สยามชำนาญกิจ
SIAM CHAMNAN KIT

Getting start with Angular 6

Somkiat Puisungnoen

Somkiat | Home

Somkiat Puisungnoen

Update Info 1 | View Activity Log 10+ | ...

Timeline | About | Friends 3,138 | Photos | More

When did you work at Opendream?

22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post | Photo/Video | Live Video | Life Event

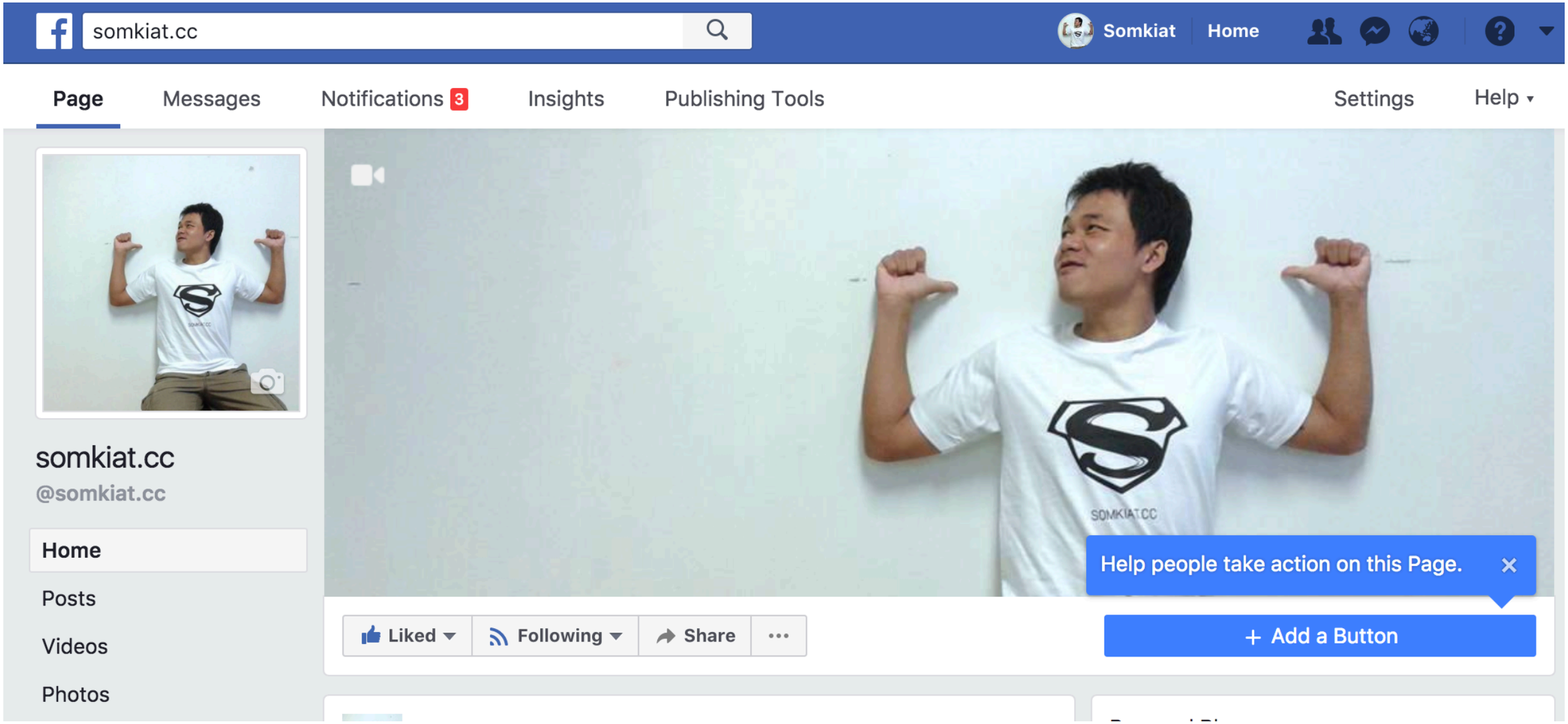
What's on your mind?

Public | Post

Somkiat Puisungnoen

15 mins · Bangkok ·

Java and Bigdata



https://github.com/up1/course-angular-2024

Agenda

- Introduction to Angular
- Installation and configuration
- Structure of Angular project
- Introduction to TypeScript
- Design and develop component
- Routing management
- Service and Dependency Injection (DI)
- Working with RESTful APIs

Agenda

- State management
- Testing strategies
- Deployment
- Optimization performance
- New features of Angular 17

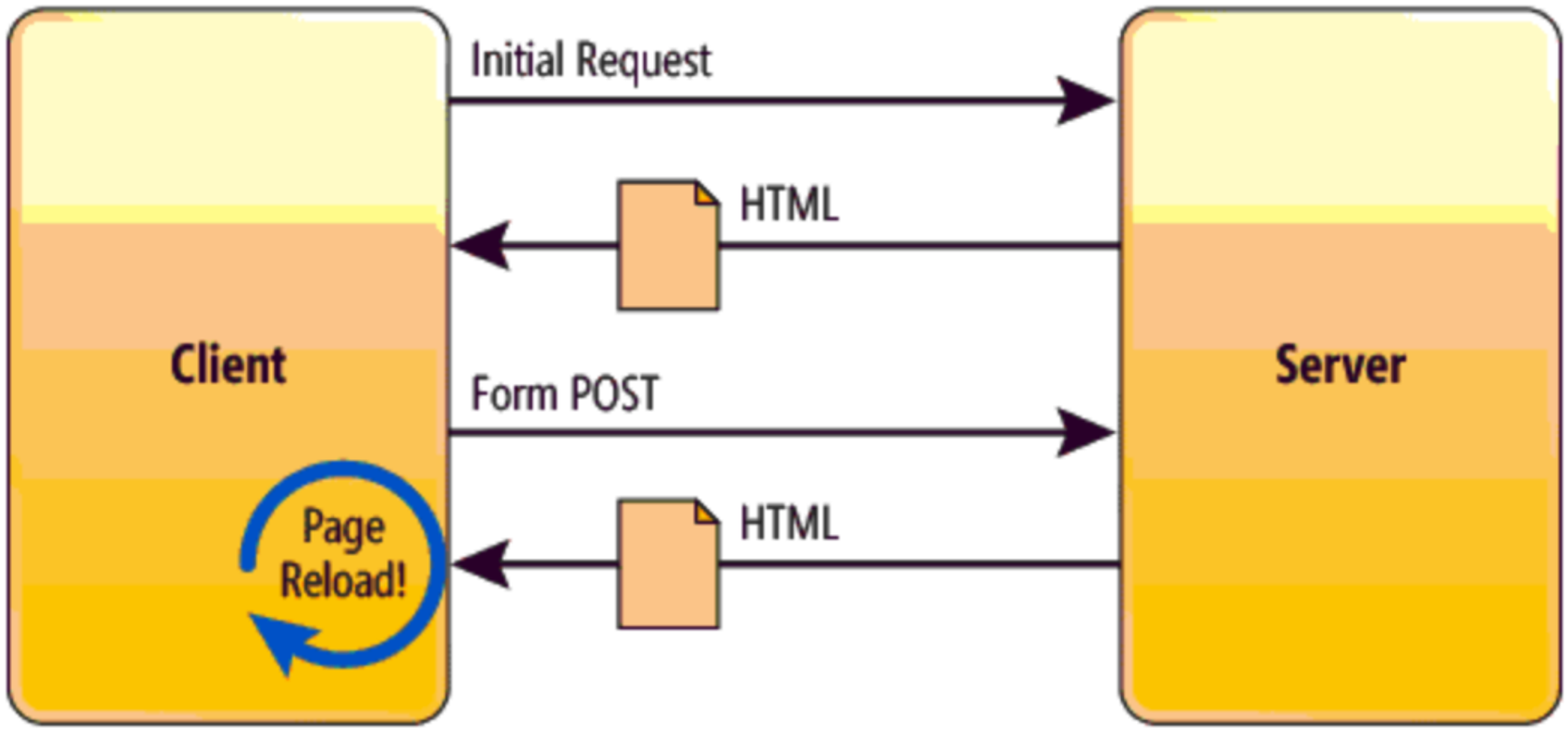
Angular

What is Angular ?

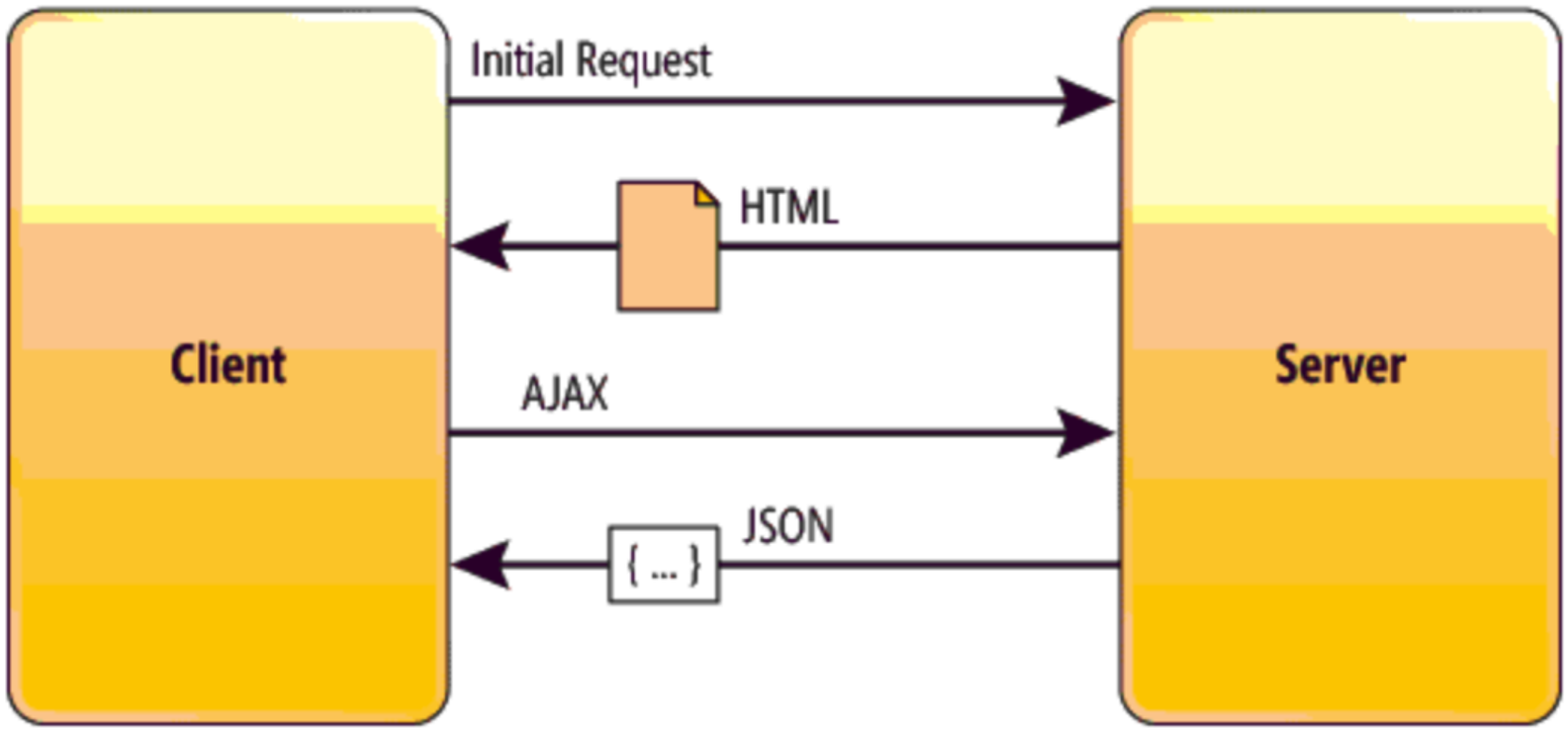
JavaScript framework with allows us to create reactive **Single Page Application** (SPA)

https://angular.io/

Traditional

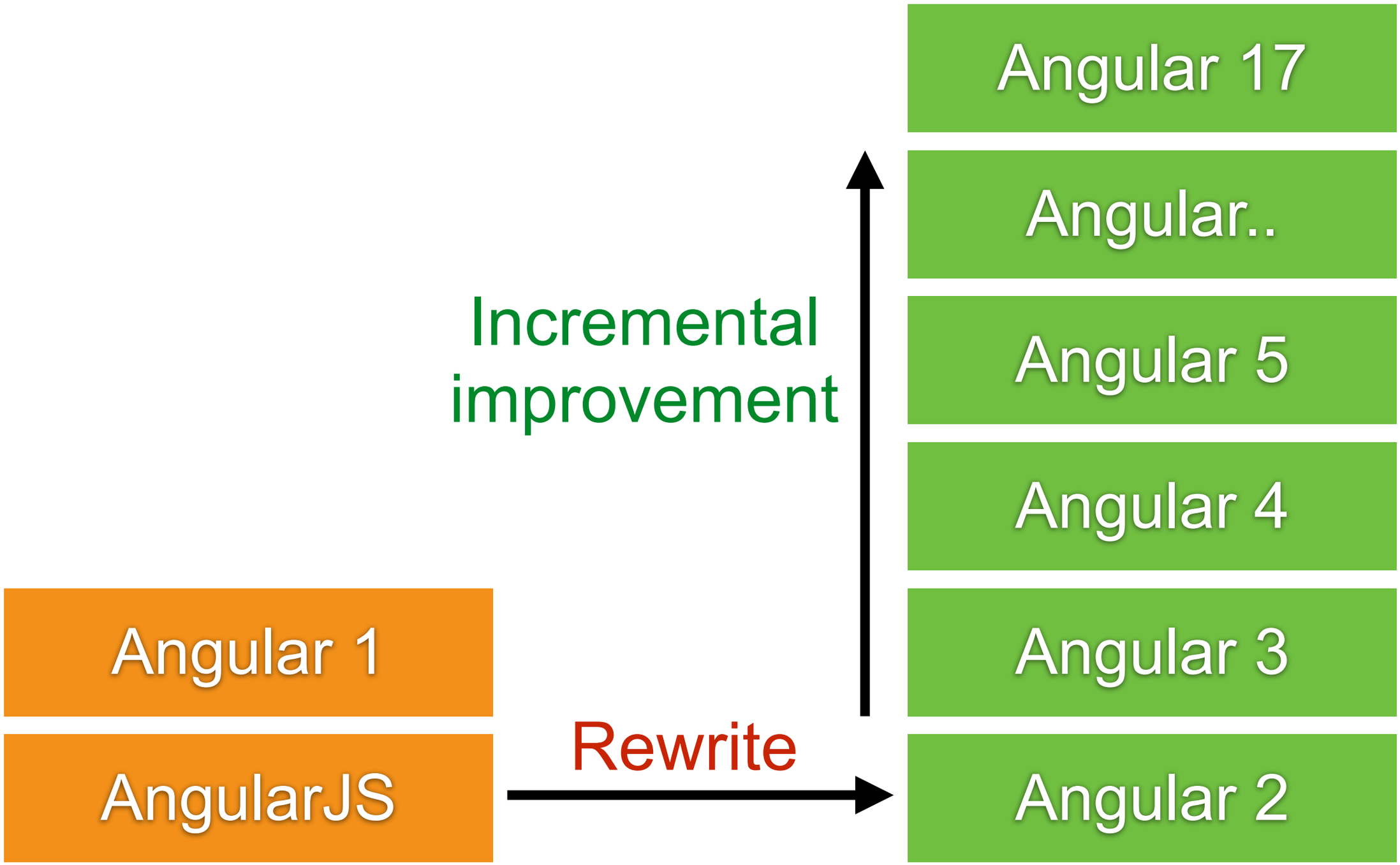


Single Page Application



Angular

Angular



Software requirement

Install NodeJS 18 !!

Node.js® is an open-source, cross-platform JavaScript runtime environment.

Download Node.js®

20.11.1 LTS
Recommended For Most Users

Other Downloads | Changelog | API Docs

21.7.1 Current
Latest Features

Other Downloads | Changelog | API Docs

For information about supported releases, see the release schedule.

