Angular Fundamentals

# Getting Started With Angular

## Introduction

Prerequisites

Basic JavaScript

App.pluralsight.com/paths/skills/javascript

Basic HTML

App.pluralsight.com/paths/skills/html5

Helpful Prequisities

Not Required

Basic Node and NPM

App.pluralsight.com/path/skills/npm-playbook

ES2015

App.pluralsight.com/courses/javascript-fundamentals-es6

TypeScript

App.pluralsight.com/courses/typescript

## Practice Exercises

## Introduction to TypeScript

TypeScript – Syntactic Sugar to the JavaScript.

You can write full javascript in TypeScript.

Writing TypeScript : javascript with additional features like

Transpilation will convert TypeScript to Plain old javascript.

Development Environment construct

Compiler will do transpilation. If no errors, transpilation done.

If errors, transpilation done.

BuildTime Checking to prevent Run time errors

TypeScript features:

Static Typing

DataTypes to variable declaration in plain old javascript declaration

Const name;

Let age;

Const name: string;

Let age: number;

Assigning different datatype will make typescript compiler

TypeScript Interface:

Interface for an object

TypeScript Class Properties:

Class Cat {

Constructor

### Class members are public by default in both ES2015 and TypeScript

Class Cat {

Private name: string

Private color: string

Constructor(name, color) {

this.name = name;

this.color = color;

}

}

### TypeScript shortcut for initializing member variables through constructor arguments

Class Cat {

Constructor(private name, private color) {

}

}

Let fluffy = new Cat(‘Fluffy’, ‘White’);

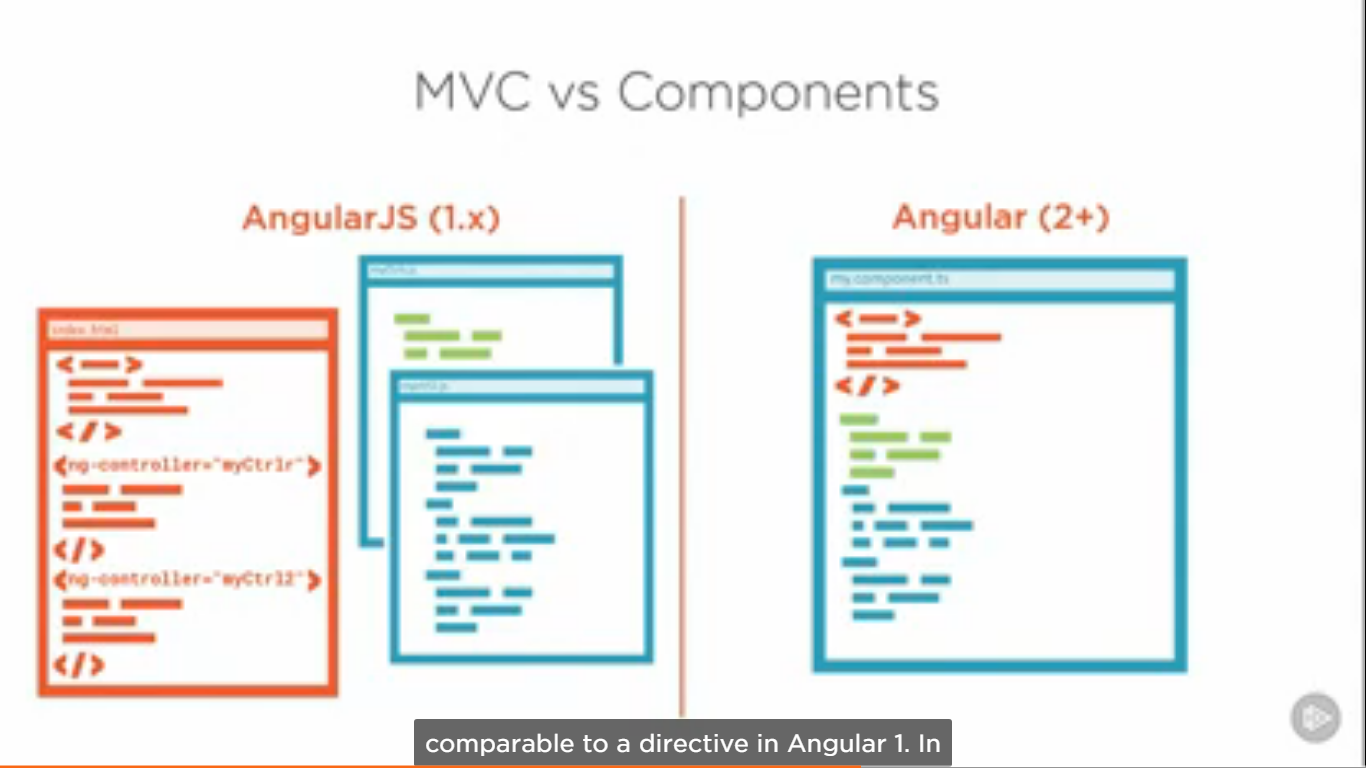
### Private properties that are initialized via constructor

## AngularJS vs Angular

Angular JS – MVC framework

Angular JS – template has more control. Identify one ore more sections

Angular – Component and templates will have one to one relationship. Template is actually part of component.



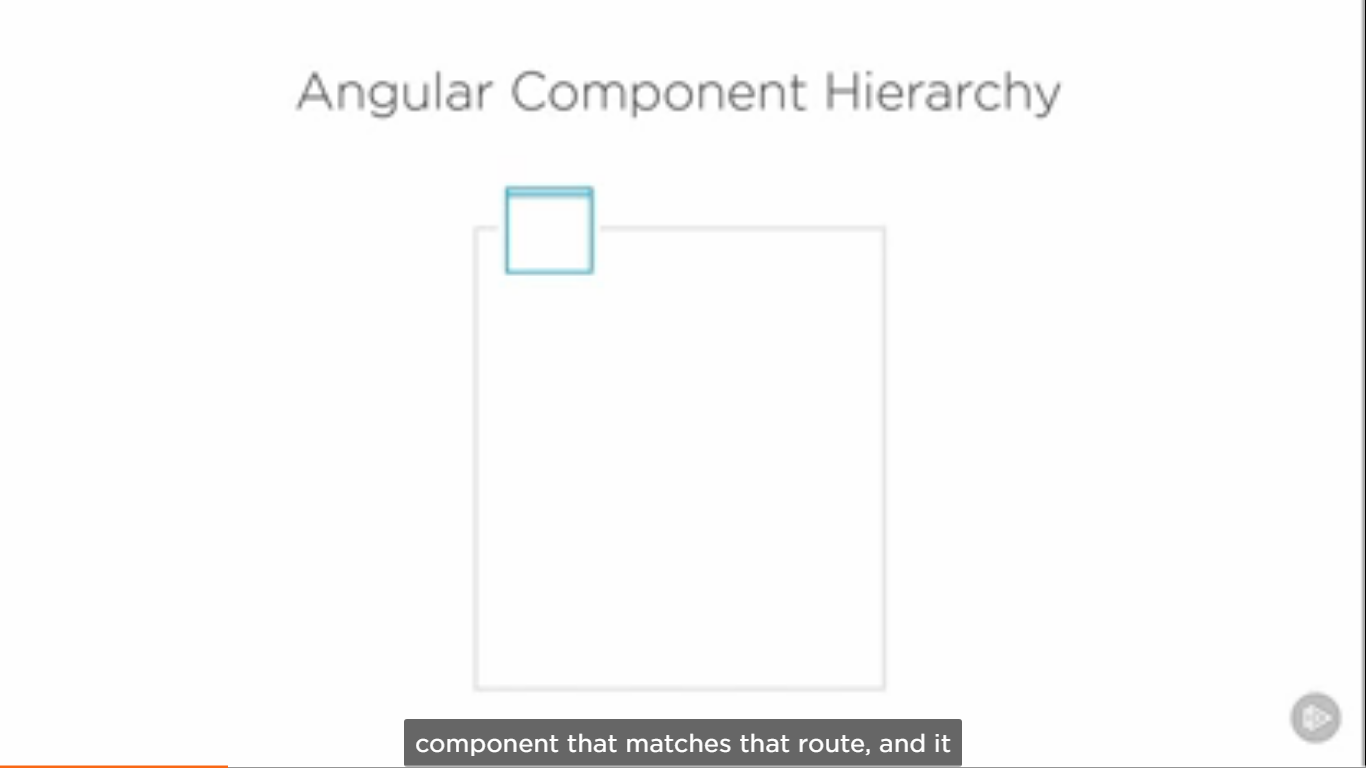
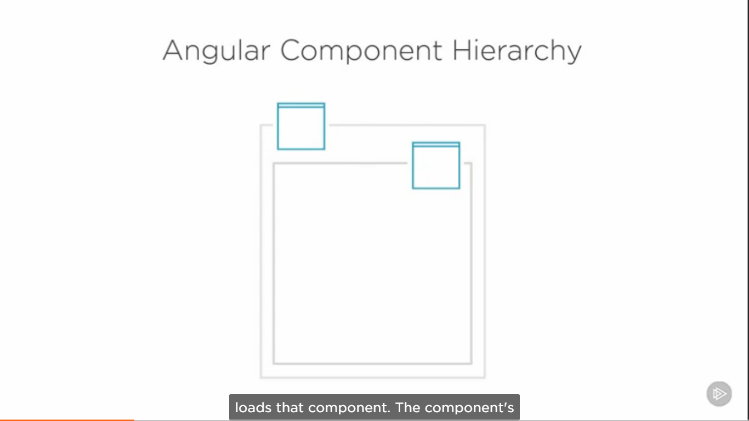
A component is angular 1 is actually directive in angular JS

