- Defining principles of AJAX

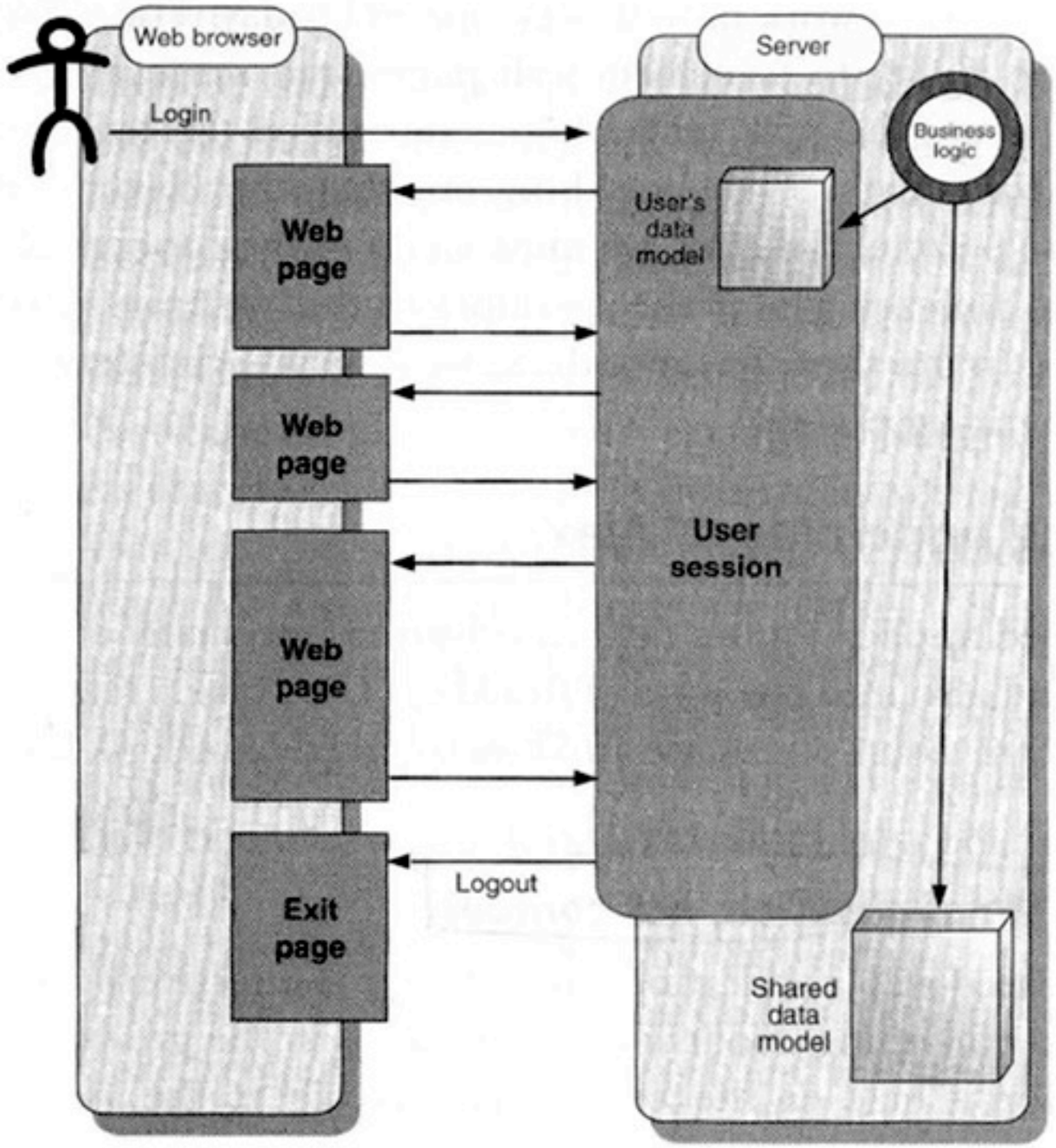
1. Browser hosts an application, not content
2. Server delivers data not content
3. User interaction with the application can be fluid and continuous
4. This is real coding

1. Browser hosts an application, not content

- Static Web model
  - every page is new content

    vs
- Real-time Web model
  - download a program at first
  - every page is new data
- Some server functionality is moved to browser
  - example, the shopping basket is in the client