https://nodejs.org/en/

Install Text Editor

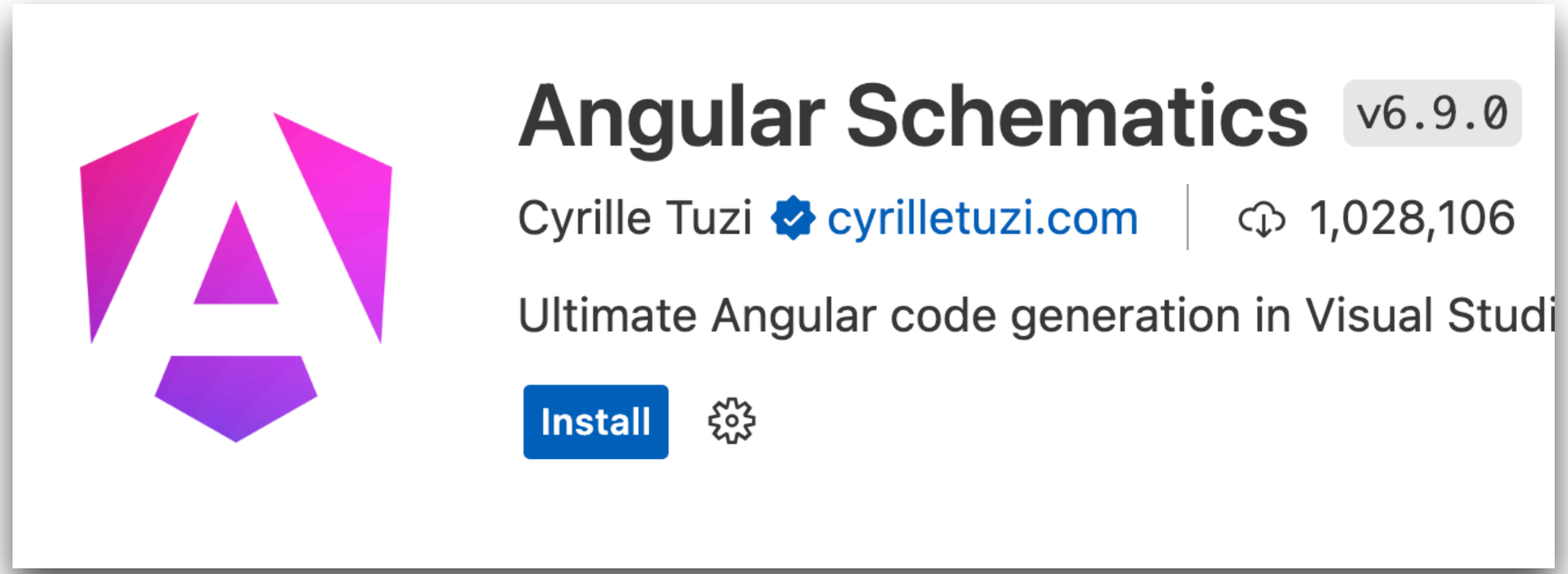
Visual Studio Code

https://code.visualstudio.com/

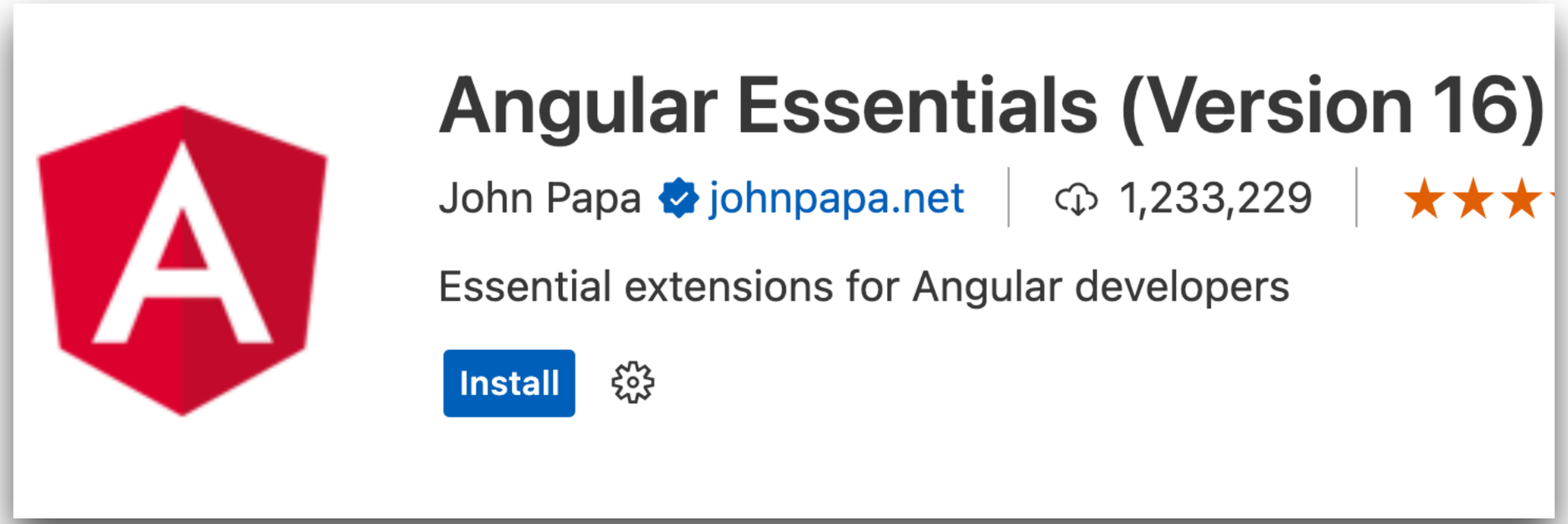
VSCode Extensions

Angular Schematics

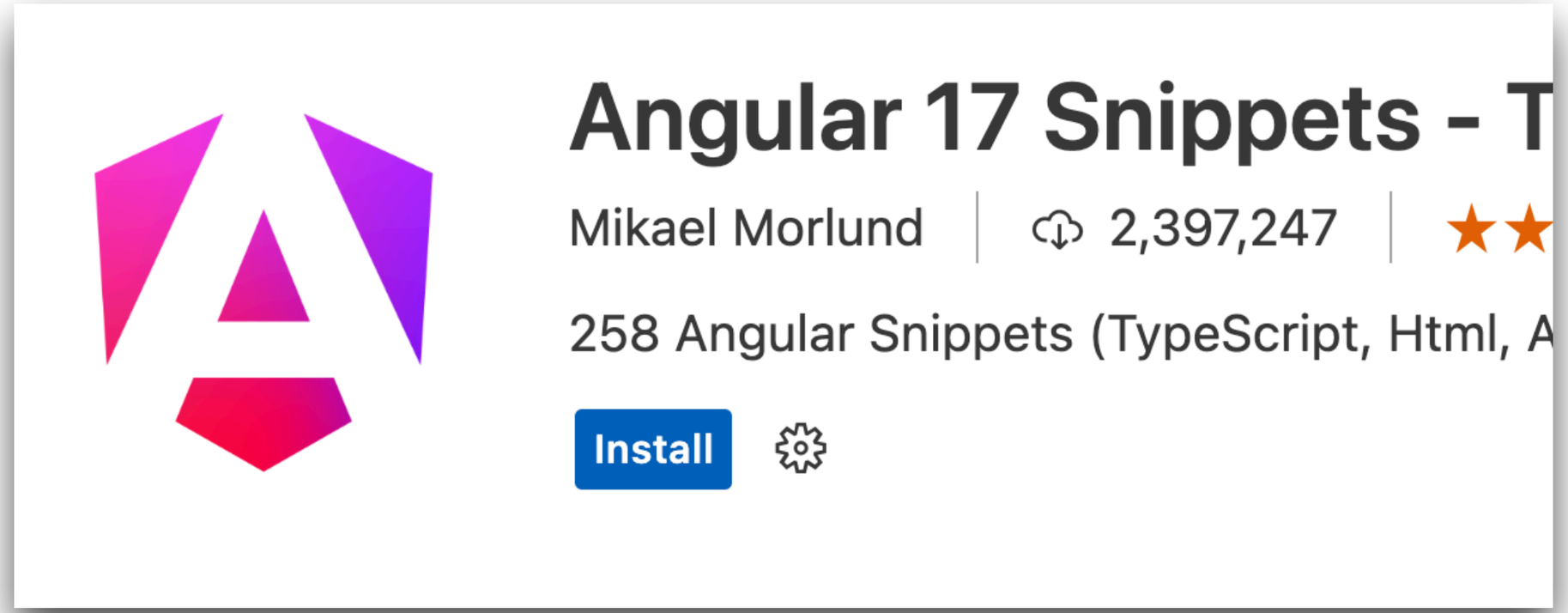
Code generation



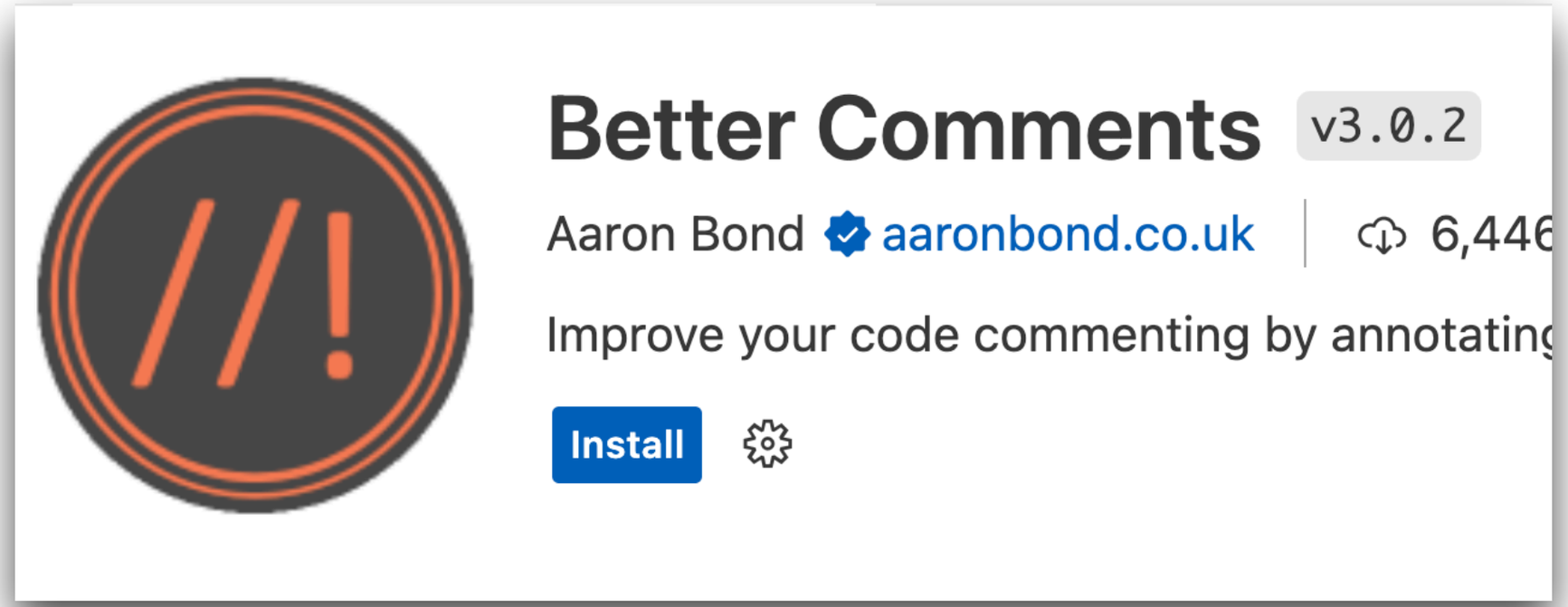
Angular Essentials



Angular Snippets



Better Comments



Angular CLI

Angular CLI

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

$npm install -g @angular/cli@latest

$ng version

https://cli.angular.io/

Install Angular CLI

$ng version

```
     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 17.3.0
Node: 20.6.0
Package Manager: npm 9.8.1
OS: darwin arm64

Angular: 17.3.0
... animations, cli, common, compiler, compiler-cli, core, forms
... platform-browser, platform-browser-dynamic, router
```

Try to create first project

$ng new hello-app

```
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE hello-app/README.md (1062 bytes)
CREATE hello-app/.editorconfig (274 bytes)
CREATE hello-app/.gitignore (548 bytes)
CREATE hello-app/angular.json (2607 bytes)
CREATE hello-app/package.json (1040 bytes)
CREATE hello-app/tsconfig.json (903 bytes)
CREATE hello-app/tsconfig.app.json (263 bytes)
CREATE hello-app/tsconfig.spec.json (273 bytes)
CREATE hello-app/.vscode/extensions.json (130 bytes)
CREATE hello-app/.vscode/launch.json (470 bytes)
CREATE hello-app/.vscode/tasks.json (938 bytes)
CREATE hello-app/src/main.ts (250 bytes)
CREATE hello-app/src/favicon.ico (15086 bytes)
CREATE hello-app/src/index.html (294 bytes)
CREATE hello-app/src/styles.css (80 bytes)
CREATE hello-app/src/app/app.component.css (0 bytes)
CREATE hello-app/src/app/app.component.html (19903 bytes)
CREATE hello-app/src/app/app.component.spec.ts (925 bytes)
CREATE hello-app/src/app/app.component.ts (305 bytes)
CREATE hello-app/src/app/app.config.ts (227 bytes)
CREATE hello-app/src/app/app.routes.ts (77 bytes)
CREATE hello-app/src/assets/.gitkeep (0 bytes)
```

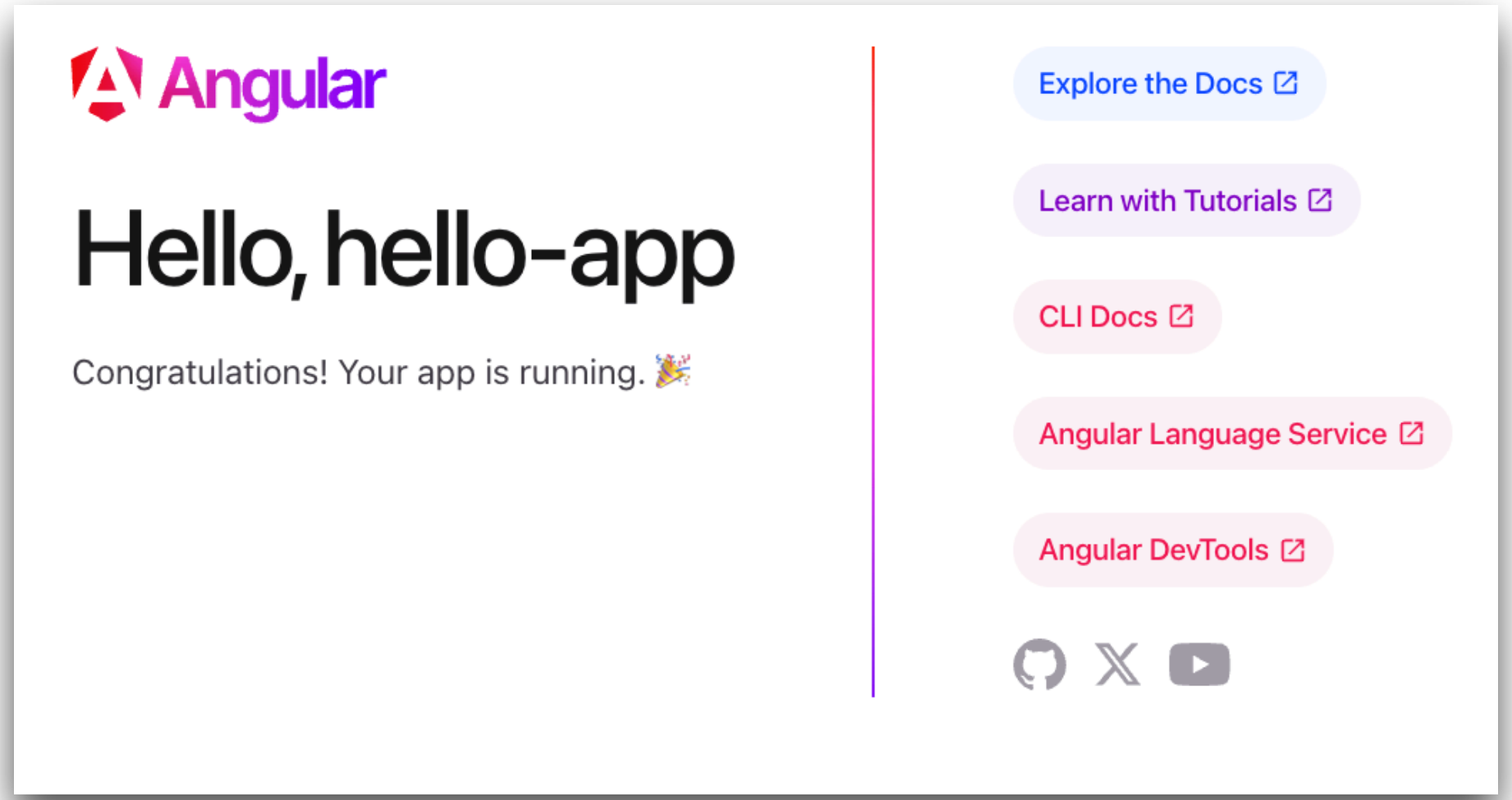
Run your app in dev mode

$cd hello-app
$ng serve

```
Initial chunk files | Names         |  Raw size
polyfills.js        | polyfills     |  83.60 kB |
main.js             | main          |  22.02 kB |
styles.css          | styles        |  95 bytes |

                    | Initial total | 105.71 kB

Application bundle generation complete. [0.696 seconds]

Watch mode enabled. Watching for file changes...
  ➜  Local:   http://localhost:4200/
  ➜  press h + enter to show help
```

Open in browser

http://localhost:4200/



Deploy to production

$ng build

```
Initial chunk files   | Names         |  Raw size | Estimated transfer size
main-PMRUNETL.js      | main          | 193.14 kB |               52.55 kB
polyfills-RT5I6R6G.js | polyfills     |  33.10 kB |               10.72 kB
styles-5INURTSO.css   | styles        |   0 bytes |                0 bytes

                      | Initial total | 226.25 kB |               63.27 kB
```

Output in folder **/dist/hello-app**

Deploy with Docker

Create Dockerfile

```dockerfile
# Stage 1: Build the Angular application
FROM node:18-alpine as build
WORKDIR /app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
RUN npm run build --omit=dev

# Stage 2: Serve the application from Nginx
FROM nginx:1.25.4-alpine
COPY --from=build /app/dist/hello-app/browser /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Build with Docker compose

Create docker-compose.yml

```yaml
version: "3"
services:
  app:
    build: .
    image: app:1.0
    ports:
      - "8888:80"
```

Build with Docker compose

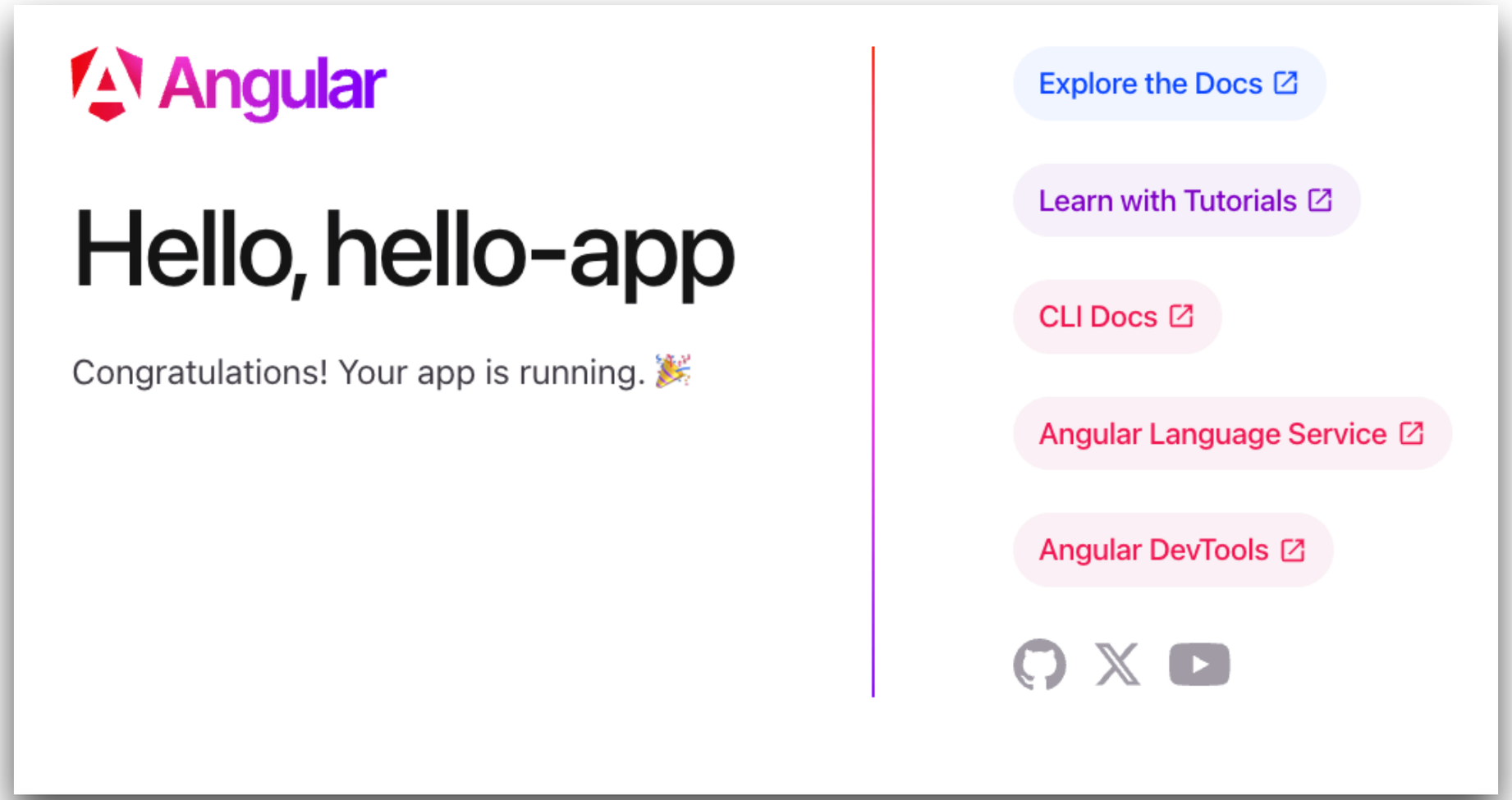
$docker compose build

$docker compose up -d

$docker compose ps

Open in browser

http://localhost:8888



Build with Docker compose

$docker compose build

$docker compose up -d

$docker compose ps

Tools

Ng Lint

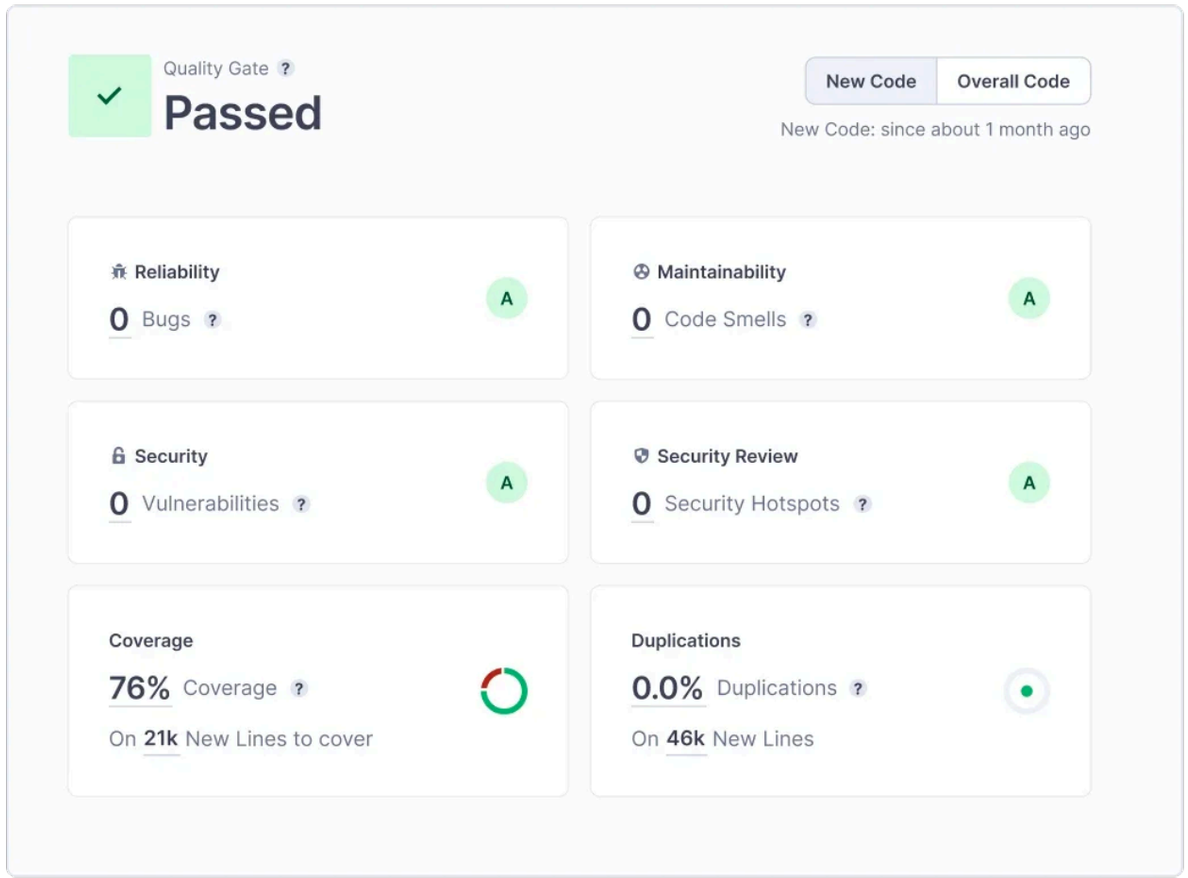
$ng lint

```
Cannot find "lint" target for the specified project.
You can add a package that implements these capabilities.

