

# Angular

**The Basics and more**



COS216

AVINASH SINGH

DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF PRETORIA

# ANGULAR - OVERVIEW



Open-source, released October 2016, latest version 16



Developed and Maintained by Google



Code in Typescript + HTML + CSS



Predecessor was AngularJS and was rewritten too Angular



Cross Platform



Needs NodeJS to run

# HOW TO USE ANGULAR?

- Firstly, you need NodeJS
- Install Angular

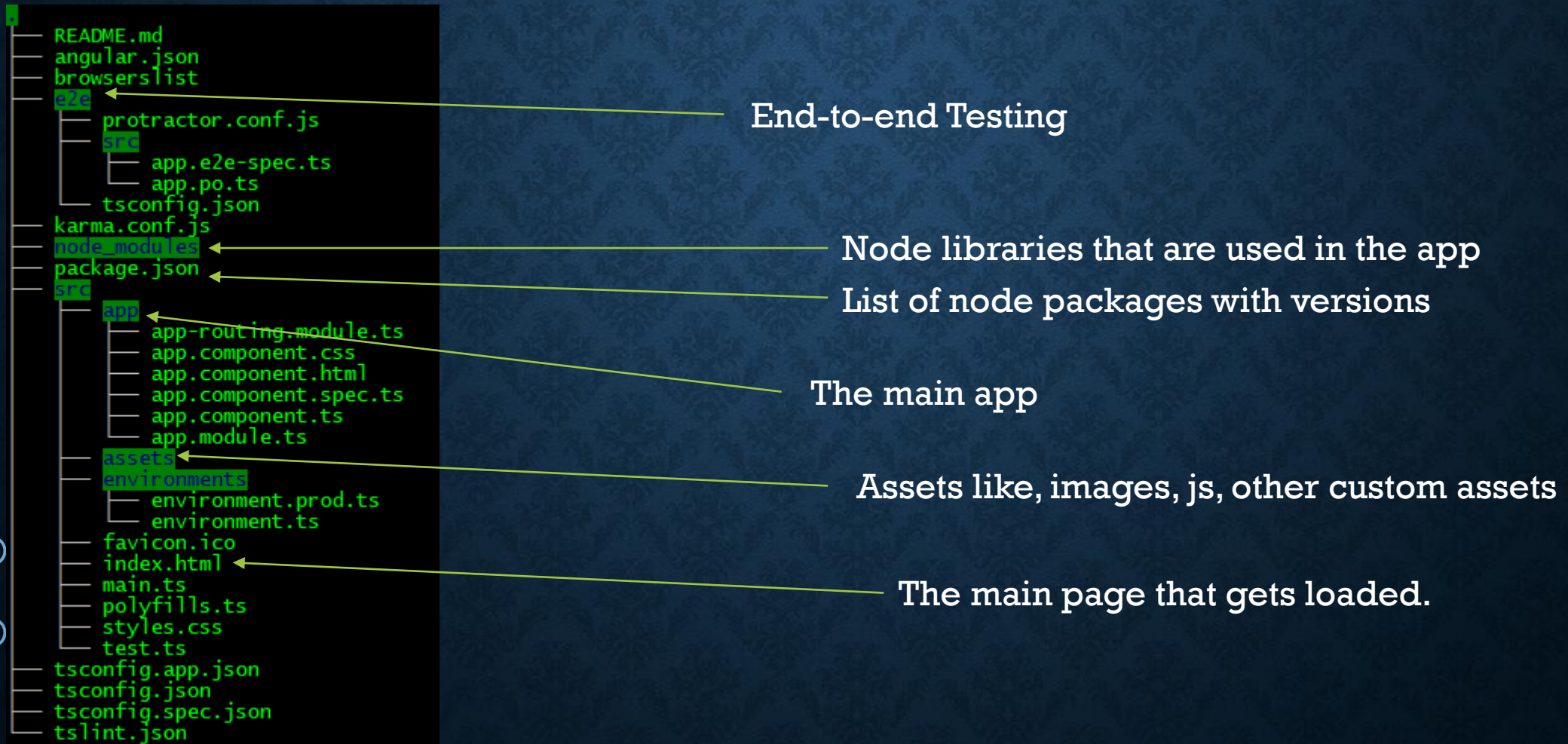
```
npm install -g @angular/cli
```
- You will now get an **ng** binary to perform any actions.
- Since angular is a web framework you need to use the CLI to start an application

```
ng new <app_name>
```
- **You will get options to follow, Choose yes for routing, when you come to stylesheets you can choose any you wish but, in most cases, SCSS is used. You can also choose traditional CSS but you have limited functionality**
- After installing

```
ng serve -o
```



# PROJECT STRUCTURE



# CONCEPTS

- Routing – describes the navigation and working of the app
- Components – the functionality you add to the app
- Tests – describes the code to test the components / routing



# BASIC STRUCTURE

```
app-routing.module.ts  
app.component.css  
app.component.html  
app.component.spec.ts  
app.component.ts  
app.module.ts
```

Describes the components and the routes

Testing file

Describes your component logic

Describes the packages or modules needed for your component

# COMPONENT

- A component has 3 parts, an Import, Decorator and a Class

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'demo';
}
```

# COMPONENT

- Generally components are like tabs or pages.
- A angular app generally consists out of many components, like home, login, trending, etc.
- To add a new component simply run:

**ng generate component <name>**



# ROUTING

- Now that you have created a new component, you must set the routing.
- To do this you have to add the routing to

**app-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from '../home/home.component';

const routes: Routes = [
  {path: 'home', component: HomeComponent},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

# INTEGRATING TYPESCRIPT WITH HTML

- So component logic, functions, variables and events can all be sent to a template HTML, this is how you can get some functionality into the HTML without much effort.
- On the left we have simple typescript function **count** that increases the counter by 1