1. Browser hosts an application, not content



HTML

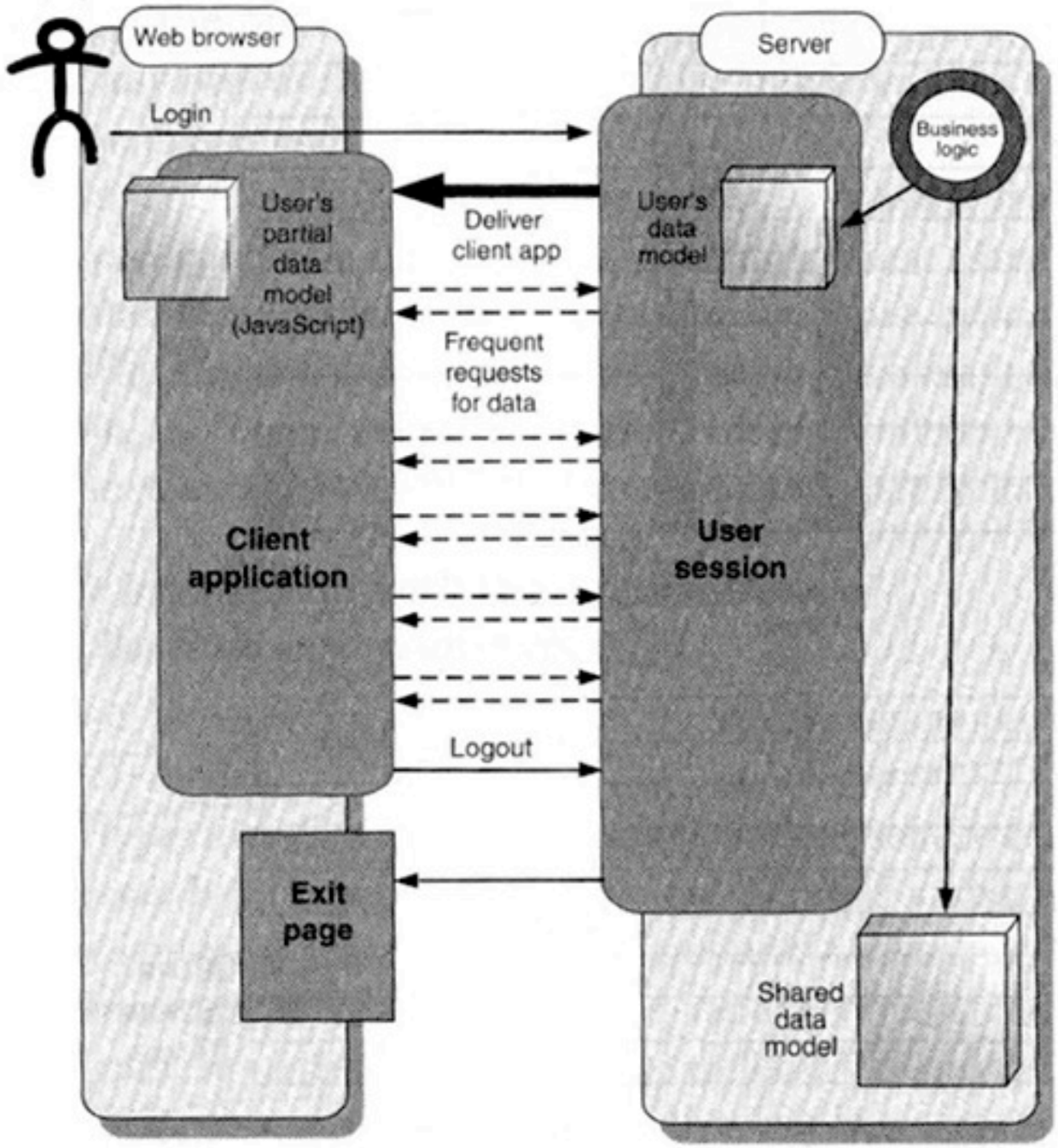
HTML

HTML

HTML

AJAX in action:Crane

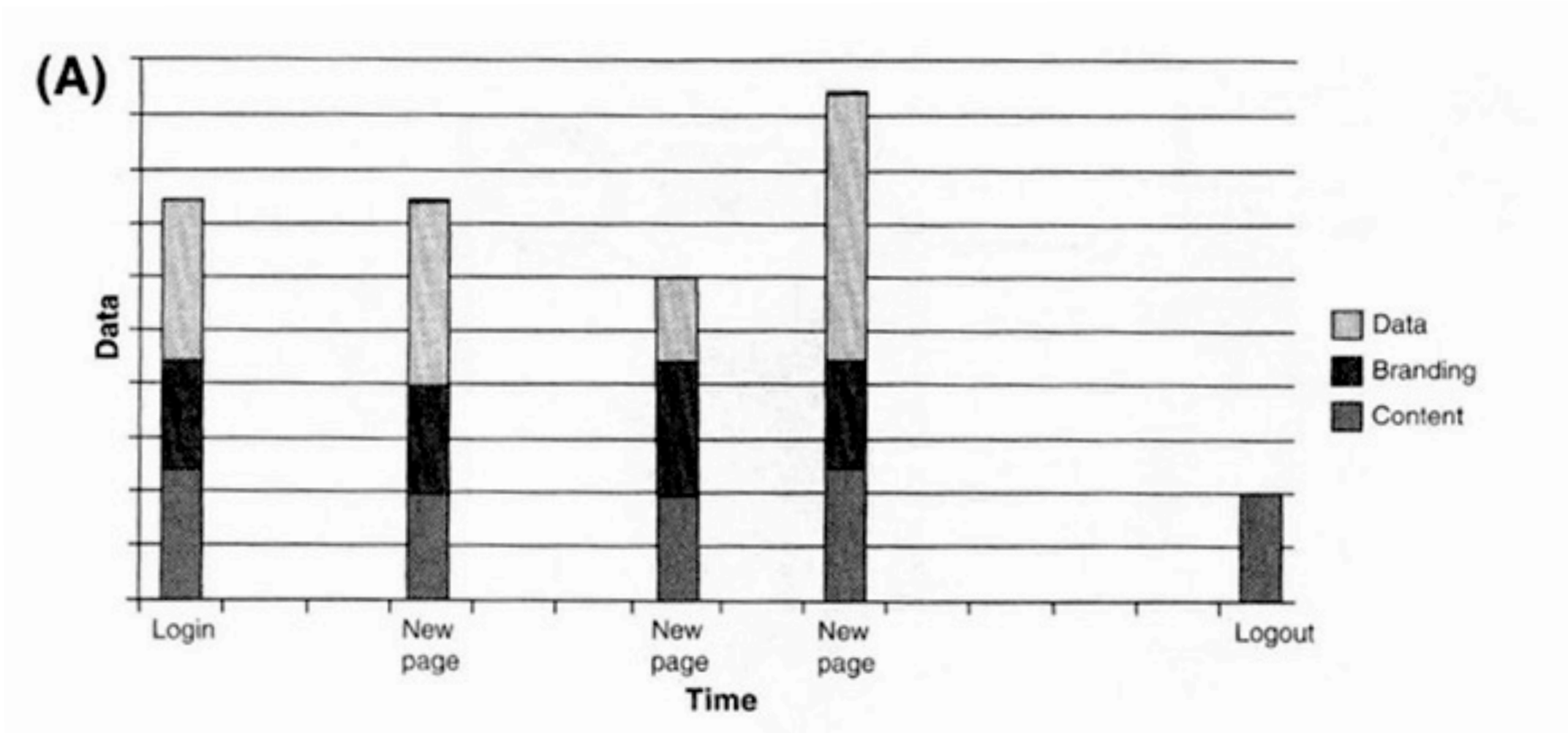
- Browser hosts an application, not content