## A Conceptual Overview of Angular

High Level At Angular

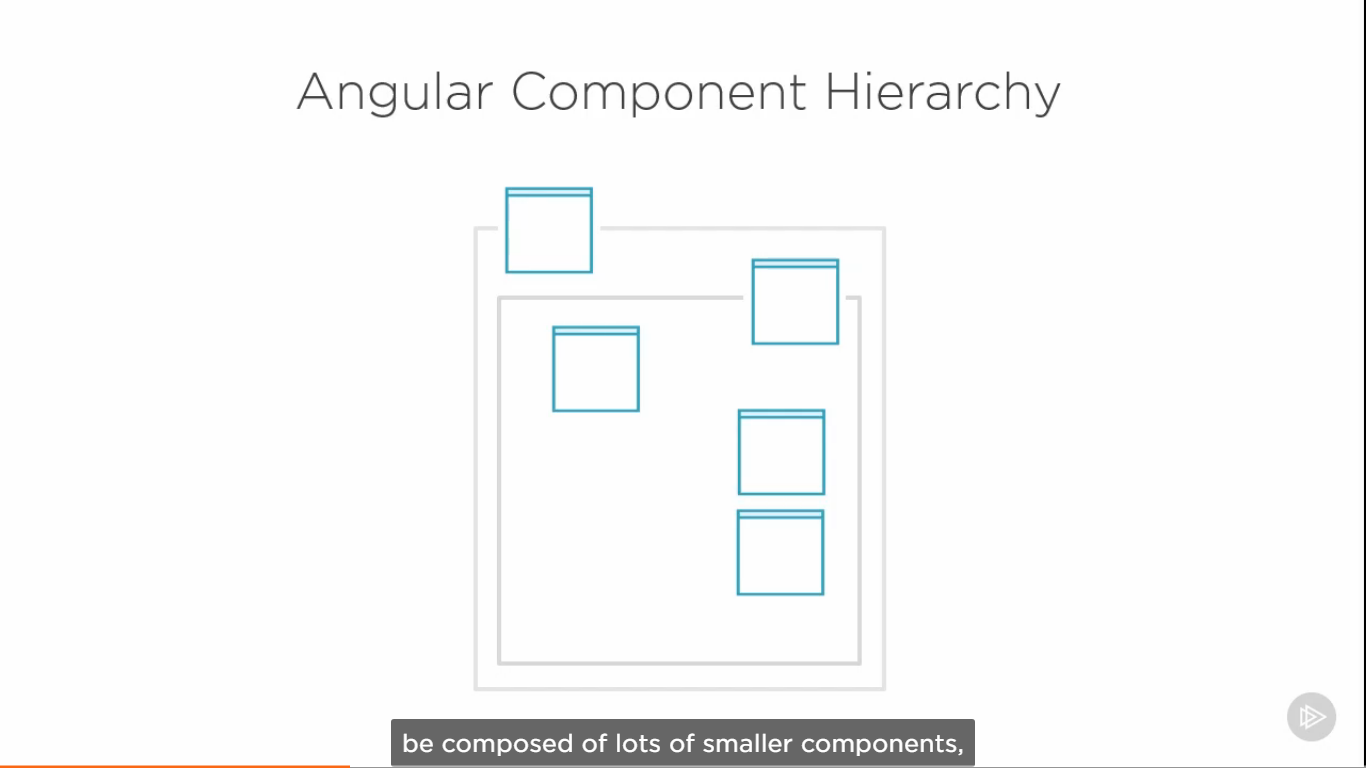
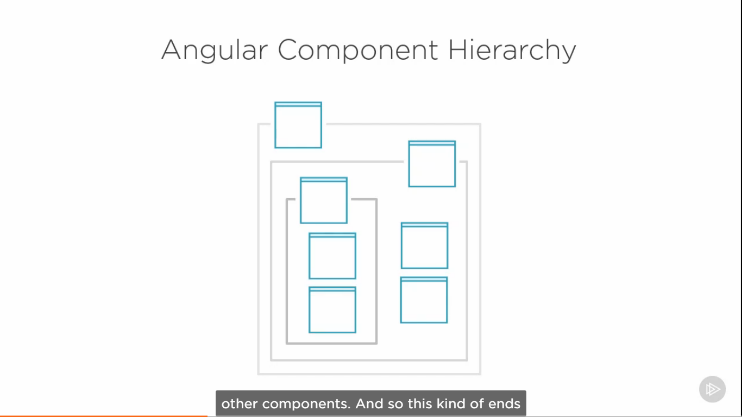
Once we have concept model how angular works, it will be easier to understand each of the pieces as we start coding them. So, let’s take a graphical at an angular application.

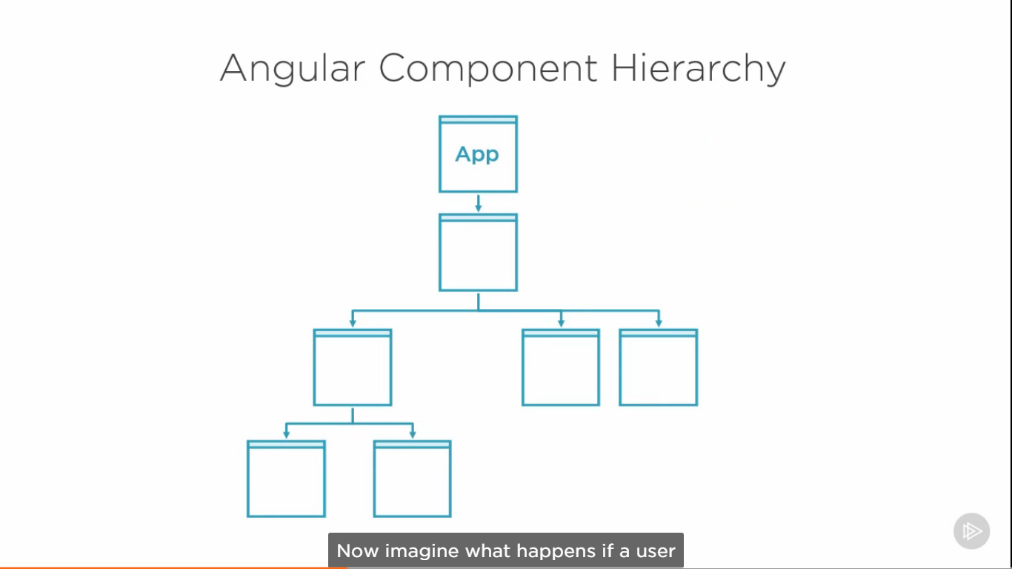
When you navigate to the URL of an angular application, there is always an root application component that is loaded first, then the angular router look at the URL and recognizes there is a component that matches the route.



