Ajax

INFO/CS 2300:
Intermediate Web Design and Programming

Crash course

Debugging – today – 5pm Accel Orange

Click in!

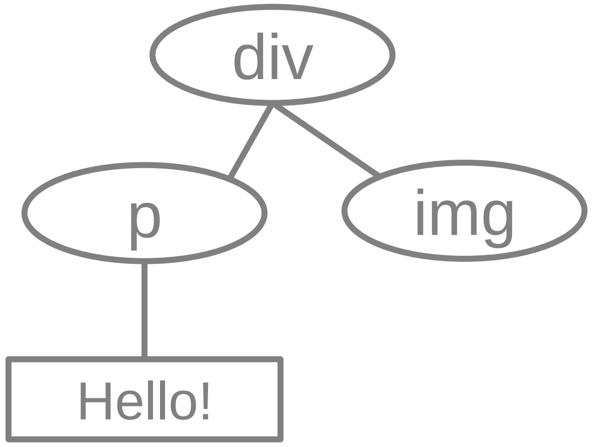
In the HTML tree, the <p> node is the _______ of the <div> node.

A. parent

B. sibling

C. child

D. ancestor



Click in!

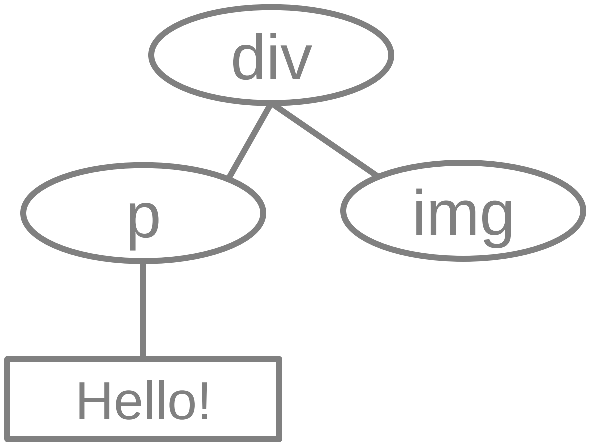
In the HTML tree, the <p> node is the _______ of the <div> node.

A. parent

B. sibling

C. child

D. ancestor



How would we use jQuery to add an onClick event handler to the image with id “myimage” that calls myFunction?

A. $(“myimage”).onClick(myFunction)
B. <img id=“myimage” onclick=“myFunction();” />
C. $(“#myimage”).click(myFunction);
D. $(“#myimage”).click(myFunction());

How would we use jQuery to add an onClick event handler to the image with id “myimage” that calls myFunction?

missing #

A. $(“myimage”).onClick(myFunction)

B. <img id=“myimage” onclick=“myFunction();” />

not jQuery

C. $(“#myimage”).click(myFunction);

D. $(“#myimage”).click(myFunction());

click is set to the return value of myFunction not myFunction itself

Which code below uses an anonymous function?

A. $(“a”).click(function () { return false;});

B. function anonymous() { return false;}

C. anonymous function name(arg) {

```
        return false;
    }
```

Which code below uses an anonymous function?

A. $("a").click(function () { return false;});

B. function anonymous() { return false;}

C. anonymous function name(arg) {

```
        return false;
    }
```

Ajax: what is it?

