

# Getting start with Angular



Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata



somkiat.cc

Page Messages Notifications 3 Insights Publishing Tools Settings Help ▾

somkiat.cc  
@somkiat.cc

Home Posts Videos Photos

Like Following Share ...

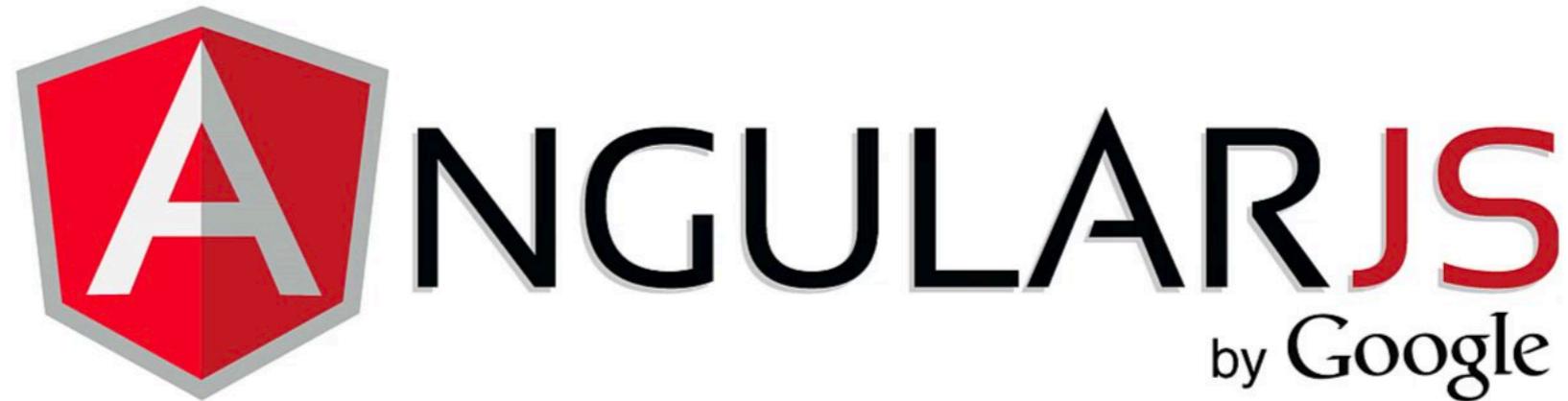
+ Add a Button



# Agenda

- Introduction to Angular
- Installation and configuration
- Structure of Angular project
- Introduction to TypeScript
- Design and develop component/service
- Routing management
- Working with RESTful APIs





<https://angularjs.org/>



# Angular



<https://angular.io/>

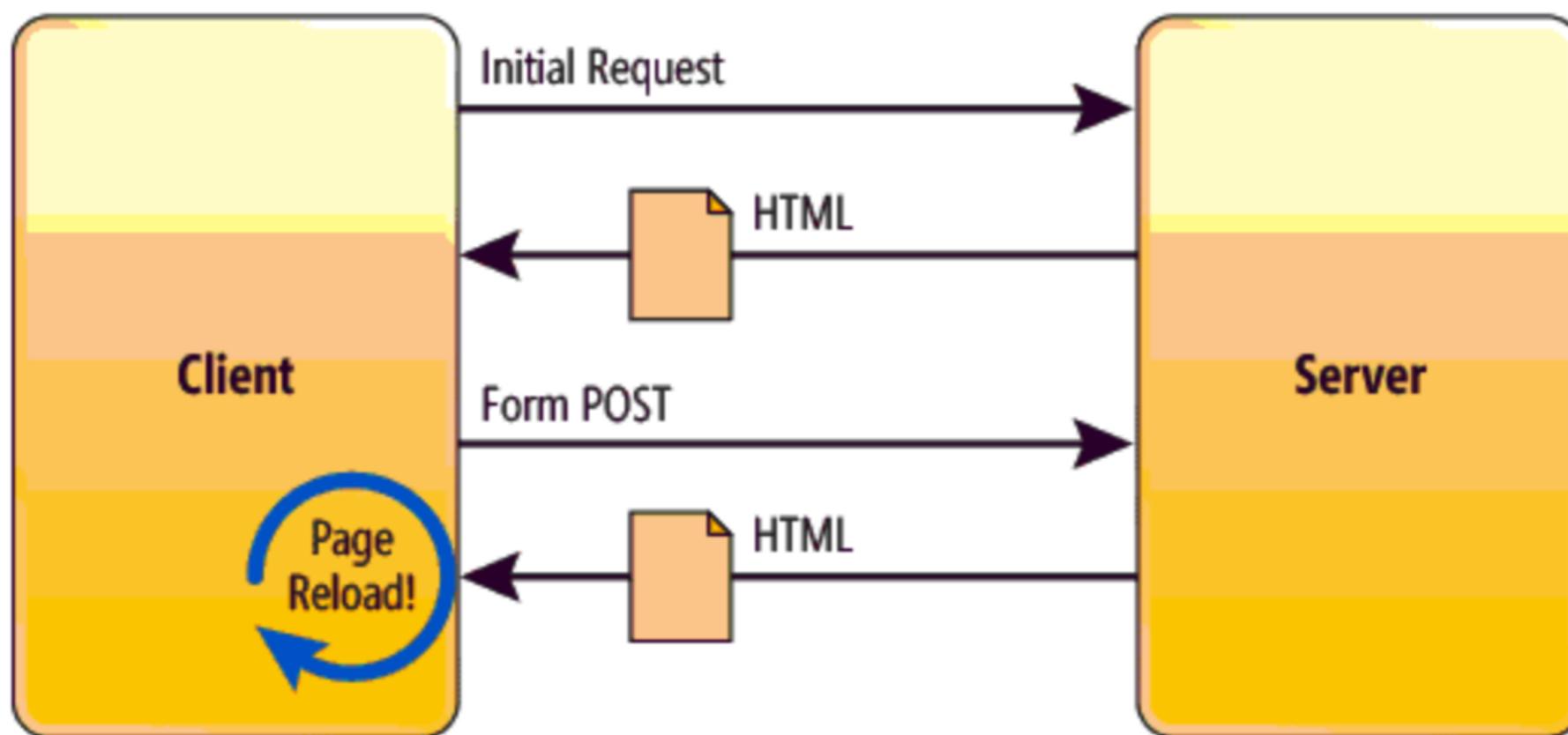


# What is Angular ?

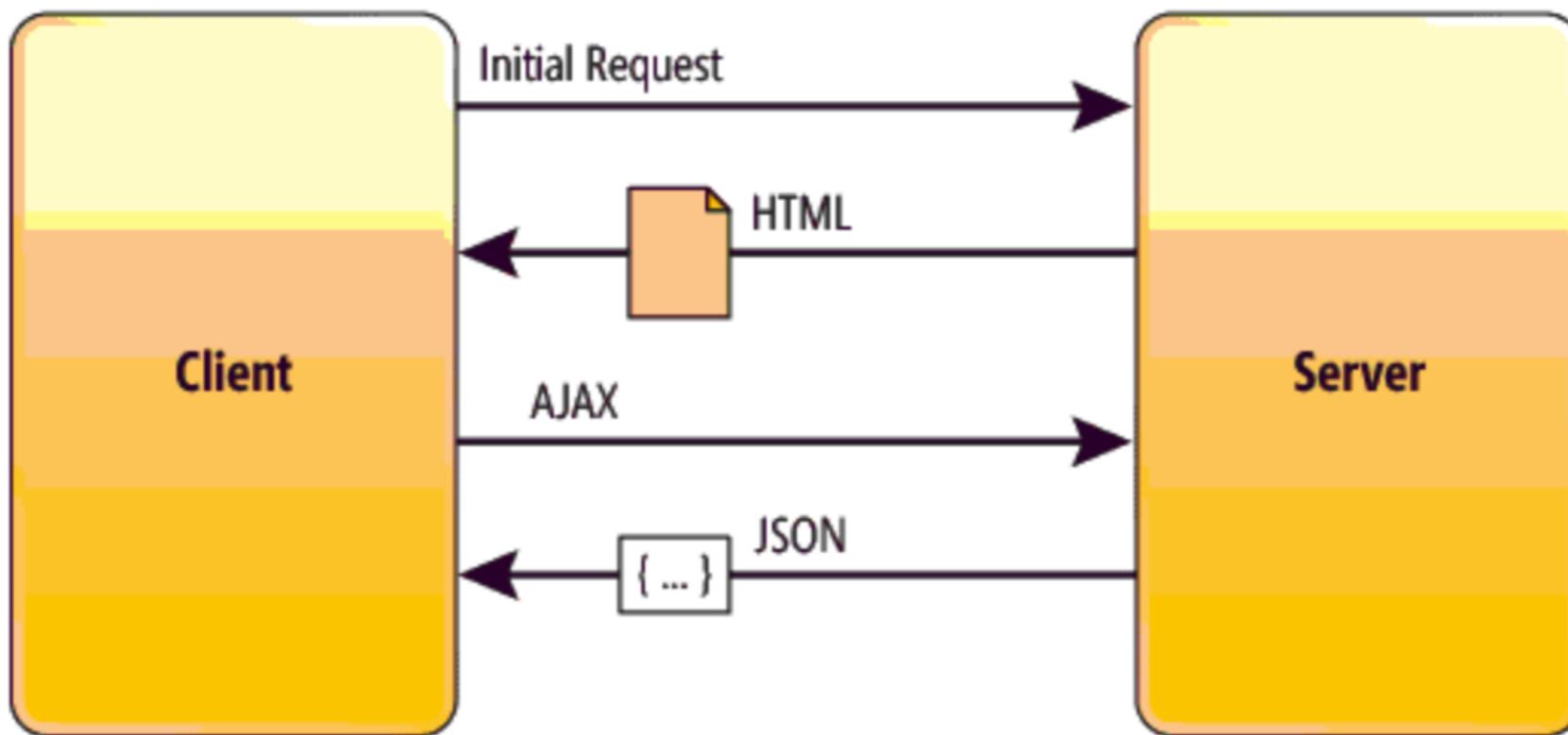
**JavaScript framework** with allows us to create  
**reactive Single Page Application (SPA)**



# Traditional



# Single Page Application



# Angular

Angular 10

Angular ...

Angular 5

Angular 4

Angular 3

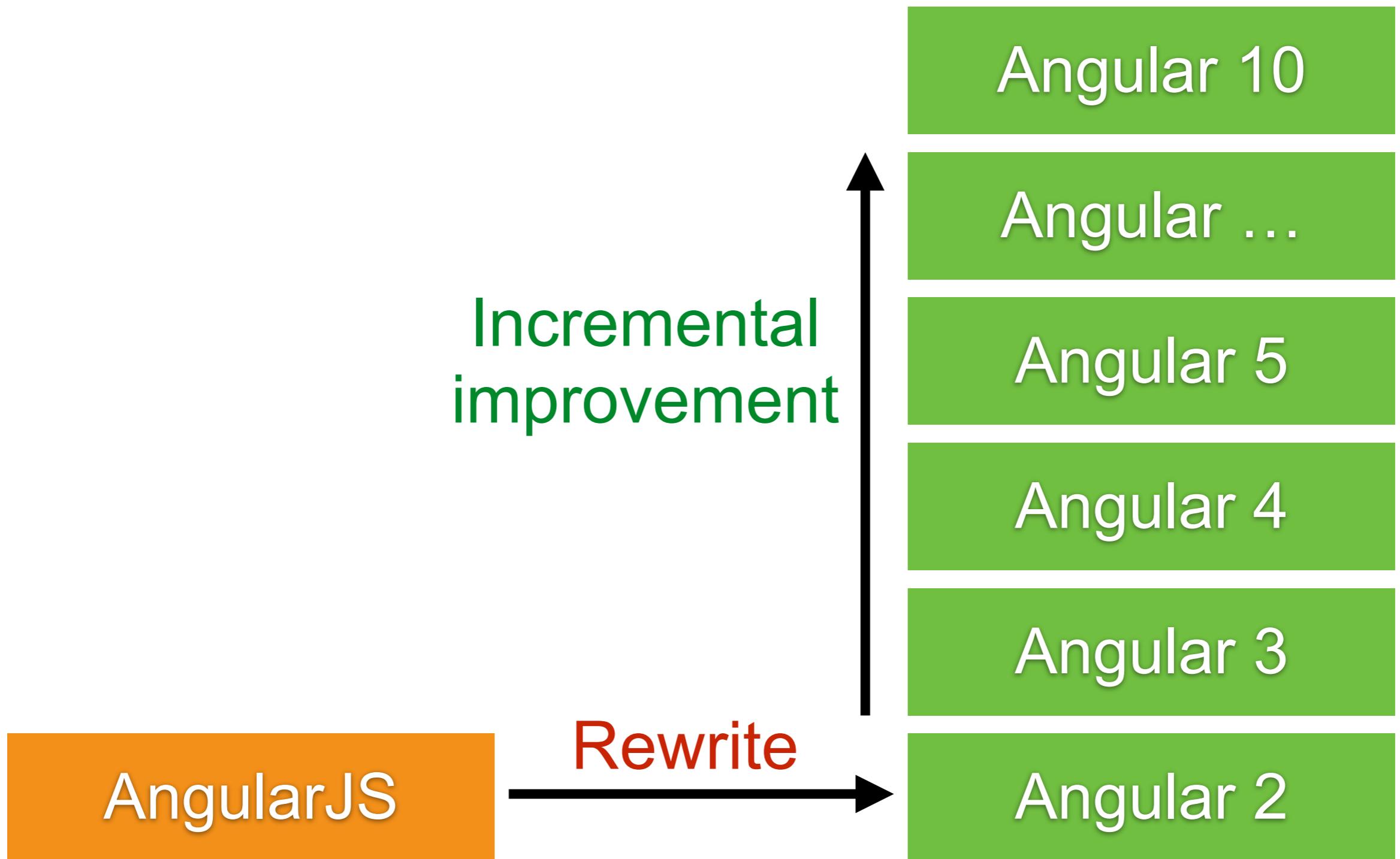
Angular 2

AngularJS

Rewrite



# Angular



# Software requirement

## Install NodeJS

**12.18.2 LTS**

Recommended For Most Users

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

**14.5.0 Current**

Latest Features

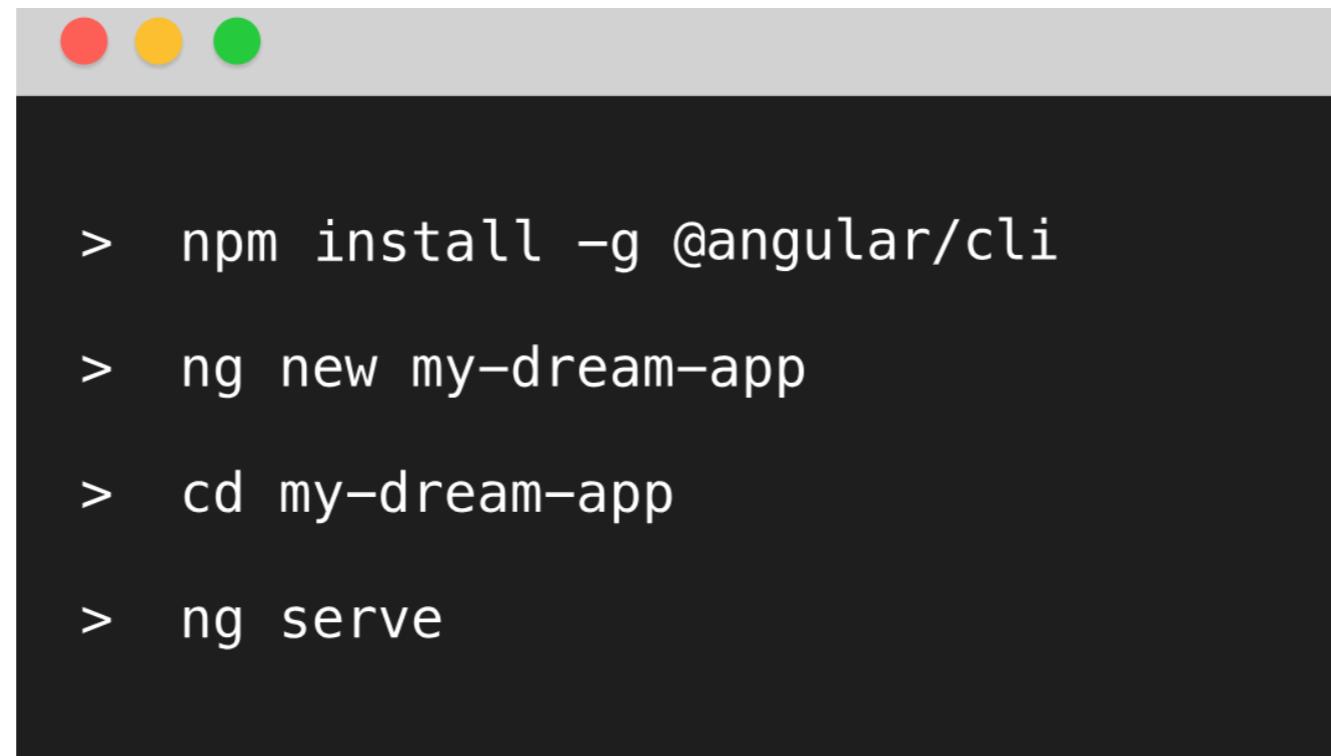
[Other Downloads](#) | [Changelog](#) | [API Docs](#)

<https://nodejs.org/en/>



# Angular CLI

A tool to initialise, develop, scaffold and maintain  
Angular application

A screenshot of a macOS terminal window. The title bar shows three colored window control buttons (red, yellow, green). The main window area contains the following text:

```
> npm install -g @angular/cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

<https://cli.angular.io/>



# Install Angular CLI

```
$npm install -g @angular/cli@latest
```



# Version of Angular

\$ng version

```
Angular CLI: 10.0.3
Node: 12.18.1
OS: darwin x64

Angular:
...
Ivy Workspace:
```



# Version of Angular

\$ng version

Package	Version
<hr/>	
@angular-devkit/architect	0.1000.3
@angular-devkit/core	10.0.3
@angular-devkit/schematics	10.0.3
@schematics/angular	10.0.3
@schematics/update	0.1000.3
rxjs	6.5.5



# Try to create first project

\$ng new hello-app

```
CREATE hello-app/README.md (1025 bytes)
CREATE hello-app/angular.json (3575 bytes)
CREATE hello-app/package.json (1313 bytes)
CREATE hello-app/tsconfig.json (384 bytes)
CREATE hello-app/tslint.json (2805 bytes)
CREATE hello-app/.editorconfig (245 bytes)
CREATE hello-app/.gitignore (503 bytes)
CREATE hello-app/src/environments/environment.prod.ts (51 bytes)
CREATE hello-app/src/environments/environment.ts (631 bytes)
CREATE hello-app/src/favicon.ico (5430 bytes)
CREATE hello-app/src/index.html (295 bytes)
```



# Run your app

