MEANWORKS: MONGODB/MONGOOSE. EXPRESS. ANGULAR AND NODE WORKSHOP WITH ANGULAR FULLSTACK GENERATOR

FULL STACK JAVASCRIPT

NODE.JS VERSION: 5.1.0
LAST UPDATED: JAN 2016

MONGODB. EXPRESS. ANGULAR. AND NODE.JS

AGENDA

- QUICK OVERVIEW OF MEAN
- MONGODB DEMO
- EXPRESS BASICS
- ANGULAR AND FULL STACK GENERATOR
- MEANWORKS ADVENTURE DEMO
- WORKSHOP 🔨

REQUIREMENTS

YOU'LL NEED THESE (INSTALL THEM BEFORE PROCEEDING):

- NODE.JS, NPM AND MONGODB
- GRUNT, YEOMAN, FULL STACK GENERATOR, WEBDRIVER
- CODE EDITOR (SUBLIME TEXT, ATOM, ETC.)–IDES ARE NOT RECOMMENDED FOR THIS WORKSHOP
- COMMAND LINE APP (TERMINAL, ITERM, ETC.)

REQUIREMENTS

YOU'LL NEED THESE (INSTALL THEM BEFORE PROCEEDING):

- INTERNET CONNECTION (DUH!)
- SLIDES & SAMPLE CODE HTTPS://GITHUB.COM/AZAT-CO/MEANWORKS

PROXY FOR NPM. BOWER. GIT. ETC.

- NPM: HTTPS://DOCS.NPMJS.COM/MISC/CONFIG
  - BOWER: HTTP://BOWER.IO/DOCS/CONFIG/
- GIT: HTTPS://GIT-SCM.COM/DOCS/GIT-CONFIG

AVOID STORING PLAIN PASSWORDS IN THE GIT OR NPM CONFIGS.

SETUP INSTRUCTIONS

DETAILED GENERIC VERSION: HTTPS://GITHUB.COM/AZAT-CO/MEANWORKS/BLOB/MASTER/SETUP.PDF

... AND IN THE MEANWORKS

REQUIREMENTS

YOU'LL ALSO NEED (WE'LL INSTALL THEM TOGETHER IN MEANWORKS):

- BOWER
- GRUNT
- NODEMON
- MOCHA

SLIDES AND CODE

- SLIDES: HTTPS://GITHUB.COM/AZAT-CO/MEANWORKS/BLOB/MASTER/SLIDES
- APP: HTTPS://GITHUB.COM/AZAT-CO/MEANWORKS/BLOB/MASTER/APP

DOWNLOADING SLIDES AND CODE

CLONE THE REPOSITORY (RECOMMENDED):

```
$ git clone https://github.com/azat-co/meanworks
```

NAVIGATE TO SLIDES:

```
$ cd meanworks/slides
$ open README.pdf
```

OR READ IN THE BROWSER:

https://github.com/azat-co/meanworks/master/blob/slides/readme.pdf

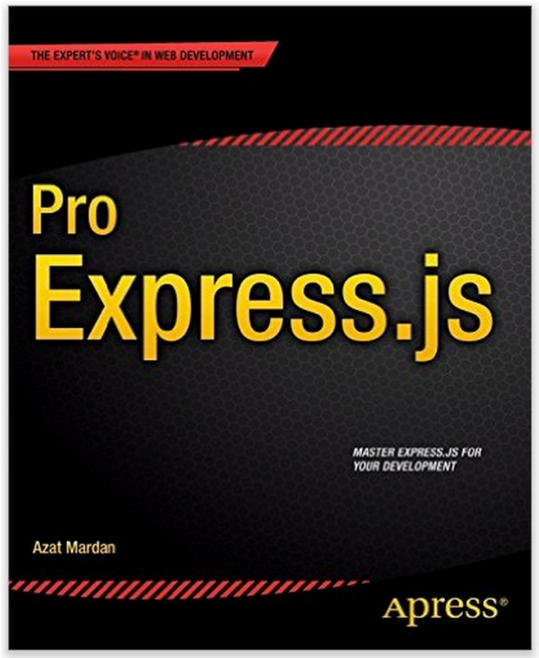
AND

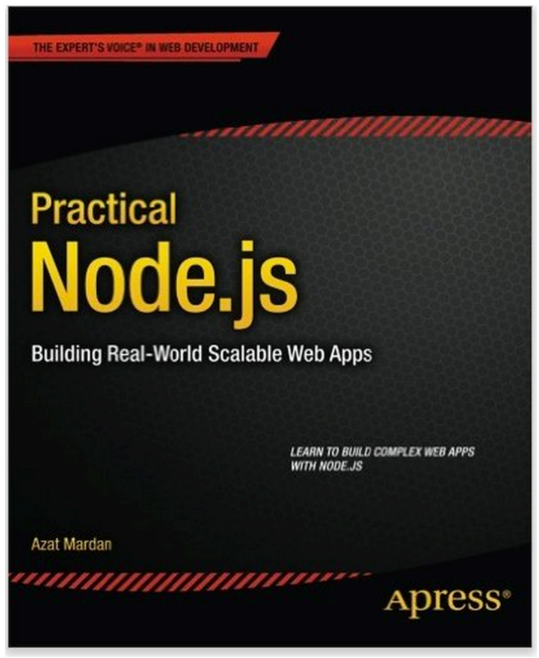
https://github.com/azat-co/meanworks/master/blob/slides/readme.md

INTRODUCTIONS

INSTRUCTOR: AZAT MARDAN

- WORK: CAPITAL ONE, STORIFY, FDIC, NIH, DOCUSIGN, AND OTHERS
- BLOG: HTTP://WEBAPPLOG.COM
- BOOKS: REACT QUICKLY, FULL STACK JAVASCRIPT, PRACTICAL NODE.JS, PRO EXPRESS.JS, MONGOOSE COURSE

