Cuttlesoft

If You Don't Node, Just Ask

How to build a web application with Node & AngularJS (MEAN Stack)

Your Guide

Who I am and how I came to be here

Katie Russ

Full Stack Developer

Facts

- Tallahassee native but attended the University of Florida
- Worked at GNV tech startup for 2+ years after graduating (RoR)
- At Cuttlesoft 1+ year after moving back to TLH (Python/JS)

Fun Facts

- I likely know more about pigeons than anyone you'll ever meet
- I make "pysanky" (Ukranian Easter eggs)
- Nerd things are my favorite things

Varric

Ellie

katie@cuttlesoft.com

https://github.com/katie7r

What is MEAN?

The MEAN stack is a common full stack for web development, consisting of
MongoDB (database), Express (server framework), AngularJS (front-end framework), and Node.js (server runtime environment)

MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

Node + Express

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Express is a fast, unopinionated, minimalist web framework for Node.js.

Angular

AngularJS is a fully extensible front-end JavaScript framework, particularly useful in building single-page web applications.

MongoDB (+ Mongoose)

MongoDB

- Documents made up of key-value pairs
  - Similar to a row in a relational database
  - Dynamic schema, not pre-defined
    - Documents in collection may or may not have the same fields as other documents in the collection, or data/types within the same fields
- Collections contain documents
  - Similar to tables in a relational database
- Databases contain collections
  - Similar to databases in a relational database

Mongoose

- Object modeling tool that makes it super easy to work with MongoDB

- You could replace MongoDB + Mongoose with your database/adapter/ORM/ etc. of choice
  - For example:
    PostreSQL database + Bookshelf.js ORM + Knex.js query builder

```
var mongoose = require('mongoose');

// Define model(s)
var Thing = mongoose.model('Thing', {
  name: String
});

// Use model(s)
Thing.create({ name: 'Foo' }, function(err, thing) {
   if (err) console.log(err);
   console.log(thing);
});
```

Node + Express

Node.js

- JavaScript environment for running our server
- Module exporting / importing
- Asynchronous and non-blocking

```
// /somewhere.js: export
var thing = { a: 1, b: 2 };
module.exports = thing;


// /elsewhere.js: import
var something = require('./somewhere');
// something.a === 1;
// something.b === 2;
```

Express

- Node.js web application framework
- Provides configuration options, utility functions, middleware, etc. for setting up and running our application
- Huge help for building APIs (and routes in general)

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
})

app.listen(3000, function () {
  console.log('Example listening on port 3000!');
});
```

AngularJS

Getting Started

- For an AngularJS application, you need (at minimum):
  - Include AngularJS (npm install, bower install, script tag from CDN, etc.)
  - Angular root element (ng-app)

```
<!doctype html>
<html ng-app>
<head>
  <script src="https://cdnjs.cloudflare.com/ajax/
  libs/angular.js/1.6.2/angular.min.js"></script>
</head>
...
```

Starting with Modules

- Once you create a module for your application (next slide), you can specify it when setting the root element (ng-app)
  - This makes available within the application everything that is available with the specified module

```
<!doctype html>
<html ng-app="MyApp">
// Assuming our module is called "MyApp",
// any of the controllers, directives, etc.
// contained within "MyApp" will be accessible
```

AngularJS

Continued

```
<!-- input "user.name" -->
<input type="text" ng-model="user.name" />

<!-- output "user.name" -->
<p>{{ user.name }}</p>
```

Data-binding

- Changes to value in one location can update another location
- Example: You have a form input for a user to enter their name and a preview of what the user's name will look like on their profile. With Angular's data-binding, changes to the value of the form input can immediately update the preview without any manual DOM manipulation.

