**Chap 1, 2, 3 Intro**

**Chap 4 What is a Single Page Application (SPA)?**

MEAN provides a dynamic single-page app. It appears to have different pages but this all do to the front end scripting modifying the DOM.

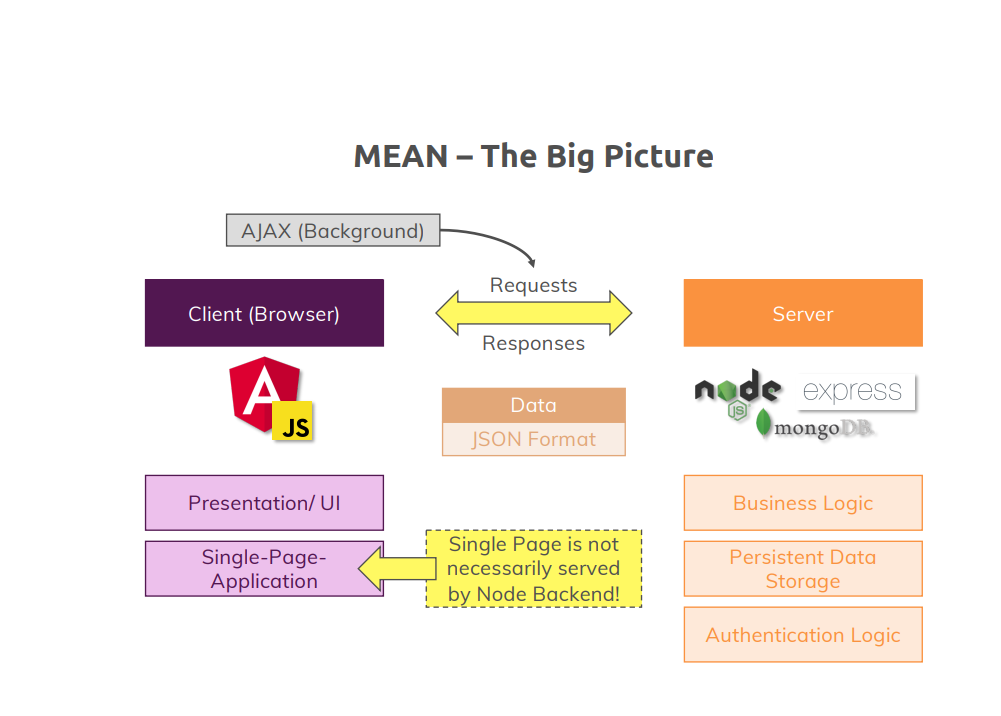
**Chap 5 How Does a MEAN Stack work?**

Mongo: non-SQL database

Express: AJAX is used and data is transmitted using JSON format.

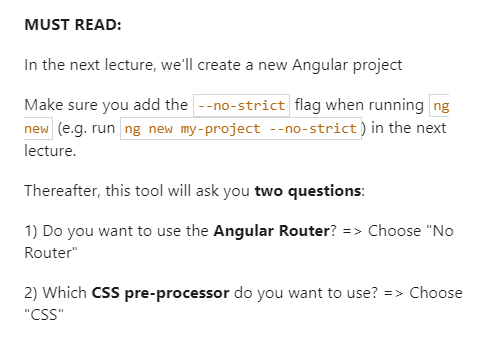
Angular: front end. Handles different screens: PC, tablet, and mobile

Node: JavaScript runtime. Provides access to server.



|  |  |
| --- | --- |
| ng | This is the angular CLI |

**Chap 6 Must read notes**



**Chap 7 Installing Node and Angular CLI**

Download and install node (see other notes)

Install Angular (see Angular course)