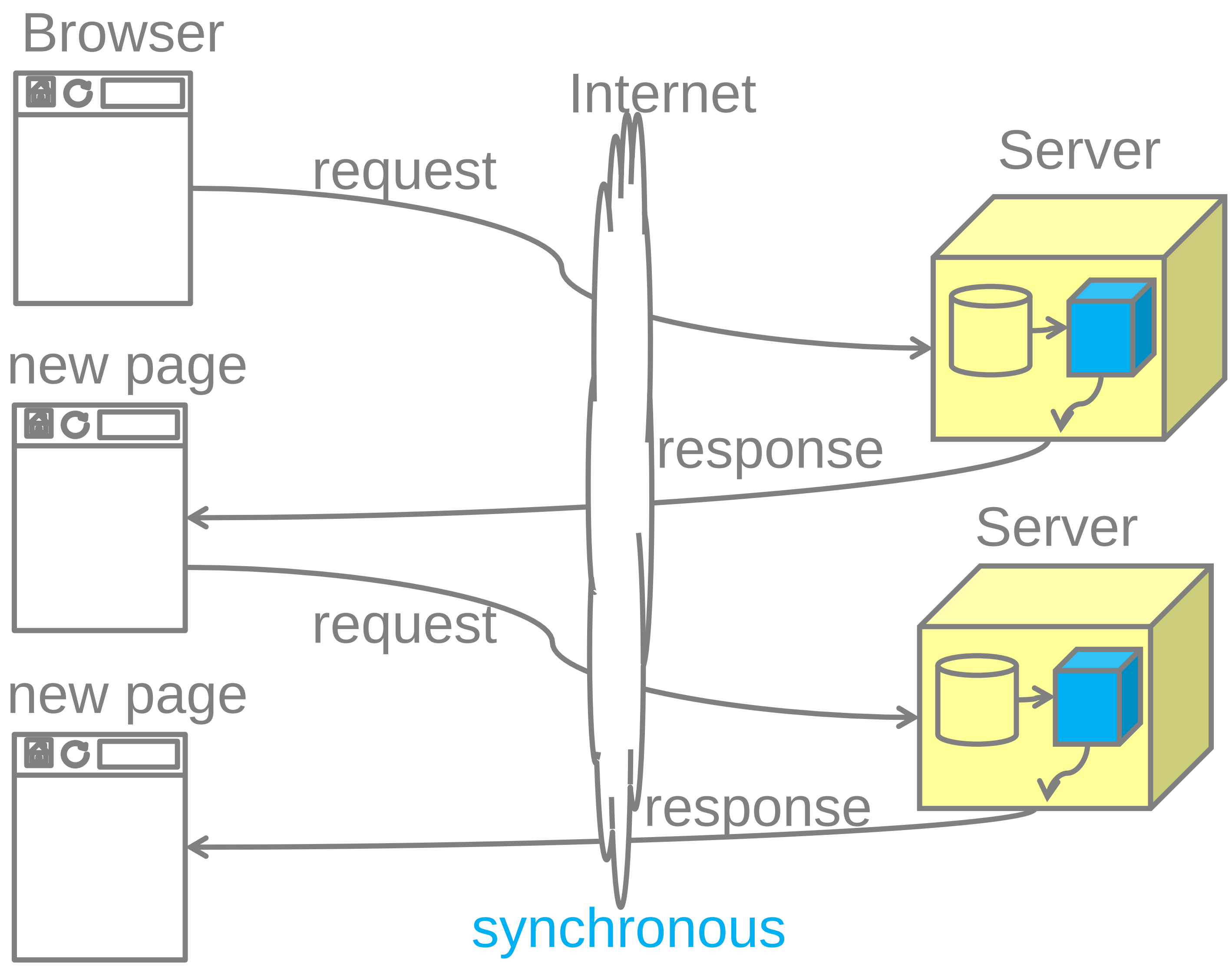
Ajax isn’t its own technology – it’s just shorthand for a cluster of technologies