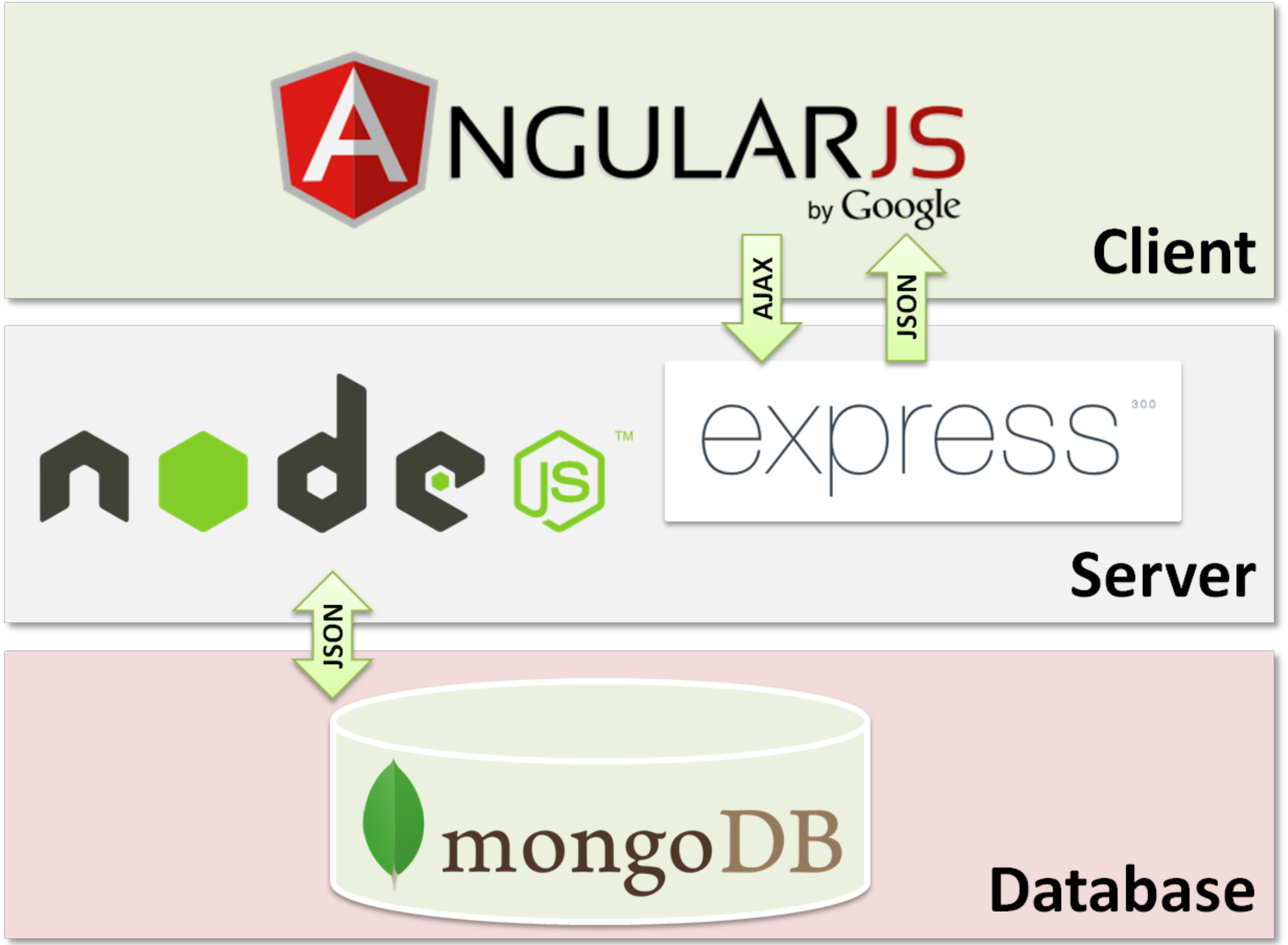

THE EXPERT'S VOICE® IN WEB DEVELOPMENT
Pro
Express.js
MASTER EXPRESS.JS FOR YOUR DEVELOPMENT
Azat Mardan
Apress®


THE EXPERT'S VOICE® IN WEB DEVELOPMENT
Practical
Node.js
Building Real-World Scalable Web Apps
LEARN TO BUILD COMPLEX WEB APPS WITH NODE.JS
Azat Mardan
Apress®

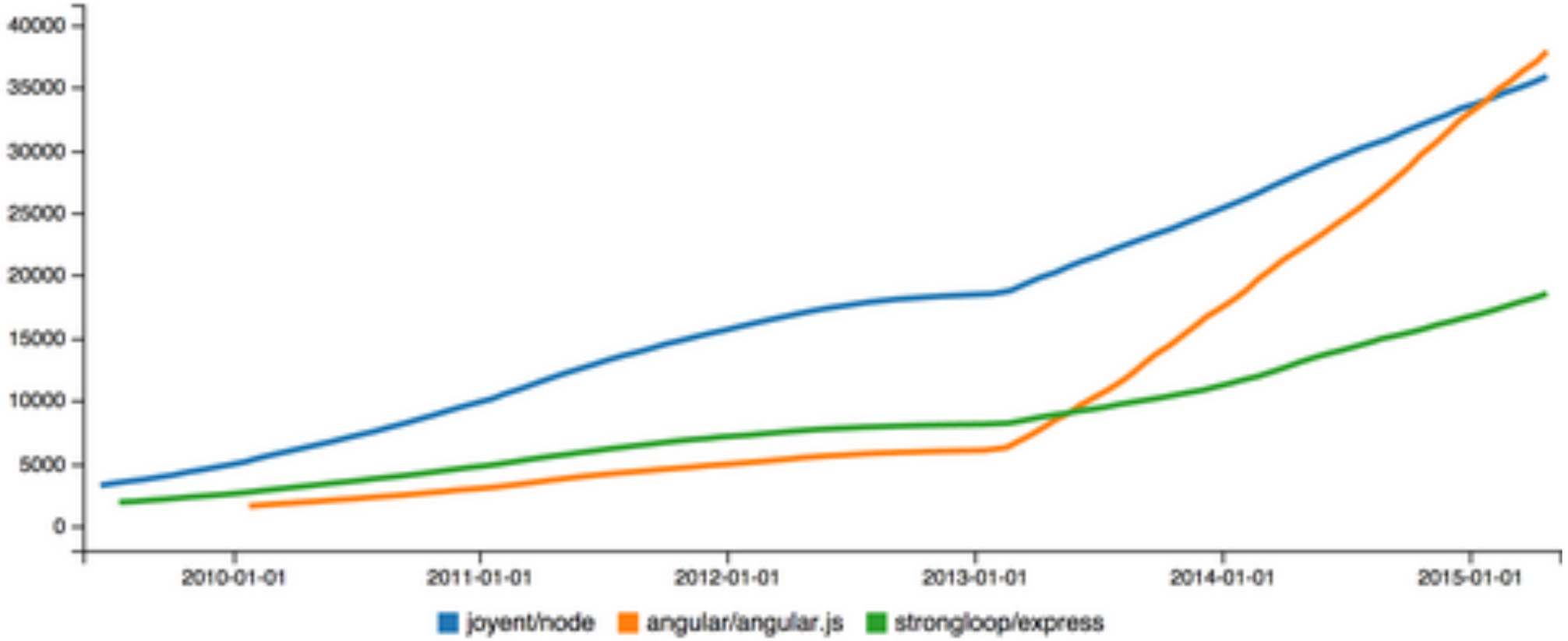
MEET M.E.A.N.!

BENEFITS OF MEAN

- RAPID PROTOTYPING
- ONE LANGUAGE TO RULE THEM ALL!
- VIBRANT COMMUNITY


ANGULARJS
by Google
Client
AJAX
JSON
node
JS
express
300
Server
JSON
mongoDB
Database

RATE OF GITHUB STAR GROWTH FOR NODE, EXPRESS AND ANGULAR



EASY PEASY


The Friendly & Fun Javascript Fullstack for your next web application
MEAN is an opinionated fullstack javascript framework - which simplifies and accelerates web application development.
Get MEAN by running...
$ sudo npm install -g mean-cli
$ mean init yourNewApp
LATEST RELEASE: v0.5.5
LATEST COMMIT: Aug 18, 2015
FORKS: 2170
FORK MEAN.IO ON GITHUB
MEAN stands for:
mongoDB
MongoDB is the leading NoSQL database, empowering businesses to be more agile and scalable.
express
Express is a minimal and flexible node.js web application framework, providing a robust set of features for building single and multi-page, and hybrid web applications.
ANGULARJS by Google
AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable,
node.js
Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.