```
$cd hello-app  
$ng serve
```

\*\* Angular Live Development Server is listening on localhost:4200, open your browser to <http://localhost:4200>

Date: 2018-07-19T17:42:59.617Z

Hash: cf107798cf25722bc556

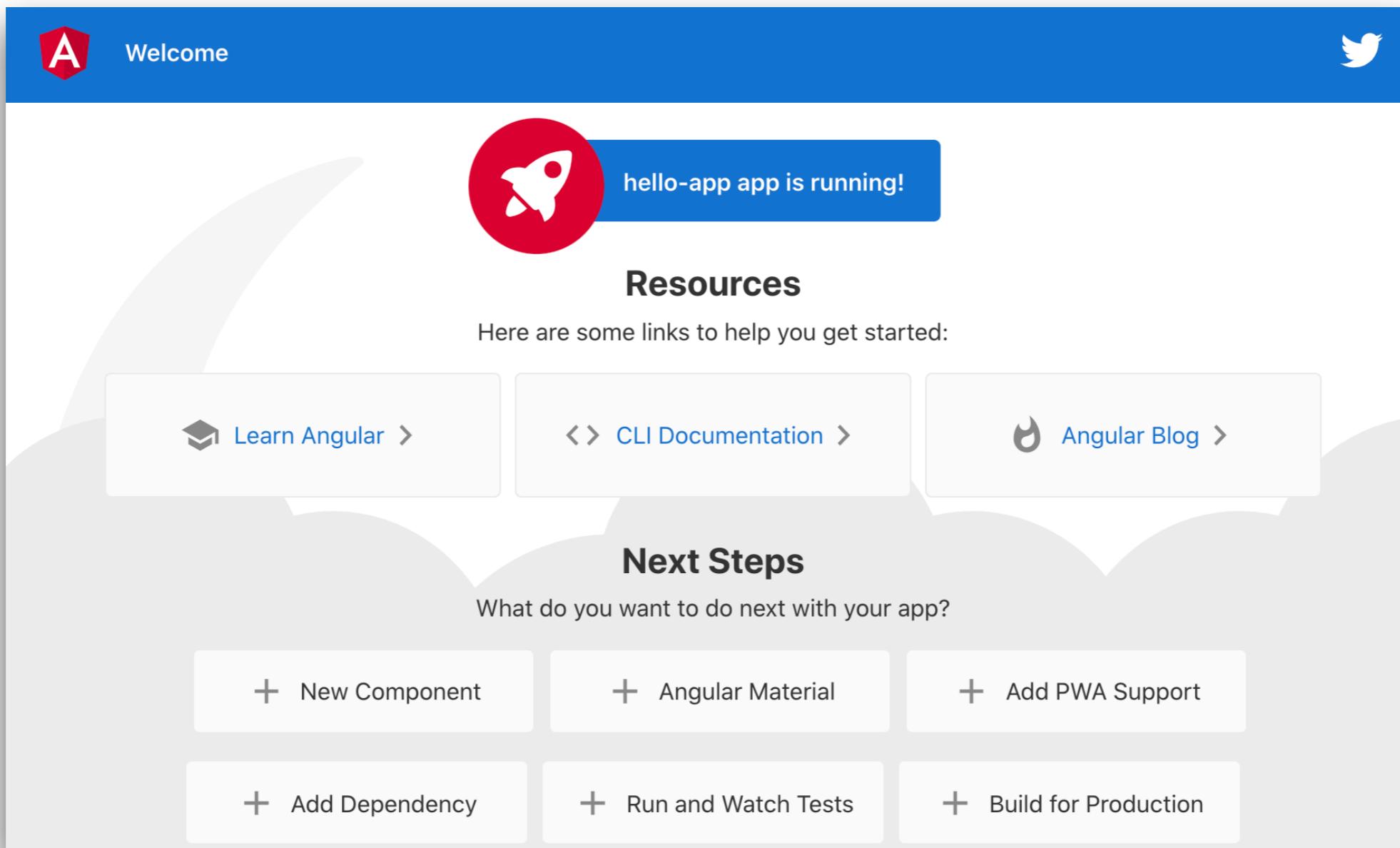
Time: 22285ms

```
chunk {main} main.js, main.js.map (main) 10.7 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 227 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 5.22 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 15.6 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.06 MB [initial] [rendered]
i 「wdm」: Compiled successfully.
```



# Open in browser

http://localhost:4200/



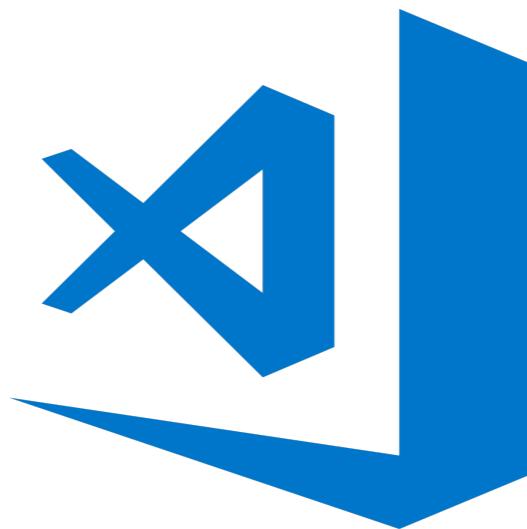
# Ng commands

\$ng build

\$ng test



# Install Text Editor



# Angular 10 features

TypeScript 2.9

Browser configuration (default)

Localization (L10n)

Default bundle size (drop 75%)

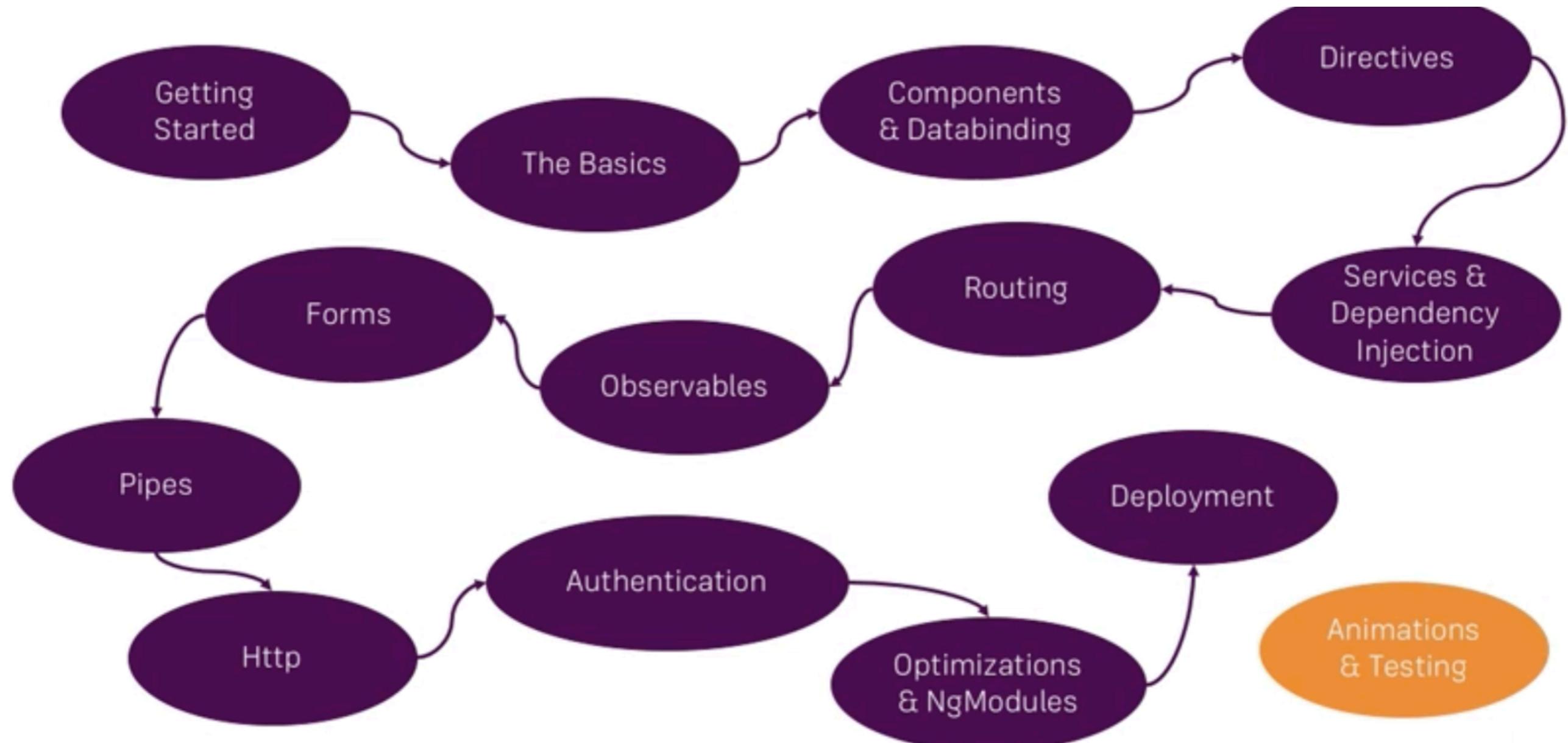
Performance improvement



# Let's start



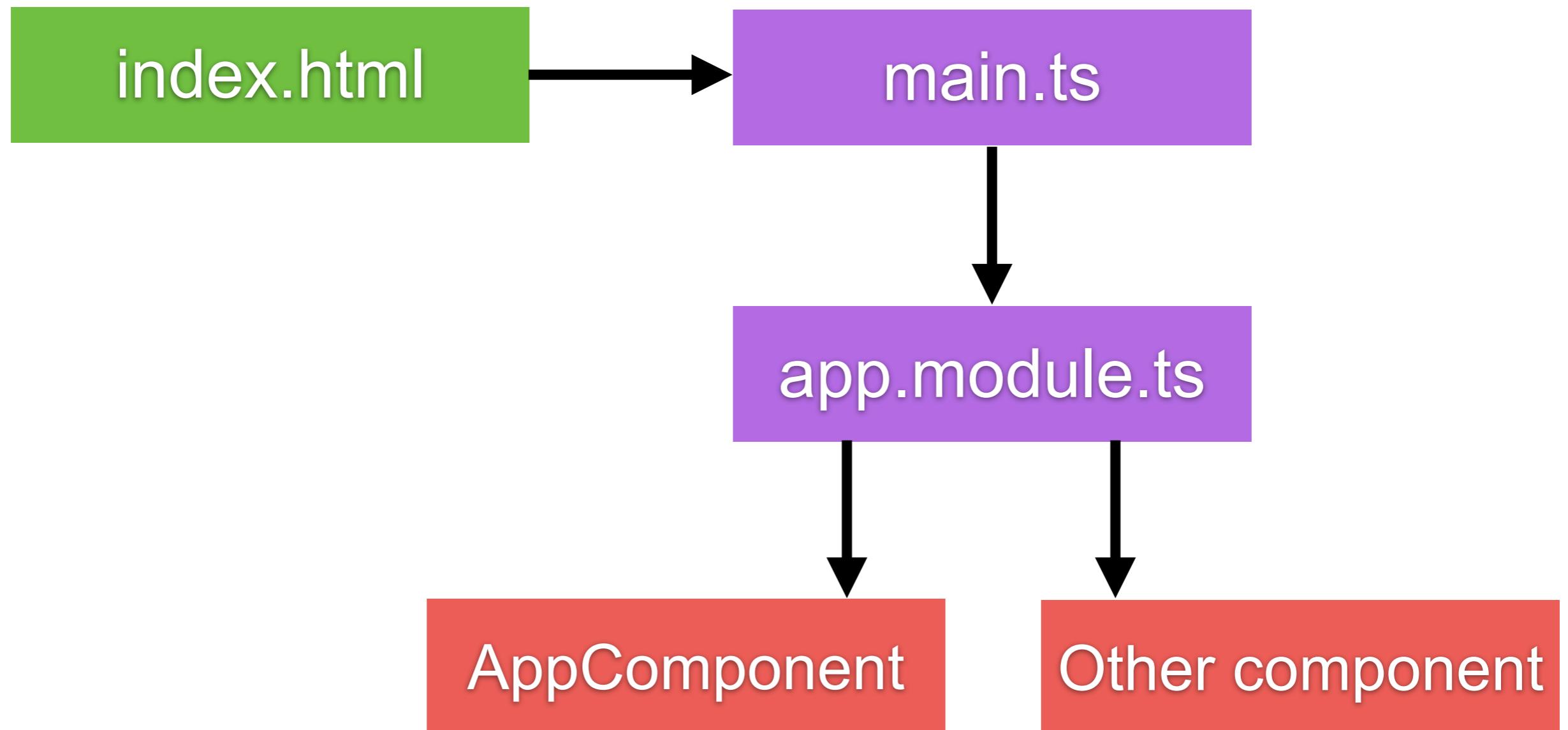
# Learning structure



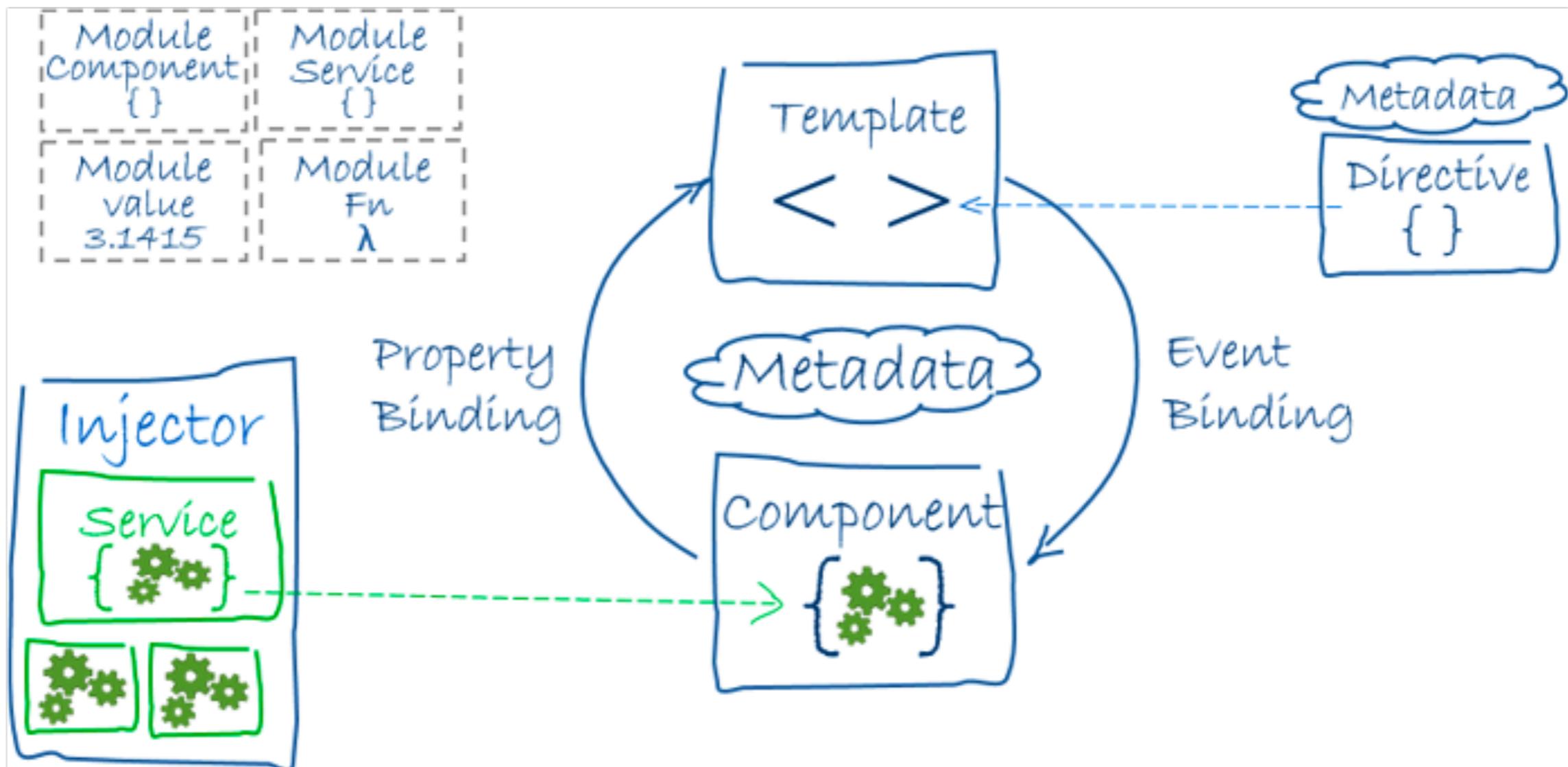
# Basic of Angular



# How Angular working ?



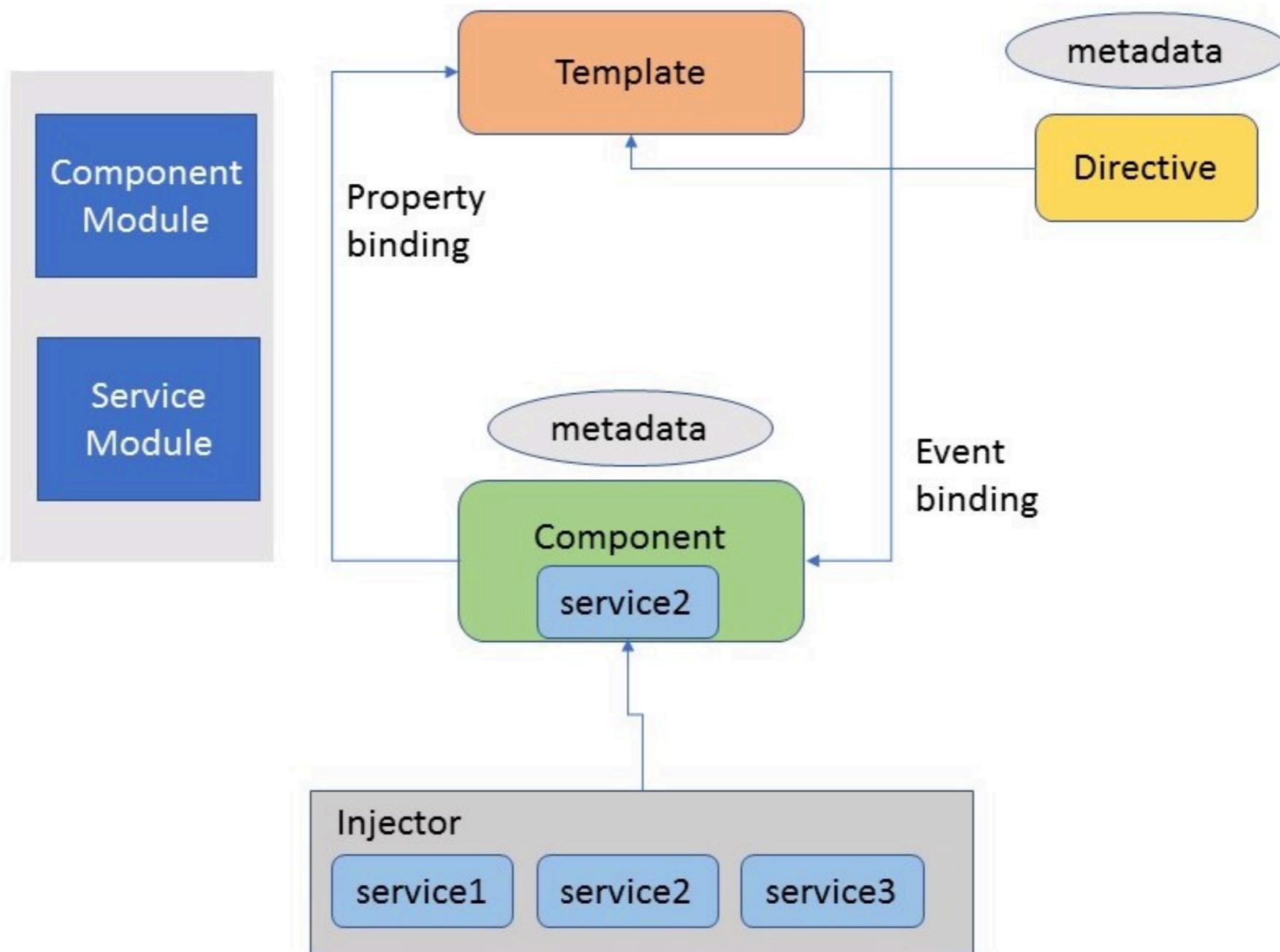
# Angular Architecture



<https://angular.io/guide/architecture>



# Angular Architecture

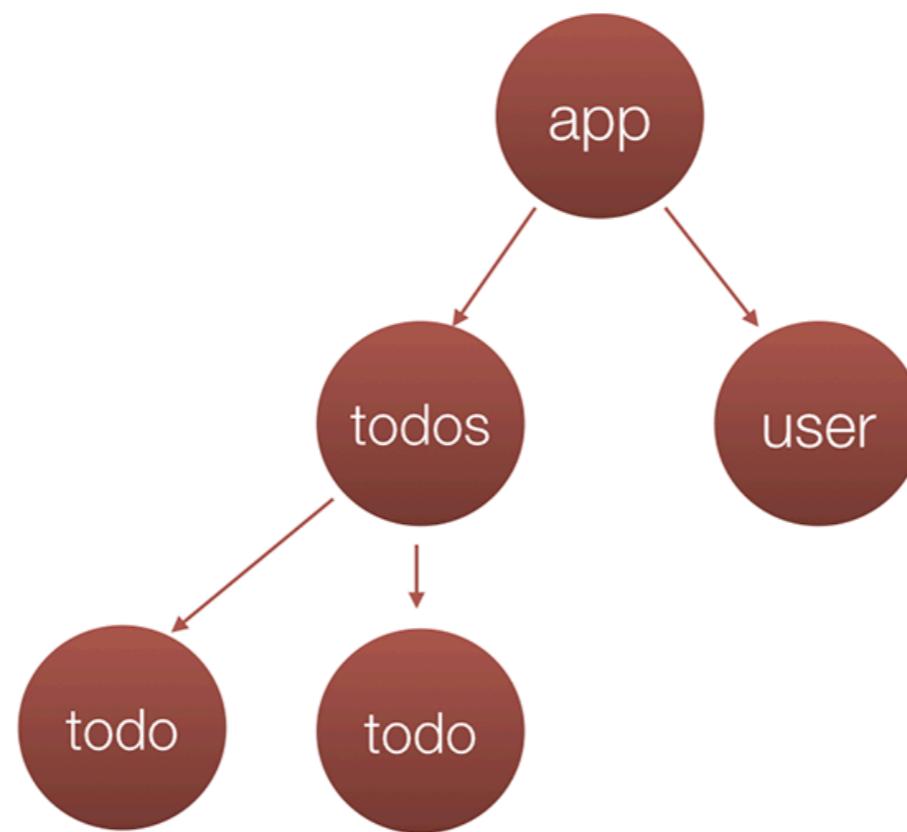


# Components

Angular application is a tree of **Components**

Top level component is the application itself

Component is rendered by the browser



# Components

Own template

Own style

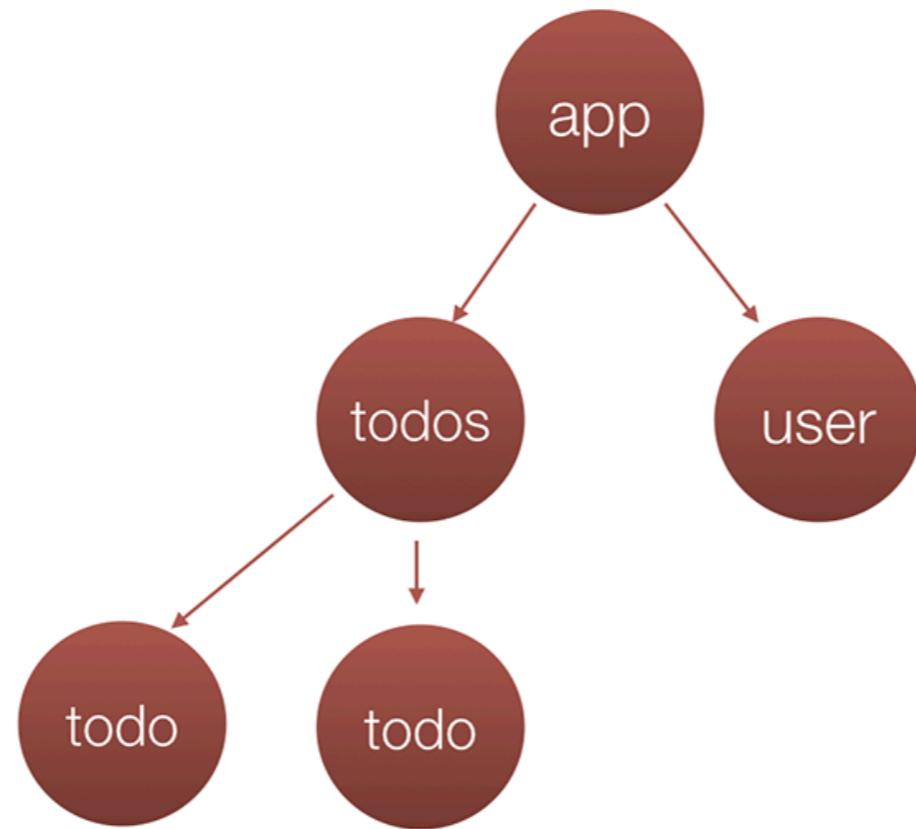
Own business logic

**Split complex app to small component !!**



# Components

Composable  
Reusable  
Hierarchical



# Design components

Home

About

Contact

Menu

Item 1

Item 2

Item 3



# Header component

Home

About

Contact

Menu

Item 1

Item 2

Item 3



# Sidebar/Menu component

Home

About

Contact

Menu

Item 1

Item 2

Item 3



# Item component

Home

About

Contact

Menu

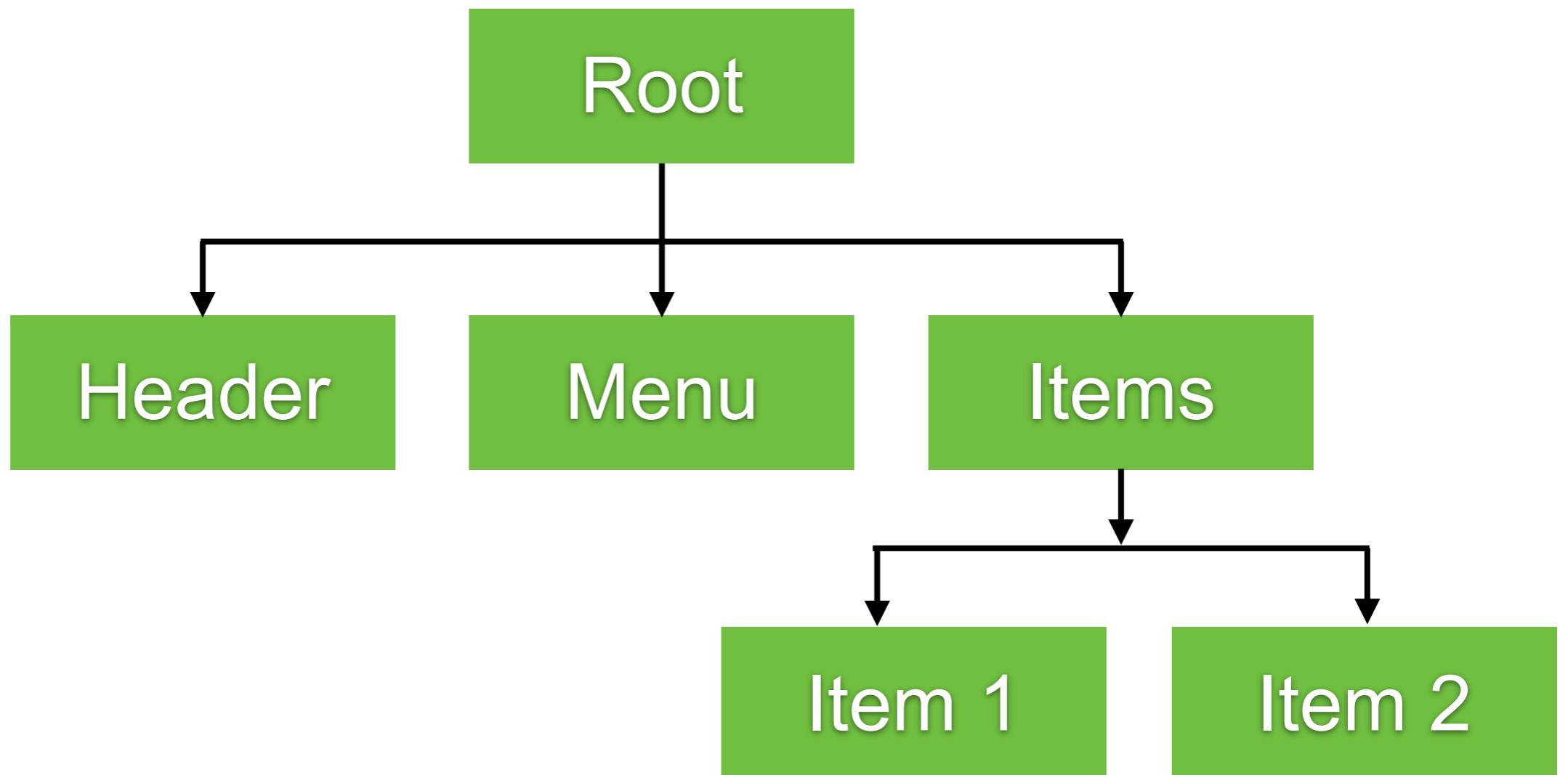
Item 1

Item 2

Item 3



# Design components



# Component

Template

Class

Metadata

View layout

Code support view

Extra data

Create with HTML

Create with TypeScript

Define decorator

Binding and directives

Data and logic

@Component



# Component

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

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

Metadata

Template