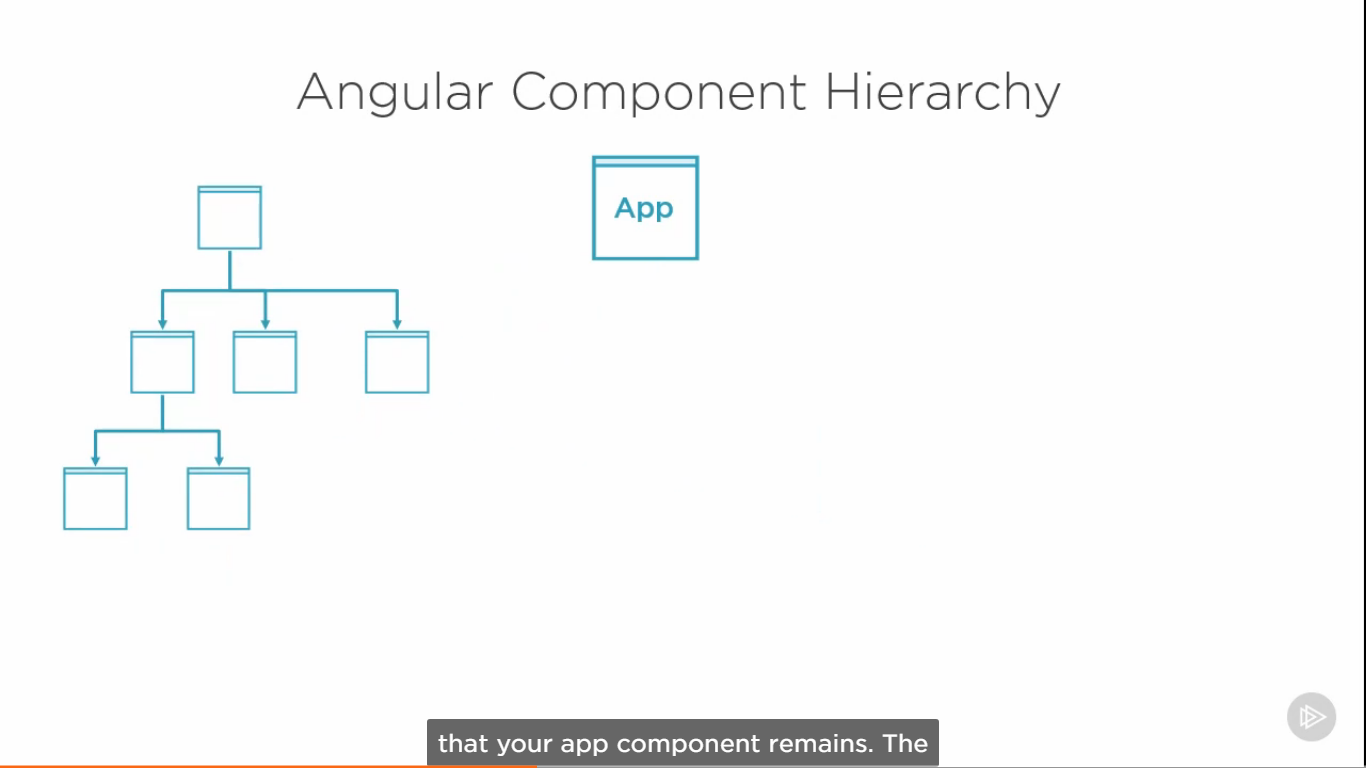
And loads the component. The component’s template is then displayed in the browser and the component may then load some data from the server and give it to the template to be displayed.

Of course if this page is very complex, it’s likely to be composed of lots of smaller components

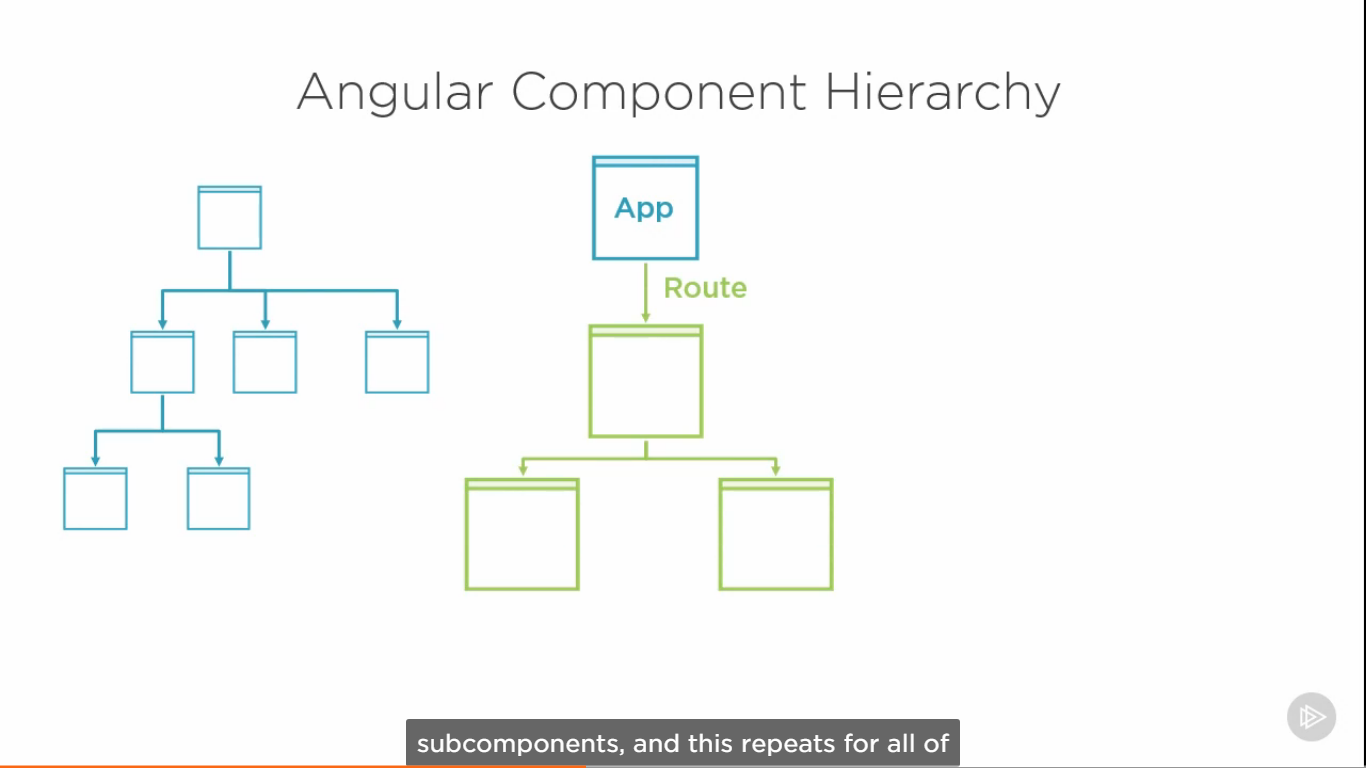


 And those components may be composed of other components. This kind of level of hierarchy ends creating a tree-like structure. It helpful to sometimes to think about your angular app in this way. So, here tree of our imaginary application.

