



# Angular

Vinod Kumar Kayartaya  
<https://vinod.co>  
vinod@vinod.co

# What is Angular?

- Angular is a JavaScript client side framework for creating powerful web or mobile applications
- Created and maintained by Google
- The most popular JavaScript framework as of date
- Create HTML-like elements (components)
  - a combination of HTML, CSS and JavaScript

# What is Angular?

- Complete re-write of AngularJS
- Component based
- No controllers or scope
- Streamlined dependency injection
- Can write apps in TypeScript, Dart or JavaScript (ES5/ES6)

# What is Angular?

- SPA (Single Page Application)
  - Just one file - index.html,
  - components rendered dynamically
- Cleaner code

# What is Angular?

- So many options
  - ES5
  - CoffeeScript
  - DART
  - ES6 and ES7
  - TypeScript

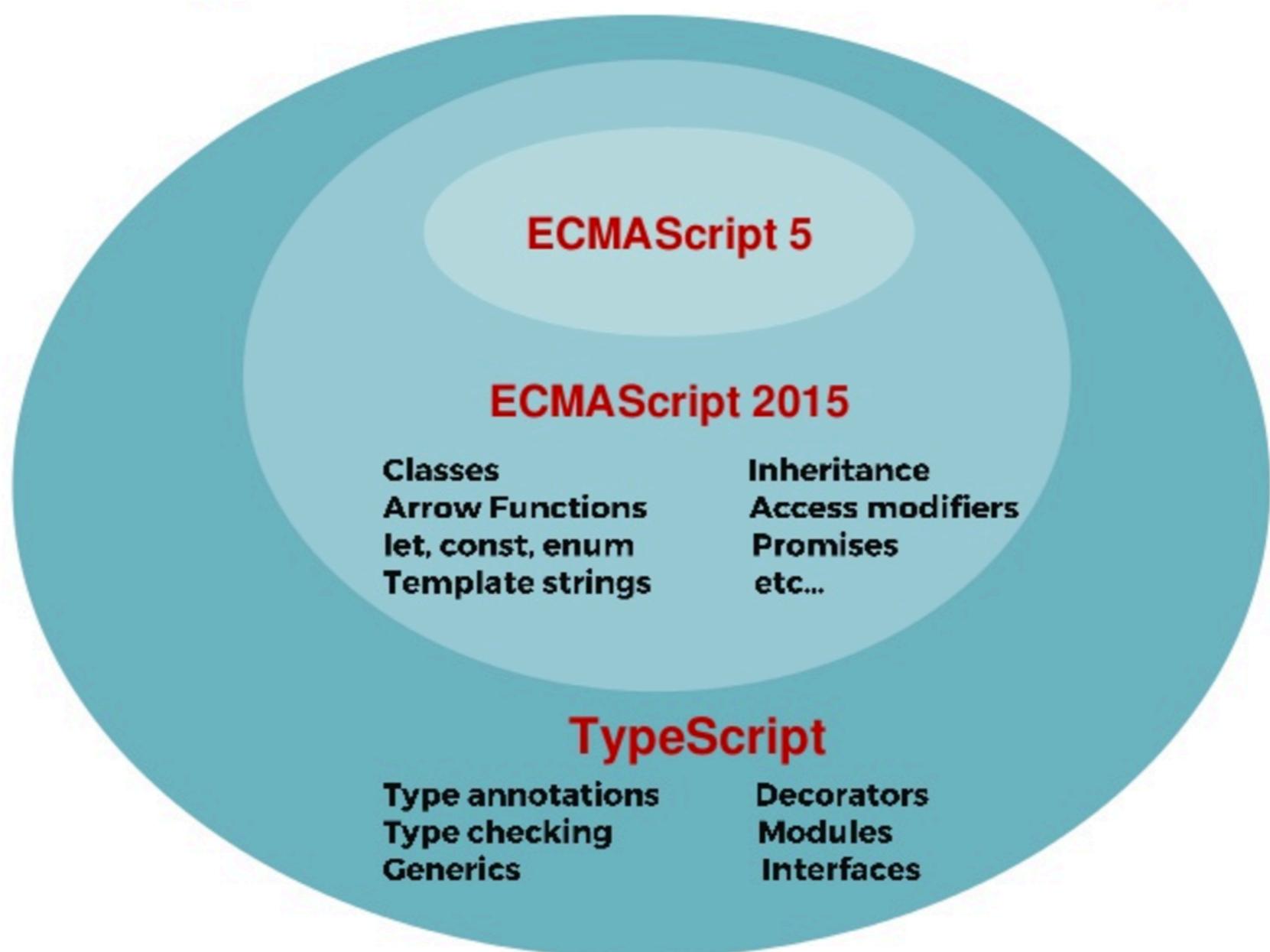
# What is Angular?

