ANGULAR 6

Gokul Simson

- What is angular ?
- How to Setup - in detail
- Basic level of angular learning
  - Data Binding & Pipes
  - Templates, Interpolation, and Directives
  - Angular Forms
- Advance level of angular
  - Services and Dependency Injection
  - HTTP Request handing
  - Creating modules
  - Navigation and Routing
- How to do the Unit testing ?
- How to deploy in production ?

What to know b4

- HTML
- CSS
- Basic JavaScript
- Programming Fundamentals (Functions, Conditions, loops, etc)

What is Angular

- Angular is a TypeScript-based open-source front-end web application platform
- Frontend/ Client Side framework
- Create & maintained by Google
- Used to build powerful RIA / SPAs single page app
- Angular empowers developers to build applications that live on the web, mobile, or the desktop
  https://angular.io/

History

- History
  - AngularJS /Angular 1 : JavaScript
  - Angular 2 - Complete rewrite of AngularJS with Typescript [2010]
  - Angular 3 : Skipped
  - Angular 4
  - Angular 5
  - Angular 6 - Now Available
  - https://www.c-sharpcorner.com/article/difference-among-angularjs-angular-2-angular-4-and-angular-5/

Why Angular ?

- Fast, Simplified Design, Simplified Design
- One framework for Mobile & desktop.
- Rapid Development and Code Generation
- Code Organization & Productivity
- Dynamic Content
- Cross Platform, Unit Testing
- Angular offers two ways to compile your application:
  - Just-in-Time (JIT), which compiles your app in the browser at runtime
  - Ahead-of-Time (AOT), which compiles your app at build time.
- ng build | ng serve | ng build —aot | ng serve --aot

- Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side
- NPM - NPM is a package manager for Node.js packages
- NODE PACKAGE MANAGER

TypeScript

- Super set of JavaScript with added features
- Create by Microsoft
- Class based object-oriented programming
- Strongly typed (:string, :number, :Class) can be Weakly (:any)
- Additional functionalities Modules, classes, interfaces, “arrow =>” syntax, namespaces, enumerated types

https://www.typescriptlang.org/

Pre-Requires

Set-UP

- Install Node JS with NPM

[https://nodejs.org/en]

> node -v

- Install Angular CLI

[npm install —g @angular/cli]

> ng —v

git clone https://github.com/angular/quickstart.git quickstart

Create A Project

- open terminal (command prompt)
- ng new <app-name>
- cd <app-name>
- ng serve —open / npm start
- Package.json — contains all dependencies of the project
- Angular.cli.json — contains angular cli command configs.

FEW ANGULAR CLI COMMANDS

- ng new <project-name> [options] — Creates a new Angular project
- ng generate [options]
  - component [c], directive [d], route , pipe [p], service [s], class [cl], enum [e], module [m]
  - Ex : ng generate component my-new-component

    ng g c my-new-component

    it will generate 4 files css,html, ts and spec.ts
- ng version — Find the angular CLI version
- ng serve [options] — Run the application
- ng build [options]
  - --base-href, --target, --environment

