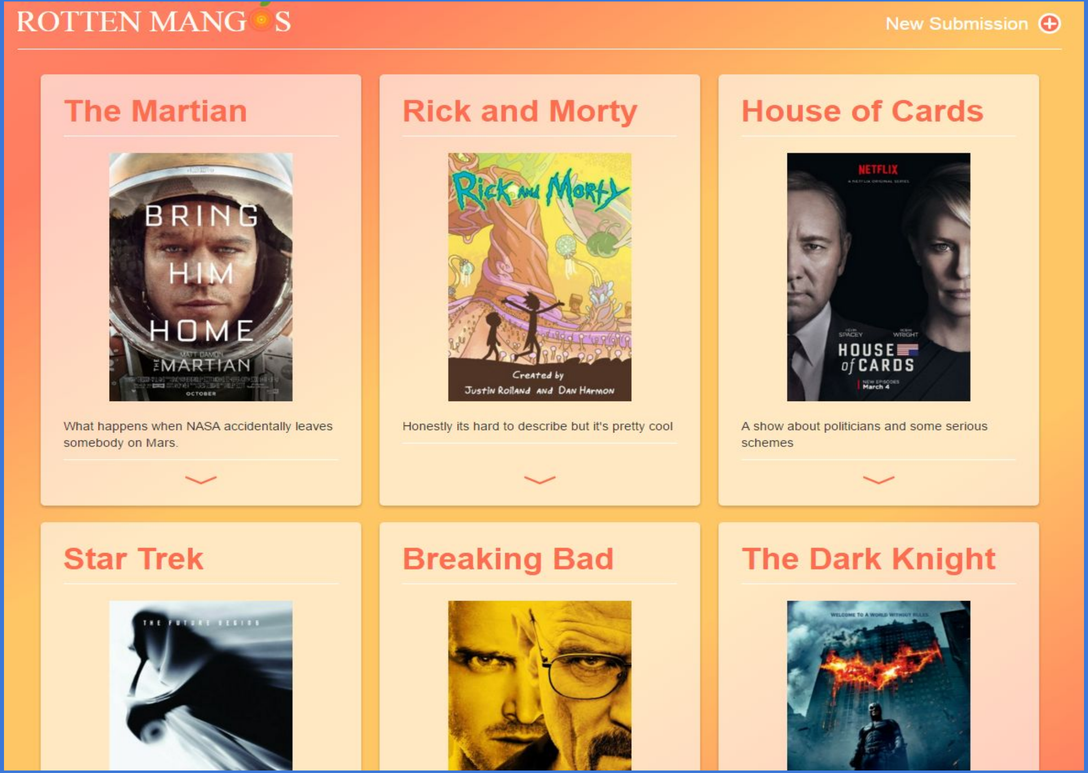
Boiler Camp

What you'll be making...



ROTTEN MANGOS

New Submission

The Martian

What happens when NASA accidentally leaves somebody on Mars.

Rick and Morty

Honestly its hard to describe but it's pretty cool

House of Cards

A show about politicians and some serious schemes

Star Trek

Breaking Bad

The Dark Knight

Here’s the plan

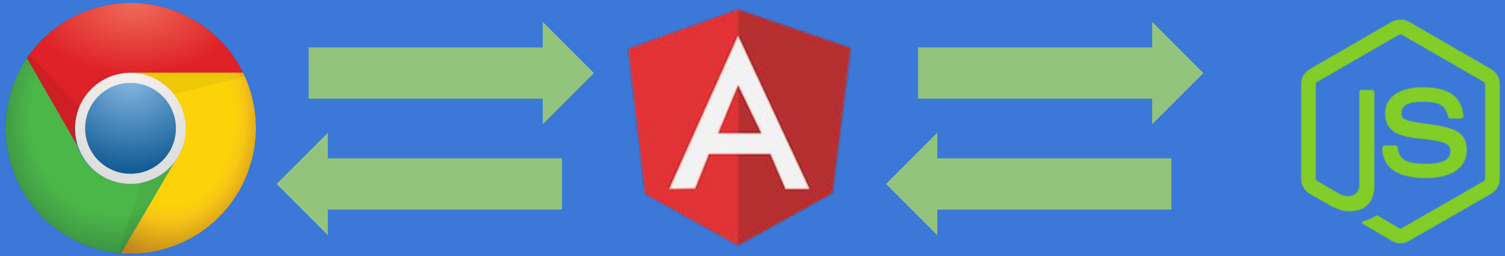
1. Intro to APIs and JavaScript
2. Intro to Node.js
3. Lunch!
4. Intro to AngularJS
5. Go home with an awesome web app!