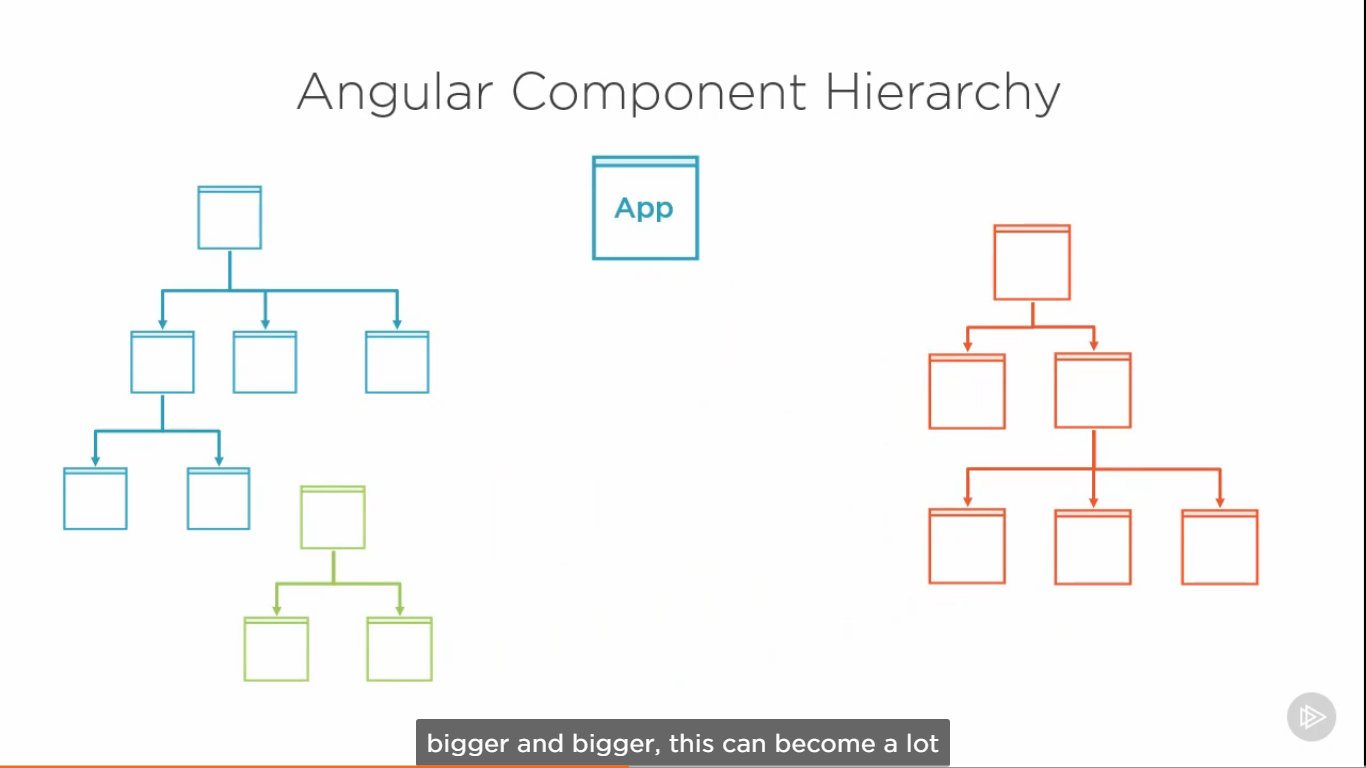
Now, imagine what happens when a user navigates to another URL. Well, then basically you start a new tree, except app component remains.



Router sees the new route and loads the corresonding component

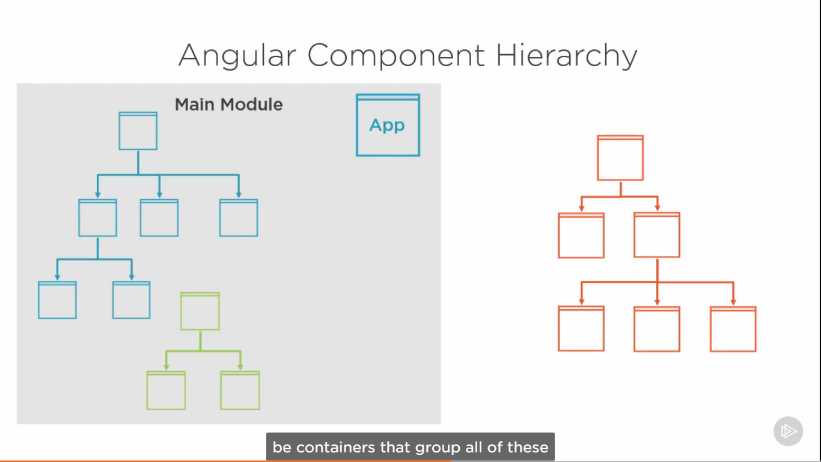


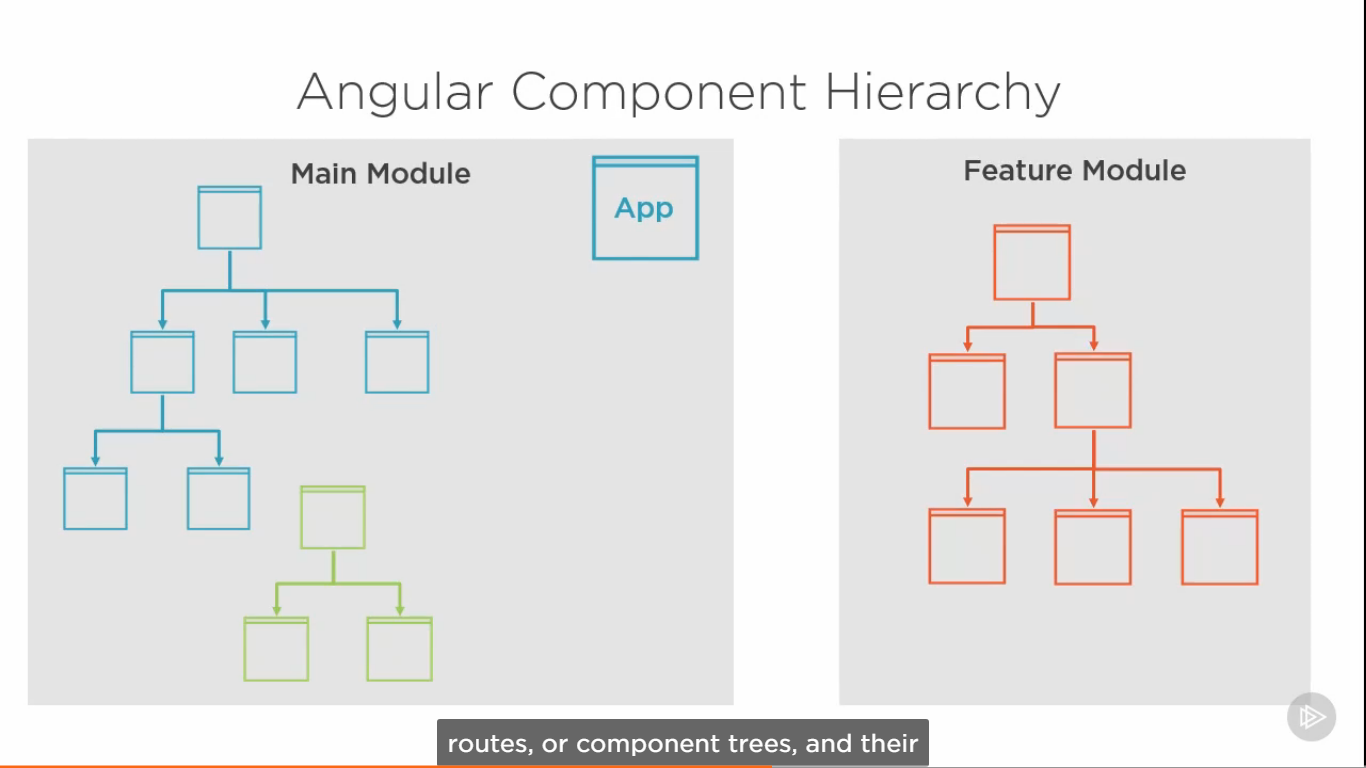
This repeats for all of the routes. As application gets bigger and bigger, this can become a lot of stuff to load into memory.