```
export class AppComponent {  
  title = 'demo01';  
}
```



# Metadata

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'demo01';
}
```

*Component decorator*

*Directive name used in HTML*



# Metadata

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

@Component({← Component decorator
  selector: 'app-root',← Directive name used in HTML
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})View layout , template and style
```

```
export class AppComponent {
  title = 'demo01';
}
```



# Component

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

```
export class AppComponent {
  title = 'demo01';
}
```

Class



# Component

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

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

Import module/library

```
export class AppComponent {  
  title = 'demo01';  
}
```



# Import ?

3-party libraries

ES modules

**Angular modules**



# Angular is Modular

@angular/  
core

@angular/  
http

@angular/  
router



# Create new component with CLI

```
$ng generate component <name>
```



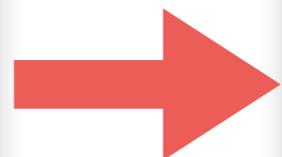
# Workshop

**Login Page**

Email address:

Password:

**Login**



**List of users**

<b>Id</b>	<b>First Name</b>	<b>Last Name</b>	<b>Email</b>
1	User 1	Last name 1	user1@gmail.com
2	User 2	Last name 2	user2@gmail.com
3	User 3	Last name 3	user3@gmail.com



# Login Page

## Login Page

Email address:

Password:

**Login**



# Create Login Component

\$ng generate component login

```
CREATE  src/app/login/login.component.css (0 bytes)
CREATE  src/app/login/login.component.html (24 bytes)
CREATE  src/app/login/login.component.spec.ts (621 bytes)
CREATE  src/app/login/login.component.ts (265 bytes)
UPDATE  src/app/app.module.ts (560 bytes)
```



# Edit file login.component.html

```
<form>
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control"
           id="email" name="email" [(ngModel)]
           ="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control"
           id="password" name="password" [
           (ngModel)]="password">
  </div>
  <button class="btn btn-success" (click)="login()"
         ">Login</button>
</form>
```



# Form have email and password

```
<form>
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control"
           id="email" name="email" [(ngModel)]="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control"
           id="password" name="password" [(ngModel)]="password">
  </div>
  <button class="btn btn-success" (click)="login()">Login</button>
</form>
```



# Handle click event of Login button

```
<form>
  <div class="form-group">
    <label for="email">Email address:</label>
    <input type="email" class="form-control"
           id="email" name="email" [(ngModel)]="email">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control"
           id="password" name="password" [(ngModel)]="password">
  </div>
  <button class="btn btn-success" (click)="login()">Login</button>
</form>
```



# Edit file login.component.ts

Add fields to keep email and password

```
export class LoginComponent implements OnInit {  
  email: string;  
  password: string;  
  
  constructor() { }  
}
```



# Edit file login.component.ts

Add login() to handle click action from HTML

```
login() {  
  if (this.email === 'test' && this.password === 'test') {  
    alert('Login success');  
  } else {  
    alert('Login failure');  
  }  
}
```



# Run and see result

## Login Page

Email address:

Password:

**Login**



# List of user page

## List of users

<b>Id</b>	<b>First Name</b>	<b>Last Name</b>	<b>Email</b>
1	User 1	Last name 1	user1@gmail.com
2	User 2	Last name 2	user2@gmail.com
3	User 3	Last name 3	user3@gmail.com



# Create User Component

\$ng generate component user

```
CREATE  src/app/user/user.component.css (0 bytes)
CREATE  src/app/user/user.component.html (23 bytes)
CREATE  src/app/user/user.component.spec.ts (614 bytes)
CREATE  src/app/user/user.component.ts (261 bytes)
UPDATE  src/app/app.module.ts (531 bytes)
```



# Create Model of User

\$ng generate class models/user

```
export class User {  
    id: number;  
    firstName: string;  
    lastName: string;  
    email: string;  
}
```



# Edit file user.component.ts

Create list of user field to UserComponent

```
import { Component, OnInit } from '@angular/core';
import { User } from '../models/user';

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {

  users: User[];

  constructor() { }

}
```



# Edit file user.component.ts

Create **fake data** of users

```
ngOnInit() {  
  const fakeUsers = [  
    {id: 1, firstName: 'User 1', lastName: 'Last name 1', email: 'user1@gmail.com'},  
    {id: 2, firstName: 'User 2', lastName: 'Last name 2', email: 'user2@gmail.com'},  
    {id: 3, firstName: 'User 3', lastName: 'Last name 3', email: 'user3@gmail.com'},  
  ];  
  this.users = fakeUsers;  
}
```



# We need routing ?

Angular have @angular/router

/login

Login Page

Email address:

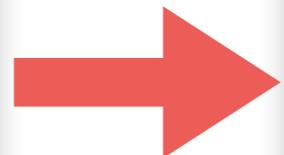
Password:

Login

/user

List of users

Id	First Name	Last Name	Email
1	User 1	Last name 1	user1@gmail.com
2	User 2	Last name 2	user2@gmail.com
3	User 3	Last name 3	user3@gmail.com



# Create AppRoutingModule

```
$ng generate module app-routing --flat --module=app
```

```
CREATE  src/app/app-routing.module.spec.ts (308 bytes)
CREATE  src/app/app-routing.module.ts (194 bytes)
UPDATE  src/app/app.module.ts (933 bytes)
```



# Open file app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [
    CommonModule
  ],
  declarations: []
})
export class AppRoutingModule { }
```



# Open file app-routing.module.ts

Add RouterModule to module

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

@NgModule({
  imports: [
    RouterModule
  ],
  declarations: []
})
export class AppRoutingModule { }
```



# Open file app-routing.module.ts

Add all route to Angular Routes

```
import { LoginComponent } from './login/login.component';
import { UserComponent } from './user/user.component';

const routes: Routes = [
  { path: 'login', component: LoginComponent },
  { path: 'user', component: UserComponent },
  { path: '', component: UserComponent },
];

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



# Edit app.component.html

Create template of app component

```
<h1 class="title">Main Page</h1>
<nav>
  <a routerLink="/login" routerLinkActive="active">Login</a> |
  <a routerLink="/user" routerLinkActive="active">List of user</a>
</nav>
<hr/>
<router-outlet></router-outlet>
```



# See result in first page

← → ⌂ ⓘ localhost:4200

# Main Page

[Login](#) | [List of user](#)

# Main Page

[Login](#) | [List of user](#)

## Login Page

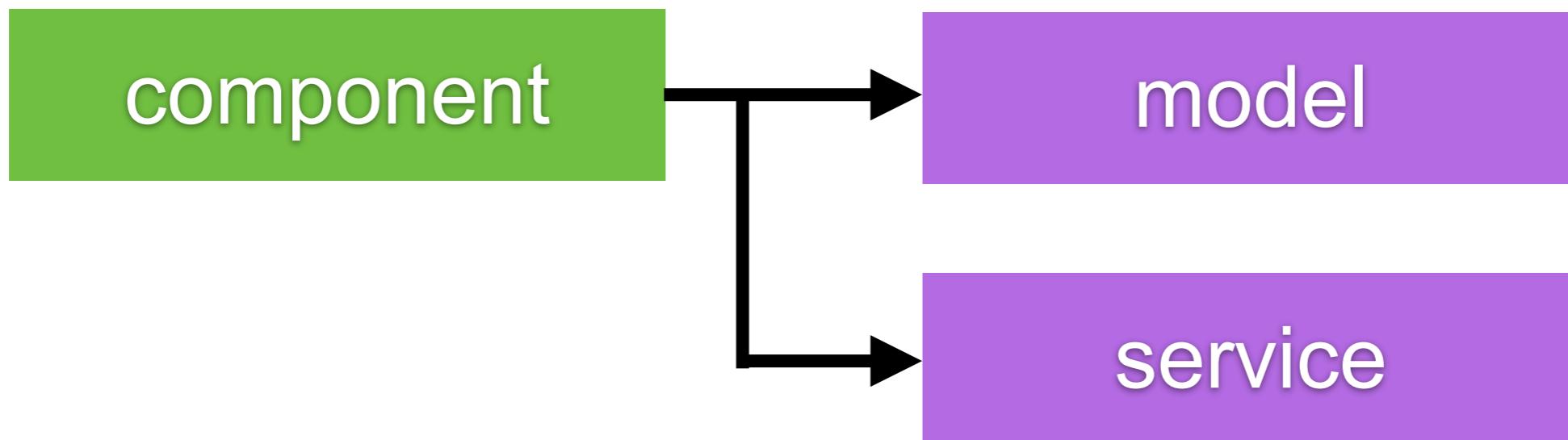
Email address:

Password:

**Login**



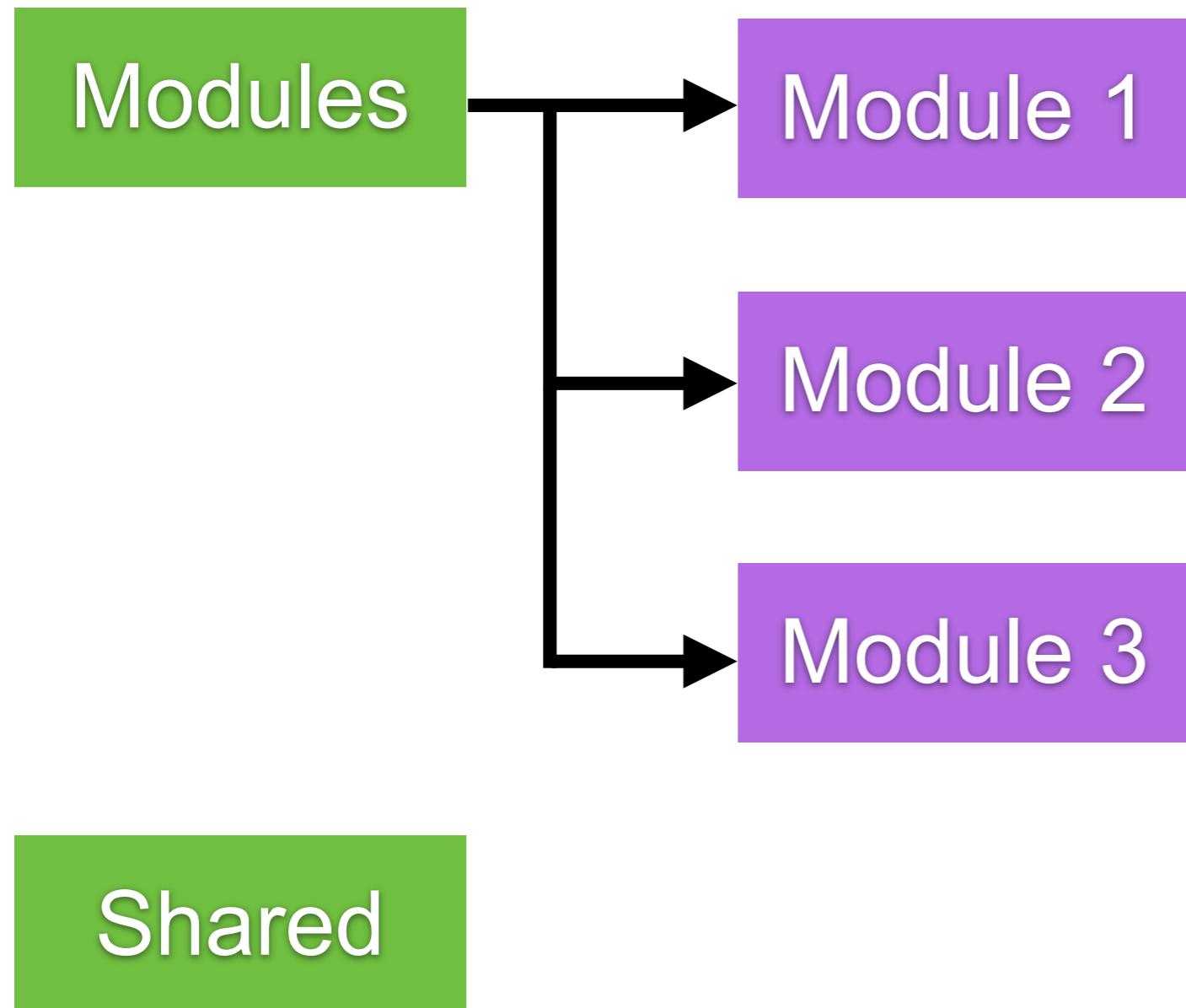
# Better Structure of project



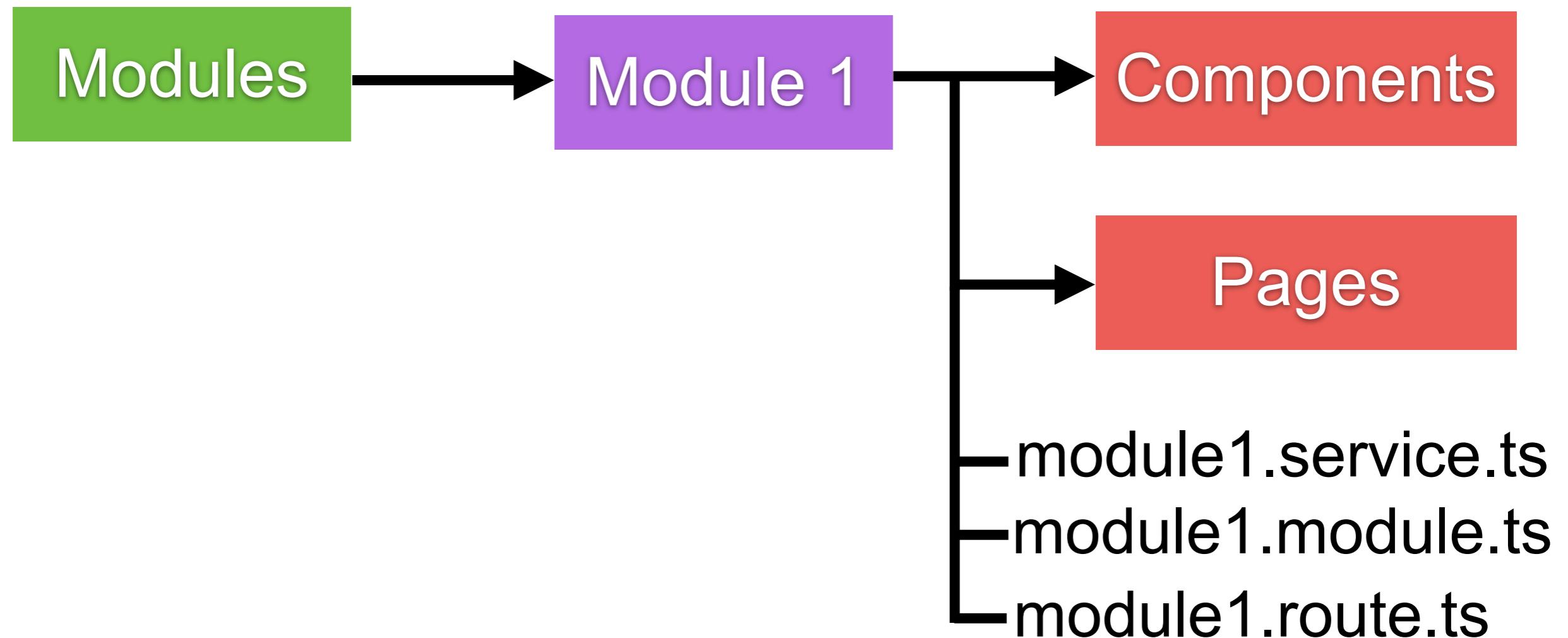
# Large Project Structure



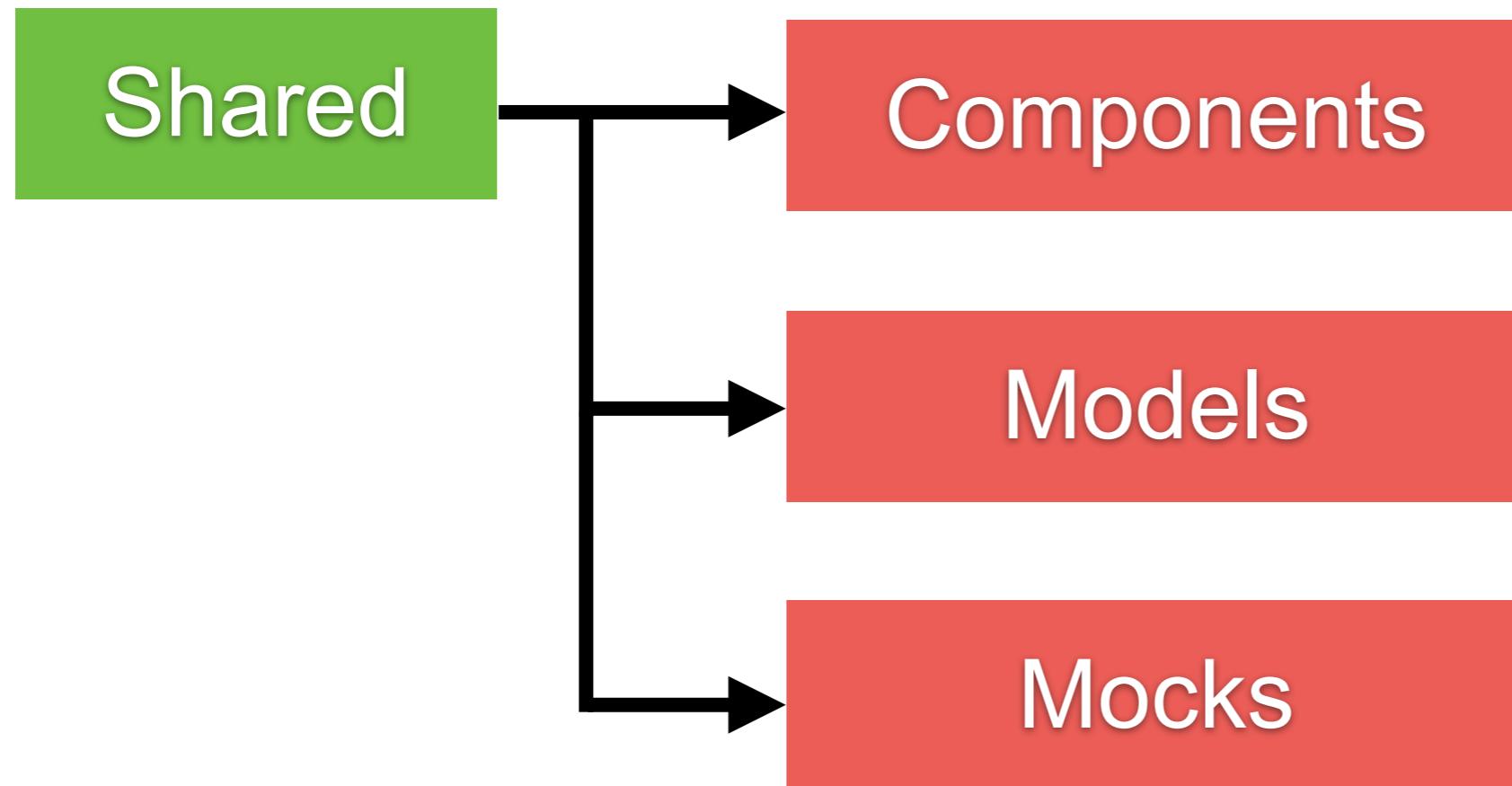
# Better Structure of project



# Module structure



# Shared



# BENWARE

...//...//...//...//...//...//...



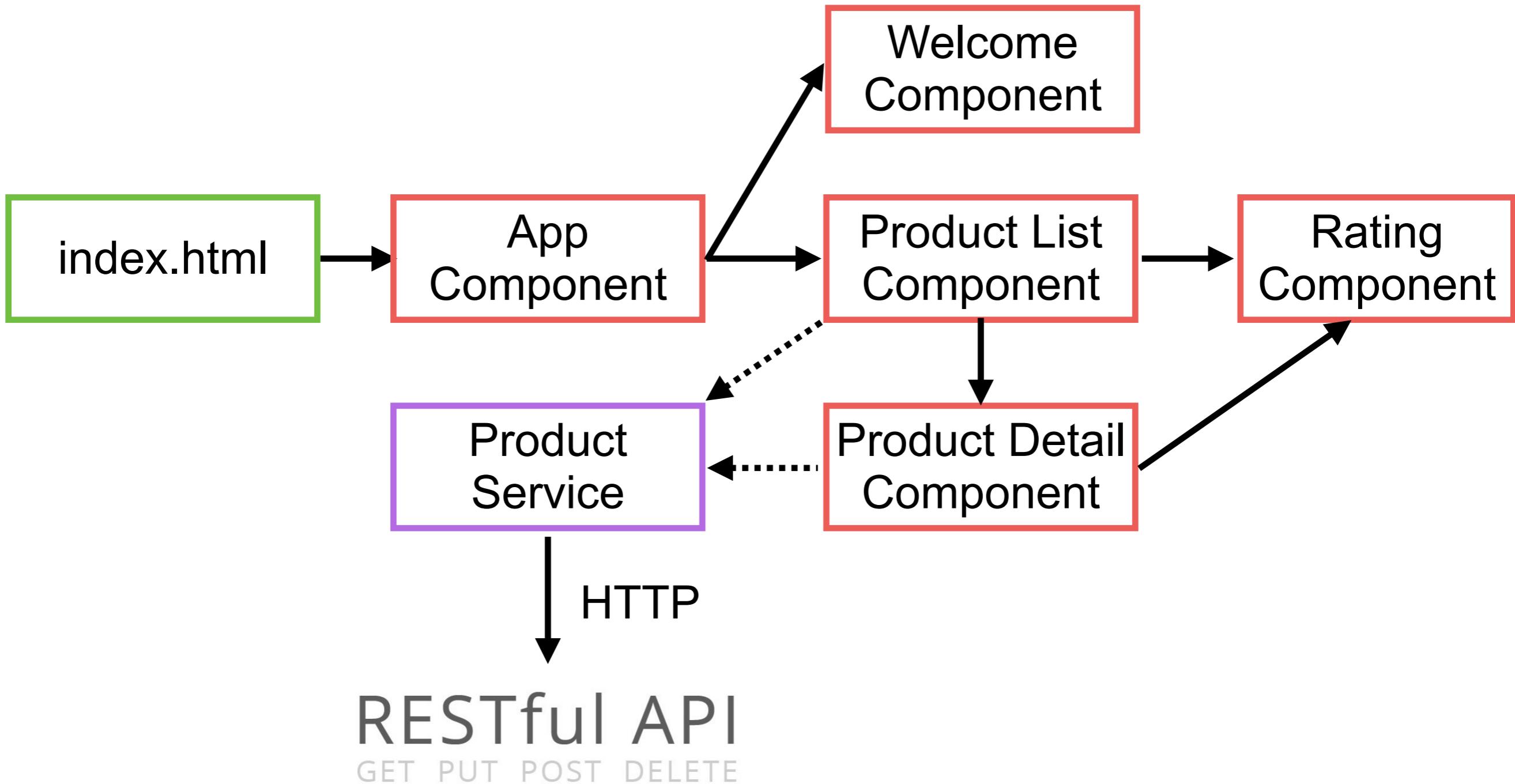
# Environment with build



# Application Architecture



# Application Architecture



# Component

Template

+

Class

+

Metadata

View layout

Code support view

Extra data

Create with HTML

Create with TypeScript

Define decorator  
@Component

Binding and directives

Data and logic



# Component



# Module



# Template



# Template

Building template

Using component as a **directive**

Binding with **Interpolation**

Add logic with directive



# Define template in component

Inline template  
Link template

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

@Component({
  selector: 'app-root',
  template: `<h1>Welcome to {{ title }}!</h1>`,
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'hello';
}
```

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

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



# Create Product List page



# Product List Page



# 1. Install bootstrap and font

```
$npm install bootstrap font-awesome
```



# 2. Setting global stylesheet

Edit in file style.css

```
@import "~bootstrap/dist/css/bootstrap.min.css";
@import "~font-awesome/css/font-awesome.min.css";

div.card-header {
  font-size: large;
}

div.card {
  margin-top: 16px
}

.table {
  margin-top: 16px
}
```

<http://bit.ly/2WTOsRt>



# 3. Create product list

Create directory /app/products

Create file product-list.component.html

\$ng generate component products



# 4. Edit product-list-component.html

## List of Product Page

Filter by:

Choosing ▾

Filtered by ...

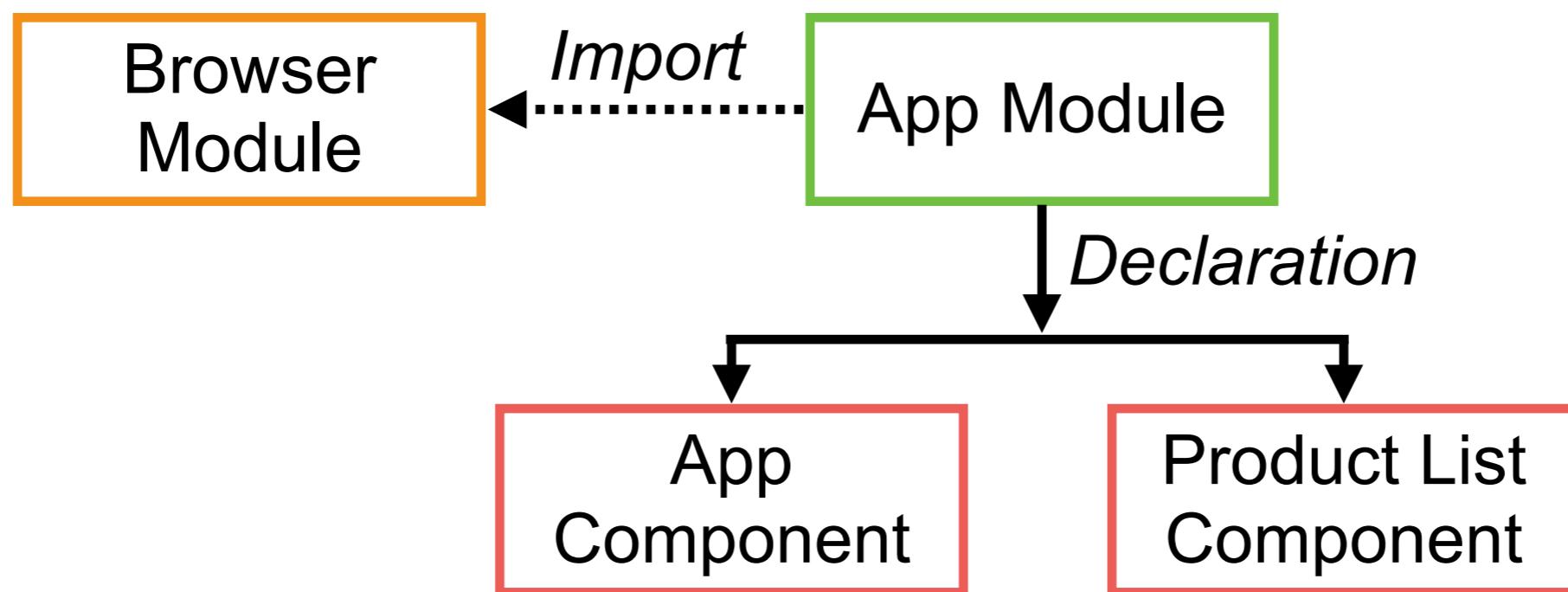
Product Image	Product Code	Product Name	Price	Available	Rating
XXX	0001	Name 01	1000.00	Yes	****
XXX	0002	Name 02	1000.00	Yes	****
XXX	0003	Name 03	1000.00	Yes	****

<http://bit.ly/2WTOsRt>



# 5. Add component to App Module

Edit file /app/app.module.ts



# 5. Add component to App Module

Edit file /app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';
import { ProductListComponent } from './product-list/product-list.component';

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



# 6. Use component as directive

Edit file /app/app.component.html

```
<div style="text-align:center">
  <h1 id="welcome_title">
    Welcome to {{ title }}!
  </h1>
</div>
```

```
<app-product-list></app-product-list>
```



*Component directive*



# What is angular directives ?

Custom HTML element or attribute use to power up and extend HTML



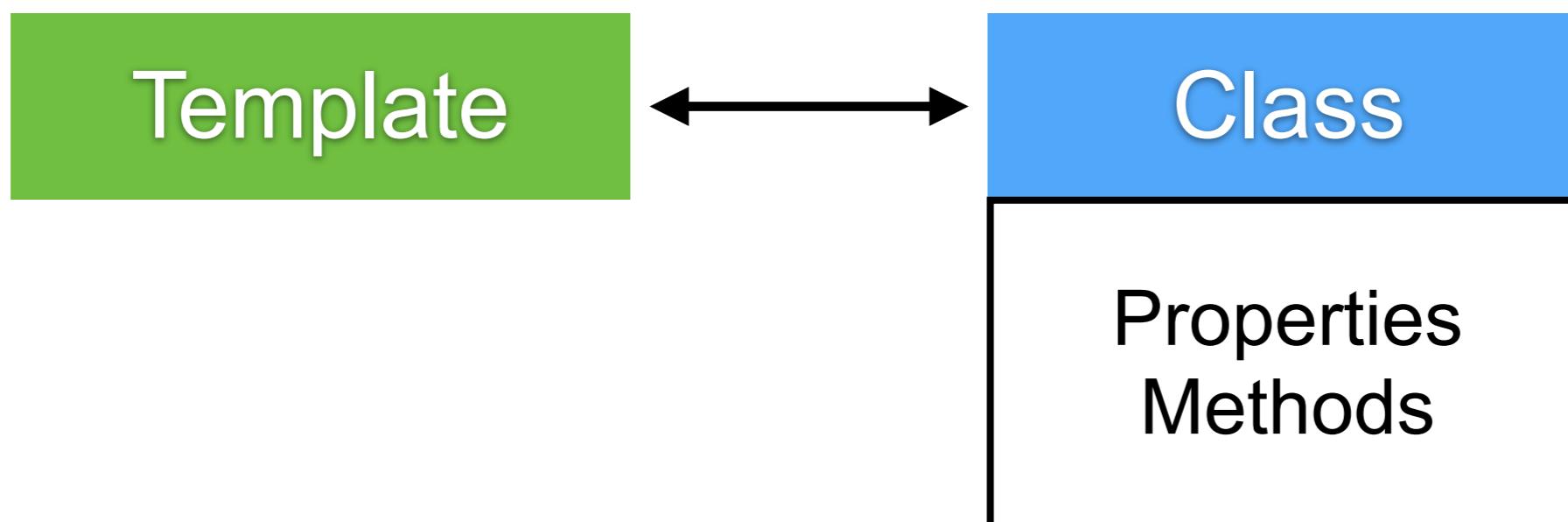
# Types of directives

Component directive  
Structural directive  
Attribute directive



# Data binding in Angular

Coordinate communication between component class and template



# Data binding in Angular

One-way data binding  
Two-way data binding



# Interpolation

One-way data binding

Template