- Asynchronous
- JavaScript
- and
- XML

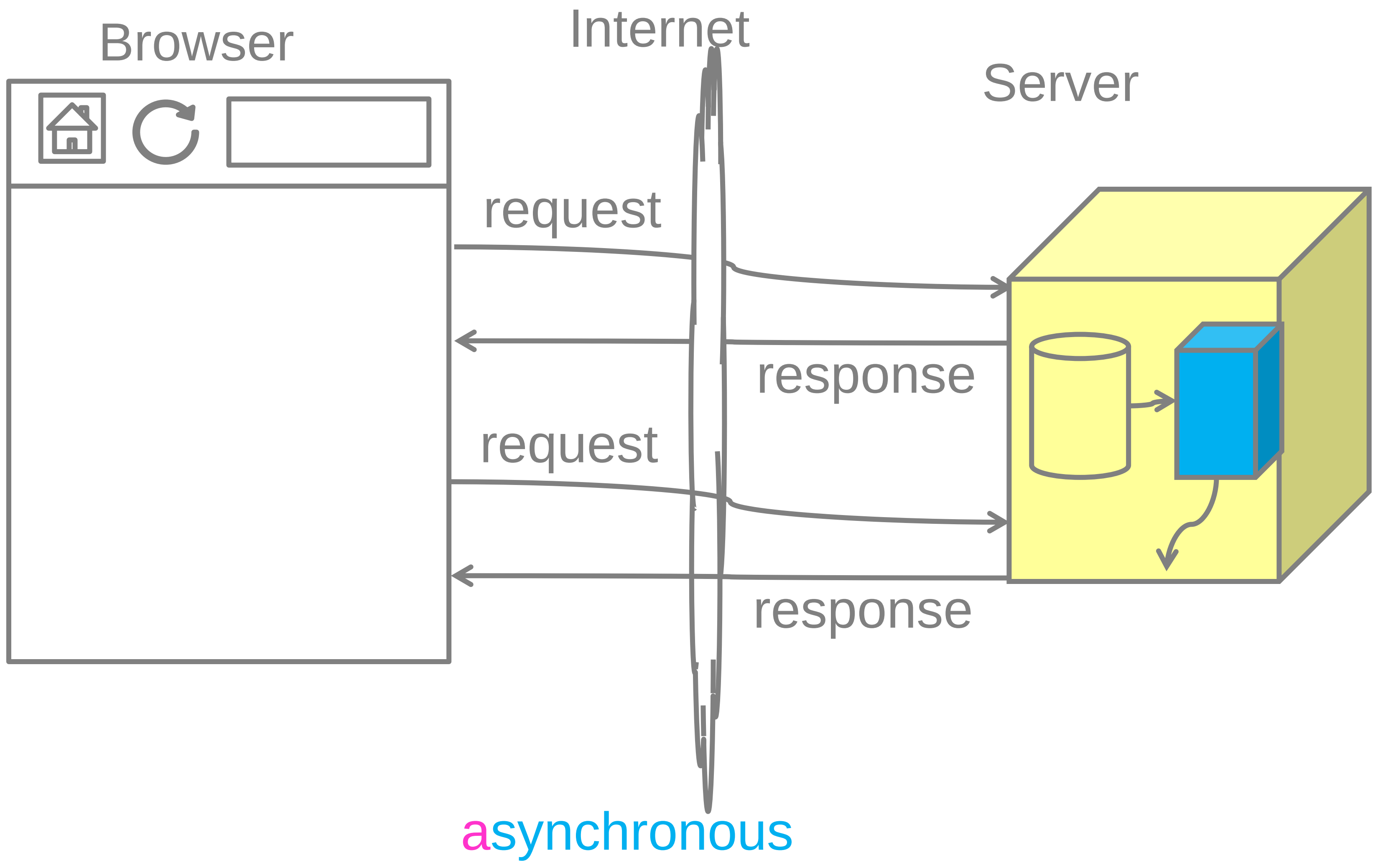
JSON is now more common than XML but AJaJ doesn't have as nice a sound to it

that help web applications substantially improve user experience

Traditional web application



Web application with Ajax



Request sent to server while user is still interacting with website. Using JavaScript and the DOM, we can update the page with results from the server.

Some examples

- http://irp.dpb.cornell.edu/
- Google search suggestions
- Google Docs save as you type
- Facebook infinite scroll

What pieces are needed?

- Browser needs a means of communicating with a server that isn't a whole page
- Format for the data exchange
- A trigger to initiate the exchange
- Browser processes server response without a page refresh

Steps for using Ajax

1. Tell JavaScript/jQuery we want to initiate an exchange with the server
2. State what program (file) on the server we want to run (for us, a PHP program).
3. Provide data to be sent to the server.
4. Give a JavaScript function to run when the response is received from the server.

Ajax with jQuery

To make an Ajax call, we use

```
request = $.ajax( {
    url : URL,
    type: "get",
    data: data,
    dataType: "json"
} );
```

request is an object variable that can be used later

$ is jQuery. $.ajax is a method of jQuery

or "post"

if we want the return to be processed as json instead of text

where URL is the program on the server we want to run and *data* is the data we want to send it.

Sending data

Data to send should be in the form

```
var mydata = { var1 : val1, var2 : val2 };
```

then on the server, the variables will appear in $_GET as $_GET["var1"], $_GET["var2"]

or $_POST

Specifying a callback function

Once we've executed the Ajax, we can tell jQuery what to do when the server is done by giving a *callback* function.

remember our request variable?

E.g.

`request.done( myFunction )`

jQuery will call myFunction when the browser receives a successful server response, and the data from the server's response will be in first argument of myfunction.