It’s okay if you don’t understand everything!

It’s okay if you don’t finish every section!

Please ask for help!
(Our mentors are very nice!)

How will it work?



boilercamp.org/go

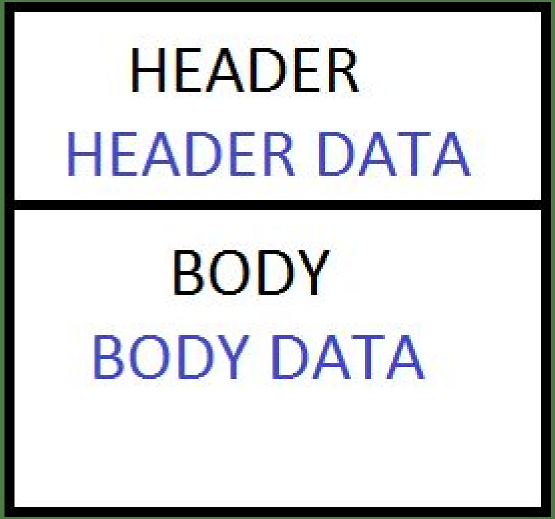
What is an API?

What is an API?

- "Application Programming Interface"
- It is a way for us to connect our own app with various other softwares
  - i.e. Google Maps API lets us use features of Google Maps in our own app
- Why use it?
  - Allows us to piggyback off the hard work of people way smarter than us so we can do cool stuff with their product

HTTP Requests

- Your browser sends an HTTP request to a server
- It has a header and body
  - These have multiple fields that help get the response you want
  - Contains the method (i.e. POST or GET)
- Server sends back an HTTP response



POST and GET requests

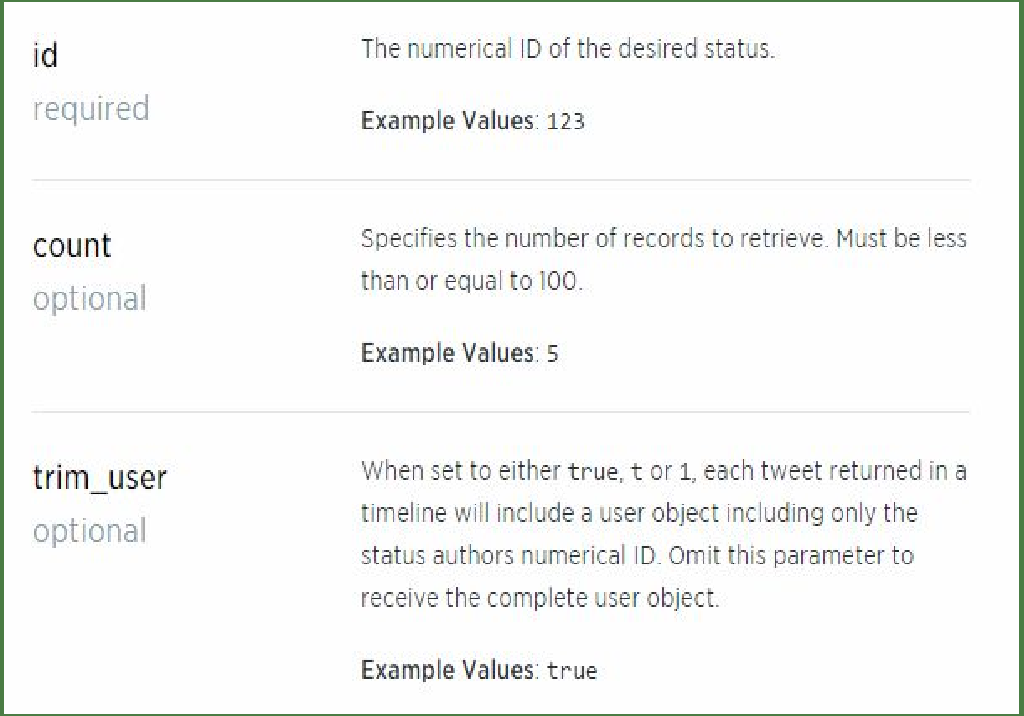
- Do what they sound like: post and get stuff
- POST = send data
- GET = receive data
- Allows for CRUD operations
  - create, read, update, delete
  - pretty much anything you would really want to do with some data

What is a RESTful API?

- “Representational State Transfer” (REST)
- A standardized way of transferring data
  - Doesn’t matter what programming language you use as long as you send information in a specific format (i.e. JSON, XML etc.)
- Tons of API’s use this format: Facebook, Twitter, Twilio, Instagram etc.