- Modular
- Testable
- Maintainable

# JavaScript advances

- ES6 (ES2015)
  - Classes
  - Modules
  - Decorators
- TypeScript
  - ES6 and ES7
  - Strong typing
  - Interfaces

# TypeScript = All JavaScripts



# Angular main concepts

- Module
- Component
- Template
- Data binding
- Metadata
- Service
- Directive
- Dependency injection

# AddressBook

search...

Mr. Patrick Warren Omaha

Ms. Angela Hawkins Charlottesville

Mr. Kevin Sims Dayton

Mr. Charles Davis San Antonio

Ms. Laura Henderson Seattle

Ms. Phyllis McDonald Charlotte

Mr. Ernest Torres New York City

Mr. Howard Lawrence Rochester

Ms. Ruth Ortiz Des Moines

Ms. Margaret Perkins Evansville

Mr. Douglas Kennedy Dallas

Ms. Paula Kelley Tulsa

© 2017 - all rights reserved

The App we are going to build

Mr. PATRICK WARREN

**Date of birth** 1965-09-27  
**Email id** pwarren0@mayoclinic.com  
**Phone** 1-(402)574-2910  
**City** Omaha  
**State** Nebraska  
**Country** United States



The App we are going to build

Contact details

**Firstname**

Patrick Warren

**Date of birth**

27/09/1965

**Gender**

Male  Female

**Email id**

pwarren0@mayoclinic.com

**Phone number**

1-(402)574-2910

**City**

The App we are going to build

AddressBook 

Contact details

**Firstname**

**Date of birth**

**Gender**

Male  Female

**Email id**

**Phone number**

**City**

---

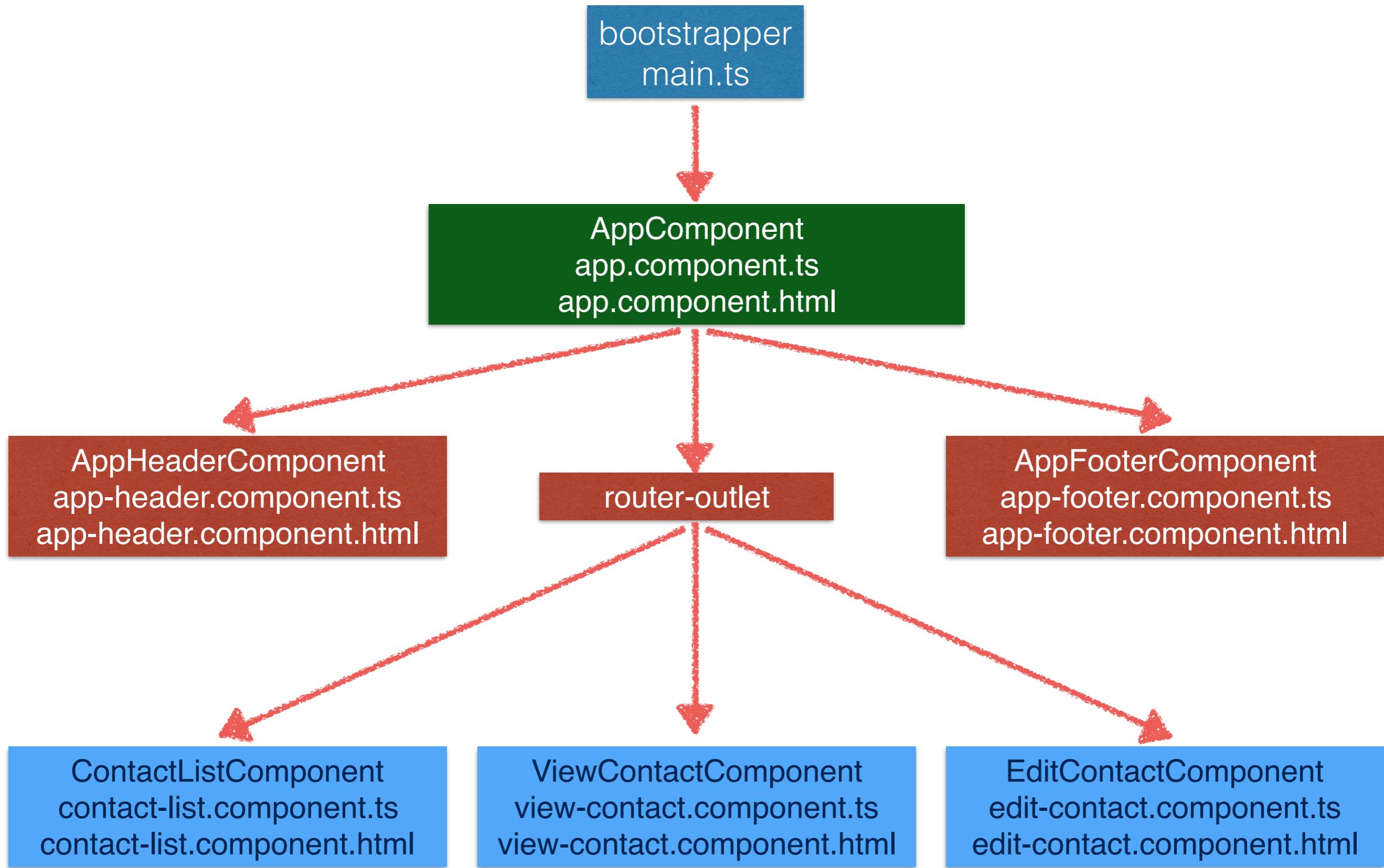
© 2017 - all rights reserved

The App we are going to build

# Component

- `@Component` makes a class a reusable component
- A component is a combination of data, functions, and HTML
  - Data —> state
  - Function —> behaviour
  - HTML —> view

# An app is a tree of components





AppRoot

HeaderComponent

router-outlet

FooterComponent



An app is a tree of components

```
<!DOCTYPE html>
<html>
<head>
  <title>ng2addressbook</title>
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

```
<app-header></app-header>
<div class="container" style="margin-top: 10px;">
  <router-outlet></router-outlet>
</div>
<app-footer></app-footer>
```

# Metadata

- Metadata is extra information which gives angular more info
- `@Component` tells angular that the class is a component
- `@Directive` tells angular that the class is a directive
- `@Injectable`, `@RouteConfig`, `@NgModule`, ...

# AppHeaderComponent

AddressBook



```
<div class="navbar navbar-default navbar-fixed-top">
