This class, web app basics, and ember
Secure Web Application Development

Dr. Hale
University of Nebraska at Omaha
Secure Web Application Development– Lecture 1

Welcome to IA8470

Jumping Ahead: Download the files you need

virtualbox.org

and

\\resfs01.unomaha.edu\ResearchData\IST\projects

(replace with forward slashes for mac users)

What is this class?

We will be making stuff. Its not for the faint of heart, but trust me it will be awesome and you will feel great when you create something new.

Why should you trust me?

I've made a lot of web apps – its fun and rewarding. Also the previous two year's groups really enjoyed the class.

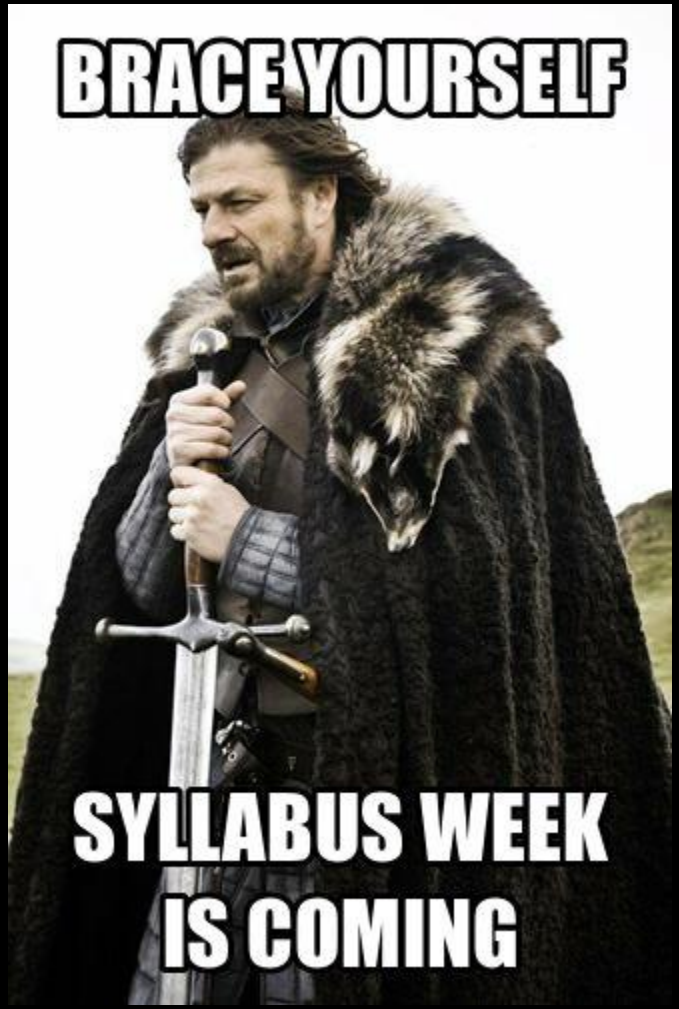
Ok so what will we cover?

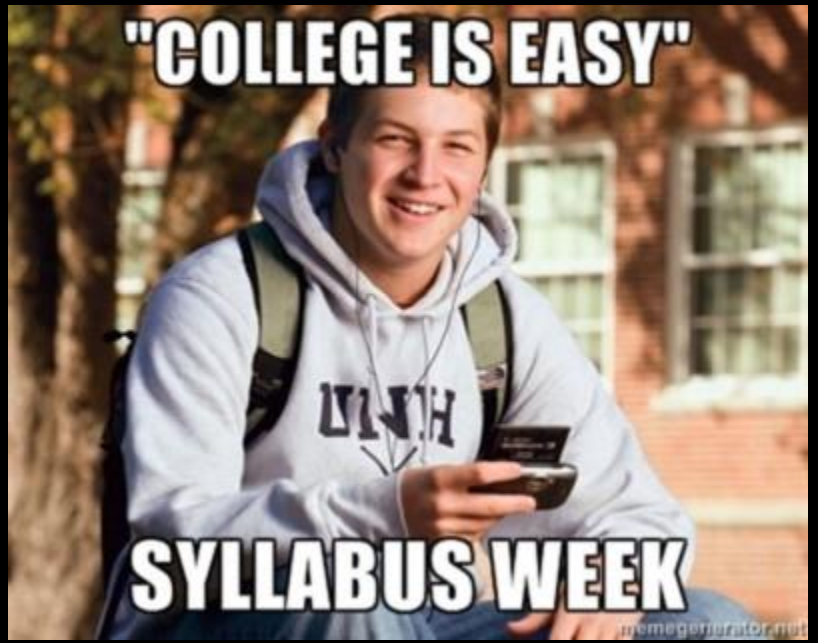
Technologies

Ok so what will we cover?

Software Engineering: Architecting and design
Software project management and Team dynamics
Hardening and secure coding in the SDLC
RESTful Web service APIs
Client-side application development

Topics





Hint: Syllabus time

Today's Class: Not spending the entire class on the syllabus

Part 1: Web App basics

- Threat vectors
- Basic architecture
- Clientside-vs-serverside

Part 2: Ember.js

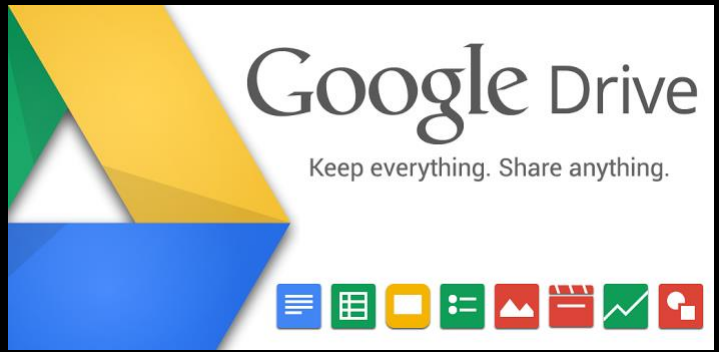
- An introduction to Ember
- Introduction to MVC
- Introduction to Handlebars

Part 3: Getting started

Part1:

Lets talk about a little about web apps and security.