For example:
  ESLint: ng add @angular-eslint/schematics

Would you like to add ESLint now? Yes
i Using package manager: npm
✔ Found compatible package version: @angular-eslint/schematics@17.2.1.
✔ Package information loaded.
```

https://github.com/angular-eslint/angular-eslint

SonarQube

Static Code Analysis tool



https://www.sonarsource.com/products/sonarqube/

Question ?

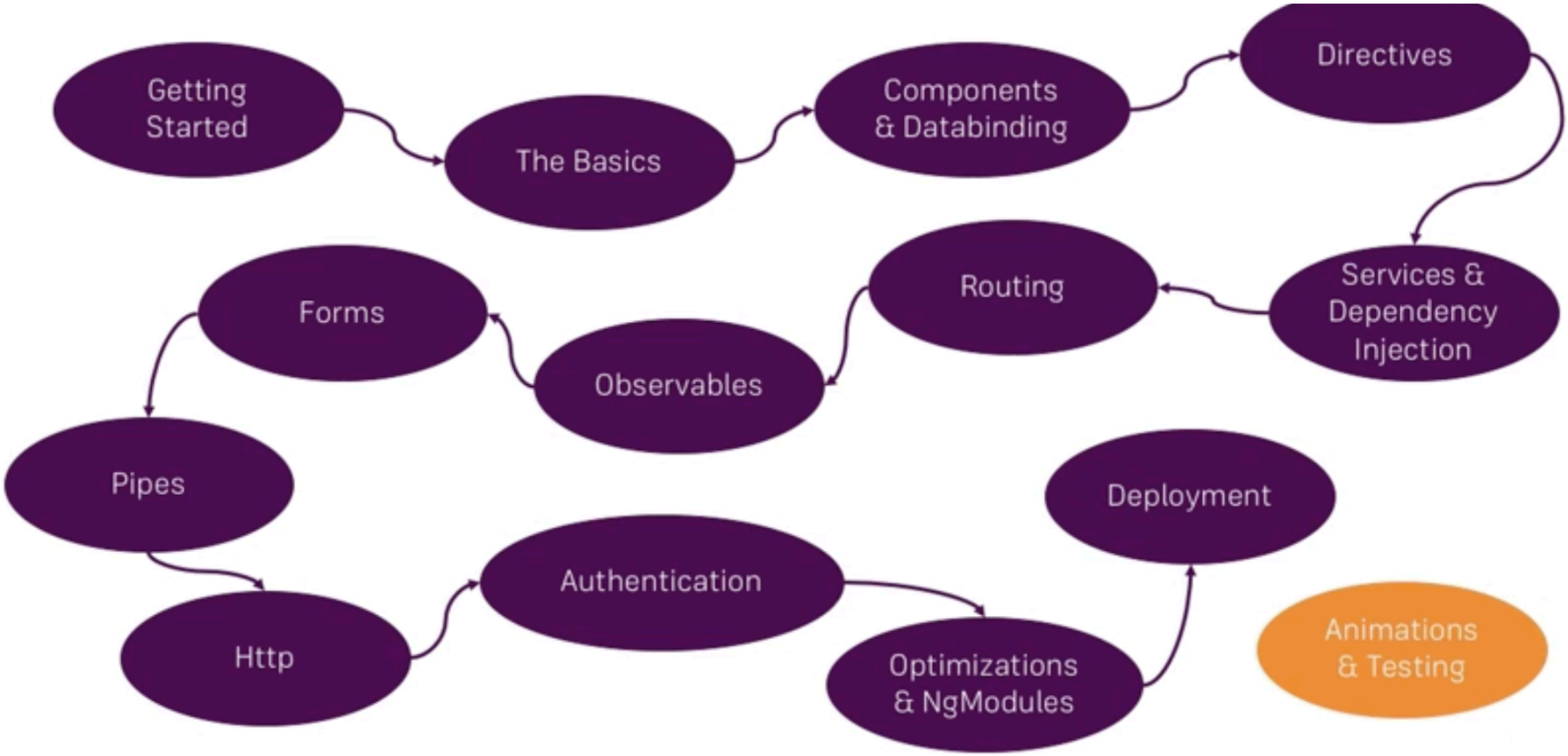
Server-Side Rendering (SSR)
Static Site Generation (SSG)
Single Page Application (SPA)

SSR vs CSR vs SSG

SSR	CSR	SSG
Rendering work completed on server	Rendering done on the user's machine within browser	Rendering completed at build time before users visits site
✔ Personalization	✔ Personalization	✘ Personalization requires rehydration
👍 SEO, FCP, TTI 👎 TTFB, Blank Page Syndrome	👍 FCP, TTFB, Low server costs 👎 SEO, TTI, Requires JavaScript	👍 SEO, TTFB, FCP, TTI 👎 Inflexible, build times
Common Frameworks: ASP.NET, Next.js, PHP (Laravel), Node.js	Common Frameworks: React, Angular, Vue	Common Frameworks: Next.js, Gatsby, Hugu, Nuxt

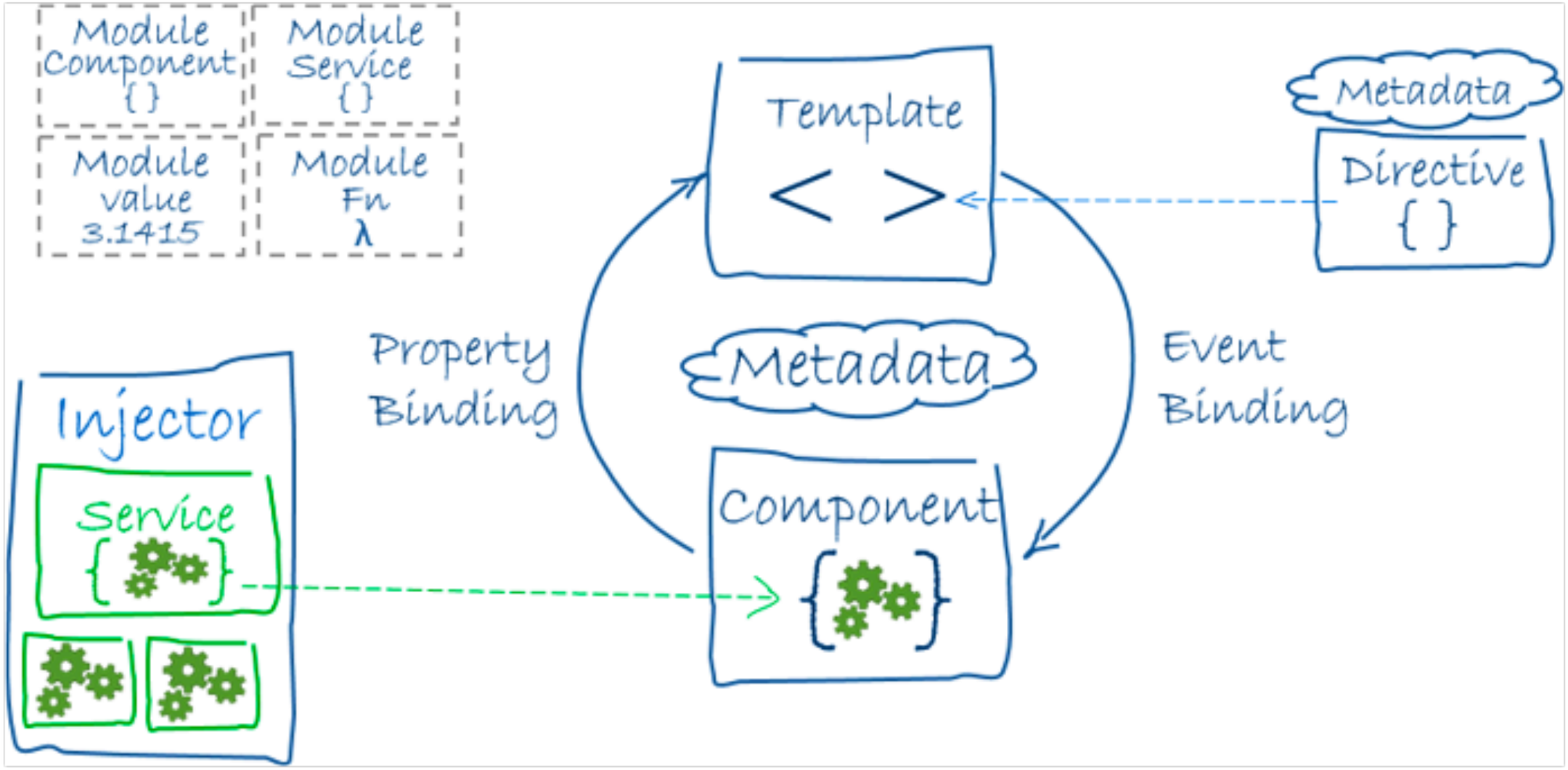
Learning Path

Learning structure



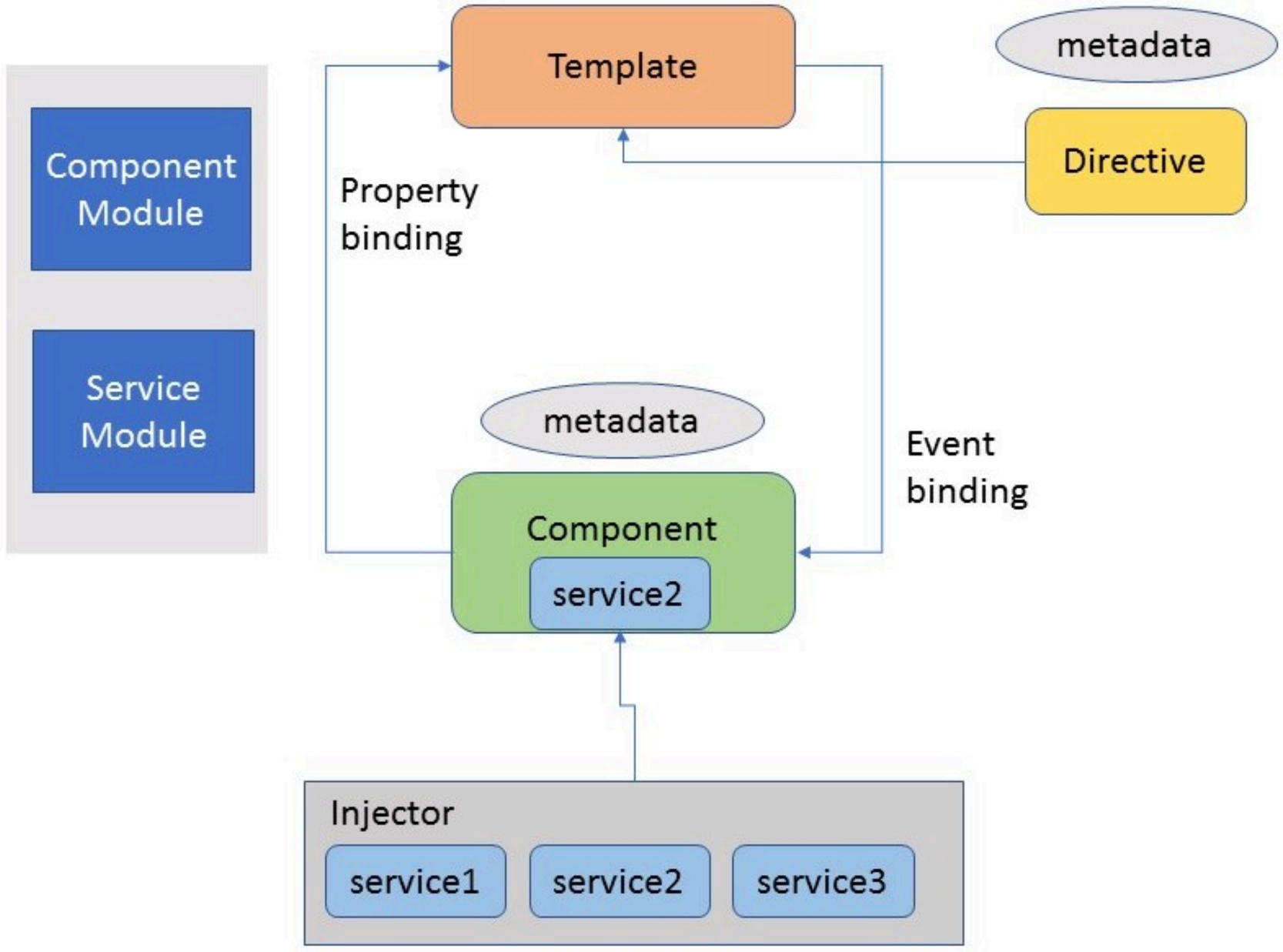
Basic of Angular

Angular Architecture

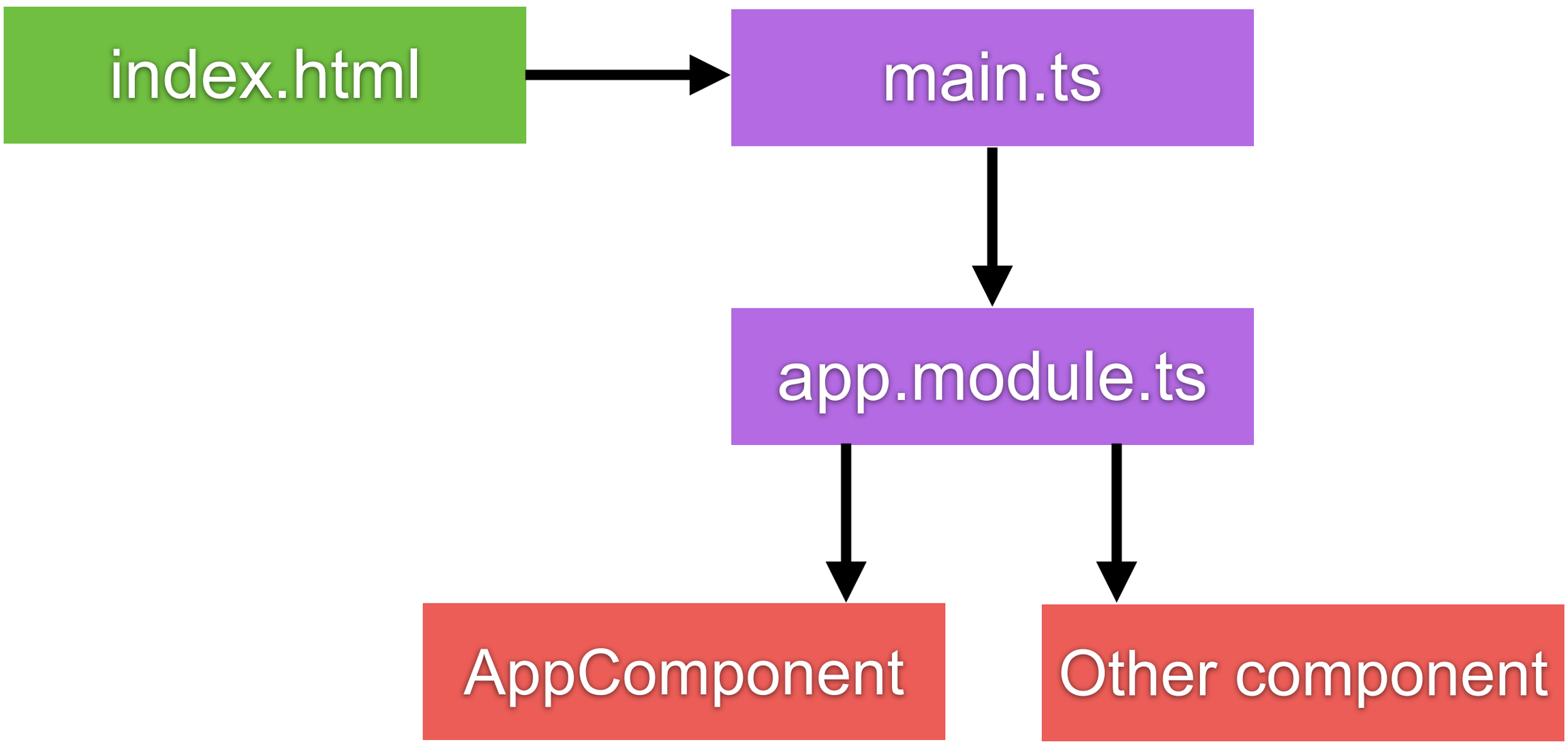


https://angular.io/guide/architecture

Angular Architecture

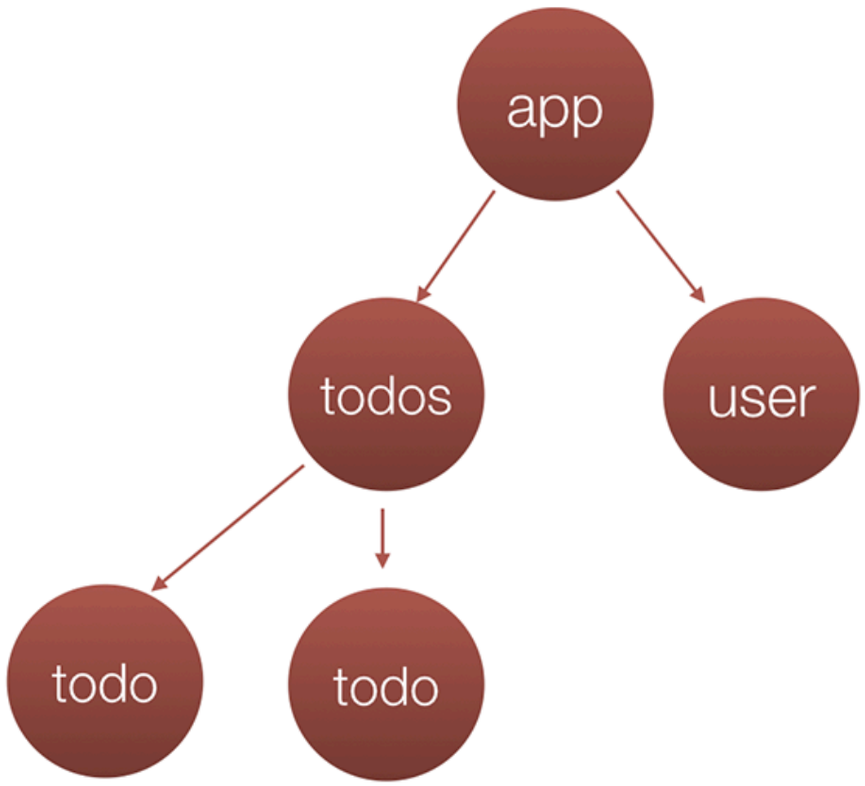


How Angular working ?



Components

Angular application is a tree of **Components**
Top level component is the application itself
Component is rendered by the browser



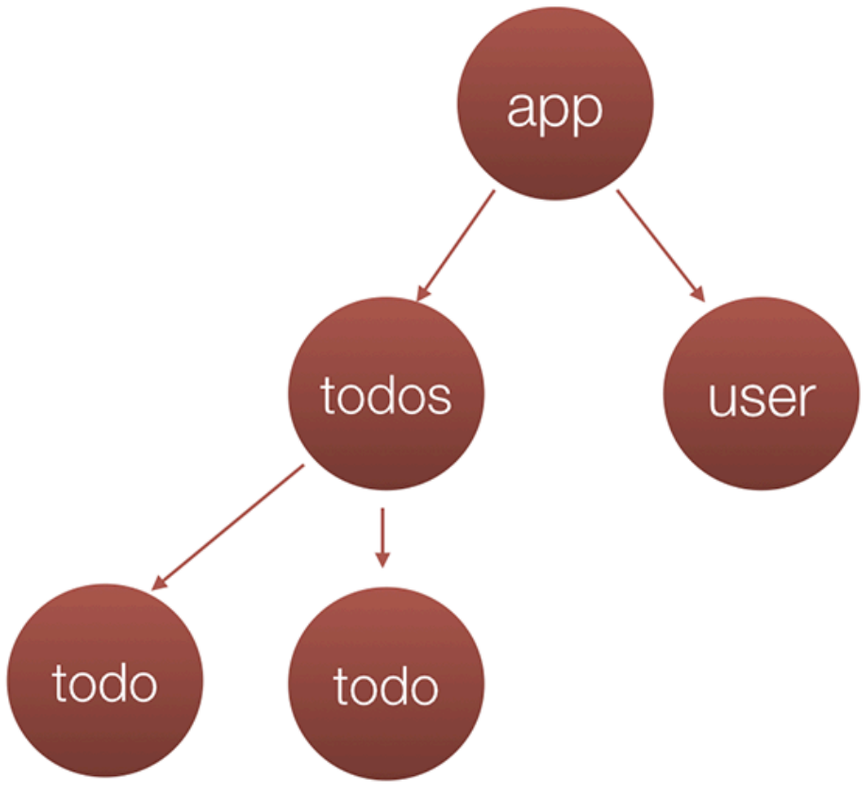
Components

Own template
Own style
Own business logic

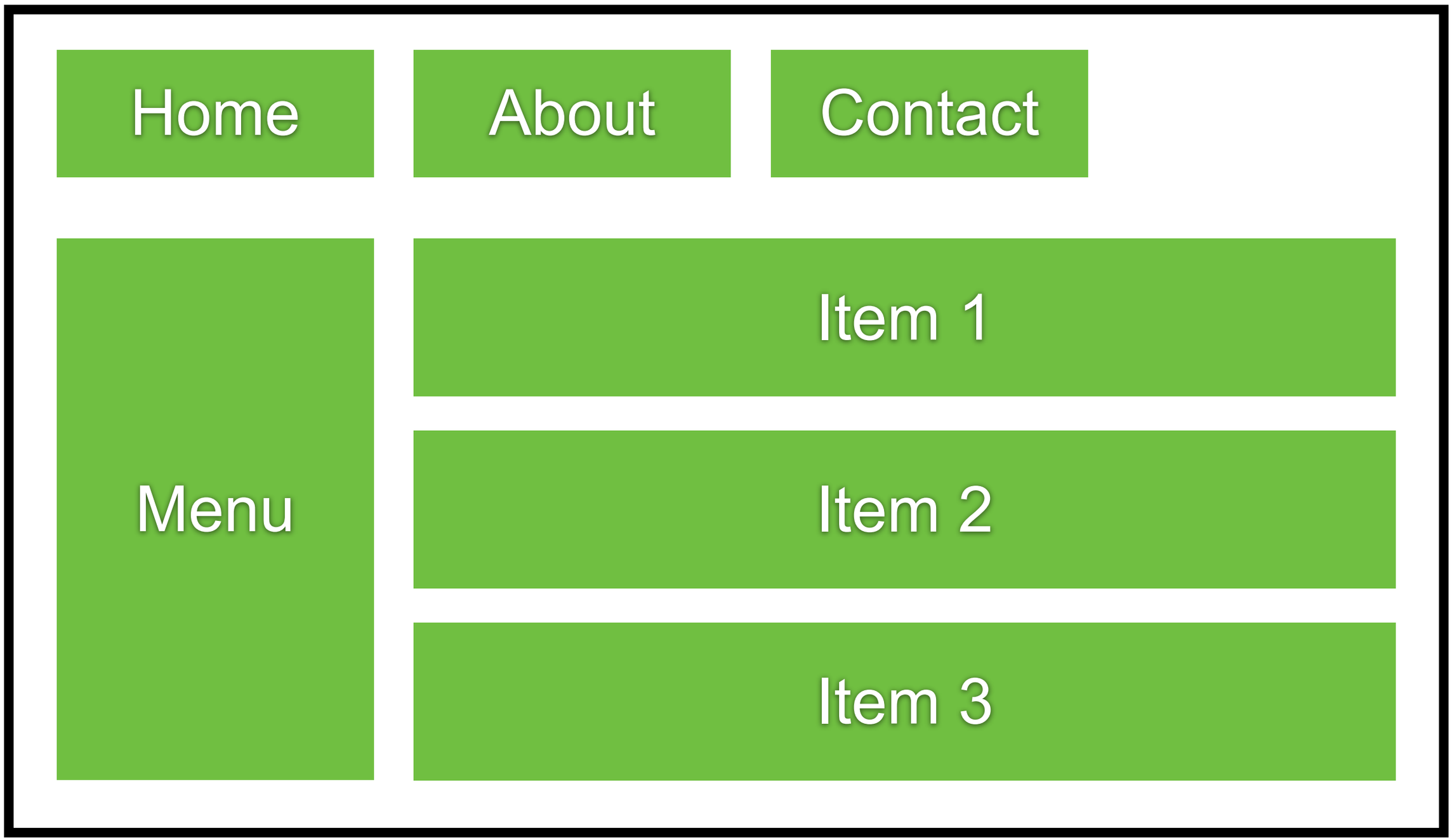
Split complex app to small component !!

Components

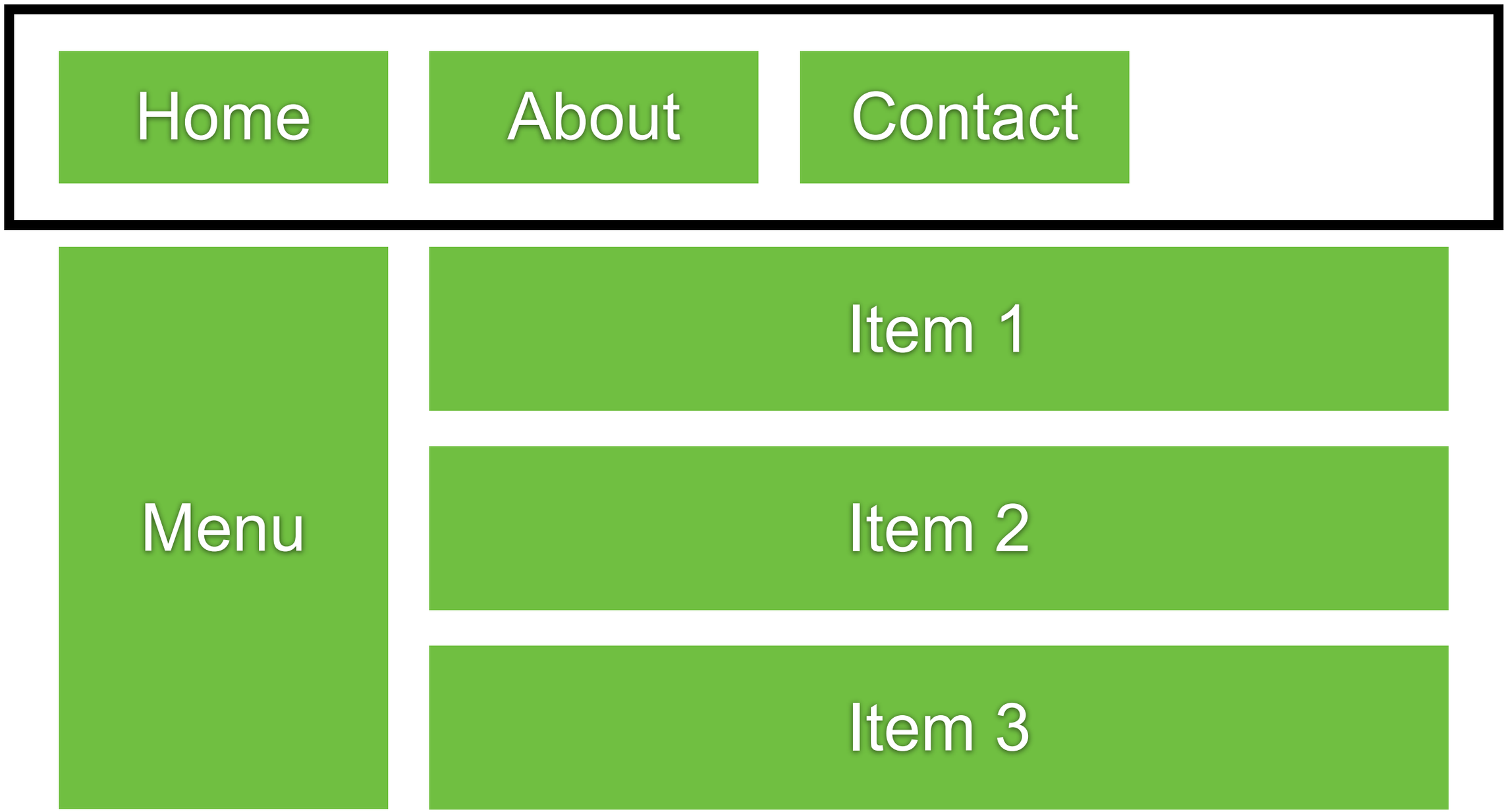
Composable
Reusable
Hierarchical



Design components



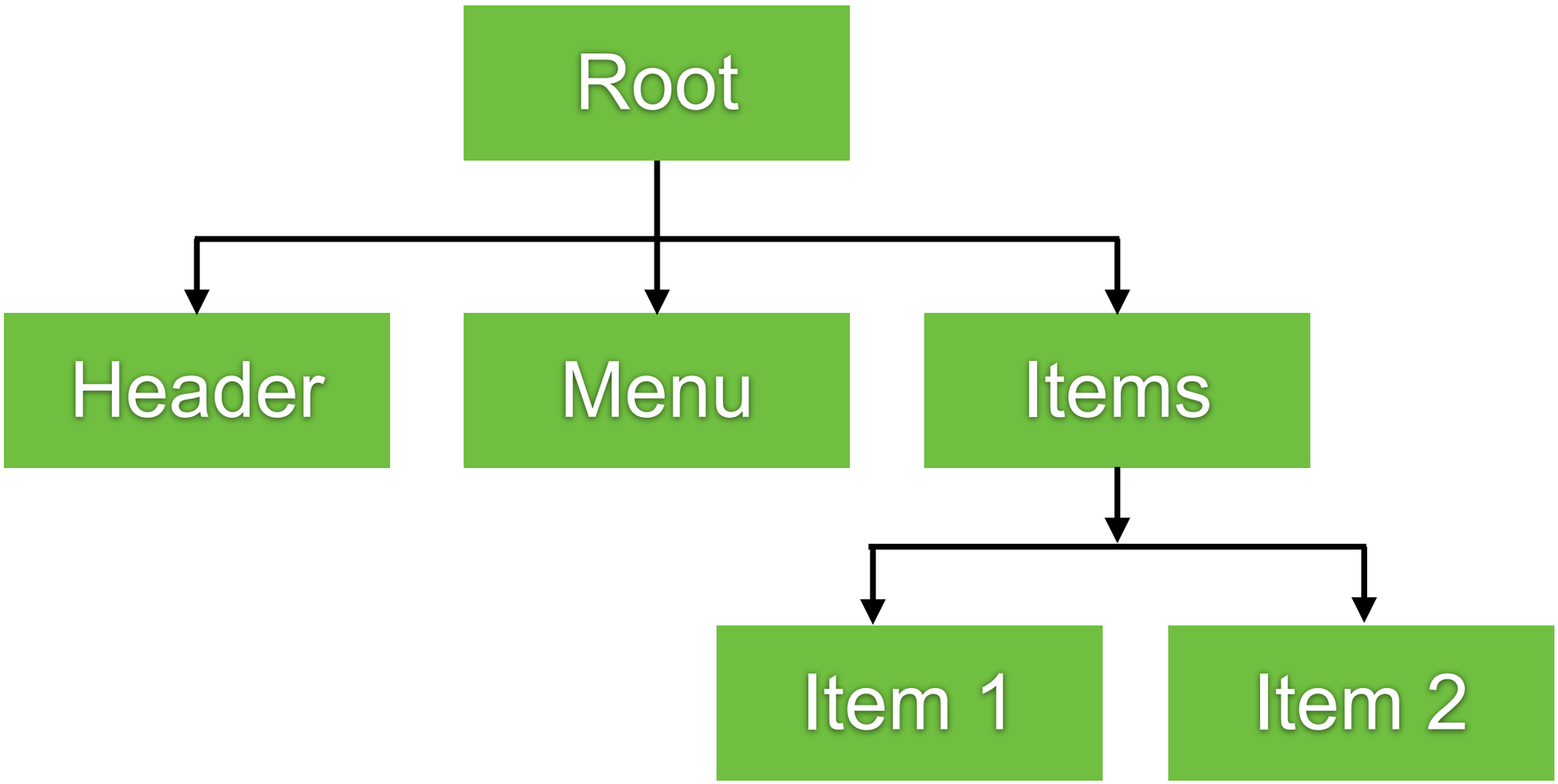
Header component



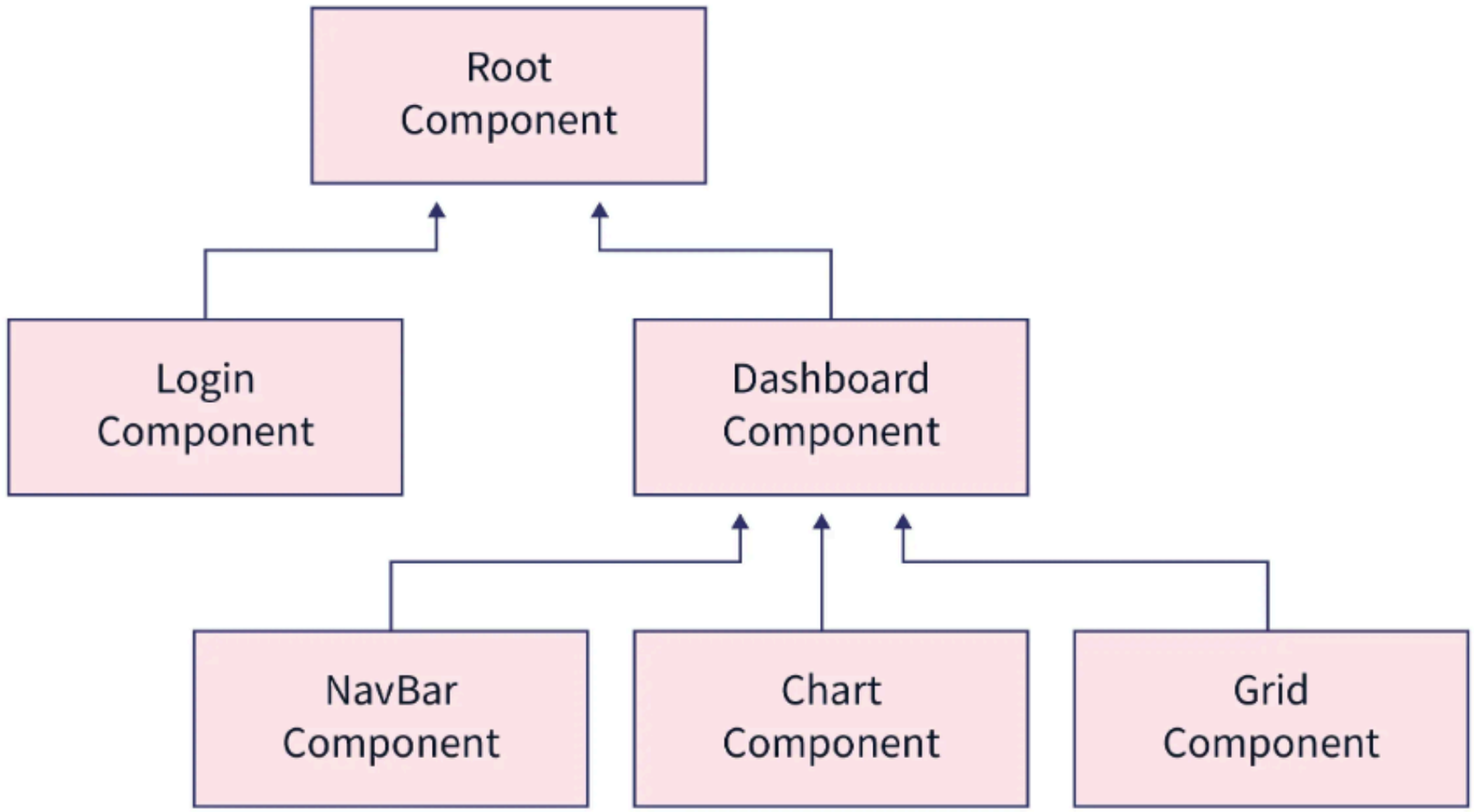
Sidebar/Menu component

Item component

Design components



Types of Components ?



Types of Components ?

Presentation/
Dumb

Smart

Presentation or Dumb

