Having Fun

BUILDING WEB APPLICATIONS

DAY 2

17 DEC 9AM-5PM

CATCHING UP...

Source files from Day 1 can be found on Dropbox, make sure that it works for you

We have created a functioning app yesterday, but it runs only in browser memory and application state is not persisted (no backend)

PROBLEMS FACED):

- manual template rendering/injecting is cumbersome
- super inefficient to reload all the “panels” every time application state changes
- Changes are propagated imperatively, but declarative is better!
- HTML is gonna become long and hard to maintain

A METEOR HAS LANDED!!

is a cohesive development platform, a collection of libraries and packages that are bound together in a tidy way to make web development easier

WHY METEOR?

ONE LANGUAGE
JS from front to back

REAL TIME
with minimal effort

PRODUCTIVE
Behind the hood!

- Auto imports everything
- Hot-reloading
- Templating engine included

COMMUNITY
there is a package for
EVERYTHING

METEOR

Web

Mobile

App Components & Logic

Blaze

Angular

React

CLIENT

Client Data Cache

DDP

SERVER

Livequery

App Microservices

SOURCES

Database

REST Services

Commonly used commands

meteor create <name>	Create a new Meteor project in /<name>
meteor run	Runs your application on localhost:3000. Changes are detected and hot reload!
meteor add <package>	Add packages to your Meteor project.
meteor reset	Reset the current project to a fresh state. Removes the local mongo database.
meteor deploy	Deploy the project in your current directory to Meteor's servers. You'll need an account for this

EXERCISE 1

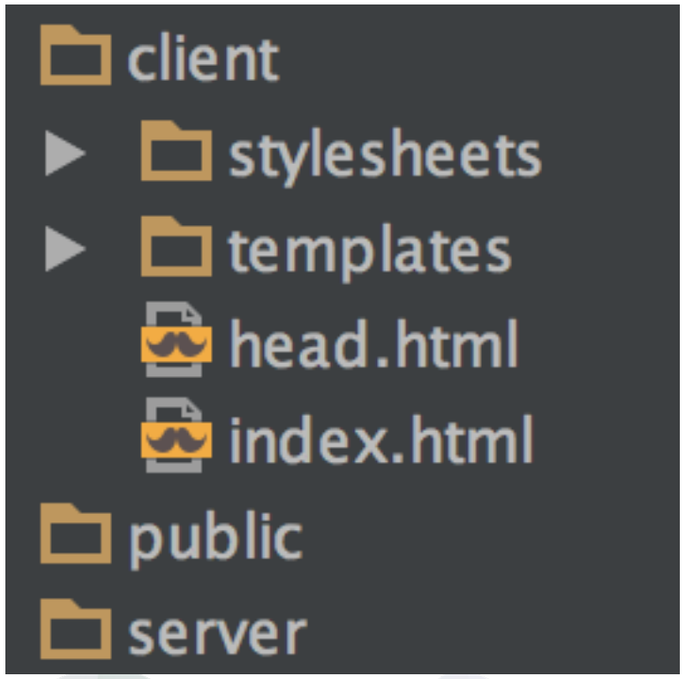
Install and create new meteor project

APP STRUCTURE

Getting organised

Special Directories

/client	Files here are not run on the server, and automatically concatenated in development
/server	Code here is not sent to the client. Put your sensitive code here!
/public	Place public assets like images here
/	Everything else in the root directory. Place common code like collections here