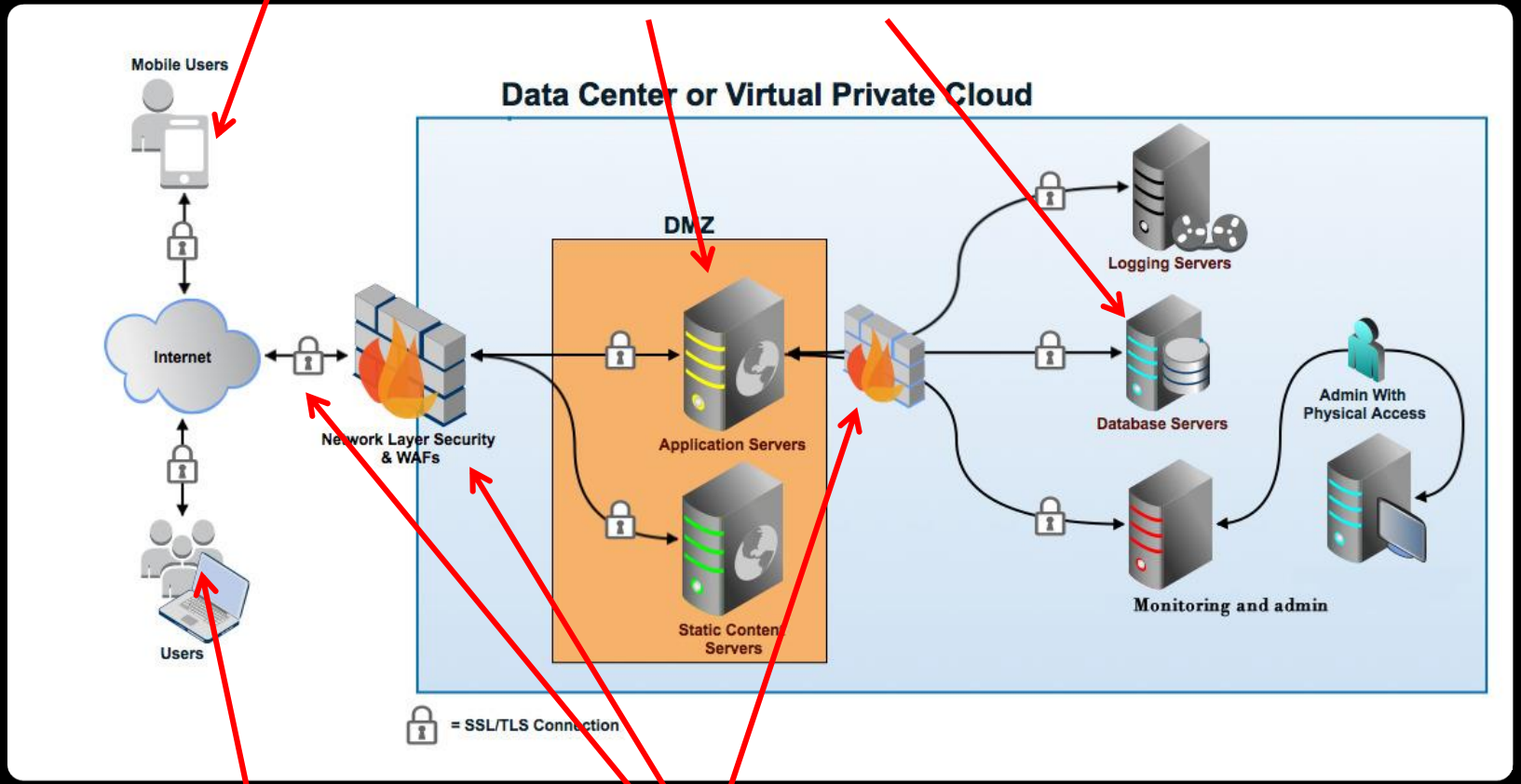
(nearly) All companies have a web app.
You probably use them, daily.



Also you

Where developers should actually focus on security

Typical Web Application Architecture



You

What most companies think "security" means

As a user, you expect web apps to be:

fast, responsive, always available, and secure

Percentage of tested apps with vulnerabilities

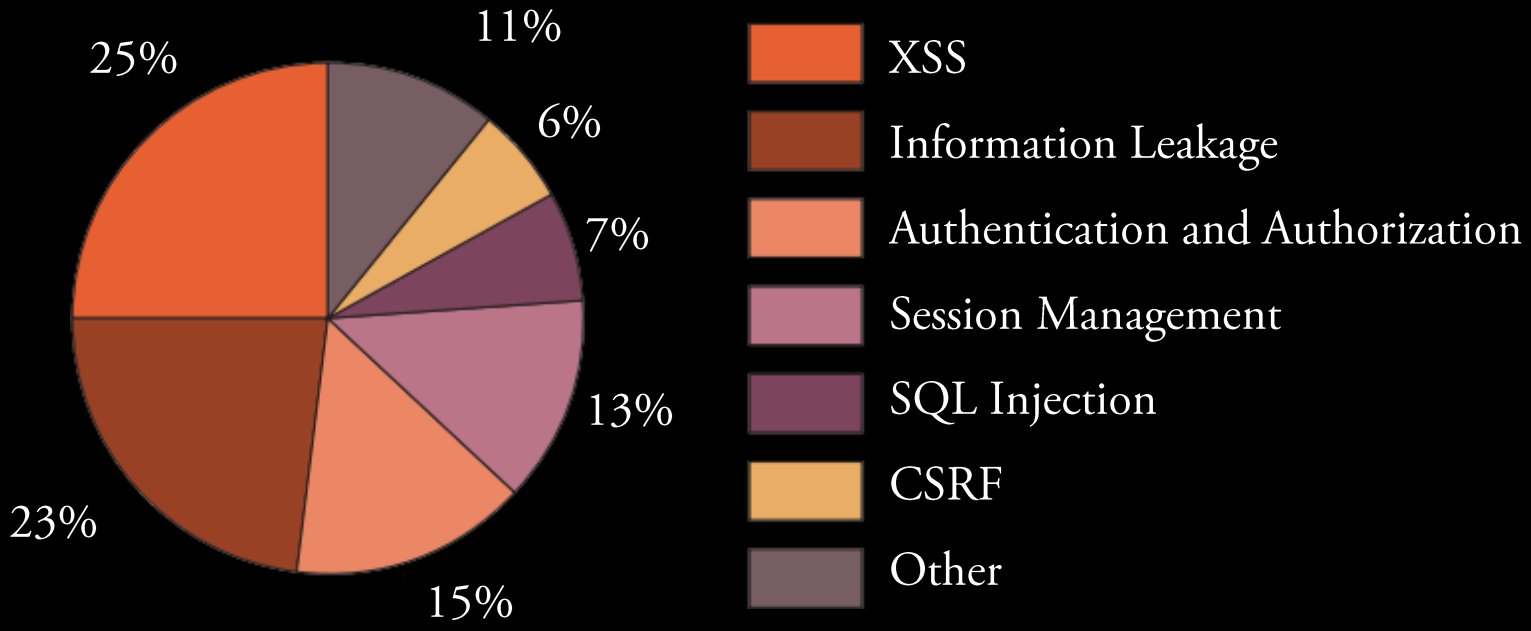
Despite your expectations…

96% of web applications have vulnerabilities*

Median number of vulnerabilities per app

Cenzic Application Vulnerability Trends Report : 2014

*…at the application level



*Cenzic Application Vulnerability Trends Report : 2014

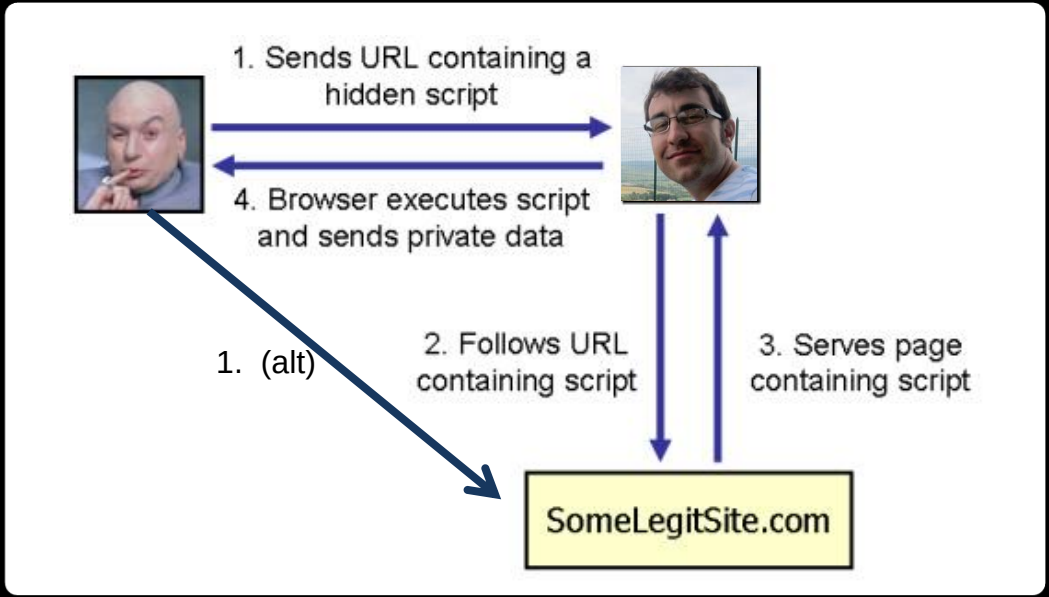
The basic principle behind all web application attacks…

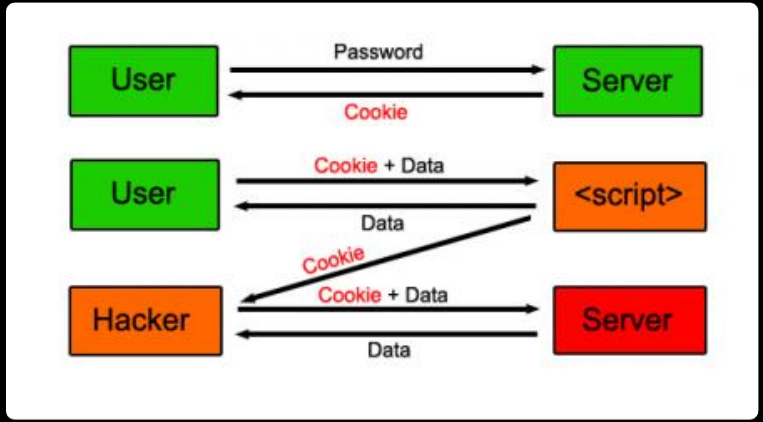
…make the web application do something it was never intended to do.

aka…user input is evil.

