Learning Objectives:
• Learn the basic concepts of distributed systems.
• Understand how making an Ajax call to a remote system is similar to, and different from, making an ordinary local function call.
• Learn the basic syntax for making an Ajax call and processing the results.
• Learn the basic syntax for passing arguments to an Ajax call.

Scroll down for content ☐
Distributed systems

So far, everything has been "running" on your own computer. In the real world (obviously) there are many computers, all networked together.

Each computer can run its own code, simultaneously with the others. When multiple computers work together in concert, it's called a distributed system.

Ajax

Ajax is a relatively new technology that lets your program invoke a function that's running on some other computer.

Ajax stands for "Asynchronous JavaScript and XML":

• asynchronous -- the two computers that are communicating (your computer and rosemary, in our case) do not have to wait for each other to progress. Instead, one registers an event handler to wait for the other one, just like button event handlers for on-page widgets.

• JavaScript -- you know that one. :)

• XML -- this is a data representation language, similar in syntax to HTML. You don't need to know much about it, but we'll be writing little bits of XML in order to send parameters as inputs to remote Ajax functions.
Making an Ajax call

In spirit, making an Ajax call is a lot like calling an ordinary function that resides on your own local machine. Syntactically, it looks a lot different though:

```javascript
$.ajax({
    url : "the link to the remote function",
    dataType : "xml",
    type : "POST"
}).done(yourEventHandlerFunction);
```

For instance,

```javascript
$.ajax({
    url : "http://rosemary.umw.edu/~stephen/endWorldHunger.php",
    dataType : "xml",
    type : "POST"
}).done(yourEventHandlerFunction);
```

Syntactic notes:

1. Notice that it's a paren/curly pair after the $.ajax, not just a paren pair. The entire curly pair (and its contents) are nested inside the paren pair.

2. The items inside the paren/curly pair are each composed of two parts, with a colon separating them.

3. Each of these items except the last ends with a comma.

4. The name of the event handler function that you want to be invoked whenever the Ajax call completes goes inside the "done()" bit at the end.
Passing arguments to an Ajax call

To pass arguments to an Ajax call, you need two additional items in your list:

```javascript
$.ajax({
    url : "the link to the remote function",
    dataType : "xml",
    type : "POST",
    contentType : "text/xml",
    data : "your string of XML data"
}).done(yourEventHandlerFunction);
```

For instance,

```javascript
$.ajax({
    url : "http://rosemary.umw.edu/~stephen/endWorldHunger.php",
    dataType : "xml",
    type : "POST",
    contentType : "text/xml",
    data : "<lastYearOfHunger>2015</lastYearOfHunger>"
}).done(yourEventHandlerFunction);
```
The "same origin" policy

One important note is that in order for a browser to "allow" a Web page to make an Ajax call, that Ajax call must reside on the same server that the Web page itself was retrieved from. This is called the "same origin policy," and is a universal security restriction among the major browsers.

What this means for this class is that from now on, all of your Web pages are going to have to live on rosemary from the beginning, instead of just when you submit them at the end. Every time you make a change to your code, you'll have to re-upload the changed files to your public_html folder on rosemary via Filezilla or some other tool.