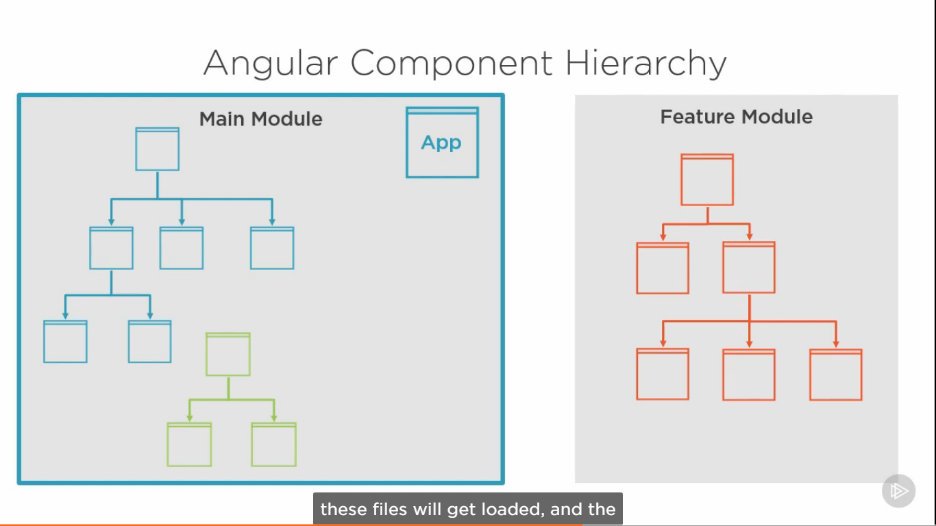
There is where Angular modules come in. The Angular Modules, or NgModule should not be confused with ES2015 modules.

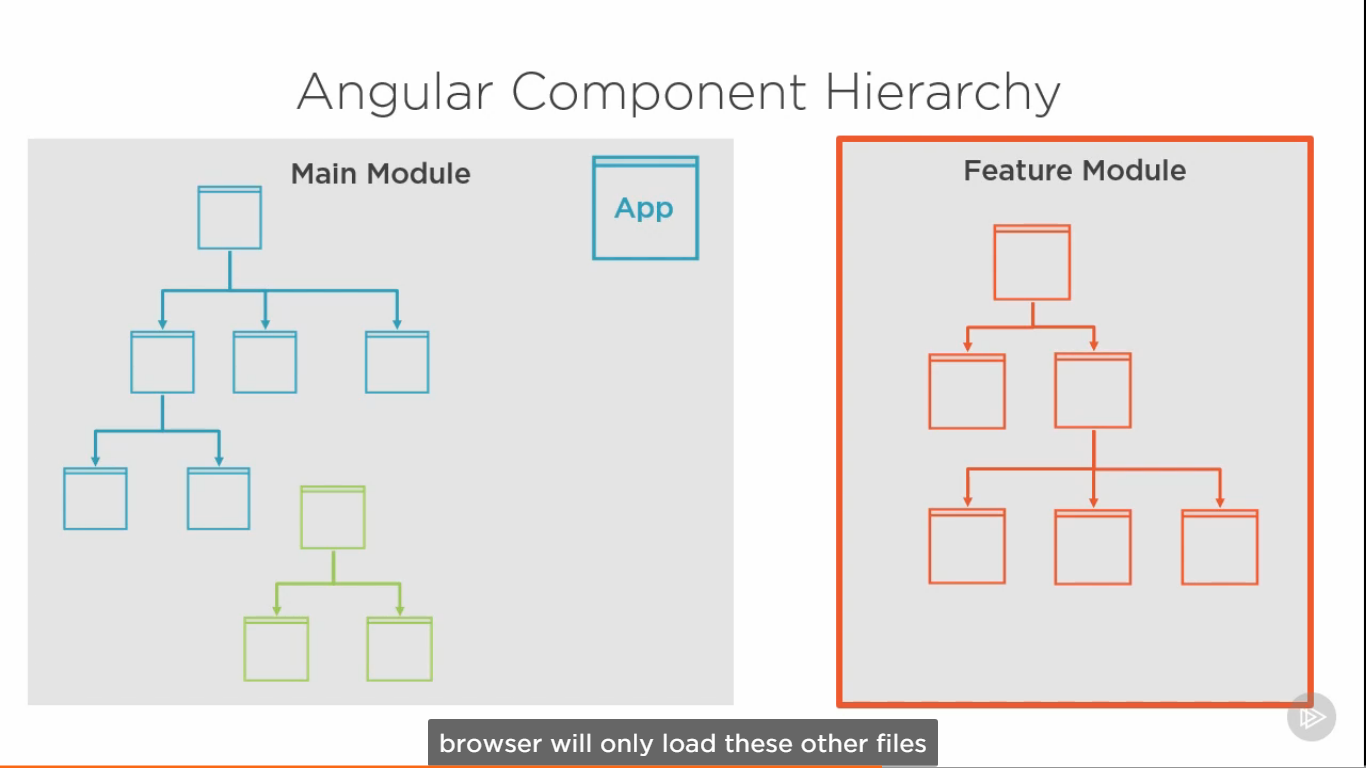
### What are angular modules

Angular modules are meant to be containers that group all of these routes, component trees and their corresponding bits of code into modules that can be loaded independently of each other.



That way, if a users ends up visiting this section of the site only these files will be loaded

and the browser will load only these others files. If the user navigates over into this section of the site. This is one of the main purposes behind Angular modules.



### You don’t have to use multiple modules in your app, but as your app gets larger you may have to consider it

One more high-level concept regarding modules that is important to mention is how modules are used to make components, services, and directives available to rest of your application.

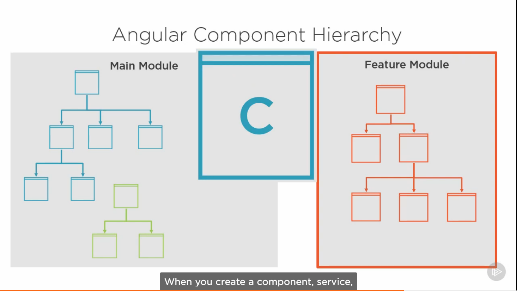
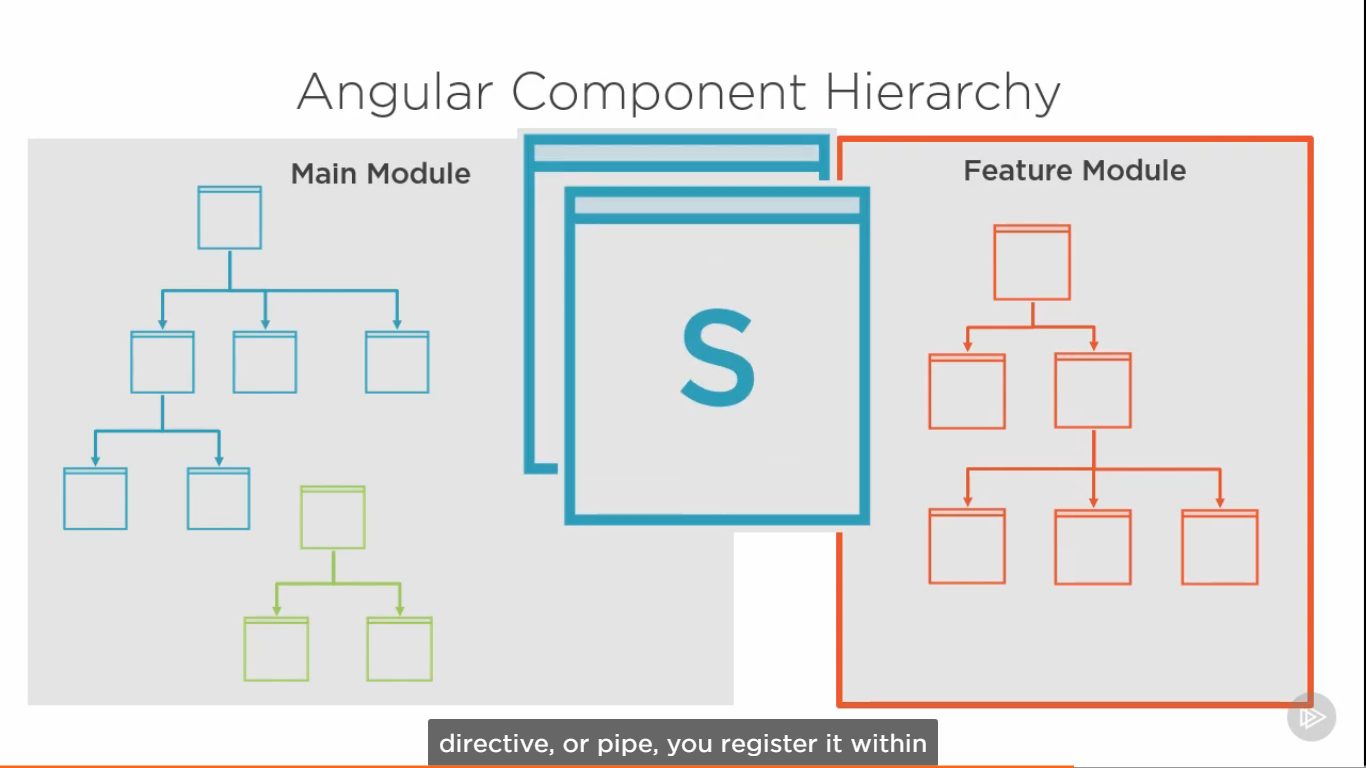
When you create a

