10 Months with Meteor.js

Tastemade

Who we are

- **Phillip Jacobs -** Runs the Austin Meteor meetup. Lots of experience in enterprise web applications. Now with Tastemade working on our Meteor powered platform.
- **Jason Griffin -** Previously at Pluck, Demand Media, and now at Tastemade. Built web apps for 15 years, managed teams large and small, and built the backend that powers eHow.com, Livestrong.com, and dozens of other sites.

Tastemade

- Our mission is to connect the world through food.
- We make apps and videos for food lovers.
- We have a 5000sqft studio in LA and development office here in Austin
- Just finished our series B.
- We're hiring web, iOS, Android, and QA.

How many have worked with Meteor, MongoDB, Node.js, Handlebars.js?

What is Meteor?

- Front to back platform for building real-time web applications
- The database, server framework, and templating engines are fixed
- Everything is written in Javascript. You can share code between the client and server.

Architecture

- MongoDB/Minimongo
- Node.js
- Sock.js
- Handlebars.js
- Plus plenty of Meteor glue
- At Tastemade we use Coffeescript and Jade HTML templates (compiles to Javascript and HTML)

What kind of apps should you build with Meteor?

- Single page
- Real-time
- Page-based is possible using routing

How many have worked with real-time pub sub?

Pub Sub

- Define collections for each entity type that are shared between server and client
- Use subscriptions to copy a subset of the database to the client based on the current session set

Querying and Updates

- Use regular MongoDB syntax directly on the client
- Insert, Updates, Deletes (only by _id)

Demo - models, collections, pubsub

Latency compensation

- Updates are applied on the client first and immediately redrawn
- Changes are propagated to the server and stored
- Changes are then broadcasted to other clients

Reactive Templates

- Access collection from the template. When the data is updated the template is automatically redrawn
- Handlers for events
- Reactivity isolation
- Constant regions

Session

- Also supports client side session state that are also reactive variables
- Use to track the state of the application

Router

- Built-in router
- Backbone router
- Supports HTML push state
- Uses session state to track the route
- At Tastemade, when you click on a link we use the backbone router to set our route (_id) session variable

Rendering Cycle

- Alter session variables or save updates to a collection
- Triggers redraws and updates to subscriptions
- When the new data arrives on the client that can further cause redraws

Demo - real time, templates, session variables

Methods

- How do you talk to services outside of Meteor?
- Not everything has to run the client.
- Define callable methods on the server that act like traditional REST calls
- For example we wrote a stripe integration where the code runs on the server

Security & Accounts

- Built-in User collection (Meteor.User) that integrates with the account system
- Current user record is a reactive data source
- Collections have ACLs that run on the client and server
- All the usual accounts api (login, reset password, signup, etc.)
- Built-in UI for accounts management or roll your own
- Supports SSL with the force-ssl package

Packages

- Meteor packages are different because they inject code on the client and server
- atmoshpere.meteor.com
- Support for NPM packages on the server

Testing

- Tinytest - simulates the meteor runtime

Deployment

- Meteor.com
- Use the bundle command to generate a plain node.js application
- Heroku buildpacks
- AWS

Hot code pushes

- Ensures all the clients are running the current code
- For connected clients, page automatically reloads and maintains all session and form variable state
- Automatically versions JS and CSS

Demo - Stripe integration, atmosphere

How does it compare to Backbone and Angular?

- Handles the backend and front end. Larger more integrated stack.
- Compared to backbone, it's not MVC. Eliminates duplicate code between server and client. Backbone isn't reactive. You still need to write the code to update the DOM.
- Compared to angular.js, doesn't require adding attributes to the HTML. Must integrate with the server.

Mindshift

- Database in the client
- Use pub sub to move a subset of the database to the client. Challenging when you try to implement infinite scroll or “get more docs”.
- You only query the database in the client
- Reactivity - don't update the DOM with JQuery. Let Meteor do the work.

Pros

- There's only one front end, one backend, and one database choice. Your limited, but they work really well together.
- Very quick development
- Very little server code

Cons

- There's only one front end, one backend, and one database choice. Limits your choices.
- **Deployment -** the Meteor hosted deployment is not very good. Hosting on Heroku or AWS is better, but requires some work
- **SEO** - the "crawlable" package is not very good.
- **Testing -** like any browser heavy application it can be difficult to test

Criticisms

- **Fibers** - Meteor is trying to make node development accessible similar to Ruby on Rails. They use Fibers to avoid nested callbacks at the expense of some performance.
- **NPM Packages -** now supported
- **Security -** now supported

Resources

- **Eventminded.com** - https://www.eventedmind.com/
- **Meteorite** - https://github.com/oortcloud/meteorite
- **Discover Meteor book** - http://www.discovermeteor.com/
- Meteor IRC channel on freenode

Questions?