**NPM:** Node package manager, allows you to install node packages instead of going to the website and downloading.

**ng new HelloWorld --dry-run:** See what a command would do without running.

**ng new HelloWorld --skip-install:** See what a command would do without running.

**<app-root></app-root>:** Where angular starts.

**Main.ts:** Telling angular where to start.

**Root component: App.component.ts:**

This is the root component of the app.

**Angular.json:** Has all configurations for the project.

**ng new HelloWorld --prefix hw --style scss --routing:** Style files should be scss.

**npm run-script**

**ng serve:** uses webpack behind the server.

**Tour of heroes** is the official kind of project by angular which explains a lot of concepts, components, directories etc.

**Decorater:** @component, decorates the class. Angular needs to know something about a component. @NgModule decorator.

**ng generate component greeting:** Creating components.

**Ng g c greeting**

**Ng g c components/TodoInput**

**ng build:** How to build.

**ng build --prod:** How to build it for production.

**http-server /project-path:** Give a server to run your build.

**ng-test:** Start testing.

spec.ts file has unit test in them.

**ng e2e:** Testing the whole app.

**Node js:** Runtime environment to let Javascript run on your machine.

Type node in terminal.

**Nvm:** Allows you to install multiple node versions on a single machine.

**Mongod:** Run the mongo server.

**Mongo:** Play with mongo.

**Create databse**

Use hello-world

Show dbs (it won’t show the new one, but it won’t be registered).

**In mongo:**

Tables = Collections

Rows = Documents

show collections;

**Adding a document in a collection.**

db.users.insert({firstName: “Ahsan”, lastName: “Ayaz”})

**List results**

db.users.find();

**Pretty**

db.users.find().pretty();

**Finding one specific:**

db.users.find({firstName: "Asaad"})

**Removing specific entry:**

db.users.remove({firstName: "Asaad"})

**Removing all:**

db.users.remove({})

**Running a node file**

node main.js

**Killing ports**

lsof -i :3000 (where 3000 is your current port in use)

kill -QUIT PID

**Import modules**

FormsModule

Inside app.module.ts

import { FormsModule } from '@angular/forms';

inside `imports`

you add:

`FormsModule`

**Add function**

<button (click)="addTodo()">Add</button>

**Smart components**

Interact with other components.

**Dumb components**

Do not interact with other components.

**Input binding**

Pass actual values from parents to child components, we do that by property binding.

We first have to make the child to be able to accept values.

We do that by adding: @Input() and also import Input from Angular

Also add [list]='todoList' to the tag in html.

**Angular pipes**

Syntax: {{todoItem.text | lowercase}}

ng g pipe pipes/LimitTo

**Router outlet**

We have a container box, in which the content would be changed.

An application would have the header that would remain the same.

So beneath the header we have a box which changes content.

That is the **router outlet**.

This supports switching pages inside the outlet.

**Redux**

Ng-rx

**How to create a service**

Ng g s myService

Services are injectable.

Use for http calls, or for running large for loops.

**Demo users**

Randomuser.me

Ajax call.

**Directives**

Use when are to interact with the actual DOM elements.

No DOM manipulation should be done in the component.

Any direction that has a \* before it, is a structural directive.

It actually modifies the html itself. Not the color, adding or removing element from the dom.

**Ng g d directives/directiveName**

**() Bind an event.**

**[] Input values.**

**[()] ngModel is both.**

**RSJX**

Reactive JS programming.

Observables.

Promise may return value once.

An observable may return values continuously whenever something happens.

Use $ to create observables or BehaviourSubject.

$colorSubject: BehaviourSubject<string> = new BehaviourSubject(‘black’);

Import from rxjs/internal/BehaviourSubject.