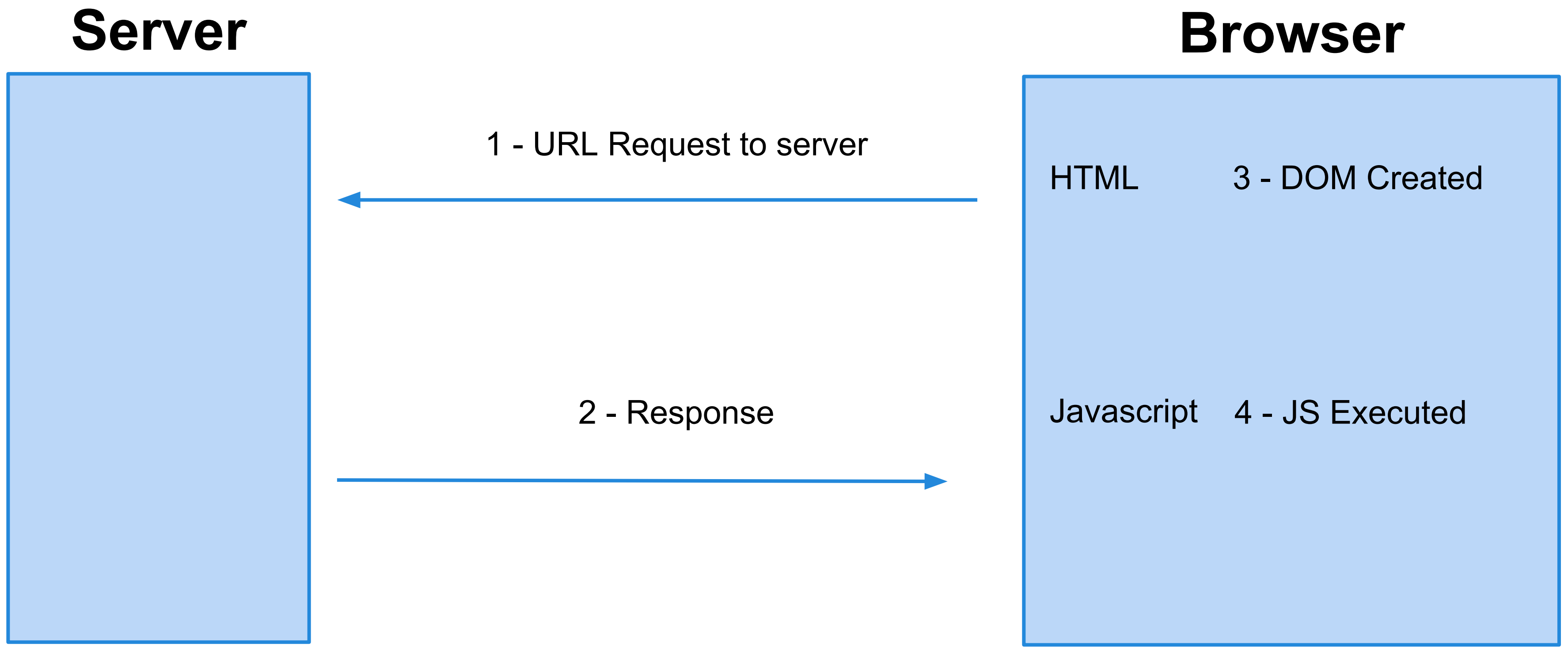
**Slides are heavily based off W3C ajax materials

What is AJAX?