WHAT IS MEAN?

- MONGODB: NOSQL DATABASE
- EXPRESS: WEB HTTP SERVER FRAMEWORK FOR REST API
- ANGULARJS: FRONT-END FRAMEWORK
- NODE.JS: NON-BLOCKING I/O TO POWER EXPRESS.JS AND CONNECT TO MONGODB

MONGODB

- DOCUMENT STORE NOSQL DATABASE
- REPLICATION & HIGH AVAILABILITY
- EASY TO SCALE AND RAPID PROTOTYPING
- SCHEMA-LESS AND NO COMPLEX JOINS
- JAVASCRIPT INTERFACE

INSTALL MONGODB

FOR ALL OS:

HTTPS://WWW.MONGODB.ORG/DOWNLOADS

FOR MAC OS X AND HOMEBREW:

```
$ brew install mongodb@3.0.6
```

LAUNCHING MONGODB

LAUNCH THE mongod SERVICE WITH:

```
$ mongod
```

YOU SHOULD BE ABLE TO SEE INFORMATION IN YOUR TERMINAL. THE DEFAULT PORT IS 27017.

MONGODB SHELL (MONGO)

FOR THE MONGODB SHELL, OR MONGO, LAUNCH IN A NEW TERMINAL WINDOW (LET THE SERVER RUN), THIS COMMAND:

```
$ mongo
```

MONGODB SHELL (MONGO)

TO TEST THE DATABASE, USE THE JAVASCRIPT-LIKE INTERFACE AND COMMANDS SAVE AND FIND:

```
> db.test.save({a:1})
> db.test.find()
```

MONGODB USES JAVASCRIPT!

MONGODB SHELL (MONGO)

USEFUL MONGODB SHELL COMMANDS:

```
> help
> show dbs
> use board
> show collections
> db.messages.remove();
```

MONGODB SHELL (MONGO)

USEFUL MONGODB SHELL COMMANDS:

```
> var a=db.messages.findOne();
> print json(a);
> a.message="hi";
> db.messages.save(a);
> db.messages.find({});
```

MONGODB SHELL (MONGO)

USEFUL MONGODB SHELL COMMANDS:

```
> db.messages.update({name:"John"},{$set:
{message:"bye"}});
> db.messages.find({name:"John"});
> db.messages.remove({name:"John"});
```

MONGODB SHELL DEMO

DEMO TIME! 👌

MONGODB VS. MONGODB NATIVE DRIVER

THEY ARE NOT THE SAME! 😕

MONGODB NATIVE DRIVER (MONGODB)

NODE.JS NATIVE DRIVER FOR MONGODB (HTTPS://GITHUB.COM/CHRISTKV/NODE-MONGODB-NATIVE)

```
$ npm install mongodb --save
```

ESTABLISHING CONNECTION

```
var MongoClient = require('mongodb').MongoClient,
  assert = require('assert');

// Connection URL
var url = 'mongodb://localhost:27017/myproject';
// Use connect method to connect to the Server
MongoClient.connect(url, function(err, db) {
  assert.equal(null, err);
  console.log("Connected correctly to server");

  db.close();
});
```

INSERTING DOCUMENTS

```
collection.insert([
  {a : 1}, {a : 2}, {a : 3}
], function(err, result) {
  console.log("Inserted 3 documents into the document collection");
  callback(result);
});
```

UPDATING DOCUMENTS

```
collection.update({ a : 2 }
  , { $set: { b : 1 } }, function(err, result) {
  console.log("Updated the document with the field a equal to 2");
  callback(result);
});
```

REMOVING DOCUMENTS

```
// Insert some documents
collection.remove({ a : 3 }, function(err, result) {
  console.log("Removed the document with the field a equal to 3");
  callback(result);
});
```

FETCHING DOCUMENTS

```
collection.find({}).toArray(function(err, docs) {
  console.log("Found the following records");
  console.dir(docs);
  callback(docs);
});
```

MONGODB AND MONGOOSE CHEATSHEET

https://github.com/azat-co/cheatsheets/tree/master/mongodb-mongoose

ANGULARJS

ANGULARJS

- SUPPORTED BY GOOGLE
- RAPID PROTOTYPING (TWO-WAY BINDING)
- FEATURE RICH, I.E., DOES A LOT OF THINGS FOR DEVELOPERS
- DE-EMPHASIZES EXPLICIT DOM MANIPULATION
- SEPARATES PRESENTATION, DATA, AND LOGIC COMPONENTS

TWO-WAY BINDING

MODELS->VIEWS
VIEWS->MODELS

NO DOM MANIPULATIONS.

NO JQUERY.

NO `$('.bnt').click(fn).`

ANGULAR STRUCTURE

- ROUTE FILE
- TEMPLATES
- CONTROLLERS

ROUTE

```
angular.module('ngFullstackNewApp')
  .config(function($stateProvider) {
    $stateProvider
      .state('main', {
        url: '/',
        templateUrl: 'app/main/main.html',
        controller: 'MainController',
        controllerAs: 'main'
      });
  });
```

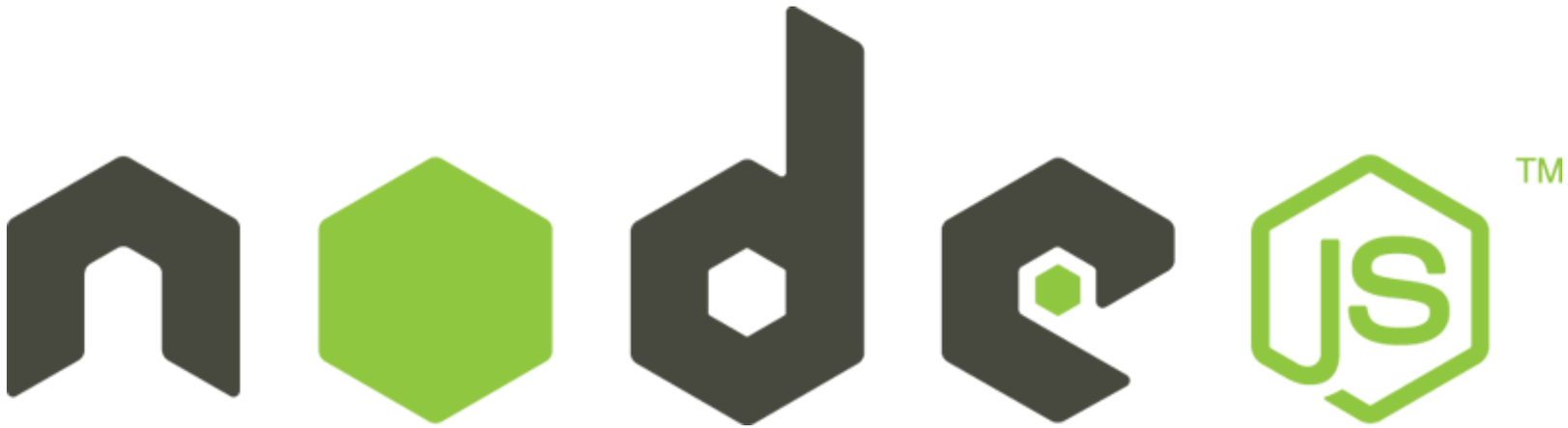
TEMPLATE

```
<navbar></navbar>

<header class="hero-unit" id="banner">
  <div class="container">
    <h1>'Allo, 'Allo!</h1>
    <p class="lead">Kick-start your next web app with Angular Fullstack</p>
    <img src="assets/images/yeoman.png" alt="I'm Yeoman">
  </div>
</header>

<div class="container">
  <div class="row">
    <div class="col-lg-12">
      <h1 class="page-header">Features:</h1>
      <ul class="nav nav-tabs nav-stacked col-md-4 col-lg-4 col-sm-6" ng-repeat="thing in main.awesomeThings">
        <li><a href="#" tooltip="{{thing.info}}">{{thing.name}}</a></li>
      </ul>
    </div>
  </div>
</div>

<footer></footer>
```