EXERCISE 2

Find and add packages
- Semantic UI
- Underscore

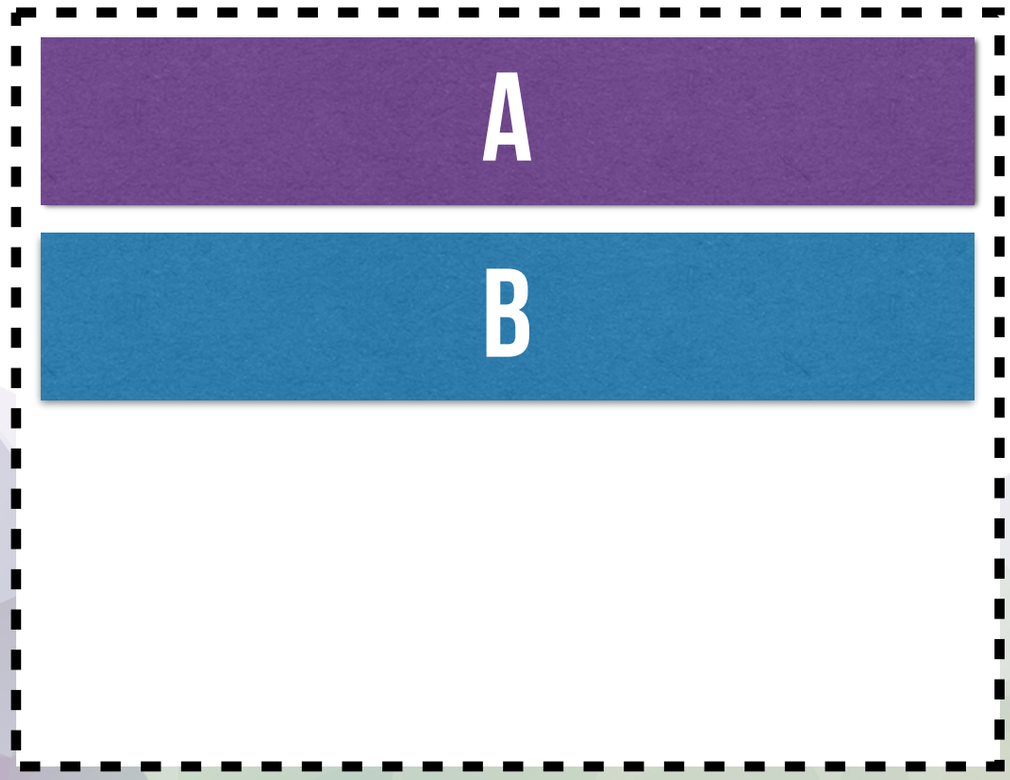
TEMPLATES

Instead of one HTML file, we compose a page with templates

```
<template name="A">
  ...
</template>
```

```
<template name="B">
  ...
</template>
```

```
<body>
  {{> A }}
  {{> B }}
</body>
```



SPACEBARS

You may use Mustache-like {{helpers}} to bind data

Reactive and automatic

HTML

```
<template name="A">
   <p>{{something}}</p>
</template>
```

JS

```
Template.A.helpers({
   something: function() {
      return ??
   }
})
```

if ?? is reactive, then any changes will flow to the UI automatically

- Session Variables
- Database queries
- Reactive Variables

TEMPLATE CALLBACKS

You can register functions that will be called during the template lifecycle

`onCreated()`
useful for declaring template variables

`onRendered()`
useful for plugin initialisation

`onDestroyed()`
used for cleanup

EXERCISE 3

Convert HTML into Meteor templates
Use template helpers to bind data
Initialise plugins within onRendered() callback

EVENT MAP

We map browser events to functions with the Event Map

```
Template.<template name>.events({
  '<eventType> <selector>': function()
    eg. click, change   same as jQuery

      this -> data context of target element
      use console.log(this) to see for yourself
  }
})
```

REACTIVE DATA FLOW

TEMPLATE HTML

```
<button class="trigger>
  {{ something }}
```

SESSION

something	'Something here!'

TEMPLATE JS

HELPERS

```
Template.<template name>.helpers({
  something: function()
    return Session.get('something');
  }
})
```

EVENT MAP

```
Template.<template name>.events({
  'click .trigger': function()
    Session.set('something', 'Something here!');
  }
})
```

EXERCISE 4

Add *Check Compatibility* to Event Map
Update contents of modal through Session object

MORE METEOR MAGIC

ACCOUNTS

Meteor comes with some great packages for managing user accounts (plug and play!)

```
> meteor add accounts-password
> meteor add iandouglas:accounts-ui-semantic-ui
```

```
Meteor.user()      Returns currently logged in user or null
Meteor.userId()    Returns id of currently logged in user or null
{{currentUser}}    Template helper that calls Meteor.user()
{{#if currentUser}}  Render something only if user is logged in
Accounts.ui.config() Specify options
```

EXERCISE 5

Set up a user account system

EXERCISE 6

Conditional display of page elements, depending on whether user has logged in

MONGO DB

Relational Database

use SQL to query

data must be stored in tabular form - rows and columns

data is normalised

Document Database

use javascript API to query

store all relevant data together in single "document" in JSON, which can nest values hierarchically

data is not normalised

METEOR COLLECTIONS

Meteor stores data in Collections

`<- Meteor.users`

The special thing about collections in Meteor is that they can be accessed from both the server and the client

Views backed by a Collection will be updated reactively!

