A quick [extra] primer on system design
What do we need for social apps?

- Communication
- Collaboration
- Shared state (model)
- Shared view
- Shared control
What do we need for communicative/sync apps?
What do we need for communicative/sync apps?

“P2P” (peer-to-peer)
What do we need for communicative/sync apps?

Good solution?
Why / Why Not?

“P2P” (peer-to-peer)
What do we need for communicative/sync apps?

(client-server model)
What do we need for communicative/sync apps?

Good solution?
Why / Why Not?

(client-server model)
Intro to MeteorJS
What is Meteor?

Javascript MVC framework

- See: “what is a framework?” from the Tools slides
- … summary: Meteor controls the flow. You link things to it.

Provides front-end + back-end support

Front End:
  - HTML + handlebars → connects HTML to JS, allows for variables/functions(!)

Back End:
  - Server-side JS. Wraps up node.js and mongoDB and abstracts them away
Meteor Basics: Files

Client/ → front-end JS code
Server/ → back-end JS code

Files are read automatically. Order: DFS over file structure.
  ● In other words: most-nested files read first
Meteor Basics: Setting Up a Project

Create a new project: `meteor create my_app`

Run with: `meteor [--port 12345]`
Meteor Basics: MVC

- Handlebars({{x}}) let us link HTML to Templates

- Templates (Views) connect HTML to JS variables (Model)

- Per-template Controllers in JS
<div> {{> test}} </div>

<template name="mytemplate">
<p>The current value is: {{myValue}}.</p>
</template>
Meteor Basics: JS

Template.mytemplate.helpers({
    myFunc() {},
    my_func2() {},
})
For now...

- Leave “autopublish” alone (leave it on, which is the default; will use later)

- Use `new ReactiveVar()`

- We’ll only focus on single clients (not connected) for now
example
Programming Assignment #1 (of 2)

Posted today

Due in 2 weeks (Feb 8). (overview next class)