asynchronous calls

Daniel Jackson
some history

in the 1990s
› most web sites issued a whole page at a time
› clunky for users, excessive bandwidth

idea
› update page incrementally
› do it asynchronously, so browser doesn’t freeze

in 1999, XMLHttpRequest arrives
› Microsoft invents XHR idea for ActiveX in IE5
› adopted by other browsers for Javascript

in 2005, called “Ajax”
› asynchronous JavaScript and XML
what you can do with Ajax

push updates to page
› status, stock quotes, weather, time

interact with user over input
› drill downs (eg, state>city), autocomplete

persist user data on server
› Google docs, stickies

execute server-side commands on same page
› vote up/down on posts, email portal
event model

- single thread
- event loop: run timed event if expired or event at front of Q
- events added by: user actions, server responses, events
particulars

two key facts
› each event runs to completion before next event
› response to Ajax request is not always next

good news
› no need to worry about mutual exclusion

bad news
› long-running event freezes the UI
› timeouts may run late
› when Ajax response comes, context may have changed
JQuery’s Ajax API

$.getScript(url, callback)
› get script at url, run it, then execute callback

e.load(url)
› get content at url, and insert into element e

$.getJSON(url, data, callback)
› pass data to server at url
› server sends JSON back; parse and pass to callback

$.get(url, data, callback, type)
› type determines preprocessing of response
› $.post similar, but does HTTP post instead

$.ajax(…)
› most flexible, lowest level method
using network inspector

› example from Safari
using network inspector

› can see here that request was get
encoding data for transit

XML
- parsing built into browser (XHR)
- comes back as DOM: not convenient

JSON
- Javascript object literals
- JQuery uses parser, not eval (why?)