CONTROLLER

```
(function() {

class MainController {

  constructor($http) {
    this.$http = $http;
    this.awesomeThings = [];

    $http.get('/api/things').then(response => {
      this.awesomeThings = response.data;
    });
  }

  addThing() {
    if (this.newThing) {
      this.$http.post('/api/things', { name: this.newThing });
      this.newThing = '';
    }
  }

  deleteThing(thing) {
    this.$http.delete('/api/things/' + thing._id);
  }
}

angular.module('ngFullstackNewApp')
  .controller('MainController', MainController);

})();
```

NODE.JS



NODE.JS

- SCALABLE, FAST WEB PLATFORM
- MATURE (V5.3 AS OF JAN '16) AND SUPPORTS A LOT OF ES6 FEATURES
- ALLOWS FOR FRONT-END CODE RE-USE ON THE SERVER: LODASH, UNDERSCORE AND VICE VERSA

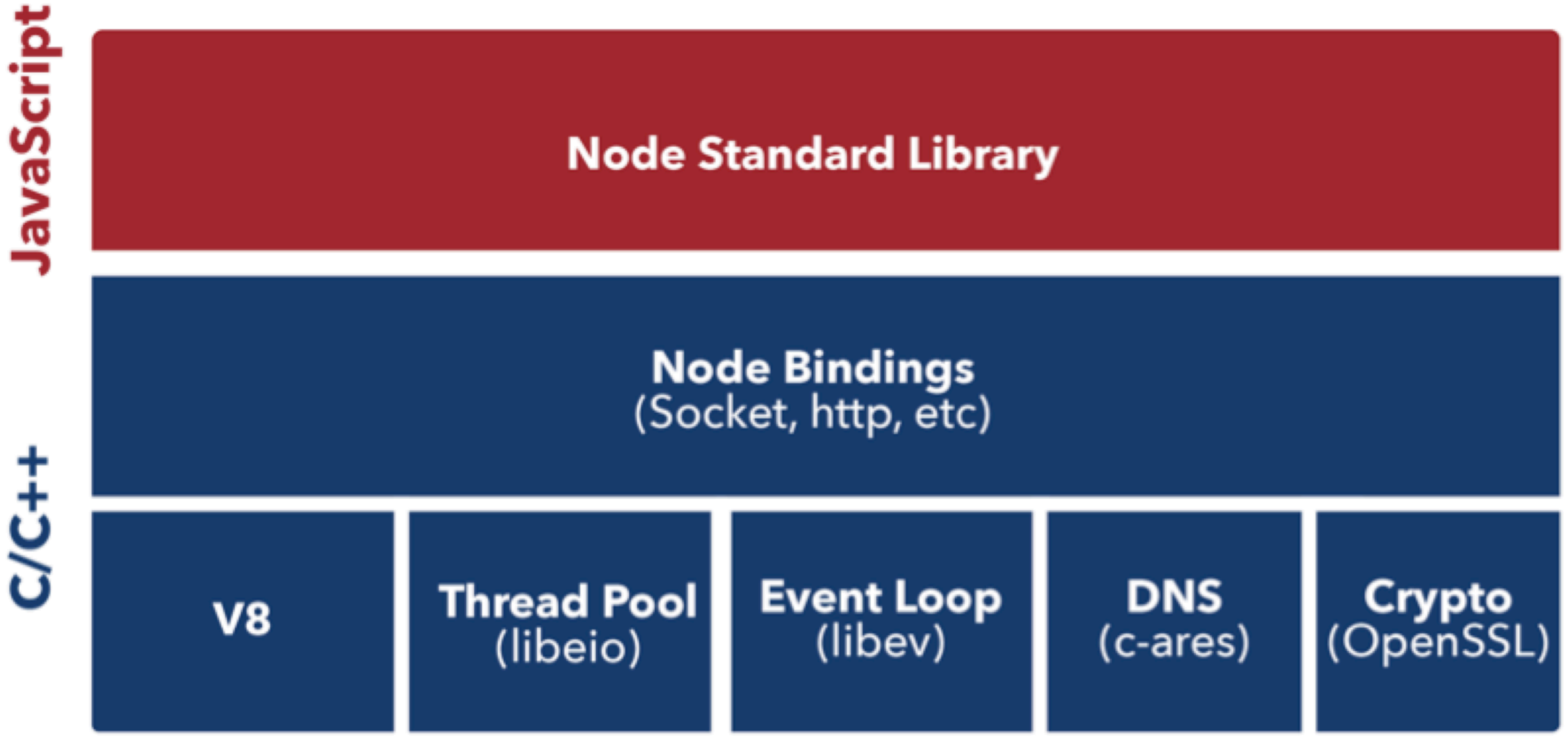
ADVANTAGES OF NODE.JS

- NON-BLOCKING I/O
- SUPER FAST (V8)
- VIBRANT ECOSYSTEM (NPM)
- ABILITY TO RE-USE CODE ON BROWSER AND SERVER
- ABILITY TO USE FRONT-END DEVS FOR BACK-END AND VICE VERSA

NODE CORE: V8, LIBEV, AND LIBEIO

- LIBEV: THE EVENT LOOP
- LIBEIO: ASYNC I/O
- LIBUV: ABSTRACTION ON LIBEIO, LIBEV, C-ARES (FOR DNS) & IOCP (FOR WINDOWS)