https://cli.angular.io/reference.pdf

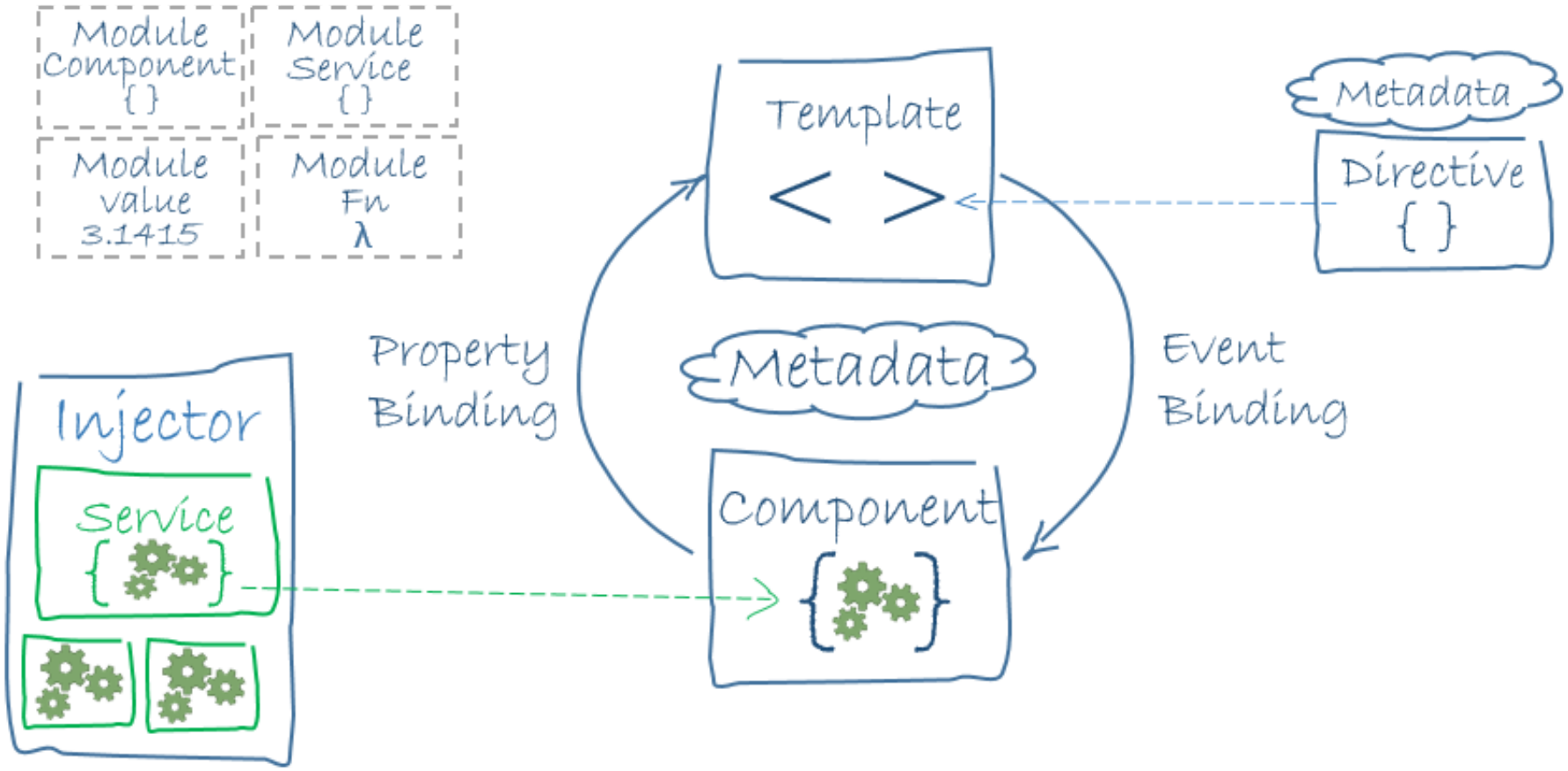
Core Features & Common Terms

- Modules
- Components
- Data Binding
- Directives
- Services
- Routing

Online editor for Practice

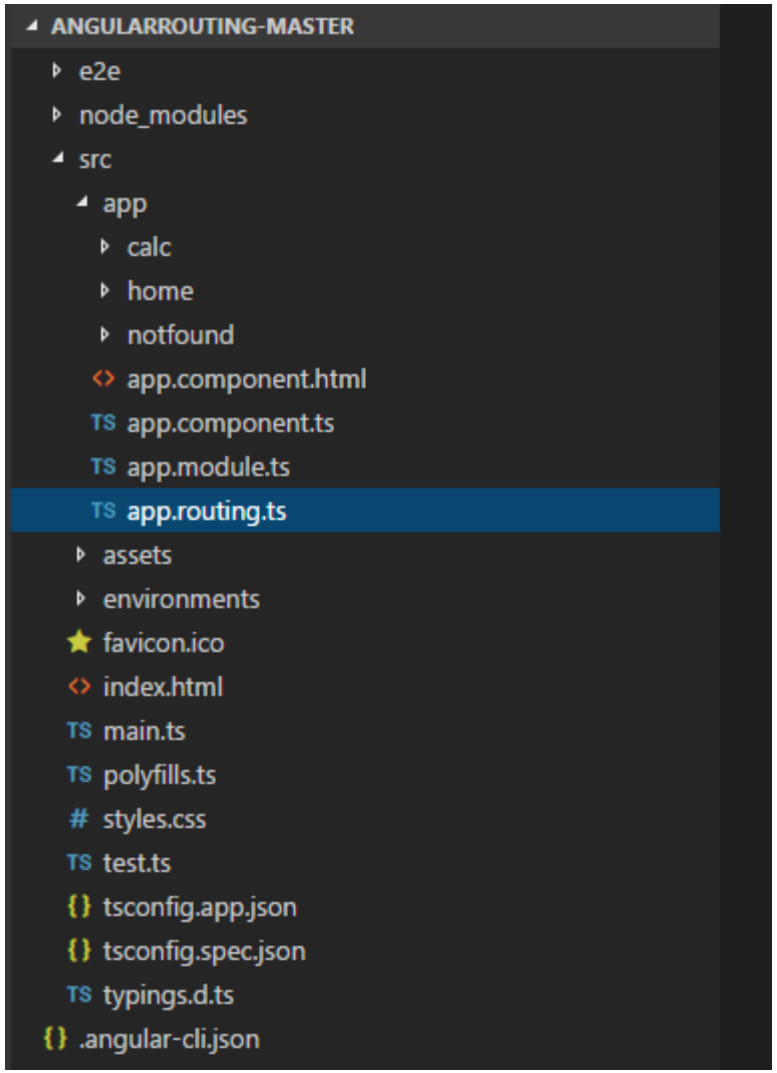
https://stackblitz.com

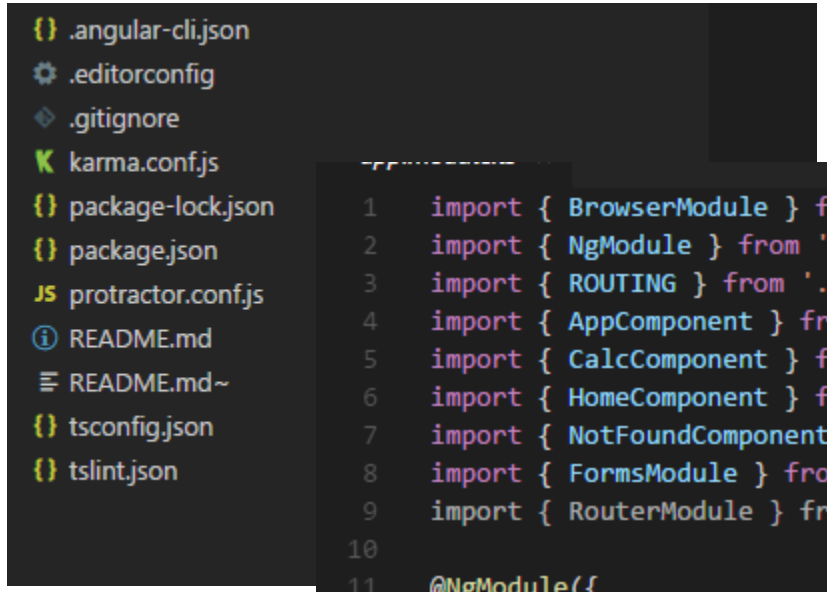
BIG PICTURE

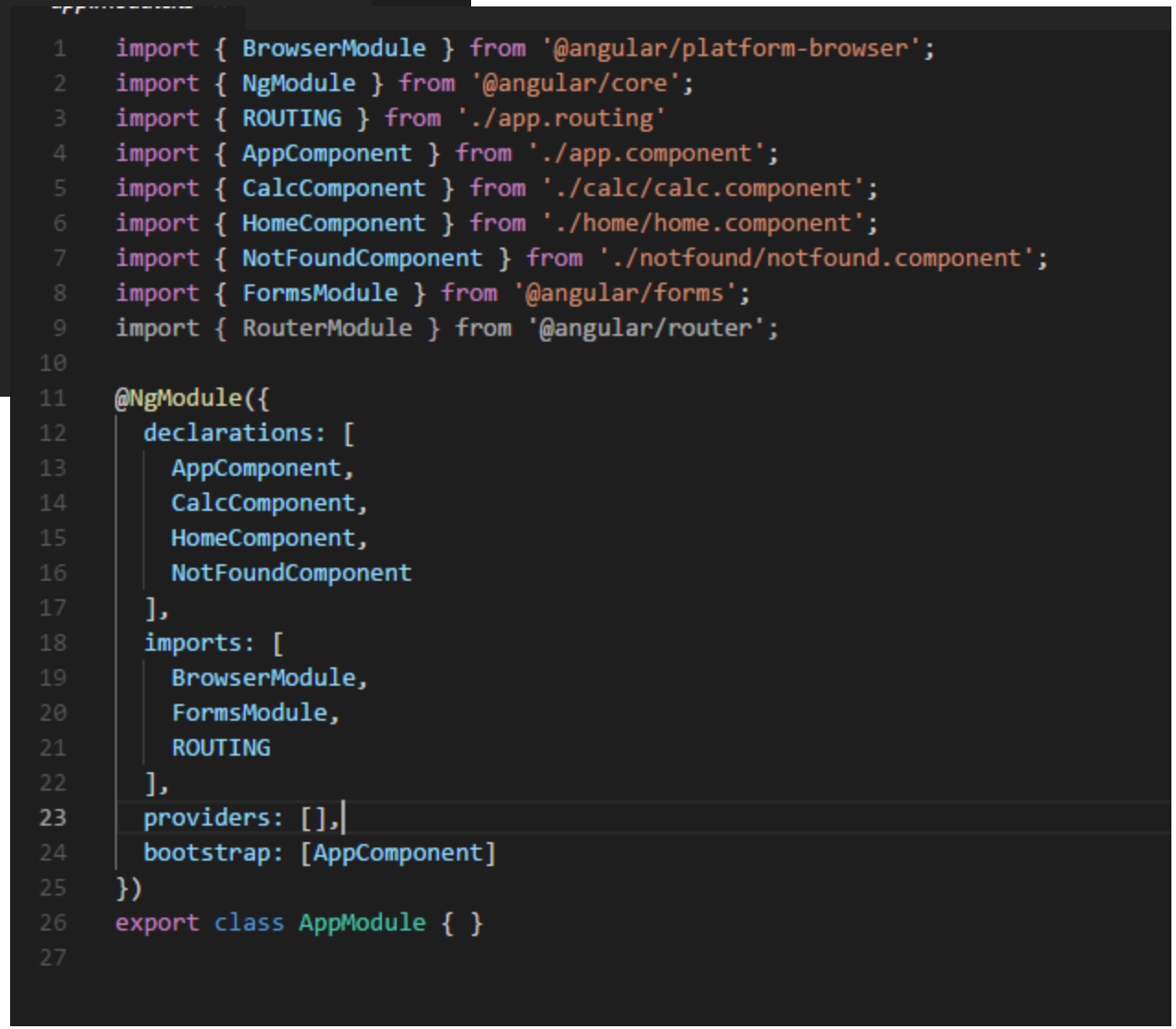

Module
Component
{}
Module
Service
{}
Module
value
3.1415
Module
Fn
λ
Template
< >
Metadata
Directive
{ }
Property
Binding
Metadata
Event
Binding
Injector
Service
Component

Topic 1

- Modules
- Components
  - Templates
  - Metadata
  - Data binding
  - Directives
  - Pipes
- Services and dependency injection







```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { ROUTING } from './app.routing'
import { AppComponent } from './app.component';
import { CalcComponent } from './calc/calc.component';
import { HomeComponent } from './home/home.component';
import { NotFoundComponent } from './notfound/notfound.component';
import { FormsModule } from '@angular/forms';
import { RouterModule } from '@angular/router';

@NgModule({
  declarations: [
    AppComponent,
    CalcComponent,
    HomeComponent,
    NotFoundComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    ROUTING