HTML
Javascript
XML
XML
XML
XML
XML

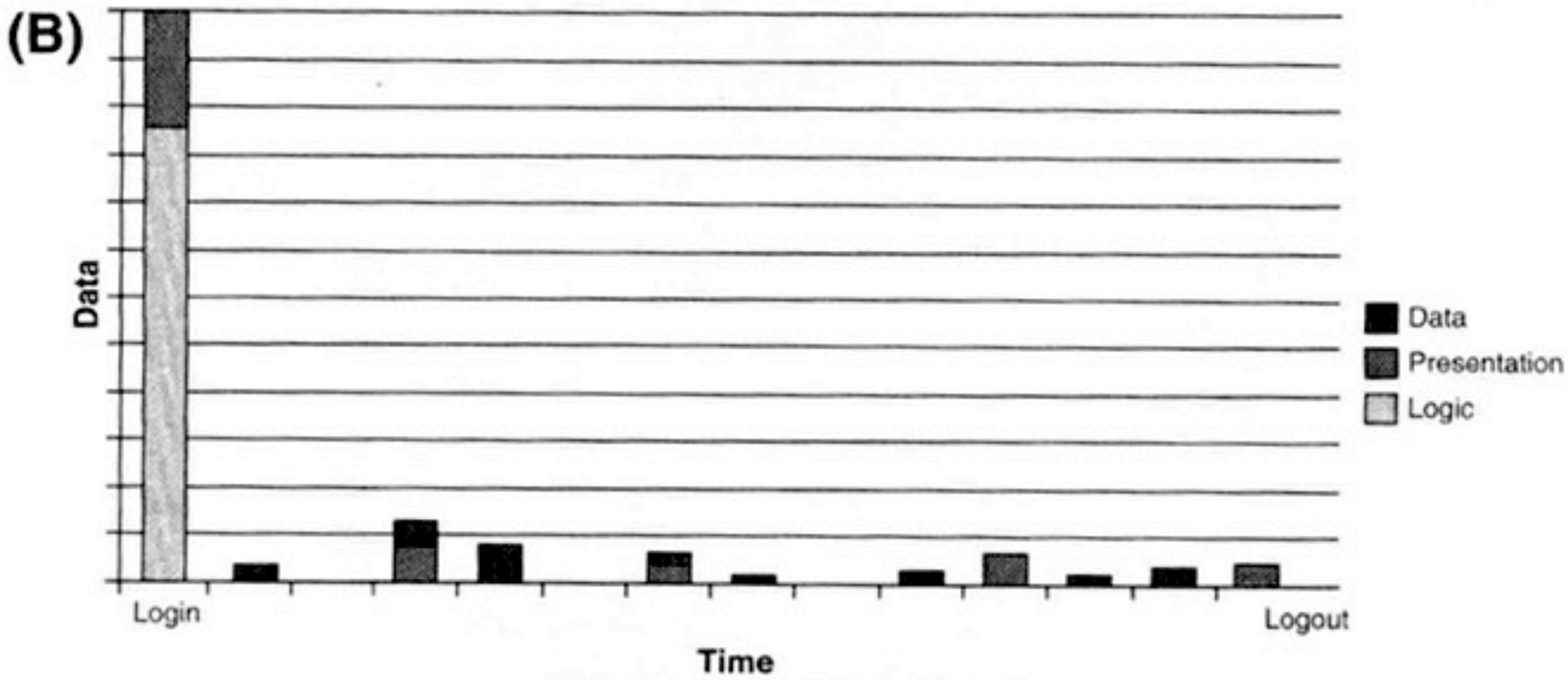
HTML

2.Server delivers data not content



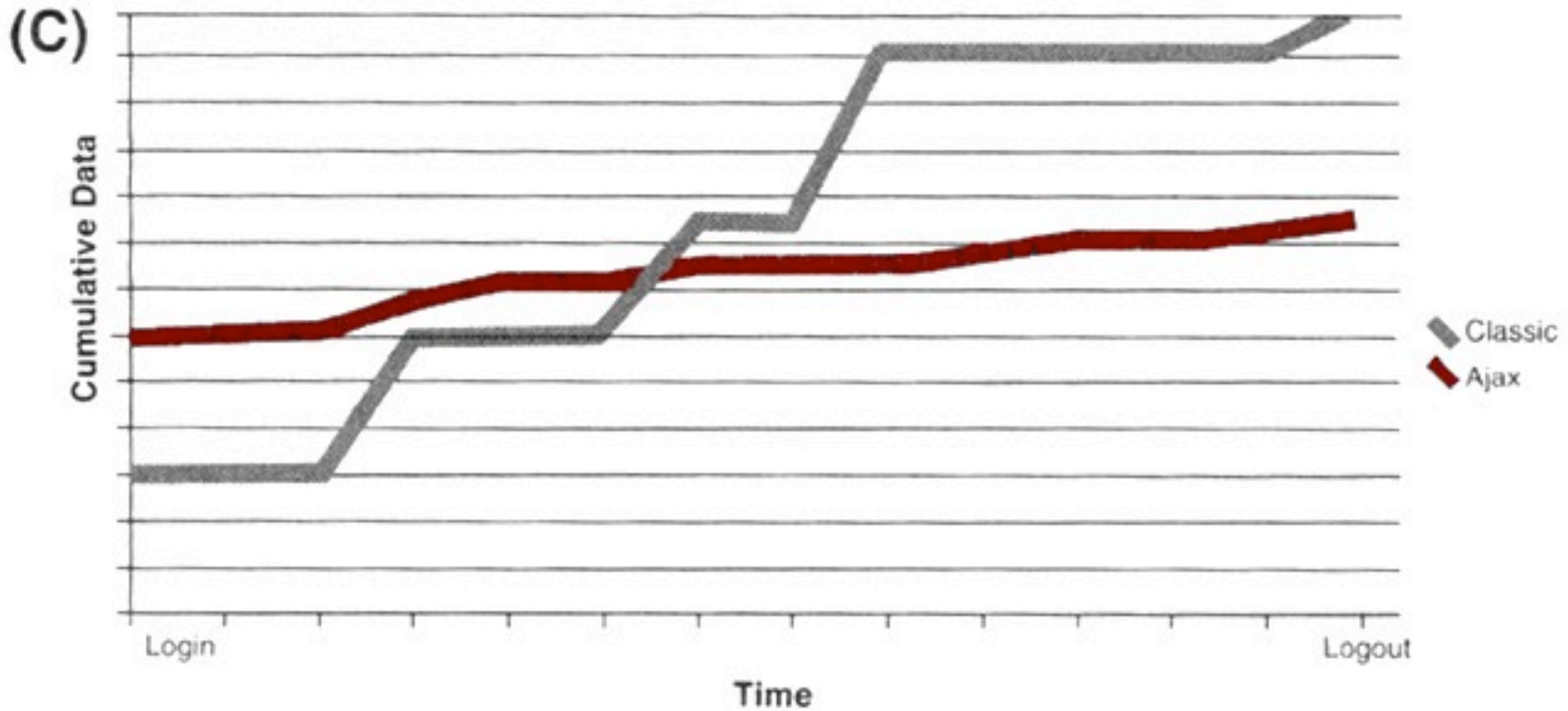
Web 1.0

2.Server delivers data not content



Web 2.0

2.Server delivers data not content



3. User interaction with the application can be fluid and continuous

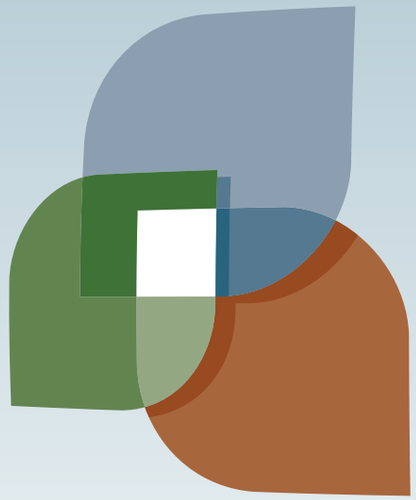
- Typically when a page is submitting data, the user is in limbo
  - Use the shopping cart example
  - Google Suggest
- Sovereign versus Transient Applications

4.This is real coding

- jQuery (http://jquery.com)
- Angularjs (http://angularjs.org/)
- Backbonejs (http://backbonejs.org/)
- emberjs (http://emberjs.com)
- Prototype (http://www.prototypejs.org/)
- ExtJS (http://www.extjs.com/)
  - very good for prebuilt themes and controls, but not very customizable
- YUI (http://developer.yahoo.com/yui/)
- MooTools (http://mootools.net/) - very compact, much smaller than the others
- Dojo (http://dojotoolkit.org/)

- Some good resources

- http://www.ibm.com/developerworks/views/xml/libraryview.jsp?search_by=XML+processing+in+Ajax



LUCI

User Interaction: Making AJAX

Assoc. Professor Donald J. Patterson
INF 133 Fall 2013

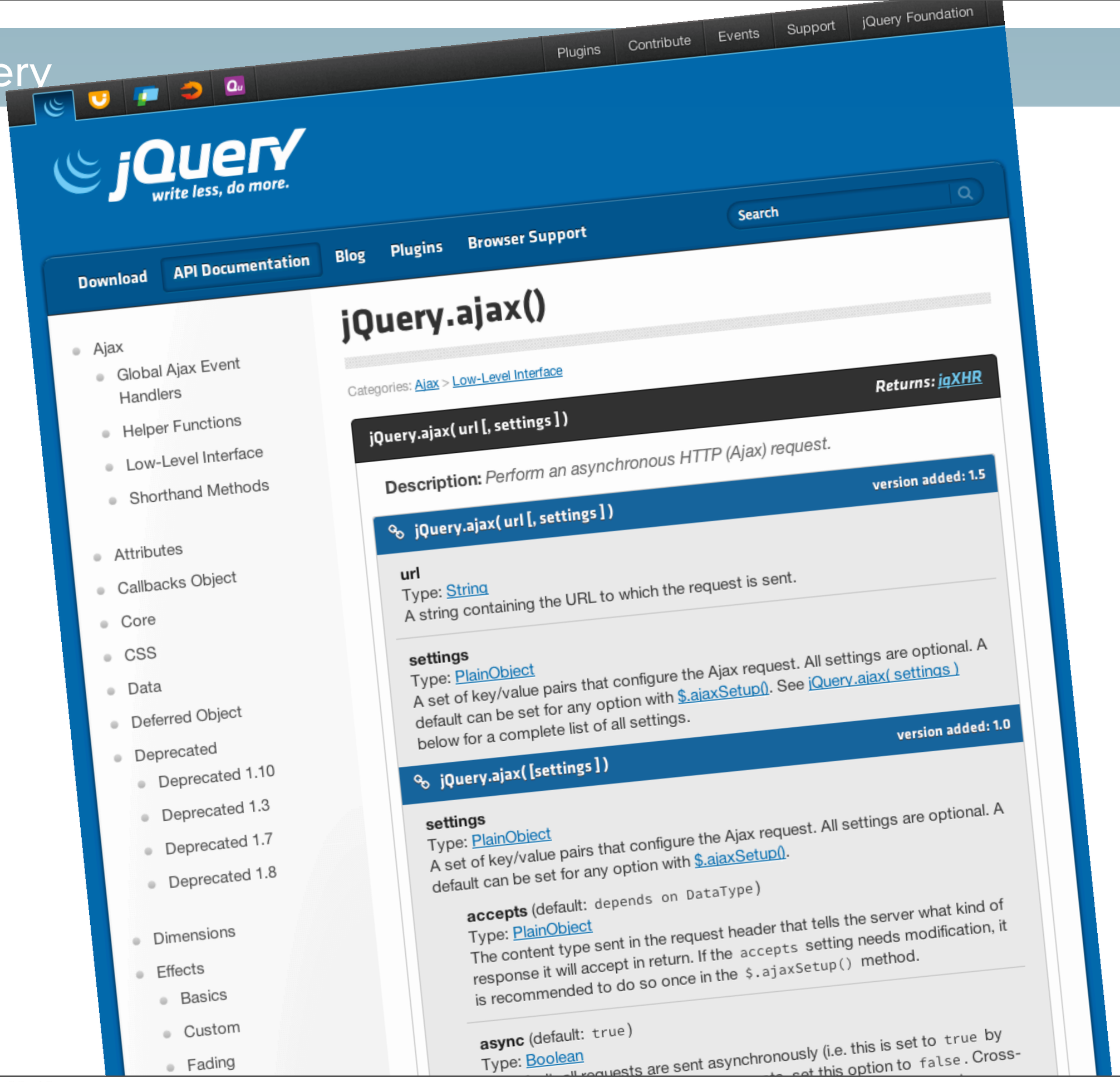
```
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
        <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.9.0/jquery-ui.min.js"></scri
        <script src="Step10.js"></script>
    </head>

    <body>
        <div class="dataHere">Replace this text!</div>
    </body>
</html>
```