Don’t access to service
Don’t store data in backend
Display only
Working with change detection

Smart Component

Manage states
Use services
Business logics

Component

Mail building blocks of Angular application
HTML template
TypeScript class
CSS selector and CSS styles
Template or template url

Component

Template + Class + Metadata

View layout
Create with HTML
Binding and directives

Code support view
Create with TypeScript
Data and logic

Extra data
Define decorator
@Component

Component

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

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

Component

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

Component decorator

Directive name used in HTML

TypeScript Class

Component

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

Template

Building template
Using component as a **directive**
Binding with **Interpolation**
Add logic with directive

Define template in component

Inline template

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

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

Link template

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

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

Component

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

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

Standalone component

Standalone Component

Angular 16

Standalone Component

Don’t need NgModule
Easier way to create component
Reuse in any part of application
Isolated from other
Lazy load
Easy to test

Angular is Modular

@angular/
core

@angular/
router

@angular/
http

Create new component with CLI

$ng generate component <name>

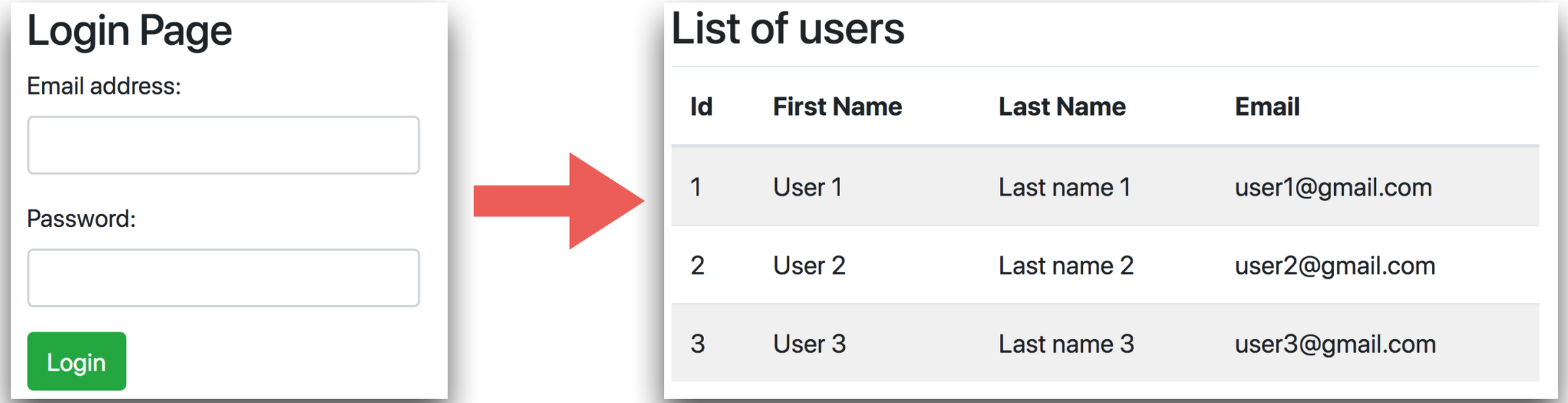
```
? What name would you like to use for the component? login
CREATE src/app/login/login.component.css (0 bytes)
CREATE src/app/login/login.component.html (20 bytes)
CREATE src/app/login/login.component.spec.ts (589 bytes)
CREATE src/app/login/login.component.ts (230 bytes)
```

With flat structure

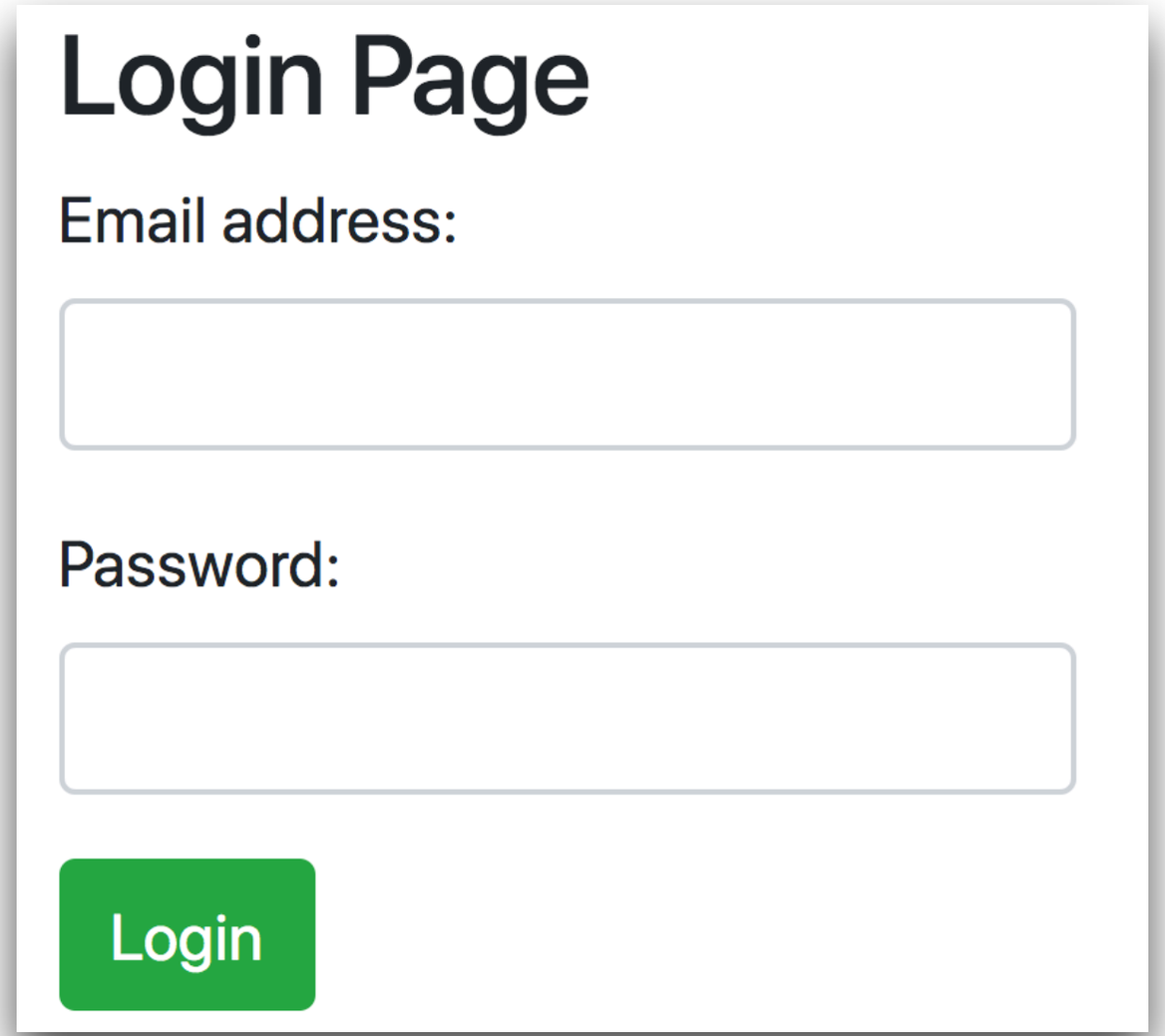
$ng generate component --flat

