MOBILE CHOICES
Native apps

- Native app
  - access to all phone features
  - best look and feel and performance
- BUT non-portable development platforms:
  - iPhone: Objective-C
  - Android: Java
MOBILE APP REQUIREMENTS

iPhone requirements:
- Mac with Snow Leopard (Lion?)
- XCode IDE and libraries
- iPhone SDK,
- developer license
- iPhone to deploy

Android requirements:
- Windows / Linux / Macs
- Eclipse IDE recommended
- Android SDK
- phone recommended but emulator works
PhoneGap / Cordova

- Portable framework for iPhone, Android, Blackberry, ..
- Runs as a native app, but coded HTML5 + CSS + Javascript
- Same requirements as native development
  - iPhone: XCode, iPhone SDK, iPhone...
  - Android: Eclipse, Java, Android SDK, ...
RESOURCES: PHONEGAP

- http://www.phonegap.com/
- http://incubator.apache.org/cordova/

Videos
- Early, non-slick 3 minute spiel on PhoneGap
- 9-minute demo converting HTML to iPhoneApp

Discussion groups
- http://groups.google.com/group/phonegap
- http://phonegapforum.com/

(books) http://www.amazon.com/s/ref=nb_sb_noss_1?url=search-alias%3Daps&field-keywords=phonegap
WEB PAGES

- Web page
  - multi-platform, easily tested
  - only needs HTML/CSS/Javascript skills
  - BUT limited access to phone features
  - feels like a web page
MOBILE WEB PAGES

- HTML5 + CSS + Javascript
  - support touch gestures
  - access to some phone features
  - can adapt to multiple screen sizes
  - provides local data store – critical for offline use
  - animations can be hardware optimized
MOBILE WEB PAGE RESOURCES

Mobile Web Page Frameworks

- (book ad) http://www.sitepoint.com/books/mobile1/
WEB APP TIPS

- Develop and test in Webkit browsers
  - Chrome or Safari
  - `<!DOCTYPE HTML>`

- Validate constantly!
  - HTML: http://validator.w3.org/
  - CSS: http://jigsaw.w3.org/css-validator/
  - Javascript: http://www.jslint.com/
RESTFUL WEB APPS
Has this ever happened to you?

You select an item from a menu.

The browser shows data about that item.
Has this ever happened to you?

You select an item from a menu.

The browser shows data about that item.

You click the back button to see something else.

A scary dialog box asks if you want to resubmit data to the server.
Has this ever happened to you?

You select an item from a menu.

The browser shows data about that item.

You click the back button to see something else.

A scary dialog box asks if you want to resubmit data to the server.

I'm talking to you, CAESAR.
Has this ever happened to you?

You select an item from a menu.
The browser shows data about that item.

You bookmark that page for future reference.

Later, you click the bookmark.
The browser takes you to the main page instead.

You click the back button to see something else.

A scary dialog box asks if you want to resubmit data to the server.

I'm talking to you, CAESAR.
Has this ever happened to you?

You select an item from a menu.

The browser shows data about that item.

You click the back button to see something else.

A scary dialog box asks if you want to resubmit data to the server.

I'm talking to you, CAESAR.

You bookmark that page for future reference.

Later, you click the bookmark.

The browser takes you to the main page instead.

I'm talking to you, Blackboard.
HOW THE WEB WORKS

(slightly simplified)
How the web works

(slightly simplified)

Browser

Clicking a link: GET + URL

Submitting a form: POST + form data

Server
How the web works

(slightly simplified)

Clicking a link: **GET + URL**

Submitting a form: **POST + form data**

HTML + CSS + media

Wednesday, October 24, 2012
GET and POST are very different actions for the browser.

Clicking a link: \textit{GET + URL}  \textbf{Browser} \rightarrow \textbf{Server}

Submitting a form: \textit{POST + form data}  \textbf{Browser} \rightarrow \textbf{Server}

HTML + CSS + media  \textbf{Server} \rightarrow \textbf{Browser}
How the web works

(Get and POST are very different actions for the browser.)