```
<div class='card'>
  <div class='card-header'>
    {{pageName}}
  </div>
```

Class

```
export class ProductListComponent {
  pageName = 'List of Product Page';
  constructor() { }
}
```

*Template expression*



# Structure directive

Build-in directive from Angular

**\*ngIf:** if logic

**\*ngFor:** for loop



# List of product

List of Product Page

Filter by:

Choosing ▾

Filtered by ...

Product Image	Product Code	Product Name	Price	Available	Rating
XXX	0001	Name 01	1000.00	Yes	***
XXX	0002	Name 02	1000.00	Yes	***
XXX	0003	Name 03	1000.00	Yes	***



Show data when found product(s)



# Use \*ngIf to check products

List of Product Page

Filter by:

Choosing ▾

Filtered by ...

Product Image	Product Code	Product Name	Price	Available	Rating
XXX	0001	Name 01	1000.00	Yes	****
XXX	0002	Name 02	1000.00	Yes	****
XXX	0003	Name 03	1000.00	Yes	****



Show data when found product(s)



# Use \*ngIf to check products

Class

```
export class ProductListComponent {  
  pageName = 'List of Product Page';  
  
  products = [  
    {  
      code: '0001',  
      name: 'Product name 1',  
      price: 100  
    },  
    {  
      code: '0002',  
      name: 'Product name 2',  
      price: 200  
    }  
  ];
```

Template

```
<table class='table'>  
  <tr>  
    <td>*ngIf="products && products.length">  
      </td>  
    </tr>
```



# Use \*ngFor to display all products

List of Product Page

Filter by:

Choosing ▾

Filtered by ...

Product Image	Product Code	Product Name	Price	Available	Rating
XXX	0001	Name 01	1000.00	Yes	****
XXX	0002	Name 02	1000.00	Yes	****
XXX	0003	Name 03	1000.00	Yes	****



Show data when found product(s)



# Use \*ngFor to display all products

```
<tbody>
  <tr *ngFor="let product of products">
    <th>
      XXX
    </th>
    <th>{{product.code}}</th>
    <th>{{product.name}}</th>
    <th>{{product.price}}</th>
    <th>Yes</th>
    <th>****</th>
  </tr>
</tbody>
```



# For loops

For ... of  
For ... in

For ... in  
1, 2

```
▼Array(2) ⓘ
  ▼0:
    code: "0001"
    name: "Product name 1"
    price: 100
    ► __proto__: Object
  ▼1:
    code: "0002"
    name: "Product name 2"
    price: 200
    ► __proto__: Object
  length: 2
```

For ... of  
Product 0  
Product 1



# Data binding in Angular

One-way data binding

**Two-way data binding**



# More ..

Property binding

Handling events with event binding

Handling input with 2-way binding



# Property binding

```
<img [src]="product.imageUrl" />
```

*Binding target*

*Binding source*



# Property binding

```
<img  
  [src]="product.imageUrl"  
  [title]="product.name"  
  [style.width.px]="imageWidth"  
/>
```

<https://angular.io/guide/template-syntax>



# List of product

List of Product Page

Filter by:

Filtered by ...

*Handling input*

Product Image

Product Code

Product Name

Price

Available

Rating

XXX

0001

Product name 1

100

Yes

\*\*\*\*

XXX

0002

Product name 2

200

Yes

\*\*\*\*

*Handling events*



# Event binding

Template

```
<button class='btn btn-primary'  
       (click)="toggleToShow()">  
>  
{isShowImage ?  
  'Hide Image' : 'Show Image'}  
</button>
```

Class

```
isShowImage = true;  
  
toggleToShowProductImage() {  
  this.isShowImage = !this.isShowImage;  
}
```



# Two-way binding



<https://angular.io/guide/user-input>



# Two-way binding

## Template

```
<div class='row'>
  <div class='col-md-2'>
    Filter by:
  </div>
  <div class='col-md-4'>
    <input type="text"
      [(ngModel)] = 'filterData'>
  </div>
</div>
<div class='row'>
  <div class='col-md-6'>
    Filtered by {{filterData}}
  </div>
</div>
```

## Class

```
export class ProductListComponent {
  filterData = '';
```



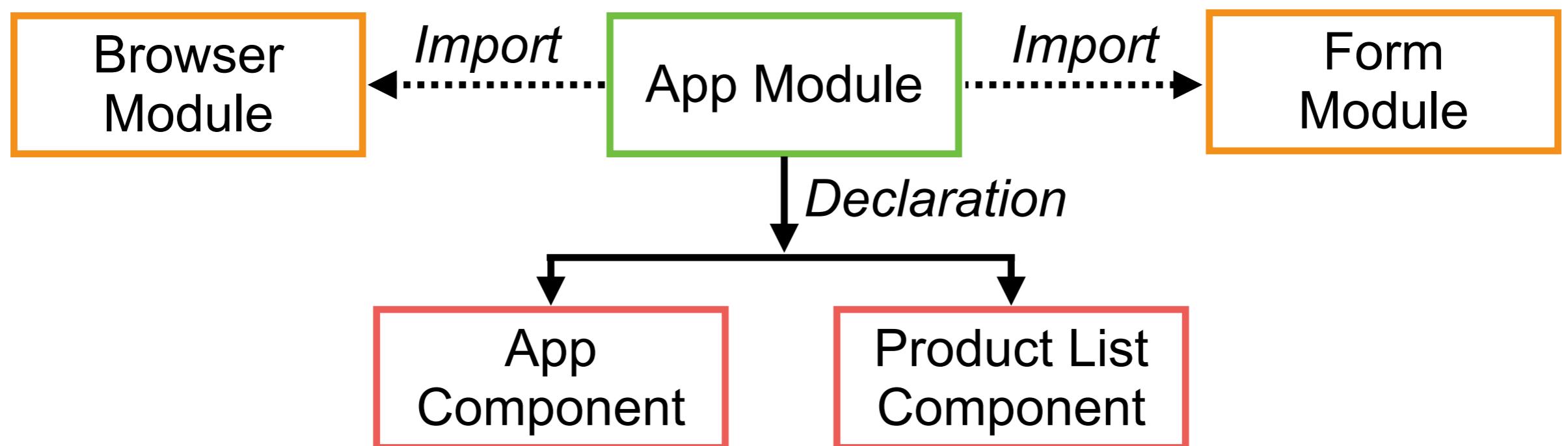
# Error !!

✖ Uncaught Error: Template parse errors:  
Can't bind to 'ngModel' since it isn't a known property of 'input'. ("  
    <div class='col-md-4'>  
        <input type="text"  
            [ERROR ->] [(ngModel)] = 'filterData'>  
        </div>  
    </div>  
"): ng:///AppModule/ProductListComponent.html@10:15  
at syntaxError ([compiler.js:2426](#))



# Add FormModule to AppModule

Edit file /app/app.module.ts



# Add FormModule to AppModule

Edit file /app/app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

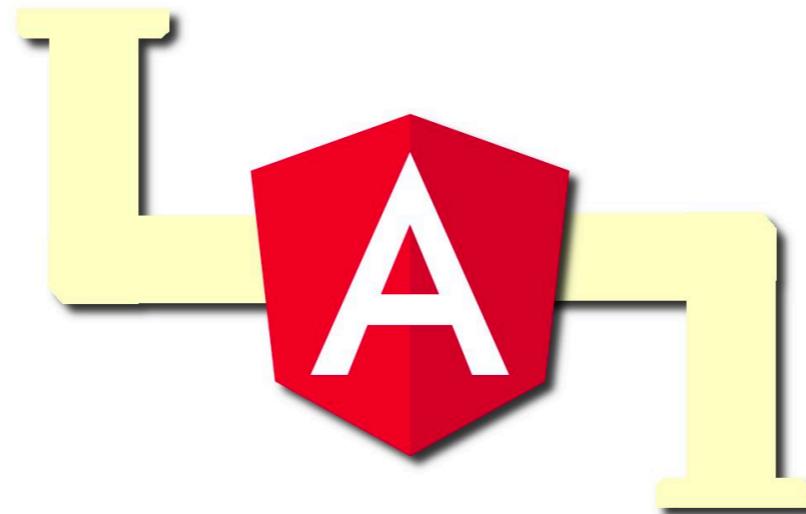
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';
import { ProductListComponent } from './product-list/product-list.component';

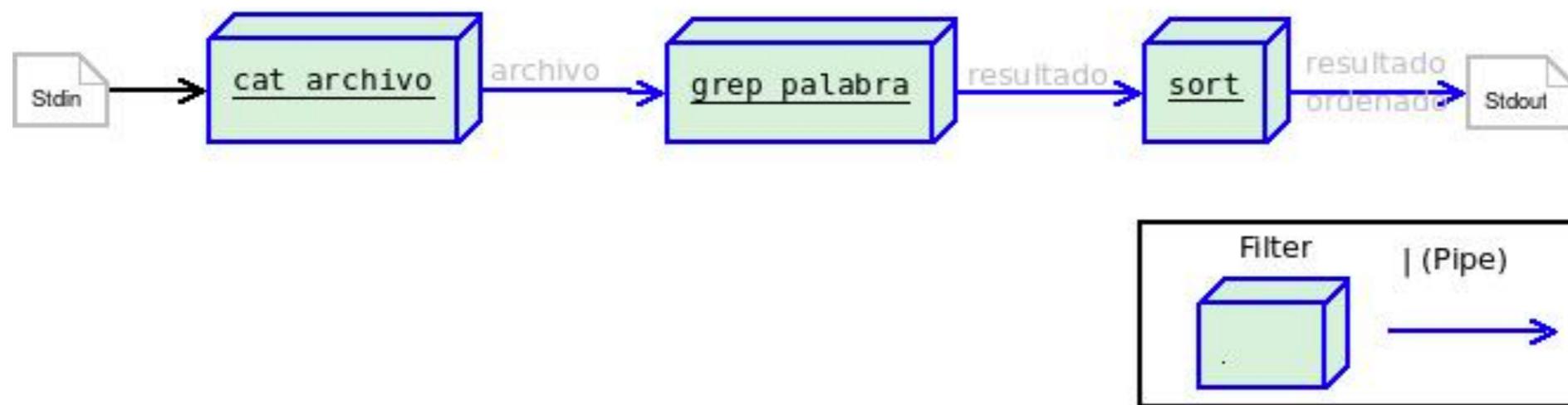
@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



# Transform data with Pipe



```
$cat Arquivo | grep palabra | sort
```



# Angular Pipe

Pipes allow us to change the way to show data and transform data in our template



# Angular Pipe

**Build-in pipe**  
**Parameterize pipe**  
**Chaining pipe**

**Custom pipe**  
**Filter**



# Build-in pipes

Date

Lowercase

Uppercase

Currency

Percent

<https://angular.io/api?type=pipe>



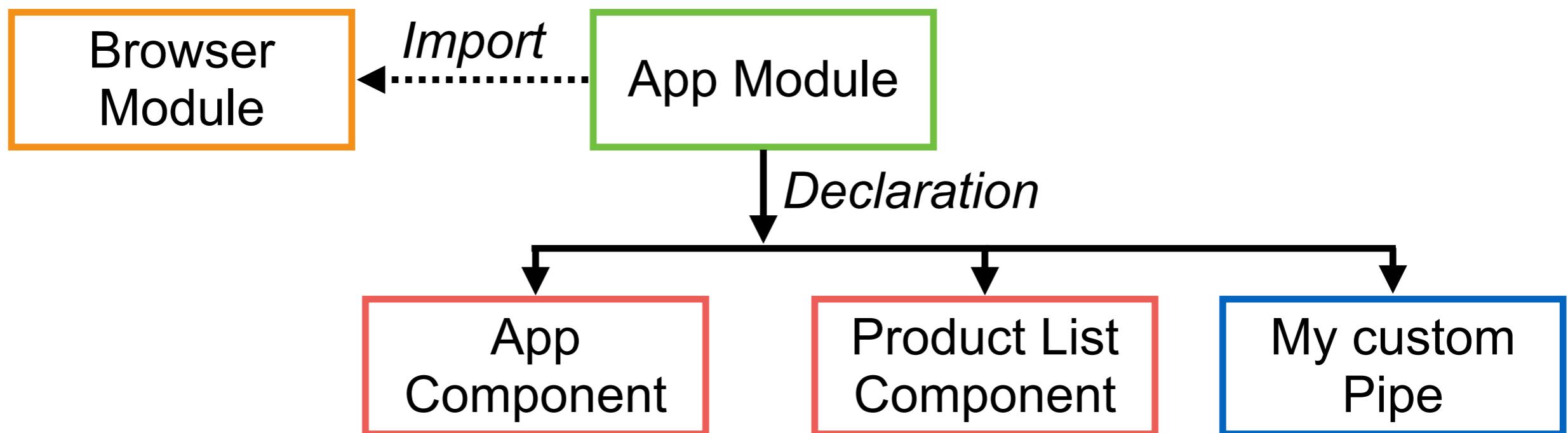
# Build-in pipes

```
<th>{{product.code}}</th>
<th>{{product.name | uppercase}}</th>
<th>{{product.price | currency: 'THB' : 'symbol': '1.2'}}</th>
<th>Yes</th>
<th>****</th>
```



# Custom pipes

\$ng generate pipe <pipe name>



# Create custom pipes

\$ng generate pipe ReplaceWithDash

replace-with-dash.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'replaceWithDash'
})
export class ReplaceWithDashPipe implements PipeTransform {

  transform(value: string, character: string): any {
    return value.replace(character, '-');
  }
}
```



# Add pipe in app module

\$ng generate pipe ReplacewithDash

app.module.ts

```
import { ReplaceWithDashPipe } from './replace-with-dash.pipe';

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
    ReplaceWithDashPipe,
  ])
})
```



# Using pipe in template

product-list.component.html

```
<th>{{product.code | replaceWithDash: '-'}}</th>
```

List of Product Page		
Filter by:		
Filtered by		
Hide Image	Product Code	Product Name
	AA-0001	PRODUCT NAME 1
	BB-0002	PRODUCT NAME 2



# Improve your component



# Improve your component

Strong type and interface  
Encapsulate style  
Component Life cycle



# Strong type

TypeScript is static type language  
Cannot change type after assigned

```
16  pageName = 'List of Product Page';
17  imageWidth = 50;
18  imageWidth = 'New';
```

TS2300: Duplicate identifier 'imageWidth'.

TS2717: Subsequent property declarations must have the same type. Property 'imageWidth' must be of type 'number', but here has type 'string'.

[Remove unused field 'imageWidth'](#) [More actions...](#)

22 **code: '0001',**

<https://www.typescriptlang.org/docs/handbook/basic-types.html>



# Improve product-list.componet.ts

*Any for unknown type or dynamic content*

```
products: any[] = [
  {
    code: '0001',
    name: 'Product name 1',
    price: 100,
    imageUrl: 'https://th-live-01.sstatic.net/c
  },
  {
    code: '0002',
    name: 'Product name 2',
    price: 2000,
    imageUrl: 'https://th-live-01.sstatic.net/c
  }
];
```



# Interface is a specification

Using **interface** instead any type  
Use interface as data type

```
export interface ProductDataModel {  
  
    code: string;  
    name: string;  
    price: number;  
    imageUrl: string;  
  
}
```

<https://www.typescriptlang.org/docs/handbook/interfaces.html>



# Use interface as data type

Edit file product-list.component.ts

```
import {ProductDataModel} from './product';

export class ProductListComponent {

  products: ProductDataModel[] = [
    {
      code: '0001',
      name: 'Product name 1',
      price: 100,
      imageUrl: '',
    }
  ];
}
```



# Encapsulate style in component

## Styles

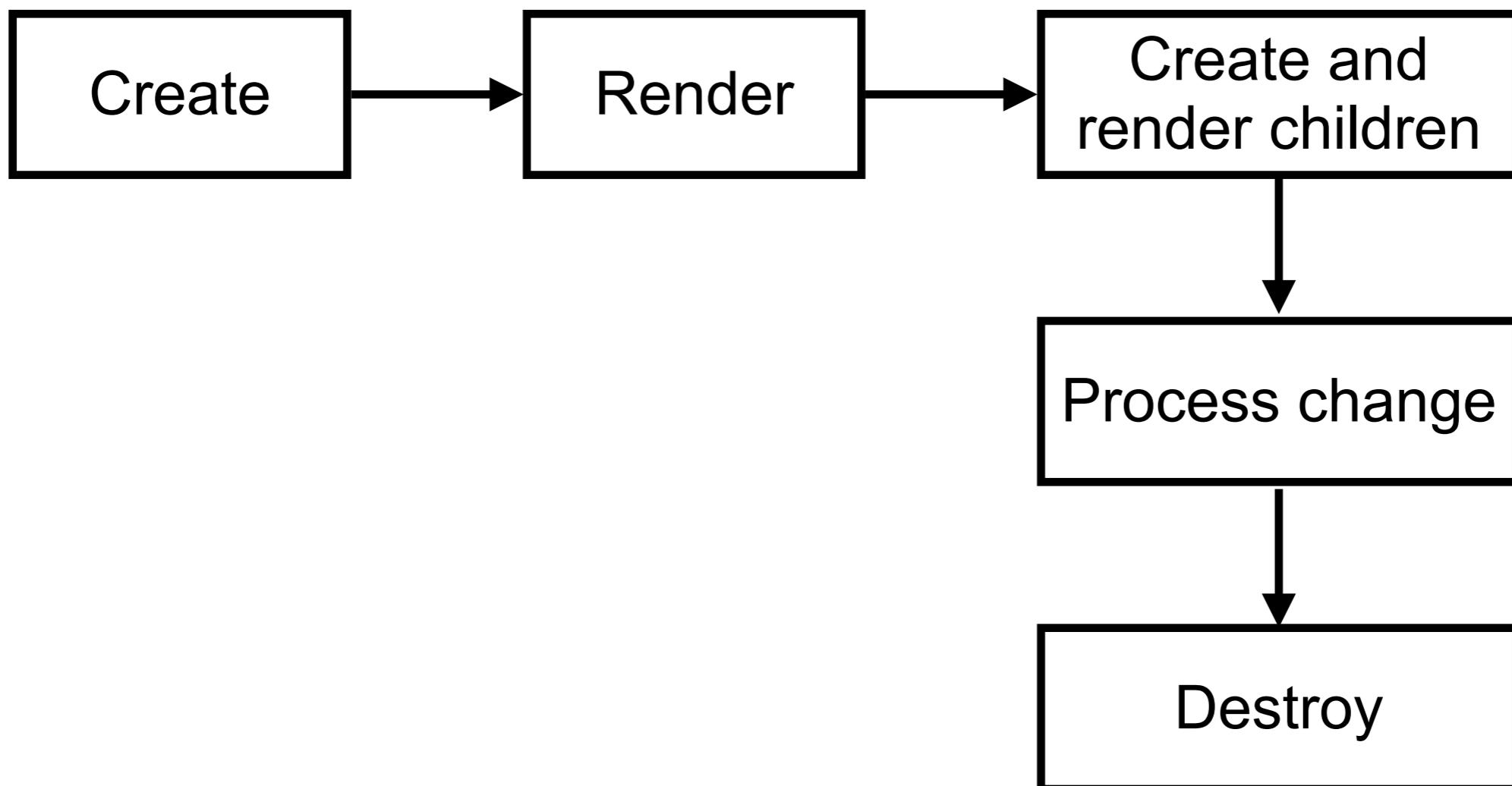
```
@Component({  
  selector: 'app-product-list',  
  templateUrl: './product-list.component.html',  
  
  styles: ['thead {background: #337AB7; color: white}']  
})
```

## StyleUrls

```
@Component({  
  selector: 'app-product-list',  
  templateUrl: './product-list.component.html',  
  
  styleUrls: ['./product-list.component.css']  
})
```



# Component life cycle



# Component life cycle hooks

Method	Description
OnInit()	Perform component initialization Retrive data
OnChanges()	Perform action after properties are changed
OnDestroy()	Perform cleanup

<https://angular.io/guide/lifecycle-hooks>



# Using life cycle hook

## 1. Import hook from @angular core

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

```
export class ProductListComponent implements OnInit {
```

```
  constructor() {  
    console.log(this.products);  
  }
```

## 2. Implement Hooks interface

```
  ngOnInit(): void {  
    console.log('Called ngOnInit');  
  }
```

```
}
```

## 3. Implement methods from hook



# Filter data in product list



# Filter data in product list

List of Product Page

Filter by:

Filtered by ...

