Week 2

XMLHttpRequest Object

This is the way that we make a request to the server for data, and handle the response. The details are different on different platforms, so after learning the details in this section, the next section will use a function provided by the Google maps API to make, send and receive the XMLHttpRequest.

You can find a pretty good definition on wikipedia:
http://en.wikipedia.org/wiki/XMLHttpRequest

Here are the steps required to get data for your application from the server:
I. Create the XMLHttpRequest object
II. Set all its attributes correctly
III. Call send() on the object
IV. Wait for our callback function to be called – this is because of the "asynchronous"
(but in the meantime, all elements of page are still functional!)

The attached program shows the details of these steps. See that the contents of the file we are loading is found in the responseText attribute of the XMLHttpRequest object. If you want to parse the response as XML, you would look at the responseXML attribute of the same XMLHttpRequest object. Every XMLHttpRequest object has both responseXML and responseText attributes. Next week we will treat the response as XML, so that we can more easily get data out of it.

Chapter 3 in "Pragmatic Ajax" shows a slightly different way of creating an XMLHttpRequest object, sending it to the server and receiving the response asynchronously. But what most programmers do is to use a predefined library function to encapsulate these details. In this class, we will use Google's version, called GDownloadUrl().

dumpTextManually.html

Instructions: The file in this module, and those in the next two modules demonstrate the details of creating, sending and receiving an XMLHttpRequest object: dumpTextManually.html, dumpTextManually.js, data

dumpTextManually.html

dumpTextManually.js

Instructions: These three files demonstrate the details of creating, sending and receiving an XMLHttpRequest object: dumpTextManually.html, dumpTextManually.js, data

dumpTextManually.js

data

Instructions: These three files demonstrate the details of creating, sending and receiving an XMLHttpRequest object: dumpTextManually.html, dumpTextManually.js, data

data

GDownloadUrl()

Google provides this function as part of its maps API. you can find documentation at:
http://code.google.com/apis/maps/documentation/reference.html#GDownloadUrl

In order to use any part of the Google API, you have to register and get a key. This key must appear in a script tag in

any program where you use functions from the Google API. This key must also be associated with your computer/server. You can register and get a key at:
http://code.google.com/apis/maps/documentation/

When you upload a program to a public webserver, you must change the key so that your program can call GDownloadUrl() from the new server. The key for http://ctislab.fhda.edu is:

ABQIAAAAKfSIpqvYw0 6ct-cn8YFARQd 2 2 Gr f 9 IKmP GUi 3 sckVmP HWTBWkTzscbD3 Epe 9 l 1 _9 t0 HqDRP qFNIpv

The following section has a program that uses GDownloadUrl() to create the XMLHttpRequest object, send it to the server, and receive the response. Pretty simple, isn't it? Note that our callback function won't be called this time until the content is fully loaded. That is because GDownloadUrl() traps all of the readyStates that do not indicate completion.

GDownloadUrl() can only be used to load data from URLs that are on the same server as the html document that needs the data. This is because most browsers do not allow what is called "Cross Site Scripting" for security reasons. You can read more about cross site scripting at wikipedia:
http://en.wikipedia.org/wiki/Cross-site_scripting
What it means for us is that our data must always reside on the same file server as our source code.

dumpTextWithGDownloadUrl.html

Instructions: These three files demonstrate the use of GDownloadUrl() to get data from the server: dumpTextWithGDownloadUrl.html, dumpTextWithGDownloadUrl.js, data (given earlier in this module)

dumpTextWGDownloadUrl.html

dumpTextWithGDownloadUrl.js

Instructions: These three files demonstrate the use of GDownloadUrl() to get data from the server: dumpTextWithGDownloadUrl.html, dumpTextWithGDownloadUrl.js, data (given earlier in this module)

dumpTextWGDownloadUrl.js

Pull down menus

Recall from your html that a pull down menu is made by a <select> tag with multiple <option> tags nested inside it. These nodes have additional variables available to your JavaScript program. Let's say you have a select tag in your html file that has id "selectWeather". If selectNode is a reference to that node in the DOM:

```
var selectNode = document.getElementById("selectWeather");
```

Then you have the following additional information avaiable to your program:
1) selectNode.options is an array of option nodes that are nested under the selectNode
2) selectNode.options.length contains the number of option nodes nested inside the selectNode.
3) You can set

```
selectNode.options.length = 0
```

if you want to clear out all the option nodes from the select node.
4) selectNode.selectedIndex is a variable containing the number of the currently selected option in the pull down menu. Note that indexes always start with 0!
5) selectNode.value contains the value of the selected option in the pull down menu.

The program in the next section demonstrates all of these capabilities.

optionArray.html

Instructions: Demonstrates additional API's for handling option and select elements from your JavaScript program.

optionArray.html

Review Arrays in JavaScript

We will be using arrays a lot in this course, so you must be familiar with how they work in JavaScript.

There are lots of ways to create arrays in JavaScript:

```
var a = doc.getElementsByTagName("div");    // this function returns an array
var array = new Array(10);
var array = new Array();
var array = [4, 99,123];
```

Remember that arrays always start with index 0