Ex: XSS

Allows attackers to inject malicious scripts into web pages and have it executed in another user's browser. Usually to capture some user information.





Ex: Defending against XSS (serverside)

(The problem children)

HTML Entities	
Character	**Encoding**
<	< or <
>	> or >
&	& or &
"	" or "
'	' or '
(	&#40;
)	&#41;
#	&#35;
%	&#37;
;	&#59;
+	&#43;
-	&#45;

Filter and validate all user input before accepting it.
Encode and escape special characters as HTML.

Never trust user input or display raw submissions.

Ex: Defending against XSS (client-side)

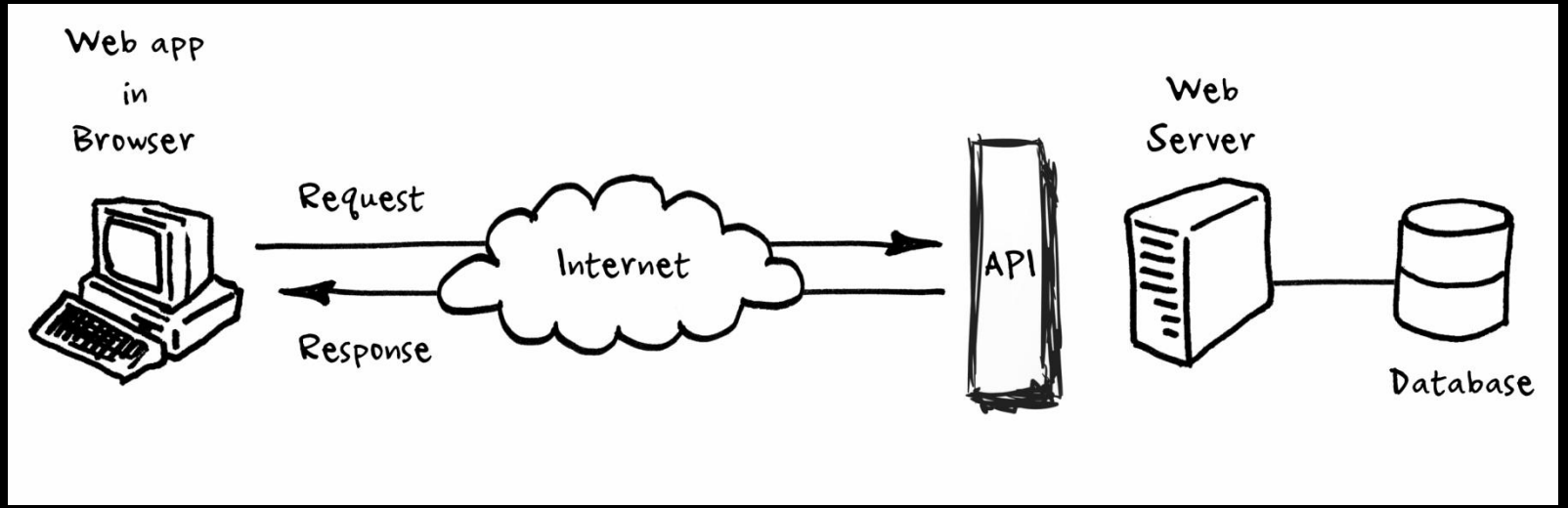
Use Content-Security-Policy

```
Content-Security-Policy: default-src: 'self'; script-src: 'self' somedomain.com
```

Ex: Defending against XSS

Use well tested frameworks

Unique opportunities (and security challenges) arise with modern web 2.0 concepts like REST, client-side MVC, and mobile apps.



REST: REpresentational State Transfer
(GET/POST/PUT/DELETE requests drive your application)

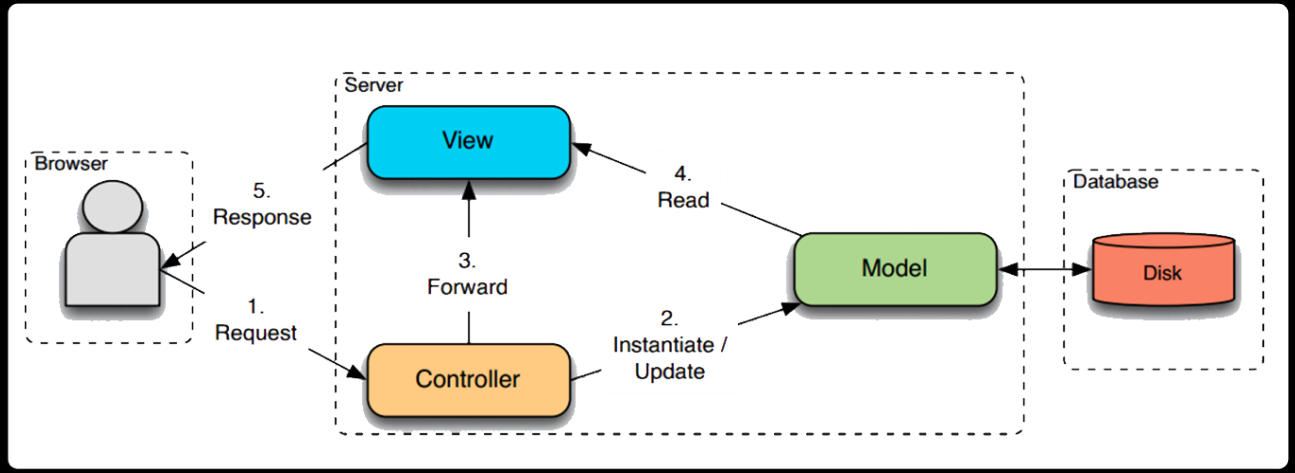
Data is exposed using a REST-ful API
Asynchronous AJAX callbacks used to dynamically load data into the application on the client-side.