*1. Key-in data*

Product Image

Product Code

Product Name

Price

Available

Rating

XXX

0001

Product name 1

100

Yes

\*\*\*\*

XXX

0002

Product name 2

200

Yes

\*\*\*\*

*2. Filter and display data from criteria*



# Filter data using Pipe !!

“Angular not offer such pipe Filtering and Sorting because they perform poorly and prevent aggressive minification”

*Filtering and sorting are expensive operation*

<https://angular.io/guide/pipes#no-filter-pipe>



# How to filtering and sorting ?

“Moving filtering and sorting login into component”

<https://angular.io/guide/pipes#no-filter-pipe>



# Filter data

Create setter/getter method in component

Product-list.component.ts

```
export class ProductListComponent implements OnInit {  
  
  private _filterData = '';  
  
  set filterData(value: string) {  
    this._filterData = value;  
  }  
  
  get filterData(): string {  
    return this._filterData;  
  }  
}
```



# Filter data

## Filter data by input from HTML

Product-list.component.ts

```
export class ProductListComponent implements OnInit {  
  
    private _filterData: string;  
    filteredProducts: ProductDataModel[];  
  
    set filterData(value: string) {  
        this._filterData = value;  
        this.filteredProducts =  
            this.filterData ?  
                this.performFilter(this.filterData) : this.products;  
    }  
  
    get filterData(): string {  
        return this._filterData;  
    }  
}
```

*Create filteredProduct to keep result*



# Filter data

Create function performFilter()

Product-list.component.ts

```
private performFilter(filterBy: string) {  
    filterBy = filterBy.toLocaleLowerCase();  
  
    return this.products.filter(  
        (p: ProductDataModel) =>  
            p.name.toLocaleLowerCase().indexOf(filterBy) !== -1);  
}
```



*Filter data from product name*

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/filter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)



# Change in template

Product-list.component.html

```
<tbody>
  <tr *ngFor="let product of filteredProducts">
    <th>
      <img *ngIf="isShowImage"
        [src]="product.imageUrl"
        [title]="product.name"
        [style.width.px]="'imageWidth'"
      />
    </th>
```



# Try to run

List of Product Page

Filter by:

Filtered by

<a href="#">Hide Image</a>	Product Code	Product Name	Price	Available	Rating
----------------------------	--------------	--------------	-------	-----------	--------



Sorry,  
No Data Found !



# Filter data

Initial data in constructor

Product-list.component.ts

```
export class ProductListComponent implements OnInit {  
  
  constructor() {  
    console.log(this.products);  
    this.filteredProducts = this.products;  
  }  
  
}  
}
```

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/filter](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter)



# Try to run

## List of Product Page

Filter by:

Filtered by

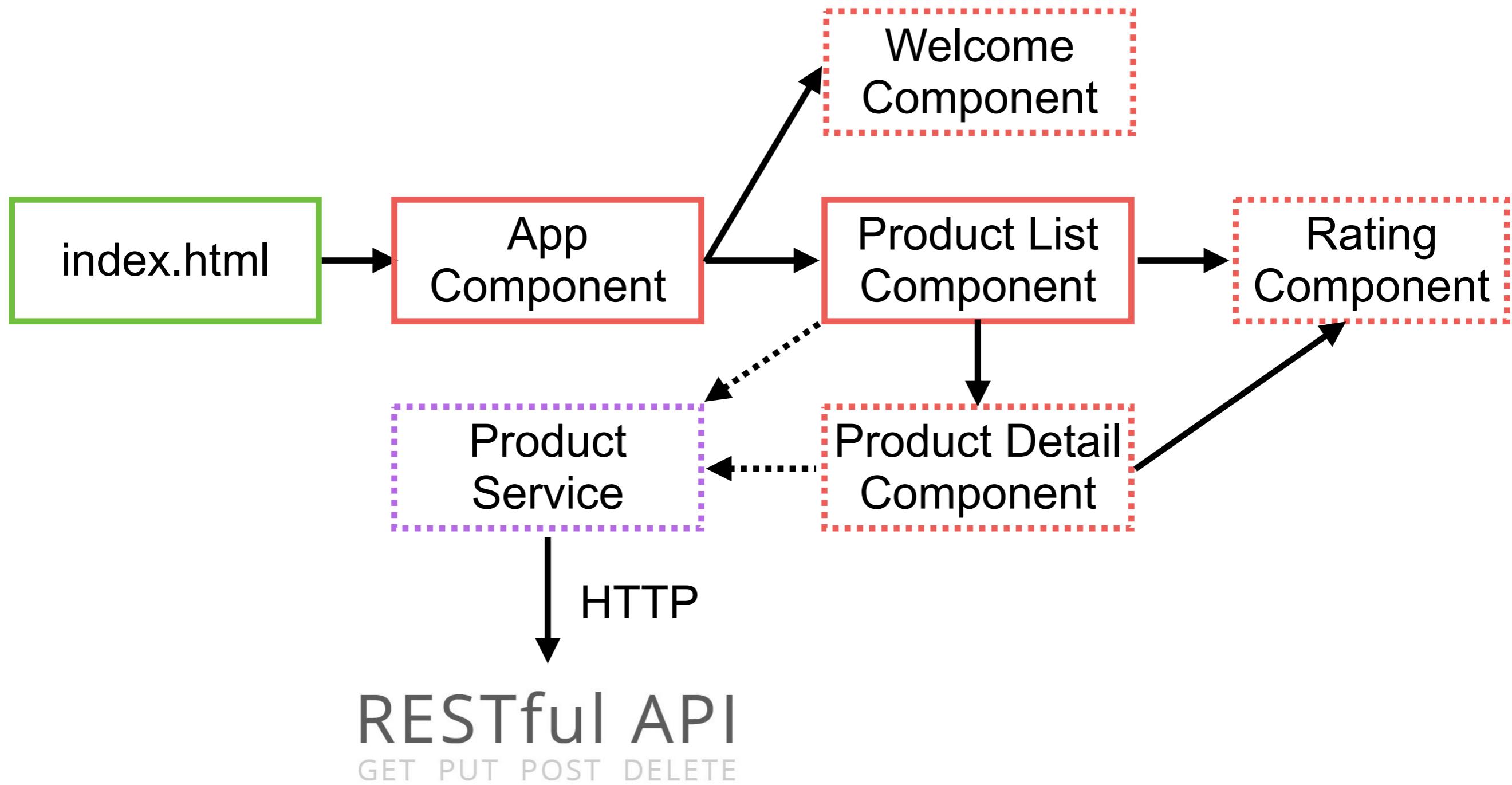
<a href="#">Hide Image</a>	<a href="#">Product Code</a>	<a href="#">Product Name</a>	<a href="#">Price</a>	<a href="#">Available</a>	<a href="#">Rating</a>
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	****
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	****



# Nested components



# Application Architecture



# Rating component

## List of Product Page

Filter by:

Filtered by

<button>Hide Image</button>	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	****
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	****



# Nested component

Building component

Using component

Passing data to component (@Input)

Raising event from component (@Output)



# Nested component

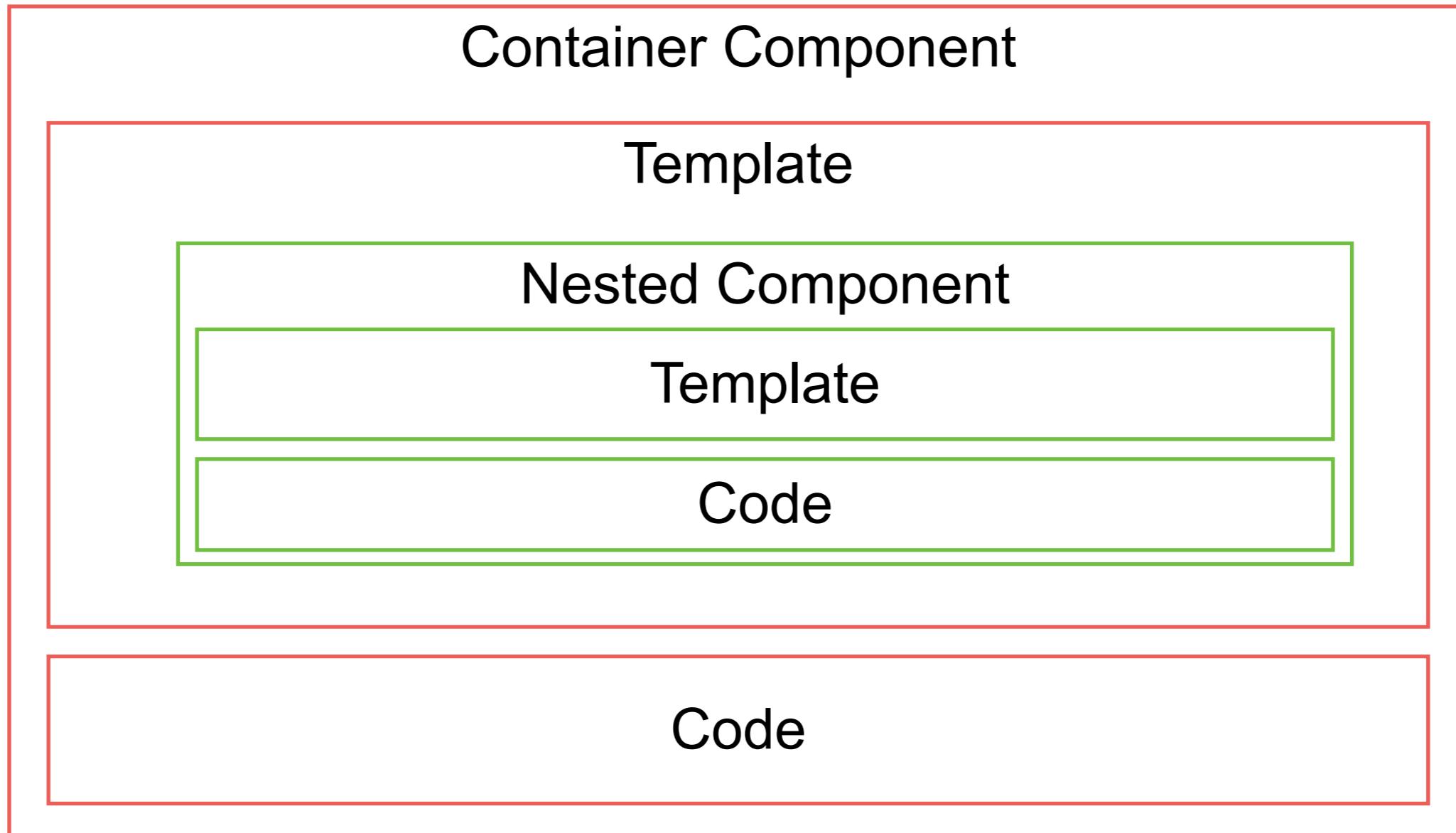
Container Component

Template

Code

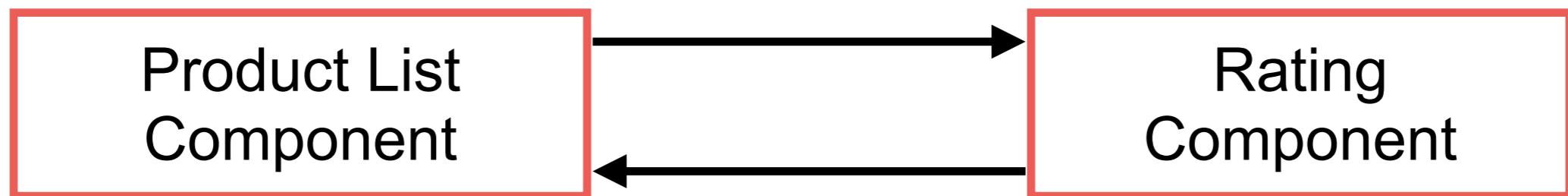


# Nested component



# Nested component

Passing data using @Input



Raising event using @Output

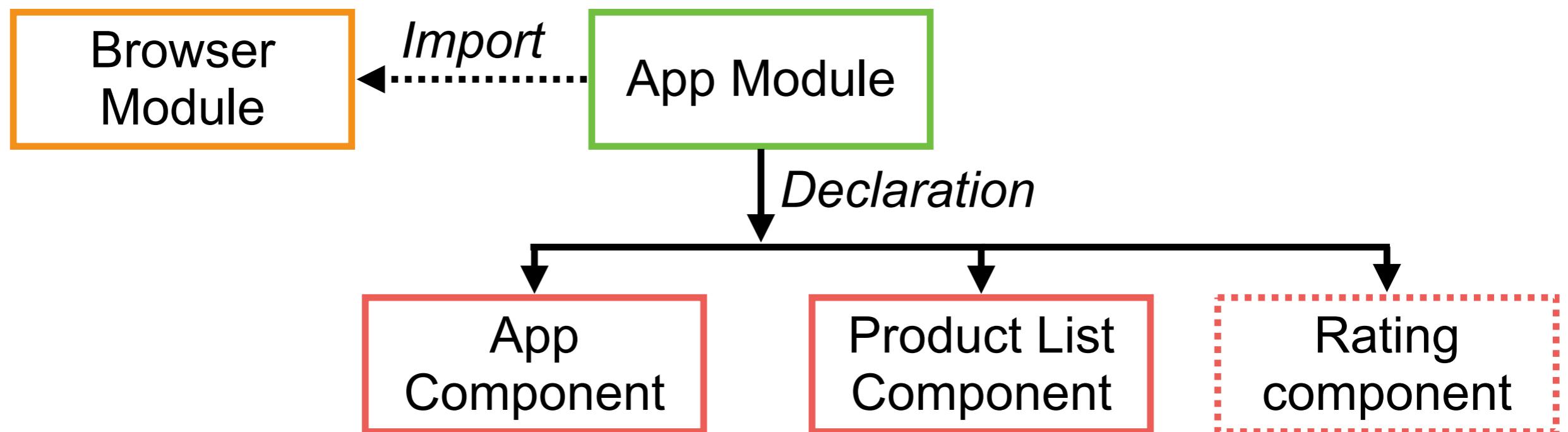


# Building component



# 1. Create rating component

\$ng generate component rating



# 1. Create rating component

\$ng generate component rating

Rating.component.ts

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

@Component({
  selector: 'app-rating' → Directive name of rating component
  templateUrl: './rating.component.html',
  styleUrls: ['./rating.component.css']
})
export class RatingComponent implements OnInit {

}
```



# 2. Add rating component in module

/app/app.module.ts

App.module.ts

```
import { RatingComponent } from './rating/rating.component';

@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
    ReplaceWithDashPipe,
    RatingComponent,
  ],
})
```

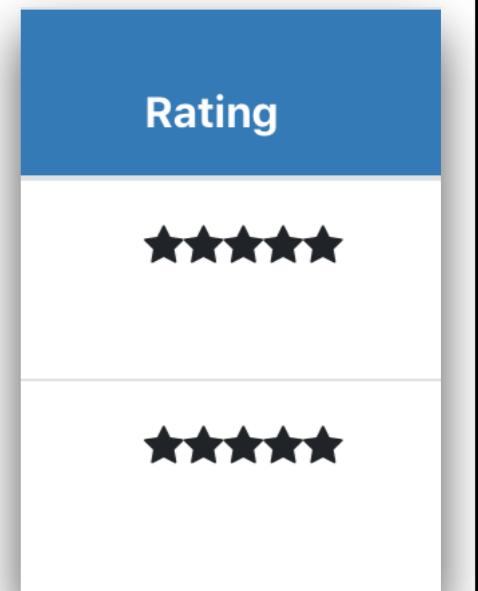


# 3. Edit template of rating

/app/rating/rating.component.html

Rating.component.html

```
<div>
  <div style="width: 75px">
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
    <span class="fa fa-star"></span>
  </div>
</div>
```



# 4. Using rating in product list

/app/product-list/product-list.component.html

Product-list.component.html

```
<tbody>
  <tr *ngFor="let product of filteredProducts">
    <th>
      <img *ngIf="isShowImage"
        [src]="product.imageUrl"
        [title]="product.name"
        [style.width.px]="'imageWidth'"
      />
    </th>
    <th>{{product.code | replaceWithDash: '-'}}</th>
    <th>{{product.name | uppercase}}</th>
    <th>{{product.price | currency: 'THB ' : 'symbol': '1.2'}}</th>
    <th>Yes</th>
    <th>
      <app-rating></app-rating>
    </th>
  </tr>
</tbody>
```



# 4. Using rating in product list

List of Product Page

Filter by:

Filtered by

<a href="#">Hide Image</a>	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	

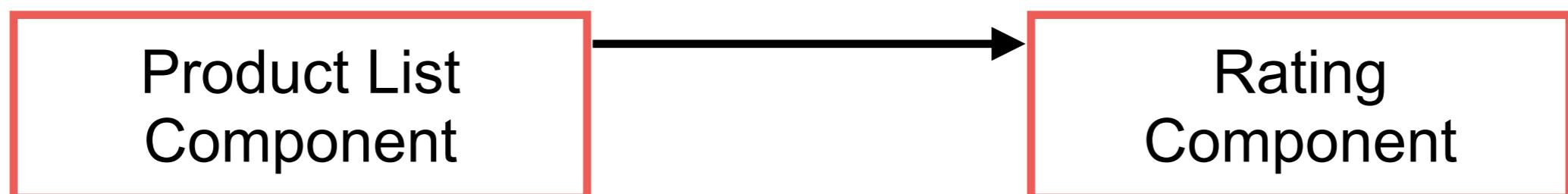


# Passing data to component



# Passing data to component

Passing data using @Input



# Passing data to component

Product-list.component.html

```
<th>
  <app-rating [rating]="product.rating"></app-rating>
</th>
```

Rating.component.ts

```
export class RatingComponent implements OnInit {
  @Input() rating: number;
}
```



# Detect data change in component

Passing data using `@Input`



# Detect data change in component

## Rating.component.html