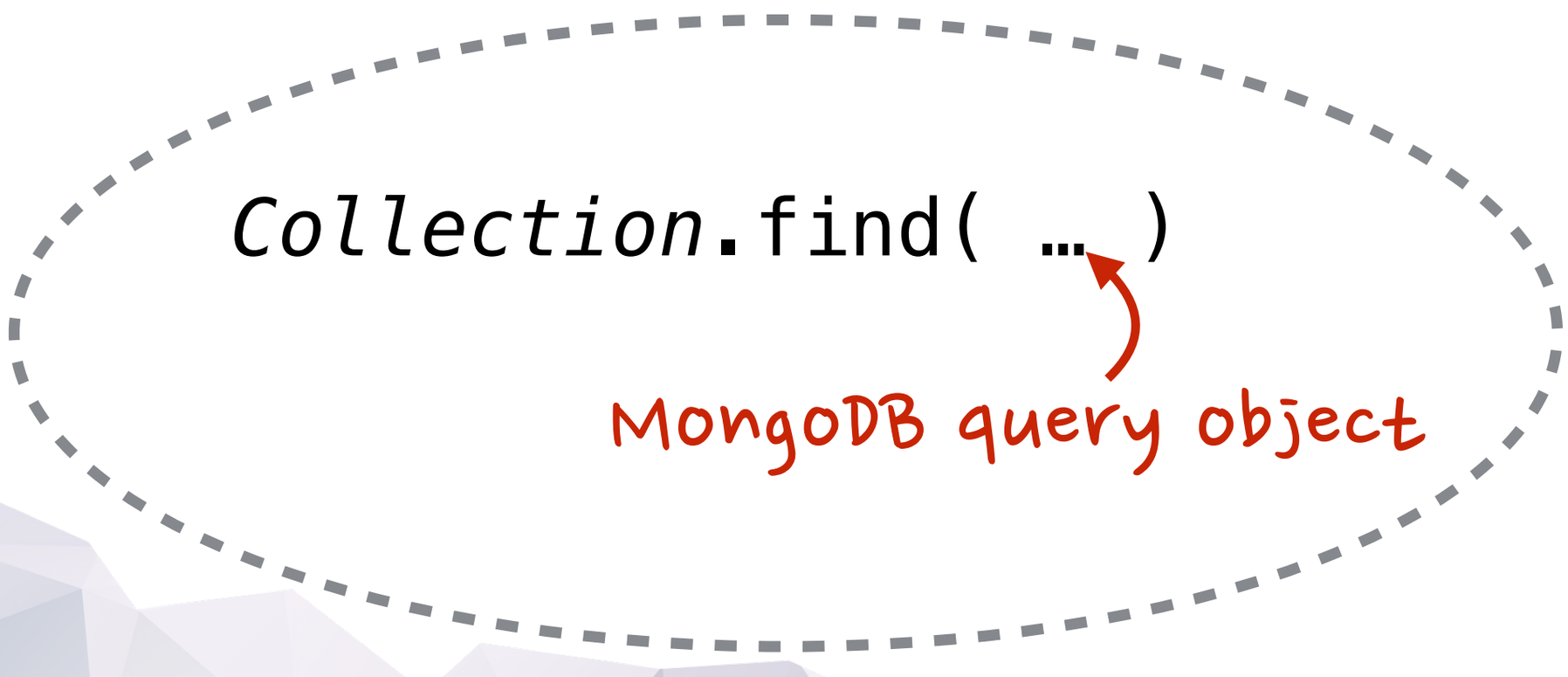
EXERCISE 6

Adding a form to add/update profile details
Save profile data to Meteor.users Collection

EXERCISE 8

Query data from MongoDB, un-hardcode

GETTING DATA FROM COLLECTIONS



REACTIVE ENTITY!

EXERCISE 9

Increment likes and views

EXERCISE 10

Deploy! Finally!!

END OF DAY 2

https://clarencengoh.typeform.com/to/kjByjA