Network Tab REST DEMO
http://jsonplaceholder.typicode.com/

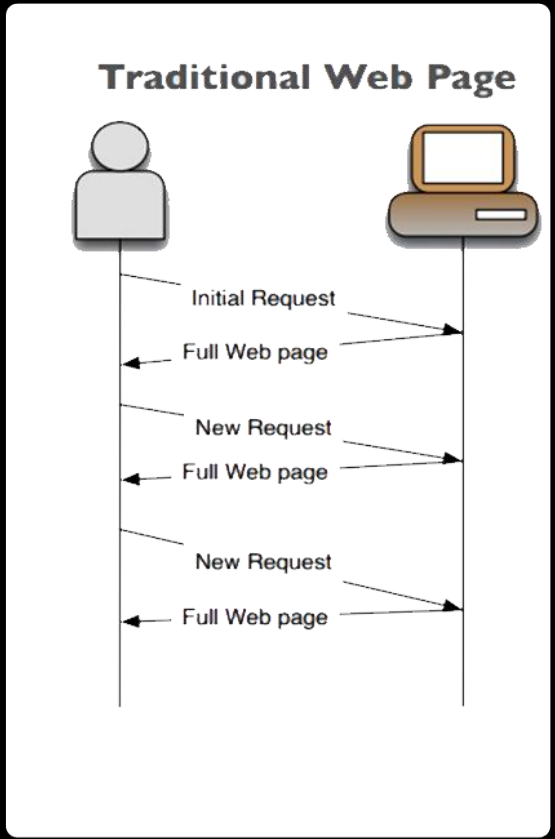
Client-side Frameworks like ember.js, angular.js, and meteor.js follow this architecture, using REST APIs built (typically) on express/node.js or django.

Why is client-side better?

Traditional server-side app

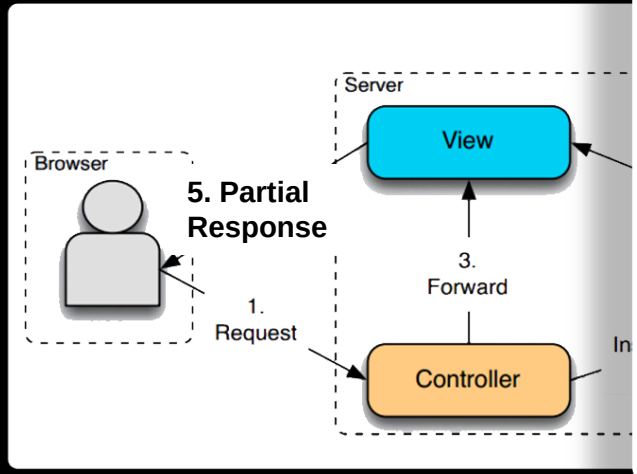


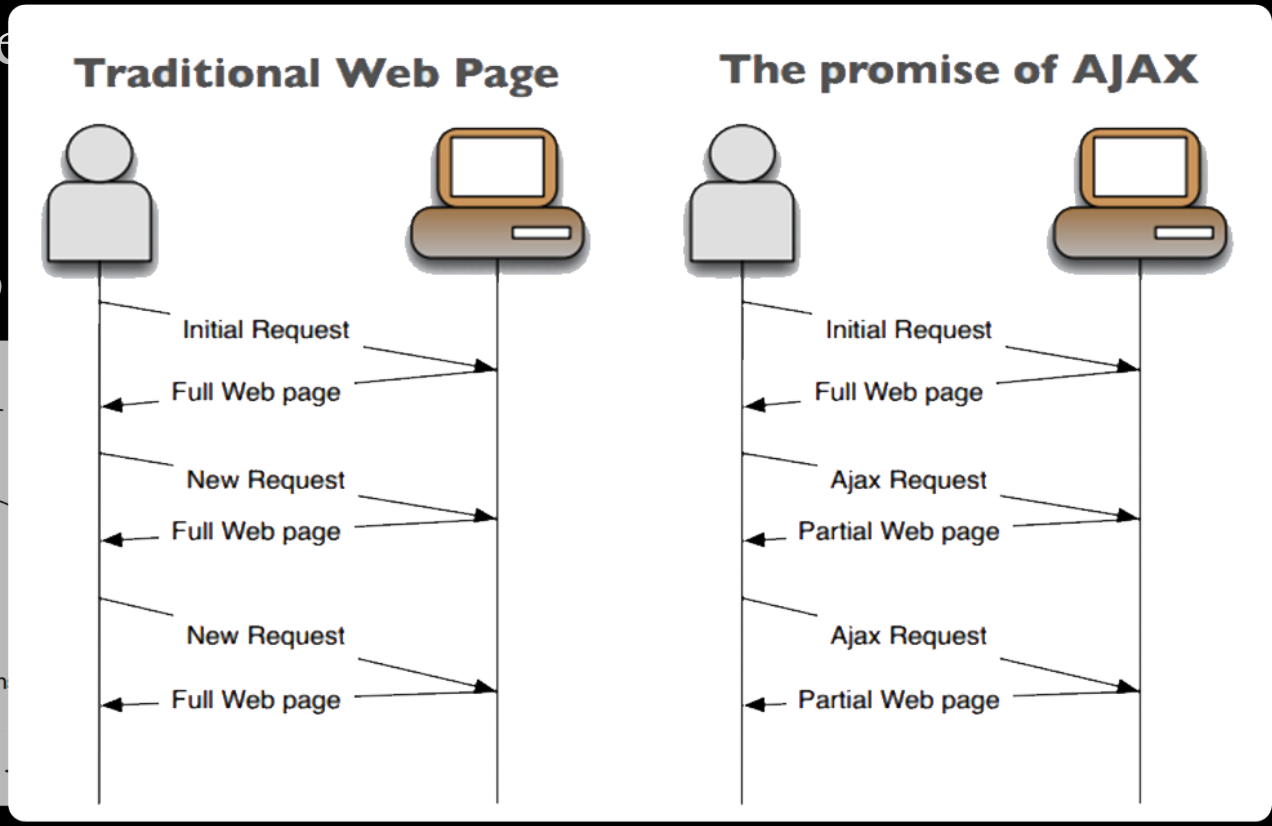
Keeps both the application state and each user's state on the server
If the user requests data to be updated, a complete new page is generated and transferred over the wire to the browser