```
<div  
  [style.width.px] = "starWidth"  
  style="overflow: hidden;"  
>  
  <div style="width: 75px"></div>  
</div>
```

## Rating.component.ts

```
import {Component, Input, OnChanges} from '@angular/core';  
  
export class RatingComponent implements OnChanges {  
  @Input() rating: number;  
  private starWidth: number;  
  
  ngOnChanges(): void {  
    this.starWidth = this.rating * 75 / 5;  
  }  
}
```



# Result

## List of Product Page

Filter by:

Filtered by

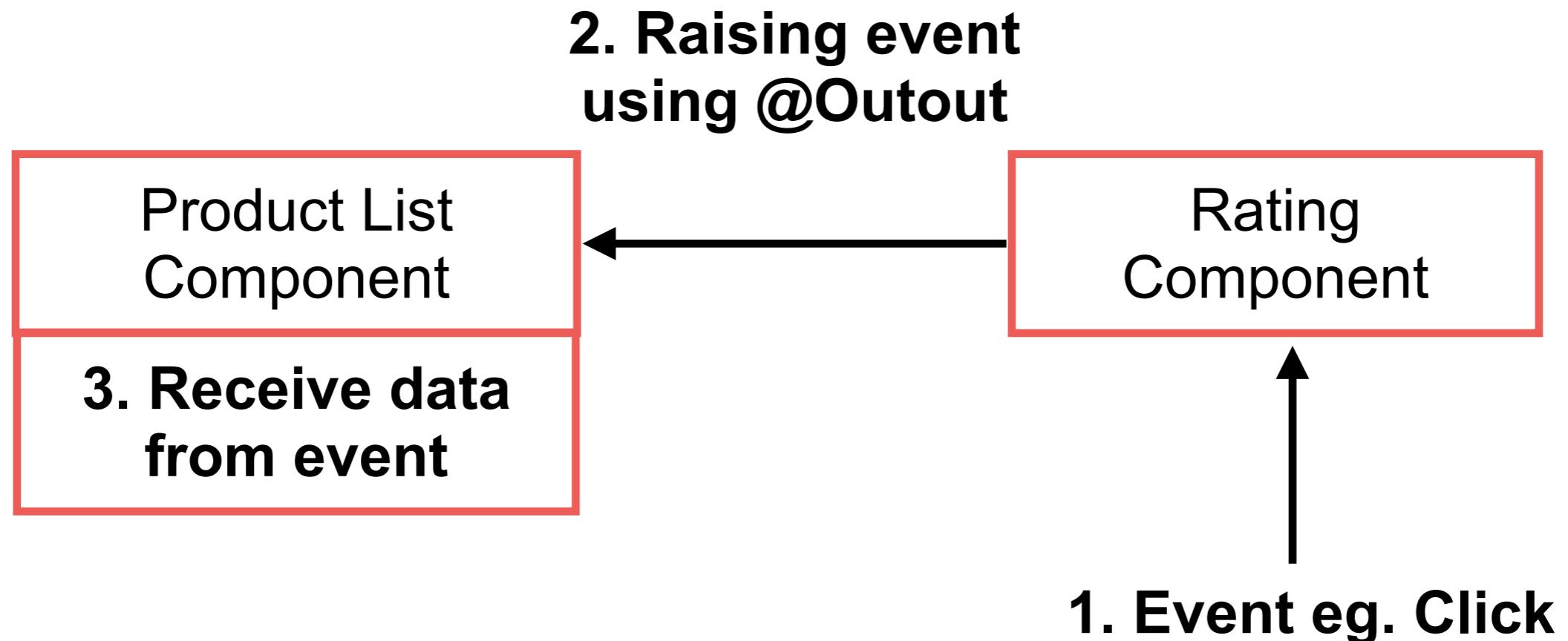
<a href="#">Hide Image</a>	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	
	BB-0003	PRODUCT NAME 3	THB 2,000.00	Yes	



# Raising event from component



# Raising event from component



# 1. Event handling on rating

Rating.component.html

```
<div  
  [style.width.px] = "starWidth"  
  style="overflow: hidden;"  
  (click)="onClickRating()">  
</div>
```

Rating.component.ts

```
import {Component, Input, OnChanges} from '@angular/core';  
  
export class RatingComponent implements OnChanges {  
  
  onClickRating() {  
    console.log('Click on rating');  
  }  
  
}
```



## 2. Raising event with @Output

Product-list.component.html

```
<app-rating  
  [rating] = "product.rating"  
  (ratingClicked) = "onRatingClicked($event)">  
</app-rating>
```

Rating.component.ts

```
export class RatingComponent implements OnChanges {  
  
  @Output() ratingClicked: EventEmitter<string>  
    = new EventEmitter<string>();  
  
  onClickRating() {  
    console.log('Click on rating');  
    this.ratingClicked.emit(`Rating ${this.rating} was clicked`);  
  }  
}
```



# 3. Receive data from event

Product-list.component.html

```
<app-rating  
  [rating] = "product.rating"  
  (ratingClicked) = "onRatingClicked($event)">  
</app-rating>
```

Product-list.component.ts

```
export class ProductListComponent implements OnInit {  
  
  onRatingClicked(message: string) {  
    this.pageName = `List of Product Page :: ${message}`;  
  }  
}
```



# Navigation and Routing



# Navigation and Routing

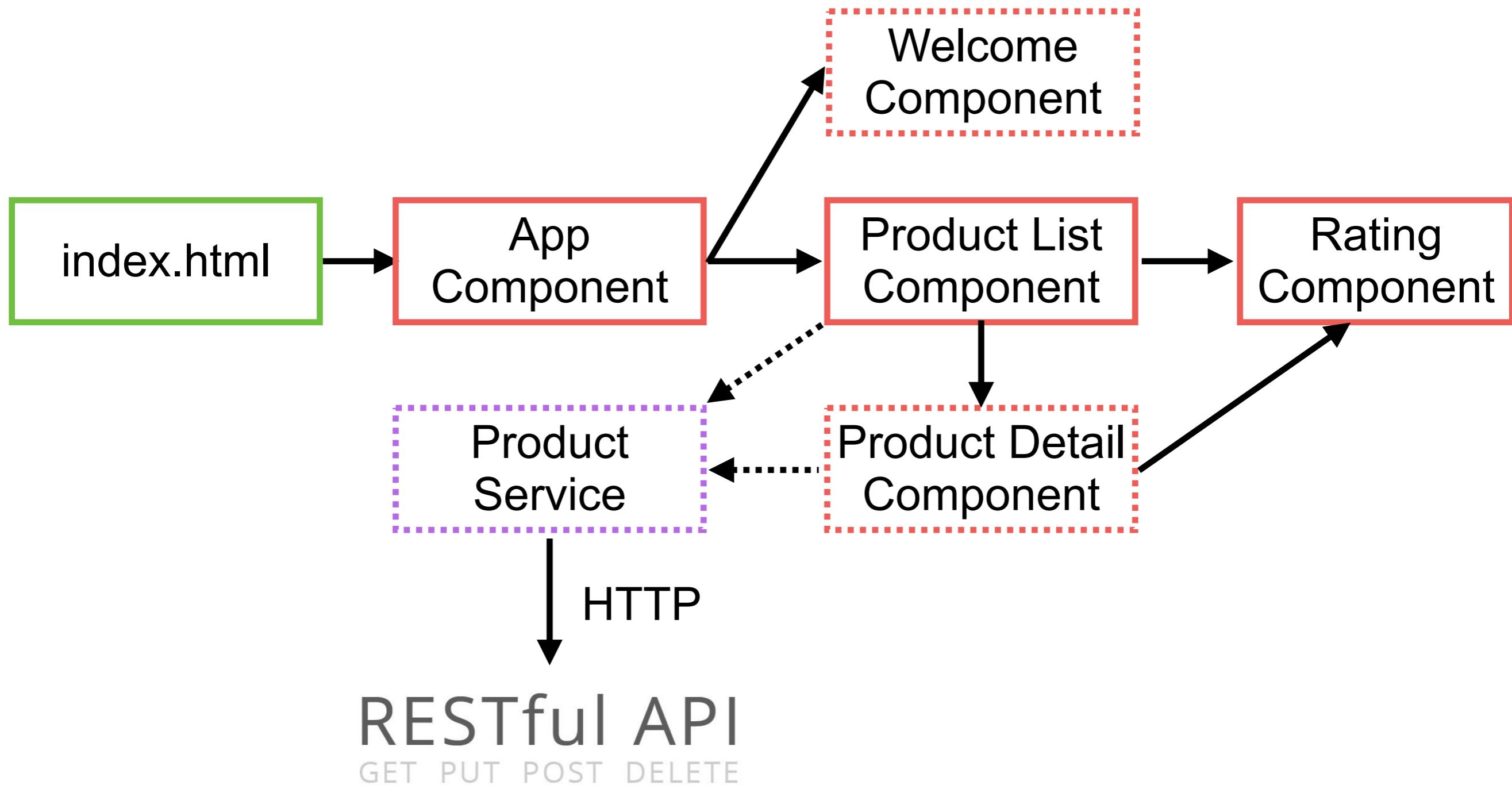
Configure route in each component

Define actions and route

Display routing in each component's view



# Application Architecture



# Create angular router

Route	Description
/welcome	Show welcome page
<b>/products</b>	Show list of product page
/product/:id	Show product detail page
“”	Default page
**	For other path (Page not found)



# Configure router in app module

Add RouterModule and configure routes

app.module.ts

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

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: '', redirectTo: 'products', pathMatch: 'full' },
      { path: '**', redirectTo: 'products', pathMatch: 'full' }
    ]),
  ],
})
```



# RouterModule

Register router service  
Declare router directives  
Expose configured routes



# Using router as directive

Add configured routes in template

app.component.html

```
<nav class='navbar navbar-expand navbar-light bg-light'>
  <a class='navbar-brand'>{{pageTitle}}</a>
  <ul class='nav nav-pills'>
    <li><a class='nav-link' routerLinkActive='active'
           [routerLink]="/welcome">Home</a></li>
    <li><a class='nav-link' routerLinkActive='active'
           [routerLink]="/products">Product List</a></li>
  </ul>

</nav>
<div class='container-fluid'>
  <router-outlet></router-outlet>
</div>
```

*Use configured routes*

*Show content from component's view*



# Result

Home Product List

List of Product Page

Filter by:

Filtered by

<button>Hide Image</button>	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	★★★★★
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	★★
	BB-0003	PRODUCT NAME 3	THB 2,000.00	Yes	★★★½



# Working with route

Passing parameter

Activate route with code

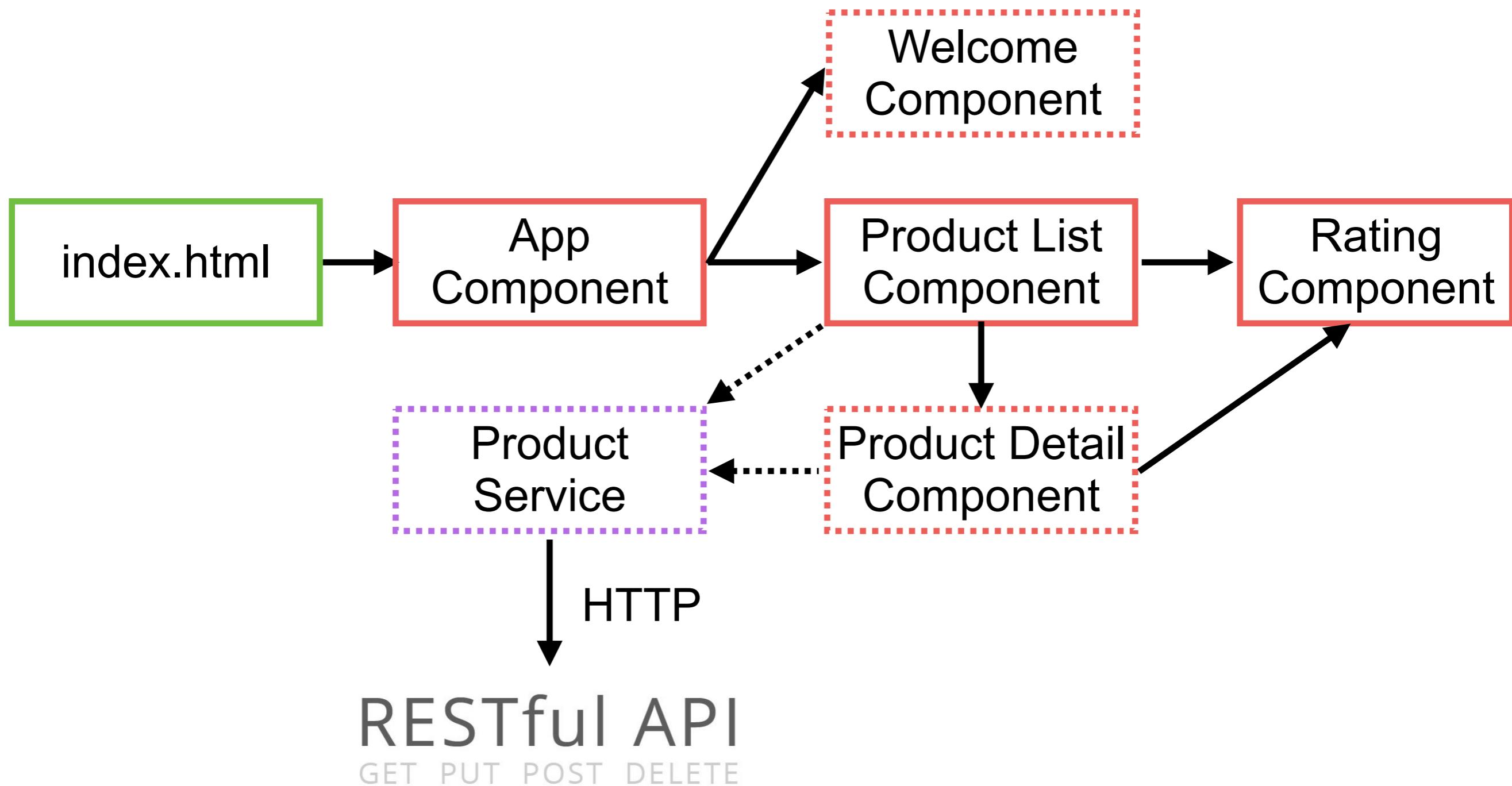
Protect routes with **guards**



# Passing parameters with route



# Application Architecture



# Product detail

URL = /product/:id



*Paramater*



# Create product detail component

```
$ng generate component productDetail
```

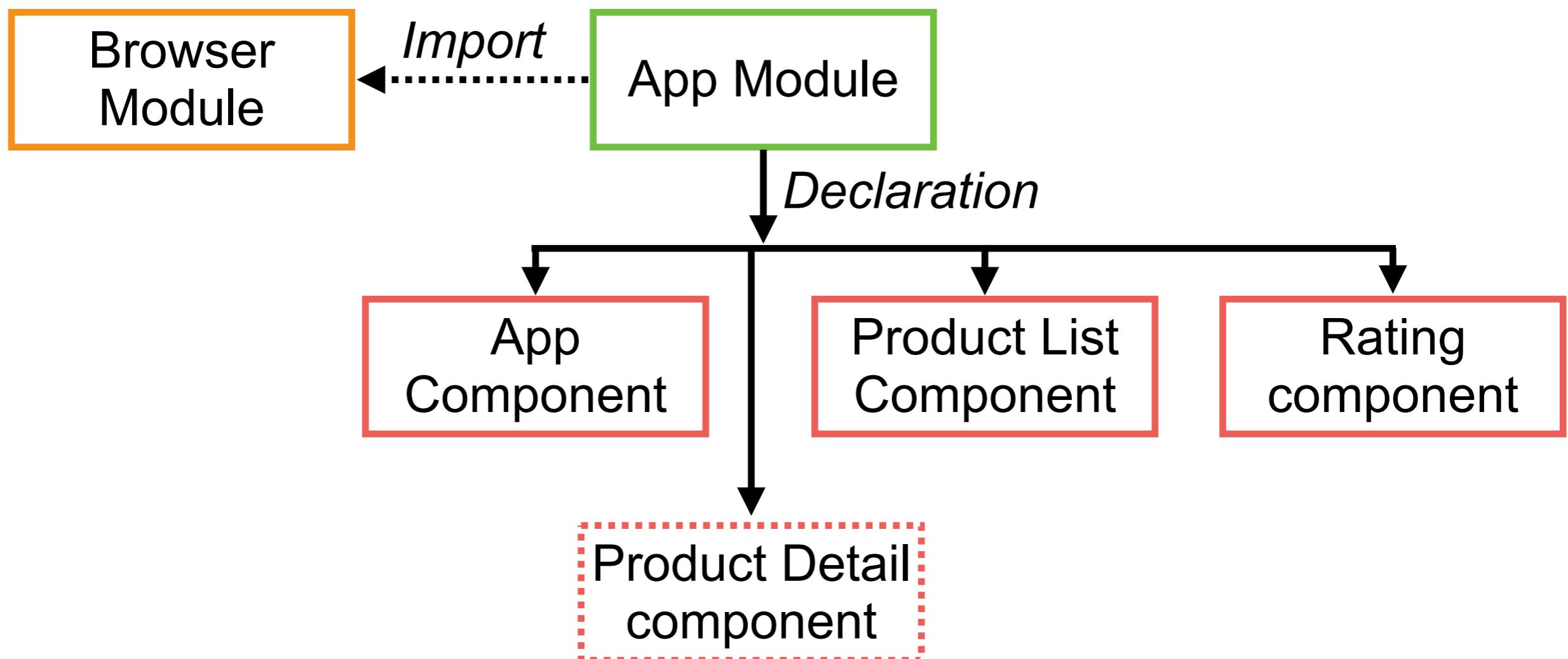
```
$ng generate component product/productDetail --flat
```

*Generate component with existing directory*



# Create product detail component

\$ng generate component productDetail



# Configure router in app module

Add RouterModule and configure routes

app.module.ts

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

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'product/:id', component: ProductDetailComponent },
      { path: '', redirectTo: 'products', pathMatch: 'full' },
      { path: '**', redirectTo: 'products', pathMatch: 'full' }
    ]),
  ],
})
```



# Link to product detail page

List of Product Page					
Filter by: <input type="text"/>					
Filtered by					
Hide Image	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	★★★★★
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	★★
	BB-0003	PRODUCT NAME 3	THB 2,000.00	Yes	★★★



# Using router as directive

Add configured routes in template

product-list.component.html

```
<a  
  [routerLink]=“[‘/product’, product.id]”>  
  {{product.name}}  
</a>
```

product-detail.component.ts

```
import {ActivatedRoute} from '@angular/router';  
  
export class ProductDetailComponent {  
  
  constructor(private route: ActivatedRoute) {  
    console.log(route.snapshot.paramMap.get('id'));  
  }  
}  
  
Read parameter from route
```



# Result

List of Product Page

Filter by:

Filtered by

<button>Hide Image</button>	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	★★★★★
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	★★
	BB-0003	PRODUCT NAME 3	THB 2,000.00	Yes	★★★

*Link to /product/:id*



# Show product detail

## Mock data to show product detail

product-detail.component.ts

```
import {ProductDataModel} from './product-list/product';

export class ProductDetailComponent {
  pageName = 'Product detail';
  product: ProductDataModel;

  constructor(private route: ActivatedRoute) {
    console.log(route.snapshot.paramMap.get('id'));
    this.product = {
      code: 'AA 0001',
      name: 'Product name 1',
      price: 100,
      rating: 5,
      imageUrl: '',
    };
  }
}
```



# Activate route with code



# Product detail

When click back button should show List product

Home   Product List

Product detail: Product name 1

Code:	AA-0001
Name:	Product name 1
Price:	100
Rating:	★★★★★



[Back](#) *Back to List of product page*



# Use route with code

Add configured routes in template

product-detail.component.ts