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

MODULES

- Modules are used in Angular JS to put logical boundaries in your application
- Angular apps are composed of modules.
- Modules export things — classes, function, values — that other modules import
- Usually Module has a single purpose and it export one thing such as Component class
- Applications are collection of modules with each module has a one specific task
- Usually Modules exports Component classes, Services, Pipes etc
- Angular provides various Modules Libraries such as, @angular/core, @angular/common, @angular/router etc

MODULES

Option	Description
providers	Defines the set of injectable objects that are available in the injector of this module.
declarations	Specifies a list of directives/pipes that belong to this module.
imports	Specifies a list of modules whose exported directives/pipes should be available to templates in this module. This can also contain ModuleWithProviders.
exports	Specifies a list of directives/pipes/modules that can be used within the template of any component that is part of an Angular module that imports this Angular module.
entryComponents	Specifies a list of components that should be compiled when this module is defined. For each component listed here, Angular will create a ComponentFactory and store it in the ComponentFactoryResolver.
bootstrap	Defines the components that should be bootstrapped when this module is bootstrapped. The components listed here will automatically be added to entryComponents.
schemas	Elements and properties that are not Angular components nor directives have to be declared in the schema.
id	An opaque ID for this module, e.g. a name or a path. Used to identify modules in getModuleFactory. If left undefined, the NgModule will not be registered with getModuleFactory.

Angular Module (ngModule)

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CricketMasterComponent} from './cricket-master.component';

@NgModule({
imports: [ BrowserModule ],
//modules to use
exports:[],
//exported members will be used in other modules
declarations: [],
//members used in html declarations i.e(component,directives,pipes)
providers:[],
//services to be injected,will lazily initialize.
bootstrap:[CricketMasterComponent]
// component inserted in DOM during,entry point of module

})
export class TrainingModule {

}
```

Decorator Object

Custom import

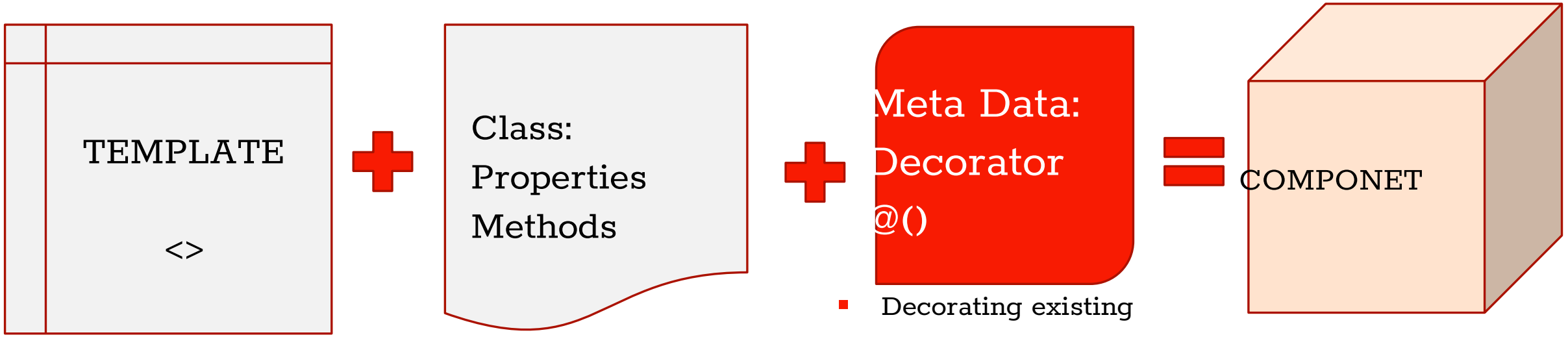
Module Decorator Function

Custom Module

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { CricketMasterComponent} from './cricket-master.component';