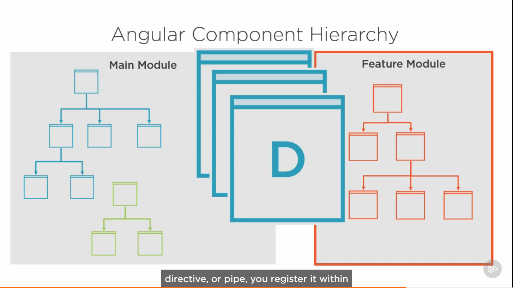
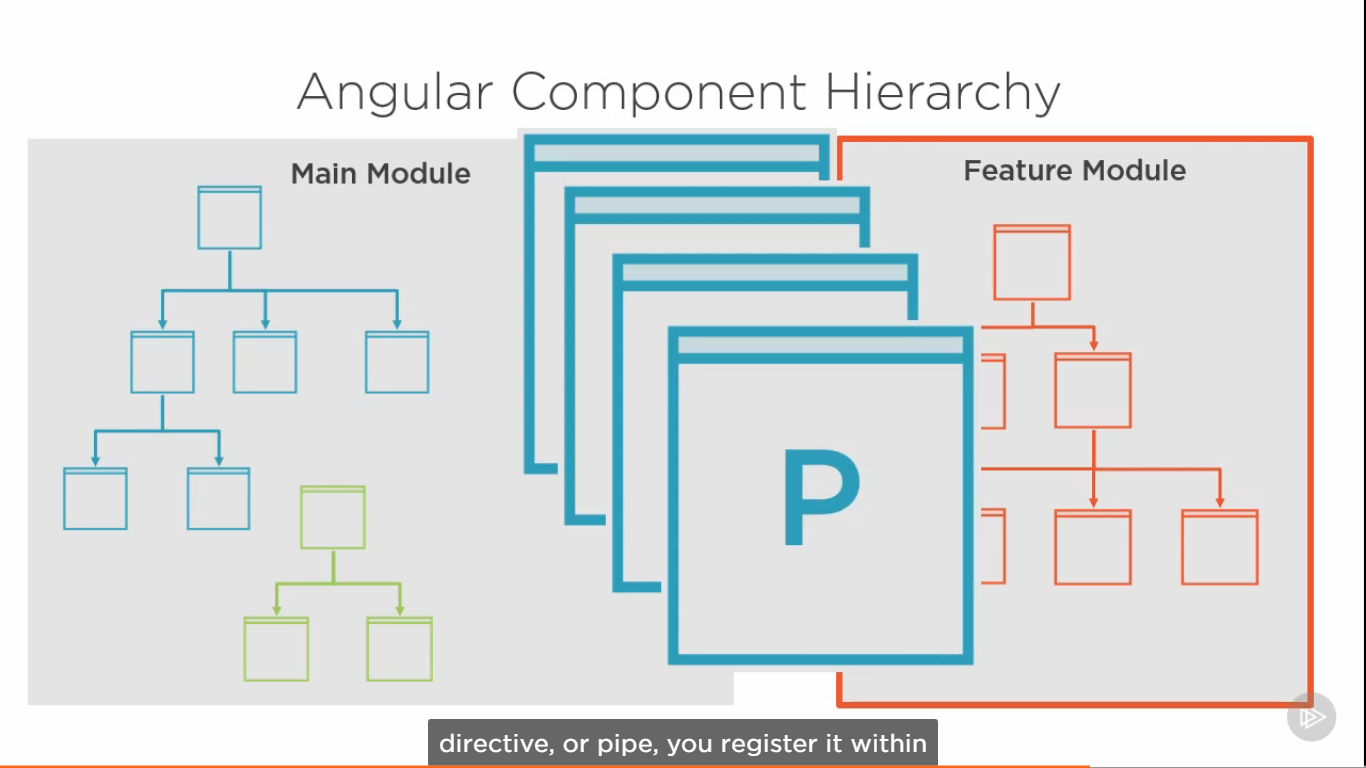
q) component