```
import {ActivatedRoute, Router} from '@angular/router';

export class ProductDetailComponent {

  constructor(private route: ActivatedRoute,
            private router: Router) {
  }

  onBack() {
    this.router.navigate(['/products']);
  }
}
```



# Protect route with guard



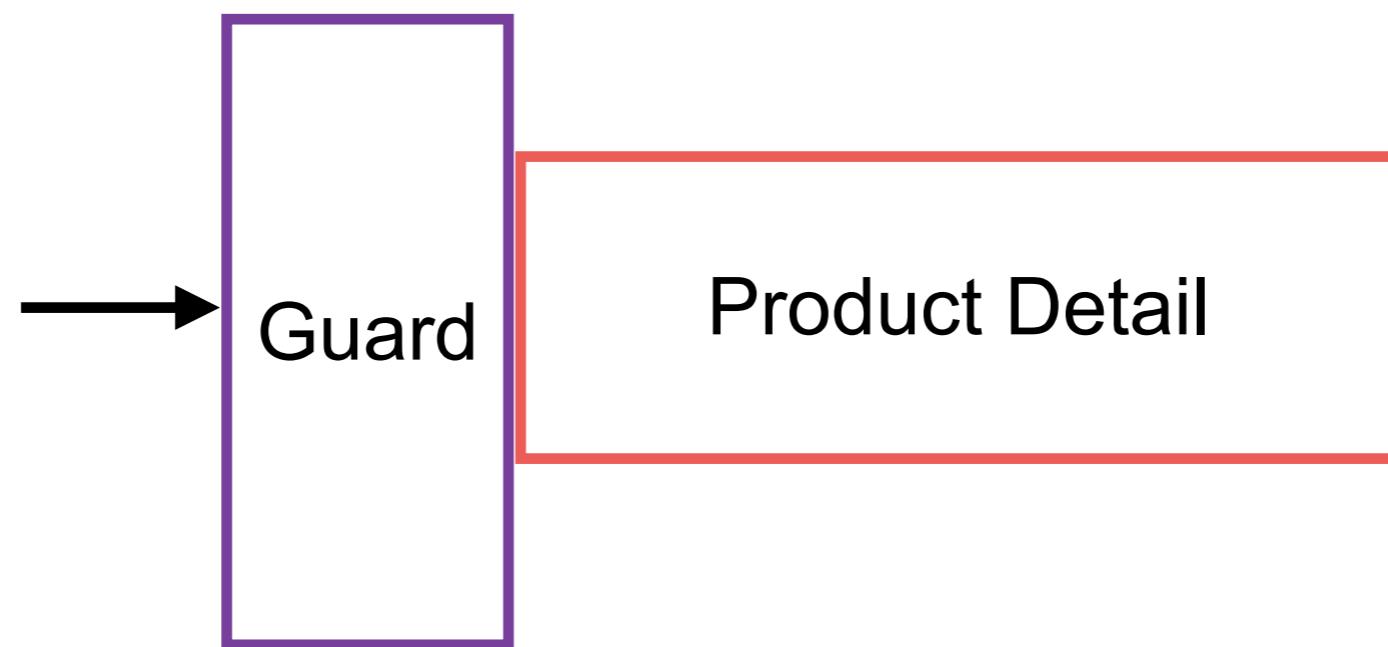
# Protect route with guard

Guard	Description
CanActivate	Guard navigate to route
CanDeactivate	Guard navigate from route
Resolve	Prefetch data before activate route
CanLoad	Prevent asynchronous routing



# Protect route with guard

Try to protect a product detail page



*Product id MUST be number !!*



# Create guard

```
$ng generate guard product-detail/product-detail  
--flat
```



# Create product detail guard

product-detail.guard.ts

```
import { Injectable } from '@angular/core';
import { CanActivate, Router, ActivatedRouteSnapshot, RouterStateSnapshot,
UrlTree } from '@angular/router';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ProductDetailGuard implements CanActivate {

  constructor(private router: Router) { }

  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    return true;
  }
}
```



# Create product detail guard

product-detail.guard.ts

```
export class ProductDetailGuard implements CanActivate {  
  constructor(private router: Router) { }  
  
  canActivate(  
    next: ActivatedRouteSnapshot,  
    state: RouterStateSnapshot): Observable<boolean | UrlTree> |  
Promise<boolean | UrlTree> | boolean | UrlTree {  
  
  const id = +next.url[1].path;  
  if (isNaN(id) || id < 1) {  
    alert('Invalid product Id');  
    this.router.navigate(['/products']);  
    return false;  
  }  
  return true;  
}  
}
```



# Services



# Service

Class with specified purpose

Independent from any component

Provide shared data and logic across components

Encapsulate external interactions

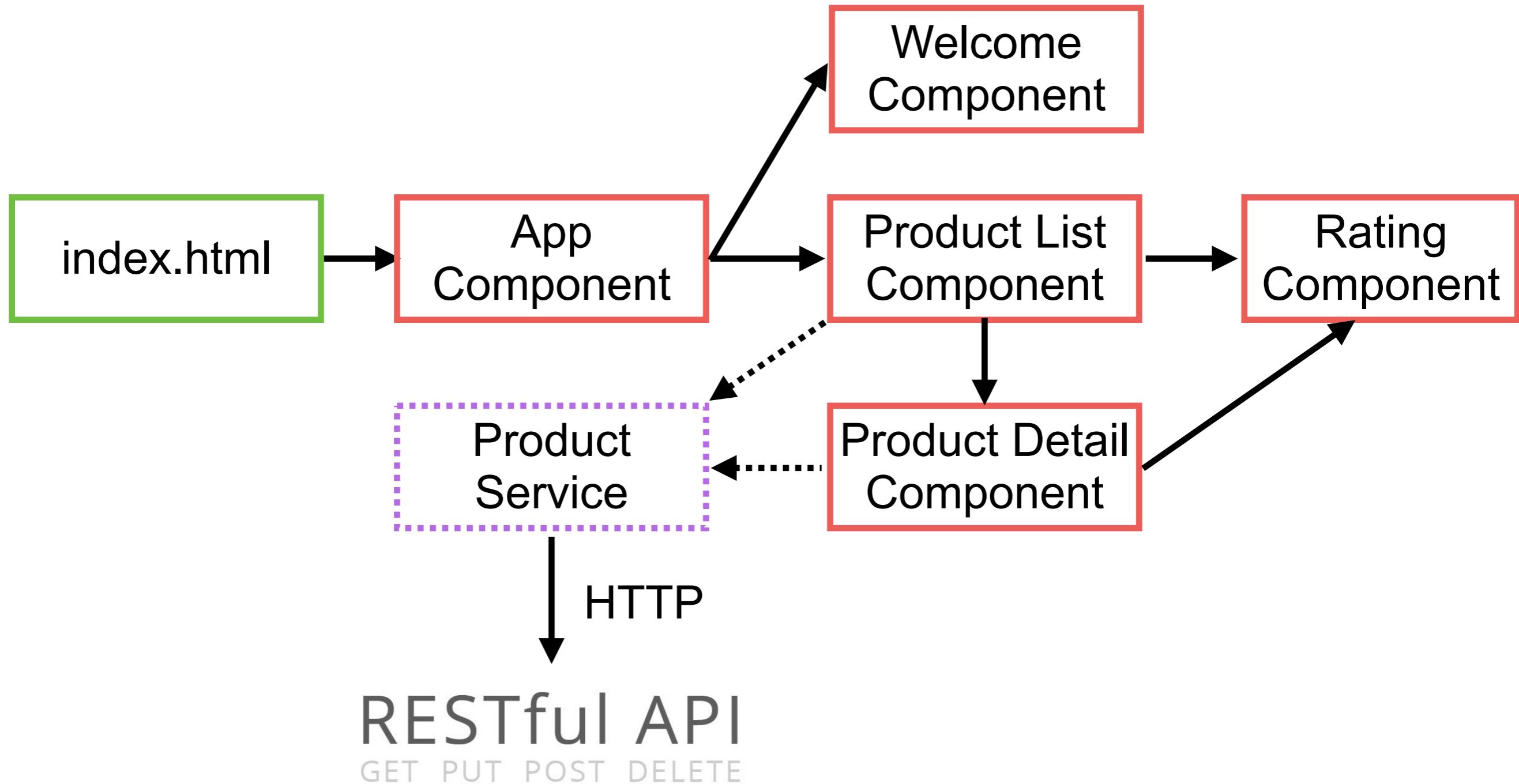


# Service

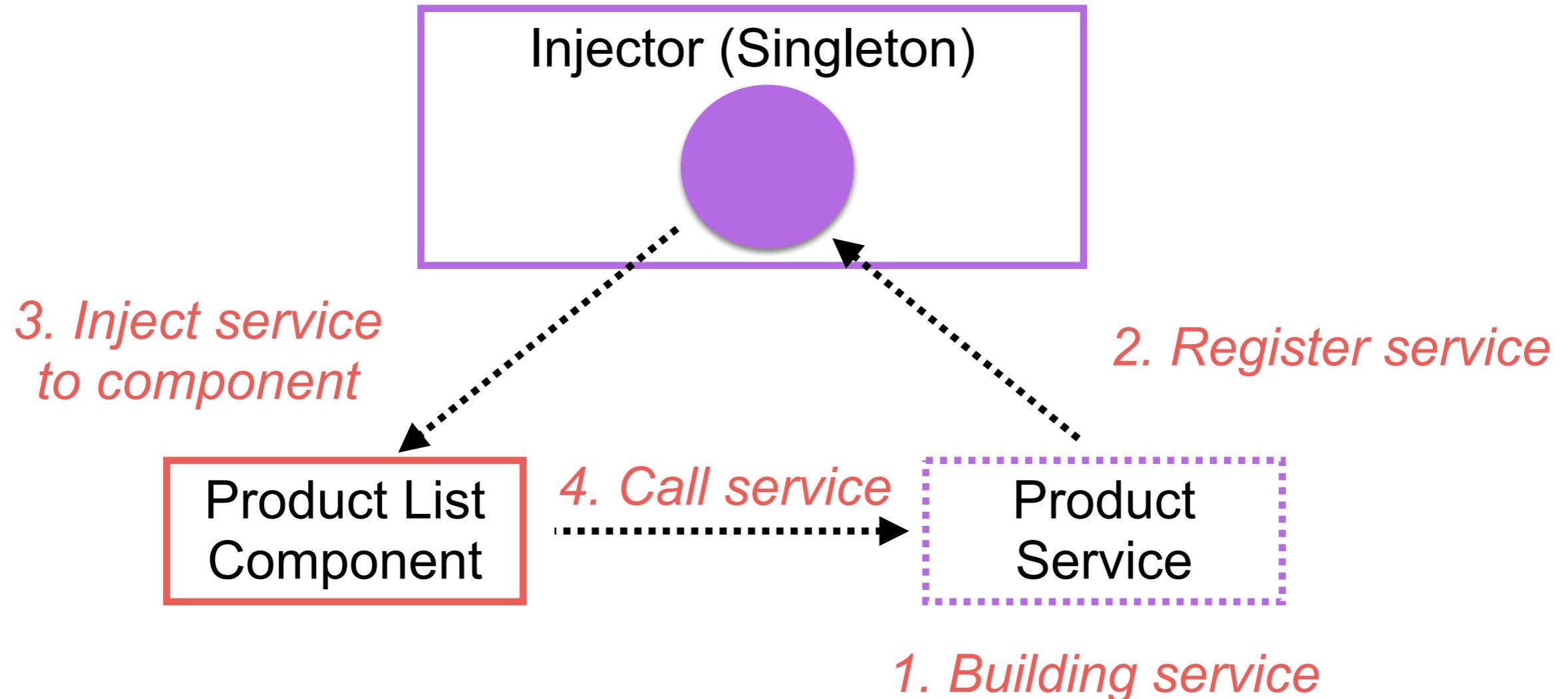
How service working ?  
Building service  
Register service  
Inject service



# Application Architecture



# How service working ?



# Building service

\$ng generate service products/product

product.service.ts

```
import { Injectable } from '@angular/core';
import {ProductDataModel} from './product';

@Injectable({
  providedIn: 'root'
})
export class ProductService {

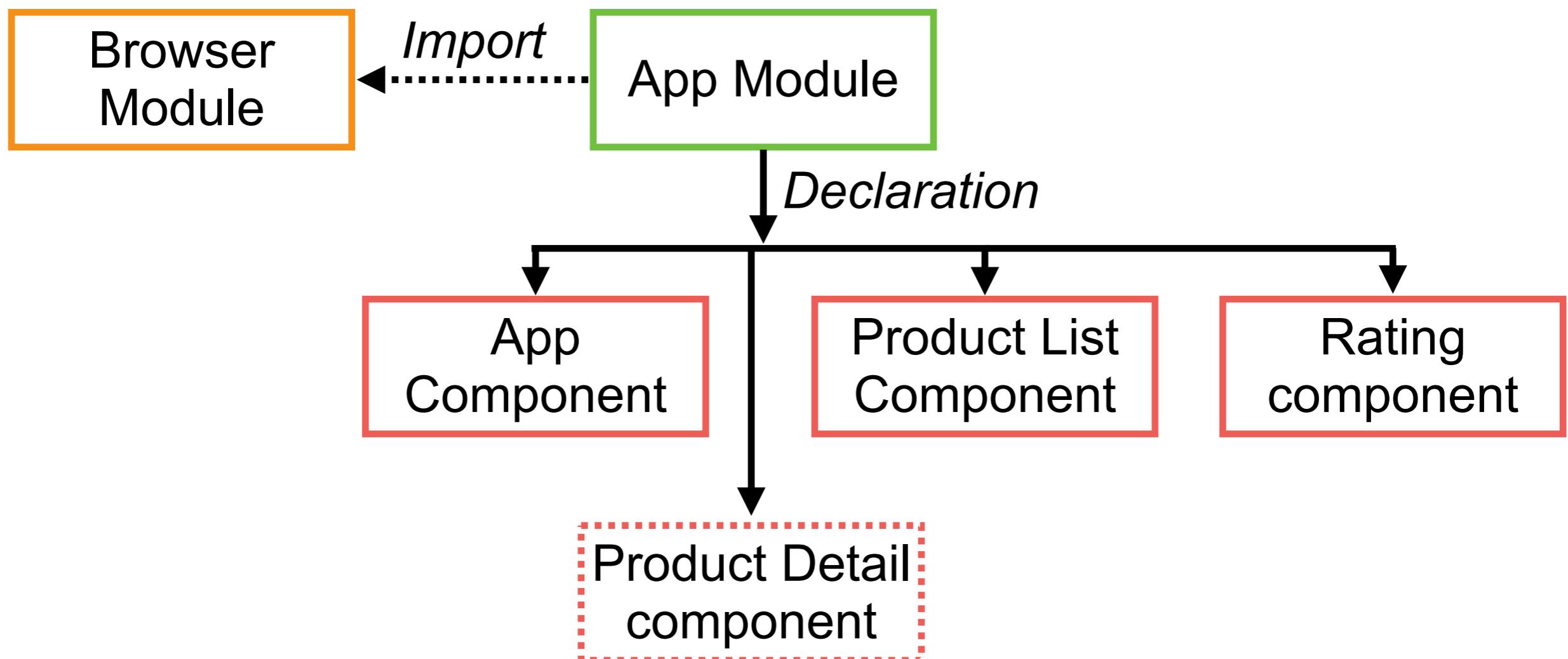
  constructor() { }

  getProducts(): ProductDataModel[] {
    return;
  }
}
```

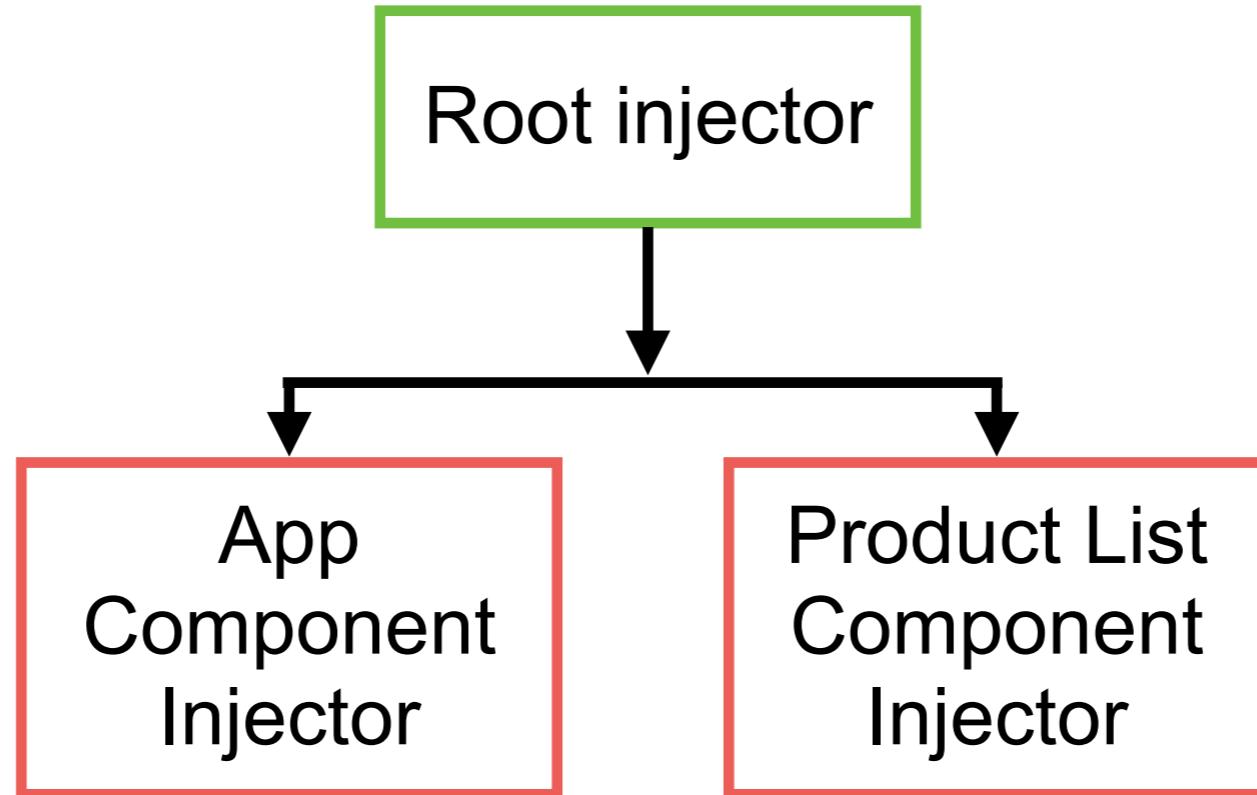


# Create product detail component

\$ng generate component productDetail



# Register service



# Register service

Root injector  
Component injector

product.service.ts

```
import { Injectable } from '@angular/core';
import {ProductDataModel} from './product';

@Injectable({
  providedIn: 'root'
})
export class ProductService {

  constructor() { }

  getProducts(): ProductDataModel[] {
    return;
  }
}
```

→ *Root injector*



# Add service in component

product-list.component.ts

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

@Component({
  selector: 'app-product-list',
  providers: [ProductService],
})
export class ProductListComponent implements OnInit {
```

*Add service providers*



# Inject service to component

## Using constructor

product-list.component.ts

```
export class ProductListComponent implements OnInit {  
  private _filterData: string;  
  filteredProducts: ProductDataModel[]; Constructor injection  
  constructor(private productService: ProductService) {  
    console.log(this.products);  
  }  
  
  ngOnInit(): void {  
    this.products = this.productService.getProducts();  
    this.filteredProducts = this.products;  
  }  
}
```



# Working with HTTP



# Angular Modules