  <div class="container">
    <div class="navbar-header">
      <a href="" [routerLink]="['']" class="navbar-brand">
        AddressBook</a>
      <a href="" class="btn btn-primary navbar-btn" href="" [routerLink]="['add-contact']"
         title="Add a new contact">
        <i class="glyphicon glyphicon-user"></i>
        <i class="glyphicon glyphicon-plus"></i>
      </a>
    </div>
  </div>
</div>
```

Importing resource from module

Decorator

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-header',
  templateUrl: "../templates/app-header.component.html",
})
export class AppHeaderComponent{
  constructor() {
  }
}
```

Component's state  
and behaviour  
defined here

# Template

- Is a way to describe a view using HTML
- Templates can be included with the component
- Or as an external file reference
- Best practice is to use an HTML file

# Template

Mr. PATRICK WARREN

Date of birth	1965-09-27
Email id	pwarren0@mayoclinic.com
Phone	1-(402)574-2910
City	Omaha
State	Nebraska
Country	United States



Data for the template (state)

```
@Component({
  selector: "edit-contact",
  templateUrl: "../templates/contact-form.com
})
export class EditContactComponent {
  contact: Contact = new Contact();
```

```
<div class="well">
  <p class="lead text-center">
    {{contact.gender | title}} {{contact.name|u
  </p>
  <div class="row">
    <label class="col-xs-4">Date of birth</label>
    <label class="col-xs-8">{{contact.dob}}</la
  </div>
  <div class="row">
    <label class="col-xs-4">Email id</label>
    <label class="col-xs-8">{{contact.email}}</l
  </div>
  <div class="row">
    <label class="col-xs-4">Phone</label>
    <label class="col-xs-8">{{contact.phone}}</l
  </div>
  <div class="row">
    <label class="col-xs-4">City</label>
    <label class="col-xs-8">{{contact.city}}</l
  </div>
  <div class="row">
    <label class="col-xs-4">State</label>
    <label class="col-xs-8">{{contact.state}}</l
  </div>
  <div class="row">
    <label class="col-xs-4">Country</label>
    <label class="col-xs-8">{{contact.country}}</l
  </div>
  <div class="row">
    <div class="col-xs-12 text-center">
      <a href="" [routerLink]="/edit-contac
        <span class="glyphicon glyphicon-pe
      </a>
      <a href="javascript:void(0)" class="bt
        (click)="deleteContact()">
        <span class="glyphicon glyphicon-tr
      </a>
    </div>
  </div>
</div>
```