b) service

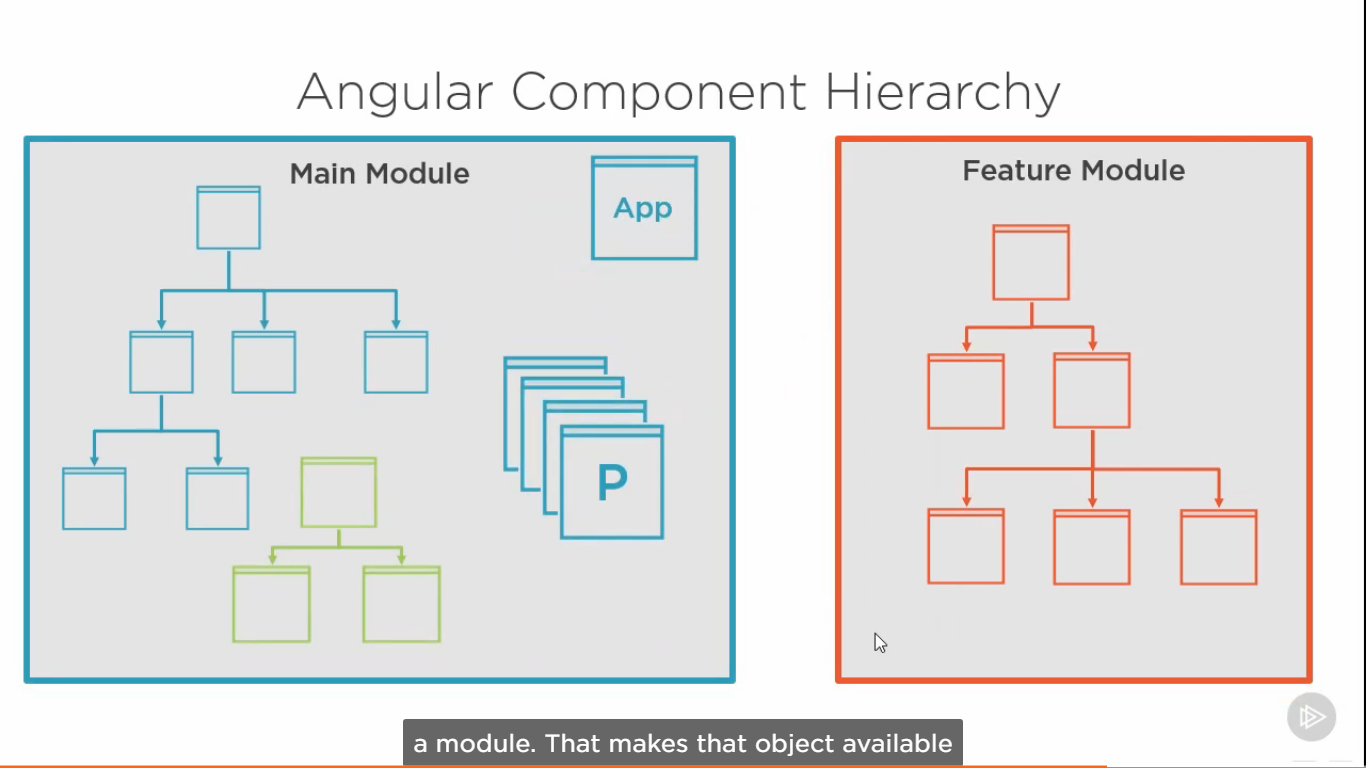
c) directive

d) pipe,



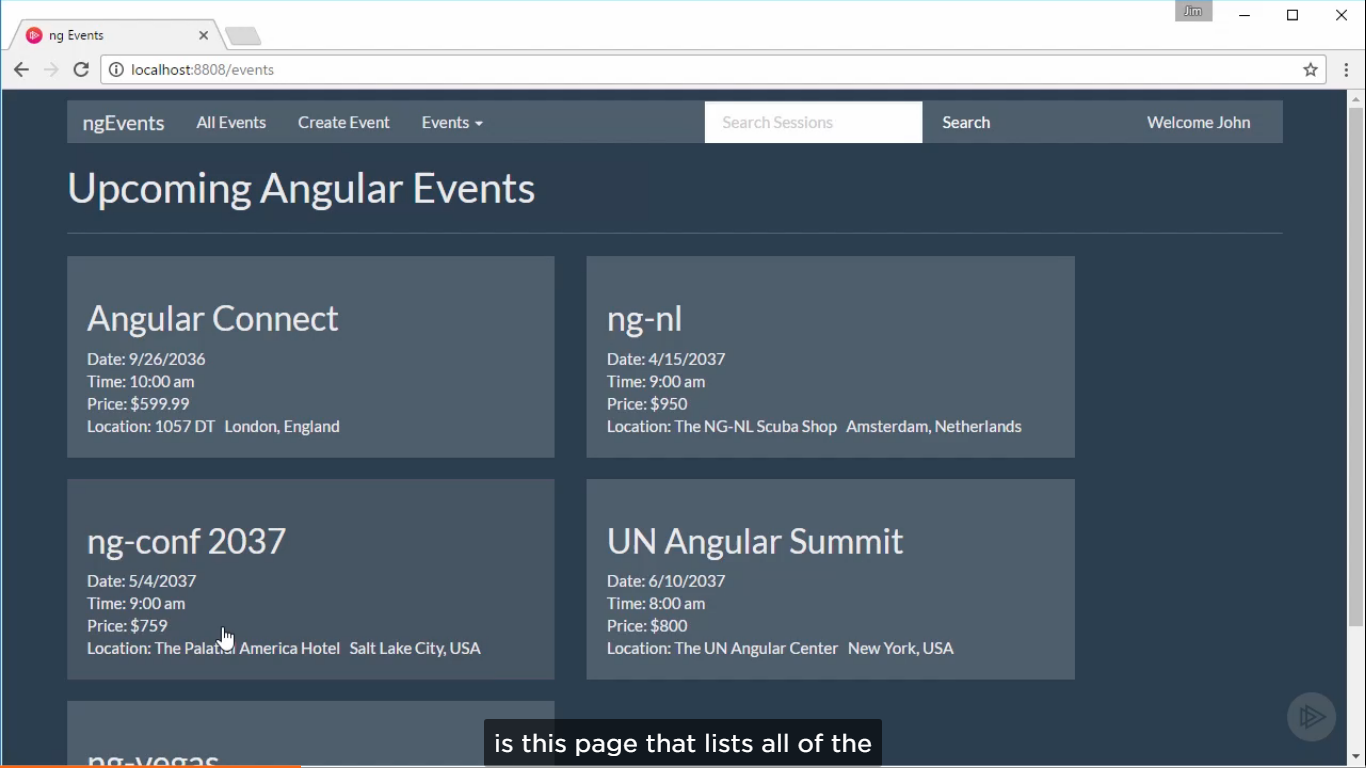
You register it within a module. That makes the object available for use by everything else that in the module. In All of these, services, they will avaialble only in the module which are register it. If you



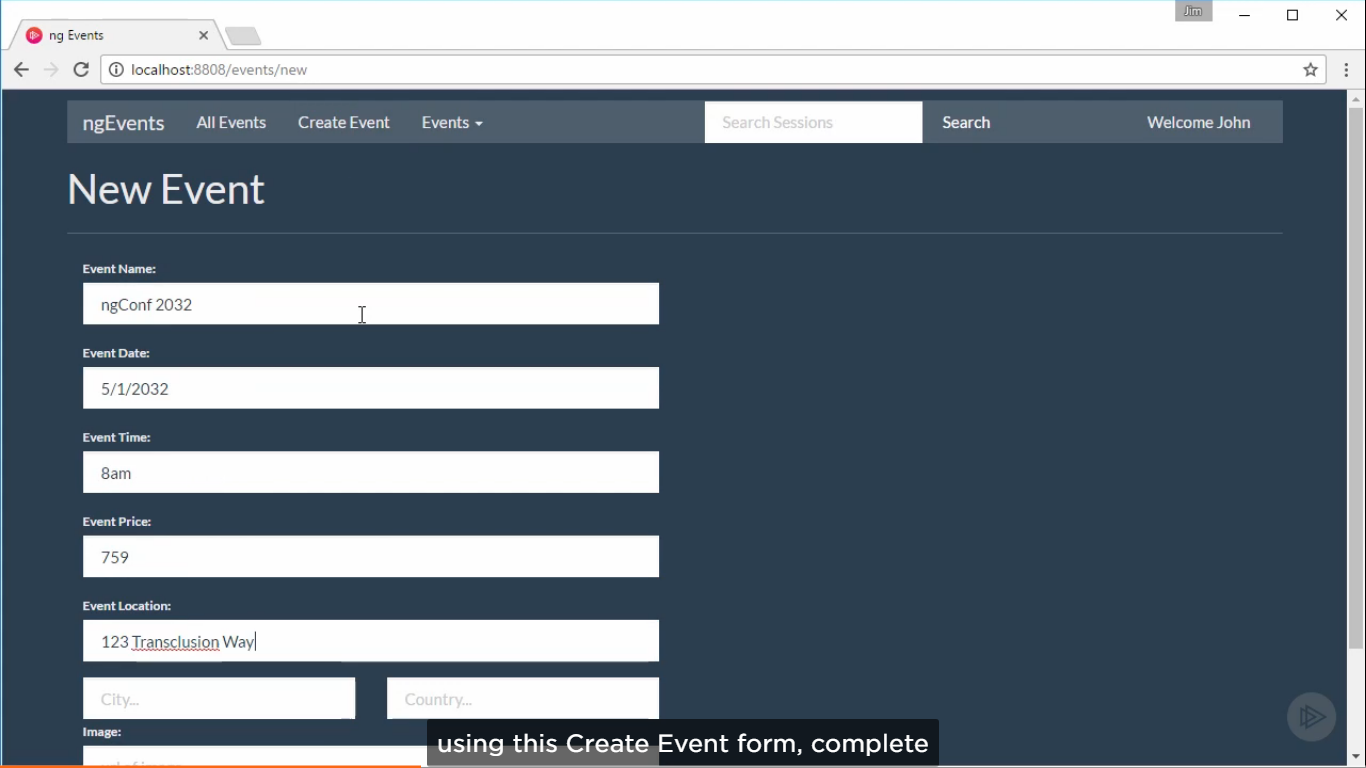
Want to use them in another module, you must register with that module also. That’s true all of these, except for services. Services or providers get registered in the root injector, so they are available across all the angular modules

## What we will be buiding

Build an application that displays and allows users to create technology events or conferences. The Starting Page is below. It will list all the upcoming events.



