INTRODUCTION TO AJAX AND JAVASCRIPT FRAMEWORKS

Chapter 10 and Online references

Note

- Examples for this chapter are at
- https://swe.umbc.edu/~zzaidi1/is448/chap10-examples/
- You will be using a combination of pre-written PHP programs, HTML files, Javascript files, and CSS in this chapter. When downloading examples for this class, always download the zip file for each example, that is in the examples folder.
  - This zip file will contain all the relevant files for the example
  - A zip file is created particularly for getting access to the PHP files, because you cannot view the PHP file in the browser, and 'view-source' and save the file to your desktop
  - You cannot right-click on a PHP file and save it to your computer either
  - Because PHP files are server-side programs, when you view them or right click and save them, you will only be saving the Output of execution of the PHP program, and not the actual code of the PHP file

What is Ajax?

- Ajax: Asynchronous JavaScript + XML
  - not a programming language
  - it's a way of using JavaScript
  - a way to download data from a web server without reloading the page

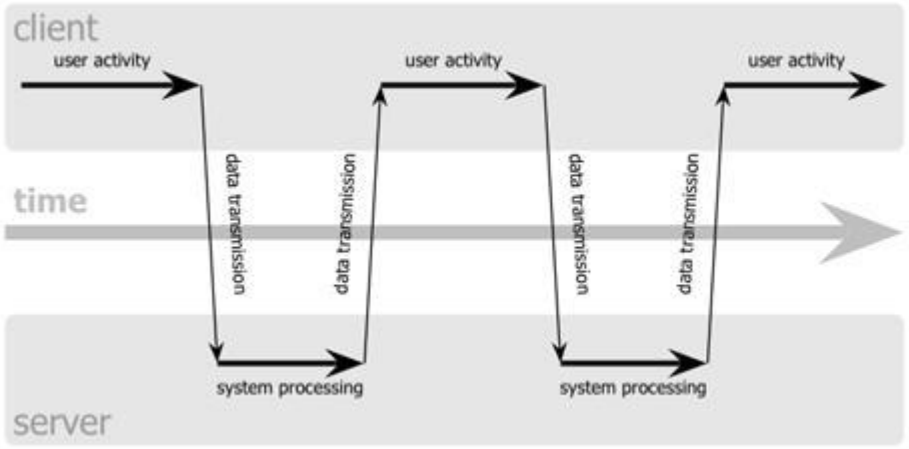
Synchronous vs. Asynchronous

- In normal request/response HTTP cycles, on issuing a request for a page, the browser locks waiting for the response and an entire page must be displayed

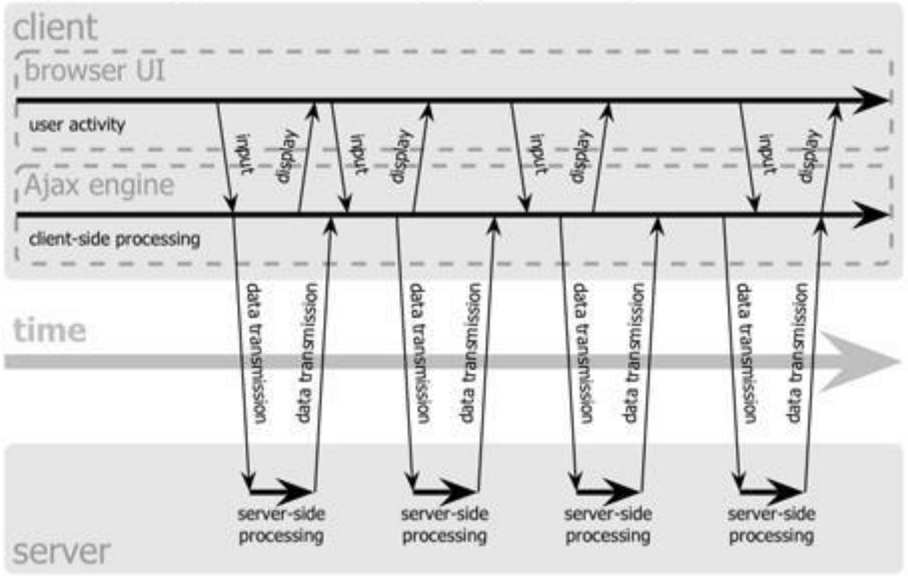
- With Ajax, asynchronous requests may be made and responses are used to update part of a page
  - User can continue to interact with the page while the request is in progress
  - Less data needs to be transmitted
  - Page update is quicker because only a part of a page is modified