# Databinding

- `{{ value }}` → interpolation
- `[property] = “value”` → property binding
- `(event) = “handler”` → event binding
- `[(ngModel)] = “property”` <→ two way binding

# Service

- “Substitutable objects that are wired together using dependency injection (DI)”
- Used to share code across an application
- Lazily initialised

# Directives

- A class with `@Directive` metadata
- Two kinds: attributes and structural
- Attribute directives alter the look or behaviour of an existing element
- Structural directives alter the layout by adding, removing, and/or replacing elements in the DOM
- ***A Component is a directive with a view***

# Dependency injection

- A way to supply a new instance of a class with the fully-formed dependencies it needs
- Most dependencies are services
- Angular knows which service/s a component depend on by looking at the types of its constructor parameters
- Services are injected by an ***Injector*** which uses a ***Provider*** to create the service

# Dependency Injection

injecting an instance of Http (service)

```
@Injectable()
export default class ContactService {

    private url:string = "http://localhost:1234/contacts";

    constructor(private http: Http){}
}
```

injecting an instance of ContactService

```
export class AddContactComponent {
    contact: Contact = new Contact();

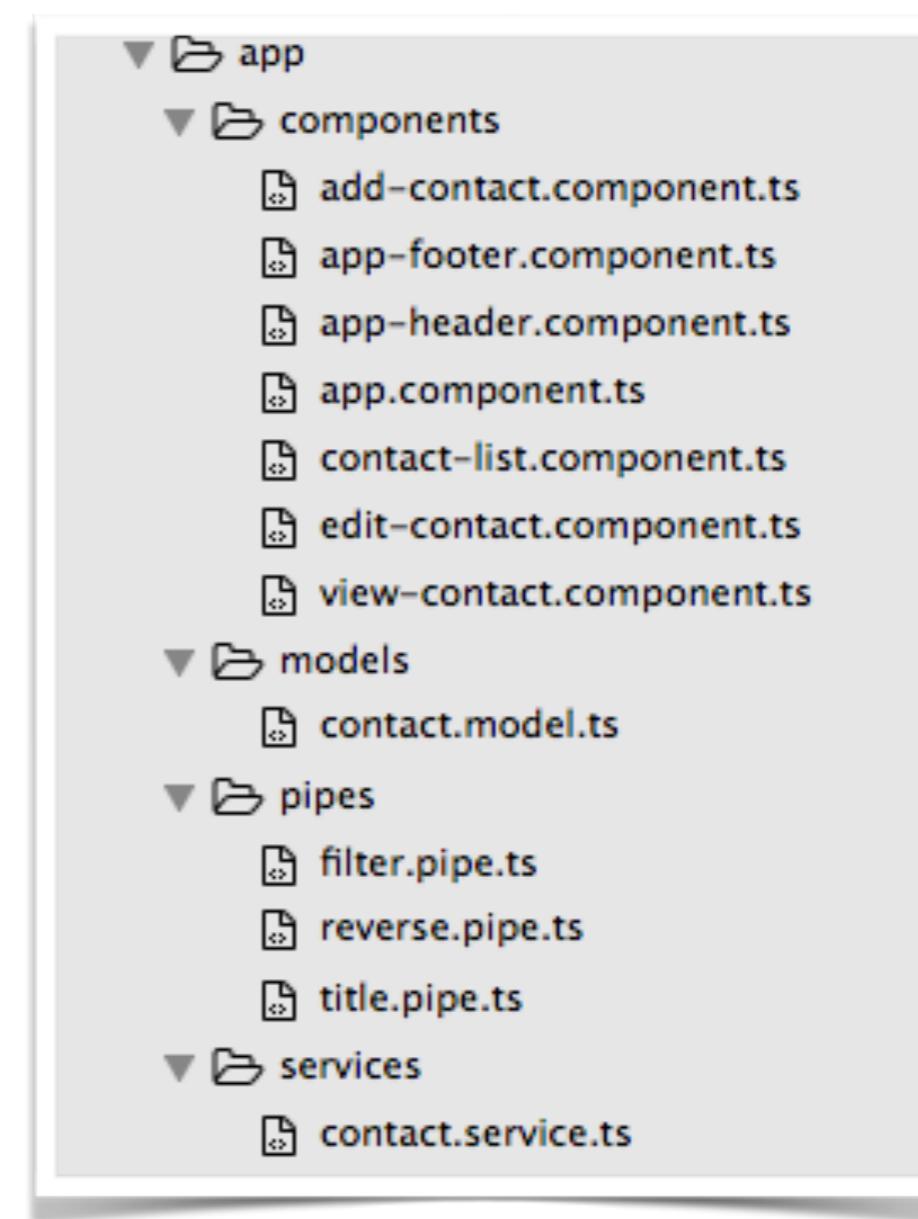
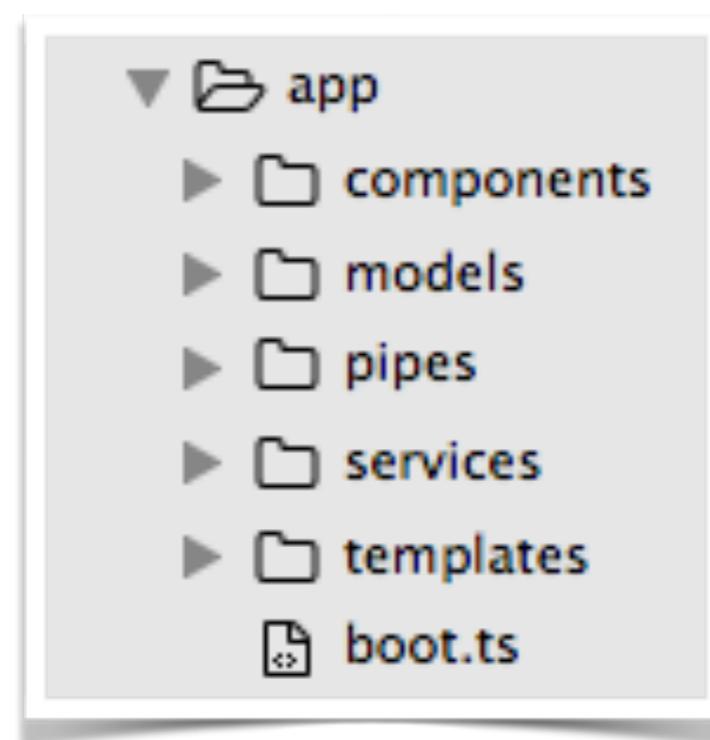
    constructor(private cs: ContactService){}
}
```