jQuery

```
function myGoodLoadFunction(data) {
    $("div.dataHere").text(data.HELLO);
}

function myBadLoadFunction(XMLHttpRequest,errorMessage,errorThrown) {
    alert("Load failed:"+errorMessage+":"+errorThrown);
}


function myReadyFunction(){
    $.ajax({
        url: "/~djp3/classes/2013_09_INF133/Lectures/Lecture07/test.json",
        dataType: "json",
        success: myGoodLoadFunction,
        error: myBadLoadFunction,
    });
}

$(document).ready(
    function(){
        myReadyFunction();
    }
);
```

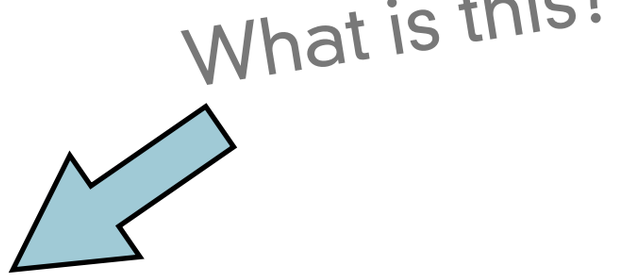
```
{
        "HELLO":"WORLD",
        "This":"is",
        "my":"data"
}
```

jQuery


Plugins
Contribute
Events
Support
jQuery Foundation
jQuery
write less, do more.
Search
Download
API Documentation
Blog
Plugins
Browser Support
Ajax
Global Ajax Event Handlers
Helper Functions
Low-Level Interface
Shorthand Methods
Attributes
Callbacks Object
Core
CSS
Data
Deferred Object
Deprecated
Deprecated 1.10
Deprecated 1.3
Deprecated 1.7
Deprecated 1.8
Dimensions
Effects
Basics
Custom
Fading
jQuery.ajax()
Categories: Ajax > Low-Level Interface
Returns: jqXHR
jQuery.ajax(url [, settings])
Description: Perform an asynchronous HTTP (Ajax) request.
version added: 1.5
jQuery.ajax(url [, settings])
url
Type: String
A string containing the URL to which the request is sent.
settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup(). See jQuery.ajax(settings) below for a complete list of all settings.
version added: 1.0
jQuery.ajax([settings])
settings
Type: PlainObject
A set of key/value pairs that configure the Ajax request. All settings are optional. A default can be set for any option with $.ajaxSetup().
accepts (default: depends on DataType)
Type: PlainObject
The content type sent in the request header that tells the server what kind of response it will accept in return. If the accepts setting needs modification, it is recommended to do so once in the $.ajaxSetup() method.
async (default: true)
Type: Boolean
requests are sent asynchronously (i.e. this is set to true by
set this option to false. Cross-

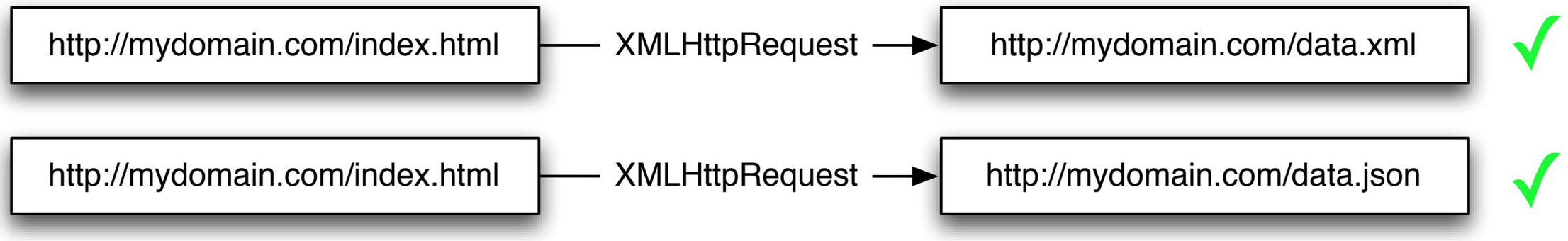
The weeds in the weeds

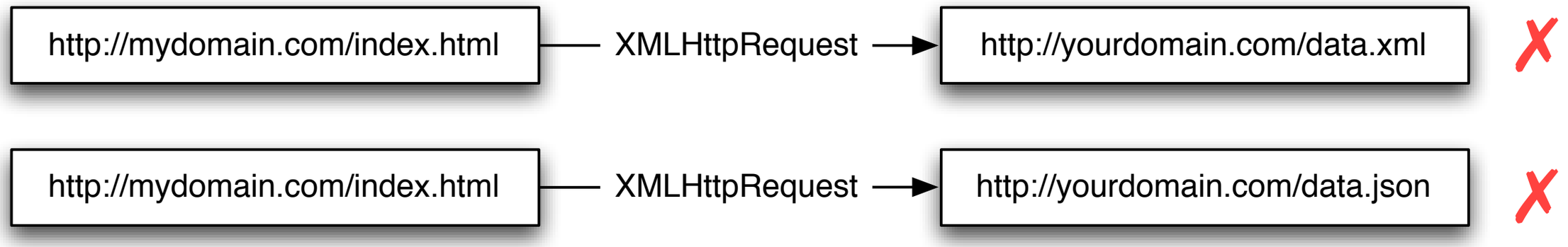
- Two problems:
  - Javascript does not allow you to make an XMLHttpRequest from an external domain
    - XMLHttpRequest is the javascript function which retrieves data asynchronously
      - Not just XML, but any data
      - For security reasons it was restricted
      - It is used by the jQuery .ajax() function
  - An http URL cannot request from an https URL and vice versa