Why is client-side be

server-side app



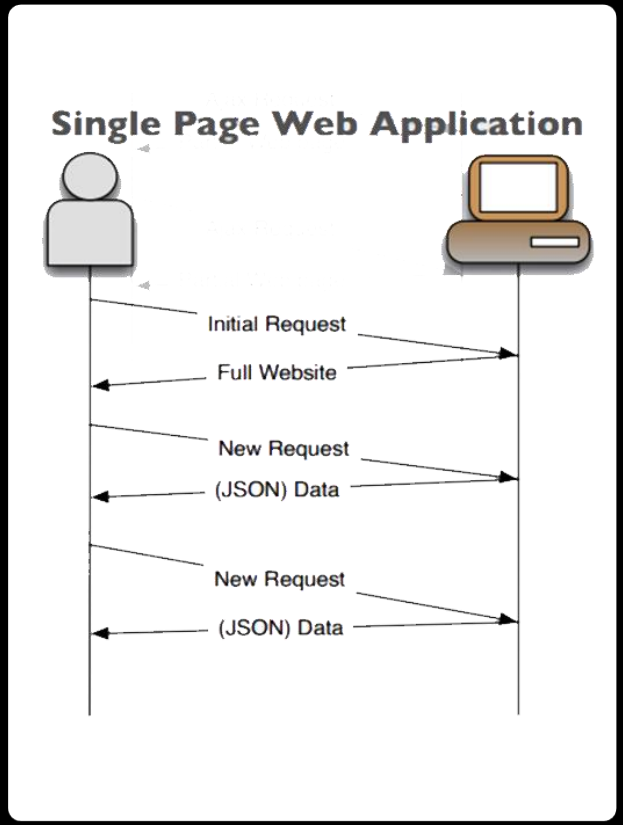


Still Keeps both the application state and each user's state on the server

Still the same number of page requests and processes on the server-side, less overall data transferred results in less bandwidth consumption.

Why is client-side better?

client-side app with RESTful backend API



Application and user state is maintained on the client side.

One full page request at the start of the session. Then just data requests.

Huge reduction in server-side resources, since application only invokes API methods when data is needed, not necessarily whenever a new page is viewed.

Why is client-side better?

Comparison: Click on 10 static pages (100KB/ea) and perform 5 dynamic actions on data (10KB) in each page (with 10,000 daily users).

PER DAY

Server-side	Server-side-with-ajax	Client-side
10*10,000 = 100,000 page loads 10 * 5 * 10,000 = 500,000 loads with data 100,000 *100KB + 500,000 *110KB = 65 GB bandwidth consumed	10*10,000= 100,000 page loads 10*5*10000=500,000 partial page loads 100,000 *100KB + 500,000*10KB = 15 GB bandwidth consumed Faster page transitions	1 * 10,000 page loads 10 * 5 * 10000=500,000 data api calls 10,000 * 100KB + 500,000*10KB = 6 GB bandwidth consumed Instant page transitions

Why is client-side better?

Comparison: Click on 10 static pages (100KB/ea) and perform 5 dynamic actions on data (10KB) in each page (with 10,000 daily users).

The savings adds up…

59 GB / Day
413 GB / Week
21535 GB / Year (21.5TB)

Amazon WS Charges about 8¢ per GB

So… over the year that's about $1722 in savings from pure server-side to client-side app. If you scale up to 100k daily users its $17,220, 1M daily users $172,200…etc
Facebook has 936M daily users
Hence this kind of change would result in about 159 million dollars in savings yearly for them.

Part2:

Ember intro

In this class, you're going to build a client-side app of your own.

Your first app will work with flickr

I'll get you started with the basics,
then you can expand from there.

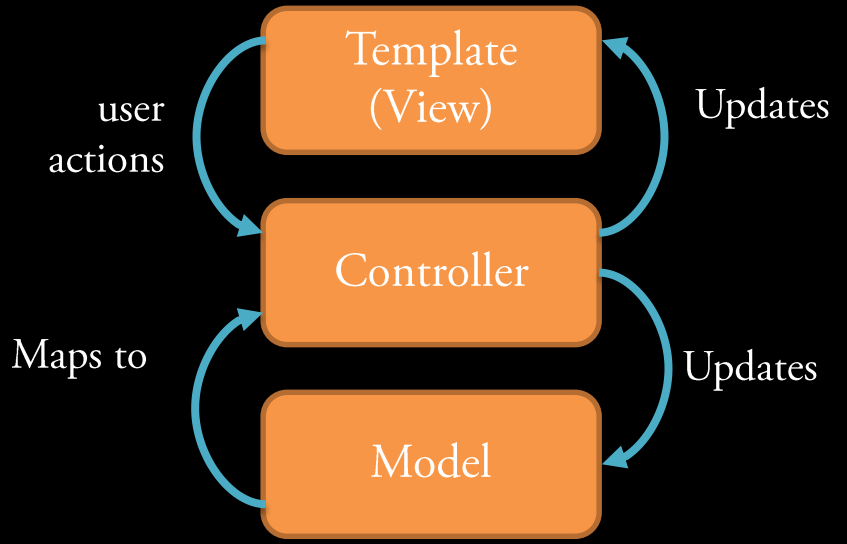
Later you'll come up with your own idea or work one of the service learning projects available
[we will talk more about this]

To build your app, you're going to use a bleeding-edge JavaScript framework called ember.js

Ember.js

What is Ember?