# Module

- Modules are optional, but a best practice
- export tells TypeScript that the resource is a module available for other modules
- import tells TypeScript which resource to import from a module
- Angular comes with a collection of library modules

# Module

...is basically a  
Script file in a folder.



## Exporting a resource in a module...

```
export class Contact {  
    public id: number;  
    public name: string;  
    public dob: Date;  
    public gender: string;  
    public email: string;  
    public phone: string;  
    public city: string;  
    public state: string;  
    public country: string;  
}
```

## Importing a resource from a module...

```
import { Contact } from "../models/contact.model";  
  
@Component({  
    selector: "contact-list",  
    templateUrl: "../templates/contact-list.component.html",  
})  
export class ContactListComponent {  
  
    contacts: Array<Contact> = [];  
    selectedContact: Contact;  
    token: string = "";
```

# Using router

- Four things to take care of:
  1. Import RouterModule, Routes from @angular/router
  2. Define the route config using RouterModule.forRoot()
  3. Use routerLink on hyperlinks or buttons
  4. Create a <router-outlet> to dynamically place a component

# 1. Import...

```
import { RouterModule, Routes } from '@angular/router';
```

## 2. Define routes

```
RouterModule.forRoot([
  {
    path: "", redirectTo: "/contact-list", pathMatch: "full"
  },
  {
    path: "contact-list", component: ContactListComponent
  },
  {
    path: "add-contact", component: AddContactComponent
  },
  {
    path: "view-contact/:id", component: ViewContactComponent
  }
], {
  useHash: true
}),
```

### 3. Links...

```
<li>
    <a href="" [routerLink]=["contact-list"]>Home</a>
</li>
<li>
    <a href="" [routerLink]=["add-contact"]>Add new contact</a>
</li>
```

```
<a href="javascript:void(0)" [routerLink]=["/view-contact", c.id] class="list-group-item"
*ngFor="let c of contacts">

<span>
    {{c.gender=="Male"? "Mr." : "Ms"}} {{c.firstname}} {{c.lastname}}
</span>
<span class="pull-right">{{c.city}}</span>

</a>
```

# 4. Router outlet

```
<div class="container" style="margin-top: 60px; padding: 0;">
  <router-outlet></router-outlet>
</div>
```

