Introduction to Computer Science I
COMP 2406A – Winter 2020

AJAX – Asynchronous Javascript and XML

Dave McKenney
david.mckenney@carleton.ca

Learning Outcomes

by the End of this Lecture, Students that have Completed the Reading Assignment and Review Questions should be Able to:

Use AJAX to make asynchronous client-side requests

Use AJAX to perform partial page updates

AJAX

AJAX – Asynchronous Javascript and XML (though often, JSON now instead of XML)

Allows asynchronous requests for data to be made from the client-side

Allows us to get SOME data without getting the ENTIRE page again (minimizing data transfer)

AJAX

A combination of technologies allow AJAX to work:

HTML/CSS

Document Object Model (DOM)

JSON/XML

Javascript

XMLHttpRequest (XHR)

AJAX History

In the 90's, any time new data was needed, a new request would be made and an entire new page would be sent back

Microsoft first implemented a way to fetch content asynchronously in Internet Explorer

This idea was implemented within other browsers and became the Javascript XMLHttpRequest object

AJAX History

Google started to make wide use of AJAX in the early/mid 2000s (search suggest, Gmail, etc.)

Kayak.com also made wide use of “the xml http thing” when they first launched in 2004

XHLHttpRequest has now become a W3C standard

http://kayaklabs.blogspot.com/2006/04/kayak-user-interface.html

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send();
```

The XMLHttpRequest Object

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send();
```

This creates a new XMLHttpRequest object

https://www.w3schools.com/js/js_ajax_http.asp

The XMLHttpRequest Object

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send();
```

This defines a function to handle the 'onreadystatechange' event

https://www.w3schools.com/js/js_ajax_http.asp

XMLHttpRequest Ready State Values

The ready state is updated throughout the lifetime of the request. The values may be:

0. **Request not initialized**
1. **Server connection established**
2. **Request received**
3. **Processing request**
4. **Request finished and response ready**

So while we only handled readyState == 4 in the example, you can react to different state changes

https://www.w3schools.com/js/js_ajax_http.asp

The XMLHttpRequest Object

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send();
```

Specifies the request with: a string method, string URL, boolean asynchronous

https://www.w3schools.com/js/js_ajax_http.asp

The XMLHttpRequest Object

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send(); //or xhttp.send(string);
```

Sends the request to the server. Optional argument for request body (e.g., for POST)

https://www.w3schools.com/js/js_ajax_http.asp

The XMLHttpRequest Object

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send();
```

If the request has finished, the response is ready, and the status is 'okay'

https://www.w3schools.com/js/js_ajax_http.asp

The XMLHttpRequest Object

A basic XMLHttpRequest looks something like this:

```
let xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if(this.readyState==4 && this.status==200){
    //Whatever you want to do with the data
    //this.responseText holds response data
  }
};
xhttp.open("GET", "someResource", true);
xhttp.send();
```

this.responseText contains the response data

https://www.w3schools.com/js/js_ajax_http.asp

The XMLHttpRequest Object

Additionally, there are the following methods:

abort(): cancels current request

getAllResponseHeaders() – returns all headers

getResponseHeader(string) – returns specific header

setRequestHeader(header, value) – sets the given request header to the given value

https://www.w3schools.com/js/js_ajax_http.asp

AJAX Demonstration

The names.html file has Javascript to perform an AJAX get request to a server that provides a list of matching names.

The page dynamically updates with matching names as the user types in the textbox.

AJAX Demonstration

We can also dynamically retrieve the restaurant list and menu data from assignment #1

In a scenario where you have many restaurants (or a lot of data in general), it is infeasible to send ALL the data for each request

Instead, we only send the specific menu data the user requires

AJAX Limitations

One limitation of AJAX is that, by default, you can only request information from the same origin

In general, the request host, protocol, and port must be the same as the page making the request

This is called the same origin policy and is an important security measure – why?

AJAX Limitations

Cross-origin resource sharing (CORS) can allow for requests across different origins

The server must support CORS and is responsible for specifying what other origins may access data - this is done using additional HTTP headers

For more information on CORS:

https://www.keycdn.com/support/cors

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Another Note

Performing dynamic web page updates can break things like bookmarking and the 'back' button in the browser

The URL indicates the initial request and not the updated data

If you try to 'go back', you go back to the last page, not back to the last piece of data

Another Note

One approach that has been used to address these issues is setting the fragment marker (#) in the browser via Javascript

This can create a new page entry in the browser

Modern browsers also allow Javascript to update the URL (see https://computerrock.com/blog/html5-changing-the-browser-url-without-refreshing-page/)

Summary

So now, once we create our own servers, we can request data to partially update the page using client-side Javascript

This will decrease the amount of data transferred and allow us to make fully function single-page apps

Questions?