BIRD'S EYE VIEW OF NODE



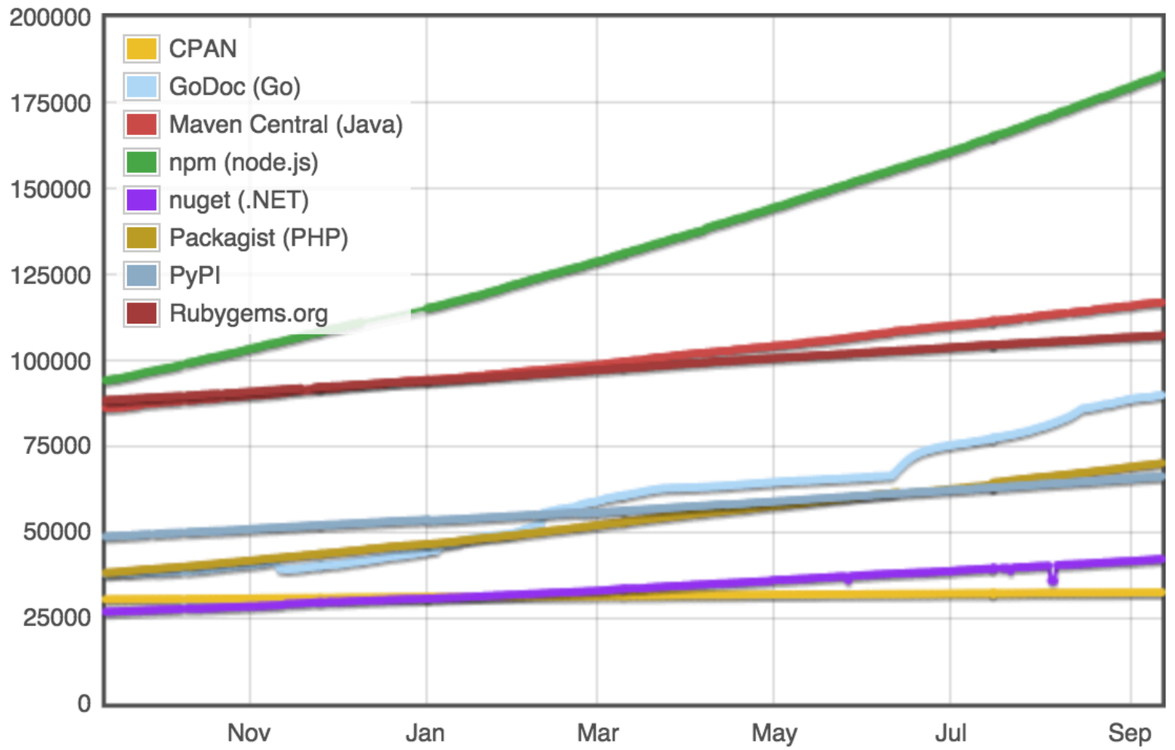
NODE STACK

- CORE MODULES: HTTP, FS, NET, UTILS, QUERY
- WEB FRAMEWORKS: EXPRESS, HAPI, LOOPBACK, SAILS
- DATABASE DRIVERS: MONGODB, POSTGRESQL, MYSQL...
- TESTING: MOCHA, SUPERTEST AND EXPECT
- DEPENDENCY MANAGEMENT: BOWER AND NPM
- BUILDING: WEBPACK, GRUNT OR GULP

NODE STACK II

- AUTHENTICATION: PASSPORT
- CODE GENERATION: YO
- PROCESS MANAGERS: PM2, STRONG-CLUSTER-CONTROL, FOREVER

NPM RULES


200000
175000
150000
125000
100000
75000
50000
25000
0
CPAN
GoDoc (Go)
Maven Central (Java)
npm (node.js)
nuget (.NET)
Packagist (PHP)
PyPI
Rubygems.org
Nov
Jan
Mar
May
Jul
Sep

NODE CONSOLE DEMO

LET'S PLAY WITH NODE!

EXPRESS.JS

ADVANTAGES OF EXPRESS.JS

- EASY TO LEARN (CONFIGURATION OVER CONVENTION)
- MYRIADS OF MODULES
- MATURE AND THE MOST POPULAR (V4.X)

CONFIGURING EXPRESS

THE EXPRESS SERVER NEEDS TO BE CONFIGURED BEFORE IT CAN START

MANAGE CONFIGURATION VIA THE `set` METHOD:

```
var app = express();
app.set('port', process.env.PORT || 3000);
app.set('views', 'views'); // the directory the templates are stored in
app.set('view engine', 'jade');
```

CONNECT FRAMEWORK

EXPRESS LEVERAGES THE CONNECT FRAMEWORK TO PROVIDE MIDDLEWARE FUNCTIONALITY.

MIDDLEWARE ARE USED TO MANAGE HOW A REQUEST SHOULD BE HANDLED.

HTTP://SENCHALABS.GITHUB.COM/CONNECT

NODE.JS MIDDLEWARE PATTERN

MIDDLEWARE PATTERN IS A SERIES OF PROCESSING UNITS CONNECTED TOGETHER, WHERE THE OUTPUT OF ONE UNIT IS THE INPUT FOR THE NEXT ONE. IN NODE.JS, THIS OFTEN MEANS A SERIES OF FUNCTIONS IN THE FORM:

```
function(args, next) {
  next(output) // error or real output
}
```

MIDDLEWARE ORDER

MIDDLEWARE ARE EXECUTED IN THE ORDER SPECIFIED:

```
app.use(express.logger('dev'));
app.use(express.basicAuth('test', 'pass'));
app.use(express.json());
```

CREATING MIDDLEWARE

CUSTOM MIDDLEWARE IS EASY TO CREATE:

```
app.use(function (req, res, next) {
  // modify req or res
  // execute the callback when done
  next();
});
```

MOST POPULAR AND USEFUL CONNECT/ EXPRESS MIDDLEWARE

```
$ sudo npm install <package_name> --save
```

- BODY-PARSER REQUEST PAYLOAD
- COMPRESSION GZIP
- CONNECT-TIMEOUT SET REQUEST TIMEOUT
- COOKIE-PARSER COOKIES
- COOKIE-SESSION SESSION VIA COOKIES STORE

CONNECT/EXPRESS MIDDLEWARE

- CSURF CSRF
- ERRORHANDLER ERROR HANDLER
- EXPRESS-SESSION SESSION VIA IN-MEMORY OR OTHER STORE
- METHOD-OVERRIDE HTTP METHOD OVERRIDE
- MORGAN SERVER LOGS
- RESPONSE-TIME: RESPONSE TIME

EXPRESS.JS CHEATSHEET

https://github.com/azat-co/cheatsheets/blob/master/express4/index.md

GENERAL MEAN GENERATORS

- HTTP://MEANJS.ORG/
- HTTP://MEAN.IO/#!/
- HTTPS://GITHUB.COM/DAFTMONK/GENERATOR-ANGULAR-FULLSTACK

ANGULAR FULL STACK GENERATOR

YEOMAN GENERATOR FOR ANGULARJS WITH AN EXPRESS SERVER

HTTPS://GITHUB.COM/ANGULAR-FULLSTACK/GENERATOR-ANGULAR-FULLSTACK

GENERATOR COMPONENTS

- MONGODB (MONGOOSE) OR SQL DATABASES (SEQUELIZE)
- EXPRESS
- AUTH (GOOGLE, FACEBOOK, ETC.)
- SOCKET.IO
- JASMINE OR MOCHA TESTS
- TWITTER BOOTSTRAP UI

```
$ npm install -g yo grunt-cli bower generator-angular-fullstack
$ mkdir my-new-project && cd $_
$ yo angular-fullstack [app-name]
```

```
# Client

? What would you like to write markup with? (Use arrow keys)
› HTML
  Jade
```

```
? What would you like to write stylesheets with?
› CSS
  Sass
  Stylus
  Less
```

```
? What Angular router would you like to use? (Use arrow keys)
  ngRoute
❯ uiRouter
```

```
? Would you like to include Bootstrap? (Y/n) Y
```

```
? Would you like to include UI Bootstrap? (Y/n) Y
```