Task 14

The weeds in the weeds





Task 14

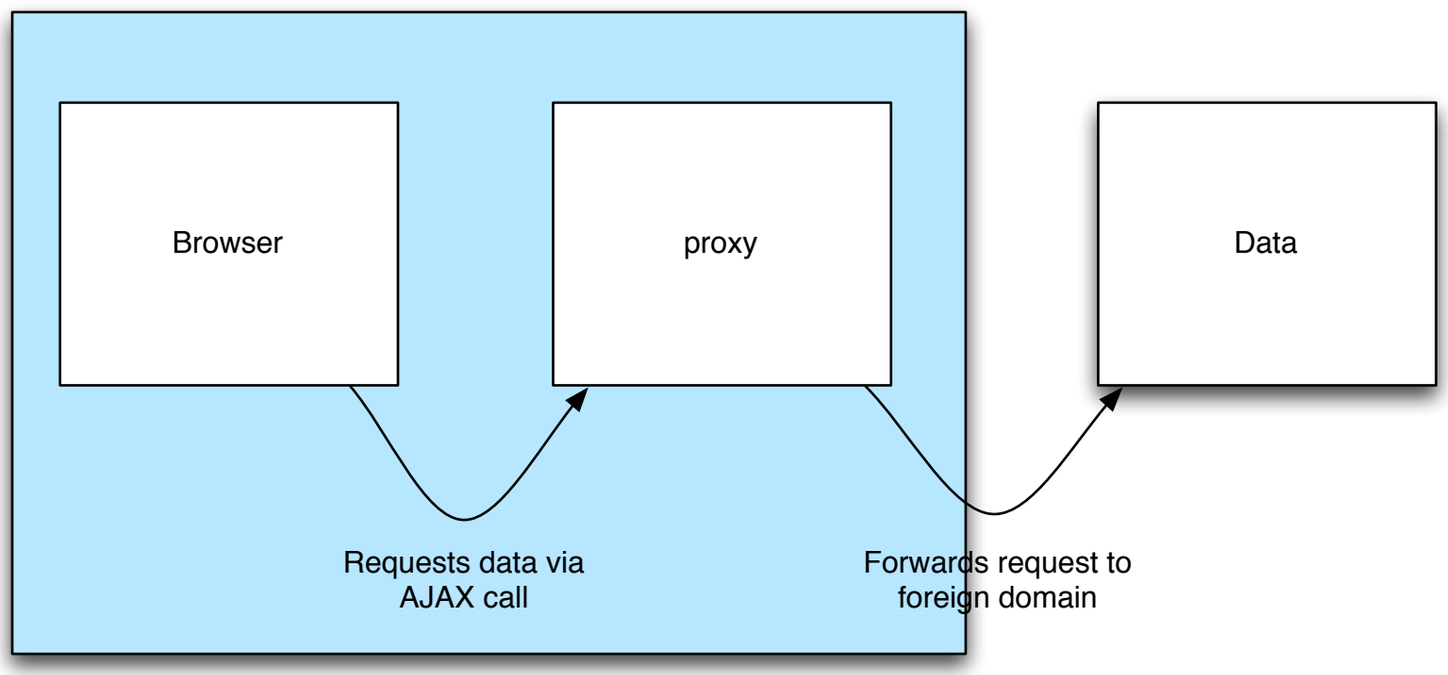
- Solutions to the cross-domain security restriction
  - Run a “proxy”
    - a php proxy
    - You request
      - http://mydomain.com/proxy.php?http://yourdomain.com/data.xml
    - Javascript thinks its coming from local server
      - proxy.php contains something like this code:

- php doesn’t run on ics.uci.edu
  - You need to use “students.ics.uci.edu”
  - http://www.ics.uci.edu/computing/web/faqs.php#students

Task 14

- This is the contents of a file on your server
- It is running a language called php
- the language is interpreted by your web server code
- server-side scripting

```php
<?php
  //Get the requested url from our proxy url
  $url = $_SERVER['QUERY_STRING'];

  // initialize curl with given url
  $ch = curl_init($url);

  //Tell curl to write the response to a variable
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

  //Tell curl to follow any redirects
  curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);

  $response = curl_exec($ch);

  $cType = curl_getinfo($ch,CURLINFO_CONTENT_TYPE);

  //Output the right mime/type
  header('Content-type: '.$cType);

  //Pass through the data
  print $response;

  curl_close($ch);
?>
```



Task 14

- Solutions to the cross-domain security restriction
  - Use JSONP
    - Requires server support
      - Yahoo APIs support this
    - Leverages Javascript loophole
      - XMLHttpRequests are restricted
      - Remote Javascript is not

Task 14

- JSONP adds a <script src=> element to your web page with the external URL as the src target

- Getting JSON looks like this:
  - Request: http://yourdomain.com/data.json
  - Return: {“hello”:”world”}

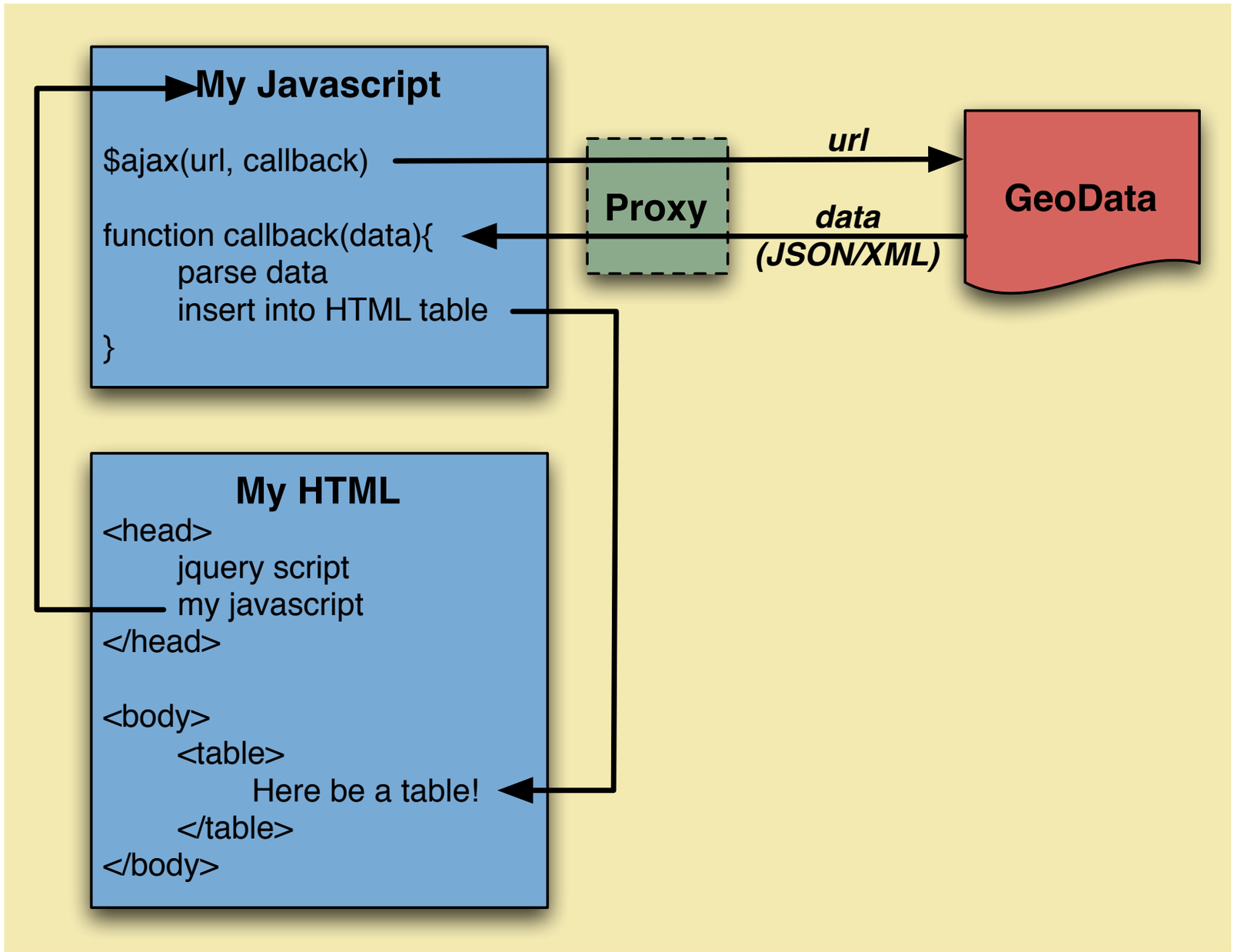
- Getting JSONP looks like this:
  - Request http://yourdomain.com/data.jsonp?callback=myCallback
  - Return: myCallback(“{\”hello\”:\”world\”}”)

- Your webpage writes a function called myCallback to deal with the data

Task 14

- Security issue
  - You are running server generated code on your machine
    - !