Example with Twitter API

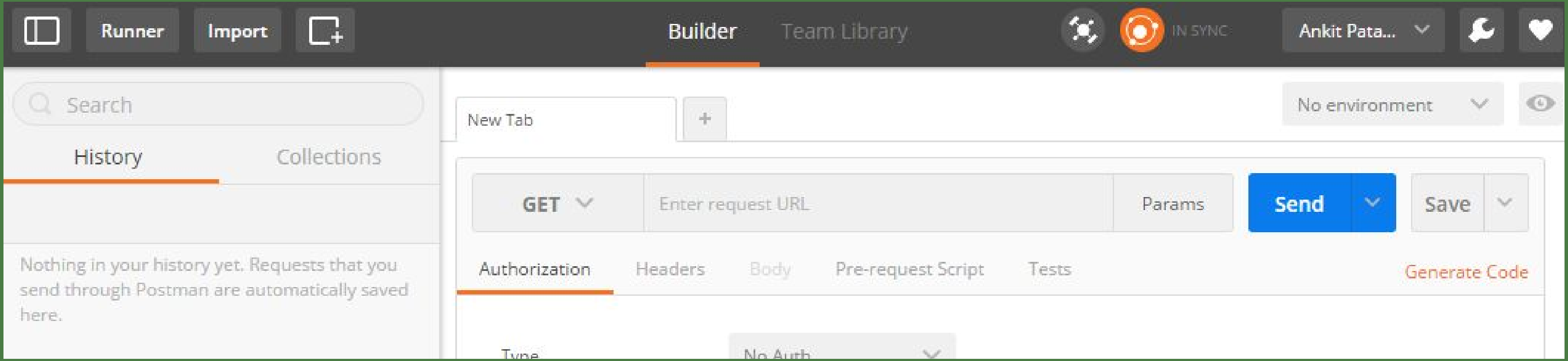
- https://api.twitter.com/1.1/statuses/user_timeline.json
- Pass in extra parameters in "body"



id required	The numerical ID of the desired status. **Example Values**: `123`
count optional	Specifies the number of records to retrieve. Must be less than or equal to 100. **Example Values**: `5`
trim_user optional	When set to either `true`, `t` or `1`, each tweet returned in a timeline will include a user object including only the status authors numerical ID. Omit this parameter to receive the complete user object. **Example Values**: `true`

Postman

- A nice tool for us to make API calls
  - POST/GET requests
- Has a nice place for authentication, parameters and other cool stuff



Nothing in your history yet. Requests that you send through Postman are automatically saved here.

Oauth

- An easy way to publish and interact with protected data
  - i.e. your Twitter account
- It’s a safe way to “log in”
- You need to use this in order to access certain things
  - i.e. various API’s
- Don’t worry about it too much

Postman Activity

Postman Activity

- Download and use Postman to make POST/GET requests using the Twitter API
- Detailed instructions will be provided on the wiki page
  - boilercamp.org/go
- Please ask for help if you are still confused, mentors are here for a reason!

boilercamp.org/go

Takeaways

- What an API is
- What a POST/GET request is

Javascript

Why do we care?

- It’s hip
- It runs in the browser OR uses special Node.js magic for server side development

Basic Syntax

- Looks similar to Java
  - But they are not the same thing!
- No data types, everything is a var
  - Dynamic Typing

Functions

- A function in is like a method in Java
- Functions are considered objects
- Can be stored in variables
- Can be passed in as parameters to other functions like strings, ints etc.

```
function functionName() {
    //your code goes here
}
```

Anonymous functions

- A function without a name
- Stored in variable or self invoking functions usually

```
var x = function() {
    /*your code goes here*/
    };
```

Callback Functions

- Anonymous functions are frequently used as parameters for other functions
- You can just as easily just define a function and pass in it’s name
- When we pass in a function as an argument, remember that you are only passing in the definition!
  - Have to call the function to actually have it execute something

```javascript
function helloWorld() {
    console.log ("Hello World!");
}

function test(callback) {
    callback();
}

function run() {
    test(helloWorld);
}
```

Back-End

Using Node.js

What is NodeJS?

