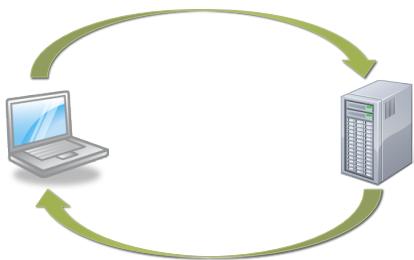
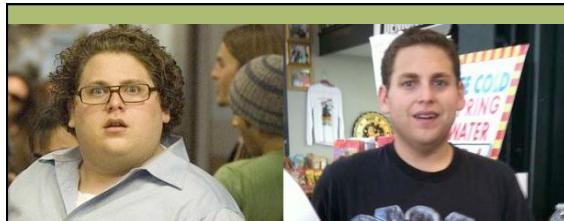


INTRODUCTION TO AJAX

Making Ajax calls at the lowest possible level

Remember that the web is a thin-client technology

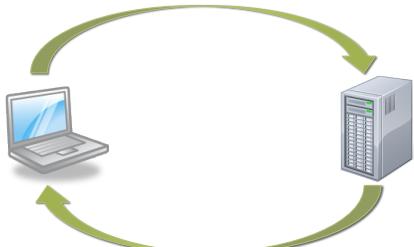




The history of server-side processing is like a yo-yo dieter.



So we clearly need some way to tell the server we want to send data to it and a way to receive data from it.



If only we could send a request while the page remains loaded and displayed

- Precondition – a page is loaded
- 1. The page sends a request for some data
- 2. The page listens for a response while the user continues to work (multithreading)
- 3. The page receives a response
- 4. It runs a callback function which usually processes the data and injects it into the DOM
- Postcondition – the same page is loaded but has new data

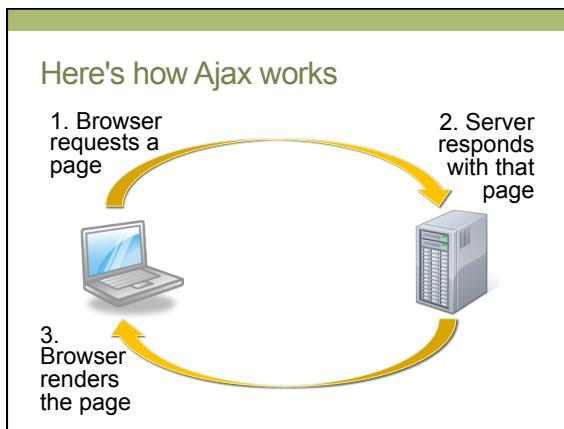
So we use JavaScript to send an XML request asynchronously and process the XML response

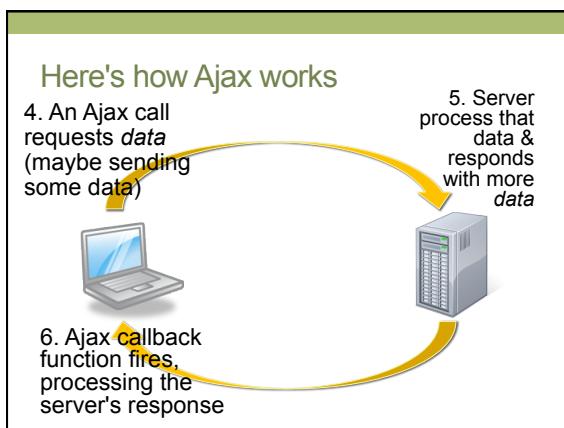
We call this technology ...

JXAA!

No, wait ...

Ajax





The key is the XMLHttpRequest* object

Methods	Properties	Events
open(httpMethod, url, async)	readyState	onreadystatechange
setRequestHeader(header, value)	status	
send(string)	statusText	
getResponseHeader(header)	responseText	
abort()	responseXML	

*The XMLHttpRequest object is often abbreviated "xhr"

The open() method prepares the Ajax call

```
xhr.open(httpMethod, url, async)
• where
  • httpMethod = "GET" or "POST" or "PUT" or "DELETE"
  • url = address we're hitting
  • async = true for asynchronous (default is true)
```



setRequestHeader() can inject values into the request

```
• Looks like a form post to the server
xhr.setRequestHeader(header, value)
```

send() instructs the system to make the actual call

```
• send() usually has no arguments
xhr.send();
• You can pass in form/querystring data using send ...
  xhr.send('Artist=CeeLo');
```



Are we ready? readyState knows!

- xhr.readyState holds a number between 0 and 4

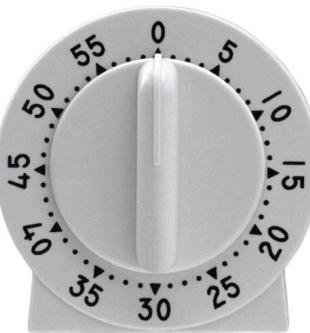
Value	Meaning
0	Request not initialized
1	We are connected to the server
2	The request was received
3	The server is processing the request
4	Request is finished and the response has been sent back

The xhr.status is like the http response state

- 100 Informational only
- 200 OK
- 300 Moved
- 400 Not found
- 500 Server error



So, when
xhr.status is
200 and
xhr.readyState
is 4, we know
that the request
is ready



readystatechange
points to a function
that fires when the
readyState changes

- We can use this to determine when a response comes back from the server.
- Then we test it to see if readyState=4 and status=200



Next, we process the response

xhr.responseText

- For JSON

xhr.responseXML

- For XML

But `responseText` is just a string

And we need it to be an object.

Converting JSON to a string and back

- Client to server
 - object to string
 - var string = JSON.stringify(object);
- Server to client
 - string to object
 - var object = JSON.parse(stringfromServer);



Let's put it all together in a very basic sample

First, let's create a page with all the hooks built in

```
<html>
<head>
  <script src='SayHello.js'></script>
</head>
<body>
  <input id="personId" />
  <button onclick='getPerson()'>Go!</button>
  <div id='pDiv'>
    <p>Hello there!</p>
  </div>
</body>
</html>
```

Next, we'll create the server-side program

```
<?php
$person= get_person($_GET['personId']);
echo($person);
?>
```

Then we write the call to the service

```
function getPerson() {
    var textbox =
        document.getElementById('personId');
    var personId = textbox.value;
    xh = new XMLHttpRequest();
    xh.open('GET','GetPerson.php', true);
    xh.addEventListener('readystatechange',
        changePersonInfo);
    xh.send('personId=' + personId);
}
```

And we write the callback function

```
function changePersonInfo() {
    if (xh.readyState==4 && xh.status==200) {
        var p = JSON.parse(xh.responseText);
        var d = document.getElementById('pDiv');
        d.innerHTML = `<p>Person: ${p.firstName}
                      ${p.lastName}</p>`;
    }
}
```