Task 14


My Javascript
$ajax(url, callback)
function callback(data){
parse data
insert into HTML table
}
Proxy
url
data
(JSON/XML)
GeoData
My HTML
<head>
jquery script
my javascript
</head>
<body>
<table>
Here be a table!
</table>
</body>

jQuery

```
<html>
    <head>
        <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
        <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.9.0/jquery-ui.min.js"></scr
        <script src="Step11.js"></script>
    </head>

    <body>
        <div class="dataHere">Replace this text!</div>
        <div class="textHere">Don't replace this text!</div>
        <div class="textAlsoHere">Don't replace this text either</div>
    </body>
</html>
```

jQuery

```
function escapeText(t){
    return document.createTextNode(t).textContent;
}


function jsonFlickrFeed(data) {
    myGoodLoadFunction(data);
}

function myGoodLoadFunction(data) {
    var newHTML= "<div style=\"padding:20px;border:solid 1px black\">"+escapeText(data.title)+"<br/>";
    newHTML += escapeText(data.link)+"<br/>";
    newHTML += escapeText(data.items[0].link)+"</div>";
    $("div.dataHere").html(newHTML);
}

function myBadLoadFunction(XMLHttpRequest,errorMessage,errorThrown) {
    alert("Load failed:"+errorMessage+":"+errorThrown);
}


function myReadyFunction(){
    $.ajax({
        url: "http://api.flickr.com/services/feeds/geo/QDd_2PObCZ4ZsRM6Sw&format=json",
        dataType: "jsonp",
        jsonp: false,
        jsonpCallback: "jsonFlickrFeed",
        error: myBadLoadFunction,
    });
}

$(document).ready(
    function(){
        myReadyFunction();
    }
);
```

jQuery

```
function escapeText(t){
    return document.createTextNode(t).textContent;
}


function jsonFlickrFeed(data) {
    myGoodLoadFunction(data);
}

function myGoodLoadFunction(data) {
    var newHTML= "<div style=\"padding:20px;border:solid 1px black\">"+escapeText(data.title)+"<br/>";
    newHTML += escapeText(data.link)+"<br/>";
    newHTML += escapeText(data.items[0].link)+"</div>";
    $("div.dataHere").html(newHTML);
}

function myBadLoadFunction(XMLHttpRequest,errorMessage,errorThrown) {
    alert("Load failed:"+errorMessage+":"+errorThrown);
}


function myReadyFunction(){
    $.ajax({
        url: "http://api.flickr.com/services/feeds/geo/QDd_2PObCZ4ZsRM6Sw&format=json",
        dataType: "jsonp",
        jsonp: false,
        jsonpCallback: "jsonFlickrFeed",
        error: myBadLoadFunction,
    });
}

$(document).ready(
    function(){
        myReadyFunction();
    }
);
```

jQuery

- What if you have something other than JSON to load?
  - XML
  - jQuery.parseXML()
- Where can I find info about jQuery actions?
  - jQuery API Reference
  - for example, html() vs text() vs append()?

jQuery - Final example Step14

```
<html>
    <head>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
        <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.9.0/jquery-ui.min.js"></script>
        <script src="Step14.js"></script>
    </head>

    <body>
        <div class="dataXML">
            <img src="coffeebean.jpg" alt="coffeebean" /><br/>
            Replace this with XML
        </div>
        <hr/>
        <div class="dataJSON">
            <img src="coffeebean.jpg" alt="coffeebean" /><br/>
            Replace this with JSON
        </div>
        <hr/>
        <div class="dataJSONP">
            <img src="coffeebean.jpg" alt="coffeebean" /><br/>
            Replace this with JSONP
        </div>
    </body>
</html>
```

jQuery

```
function myBadLoadFunction(myXMLHttpRequest,myErrorMessage,myErrorThrown) {
    alert('status: ' + myErrorMessage + '\n' + myXMLHttpRequest.responseText);
}


function myReadyFunction(){
    $.ajax({
        url: "https://students.ics.uci.edu/~djp3/myProxy.php?http://api.flickr.com/servic
        dataType: "xml",
        success: myGoodLoadXML,
        error: myBadLoadFunction
    });
    $("div.dataXML").html("AJAX XML call initiated:<br/>");

    $.ajax({
        url: "https://students.ics.uci.edu/~djp3/myProxy.php?https://graph.facebook.com/c
        dataType: "json",
        success: myGoodLoadJSON,
        error: myBadLoadFunction
    });
    $("div.dataJSON").html("AJAX JSON call initiated:<br/>");

    $.ajax({
        url: "https://gdata.youtube.com/feeds/base/users/djp3/uploads?alt=json-in-script&
        dataType: "jsonp",
        success: myGoodLoadJSONP,
        error: myBadLoadFunction
    });
    $("div.dataJSONP").html("AJAX JSONP call initiated:<br/>");

}

$(document).ready(
    myReadyFunction
);
```

jQuery - XML

```
function escapeText(t){
    return document.createTextNode(t).textContent;
}

function myGoodLoadXMLHelper()
{
    var newHTML= "<div style=\"margin-left:10px;padding:5px;border:solid 1px red\">";
    newHTML += escapeText($(this).text());
    newHTML += +"</div>";
    $("div.dataXML").append(newHTML);
}

function myGoodLoadXML(data) {
    $("div.dataXML").html("<h1>AJAX XML call returned:</h1>");
    $(data).find("item title").each(myGoodLoadXMLHelper);
}
```

jQuery - JSON

```
function myGoodLoadJSON(data) {
    $("div.dataJSON").html("<h1>AJAX JSON call returned:</h1>");

    var newHTML= "<div style=\"margin-left:10px;padding:5px;border:solid 1px green\">";
    newHTML += escapeText(data.name);
    newHTML += +"</div>";
    $("div.dataJSON").append(newHTML);

}
```

jQuery - JSONP

```
function myGoodLoadJSONP(data) {
    $("div.dataJSONP").html("<h1>AJAX JSONP call returned:</h1>");

    for(var i = 0; i< data.feed.entry.length; i++){
        var newHTML= "<div style=\"margin-left:10px;padding:5px;border:solid 1px blue\">";
        newHTML += escapeText(data.feed.entry[i].title.$t);
        newHTML += +"</div>";
        $("div.dataJSONP").append(newHTML);
    }

}
```