# Navigating via code

```
1 import { Component } from "@angular/core";
2 import ContactService from "../services/contact.service";
3 import { Contact } from "../../models/contact.model";
4 import { Router } from "@angular/router";
5
6 @Component({
7   selector: "contact-list",
8   templateUrl: "../templates/contact-list.component.html",
9 })
10 export class ContactListComponent {
11
12   contacts: Array<Contact> = [];
13   selectedContact: Contact;
14
15   constructor(private cs: ContactService, private router: Router){
16     cs.getAll().then(data=>{
17       this.contacts=data;
18     })
19     .catch(resp=>{
20       console.error(resp);
21     });
22     console.log(`There are ${this.contacts.length} contacts`);
23   }
24
25   viewContact(id: number): void {
26     this.router.navigate(['/view-contact', id]);
27   }
28 }
29 }
```

# Reading router parameters

```
1 import { Component, Input, OnInit } from "@angular/core";
2 import { Contact } from "../models/contact.model";
3 import ContactService from "./services/contact.service";
4 import { ActivatedRoute, Params, Router } from "@angular/router";
5 import "rxjs/add/operator/switchMap",
6
7 @Component({
8   selector: "view-contact",
9   templateUrl: "../templates/view-contact.component.html",
10 })
11 export class ViewContactComponent implements OnInit {
12
13   contact: Contact = new Contact();
14
15   constructor(private cs: ContactService,
16     private route: ActivatedRoute,
17     private router: Router){
18   }
19
20   ngOnInit(){
21     this.route.params
22       .subscribe(params => {
23         this.cs.getContact(+params["id"])
24           .then(contact=>{
25             this.contact = contact;
26           })
27       });
28   }
29 }
```