`function myFunction(result) { …. }`

More callbacks

request.done only executes on success

```
request.fail( myErrorFunction );
```

```
request.always( myAlwaysFunction);
```

Another syntax

```
$.ajax({
    url: url,
    type: 'get',
    data: data,
    dataType: 'json'
})
.done( function() {
    // handle success
})
.fail( function() {
    // handle request failures / errors
})
.always( function() {
    // Run this code on success or failure – after done or fail
});
```

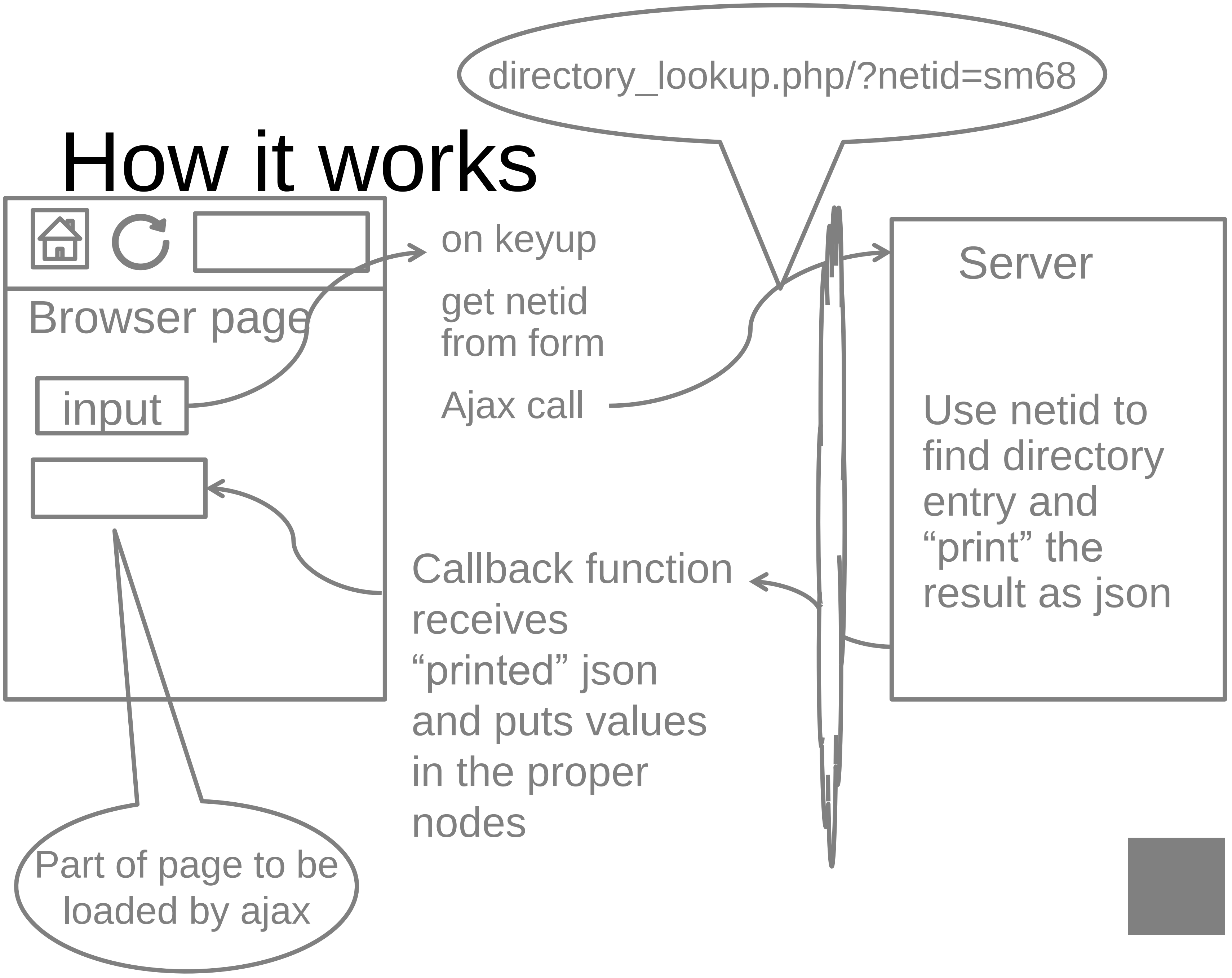
The meaning of success

"Success" in this case means that the server responded to the ajax request successfully. The response could include a variable called "error" that is part of the data.