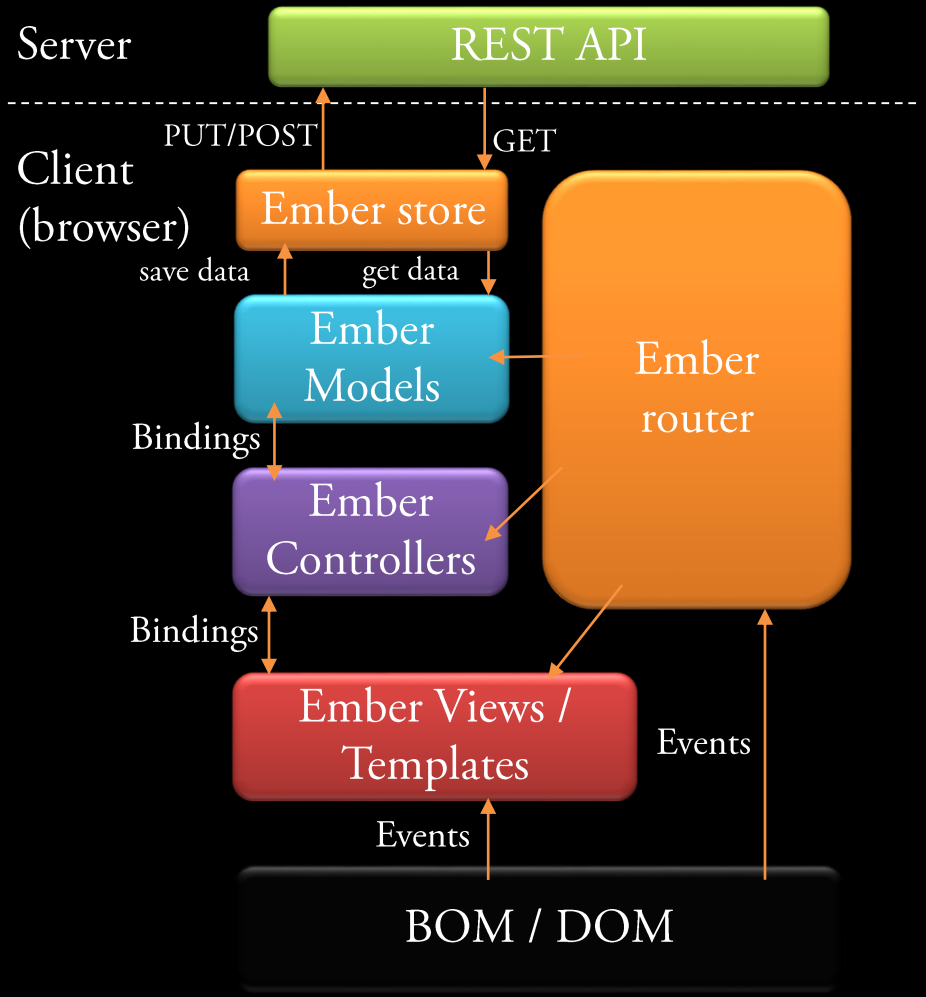
- Browser-based MVC framework
- Eliminates spaghetti code
- Provides a standard (good) architecture for building apps
- Makes hard things (bindings) easy to do



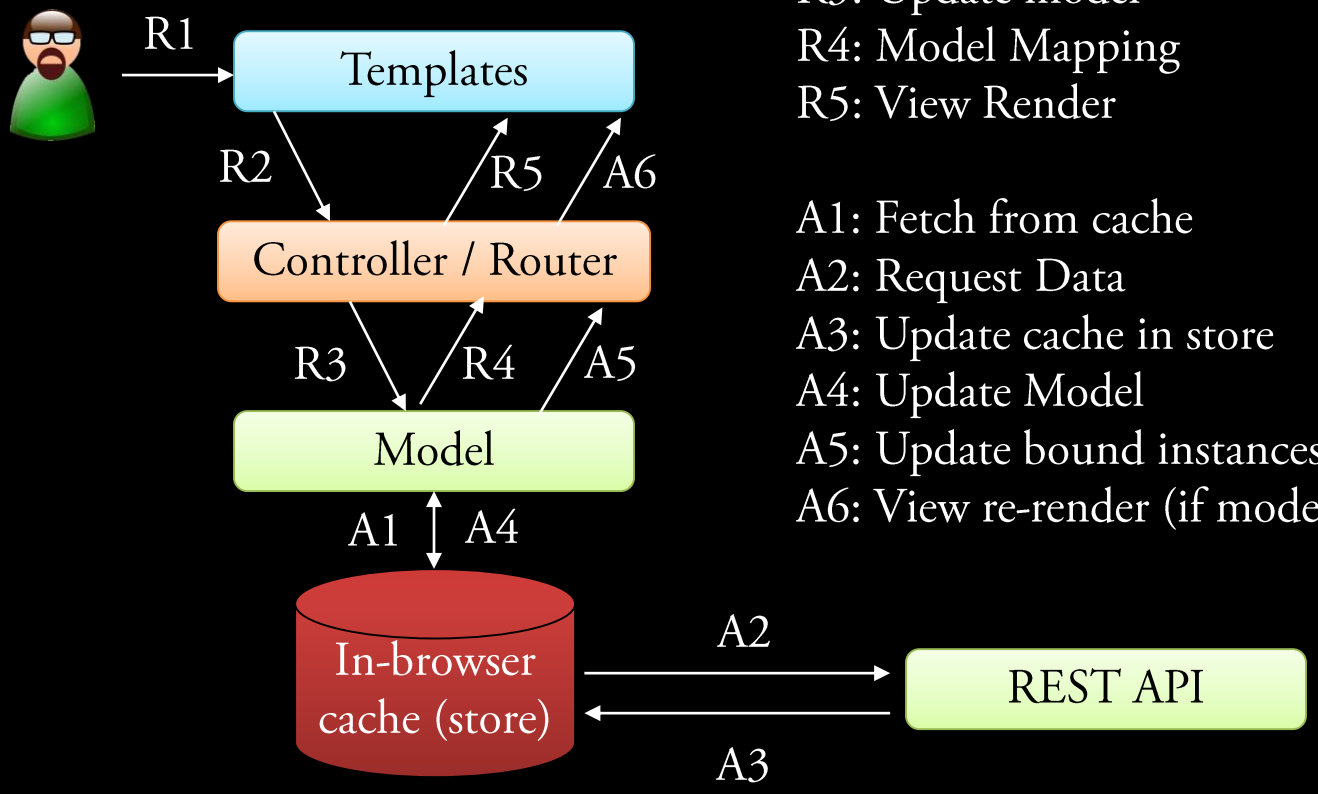
Who uses ember?