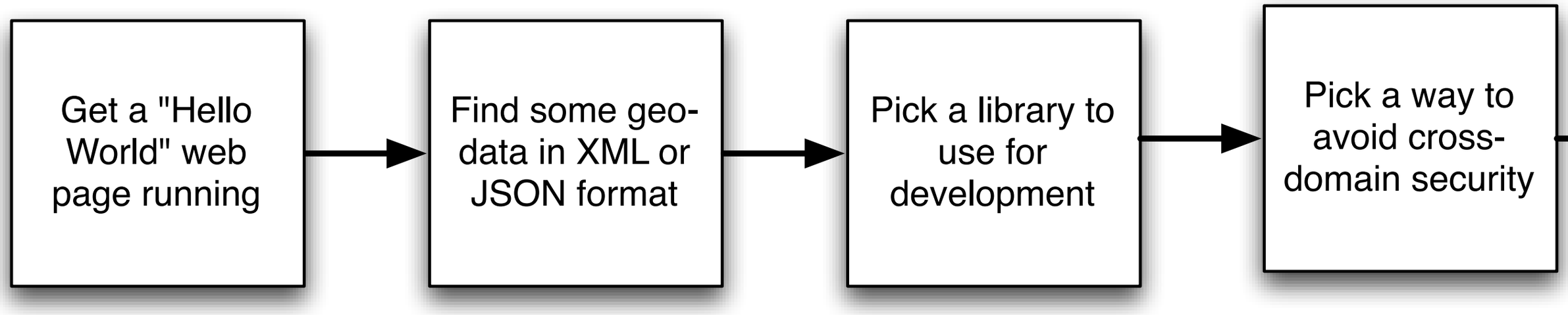
jQuery

- Task 14
  - Geocoded Feeds
    - http://api.flickr.com/services/feeds/geo/QDd_2PObCZ4ZsRM6Sw&format=json
    - Use dynamic data, not static data
  - jQuery
  - HTML Table
  - AJAX request

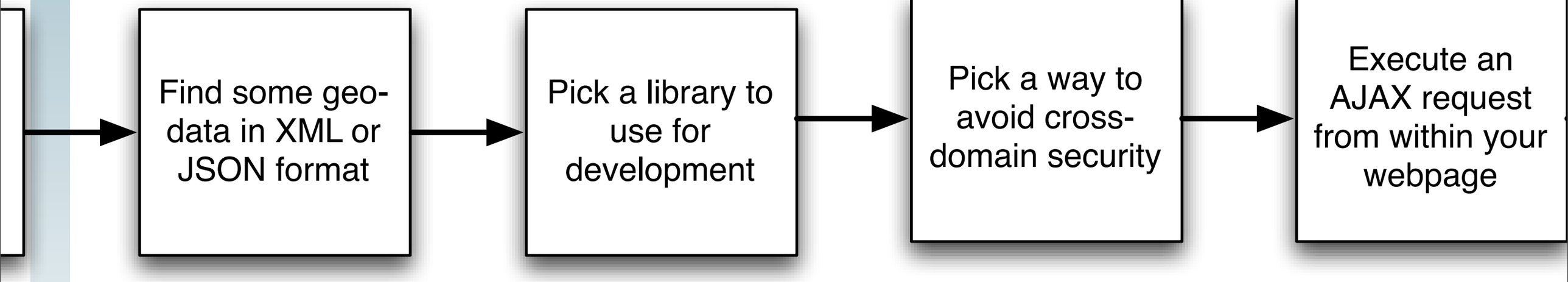
Task 14 strategy

- The goal is a web page
  - which has issued an AJAX request
  - parsed the data
  - and displayed it in a new way