```
// module dependency of 'ui.router'
app.module('MyApp', ['ui.router']);

// controller dependency of '$scope'
app.module('MyApp')
   .controller('MyCtrl', function($scope) {
     $scope.whatIsAngular = 'such fun';
   }
```

Dependency injection

- Each controller, service, directive, etc. clearly describes what its dependencies are
- Each controller, service, directive, etc. works with its own dependencies

AngularJS

Continued

```
app.module('MyApp')
   .controller('ColorsCtrl', function() {
     // anything assigned to 'vm' will be available
     // in the view where your controller is applied
     var vm = this;
     vm.myFavorite = '#BADA55';
});

<!-- with 'controllerAs' syntax, access controller
with whatever name that you give it -->
<div ng-controller="ColorsCtrl as colors">
  My favorite color is {{ colors.myFavorite }}.
</div>
```

Controllers

- Controllers are "the behavior behind the DOM elements"
- Allow logic, service calls, functions, initializations, etc. to be moved out of the views while still controlling the views
- Help manage application behavior
- Controller scope only available where it is assigned
  - ng-controller directive
  - State/directive/component definition
- "controllerAs" syntax essentially assigns controller to whatever name you give it, for easy use within view

AngularJS

Continued

```
app.module('MyApp').service('fruitsSvc',
function($http) {
   var service = { getFruits: getFruits };
    return service;

   function getFruits() {
       return $http.get(fruitsApiUrl); // assume set
    }
});

app.module('MyApp').controller('FruitsCtrl',
function(fruitsSvc) {
   var vm = this;
   fruitsSvc.getFruits().then(function(res) {
      vm.fruits = res.data;
   }, function(err) { console.log(err); });
});
```

Services, Factories, and Providers

- Objects whose API you get to define yourself
  - Example: a configuration service that holds and distributes configuration variables as needed
  - Example: a service to manage CRUD for an existing object/model
  - Example: a provider for configuring external modules during application configuration

AngularJS

Continued

```
// Directive
app.module('MyApp').directive('appFooter',
function() {
   return {
      restrict: 'E', // 'E'lement v 'A'ttribute
      controller: 'footerCtrl', // assume defined
      controllerAs: 'footer',
      templateUrl: 'path/to/footer.html',
      scope: { bgColor: '=' }
   };
});

<!-- results in 'path/to/footer.html' template with
'footerCtrl' as 'footer' and 'bgColor'='#bada55' -->
<app-footer bg-color="#bada55"></app-footer>
```

Directives/Components

- Directives are essentially custom HTML elements or attributes
- Components are simplified directives, typically consisting of a template, a controller
- Data-bindings can be one- or two-way (i.e., changes to the model outside the directive may be reflected inside the directive / changes to the model inside the directive may be reflected outside the directive)
- Examples of built-in Angular directives:
  - ng-app / ng-controller
  - ng-if / ng-show / ng-hide
  - ng-repeat
  - ng-click

AngularJS

Continued

```
// Directive
app.module('MyApp').filter('capitalize', function()
{
   return function(input) {
      if (!input) return "";
      return input.charAt(0).toUpperCase() +
         input.substr(1);
   };
});

<!-- call the filter with a pipe -->
<p>{{ someStringToFormat | capitalize }}</p>
```

Filters

- Filters take some input and return some output, typically in terms of formatting a value to display to users (e.g., in templates)
- Can be defined without additional params or can pass in options
- Examples of built-in Angular filters:
  - lowercase / uppercase
  - currency
  - orderBy
  - json

To the Code!

github.com/cuttlesoft/mean-workshop

Resources

Workshop Resources

Workshop code: http://github.com/cuttlesoft/mean-workshop

MongoDB: https://docs.mongodb.com/

Express: http://expressjs.com/en/guide/routing.html

AngularJS: https://docs.angularjs.org/guide

Node.js: https://nodejs.org/en/docs

Additional Resources

scotch.io guides: https://scotch.io/tag/javascript

MEAN.io starter: http://mean.io/

MEAN.js starter: http://meanjs.org/

Angular style guide: https://github.com/johnpapa/angular-styleguide

Angular UI-Router: https://github.com/angular-ui/ui-router

Angular UI Bootstrap: https://angular-ui.github.io/bootstrap/

Thanks!

@cuttlesoft • info@cuttlesoft.com

Cuttlesoft