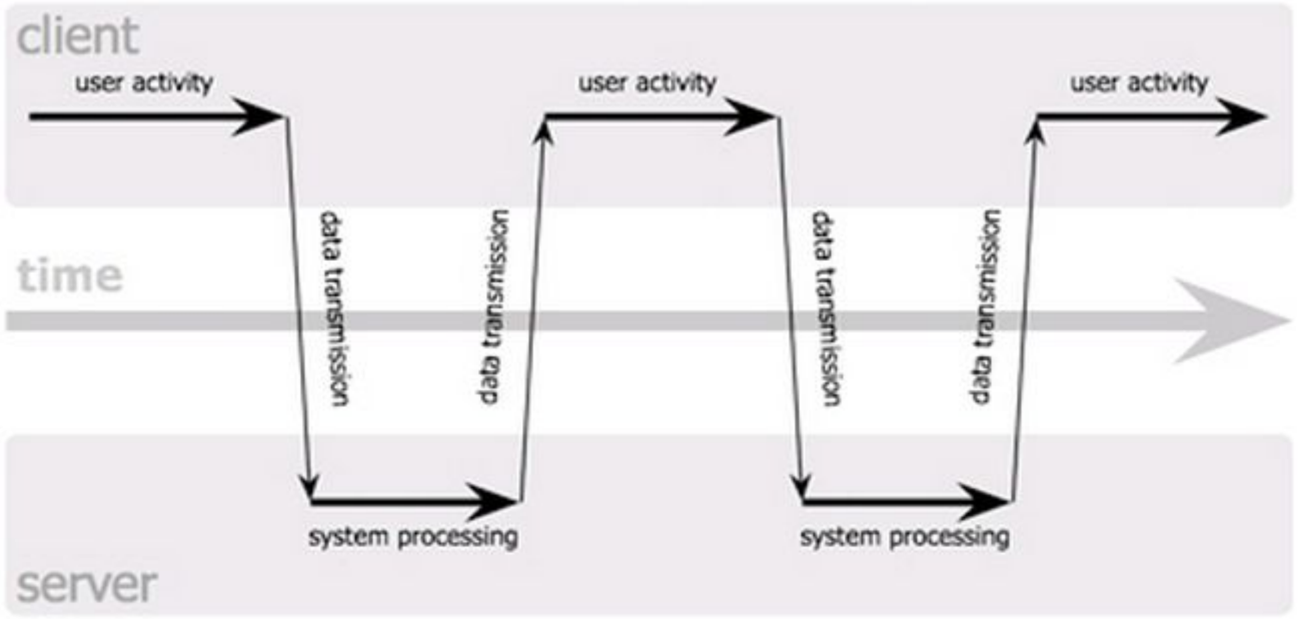
Asynchronous Javascript And XML

- Don't have to understand XML to use AJAX
- Technique for creating fast and dynamic web pages
- AJAX allows web pages to be updated asynchronously
- Who uses AJAX?
  - Google Maps, Gmail, YouTube, Facebook, ...