@NgModule({
    imports: [ BrowserModule ],
    declarations: [CricketMasterComponent],
    providers:[],
    bootstrap:[CricketMasterComponent]
})
export class TrainingModule {
        // module logic
}
```

What is component



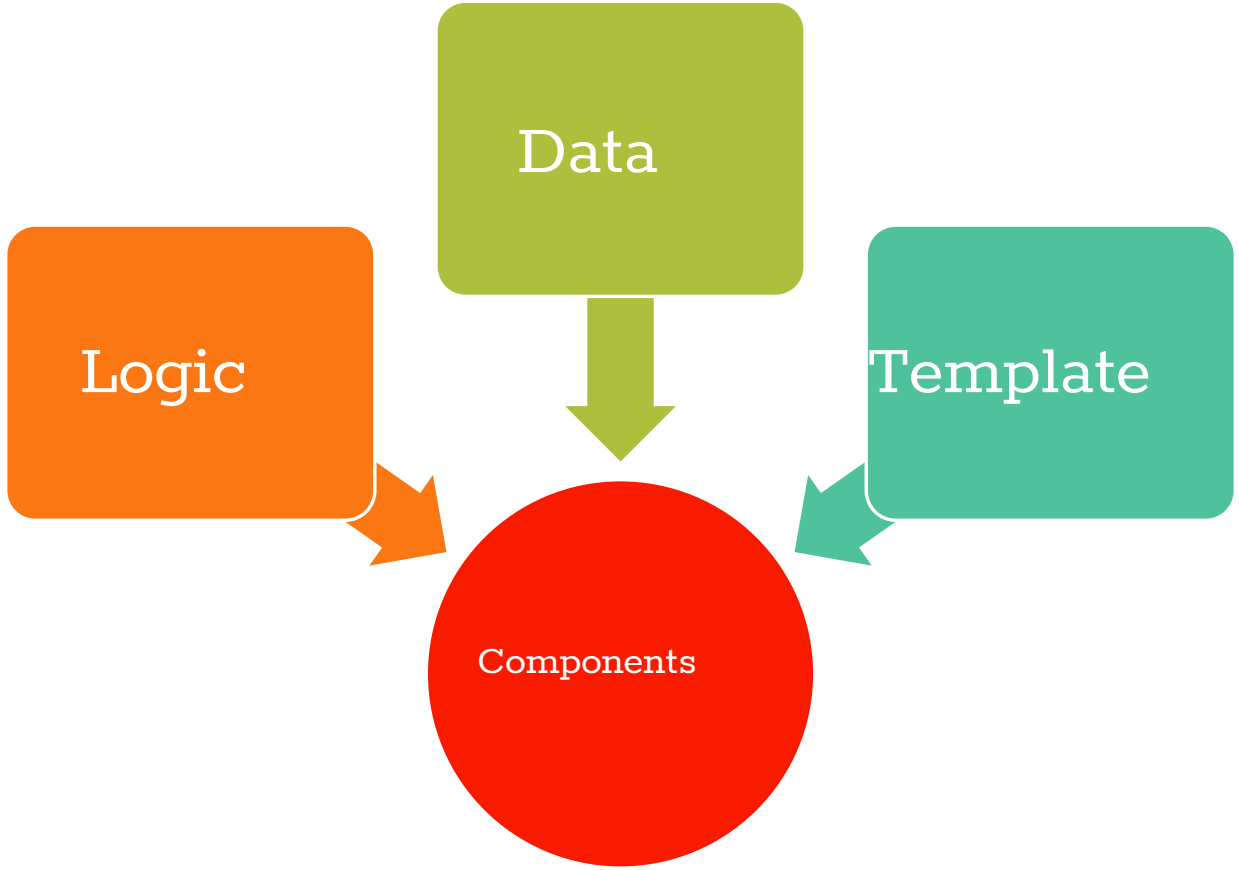
- View or layout created with HTML, CSS
- Data binding
- Directive

- Properties :data
- Method : Logic

- Decorating existing class
- Used by angular @Component, - Template, Styles

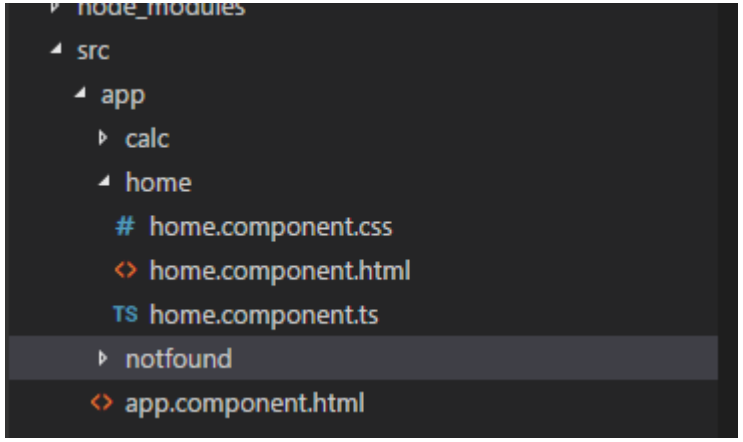
COMPONENTS

- Components are just ES6 classes
- A Component is a main building block of an AngularJS application