```
? What name would you like to use for the component? login2
CREATE src/app/login2.component.css (0 bytes)
CREATE src/app/login2.component.html (21 bytes)
CREATE src/app/login2.component.spec.ts (596 bytes)
CREATE src/app/login2.component.ts (234 bytes)
```

Workshop



Login Page



Create Login Component

$ng generate component login

HTML	CSS	TS	Spec

Login Component

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

@Component({
  selector: 'app-login',
  standalone: true,
  imports: [],
  templateUrl: './login.component.html',
  styleUrl: './login.component.css'
})
export class LoginComponent {

}
```

List of user page

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 User Component

$ng generate component user

```
CREATE src/app/user/user.component.css (0 bytes)
CREATE src/app/user/user.component.html (19 bytes)
CREATE src/app/user/user.component.spec.ts (582 bytes)
CREATE src/app/user/user.component.ts (226 bytes)
```

User Component

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

@Component({
  selector: 'app-user',
  standalone: true,
  imports: [],
  templateUrl: './user.component.html',
  styleUrl: './user.component.css'
})
export class UserComponent {

}
```

Create Model of User

$ng generate class models/user

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

We need routing ?

Angular have @angular/router

/login

Login Page

Email address:

Password:

Login

/user

List of users

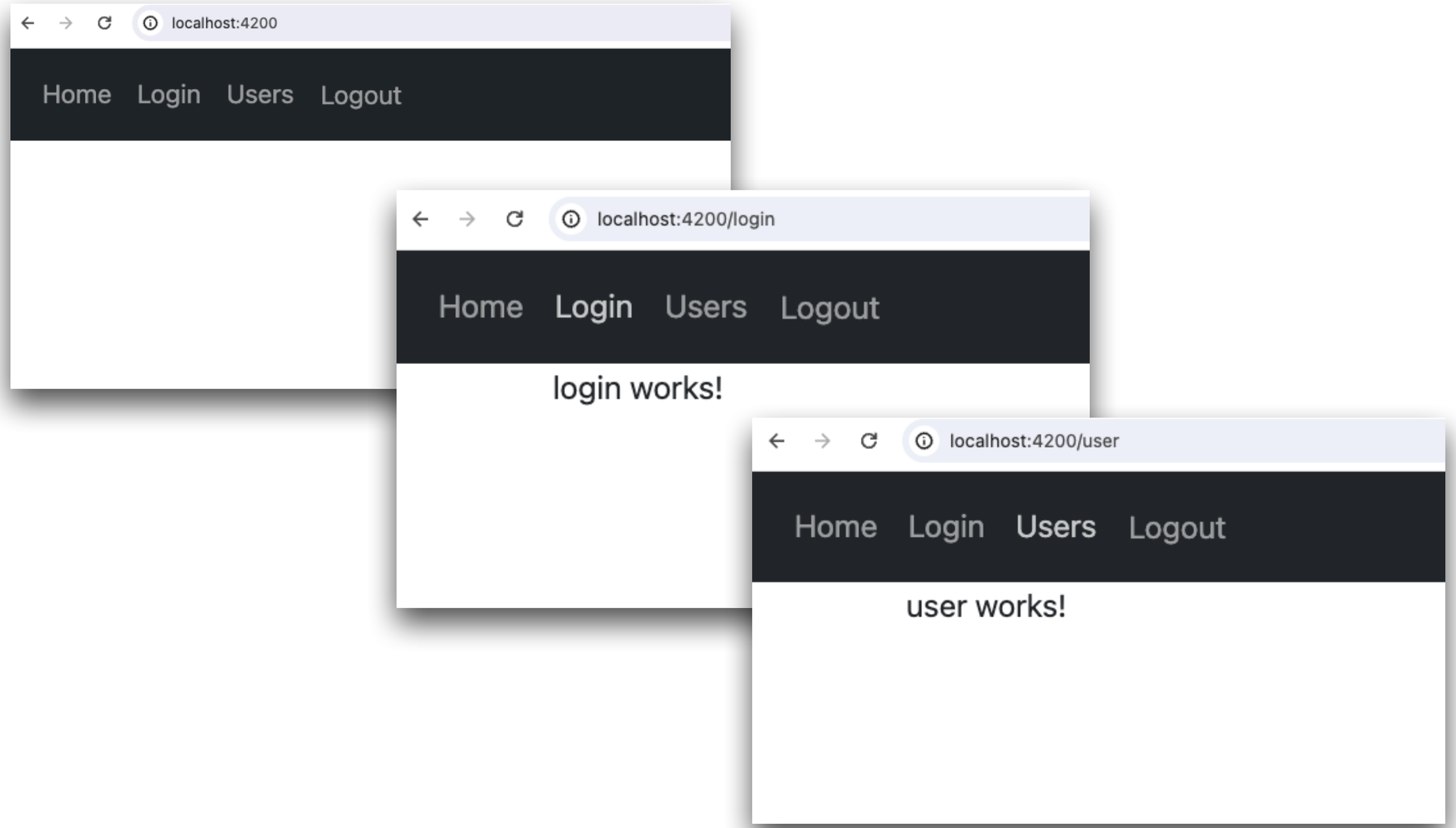
Id	First Name	Last Name	Email
1	User 1	Last name 1	user1@gmail.com
2	User 2	Last name 2	user2@gmail.com
3	User 3	Last name 3	user3@gmail.com

Config routing

Edit file app.routes.ts

```ts
import { Routes } from '@angular/router';
import { LoginComponent } from './login/login.component';
import { UserComponent } from './user/user.component';