cd ~

mkdir project-mean

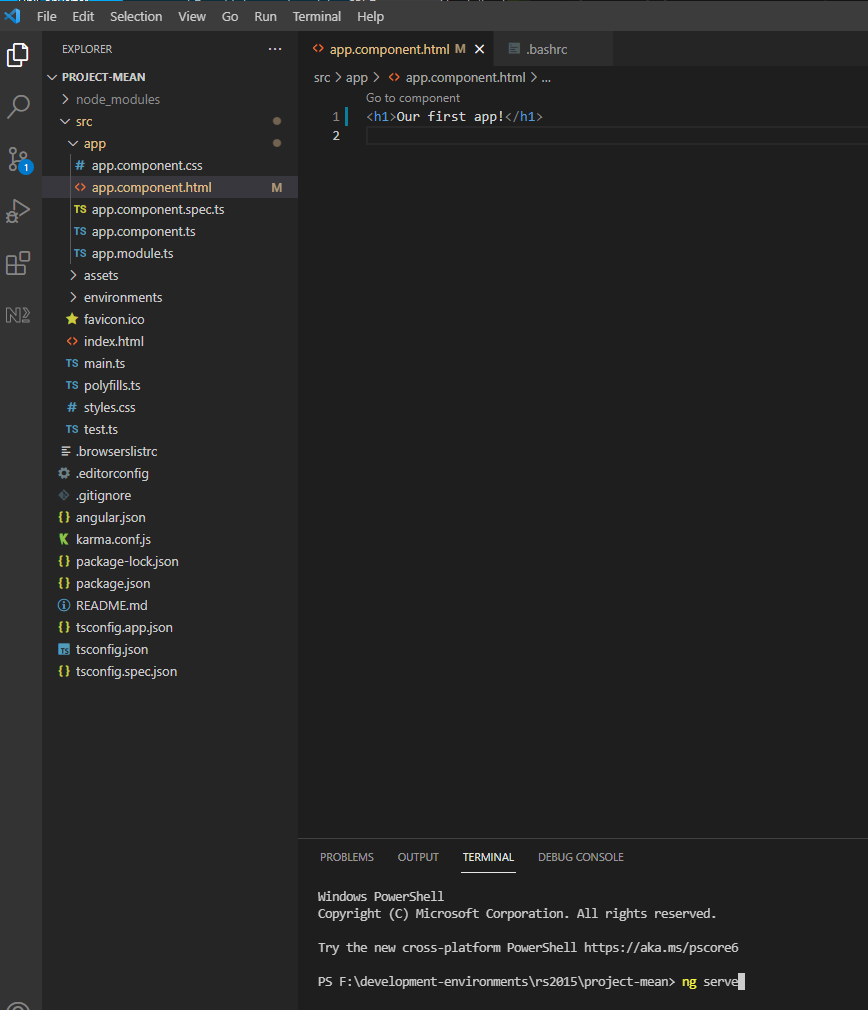
cd project-mean

ng new --no-strict first-project

cd first-project/

click on Terminal->New

then type ng serve



Open [**http://localhost:**4200/](http://localhost:4200/) in chrome

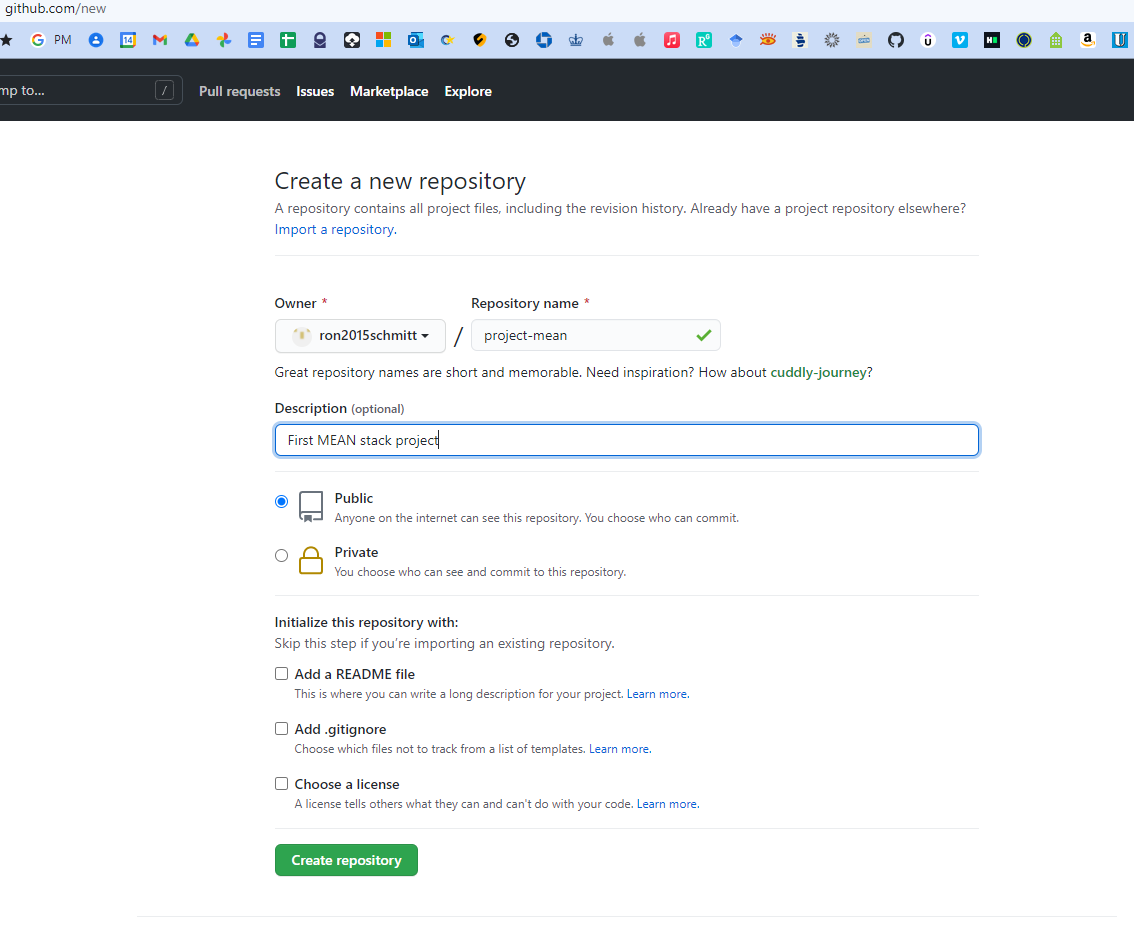
The following worked for github but probably is not best way to do it.