- An application may have any number of Components
- Data and logic can be created or brought on the page using Components
- Custom elements can be created or brought on the page using Components



Create Component

- Step 1: Create a typescript class with properties and behavior
- Step 2: Decorate class with Component metadata
- Step 3: Import statement- importing required modules to create this component.
- Step 4: To use, either bootstrap the component or use as directive in another components



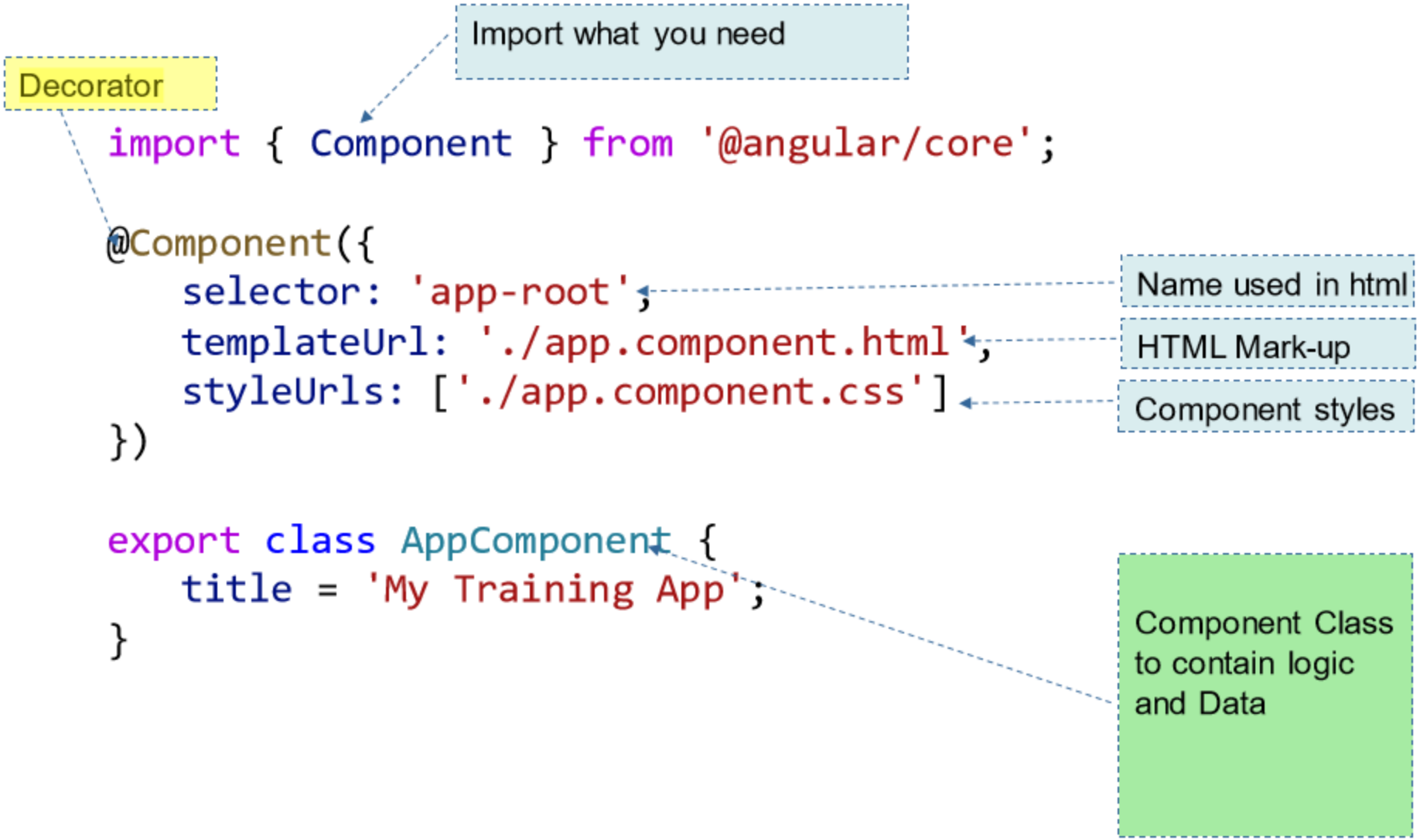