An example: directory lookup

This is on the handout exercise and your server account in the lecture07 folder

How it works


directory_lookup.php/?netid=sm68
Browser page
input
on keyup
get netid from form
Ajax call
Server
Use netid to find directory entry and "print" the result as json
Callback function receives "printed" json and puts values in the proper nodes
Part of page to be loaded by ajax

The steps

1. On keyup event, get input from field and use an Ajax call to send the field input to directory_lookup.php.
2. PHP program directory_lookup.php looks up netID in an internal array. Combine the looked up value into a json string, and return.
3. The callback function takes the returned information and puts all the data in the appropriate nodes.

The HTML file

```
<body>
    <h1>2300 Directory Service</h1>

    <form action="submits_by_jquery_instead.txt" method="get">
          Net Id:<input type="text" id="netid">
    </form>

    <p><span id="error">&nbsp;</span></p>
    <p>
          <span class="label">First name: </span>
          <span id="first_name"></span>
    </p>
    <p>
          <span class="label">Last name: </span>
          <span id="last_name"></span>
    </p>
    <p>
          <span class="label">E-mail: </span>
          <span id="email"></span>
    </p>
</body>
```

As a non-ajax backup we would set ‘action’ to a real value in case javascript is not working

Adding the event handler

How do we set the onKeyUp event handler on the netid input to the function “findNetIDInfo”?

```
$(document).ready( function () {




} );
```

Triggers once the html is loaded

```
var request;

$(document).ready( function () {
   //Initialize the request variable to null
   request = null;

   $("#netid").keyup(findNetIDInfo);

} );
```

```
function findNetIDInfo() {
    //abandon any active server requests
    if (request) { request.abort(); }

    //Get the value of the netID from the input


    //Prepare the data by putting it in JSON format


    //Initiate the ajax call









    //Set the displayNetIDInfo function to run on success


}
```

```
function findNetIDInfo() {
    //abandon any active server requests
    if (request) { request.abort(); }

    //Get the value of the netID from the input
    var netIdFromForm = $("#netid").val();

    //Prepare the data by putting it in JSON format
    var dataToSend = { netid: netIdFromForm };

    //Initiate the ajax call
    request = $.ajax( {
            url: "includes/directory_lookup.php",
            type: "get",
            data: dataToSend,
            dataType: "json"
    } );

    //Set the displayNetIDInfo function to run on success
    request.done(displayNetIDInfo);
}
```

directory_lookup.php

Returns a string that looks like this:

Index values correspond to the node id in the html

```
{"first_name":"Steve",
"last_name":"Mohlke",
"email":"smohlke@cornell.edu"
}
or
{"error":"NetID not found"}
```

This is the native JavaScript form which can be referenced like this:

```
var first_name = response[ 'first_name' ];
```

It can be looped through like an associative array

```
function displayNetIDInfo(response) {
    //Define an array of nodes to clear of text
      var result_nodes = Array("error", "first_name",
                                                  "last_name", "email");

    //Set the text to an empty string




    // Each json entry will be of the form "nodeid" : "value".
    // Use this fact to stuff the values in the node with the id
    of index.



}
```

```
function displayNetIDInfo(response) {
    //Define an array of nodes to clear of text
      var result_nodes = Array("error", "first_name",
                                                    "last_name", "email");

    //Set the text to an empty string
    for (i in result_nodes) {
          $("#"+result_nodes[i]).text("");
    }

    // Each json entry will be of the form "nodeid" : "value".
    // Use this fact to stuff the values in the node with the id
    of index.
    for (i in response) {
          $("#"+i).text( response[ i ] );
    }
}
```

A word about JSON

“JSON” stands for “JavaScript Object Notation”.

We needed to tell the Ajax call that’s what was coming back.

```
request = $.ajax( {
        url: “directory.php",
        type: "get",
        data: getdata,
        dataType: “json”
  } );
```

JSON shortcut

Instead of $.ajax there is shorthand

request = $.getJSON(*URL, data, callback);*

E.g.

request = $.getJSON(“directory_lookup.php”, dataToSend, displayNetIDInfo);

More Ajax possibilities

http://api.jquery.com/jquery.ajax/

Review

- Ajax allows us to interact with the server, run code there and send answers back without forcing the user to wait for a response.
- This will become even more interesting once the server can start looking up information in a database…