Model-View-Controller (MVC)

[Diagram showing the Model-View-Controller (MVC) framework]

Dan Olsen, Developing User Interfaces (1998)
Model-View-Controller (MVC)

- **Model**: system state
  - This is the core, underlying representation

- **View**: what users see
  - Takes in the model, and generates output to the user

- **Controller**: how stuff gets changed
  - Takes in user input, generates a change in the model

- User side of the equation parallels this…
  - Mental model: user state [model]
  - Actions (input): what the system sees [view]
  - Perceptions: updates user model [controller]
Multi-User Systems
Multi-user Architecture
Model-View-Controller (MVC)
Model-View-Controller (MVC): Multi-User Systems
MVC: **Local** Multi-User Systems
MVC: **Local** Multi-User Systems
MVC: Remote Multi-User Systems
MVC: Remote Multi-User Systems

Requires locking/sync of state on remote server
MVC: Remote Multi-User Systems

Remote systems require locking/sync of state on the remote server.
Questions?
Terms / Concepts
Multi-User Systems

Distributed UIs
Model: Represents Joint+Individual State

The model as a state representation does not change much.

However, with the addition of more users, the state becomes more complex.

The model is responsible for tracking both the overall state of the system, as well as the user-specific state (which may not be shared equally among others).
View: User Specifiers (View Multiplexing)

(implicit specifiers)

(explicit specifiers)
Controller: Joint Input

combined / predicated actions
- user can jointly interact, or modify others' potential interactions, which must be handled by the controller
Questions?

(I’m looking to extend this… how should I? What would you want to know?)
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
Framework vs. Library: What is the difference?

METEOR CLIENT

- Handlebars
- JavaScript
- Mini Mongo

Local Storage

ATMOSPHERE

NPM

HTTP
(Explicit Method calls)

NODE JS
RUNTIME

METEOR
SERVER

- JavaScript
- Mongo

DDP
(Transparent Publications, Subscriptions, Observes)

Mongo DB

Some changes in METEOR 1.0

Source:
https://www.gravitywell.co.uk/
Meteor Basics: Files

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

Files are read automatically. DO NOT “include” your own JS files!
→ 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]`

→ Defaults to port 3000
Meteor Basics: MVC

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

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

- Per-template Controllers in JS
Meteor Basics: HTML / Templates

```html
<div> {{> test}} </div>

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

Template.mytemplate.helpers({
    myFunc() {},
    my_func2() {},
})
Meteor Basics: MongoDB

● “No-SQL” Database

● Wrapped up in Meteor itself
  ○ To reset, use `meteor reset`
  ○ Runs on METEOR_PORT (←-- 3000 by default) +1

● Mini-mongo is the client-local “version” that
For now...

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

- Use `new ReactiveVar()`
example
quiz
Deadlines

Phase 3: due Friday 10/19
Assignment 1: Mon, 10/22
Phase 4: Mon, 10/29
Phase 5: Tues, 11/20
Presentations: Thurs, 12/6
Final Project (Phase 6): 12/11
Assignment 2: Dec.