Where ember resides

Ember executes purely on the clientside. It requires 0 server side functionality to work. Serverside functionality is only used for getting or persisting data.



client-side app with RESTful backend API



R1: User Event
R2: Invoke Controller Action
R3: Update model
R4: Model Mapping
R5: View Render

A1: Fetch from cache
A2: Request Data
A3: Update cache in store
A4: Update Model
A5: Update bound instances
A6: View re-render (if model is updated)

Ember

Handlebars Templates

Handlebars templates are the user interface in ember. They generate HTML when rendered and maintain bindings to controller/model variables.

- Update automatically whenever bound variables are changed in the controller/model
- Pass user events to controller/router to be handled

Ember

Controller

The controller is the brains of the operation.

- Handles user actions that don't result in a url change.
- Manipulates model data
- Can make AJAX requests to API [we will talk extensively in this class about AJAX]
- Used to control elements rendered in a template/view

Ember

Model

Ember models are just JavaScript objects to represent data on the client side

- They contain various attributes that define your data type
- They can be loaded into and used by your app

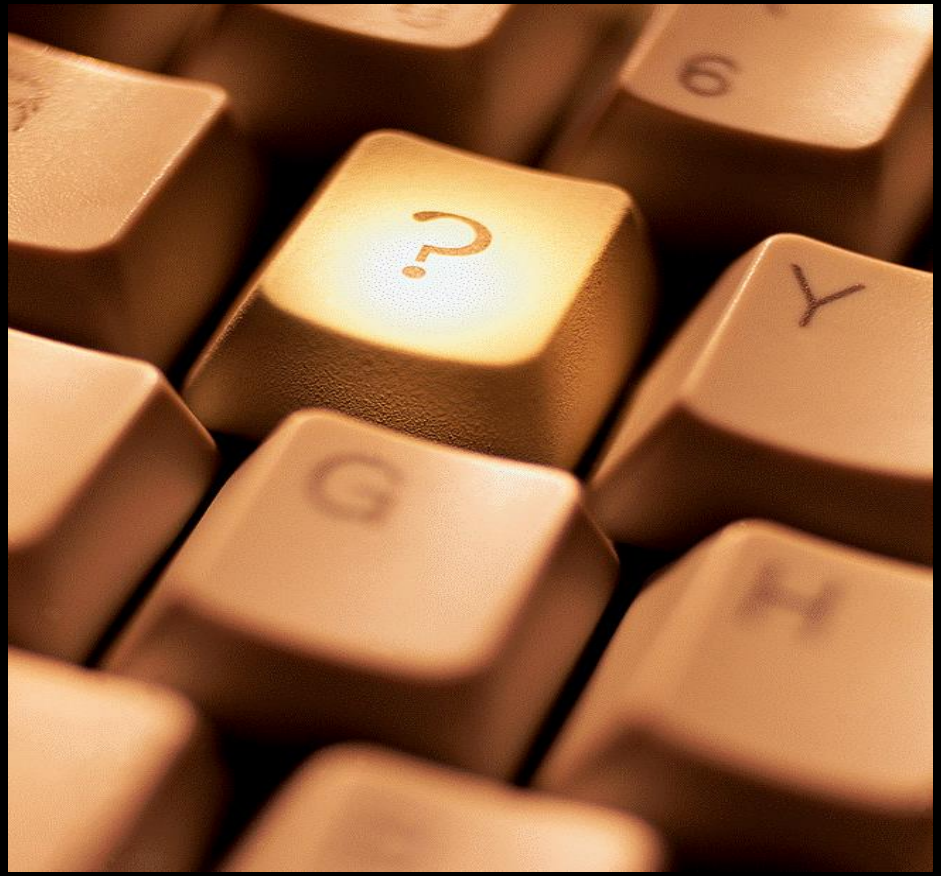
Part 3: Get Started

see (very detailed) development guide

(goto: https://secwebdev.mlhale.com/)

ASK QUESTIONS!

Questions?



Matt Hale, PhD

University of Nebraska at Omaha

Interdisciplinary Informatics

mlhale@unomaha.edu

Twitter: @mlhale_

© 2014-2016 Matthew L. Hale or as listed