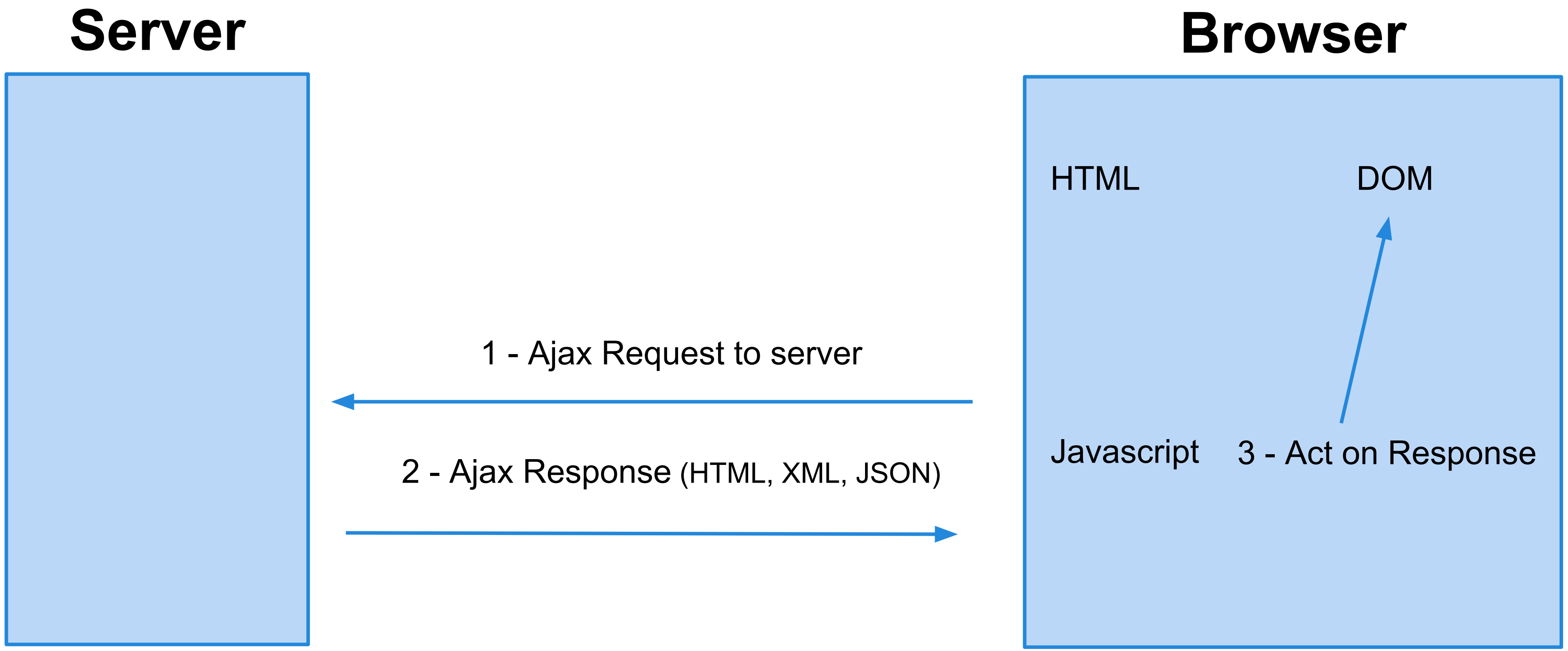
Web 1.0: Typical Web Request



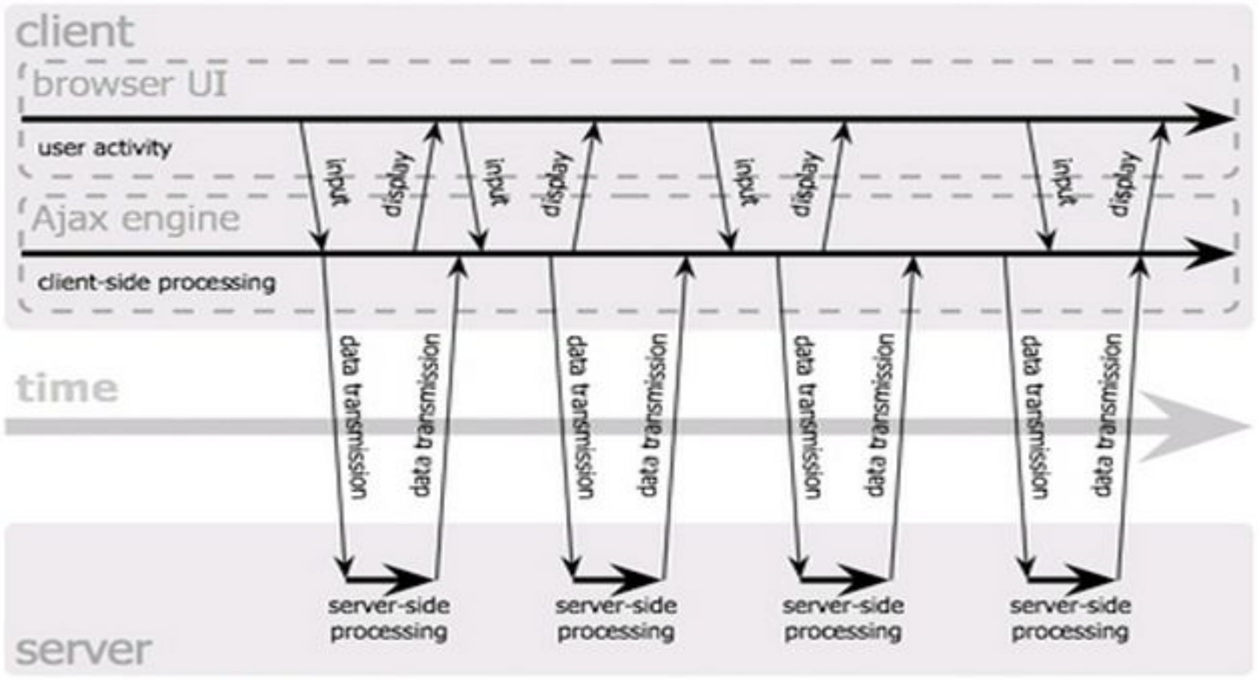
Web 1.0: Synchronous Interaction



Web 2.0: Typical Ajax Request


Server
Browser
HTML
DOM
1 - Ajax Request to server
2 - Ajax Response (HTML, XML, JSON)
Javascript
3 - Act on Response

Web 2.0: Asynchronous Interaction



Jesse James Garrett / adaptivepath.com

What is Ajax?

- Enables us to do a single page load
  - Further updates done through background HTTP requests
  - Can selectively update interface
- Based on a combination of
    - XMLHttpRequest object (to retrieve data from a web server)
    - JavaScript/DOM (to display/use the data)
- Made popular in 2005 with Google Suggest

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 asynchronous request to the Webserver.
5. The Webserver returns the result containing XML document.
6. The XMLHttpRequest object calls the callback() function and processes the result
7. The HTML DOM is updated.