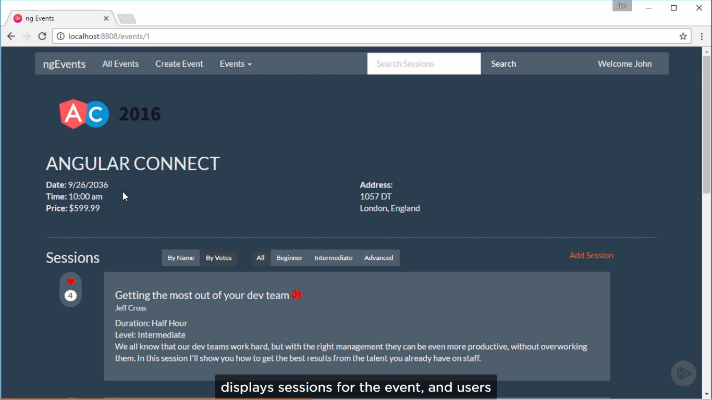
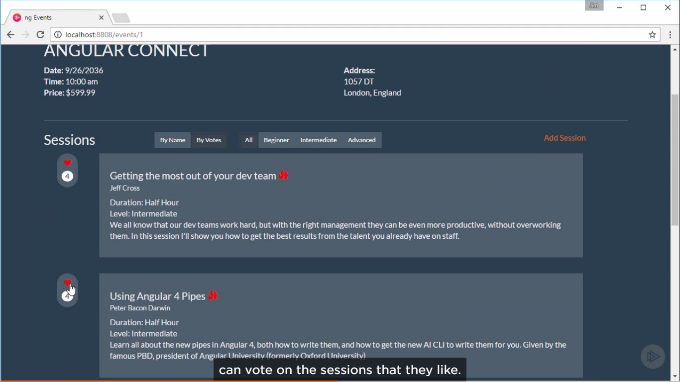
From here, we ll add the ability for the users to create new events using create Event Form complete with validation



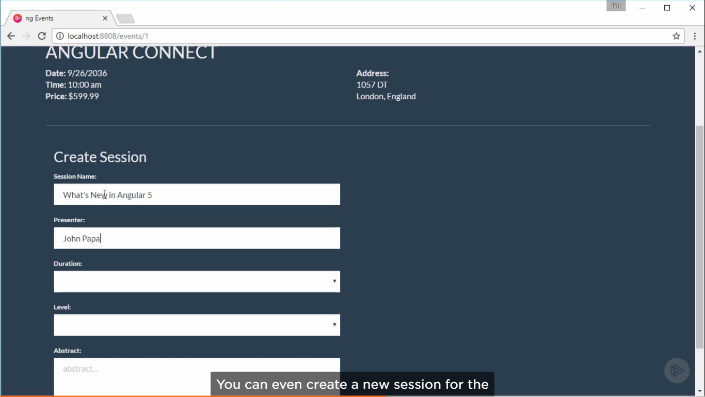
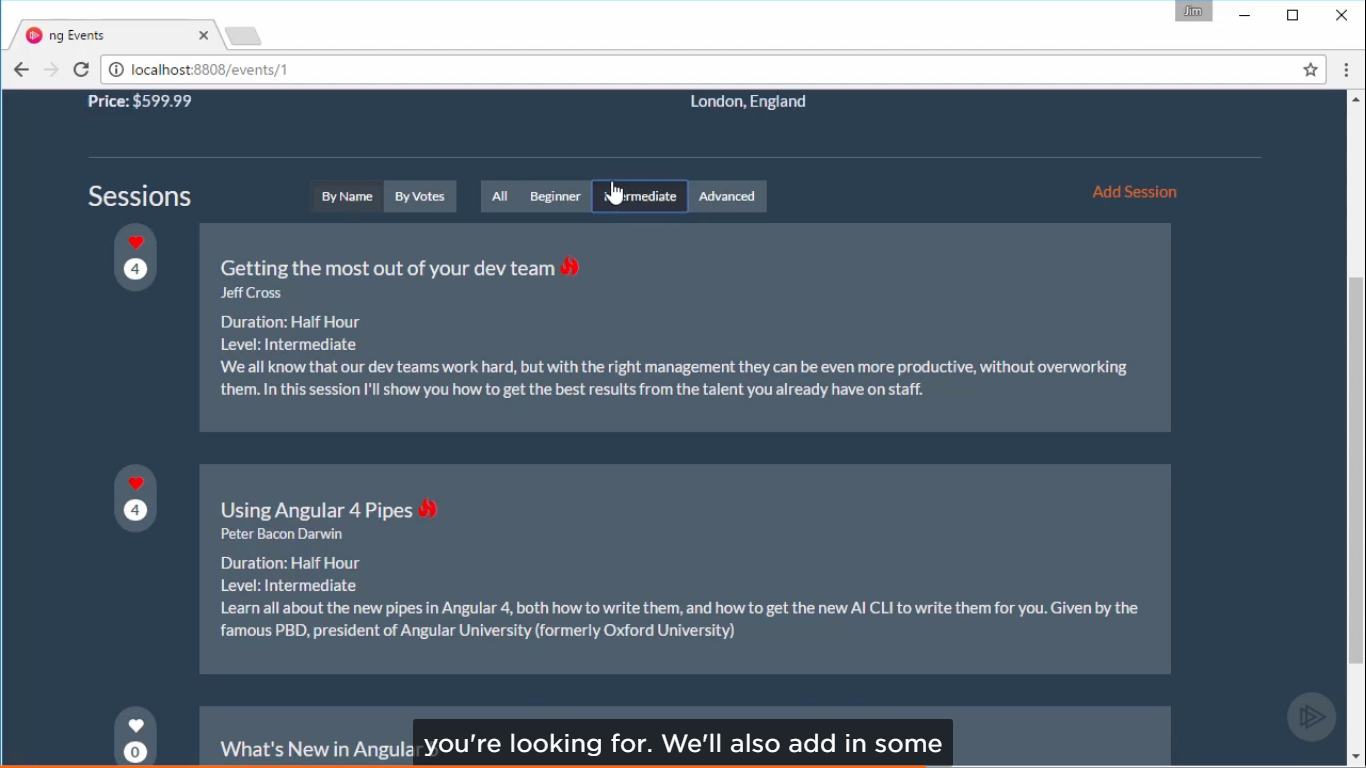
Then we ll wire these things up so that users can click on one of these events to see more detailed information about that event.



Events Details Page

This page displays sessions for that event

And user can can vote on the sessions that they like. Even user can create a new session for the event. Even, add the ability of sort and filter the sessions to find what you are looking for.



We ll also add in basic authentication and edit profile features.

## Installing Git and Node

Git bash

Nvm install 8.11.2

## Getting Started with the Angular Application

Best way to get start with new angular project using angular CLI

### What is angular CLI

The Angular CLI is a command Line interface that we can use to create a new project complete with webpack config and tools for packaging up your app for production, plus a host of other features. It is highly recommended you always start your projects with CLI. So to get started, let’s install the angular CLI.

Npm install –g @angular/cli@7.1.2

Creating and Communication Between Angular Components.

1. Parent Component
2. Child Component
3. Input Decorators
4. Output Decorators
5. Template Reference Variables
6. Styling Components.

Exploring Angular CSS Encapsulation

Angular is taking care of encapsulating CSS Styles and cause them to affect only on components that they are added. Also it doesn’t apply on the child component. This is angular’s built view encapsulation. Angular is applying attributes to all the components.

You can also mix global styles and component specific styles.

::deep selector

## Exploring Angular Template Syntax

Interpolation

Propoerty Binding

Expressions.

Expressions recommendations

No Side-Effects.

Fast

Simple Idempotent = it should always give the same result

### Event Bindings and Statements

Dom event bindings to the functions/statements.

Guidelines for the template statements.

* Assignment Except =
* new Keyword
* Global Namespace should be not be used

### Repeating Data with ngFor

### Handline Null Values with the Safe-Navigation Operator ?

### Hiding and Showing Content with ngIf

### Hiding Content with the [Hidden] Binding

### Hiding and Showing Content with ngSwitch

### Styling Components with ngClass

* Class binding [class.<classname>] will apply single class
* ngClass -> expressions and statement, function

### Styling Components with ngStyle

* Style binding [style:<style>] will apply single style
* ngStyle -> expressions and statement, function

## Creating Reusable Angular Services

### Why we need dependency injection?

### Creating your first service

### Wrapping Third Party Services

Adding only in angular.json, third party library can be used. But the code can’t be tested. Since, library not added as dependency through DI