MEAN

M- Mongo DB

E- Express

A-Angular

N- Node

Microsoft Virtual Academy

<https://www.youtube.com/watch?v=QdFIyH-Hs9Q&list=PLsrZV8shpwjPuVqMcqAsxTHgGm8lW15nN&index=2>

Github

<http://github.com/microsoftlearning/chirp>

Pre-requisite

* Strong grasp of JavaScript
* Familiar with HTML / CSS Bootstrap
* Introduction to Angular MVA
* Mongo DB Jumpstart

D:\Litopascua\GitHub\AngularJS\Tron

* One language to Rule Them All
* All Open Source
* Huge Module Library
* Get Started Quickly

Angular JS

Our apps entire front end

* Open source maintained by Google
* Client Side MVC ( MVVM ) framework
* Excellent data bindings
* Easy to test
* **Single page apps**

If you click or load another link or loading another pages

**Even though the URL changes your not making a full request to the server for loading completely new page your only loading parts of the page, and your doing it on 1 single page**

**Where to get Angular?**

<https://docs.angularjs.org/misc/downloading>

* Extremely Lightweight – download/include only what you need
* Angular.js
* Angular route.js
* Angular-resource.js
* Single page applications

**Node.js**

Nodejs.org/download/

Lightweight web server

* Built on Google Chromes V8 javascript engine
* Extremely Lightweight and Efficient

Sample

Var http=require(‘http’);

http.createServer(function(req,res){

read.writeHead(200,{‘Content-Type’:’text/plain’});

res.end(‘Hello\n’);

}).listen(1337,’127.0.0.1’);

Console.log(‘Server running ‘);

**Express**

* Minimal and flexible node.js web application framework
* Abstract away a lot of low level logic
* Help organize your Node App into a MVC structure

**How to install Express**

* Npm install –g express

**MongoDB**

<http://www.mongodb.org/downloads>

**Our Datastore**

* Top NoSQL Database
* Open Source, maintained by MongoDB( formerly 10gen)
* JSON like syntax
* Key-values stores

Module Overview

* What is front end framework?
* Directives

How you insert angular code to html codes

* Modules and Controllers

Application logic

* Models and Data Bindings

Display to user our data

* Routing

Backend directions

<https://www.facebook.com/messages/t/1356241312>Front End Frameworks

A framework not a library

* JQuery is a libray with functions that makes javascript interaction less painful
* Angular.js is an MVC framework . You use it to structure out your applications look (view) from your data(model) and the logic and functions that are executed (controller). It’s also extremely test driven.

**Why use Angular?**

* What HTML would look like if it was designed from Web Apps.
* Keep your code organized and structure
* Two way data bindings
* Great for Single page applications
* Easy to test
* Two way databindings ( if you change in view and will automatically change in your controller)

**Other Front End Frameworks**

* **React.js 🡪 The V highly much more performance**
* **Ember.js 🡪 more ruby on rails like**
* **Backbone.js 🡪 ligther / doesn’t have data bindings**

**Directives**

* Angular-only HTML attributes
* Attaches some specific behavior to the element
* Usually begins with ‘ng-‘ or ‘data-ng’

Sample

*<html ng-app>*

*<head>*

*<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.0/angular.js"></script>*

*</head>*

*<body>*

*<input type="text" ng-model="inputText" placeholder="World" />*

*<h1>Hello {{inputText}}</h1>*

*</body>*

*</html>*

**Modules and Controllers**

**Modules**

* A container for your application
* Var myModule= angular.module(‘myApp’,[]);

**Controllers**

* Contains the business logic for the part of your application
* Set up your data to be viewed in your HTML

(massage your data, ,reset in the controller, )

* myModule.controller(‘myController’, function($scope){});

(scope everything passed around to your views )

**Dependency Injection**

* How we specify the dependencies that an Angular component will need.

**Templates**

* The angularized HTML we’re created
* Use to render the View ( what the end-user will see)