- Let’s JavaScript run outside the browser
- It’s super fast, scales well, and can handle lots of requests

npm

- npm lets us install packages we want instantly
- **npm init** will get the party started (package.json)
- **npm install <something>** will install that thing
- **npm install <something> --save** will install it, and remember for later

Export & Require

- We can use require() to load data
  - Can load a package we installed with npm
  - Can load a file we created
- Require() always loads the ‘module.exports’ variable
  - module.exports = { someData: x }
  - module.exports.someData = x;

Export & Require

```
//server.js

x = require(myFile.js);

x.someData
```

```
//myFile.js

module.exports = {

   someData:  y

};
```

Express

- Express is a package that makes it easy to create a server
- Important Express Functions
    - get(url,callback) runs the callback function when we receive a GET request
    - post(url,callback) runs the callback function when we receive a POST request
    - listen(port,callback) runs our server, then runs the callback when it's started

Express Server

```
var express = require('express');
var app = express();
app.get('/',function(request,response) {
    response.send("Hello World!");
});
var server = app.listen(8080,function() {
    console.log("Server is listening on port 8080!");
});
```

What is MongoDB?

- A cross-platform NoSQL database
- Uses JSON “documents” instead of structured tables/rows
- Allows for “flexible schema”

What is Mongoose?

- A npm package that lets us work with mongoDB easily
- Lets us model data with simple schema
- Lets us easily validate data before saving

Schema

```
var mongoose = require('mongoose');
var catSchema = new mongoose.Schema({
  age: Number,
  name: String,
});
module.exports = mongoose.model('Cat', catSchema);
```

Validation

```
var mongoose = require('mongoose');
var catSchema = new mongoose.Schema({
  age: {
    type: Number,
    required: [true, "Missing Age!"],
    min: [0, "Age can’t be negative!"],
  },
});
```

Coding Time!

Front-end with AngularJS

What is the front end anyway?

The part of the web app that the user interacts with.

Appearance:

- HTML (text)
- CSS (style)

Behavior:

- JavaScript

In this project, you won’t have to worry about writing very much HTML or any CSS. We will focus on Angular.

What is AngularJS?

Angular is a framework for front-end JavaScript

- Creates <u>responsive</u> (fast) websites
- Adds interactivity to HTML through *directives*

First thing’s first: Modules

Where we write pieces of our Angular Application

Our very first lines of code:

app.js

```
var app = angular.module('movies', []);
```

index.html

```
<html ng-app='movies'>
```

Angular Controllers

Defines app’s behavior with values and functions
Think of it like a class in Java

app.js:

```
app.controller("MoviesController", function() {});
```

index.html:

```
<body ng-controller='MoviesController as collection'>
```

Expressions

Allow you to insert *dynamic* values into your HTML code
Very important for rendering info from our JavaScript

Simple example:

```
<p>
  I am {{ 4 + 6 }}
</p>
```

Angular magic

```
<p>
  I am 10
</p>
```

Directives

A directive is a marker on a HTML tag that tells Angular to run or reference some JavaScript code

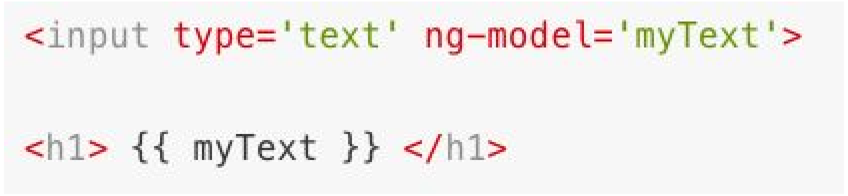
Built-in directives start with *ng*

Common directives:

- ng-app, ng-controller
- ng-model
- ng-repeat

Dynamic binding with Directives

Page updates in real time



```
<input type='text' ng-model='myText'>

<h1> {{ myText }} </h1>
```

More directives

app.js

```
var states = ["Alabama", "Alaska", "Arizona", "Arkansas"]
```

index.html

```
<ul ng-repeat='state in states'>
 <li> {{ state }} </li>
</ul>
```

More directives

app.js

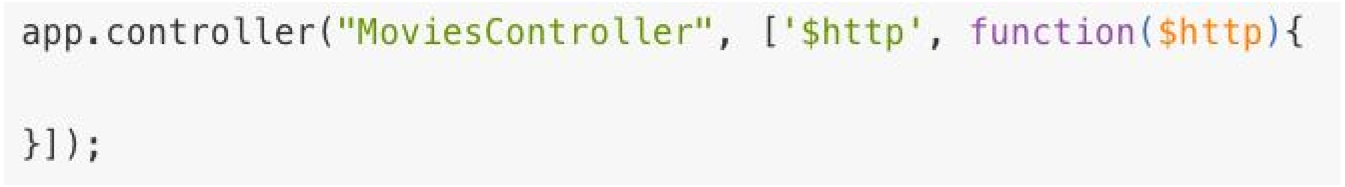
```
var iLovePurdue = true
```

index.html

```
<div ng-show='iLovePurdue'>
  <h1> Boiler Up! </h1>
</div>
```

Services

Give Controller additional functionality
- ex: Can fetch JSON from web services using the $http service