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

See result



Angular Component Style

Global	Component
index.html	

https://angular.io/guide/component-styles

Structure of Project

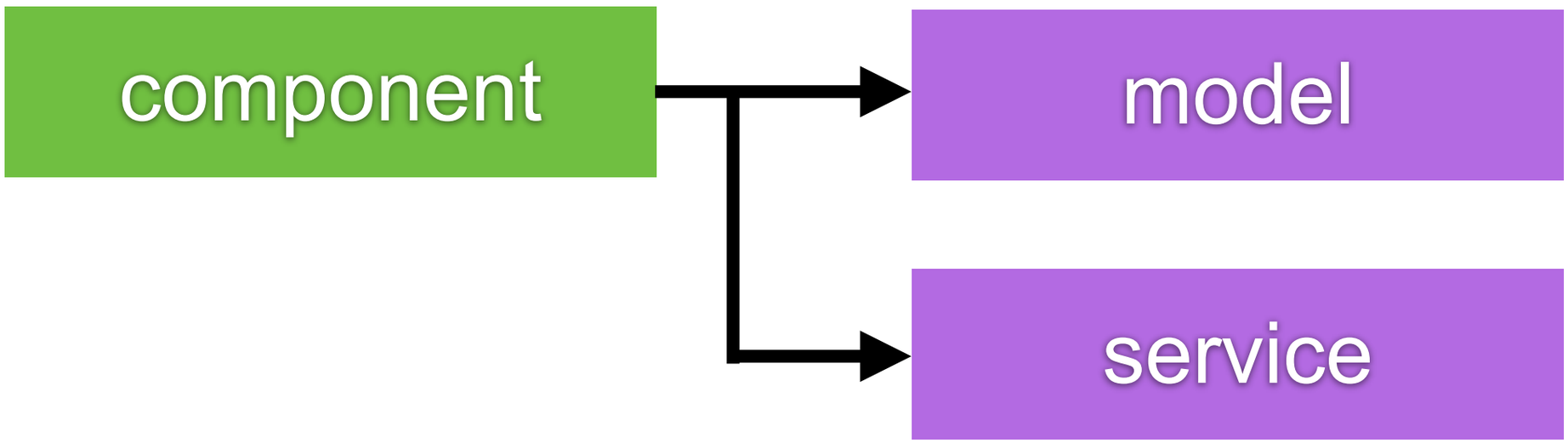
Types of project structure

Feature-based
Layer-based (roles and responsibilities)
Modular
Shared modules

Consistency

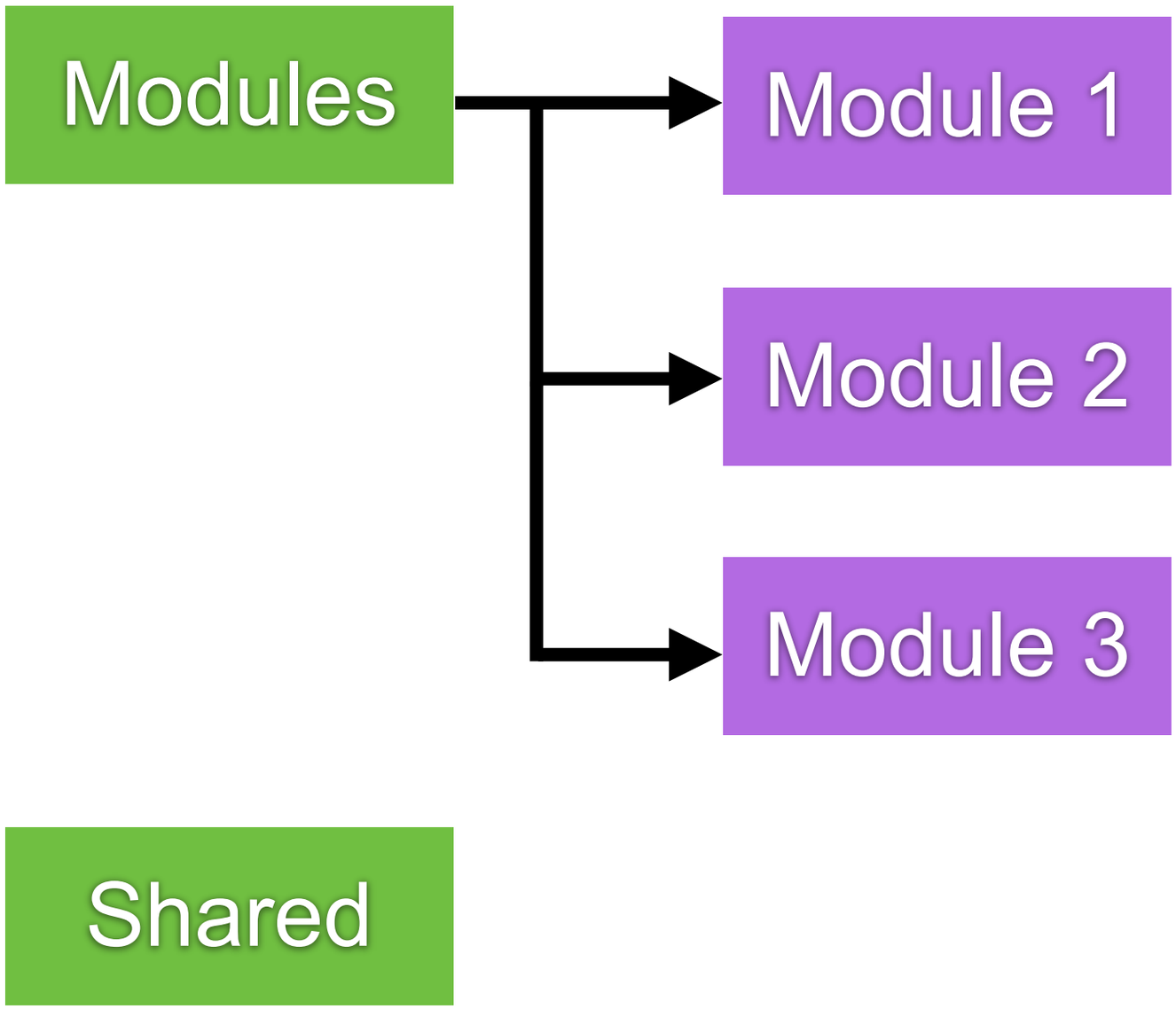
Maintainability

Better Structure of project

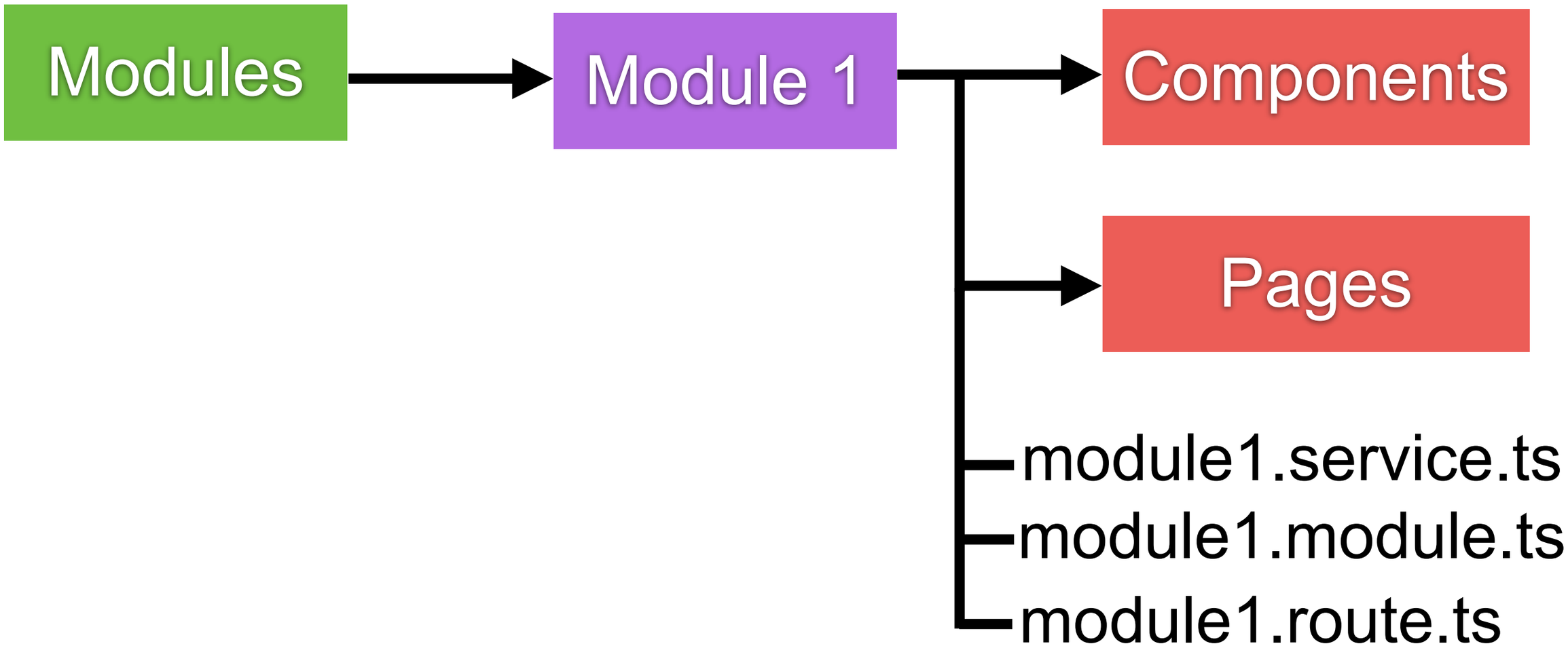


shared

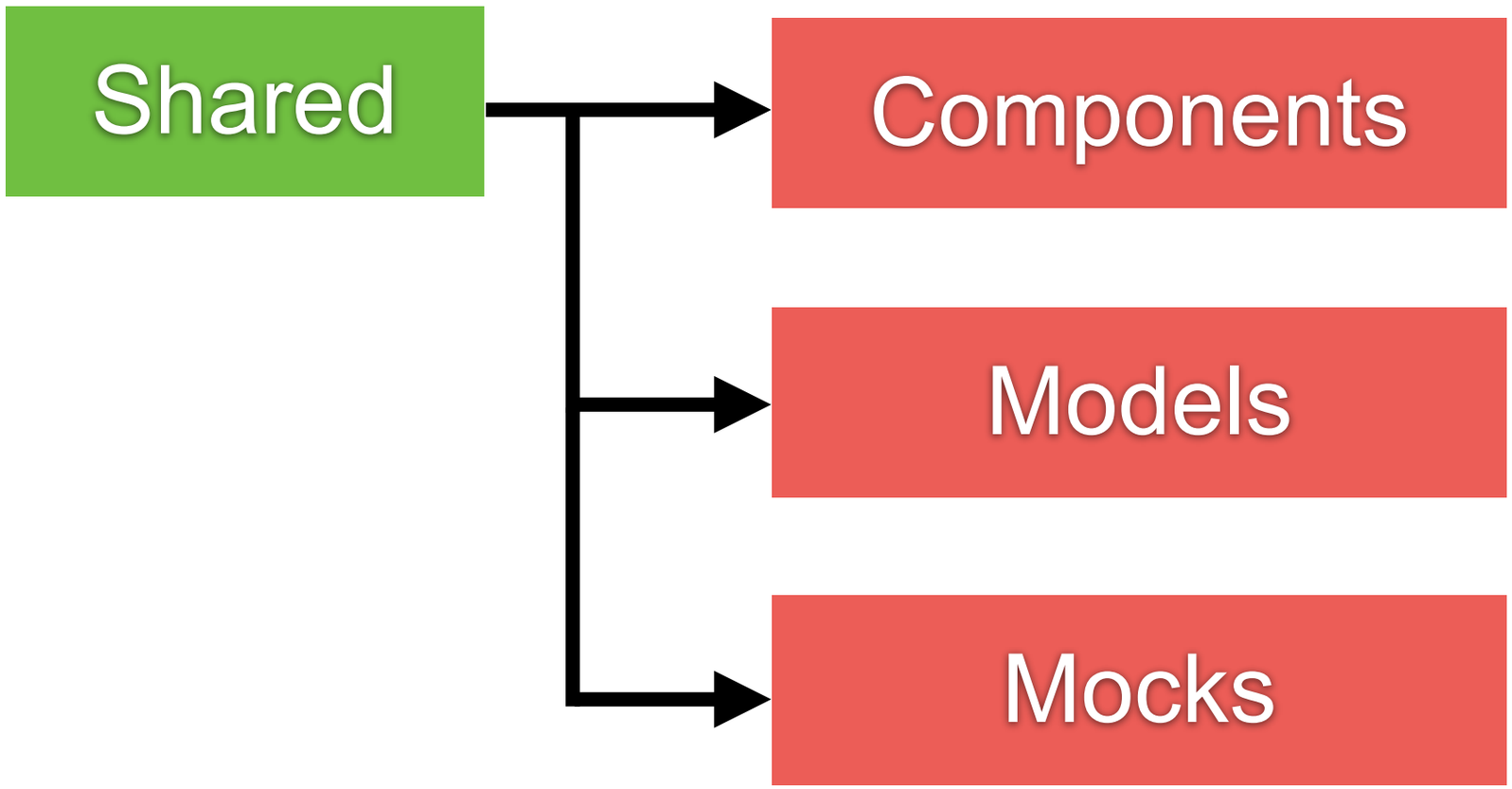
Better Structure of project



Module structure



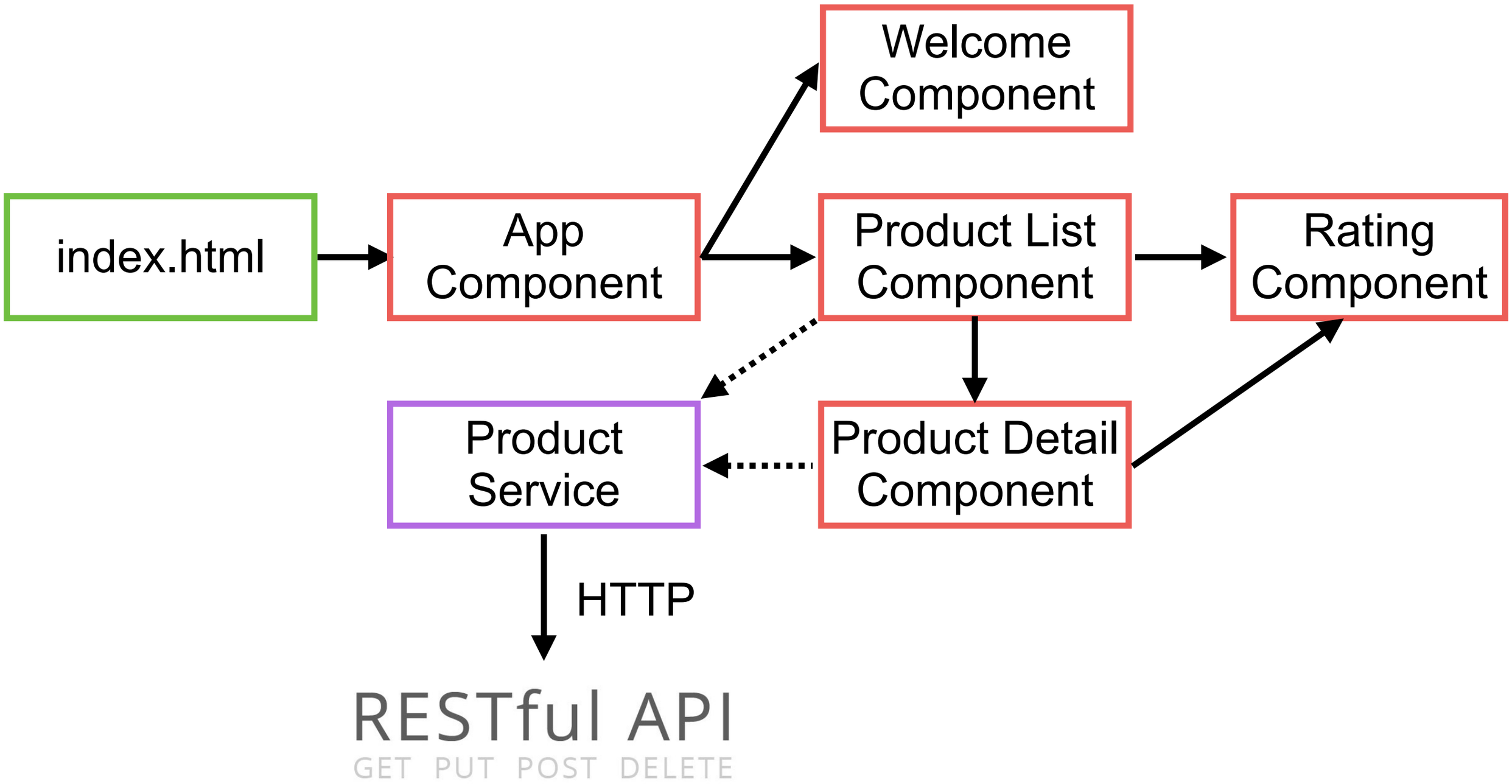
Shared



BEWARE...

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

Application Architecture



Create Product List page

Create product list

Create directory /app/products
Create file product-list.component.html

$ng **g**enerate **c**omponent products

Edit product-list-component.html

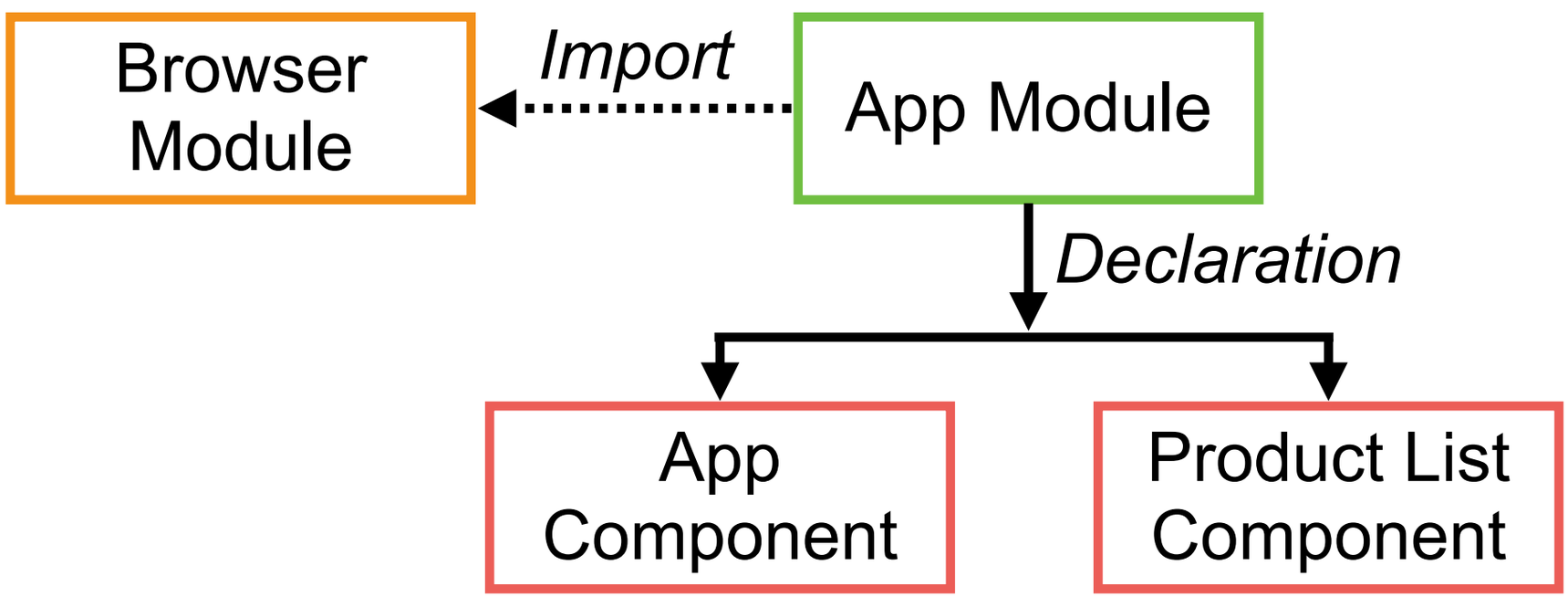
List of Product Page

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	****

Add component to App Module

Edit file /app/app.module.ts



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 { }
```

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 ?

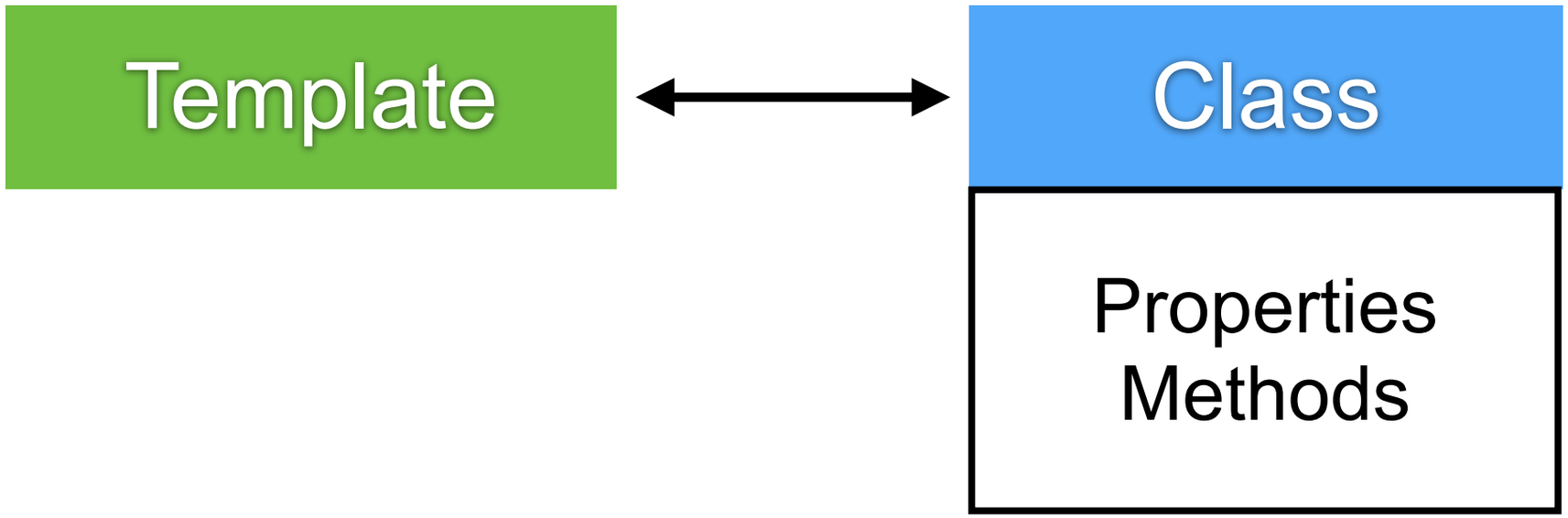
Custom HTML element or attribute use to power up and extend HTML

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'

 *ngIf="products && products.length"

>
```

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) i
  ▼ 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

Signal

More ..

Property binding
Handling events with event binding
Handling input with 2-way binding

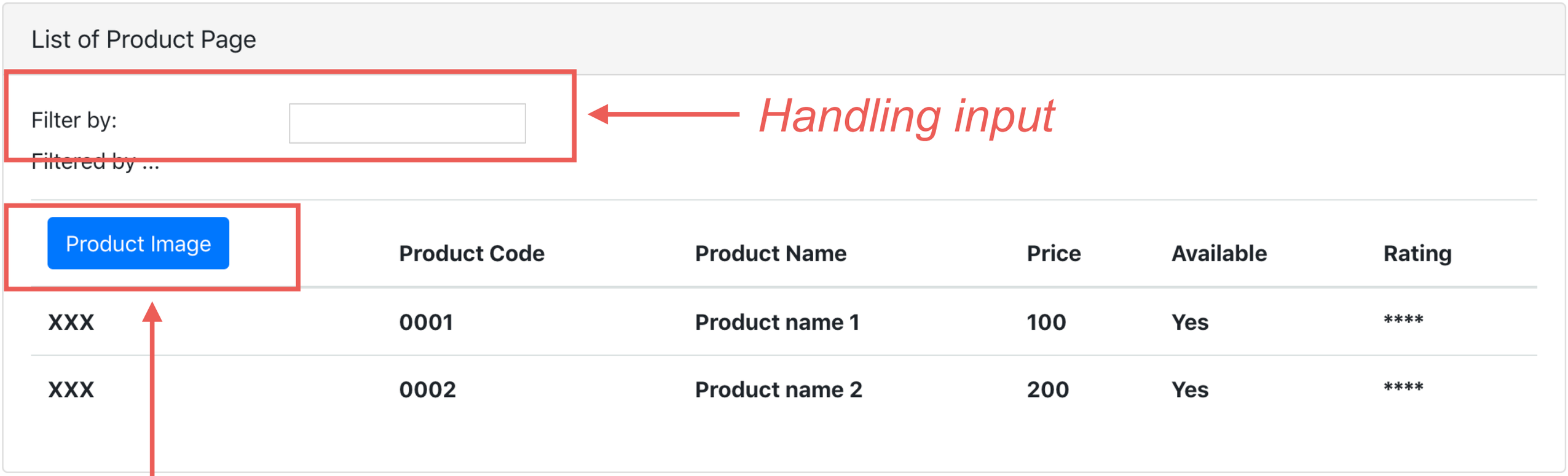
Property binding

Property binding

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

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

List of product



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 = '';
```

1

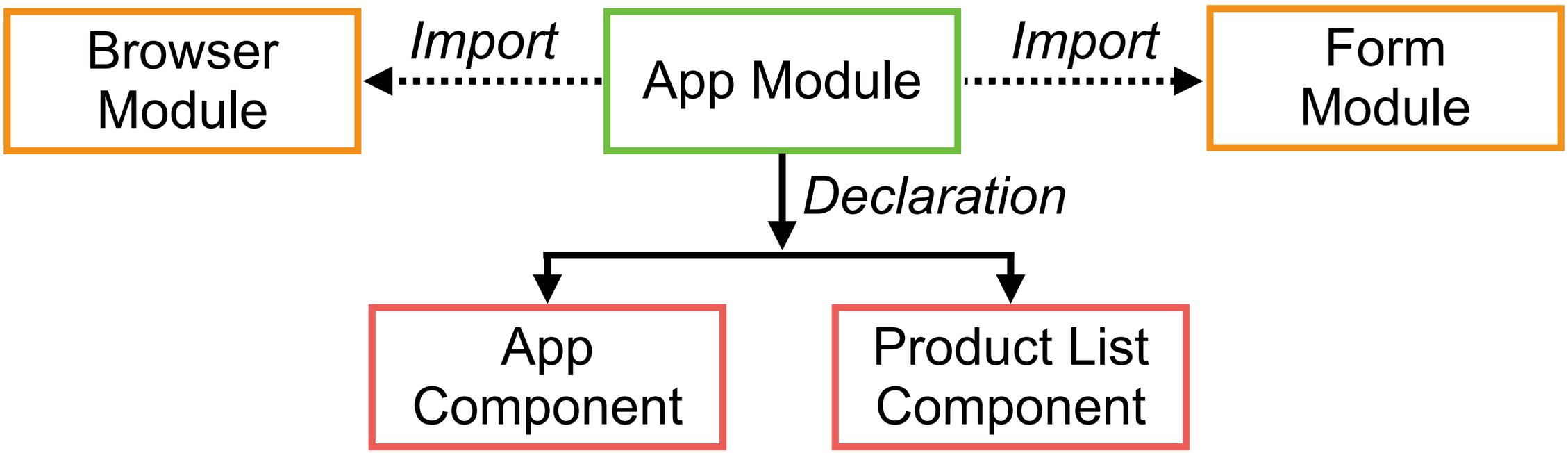
2

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

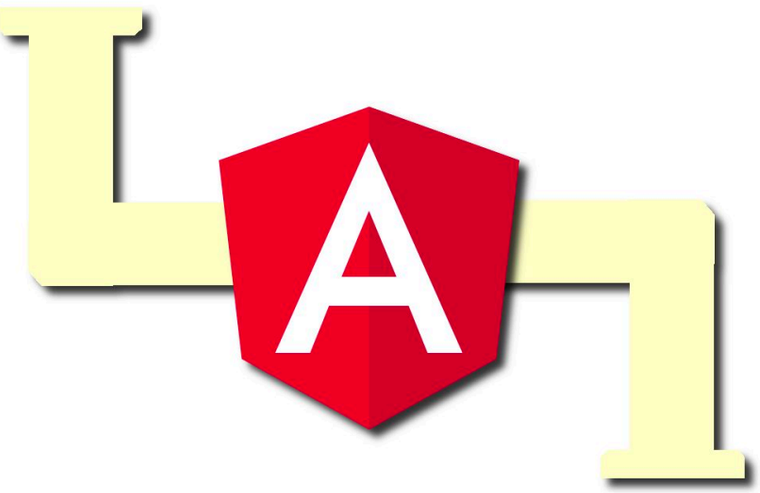


Add FormModule to AppModule

Edit file /app/app.module.ts

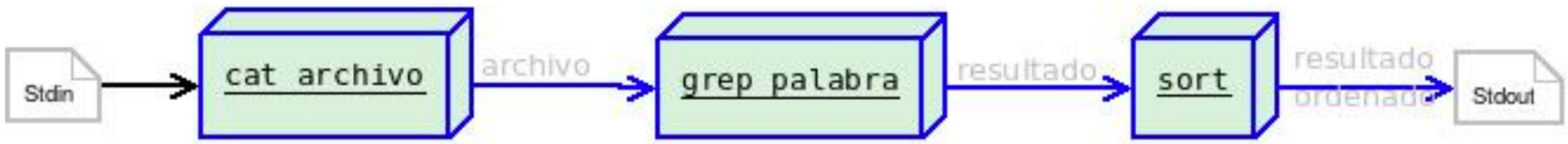
```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

