AJAX == Asynchronous Javascript And Xml

- JSAPI call XmlHttpRequest (a/k/a xhr) contacts server *asynchronously* (in background) and *without redrawing page*
- Normal HTTP request, w/special XHR header
- Controller action receives request via route
- What should it render in response?

```
render :layout => false
render :partial => 'movies/show'
render :json => @movies  (calls to_json)
render :xml => @movies  (calls to_xml)
render :text => @movie.title
render :nothing => true
```

The basic AJAX cycle, as seen from browser JSAPI

r = new XmlHttpRequest;
r.open(method,URL,async);

method ∈ GET,POST,HEAD,PUT,DELETE...

async: true means script should not block (important!)

r.send("*request content*");

r.abort();

- Callbacks during XHR processing

r.onReadyStateChange=function(XmlHttpRequest r) { ... }

- function inspects r.readyState ∈ uninitialized,open, sent,receiving,loaded
- r.status contains HTTP status of response
- r.responseText contains response content string

The jQuery way

```
$.ajax({type: 'GET',
        url: URL,
        timeout: milliseconds,
        success: function,
        error: function
            // many other options possible
       });
```

Rails Cookery: AJAX+UJS with Rails 3

- javascript_include_tag :defaults
- your code in app/assets/javascripts/*.js
- Define *handler function* that responds to user action, and *bind* to appropriate element(s) in setup
  - Handler can inspect form element state, data-* attributes, etc.
  - Eventually call $.ajax()
- Define controller action & route to receive request, and determine what will be returned
- Define *callback function* to receive server reply
  - Unpack JSON objects & modify DOM?
  - Replace existing HTML element (e.g.) in place?
  - Add/change/remove CSS classes/properties?

When using JavaScript for client-side form validation, which is NOT true?

- JavaScript code can inspect DOM element attributes to see what the user typed
- JavaScript code can prevent the "Submit" button from submitting the form
- Some validations may be impractical to perform on client, so must be done on server
- The server doesn't have to repeat the validations performed by JavaScript

Emerging JS use cases

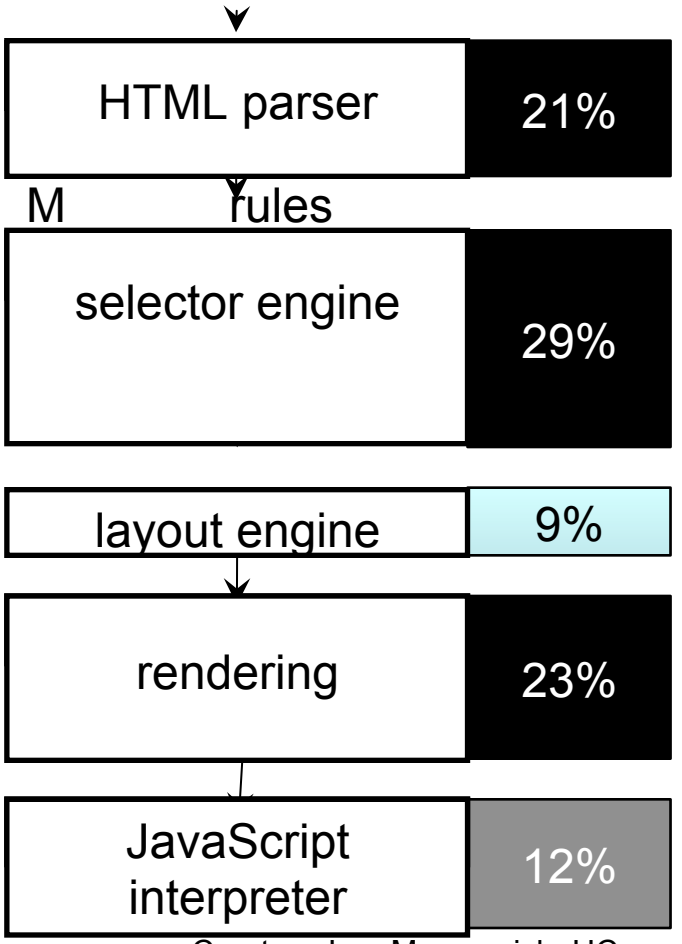
- Server-side frameworks: Node.js
- A server-side JS library for creating the server side of apps
- Uses event-driven programming in JS
- Potentially much more memory-efficient per user compared to traditional server-side frameworks
- Client-side frameworks: Backbone.js
- Express *views & models* entirely in JS
- Server just returns JSON
- Client-side frameworks: Yahoo Mojito

JavaScript & Performance

- The browser is increasingly the “client-side OS” that matters
- CSS has made layout increasingly sophisticated
- JavaScript + AJAX has made UI increasingly rich & desktop-like
- What’s the cost of all this?

Where does the time go?

- CSS selectors + JavaScript interpreter = 41% of total client rendering time
  - Esp. selectors that require walking DOM tree, eg div > li
- Browsers compete on speed of JavaScript interpreter => selector/ parser performance increasingly important
- Work at UC Berkeley (Prof. Ras Bodík's group):
  - Parallelizing parsing & CSS selectors
  - Increasing use of GPU for rendering



Courtesy Leo Meyerovich, UC Berkeley

Pitfall: “Adding JavaScript will make my app more responsive”

- Risk: too-wide API to server, with needlessly expensive database calls
- Risk: at mercy of JS interpreter on a *wide variety* of devices
- Risk: testing/debugging is more complicated, since must handle client-side errors

Using unobtrusive vs. traditional (obtrusive) JS improves compliance with which SOLID principle?

- Single Responsibility principle
- Open/Closed principle
- Demeter Principle
- Dependency Inversion/Injection Principle

*