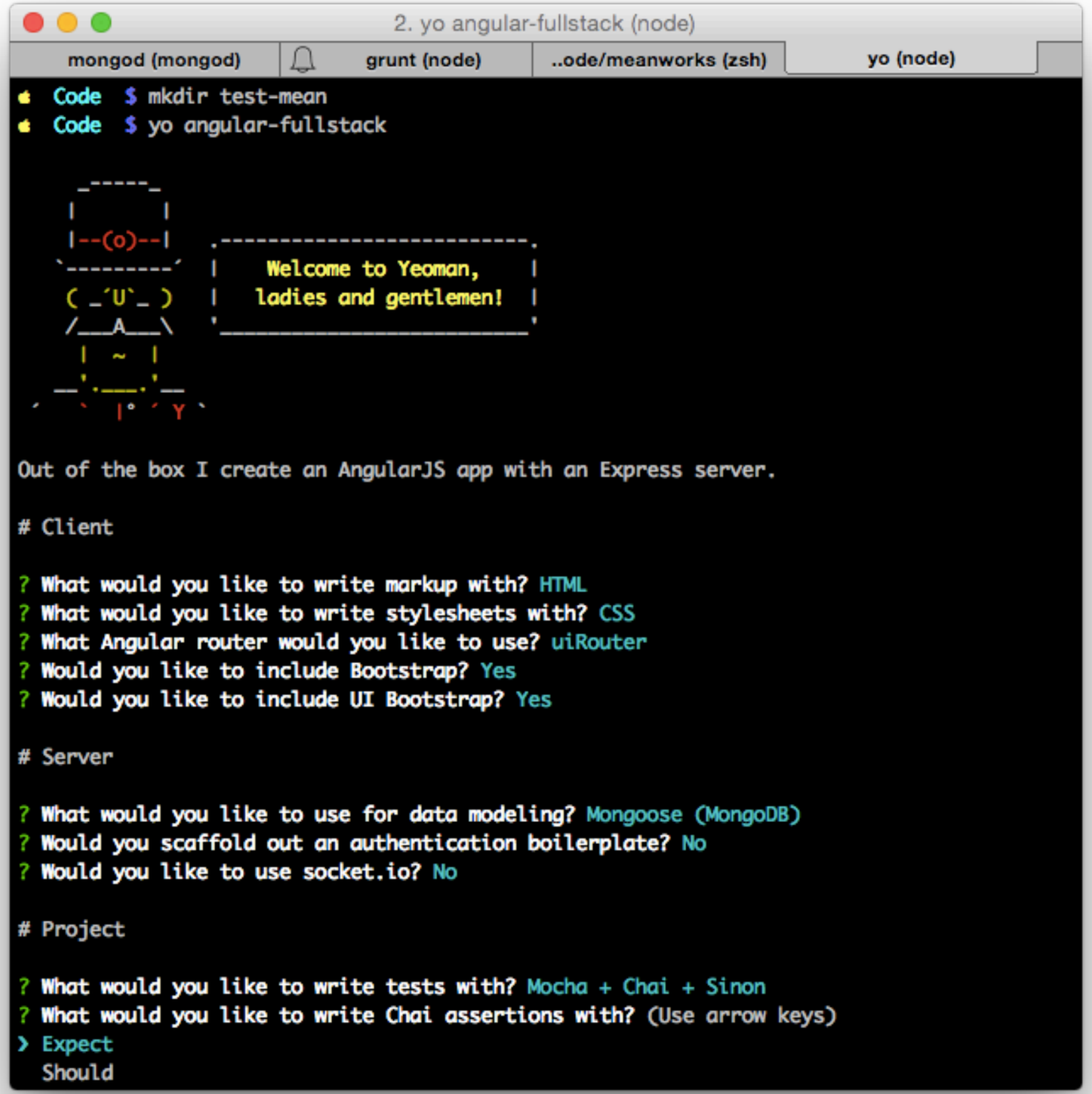
```
# Server

? What would you like to use for data modeling? (Press <space> to select)
❯◉ Mongoose (MongoDB)
 ◯ Sequelize (MySQL, SQLite, MariaDB, PostgreSQL)
```

```
? Would you scaffold out an authentication boilerplate? (Y/n) n
```

```
? Would you like to use socket.io? (Y/n) n
```

```
# Project

? What would you like to write tests with? (Use arrow keys)
  Jasmine
> Mocha + Chai + Sinon
```

```
? What would you like to write Chai assertions with? (Use arrow keys)
❯ Expect
  Should
```


2. yo angular-fullstack (node)
mongod (mongod)
grunt (node)
..ode/meanworks (zsh)
yo (node)
Code $ mkdir test-mean
Code $ yo angular-fullstack
Welcome to Yeoman, ladies and gentlemen!
Out of the box I create an AngularJS app with an Express server.
Client
? What would you like to write markup with? HTML
? What would you like to write stylesheets with? CSS
? What Angular router would you like to use? uiRouter
? Would you like to include Bootstrap? Yes
? Would you like to include UI Bootstrap? Yes
Server
? What would you like to use for data modeling? Mongoose (MongoDB)
? Would you scaffold out an authentication boilerplate? No
? Would you like to use socket.io? No
Project
? What would you like to write tests with? Mocha + Chai + Sinon
? What would you like to write Chai assertions with? (Use arrow keys)
> Expect
Should

```
create bower.json
create package.json
create .gitignore
create .bowerrc
create .buildignore
create .editorconfig
create .gitattributes
create .jscsrc
create .travis.yml
```

```
create client/.htaccess
create client/.jshintrc
create client/app/app.js
create client/app/app.css
create client/app/main/main.controller.js
create client/app/main/main.controller.spec.js
create client/app/main/main.js
create client/app/main/main.css
create client/app/main/main.html
create client/assets/images/yeoman.png
```

```
create client/components/footer/footer.directive.js
create client/components/footer/footer.css
create client/components/footer/footer.html
create client/components/modal/modal.service.js
create client/components/modal/modal.css
create client/components/modal/modal.html
create client/components/navbar/navbar.controller.js
create client/components/navbar/navbar.directive.js
create client/components/navbar/navbar.html
create client/components/ui-router/ui-router.mock.js
create client/components/util/util.module.js
create client/components/util/util.service.js
create client/favicon.ico
create client/index.html
create client/robots.txt
```

```
create e2e/components/navbar/navbar.po.js
create e2e/main/main.po.js
create e2e/main/main.spec.js
create Gruntfile.js
create karma.conf.js
create mocha.conf.js
create protractor.conf.js
create README.md
```

```
create server/.jshintrc
create server/.jshintrc-spec
create server/app.js
create server/components/errors/index.js
create server/config/local.env.js
create server/config/local.env.sample.js
create server/config/environment/development.js
create server/config/environment/index.js
create server/config/environment/production.js
create server/config/environment/shared.js
create server/config/environment/test.js
create server/config/express.js
create server/config/seed.js
create server/index.js
create server/routes.js
create server/views/404.html
create server/api/thing/thing.controller.js
create server/api/thing/thing.events.js
create server/api/thing/thing.integration.js
create server/api/thing/thing.model.js
create server/api/thing/index.js
create server/api/thing/index.spec.js
```

GENERATOR HELP

WHEN IN DOUBT:

```
$ yo angular-fullstack --help
```

GENERATOR COMMANDS

SERVER SIDE:

```
angular-fullstack:endpoint
```

CLIENT SIDE:

```
angular-fullstack:route
angular-fullstack:controller
angular-fullstack:filter
```

GENERATOR COMMANDS II

CLIENT SIDE:

```
angular-fullstack:directive
angular-fullstack:service
angular-fullstack:provider
angular-fullstack:factory
angular-fullstack:decorator
```

DEPLOYMENT:

```
angular-fullstack:openshift
angular-fullstack:heroku
```

GENERATOR DEMO!