```
import {Component} from 'angular2/core';

@Component({
    selector:"helloworld",
    template:`
    <h1>{{message}}</h1>
   `,
   styles:["h1{color:red}"]
})
export class HelloworldComponent{

    message : string = "Hello World";

}
```

Component in action



Lifecycle Hooks [starts on contractor]

Hook	Purpose and Timing
ngOnChanges()	Called after a bound input property changes. Called before ngOnInit() and whenever one or more data-bound input properties change.
ngOnInit()	Called when once directive/component Initialize. Called *once*, after the *first* ngOnChanges().
ngDoCheck()	Called during every changes detection run.
ngAfterContentInit()	Respond after Angular projects external content (ng-content) into the component's view / the view that a directive is in. Called *once* after the first ngDoCheck().
ngAfterContentChecked()	Respond after Angular checks the content projected into the directive/component. Called after the ngAfterContentInit() and every subsequent ngDoCheck().
ngAfterViewInit()	Respond after Angular initializes the component's views and child views / the view that a directive is in. Called *once* after the first ngAfterContentChecked().
ngAfterViewChecked()	Respond after Angular checks the component's views and child views / the view that a directive is in. Called after the ngAfterViewInit and every subsequent ngAfterContentChecked().
ngOnDestroy()	Cleanup just before Angular destroys the directive/component. Unsubscribe Observables and detach event handlers to avoid memory leaks. Called *just before* Angular destroys the directive/component.

Template

- A template is HTML that tells Angular how to render a component
- Templates include data bindings as well as other components and directives
- Angular leverages native DOM events and properties which dramatically reduces the need for a ton of builtin directives
- Angular leverages shadow DOM to do some really interesting things with view encapsulation

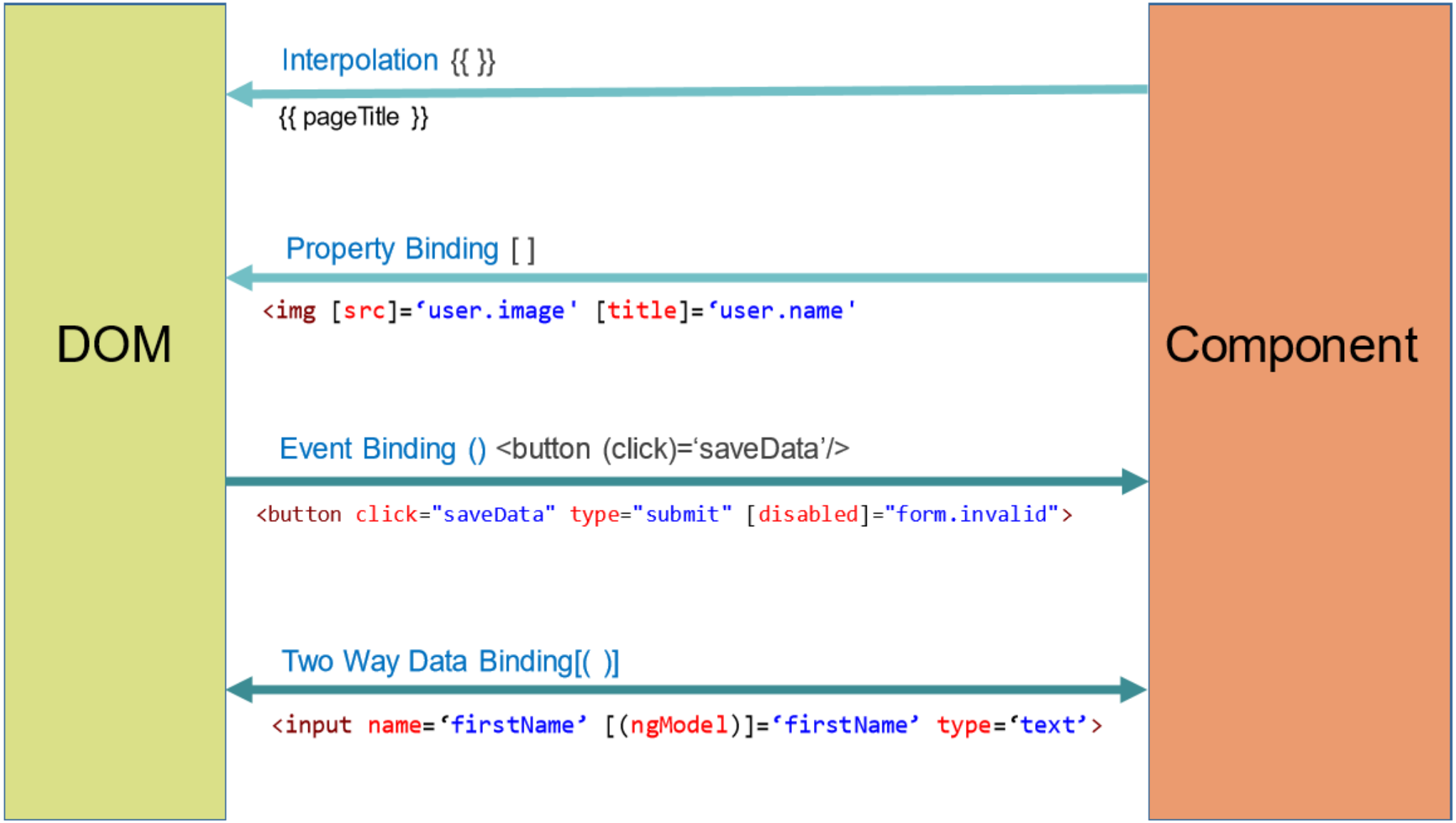
```
<h1>{{title}}</h1>
<p>{{body}}</p>
<hr/>
<experiment *ngFor="#experiment of experiments"
  [experiment]="experiment"></experiment>
<hr/>
<div>
  <h2 class="text-error">Experiments: {{message}}</h2>
  <form class="form-inline">
    <input type="text"
      [(ngModel)]="message" placeholder="Message">
    <button type="submit" class="btn"
      (click)="updateMessage(message)">Update Message
    </button>
  </form>
</div>
```

Metadata

- Metadata allows Angular to process a class
- We can attach metadata with TypeScript using decorators
- Ex. @Component() decorator
- Takes a config option with the selector, template(Url), providers, directives, pipes and styles