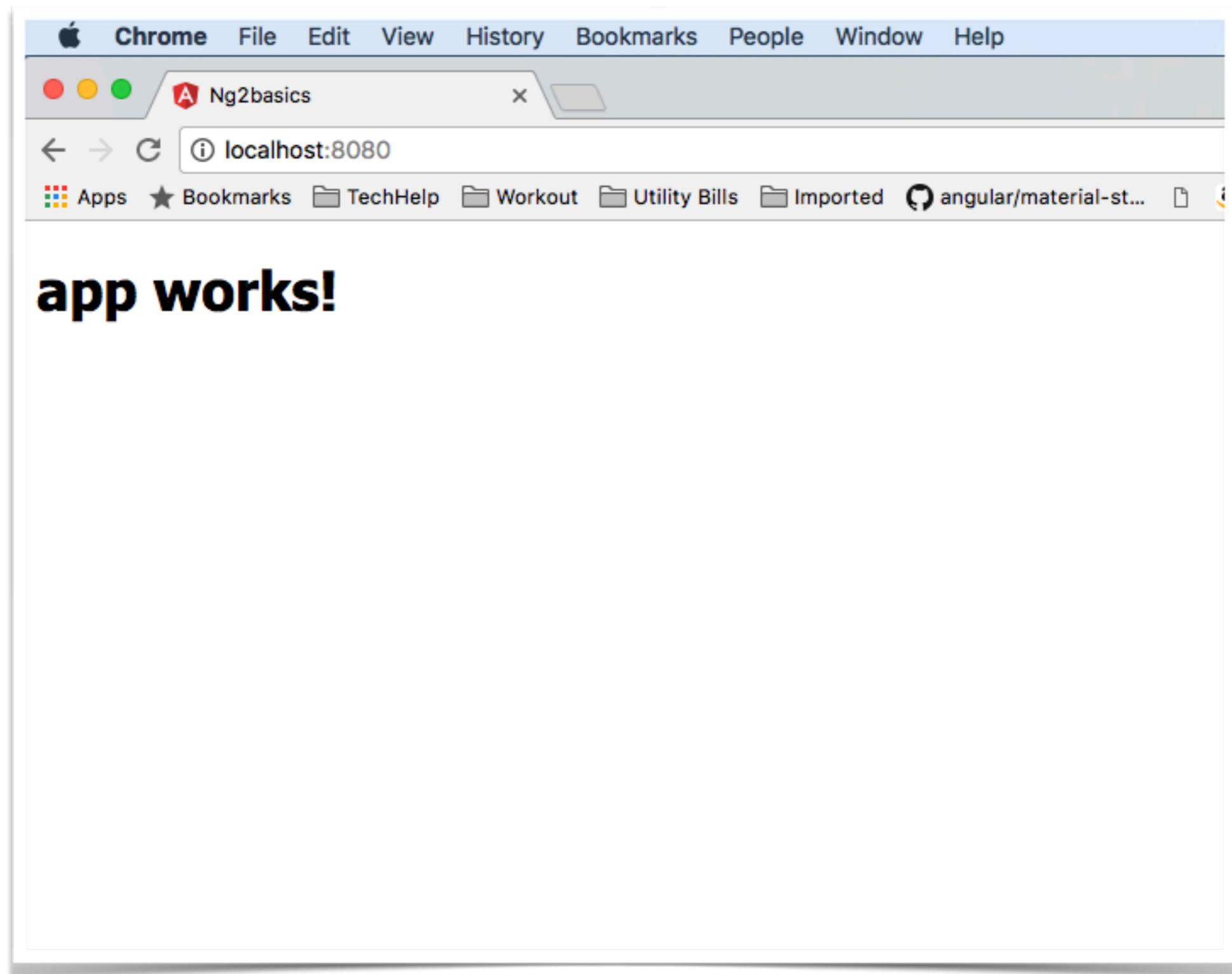
# Setup

- Create a work directory and init angular-cli
  - Type the command :
  - `ng new angular-demo-app`
  - It takes a long time to finish, have patience

# Run

- cd in to the app directory
- Run the app
  - ng serve --port 8080 -o

```
Vinods-MacBook-Pro:ng2basics vinodkumar$ ng serve --port 8080
[sass-work]
  fallbackLoader option has been deprecated - replace with "fallback"
  loader option has been deprecated - replace with "use"
  fallbackLoader option has been deprecated - replace with "fallback"
  loader option has been deprecated - replace with "use"
  fallbackLoader option has been deprecated - replace with "fallback"
  loader option has been deprecated - replace with "use"
  fallbackLoader option has been deprecated - replace with "fallback"
  loader option has been deprecated - replace with "use"
** NG Live Development Server is running on http://localhost:8080. **
Hash: 5837e716cf7e47276058
Time: 8565ms
chunk {0} main.bundle.js, main.bundle.map (main) 4.59 kB {2} [initial] [rendered]
chunk {1} styles.bundle.js, styles.bundle.map (styles) 9.96 kB {3} [initial] [rendered]
chunk {2} vendor.bundle.js, vendor.bundle.map (vendor) 2.84 MB [initial] [rendered]
chunk {3} inline.bundle.js, inline.bundle.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.
```



```
▼ └ ng2basics
    ► └ e2e
    ► └ node_modules
    ▼ └ src
        ▼ └ app
            └ app.component.css
            └ app.component.html
            └ app.component.spec.ts
            └ app.component.ts
            └ app.module.ts
        ► └ assets
        ► └ environments
        └ favicon.ico
        └ index.html
        └ main.ts
        └ polyfills.ts
        └ styles.css
        └ test.ts
        └ tsconfig.json
```

# main.ts

The image shows a screenshot of a code editor with the file 'main.ts' open. The code is written in TypeScript and follows the Angular bootstrap pattern. It starts with imports for polyfills, platformBrowserDynamic, enableProdMode, environment, and AppModule. It then checks if the environment is production and enables production mode. Finally, it bootstraps the AppModule using platformBrowserDynamic.

```
1 import './polyfills.ts';
2
3 import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
4 import { enableProdMode } from '@angular/core';
5 import { environment } from './environments/environment';
6 import { AppModule } from './app/app.module';
7
8 if (environment.production) {
9   enableProdMode();
10 }
11
12 platformBrowserDynamic().bootstrapModule(AppModule);
13 |
```

# src/app.module.ts

```
app.module.ts *  
1 import { BrowserModule } from '@angular/platform-browser';  
2 import { NgModule } from '@angular/core';  
3 import { FormsModule } from '@angular/forms';  
4 import { HttpClientModule } from '@angular/http';  
5  
6 import { AppComponent } from './app.component';  
7  
8 @NgModule({  
9   declarations: [  
10     AppComponent  
11   ],  
12   imports: [  
13     BrowserModule,  
14     FormsModule,  
15     HttpClientModule  
16   ],  
17   providers: [],  
18   bootstrap: [AppComponent]  
19 })  
20 export class AppModule {}  
21
```

src/index.html

index.html \*

```
1 <!doctype html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Ng2basics</title>
6   <base href="/">
7
8   <meta name="viewport" content="width=
9     <link rel="icon" type="image/x-icon"
10    </head>
11    <body>
12      <app-root>Loading...</app-root>
13    </body>
14  </html>
15
```