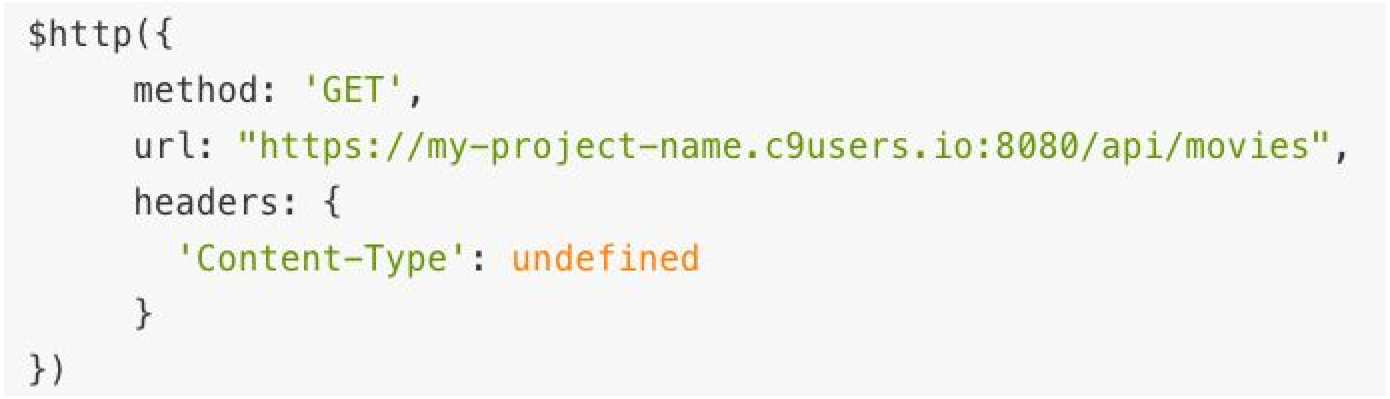


```
app.controller("MoviesController", ['$http', function($http){


}]);
```

Calling our API

HTTP request header:

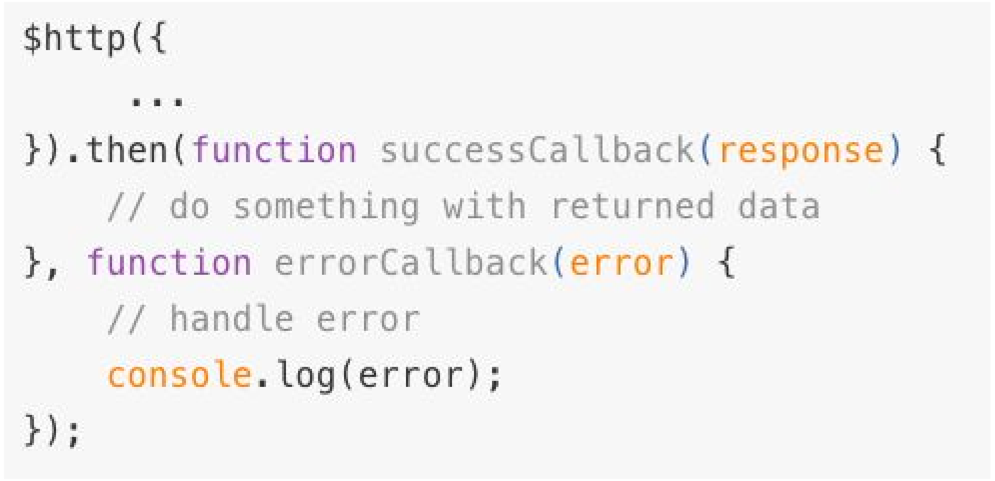


```
$http({
      method: 'GET',
      url: "https://my-project-name.c9users.io:8080/api/movies",
      headers: {
        'Content-Type': undefined
      }
})
```

This service sends a request to the
API’s POST endpoint.

Calling our API (cont.)

The request has been sent to our API. The API will respond and the $http will return a *promise* of either SUCCESS or FAILURE.



```
$http({
    ...
}).then(function successCallback(response) {
   // do something with returned data
}, function errorCallback(error) {
   // handle error
   console.log(error);
});
```

Questions?