IT'S DEMO TIME! 🔭🖥👌

```
$ mkdir app
$ yo angular-fullstack app
$ grunt test
$ grunt serve
```

HTTP://LOCALHOST:9000/

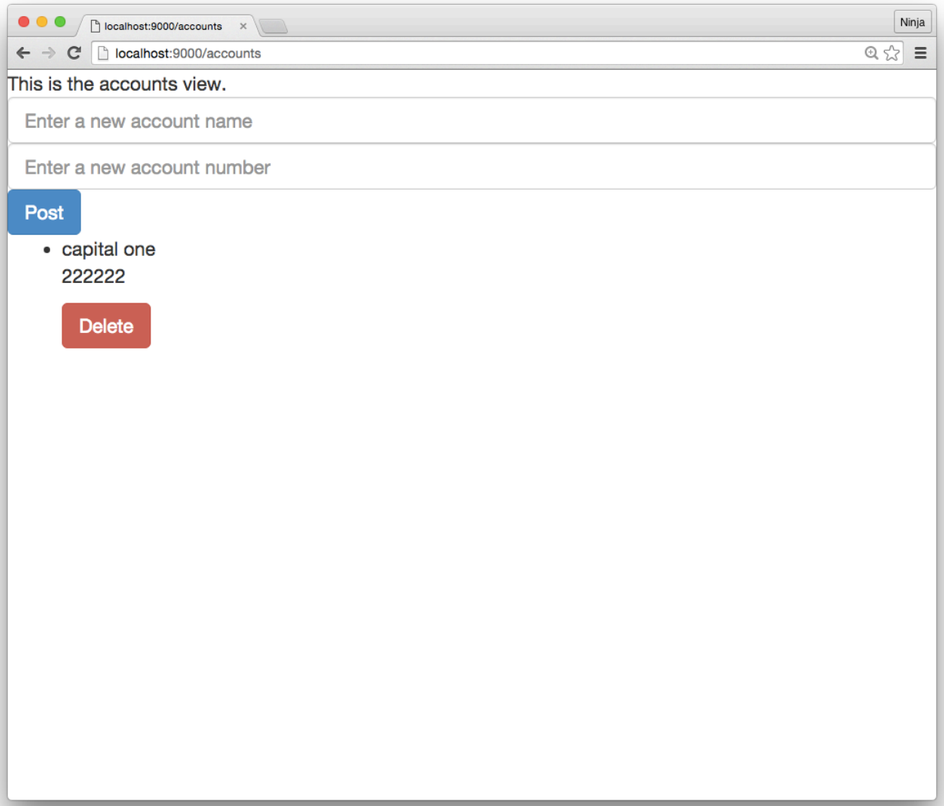
WORKSHOP!

PROJECT

USE MEANWORKS TO VERIFY VERSIONS AND BUILD THE PROJECT APP:

1. BUILD REST API
2. BUILD ANGULAR APP WITH FULL STACK GENERATOR
3. CREATE CRUD FOR ACCOUNTS

THE APP


localhost:9000/accounts
Ninja
localhost:9000/accounts
This is the accounts view.
Enter a new account name
Enter a new account number
Post
capital one
222222
Delete

APP DEMO

IT'S DEMO TIME! 🔭🖥👌

APP: https://github.com/azat-co/tree/master/app

RUNNING THE APP, I.E., OFFICIAL SOLUTION TO MEANWORKS (OPTIONAL, BECAUSE YOU'LL BE BUILDING YOUR OWN):

https://github.com/azat-co/tree/master/app/readme.md

PROJECT STRUCTURE

```
/node_modules
/e2e
/server
/client
index.js
package.json
npm-shrinkwrap.json
readme.md
test.js
transactions.json
accounts.json
```

LOCAL SETUP

CREATE A NEW FOLDER:

```
$ mkdir mean
```

TEST IF YOU HAVE NODE.JS, MONGODB, NPM AND GIT:

```
$ node -v
$ npm -v
$ mongod --version
$ git --version
```

INSTALLATION

```
$ npm install -g yo grunt-cli generator-angular-fullstack
```

PROJECT INIT

```
$ cd mean
$ yo angular-fullstack
```

ENDPOINTS AND ROUTES

```
$ yo angular-fullstack:endpoint account
$ yo angular-fullstack:route accounts
$ yo angular-fullstack:endpoint transaction
$ yo angular-fullstack:route transactions
```

(ANSWER GENERATOR QUESTIONS)

TEST FIRST

```
$ grunt test
```

EXPRESS APP

```
var express = require('express'),
  bodyParser = require('body-parser'),
  errorHandler = require('errorhandler'),
  app = express(),
  mongo = require('mongodb'),
  cors = require('cors')

app.use(cors({
  origin: 'http://localhost:8080'
}))
app.use(bodyParser.json())
app.use(express.static('public'))
...
```

MONGODB CONNECTION

```
...
var url = 'mongodb://localhost:27017/ngfullstacknew-dev';

mongo.connect(url, function(err, db) {
  if (err) {
    console.error(err)
    return process.exit(1)
  }
  var transactionsCollection = db.collection('transactions')
  ...
```

GET /api/transaction

```
...
  app.get('/api/transactions', function(req, res, next){
    transactionsCollection.find({}, {sort: {_id:-1}}).toArray(function(error, transactions){
      if (error) return next(error)
      console.log(transactions)
      return res.send(transactions)
    })
  })
})
...
```

STARTUP

```
...
app.use(errorHandler())

app.listen(3000)
...
```

STARTING THE SERVER

IN TERMINAL:

```
$ node ./node_modules/nodemon/bin/nodemon.js index.js
```

SEEDING THE DATABASE

USE THIS COMMAND IN TERMINAL TO POPULATE THE `transactions` COLLECTION IN THE DATABASE `ngfullstacknew-dev` WITH TRANSACTIONS:

```
$ mongoimport  --host=127.0.0.1 --port=27017 --db ngfullstacknew-dev --collection transactions --file transactions.json --jsonArray
$ mongoimport  --host=127.0.0.1 --port=27017 --db ngfullstacknew-dev --collection accounts --file accounts.json --jsonArray
```

DATA JSON

TRANSACTION.JSON: HTTPS://GITHUB.COM/AZAT-CO/MEANWORKS/BLOB/MASTER/APP/TRANSACTIONS.JSON
ACCOUNTS.JSON: HTTPS://GITHUB.COM/AZAT-CO/MEANWORKS/BLOB/MASTER/APP/ACCOUNTS.JSON

CHECKING SEED DATA

CHECK THE DATA IN THE MONGO SHELL:

```
> use ngfullstacknew-dev
> db.transactions.find({})
```

RUNNING APP

OPEN THIS URL IN CHROME, FIREFOX OR SAFARI:

HTTP://LOCALHOST:9000/ACCOUNTS

ROUTE

```
angular.module('ngFullstackNewApp')
  .config(function ($stateProvider) {
    $stateProvider
      .state('accounts', {
        url: '/accounts',
        templateUrl: 'app/accounts/accounts.html',
        controller: 'AccountsCtrl'
      });
  });
```

CONTROLLER

```
angular.module('ngFullstackNewApp')
  .controller('AccountsCtrl', function ($scope, $http) {
    $scope.accounts = [];

    $http.get('/api/accounts').success(function(accounts) {
      $scope.accounts = accounts;
    });
  });
```

DIRECTIVES

NG-REPEAT

```
<ul class="account-list">
  <li ng-repeat="account in accounts">
    <span>{{ account.name }}</span>
    <span>{{ account.number }}</span>
  </li>
</ul>
```

NG-IF

```
<a class="btn btn-info" href="#"  ng-click="disputeTransaction(transaction)" ng-if="transaction.dispute==false">
  <i class="glyphicon glyphicon-question-sign"></i>
  Dispute
</a>
```

NG-HIDE

```
<i ng-hide="transaction.date!='pending'" class="glyphicon glyphicon-refresh">?</i>
```

NG-CLICK

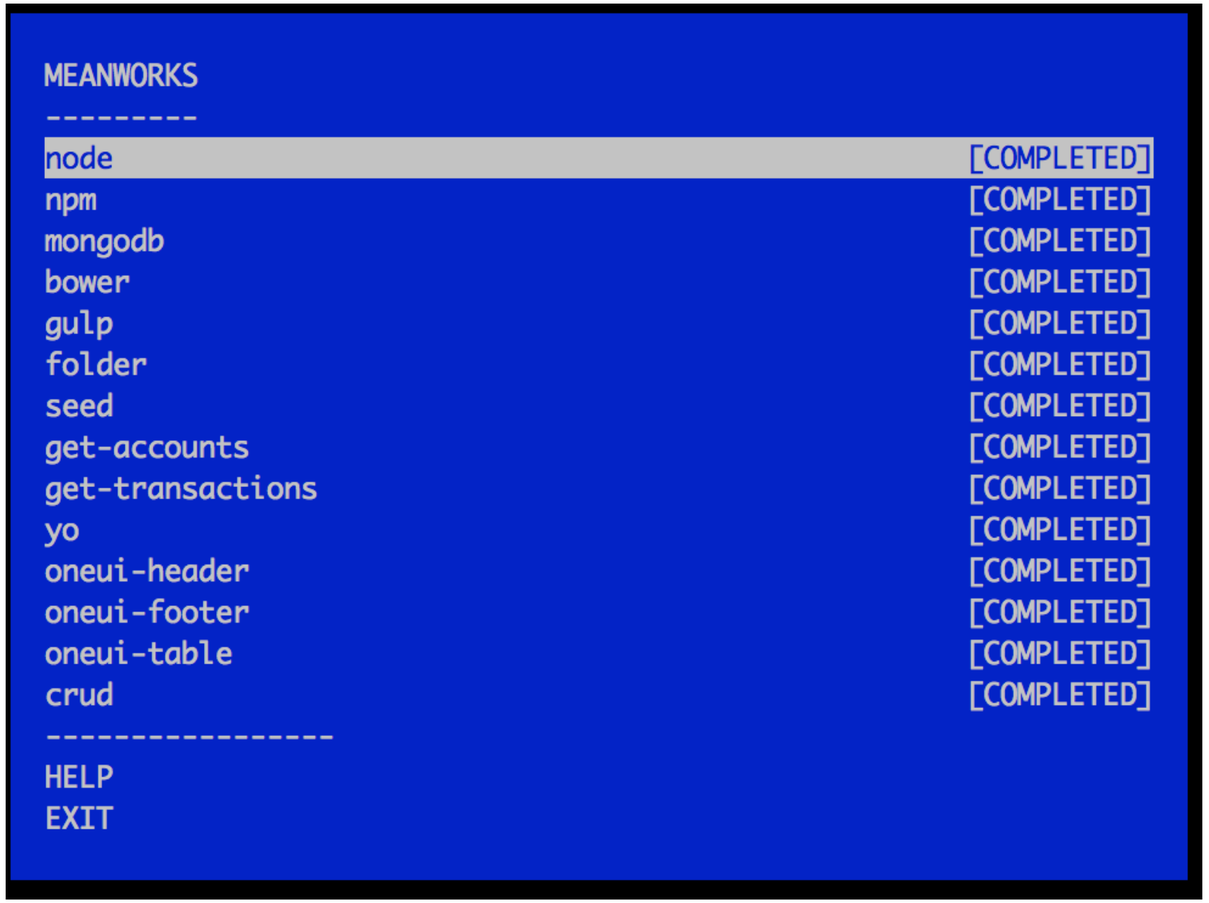
```
<a class="btn btn-danger" href="#"  ng-click="removeAccount(account)">
  <i class="glyphicon-remove-circle glyphicon"></i> Delete
</a>
```

ICON STYLES

HTTP://GLYPHICONS.COM

QUESTIONS

ADVENTURE TIME! 🚂



INSTALLATION

```
$ git clone https://github.kdc.capitalone.com/secollege/meanworks.git
$ cd meanworks
$ npm install
$ npm link
```

FOR WINDOWS: YOU'LL NEED PYTHON FROM .NET PACKAGE.

MEANWORKS USAGE

- `$ meanworks` OR `node meanworks.js`: LAUNCH MENU TO SELECT THE ADVENTURE AND MONITOR PROGRESS
  - `$ meanworks verify`: VERIFY SOLUTION
- `$ meanworks verify YOUR_FILE_NAME`: TO VERIFY THAT YOU HAVE FINISHED AN EXERCISE WITH A FILENAME
- `$ meanworks help`: TO GET HELP WITH THE WORKSHOP

MEANWORKS USAGE

- `$ meanworks solution`: TO SHOW THE SOLUTION FOR THE CURRENT EXERCISE
  - `$ meanworks verify skip` TO SKIP IT.

MEANWORKS ADVENTURES

- 01-NODE-NPM
- 02-MONGODB
- 03-INSTALLS
- 04-FOLDER
- 05-SEED

MEANWORKS ADVENTURES

- SEED
- ENDPOINTS
- 07-UI-TRANSACTIONS
- 08-UI-ACCOUNTS
- 09-UI-MAIN

MEANWORKS ADVENTURES

1. U
2. UI TABLE
3. CRUD-API
4. CRUD-UI

MEANWORKS DEMO

IT'S DEMO TIME! 🔭🖥 👌

FIRST ADVENTURE: INSTALL COMPATIBLE VERSION OF NODE.JS (5.1.0).

EXERCISE

IT'S TIME TO BUILD THE APP BY SOLVING ADVENTURES!

1. RUN `$ node meanworks` IN YOUR TERMINAL.
2. SOLVE ADVENTURES ONE-BY-ONE STARTING FROM THE TOP.
3. HAVE FUN.

USE COMMAND TO RUN, VERIFY OR LOOK THE OFFICIAL SOLUTION.

FEEDBACK

BUGS? SEND THEM TO ME. I 😋 THEM FOR BREAKFAST!

🐛🐛🐛 👉 https://github.com/azat-co/meanworks/issues

YOU ARE ⭐!

WORKSHOP TIME

```
$ node meanworks
```