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 { }
```

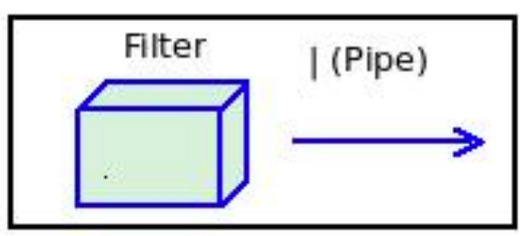
Transform data with Pipe



https://angular.io/guide/pipes-overview

```
$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

https://angular.io/guide/pipes-overview#built-in-pipes

Build-in pipes

Date
Lowercase
Uppercase
Currency
Percent

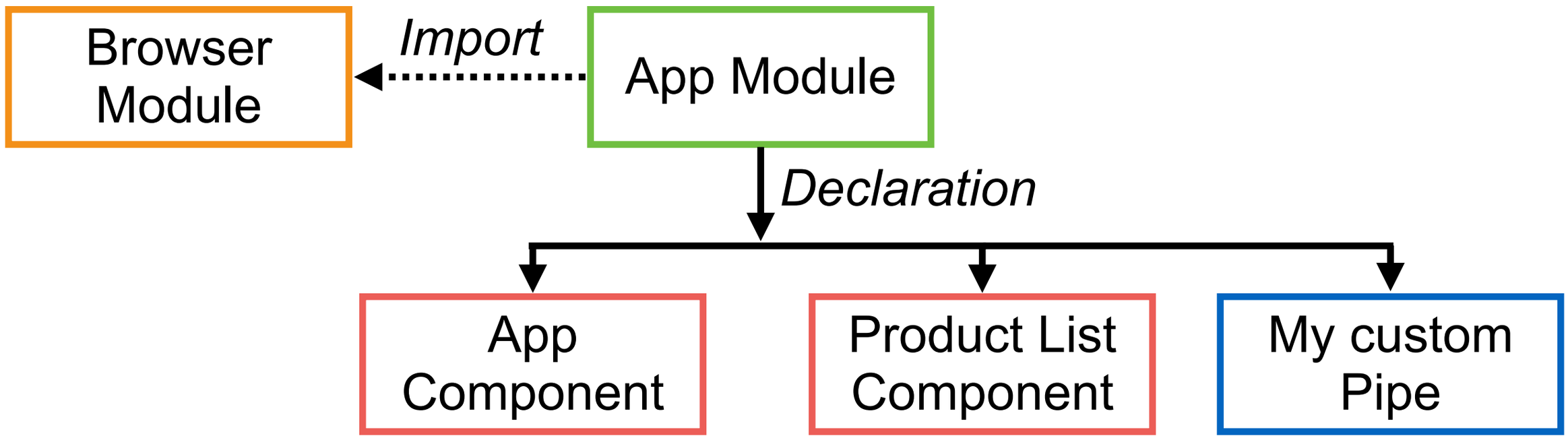
https://angular.io/guide/pipes-overview#built-in-pipes

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 **g**enerate **p**ipe <pipe name>



Create custom pipes

$ng **g**enerate **p**ipe ReplaceWithDash

replace-with-dash.pipe.ts

```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 **g**enerate **p**ipe ReplacewithDash

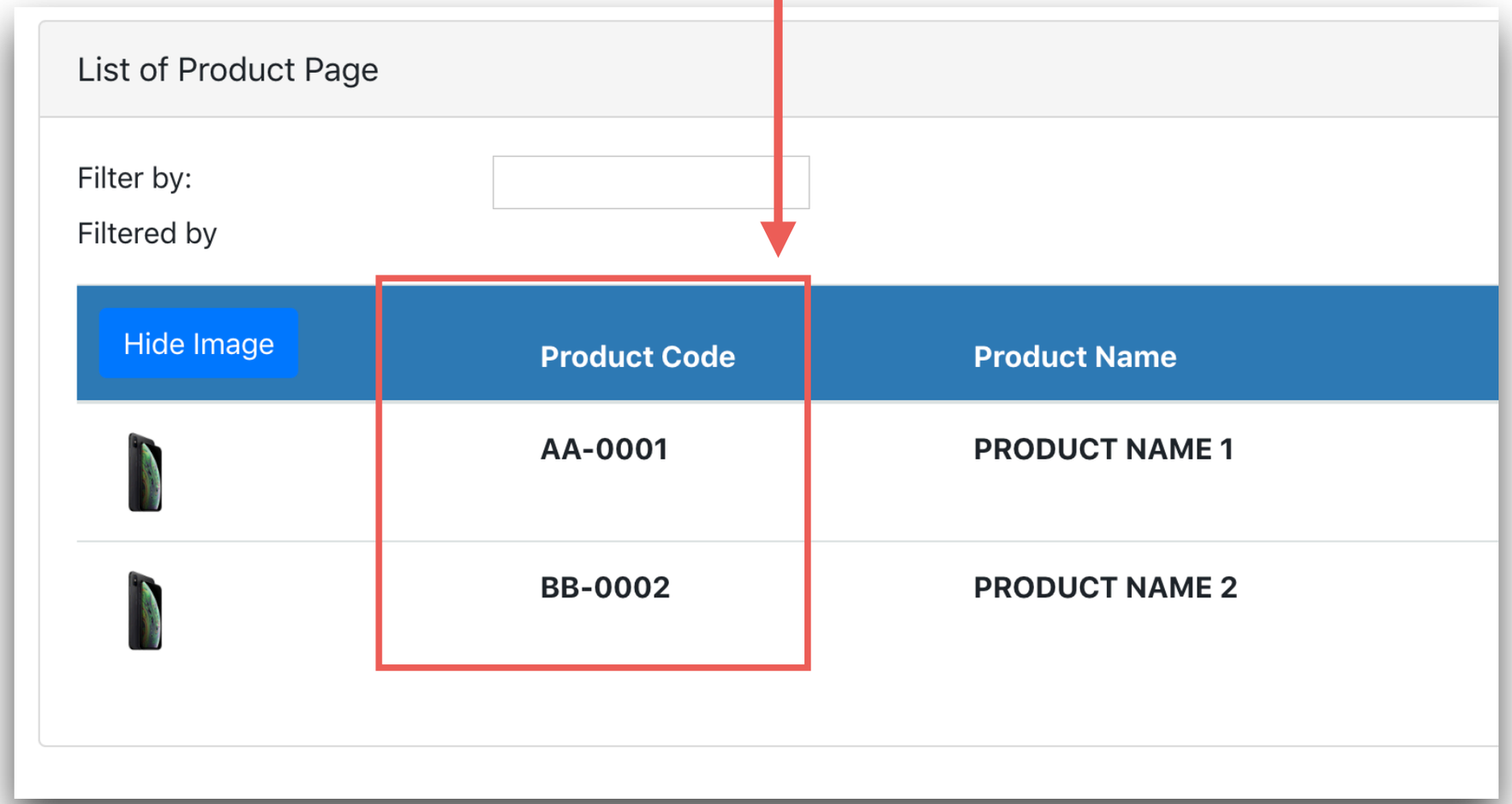
app.module.ts

```
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>
```



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.slatic.net/(
  },
  {
    code: '0002',
    name: 'Product name 2',
    price: 2000,
    imageUrl: 'https://th-live-01.slatic.net/(
  }
];
```

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

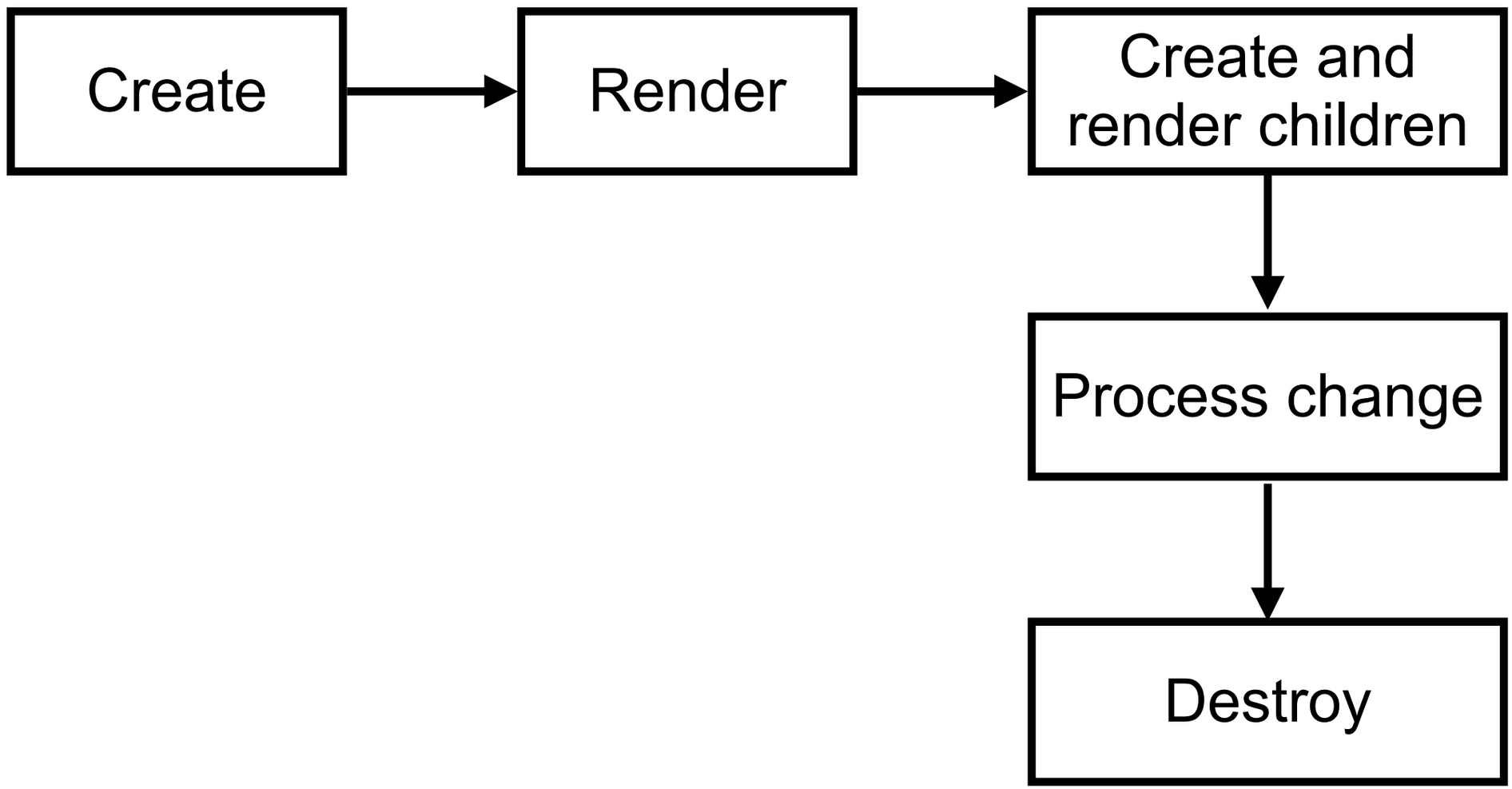
Styles

```
@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);
  }

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

}
```

2. Implement Hooks interface

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

```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

Create filteredProduct to keep result