```
export class HomeComponent implements OnInit {  
  counter: number = 0;  
  
  constructor() {  
  
  }  
  
  ngOnInit(): void {  
  
  }  
  
  count() {  
    this.counter++;  
  }  
  
}
```

# INTEGRATING TYPESCRIPT WITH HTML

- Now in the HTML we can access the global counter variable by using handlebars (aka moustache) {{ }}
- There are also JS events and processors that can be used for example (click) which is when the element is clicked on

```
<p>home works!</p>  
<button (click)="count()">Click Me</button>  
<p>You have Clicked me {{ counter }} times.</p>
```



# INTEGRATING TYPESCRIPT WITH HTML

home works!

Hi !

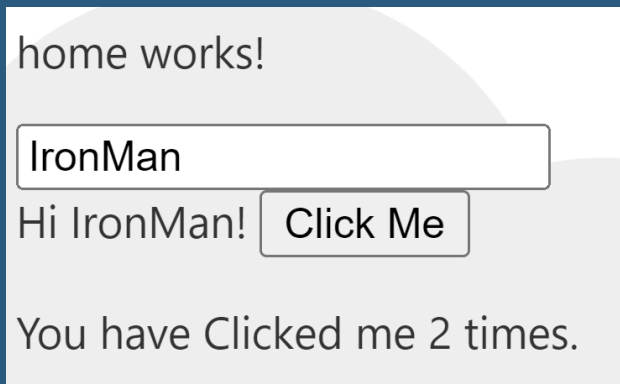
Click Me

You have Clicked me 14 times.

# MODELS

- Up until now we saw how we can get data from TypeScript into HTML templates but what if we want it to be bidirectional?

```
<p>home works!</p>
<input type="text" [(ngModel)]="name"><br>
Hi {{ name }}!
<button (click)="count()">Click Me</button>
<p>You have Clicked me {{ counter }} times.</p>
```



home works!

Hi IronMan!

You have Clicked me 2 times.

```
export class HomeComponent implements OnInit {
  counter: number = 0;
  name: string = "";

  constructor() {

  }

  ngOnInit(): void {

  }

  count() {
    this.counter++;
  }

}
```

# ANGULAR LOGIC

- You can also perform or show certain functionality based on conditions for example:

```
<p>home works!</p>
<input type="text" [(ngModel)]="name"><br>
Hi {{ name }}!
<button (click)="count()">Click Me</button>
<p>You have Clicked me {{ counter }} times.</p>

<ng-template [ngIf]="counter > 5" [ngIfElse]="none">
  You are clicking too much {{ name }}!!!
</ng-template>
<ng-template #none>
  You not clicking enough
</ng-template>
```

home works!

Hi IronMan!

Click Me

You have Clicked me 6 times.

You are clicking too much IronMan!!!



# ANGULAR LOGIC

- You can also perform CSS based on logic:

```
<ng-template [ngIf]="counter > 5" [ngIfElse]="none">
  <p [ngStyle]="{'color': counter > 10 ? 'red': 'green'}">
    You are clicking too much {{ name }}!!!
  </p>
</ng-template>
<ng-template #none>
  You not clicking enough
</ng-template>
```

# ANGULAR LOGIC

- You can also perform CSS based on logic using classes:

```
<ng-template [ngIf]="counter > 5" [ngIfElse]="none">
  <p [class.active]="counter > 10">
    You are clicking too much {{ name }}!!!
  </p>
</ng-template>
<ng-template #none>
  You not clicking enough
</ng-template>
```

```
.active {
  background-color: yellow;
}
```

home works!

IronMan

Hi IronMan! Click Me

You have Clicked me 7 times.

You are clicking too much IronMan!!!

# ANGULAR LOGIC

- But what if I want to use many classes based on the counter?

```
<ng-template [ngIf]="counter > 5" [ngIfElse]="none">
  <p [ngClass]="setClasses()">
    You are clicking too much {{ name }}!!!
  </p>
</ng-template>
```

```
setClasses() {
  let classes = {
    active: this.counter > 6,
    notactive: this.counter < 6,
    bold: this.counter > 10
  }
  return classes;
}
```

```
.active {
  background-color: yellow;
}

.notactive {
  background-color: gray;
}

.bold {
  font-weight: bold;
}
```

home works!

Hi IronMan!

You have Clicked me 31 times.

**You are clicking too much IronMan!!!**



# ANGULAR LOGIC

- What if I have an array that I want to display like music tracks?

```
constructor() {  
  this.music = [  
    { title: 'Some title1',  
      album: 'some album1', rank: 500 },  
    { title: 'Some title2',  
      album: 'some album2', rank: 502 },  
    { title: 'Some title3',  
      album: 'some album3', rank: 30 },  
    { title: 'Some title4',  
      album: 'some album4', rank: 220 },  
  ];  
}
```

Title: Some title1

Album: some album1

Rank: 500

Title: Some title2

Album: some album2

Rank: 502

Title: Some title3

Album: some album3

Rank: 30

Title: Some title4

Album: some album4

Rank: 220

```
<div *ngFor="let m of music">  
  <p>Title: {{ m.title }}</p>  
  <p>Album: {{ m.album }}</p>  
  <p>Rank: {{ m.rank }}</p>  
</div>
```

# ANGULAR - INFO

- There is a lot more about Angular that was not taught, but this serves as the foundations that is needed such that we can tackle Hybrid Mobile.
- More information about angular can be found here:
- <https://angular.io/tutorial>