The XMLHttpRequest Object

Modern browsers have this object built in

Syntax to create the object

```
variable  = new XMLHttpRequest();

var xhr = new XMLHttpRequest();
```

The Request

The `.open()` method prepares the request:

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

The Request: First argument

The first argument can be either HTTP GET or POST:

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

What is GET?

- Requests data from a specified resource
- the query string (name/value pairs) is sent in the URL of a GET request
    - /test/demo_form.asp?name1=value1&name2=value2
- requests can be cached
- requests remain in the browser history
- requests can be bookmarked
- requests should never be used when dealing with sensitive data
- requests have length restrictions
- requests should be used only to retrieve data

The Request: Second argument

The second argument specifies the file to be loaded:

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

The Request: Third argument

The third argument states whether the request is asynchronous or not:

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

Asynchronous - True or False?

```
xhr.open("GET", "ajax_test.asp", true);
```

True

- execute other scripts while waiting for server response
- deal with the response when the response ready
- when true - specify a function to execute when the response is ready in the onreadystatechange event

```
xhr.onreadystatechange = function() {....
```

Asynchronous - True or False?

```
xhr.open("GET", "ajax_test.asp", false);
```

False

- Not recommended
- JavaScript will NOT continue to execute, until the server response is ready.
- do NOT write an onreadystatechange function - just put the code after the send() statement (see next slide)

```
xhr.open("GET", "ajax_info.txt", false);
xhr.send();
```

The Request: Send

An additional line is then written to send the request:

```
var xhr = new XMLHttpRequest;
xhr.send();
```

.send() can also include arguments

```
xhr.send('search=arduino');
```

Request Review with GET

Method	**Description**
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)

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

What about POST?

- A POST is a query string where the (name/value pairs) is sent in the HTTP message body
- requests are never cached
- requests do not remain in the browser history
- requests cannot be bookmarked
- requests have no restrictions on data length

POST Request to Server

Method	**Description**
setRequestHeader(header, value)	Adds HTTP headers to the request header: specifies the header name value: specifies the header value

```
xhr.open("POST", "ajax_test.asp", true);
xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhr.send("fname=Henry&lname=Ford");
```

Get vs Post

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data.
Restrictions on data length	maximum URL length is 2048 characters	No restrictions
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs

Server Response

When the server has responded, the onload event calls an anonymous function:

```
xhr.onload = function() {
  // process response
};
```

Server Response

A property of the object called status is then used to make sure the data loaded okay (more on this later):

```
xhr.onload = function() {
  if (xhr.status == 200) {
    // process response
  }
};
```

Data Formats: HTML

HTML is the simplest way to get data into a page:

```
<div class="event">
  <img src=“img/map-ny.png"
       alt="New York, NY” />
  <p>New York, NY<br>
  <span id="ajaxInfo"></span></p>
</div>
```

Data Formats: HTML

To get the response use responseText or responseXML

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

responseText syntax

```
document.getElementById("ajaxInfo").innerHTML = xhr.responseText;
// $("ajaxInfo").text(xhr.responseText);
var xmlDoc = xhr.responseXML; - You need to write JavaScript to convert
                                the XML data into HTML so it can be displayed
```

responseXML

Need to parse the response as an XML object

```javascript
xmlDoc = xhr.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
  txt += x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("demo").innerHTML = txt;
```

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
      <CATALOG>
      <CD>
            <TITLE>Empire Burlesque</TITLE>
            <ARTIST>Bob Dylan</ARTIST>
            <COUNTRY>USA</COUNTRY>
            <COMPANY>Columbia</COMPANY>
            <PRICE>10.90</PRICE>
            <YEAR>1985</YEAR>
      </CD>
</CATALOG>
```

Back to Server Response:
The onreadystatechange Event

Need to perform actions based on server response

The onreadystatechange event is triggered every time the readyState changes.

The readyState property holds the status of the XMLHttpRequest.

onreadystatechange Event

Three important properties of the XMLHttpRequest object:

Property	Description
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 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

onreadystatechange Event

- We specify what will happen when the server response is ready to be processed.
- When readyState is 4 and status is 200, the response is ready:

