ZocDoc Tech Talk

Hosted by Harvard Computer Society

February 20, 2013

Talk 1: Writing Maintainable JavaScript Using Backbone, Mustache, & Jasmine

Part 1: A Mustache Template
Mustache provides basic control flow management

Part 2: Backbone.JS
Backbone used to generate models and views.

Backbone Models
- Models manage data/state and perform CPU intensive operations.
- Pass into template to provide variables in the template

Backbone Views
- bind events to functions
- with a view, you listen on the wrapper container, and don't look for the element itself
- cache container and delegate events to it, not elts

Using backbone and mustache, the code can be modular, so it's much easier to change. Plus performance benefits!!

```
var docModel = new Doctor({
      DoctorId: doctorId,
      // etc...
```

```
})
var view = new DoctorView({
      el: $('.showMeTheInfoWrapper'),
      model: docModel
});
```

Now what? We should test the new code!

Part 3: Jasmine - Testing Framework used at ZocDoc

- traditional test framework, assert conditions, mock out function calls
- ZD tests models b/c models maintain state and do heavy computation
- all green, good to go!

Talk 2: Highly Configurable Synchronization

Scheduling

- Synchronizing schedules is a big problem across the board.
- many doctors, many schedules

Initial solutions

- location-based
- store settings per instance of the PMS (practice management system)
- each doctor's computer is running a ZocDoc config of their PMS

Lessons Learned

- all doctors used the system differently, even if its the same system
- multiple practices on one system
- enterprise clients (> 50 practices/locations/multiple PMS systems)
- settings might be per system, or per enterprise, etc and its just unmanageable.
- Everyone does it wrong, so have one-off settings for everyone
- people use the same data in different ways...
- short term fix => KV pair of settings per ZD config

New Solution

- create graph of entity mappings
- prioritize settings based on specificity
    - map specific items based on location, enterprise, settings for doctor, etc.
    - then, you can customize everything for individual doctors! hurray!
- store everything in a KV based pair

What do we get?

- different values for settings
- adding a new setting is easy
- set values for entire systems through inheritance
- want to be able to support everything that a doctor could eventually want

(In)Frequently Asked Questions and Answers

Question: Why are there so many configs?

- If we can make it as configurable, then the doctors can use things as they have already used it. It's very hard to retrain people
- Example: system has concept of location where doctor works, doctors just use this field to mark down where appointments are used.
- Everyone uses the system in a different way, and we need a one-off solution each time. Making a good, configurable solution allows for adaptation and makes it easier to deploy for new doctors.