cd ~/project-mean

git init

git add .

create new repo on github

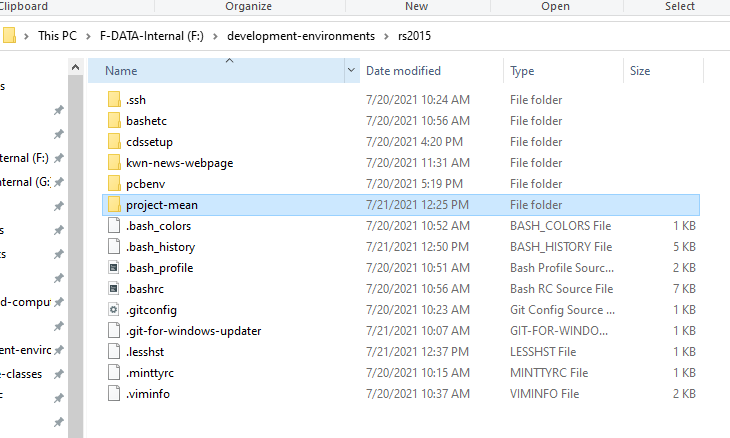


git branch -M main

git remote add origin https://github.com/ron2015schmitt/project-mean.git

git push -u origin main

Right-click folder and open in VSCode



**Chap 8 Installing Our IDE**

Install VSCode (see other notes)

Install VSCode extensions:

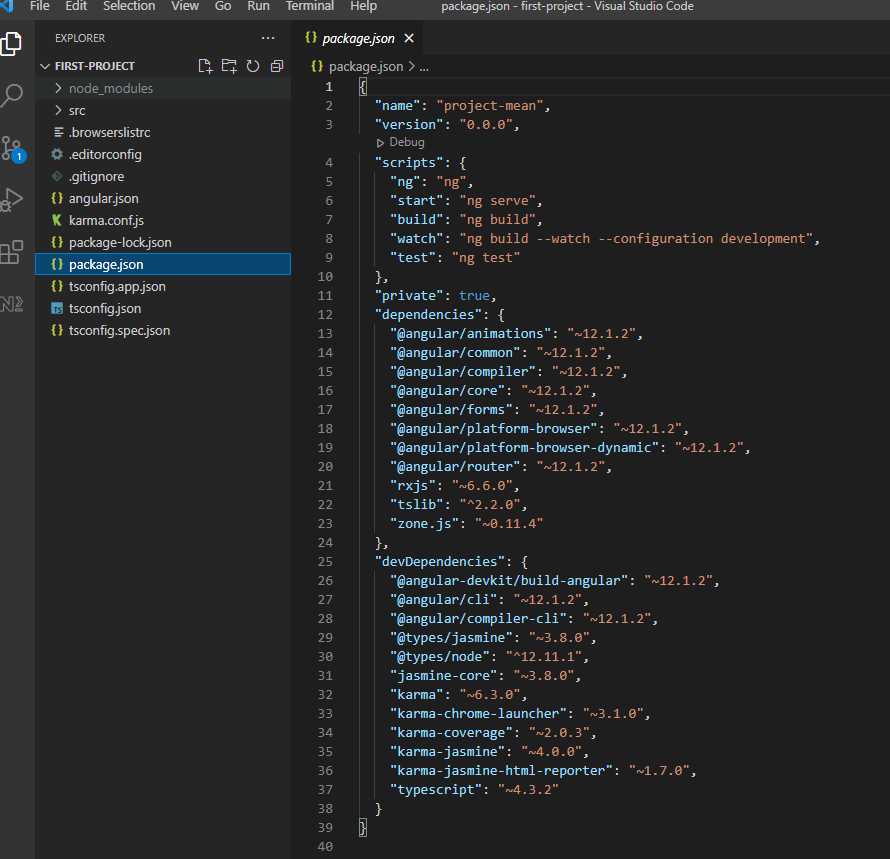
* Angular Essentials
* Material Icon Theme

