Meteor.js

Quick doesn’t have to be dirty

Alex Westholm, Third Reef Partners

www.thirdreef.co

acw@alexwestholm.com

Likes: Haskell, Clojure, Surfing

What is Meteor?

JS full-stack framework + Two-way data binding + Reactive front-end framework

= Elegantly simple

More specifically:

Simple client-side templating framework	MongoDB everywhere
DDP protocol: two way databinding	Node.js and Fibers

Client-Side Templating

- Handlebars compatible (nesting, control flow)
- JavaScript hooks for:
  - helpers
    - Template.name.helpers({x: function(){ … }});
  - lifecycle events
    - Template.name.onRendered(function(){ … });
  - page events
    - Template.name.events({‘click #foo’: function(){ … }});

```
<body>
  <h1>Surf Travel: Can I use American
Dollars?</h1>
  <main>
      {{> surfDestinations }}
  </main>
</body>

<template name="surfDestinations">
  <ul>
      {{#each countries}}
      <li>
      {{location}} -
      {{#if payInDollars}}
           Yes
      {{else}}
           No
      {{/if}}
      </li>
      {{/each}}
  </ul>
</template>
```

```
if(Meteor.isClient){
  Template.surfDestinations.countries = [
      {location: "Hawaii", payInDollars: true},
      {location: "Puerto Rico", payInDollars: true},
      {location: "Indonesia", payInDollars: false},
      // just 'cause you can doesn't mean you should...
      {location: "Costa Rica", payInDollars: true},
      {location: "France", payInDollars: false},
      {location: "South Africa", payInDollars: false},
  ];

  Template.surfDestinations.events({
    'click li': function(e){
      e.target.style.backgroundColor = 'blue';
    }
  });
}
```

MongoDB & Client-side Emulation

- Mongo - the Meteor of datastores
  - No schema
- Database Everywhere: collections can live on client, server or both
- Simple API: find, findOne, insert, update

```
  <main>
    {{> surfDestinations }}
    {{> destinationForm }}
  </main>
</body>

<template name="destinationForm">
  <form>
    <label for="location">Country:
</label>
    <input type="text" name="
location">
    <label for="payInDollars">Pay in
Dollars?</label>
    <input type="radio" name="
payInDollars" value="Yes"> Yes
    <input type="radio" name="
payInDollars" value="No"> No
    <button>Add Destination</button>
  </form>
</template>
```

```
Destinations = new Meteor.Collection("destinations");

if(Meteor.isClient){
  Template.surfDestinations.countries = [ … ];
  Template.surfDestinations.helpers({
    countries: function(){
      return Destinations.find();
    }
  });
  Template.surfDestinations.events({ …,
    'submit form': function(e){
      e.preventDefault();
      var useDollars = e.target.payInDollars.value ==
"Yes";
      Destinations.insert({
        location: e.target.location.value,
        payInDollars: useDollars
      });
      e.target.reset();
      return false;
    }
  });
}
```

Two-Way Data Binding

- Distributed Data Protocol
- Sort of like a specialized AMPQ
  - Publish/Subscribe
  - Plus simple RPC
- JSON based
- Transport is WebSockets or SockJS
- Security: insecure and autopublish

```
$ meteor remove autopublish
$ meteor remove insecure
```

```
if(Meteor.isServer){
  Meteor.publish("countries",
function(){
    return Destinations.find();
  });
}

Meteor.methods({
  addDestination: function(country,
useDollars){
    Destinations.insert({
      location: country,
      payInDollars: useDollars
    });
  }
});
```

```
Destinations = new Meteor.Collection("destinations");
if(Meteor.isClient){
  Meteor.subscribe("countries");
  Template.surfDestinations.helpers({
    countries: function(){
      return Destinations.find();
    }
  });
  Template.surfDestinations.events({ …,
    `submit form': function(e){
      e.preventDefault();
      var useDollars = e.target.payInDollars.value ==
"Yes";
      Meteor.call("addDestination", e.target.location.
value, useDollars);
      Destinations.insert({
        location: e.target.location.value,
        payInDollars: useDollars
      });
      e.target.reset();
      return false;
    }
  });
}
```

...and then

- Meteorpad
- Ecosystem: atmosphere
- Sychronous programming - Fibers
- Scalability
  - Mongo oplog
  - Avoid publications tied to users
- Mobile / Cordova