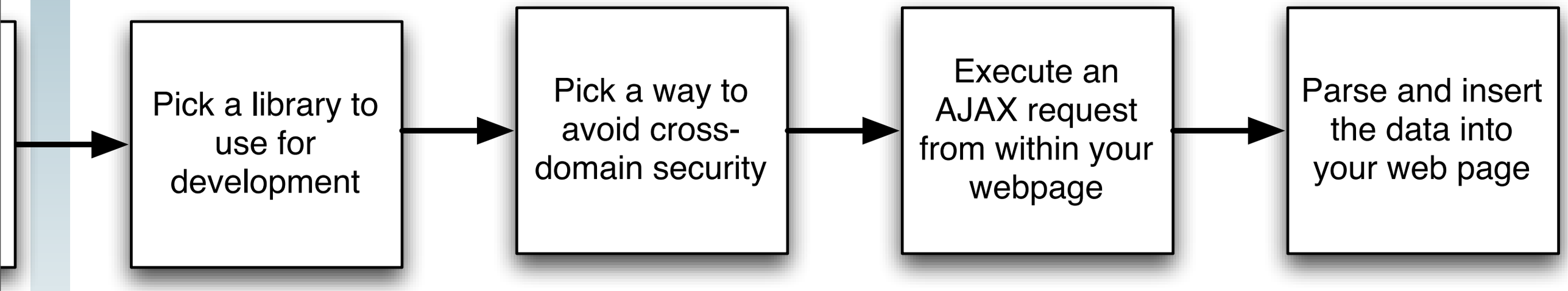
Task 14 strategy



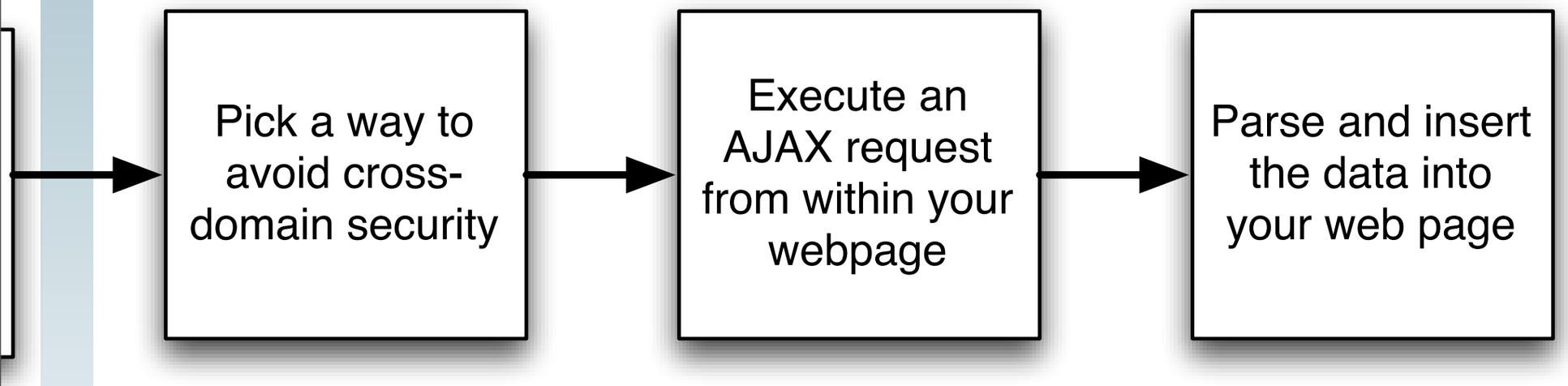
Task 14 strategy



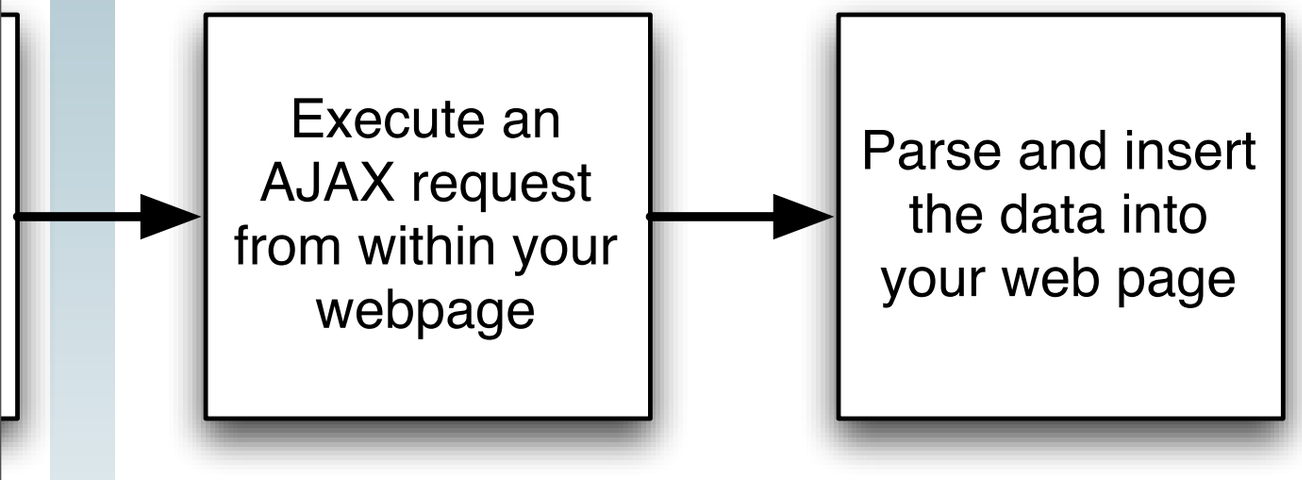
Task 14 strategy



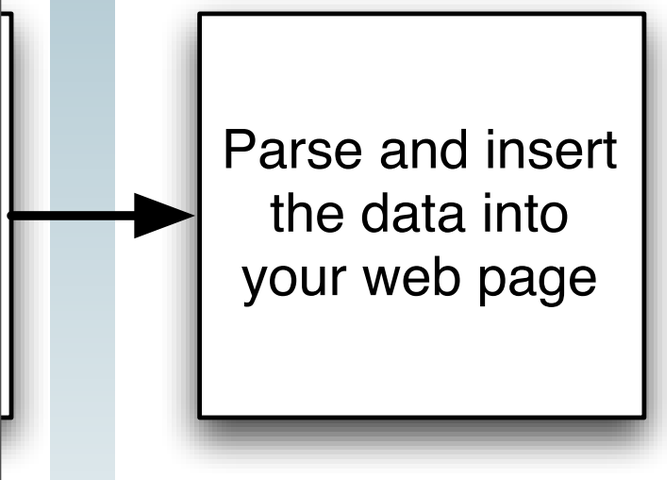
Task 14 strategy



Task 14 strategy



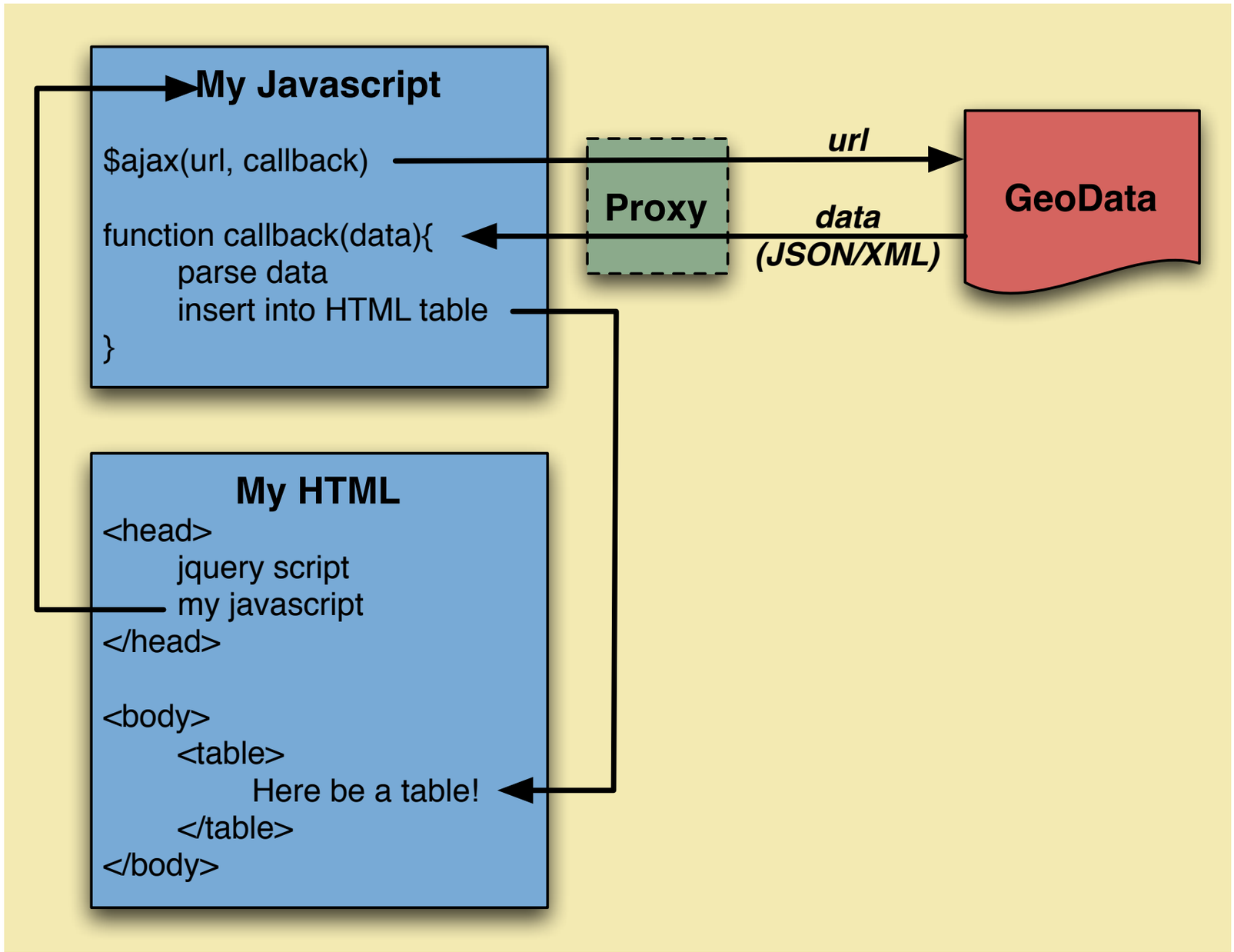
Task 14 strategy



Task 14 strategy

- Find some geo-data in XML or JSON format
  - (Hint: RSS is a specific type of XML)
- What is geo-data?
  - Anything that relates data to a spot on the earth
  - Data with a latitude and longitude
  - Data with an address
  - Data with a zip code
  - Data with a county
- For example:
  - Photos taken in Los Angeles
  - http://www.flickr.com/places/United+States/California/Los+Angeles

Task 14


My Javascript
$ajax(url, callback)
function callback(data){
parse data
insert into HTML table
}
Proxy
url
data
(JSON/XML)
GeoData
My HTML
<head>
jquery script
my javascript
</head>
<body>
<table>
Here be a table!
</table>
</body>

Task 14

- At a high-level requesting data asynchronously requires:
  - The location of where you want to get the data from
    - (for us these are the 5 geo-feeds)
  - What function to call when the data is ready
    - Because we aren't waiting
    - aka, "the call back function"
- The native function call to do this is called
  - "XMLHttpRequest"

jQuery

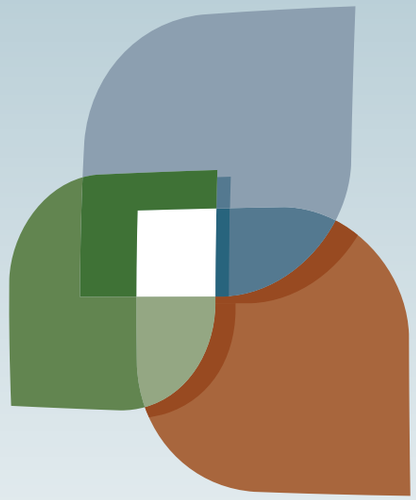
- Task 14
  - Present the data as a table
    - http://www.w3schools.com/html/html_tables.asp
  - HTML tables overview
    - 3 primary tags
      - <table>
      - <tr>
      - <td>

jQuery

- Basic table
  - <table>
  - <tr>
  - <td>

1	2
3	4

```
<html>
    <body>
        <table border="1">
            <tr>
                <td>
                    1
                </td>
                <td>
                    2
                </td>
            </tr>
            <tr>
                <td>
                    3
                </td>
                <td>
                    4
                </td>
            </tr>
        </table>
    </body>
</html>
```



LUCI