Synchronous vs. Asynchronous

classic web application model (synchronous)



Ajax web application model (asynchronous)



What is Ajax?

- allows dynamically displaying data or updating the page without disturbing the user experience
- aids in the creation of rich, user-friendly web sites
- examples: Google Suggest, Facebook, Flickr, A9

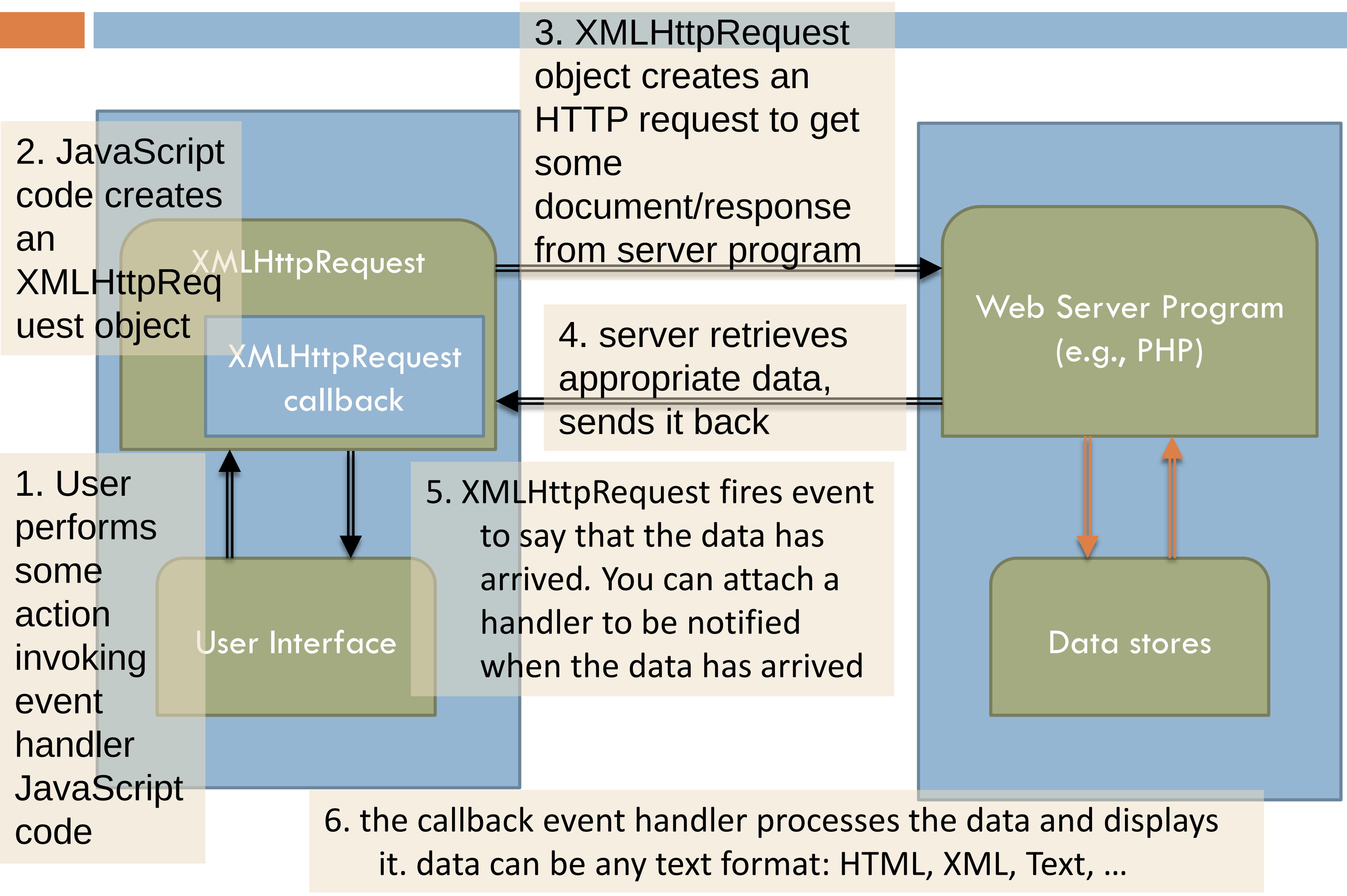
What is Ajax?