src/app/app.component.html

app.component.html \*

```
1 <h1>
2   {{title}}
3 </h1>
4
```

app.component.ts \*

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'app works!';
10 }
```

src/app/app.component.ts

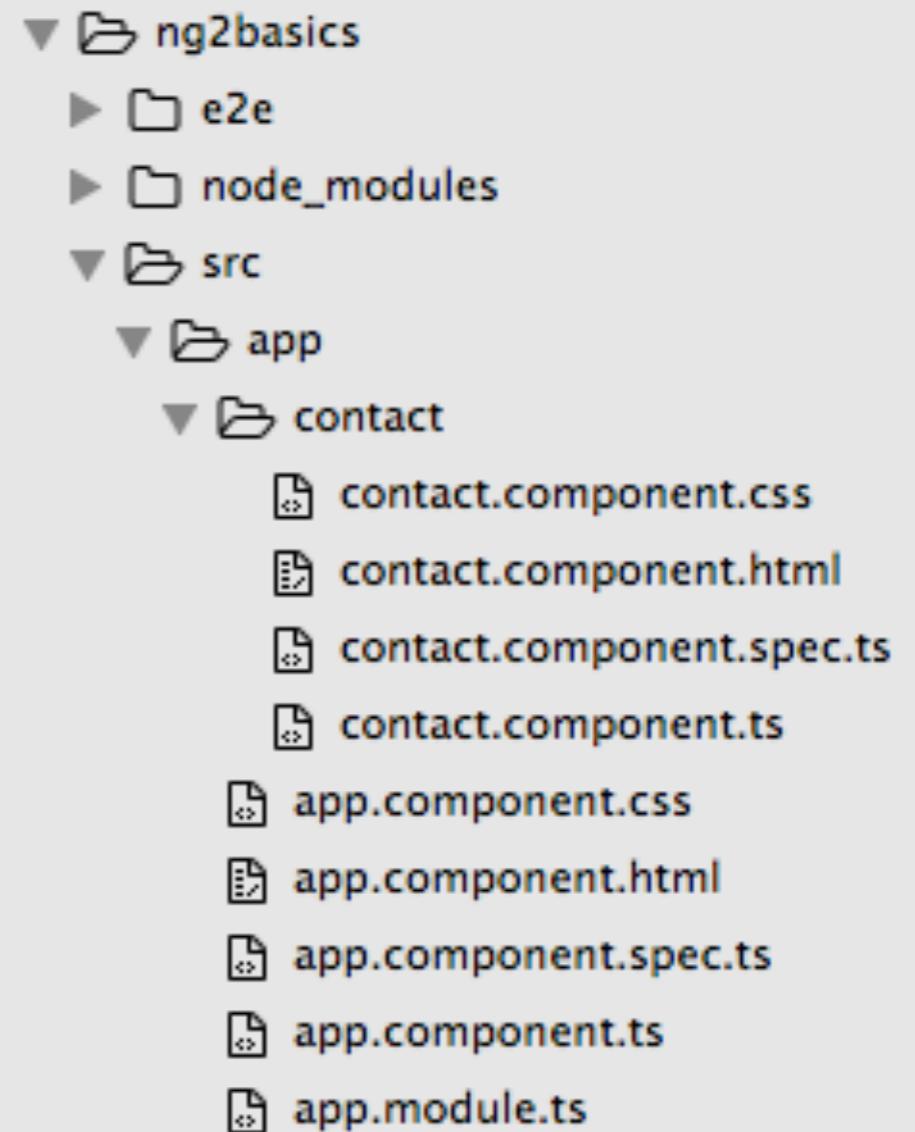
# Adding a new component:

***ng generate component <Component-Name>***

```
[Vinods-MacBook-Pro:ng2basics vinodkumar$ ng generate component Contact  
installing component
```

```
create src/app/contact/contact.component.css  
create src/app/contact/contact.component.html  
create src/app/contact/contact.component.spec.ts  
create src/app/contact/contact.component.ts  
update src/app/app.module.ts
```

```
Vinods-MacBook-Pro:ng2basics vinodkumar$
```



# More options

- Generate components:
  - `ng generate component components/counter`
  - `ng g c components/counter`
- Generate service (injectables):
  - `ng g s services/contacts`
- Generate directives:
  - `ng g d directives/box`
- Generate pipes:
  - `ng g p pipes/fullname`

# Check out my TOP rated Angular 7 course

- Udemy: <http://bit.ly/2LsqmMa>
- SkillShare: <https://skl.sh/2JfpU2I>
- Visit <https://vinod.co> for other courses like ReactJS, MongoDB etc.