```
@Component({
  selector: 'home',
  templateUrl: 'app/home/home.component.html'
})
export class HomeComponent{ }
```

Data Binding



Data Binding

```
<!-- no binding -->
<input type="text" ngModel/>
<!-- one Way Binding or property binding -->
<input type="text" [ngModel]="firstName"/>
<!-- Two Way Binding -->
<input type="text" [ngModel]="firstName" (ngModelChange) =
"firstName=$event"/>
<!-- Two way binding (Banana in a Box binding) -->
<input type="text" [(ngModel)]="firstName"/>
```

What is DIRECTIVE ?

- A directive is a custom HTML element that is used to extend the power of HTML.
- Change the DOM layout by adding and removing DOM elements.
- There are three kinds of directives in Angular:
  1. Components — directives with a template.
  2. Structural directives —change the DOM layout by adding and removing DOM elements.
  3. Attribute directives / Custom Directive —change the appearance or behavior of an element, component, or another directive.

Structural directives

- NgIf

```
<div *ngIf="hero" class="name">{{hero.name}}</div>
```

- NgFor

```
<ul>
  <li *ngFor="let hero of heroes">{{hero.name}}</li>
</ul>
```

- NgSwitch

```
<div [ngSwitch]="hero?.emotion">
  <app-happy-hero    *ngSwitchCase="'happy'"    [hero]="hero"></app-happy-hero>
  <app-sad-hero      *ngSwitchCase="'sad'"      [hero]="hero"></app-sad-hero>
  <app-confused-hero *ngSwitchCase="'app-confused'" [hero]="hero"></app-confused-hero>
  <app-unknown-hero  *ngSwitchDefault           [hero]="hero"></app-unknown-hero>
</div>
```

Attribute Directives

- An Attribute directive changes the appearance or behavior of a DOM element.
- An attribute directive minimally requires building a controller class annotated with @Directive, which specifies the selector that identifies the attribute.
- *Attribute directives* are used as attributes of elements. The built-in NgStyle directive in the Template Syntax guide, for example, can change several element styles at the same time.
- The controller class implements the desired directive behaviour.
  - Ex : ng generate directive highlight

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor() { }
}
```

  - https://angular.io/guide/attribute-directives

Pipes

- An Attribute directive changes the appearance or behavior of a DOM element.
- An attribute directive minimally requires building a controller class annotated with @Directive, which specifies the selector that identifies the attribute.
- *Attribute directives* are used as attributes of elements. The built-in NgStyle directive in the Template Syntax guide, for example, can change several element styles at the same time.
- The controller class implements the desired directive behaviour.
  - Ex : ng generate directive highlight

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

@Directive({
  selector: '[appHighlight]'
})
export class HighlightDirective {
  constructor() { }
}
```

  - https://angular.io/guide/attribute-directives

Services and Dependency Injection

- Represent shared logic class
- Share data or functions between different parts of angular application
- Independent from other component of application
- @Injectable() decorator.

```
import {Injectable} from 'angular2/core';
import {Experiment} from './experiment.model';

@Injectable()
export class ExperimentsService {
  private experiments: Experiment[] = [];

  getExperiments(): Experiment[] {
    return this.experiments;
  };
}
```

Services

Create

```
import { Injectable } from
@angular/core‘;
@Injectable()
export class ProductService {
     getProducts(): IProduct[] {
     }
}
```

Register

```
import { ProductService } from './product.service';
@Component({ selector: 'pm-root',
template: `<div><h1>{{pageTitle}}</h1>
 <pm-products></pm-products> </div> `,
providers: [ProductService]
})
export class AppComponent { }
```

Inject

```
import { ProductService } from './product.service';

@Component({ selector: 'pm-products'
, templateUrl: './product-list.component.html'
})

export class ProductListComponent {
     constructor(private _productService: ProductService) {
     }
}
```

HTTP

- Service that allows data to be returned or saved using Promises or Observables
- Registering HTTP_PROVIDERS