```
function loadDoc() {
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 && xhr.status == 200) {
     document.getElementById("demo").innerHTML = xhr.responseText;
     // $("demo").text(xhr.responseText);
  }
};
```

Callback Function

- Function passed as a parameter to another function.

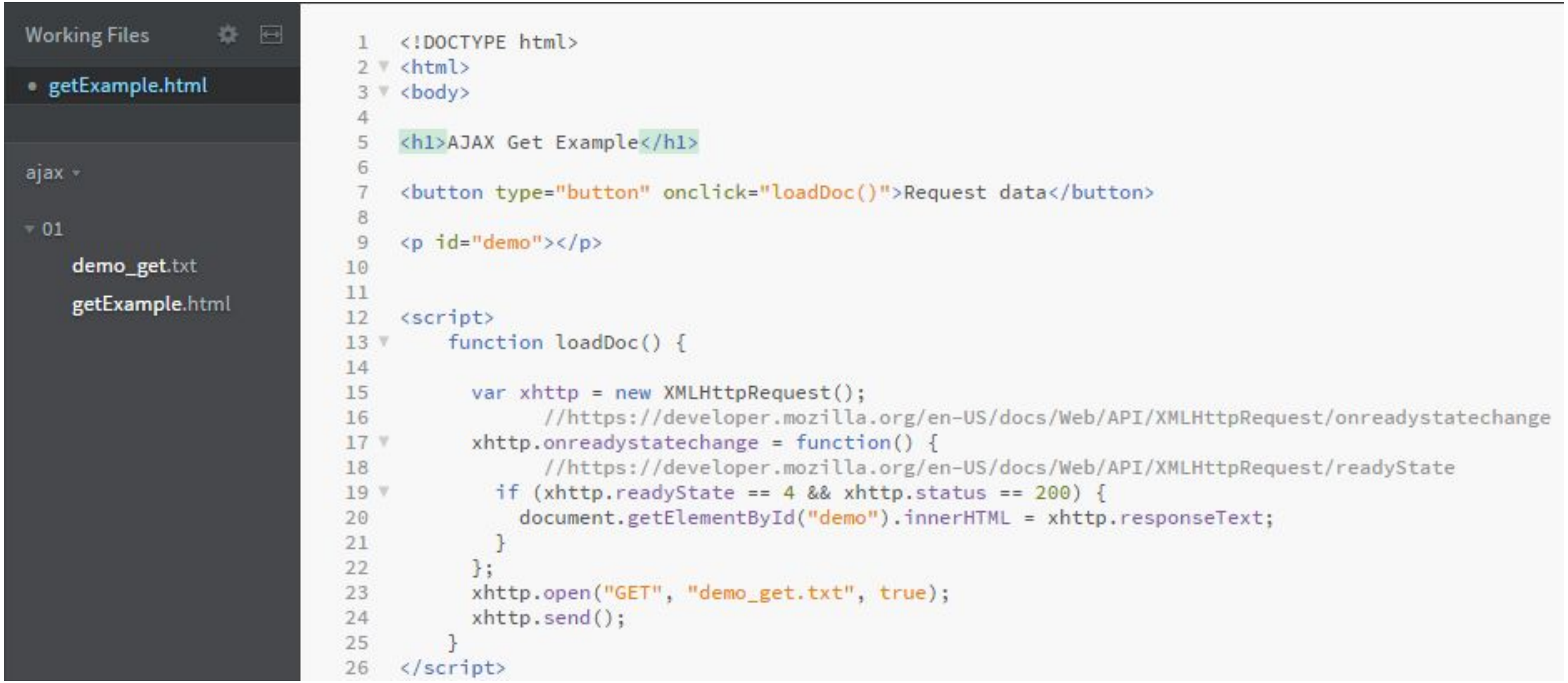
- If you have more than one AJAX task on your website, you should create ONE standard function for creating the XMLHttpRequest object, and call this for each AJAX task.

- The function call should contain the URL and what to do on onreadystatechange (which is probably different for each call).

Callback Function Example

```
function loadDoc(cFunc) {
  var xhr = new XMLHttpRequest();
  xhr.onreadystatechange = function() {
  if (xhr.readyState == 4 && xhr.status == 200) {
     cFunc(xhr);
  }
```

Get example



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

<h1>AJAX Get Example</h1>

<button type="button" onclick="loadDoc()">Request data</button>

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


<script>
    function loadDoc() {

      var xhttp = new XMLHttpRequest();
            //https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/onreadystatechange
      xhttp.onreadystatechange = function() {
            //https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState
        if (xhttp.readyState == 4 && xhttp.status == 200) {
          document.getElementById("demo").innerHTML = xhttp.responseText;
        }
      };
      xhttp.open("GET", "demo_get.txt", true);
      xhttp.send();
    }
</script>
```

Steps of AJAX Operation Review

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