- many web apps use Ajax to battle these problems of web pages:
  - slowness/lack of UI responsiveness
  - lack of user-friendliness
  - jarring nature of "click-wait-refresh" pattern

Core Ajax Concept

- JavaScript's XMLHttpRequest object can fetch files from a web server
  - supported in IE5+, Safari, Firefox, Opera (with minor compatibilities)
- it can do this asynchronously (in the background, transparent to user)
  - synchronous: user must wait while new page loads
  - asynchronous: user can keep interacting with page while data loads
- contents of fetched file can be put into current web page using DOM
- result: user's web page updates dynamically without a page reload

Typical Ajax request



1. User performs some action invoking event handler JavaScript code

2. JavaScript code creates an XMLHttpRequest object

3. XMLHttpRequest object creates an HTTP request to get some document/response from server program

4. server retrieves appropriate data, sends it back

5. XMLHttpRequest fires event to say that the data has arrived. You can attach a handler to be notified when the data has arrived

6. the callback event handler processes the data and displays it. data can be any text format: HTML, XML, Text, …

Sending asynchronous requests

```
var ajax = new XMLHttpRequest();

function foo(){
      ajax.onreadystatechange = function_with_code;
      ajax.open("get", url, true);
      ajax.send(null);
}


function function_with_code(){
     //do something in response to
        //ajax request being answered
        //by web server
}
```

Create new XMLHttpRequest object

Attach an event handler function, *function_with_code* to the *onreadystatechange* property of XMLHttpRequest object

Open a connection to the web server. First parameter is GET or POST, second parameter is name of PHP program that will handle request, and last parameter set to 'true' creates an asynchronous request

Send the request to server. Response from server if in text form is stored in *responseText* property of XMLHttpRequest object

Sending asynchronous requests

```
1.   var ajax = new XMLHttpRequest();
function some_javascript_function(){
        2.   ajax.onreadystatechange = function_name;
        3.   ajax.open("get", url, true);
        4.   ajax.send(null);
}
```

```
function function_name(){
   //do something in response to
   //ajax request being answered
   //by web server
}
```

- Basic idea
  1. create the request object, by name ajax
  2. attach an event handler function, function_name to the onreadystatechange property
     - function_name contains code to run when request is complete
     - Note: the event handler function is not invoked at this statement, instead the address of the function is assigned to the onreadystatechange property.
     - Event handler function will be called several times whenever request state changes
     - As the server is processing the request, the server will change/set the readyState property of the XMLHTTPRequest object
  3. open a connection: makes the necessary arrangements for the server request
  4. send the request: this is when the request is actually sent to the server
     - when send returns, the fetched text will be stored in request object's responseText property

The readyState property

- holds the status of the XMLHttpRequest
- is set several times by the web server when the server is processing the request
- possible values for the readyState property:

State	Description
0	not initialized
1	set up
2	sent
3	in progress
4	complete

- whenever readyState changes → onreadystatechange handler runs
- usually we are only interested in readyState of 4, because this implies that the request is completely processed by the server, and some response from the server is available