Clicking a link:

GET + URL

Submitting a form:

POST + form data

HTML + CSS + media

(GET and POST are often handled by the same code on the server.)
GET and POST are very different actions for the browser.

Clicking a link: GET + URL

Submitting a form: POST + form data

HTML + CSS + media

GET and POST are often handled by the same code on the server.

But POST may modify state, GET should not

(slightly simplified)
### HTTP METHOD TYPES

<table>
<thead>
<tr>
<th>GET, HEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
</tr>
</tbody>
</table>
HTTP METHOD TYPES

GET, HEAD

Safe

only retrieves data

Idempotent

repeated calls get the same results

POST
HTTP METHOD TYPES

- GET, HEAD
  - Safe
  - only retrieves data
  - Idempotent
    - repeated calls get the same results

- POST
  - Neither
  - Ergo, browsers ask before repeating
WHY THE WEB WORKS

The web scaled because sites are mostly repositories of self-describing resources, not applications.

Resources

server state = a collection of resources

GET

retrieve a resource

stable unique
bookmarkable URLs

safe, stateless, re-execute at any time

POST

update server state

not safe to re-execute

Roy Fielding "Architectural Styles and the Design of Network-based Software Architectures" (PhD, UCI, 2000)
RESTFUL WEB APPS

(REpresentational State Transfer)

- define as much of your system as possible in terms of resources (lists, item details, carts, user profiles, ...)
- provide an initial unchanging home URL with links to other resources
- make all resources available via some chain of links starting from the home page
- use GET to retrieve resources
- use POST (and/or PUT and DELETE) to update resources
COMMON MISTAKE #1

**Bug**

A link (GET) that updates a resource.

Example: the PHP tutorial's demo blog has a delete link next to each blog entry

A web crawler, e.g., Google, a site map generator, a broken link finder, even a browser pre-fetch loop, will delete every blog entry!

**Fix**

Use a form with POST for all actions that modify server state.
COMMON MISTAKE #2

**Bug**

A form (POST) that just gets a resource.

Example: a search field

Search results can't be bookmarked.

Going back to the results page triggers "do you want to resend data?"

**Fix**

Use form with GET method

Use POST-REDIRECT-GET pattern
POST-REDIRECT-GET

Client

POST + form data

Server

http://blog.andreloker.de/post/2008/06/Post-Redirect-Get.aspx
POST-REDIRECT-GET

Client

POST + form data

Redirect to URL

GET + URL

Server

http://blog.andreloker.de/post/2008/06/Post-Redirect-Get.aspx
POST-REDIRECT-GET

Client

POST + form data

Redirect to URL

GET + URL

Server

after update, redirect browser to resource with results

no "resubmit?" on browser back

http://blog.andreloker.de/post/2008/06/Post-Redirect-Get.aspx
REST solutions are fast solutions

Example: don't create user accounts, create user-specific URLs
  - no need to implement authentication
  - no friction for users
    - "I'm not giving these guys my email (Facebook access, ...)")"
    - "Cripes, another password to keep track of..."

URL for user's home page, video stream, circle of friends
  - Trivial to implement or do by hand
  - Easy to add authentication and access control later
  - Much easier to test with test scripts

Examples: join.me and Google+ Hangout
Single Page App: all or almost all interaction occurs on a single web page

Apparent multiple pages actually DIV's hidden and shown using Javascript and CSS

Data retrieved using XHR (XMLHttpRequest), a.k.a. AJAX (asynchronous Javascript and XML)

POST-REDIRECT-GET not needed with XHR
RESTFUL APIs

- URL's for XHR calls designed following RESTful principles
  - GET to retrieve data with no change in server state
  - POST to modify server state
    - Also PUT and DELETE
  - Decouples the service from the client
  - Enables wider user of the service, e.g., mashups