**Chap 9 Exploring the Project Structure**

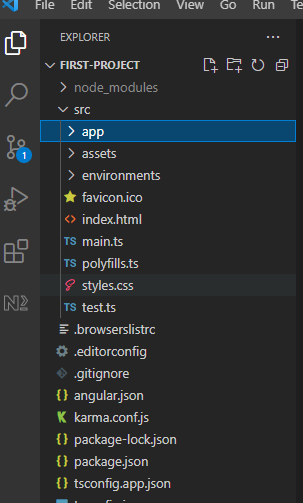
Completed through Chap 8

Open project-mean folder in VSCode

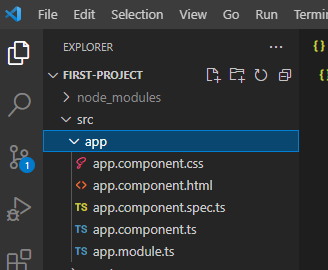
The package.json file list all the node packages needed, our dependencies. When you type npm install these package downloaded and built in the node\_modules folder



Our source code resides in the src folder



The main Angular component is defined in the app folder



**Chap 10 Course Outline**

Data Pagination is the process of downloading large amounts data from the database as needed. Example is a grid with thousands of rows.

**Chap 11 How to get the most out of this course**

Code along with the video

Ask questions on Q&A

**Chap 12: section resources**

**Chap 13: Angular: Introduction**

We’re going to build a mini social network

**Chap 14: Understanding the folder structure**

See Chap 7 of the Angular course

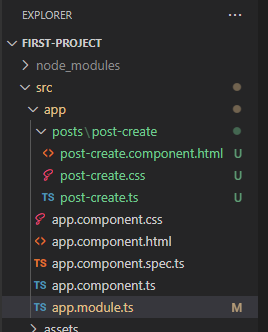
**Chap 15: Understanding Angular Components**

Angular constructs web page via components.

Similar to a word processor, the page layout itself is divided up into regions that are defined by components.

**Chap 16: Adding Our Component**

Create folder and file structure highlighted below in green text



Leave the CSS blank.

|  |
| --- |
| post-create.component.html |
| <p>post-create</p> |

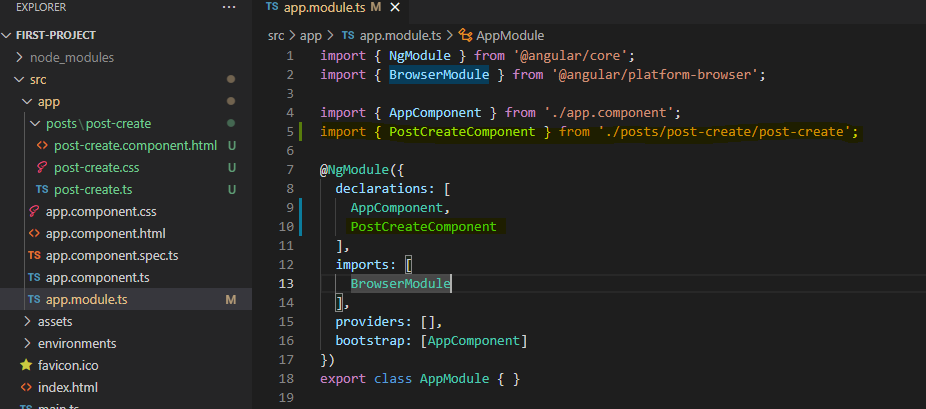
|  |
| --- |
| post-create.ts |
| import { Component } from "@angular/core";  @Component({    selector: 'app-post-create',    templateUrl: './post-create.component.html'  })  export class PostCreateComponent {  } |

Add the following line to app.module.ts

import { PostCreateComponent } from './posts/post-create/post-create';

Then also add PostCreateComponent to the declarations

result:



Now in the app html file, write

|  |
| --- |
| app.component.html |
| <h1>Our First App!</h1>  <app-post-create></app-post-create> |

|  |
| --- |
| Our app in Chrome |
|  |

**Chap 17: Listening to Events**

Rename post-create.ts to post-create.component.ts including all its references

Add function onAddPost to definition

|  |
| --- |
| post-create.component.ts |
| import { Component } from "@angular/core";  @Component({    selector: 'app-post-create',    templateUrl: './post-create.component.html'  })  export class PostCreateComponent {    onAddPost() {      alert('Post Added!');    }  } |

Change html to have a text area and a button tied to that function

|  |
| --- |
| post-create.component.html |
| <textarea rows="6"></textarea>  <hr>  <button (click)="onAddPost()">Save Button</button> |

|  |
| --- |
| Our app in Chrome |
|  |

**Chap 18: Outputting the content**

Add function onAddPost to definition

|  |
| --- |
| post-create.component.ts |
| import { Component } from "@angular/core";  @Component({    selector: 'app-post-create',    templateUrl: './post-create.component.html'  })  export class PostCreateComponent {    onAddPost() {      alert('Post Added!');    }  } |

Change html to have a text area and a button tied to that function, with initial value as shown.

|  |
| --- |
| post-create.component.html |
| <textarea rows="6" value="hello"></textarea>  <hr>  <button (click)="onAddPost()">Save Button</button> |

|  |
| --- |
| Our app in Chrome |
|  |

Now, let’s use Angular in the **textarea**: We use [value] and "'hello'" for the value.

|  |
| --- |
| **post-create.component.html** |
| <textarea rows="6" [value]="'hello'"></textarea>  <hr>  <button (click)="onAddPost()">Save Button</button> |

Lastly, we put a variable inside the method

|  |
| --- |
| **post-create.component.ts** |
| import { Component } from "@angular/core";  @Component({    selector: 'app-post-create',    templateUrl: './post-create.component.html'  })  export class PostCreateComponent {    newPost = 'type here';    onAddPost() {    }  } |

Change html to have a text area and a button tied to that function.

|  |
| --- |
| **post-create.component.html** |
| <textarea rows="6" [value]="newPost"></textarea>  <hr>  <button (click)="onAddPost()">Save Button</button> |

|  |
| --- |
| **Our App in Chrome** |
|  |

**Chap 19: Getting the user input**

First version: use 2 one-way bindings.

We bind the **textarea** value to **#postInput** and then feed to the onAddPost method

|  |
| --- |
| **post-create.component.html** |
| <textarea rows="6" [value]="newPost" #postInput></textarea>  <hr>  <button (click)="onAddPost(postInput)">Save Button</button>  <p>{{ newPost }}</p> |

|  |
| --- |
| **post-create.component.ts** |
| import { Component } from "@angular/core";  @Component({    selector: 'app-post-create',    templateUrl: './post-create.component.html'  })  export class PostCreateComponent {    newPost = 'type here';    onAddPost(postInput: HTMLTextAreaElement) {      console.log(postInput);      this.newPost = postInput.value;    }  } |

|  |
| --- |
| **Our App in Chrome** |
|  |

Type and hit save and the value at the bottom now updates

Second version: use two-way binding.

Add import line for NgModule and add to imports array to **app.module.ts**

|  |
| --- |
| **app.module.ts** |
| import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { FormsModule } from '@angular/forms';  import { AppComponent } from './app.component';  import { PostCreateComponent } from './posts/post-create/post-create.component';  @NgModule({    declarations: [      AppComponent,      PostCreateComponent    ],    imports: [      BrowserModule,      FormsModule    ],    providers: [],    bootstrap: [AppComponent]  })  export class AppModule { } |

Use the forms modules. Note this **updates the variable for every key stroke** not just when hitting the button

|  |
| --- |
| **post-create.component.html** |
| <textarea rows="6" [(ngModel)]="userValue"></textarea>  <hr>  <button (click)="onAddPost()">Save Button</button>  <p>{{ newPost }}</p> |

|  |
| --- |
| **post-create.component.ts** |
| import { Component } from "@angular/core";  @Component({    selector: 'app-post-create',    templateUrl: './post-create.component.html'  })  export class PostCreateComponent {    newPost = 'NO CONTENT';    userValue = '';    onAddPost() {      console.log(this.userValue);      this.newPost = this.userValue;    }  } |

|  |
| --- |
| **Our App in Chrome** |
|  |

Completed thru Chap 19