Complete Ajax code: v1

- Using a named function as event handler

```
var ajax = new XMLHttpRequest();  //global variable
//request function
function some_javascript_function(){
                ajax.onreadystatechange = another_function_name;
                ajax.open("get", url, true);
                ajax.send(null);
}
//receiver function
function another_function_name(){
                if (ajax.readyState == 4){ //4 means request finished
                                //do_something with ajax.responseText
                                alert(ajax.responseText);
                }
}
```

Complete Ajax code: v2

- Use an anonymous function as the event handler
- Useful to declare it as an inner anonymous function, because then it can access surrounding local variables (e.g. ajax)

```
//request function
function some_javascript_function(){
		var ajax = new XMLHttpRequest();
		ajax.onreadystatechange = function() {//receiver function
		  if (ajax.readyState == 4) {   // 4 means request is finished
				//do something with ajax.responseText
				 alert(ajax.responseText);
		  }
		};
		ajax.open("get", url, true);
		ajax.send(null);
}
```

Cross browser compatibility

- Need to add this piece of code in all your Javascript files, to allow cross-browser compatibility
- A mess!
- Instead, we will use a JavaScript framework in this class, which already implements all these cross-browser compatibility issues and has many other features
- Frameworks will give you a clean interface to use for your JavaScript/Ajax code

```
function ajaxFunction() {
        var ajax;
        try {
           // Firefox, Opera 8.0+, Safari
           ajax=new XMLHttpRequest();
        }
        catch (e)
        {
            // Internet Explorer
            try {
                ajax=new ActiveXObject("Msxml2.XMLHTTP");
            }
            catch (e)
            {
                try {
                 ajax=new ActiveXObject("Microsoft.XMLHTTP");
                }
                catch (e) {
                  alert("Your browser does not support AJAX!");
                  return false;
                  }
              }
        }
}
```

JavaScript Frameworks

- Collection of JavaScript libraries that allow for easy JavaScript development
- Several frameworks available
  - Dojo
  - Ext JS
  - jQuery
  - MochiKit
  - Mootools
  - Prototype
  - Script.aculo.us

Prototype Framework

- Prototype JavaScript library
  - A set of useful additional objects, methods, and cross-browser compatibility fixes
- Documentation
  - API: http://www.prototypejs.org/api
  - Tutorials: http://www.prototypejs.org/learn

To use Prototype in your JavaScript files

- Include the following line in the <head> element of your HTML page
- This line is in addition to all the other script tags and css references that you have in the <head> of your page

```
<script type="text/javascript" src=" https://ajax.googleapis.com/ajax/libs/prototype/1.7.3.0/prototype.js"></script>
```

Prototype

- Prototype JavaScript library adds many useful features to JavaScript:
  - many useful extensions to the DOM
  - added methods to String, Array, Date, Number, Object
  - improves event-driven programming
  - many cross-browser compatibility fixes
  - makes Ajax programming easier

Basic Prototype features

- '$' function returns the DOM object representing the element with a given id, e.g., *uname*

Use this:

```
$(“uname")
```

JavaScript using Prototype

Instead of:

```
document.getElementById(“uname”)
```

Traditional JavaScript

Prototype’s Ajax model

- Prototype's Ajax.Request object constructor accepts 2 parameters:
  - the **URL** of the request, as a String,
  - a set of **options**, as an array of *key:value* pairs in {} braces
- hides some of the details of Ajax (XMLHttpRequest, onreadystatechange, etc.)

```
new Ajax.Request
( "url",
{ option : value, option : value, ... option : value }
);
```

Ajax using Prototype

- Options that can be passed to the Ajax.Request constructor:
  - **method** : how to fetch the request from the server ("GET" or "POST", default is "POST")
  - **parameters** : query parameters to pass to the server, if any
  - asynchronous (default true), contentType, encoding, requestHeaders
- Events in the Ajax.Request object that you can handle:
  - **onSuccess** : request completed successfully
  - **onFailure** : request was unsuccessful
  - onCreate, onComplete, onException, on### (handler for HTTP error code ###)

