

Computer Science & Engineering 120
Learning to Code

Consuming Data I – Ajax

Christopher M. Bourke
cbourke@cse.unl.edu

Part I: Introduction to Ajax

Topic Overview

- ▶ Client/Server Model & HTTP
- ▶ Ajax
- ▶ jQuery's `$.ajax()` function
- ▶ Security & Other Concerns

Client-Server Model

- ▶ Many applications follow a client-server model
- ▶ Client is responsible for end-user interaction
- ▶ Server provides various services (data, persistence, etc.)
- ▶ Basic interaction: client makes *requests*, server gives a *response*

title

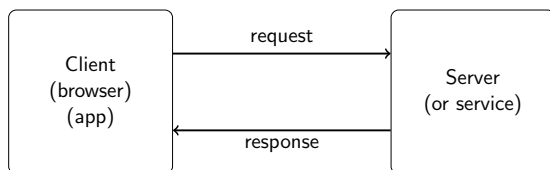


Figure : Client-Server Model

HTTP

- ▶ HyperText Transfer Protocol (HTTP) is the most widely used protocol for the web
- ▶ A *protocol* is a set of rules for communication
- ▶ Client: web browser (doesn't have to be though)
- ▶ Server: web server
- ▶ Demonstration

HTTP

- ▶ Most common types of requests: `GET` and `POST`
- ▶ Both support submission of *parameters* (inputs) to the request
- ▶ Responses include status codes (200 = success, 403 = Forbidden, 404 = Not found, 500 = Internal Server Error)
- ▶ Responses also include response data
- ▶ Demonstration

Ajax I

- ▶ Goal: want to be able to use various web services in our application
- ▶ Example: Google Map (distance) data
- ▶ Example: Yahoo Finance (offers data on stocks, exchange rates, etc.)
- ▶ Example: UNL's Bulletin
- ▶ Example: Our application can interact with our own server to get additional data (student data, enrollment data)

Ajax II

- ▶ Solution: JavaScript supports Ajax (through `XMLHttpRequest`)
- ▶ Ajax = Asynchronous JavaScript and XML
- ▶ JavaScript code can be run in a browser to make *additional* requests to a remote server for data and process the response
- ▶ Asynchronous: request/response requires network connection and communication; done so "in the background" so that it doesn't freeze ("block") the rest of the application
- ▶ Named with "XML" (eXtensible Markup Language) mostly due to legacy; we'll deal with JSON

jQuery's ajax() Function

- ▶ We'll focus on using jQuery's `$.ajax()` function (easier to use)
- ▶ Takes one parameter: an object with the request's full configuration
- ▶ Full documentation: <http://api.jquery.com/jquery.ajax/>
- ▶ Essential elements:
 - ▶ `url` – the URL to connect to
 - ▶ `data` – a JSON object representing any parameters to be sent
 - ▶ `success` – a callback function to be executed if the request is successful (receives the response data so that you can process it)
 - ▶ `error` – a callback function to be executed if the request fails
- ▶ Default is to use `GET`, can be changed using `method: "POST"`
- ▶ Demonstration: UNL Bulletin Data

Other Considerations

- ▶ It is best to give some visual cue (loading graphic) to the user to indicate that a request is being processed
- ▶ Asynchronicity needs to be considered: you cannot assume a request will always succeed
- ▶ You cannot assume a request is "fast" (need to use proper callback mechanisms)
- ▶ Feedback (success or failure) needs to be communicated to the user
- ▶ Requests to the same server can be made *relative*
- ▶ Example: script from `http://example.com` want to make a request to service at `http://example.com/services/dataService.php`, the URL only needs to be `/services/dataService.php`
- ▶ Beware: many web services require the use of an API Key

Security Considerations

- ▶ Most browsers will enforce a *Same Origin Policy*
- ▶ Scripts from server *A* are not allowed to make requests to server *B*
- ▶ Scenario: criminal lures us to bad server *A* whose scripts make requests to bank server *B* (bad because its coming from the same client)
- ▶ Request is not to the Same Origin, so denied

Security Workarounds

- ▶ Very limiting: we *want* to be able to use 3rd party servers (Google Maps, UNL Bulletin)
- ▶ Workaround: Server *B* can set a *Cross Origin Resource Sharing* policy
- ▶ Only recommended for truly non-sensitive public data (UNL Bulletin)
- ▶ Workaround: use a proxy or proxy service (a non-browser application that makes requests from server *A* on our behalf and relays the data)

Part II: Demonstration & Exercise

Formulate and execute an ajax request to a proxy to Google Map's distance API service. Integrate your call into an application.

Demonstration:

Google: <https://maps.googleapis.com/maps/api/distancematrix/json?origins=Lincoln+NE+68503&destinations=Omaha+NE+68116>
Proxy: <http://cse.unl.edu/~cbourne/CSCE120/proxies/distance.php?originCity=Lincoln&originState=NE&originZip=68503&destCity=Omaha&destState=NE&destZip=68116>