URLs and web servers

- Usually when you type a URL in your browser:

  http://server/path/file
  - your computer looks up the server's IP address using DNS
  - your browser connects to that IP address and requests the given file
  - the web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its contents to you

- Some URLs actually specify *programs* that the web server should run, and then send their output back to you as the result: http://www.google.com/search?*q=colbert&ie=utf-8*
  - the above URL tells the server www.google.com to run the program search with certain parameters

Query Strings

http://www.google.com/search?*query=obama&encoding=utf-8*

- Query string: a way of encoding parameters into a URL

  http://server/path/program?*query_string*

- A query string has the following format:
  `parameter1=value1&parameter2=value2...`
  - preceded by a '?', name=value pairs separated by '& '
  - the above URL runs the program search, with
    - query-string-parameter-name `query` set to query-string-parameter-value `obama`, and
    - query-string-parameter-name `encoding` set to query-string-parameter-value `utf-8`
    - the program outputs the HTML search results

Example

http://www.getState.php?zip="21250"&city="baltimore"

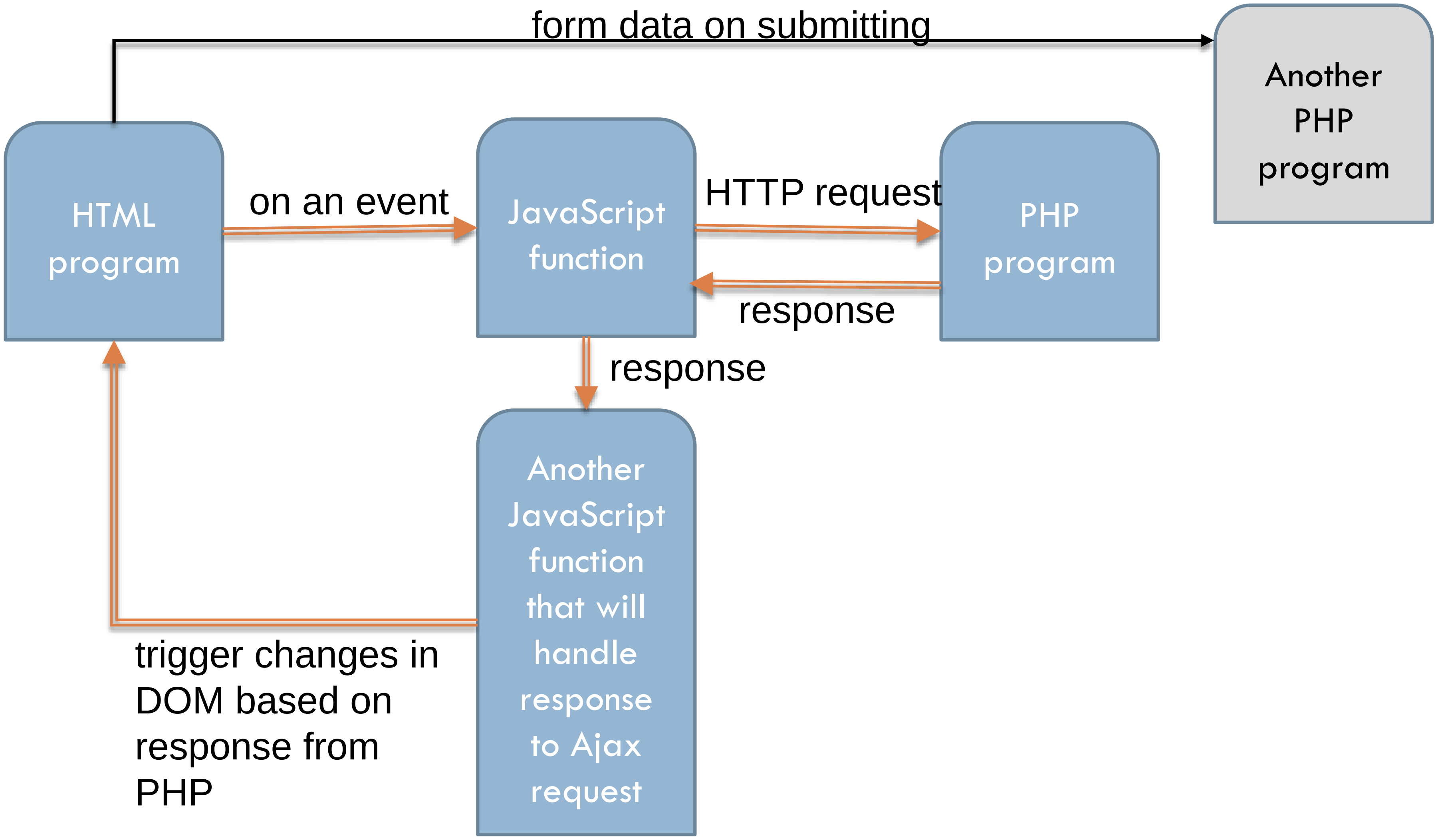
Query string parameter name(s): ____________

Query string parameter value(s): ____________

Example

- See urban.html, urban.php, urban.css, urban.js
- urban.php is a php program that has an array matching each abbreviation to a full-form and returns the full-form corresponding to the abbreviation that the user entered
- See how the innerHTML property is updated in the JavaScript file

Programs that you will be writing here



Example

- See popcornA.html
  - Note the onblur event is set to call the event handler getPlace in the .js file
- See popcorn.php
  - this is a .php file
  - for now, understand that the PHP file takes as input the value for ‘zip’ that user enters, looks up the city and state for that zip code, and returns as output the name of the city and state to which the zipcode belongs

More examples

- See suggest.html, suggest.js, getHint.php
- See time.html, time.js, displayTime.php
- See files in post-and-multiple-parameters.zip to learn how to send more than one query string parameter and how to send by a POST request

Ajax files security restrictions

- Cannot be run from a web page stored on your hard drive
- Can only be run on a web page stored on a web server
- You must have your files on GL to see how they work
- You must access the page using your userpages URL and not the local file-system URL

Lab: Starting files

- Download the zip file login-lab.zip
- See the simple HTML form given in login.html
- We want to check if the username a user enters is available or not and display a message in the <span> field indicating the same
- The file avail.php takes care of the backend processing
  - download this file and store it in same directory as login.html
  - the avail.php file takes the username sent in the query, checks if username exists in the pre-defined array and echoes the string ‘taken’ or ‘available’

Lab: Part 1

- Augment login.html as follows
  - add a link to Prototype's library in the head
  - the JavaScript file, check.js, in which you will be adding Ajax code is already linked for you in the html
- Create the file that will contain your JavaScript code, check.js
- In check.js, do the following:
  - add the code to call the event handler function validateUsername on the onblur event for the username textbox
  - You will be defining the validateUsername function in the JavaScript file, check.js in the next part of this lab

Lab: Part 2

- In check.js, continue to do the following:
- Define the function validateUsername in check.js using Prototype's Ajax request model
  - create a new Ajax request, where
    - URL is avail.php,
    - query-string-parameter-name is name – this should match with what the php program is expecting in line 12 of PHP program
    - query-string-parameter-value is the value that was passed as a parameter to the function
    - And for the onSuccess event, associate it with a new function-name, called displayResult that you will need to define next
  - Define your displayResult function, to display the response returned by the PHP program, in the content of the <span> tag
    - note: you will need to use the innerHTML property to change the content of the <span> tag
    - refer to the tutorial at http://www.tizag.com/javascriptT/javascript-innerHTML.php to see how innerHTML is used
- View the html page in the browser and see the Ajax at work!
  - Note: to see the Ajax work, all your files must be on GL and you should view the HTML file through your userpages URL, i.e., *http://userpages.umbc.edu/~yourusername/directories_to_files_for_this_lab/login.html*