```
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;
  }
```

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

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

Hide Image	Product Code	Product Name	Price	Available	Rating

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

Try to run

List of Product Page

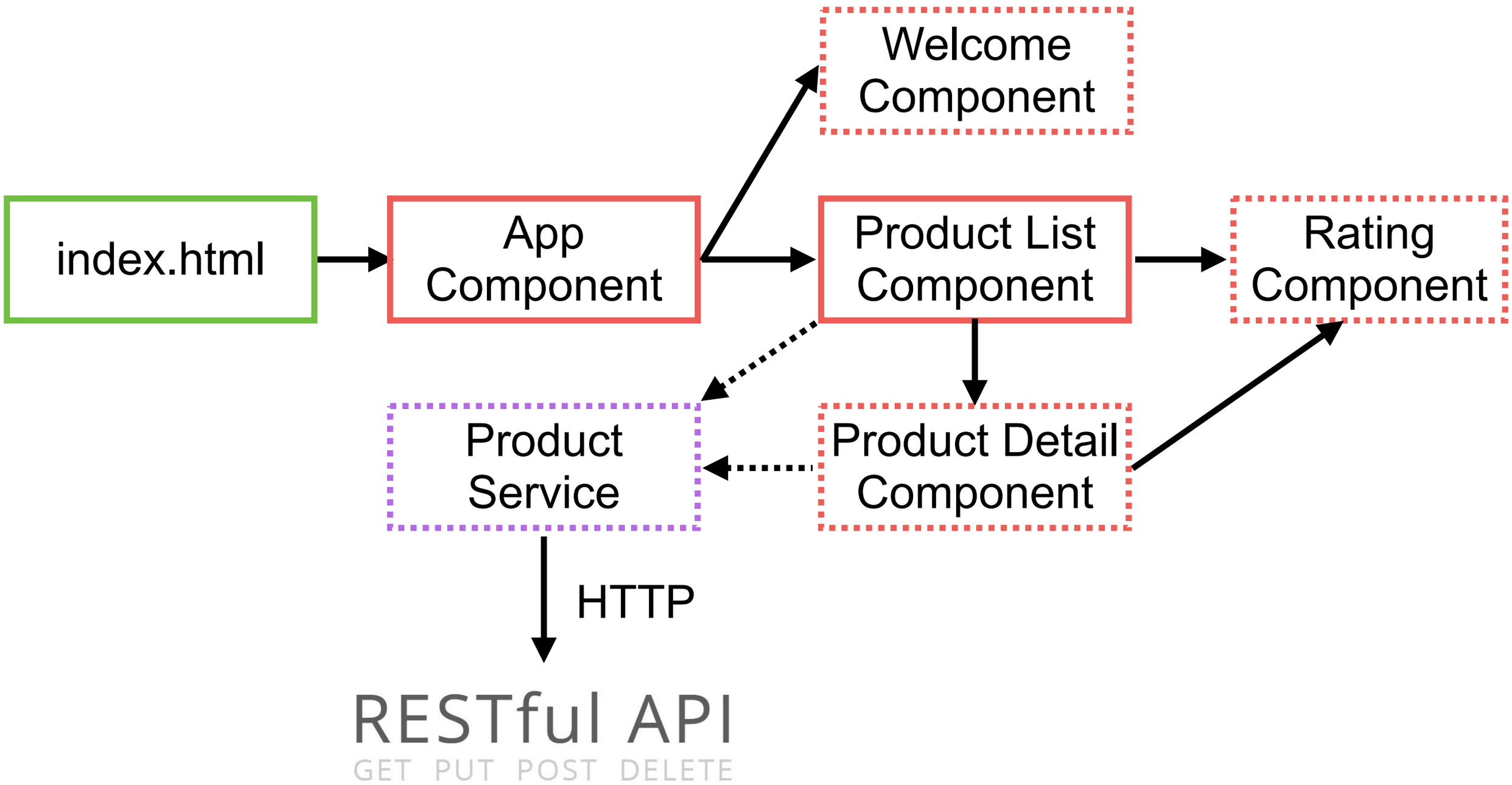
Filter by:

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	****

Nested components

Application Architecture



Rating component

List of Product Page

Filter by:

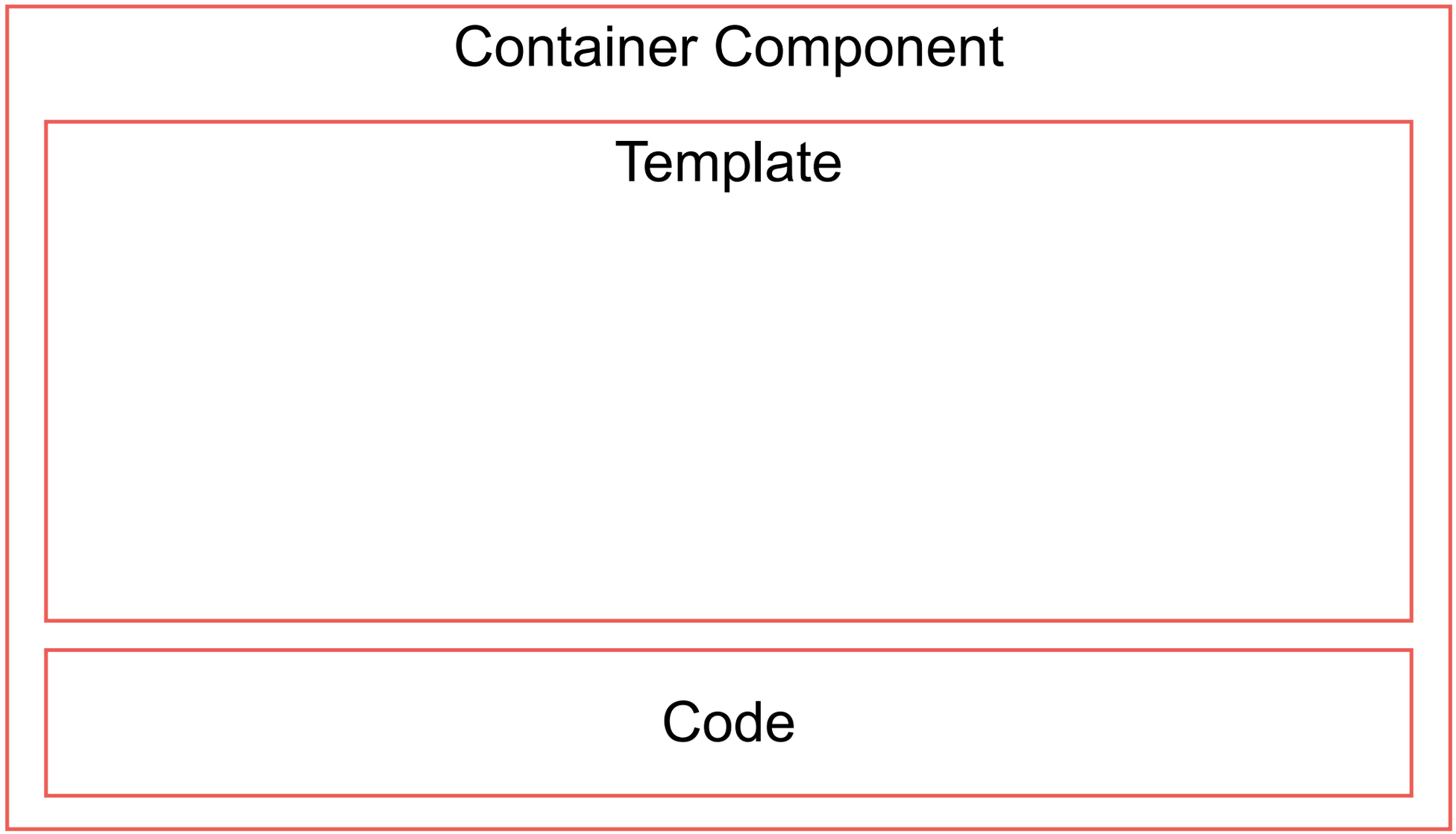
Filtered by

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	****

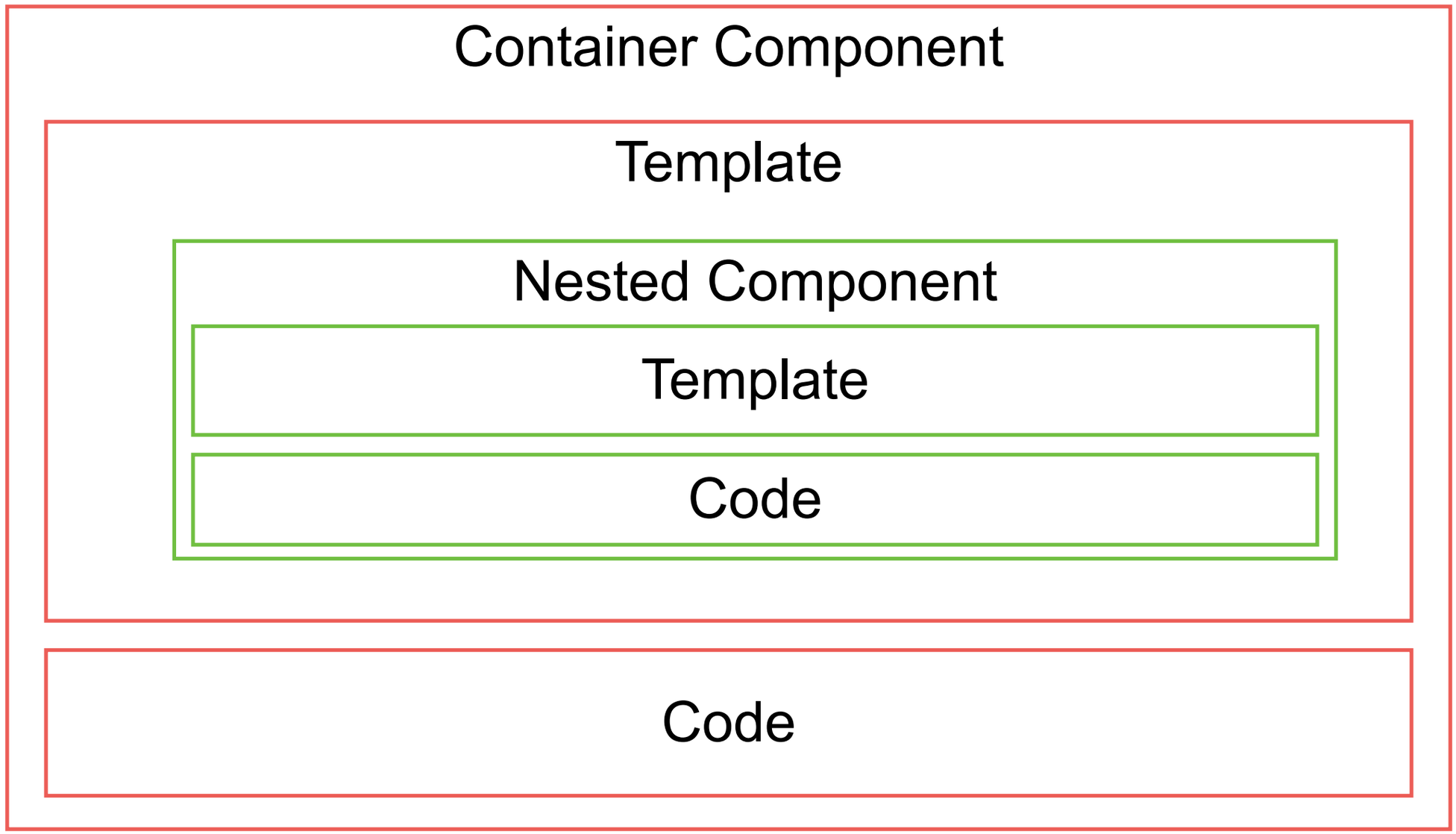
Nested component

Building component
Using component
Passing data to component (@Input)
Raising event from component (@Output)

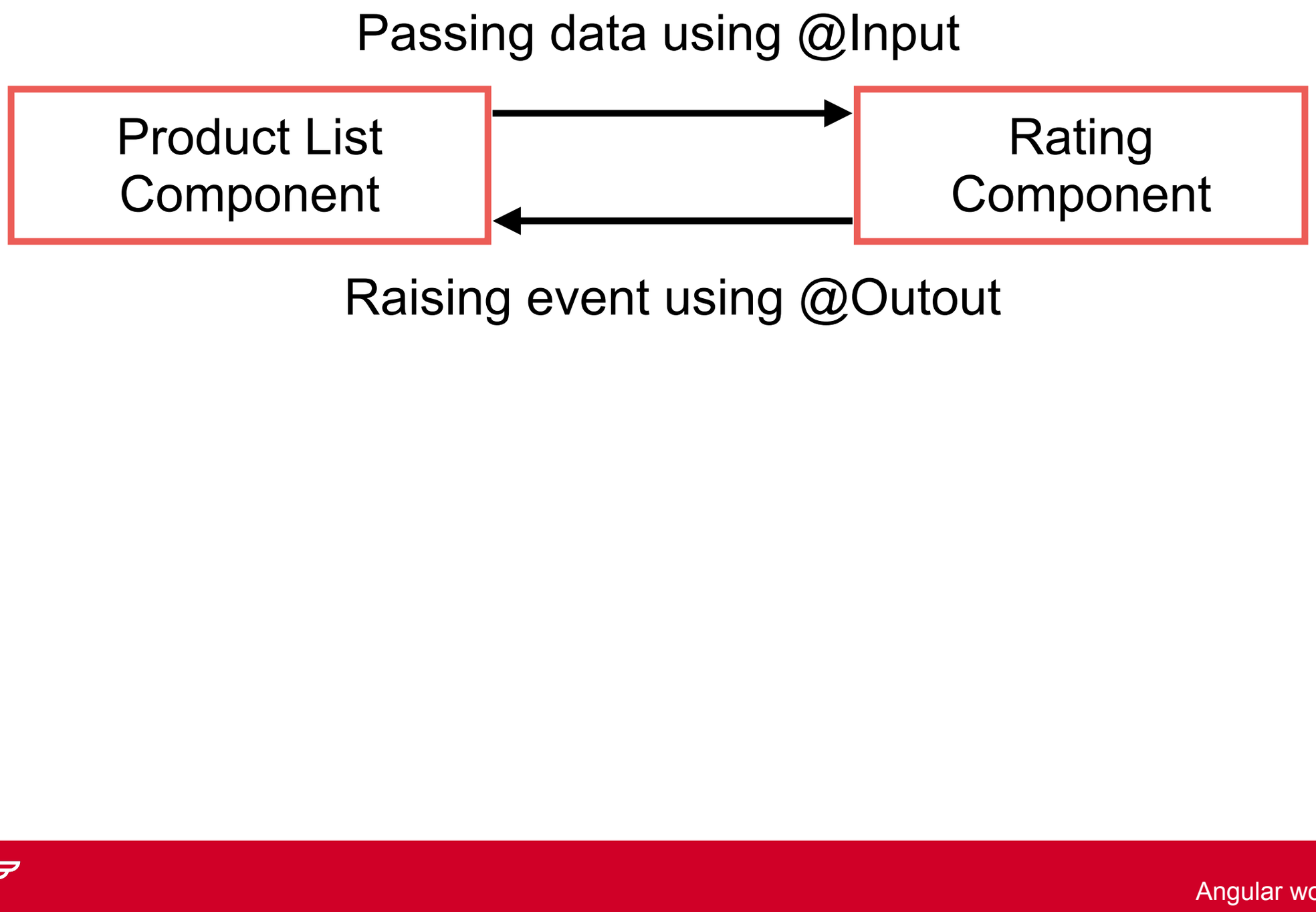
Nested component



Nested component



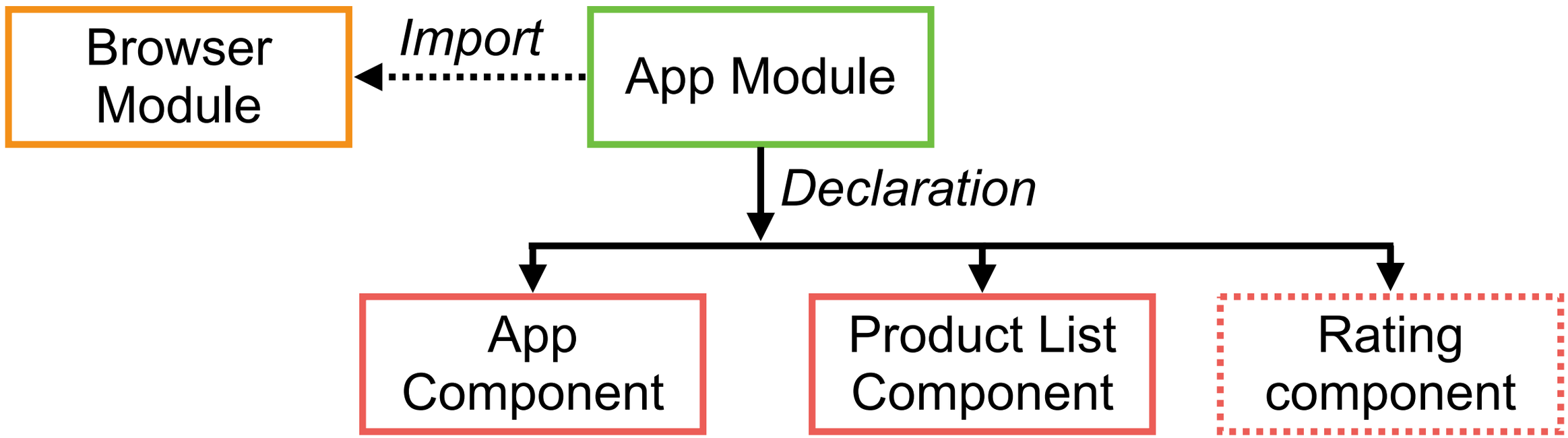
Nested component



Building component

1. Create rating component

$ng **g**enerate **c**omponent rating



1. Create rating component

$ng **g**enerate **c**omponent rating

Rating.component.ts

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

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

}
```

→ *Directive name of rating component*

2. Add rating component in module

/app/app.module.ts

App.module.ts

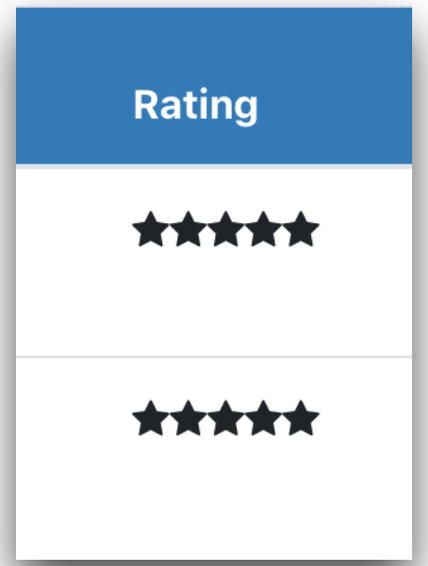
```ts
import { RatingComponent } from './rating/rating.component';

@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

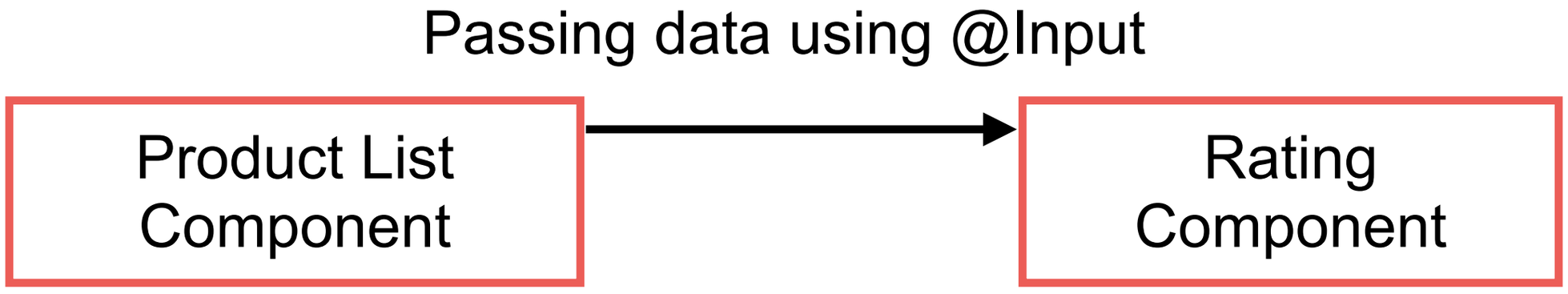
Filter by:

Filtered by

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	★★★★★

Passing data to component

Passing data to component



Passing data to component

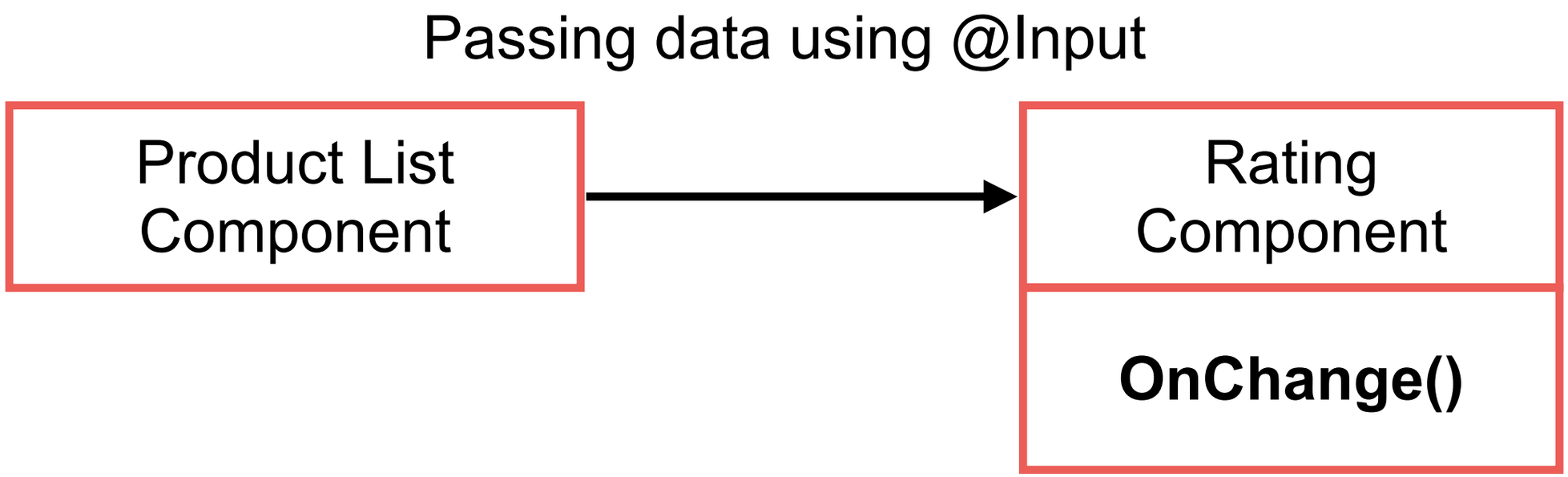
Product-list.component.html

```
<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



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

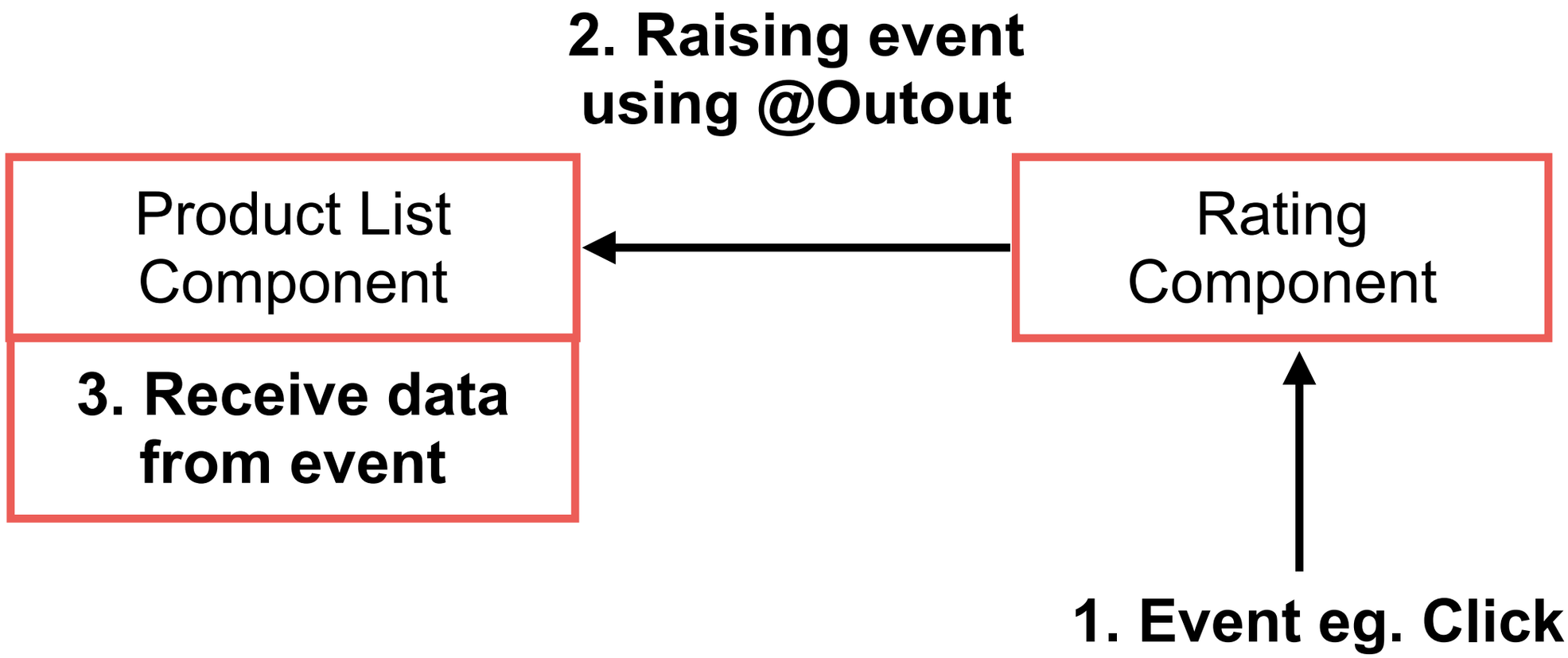
Filter by:

Filtered by

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	★★★½

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

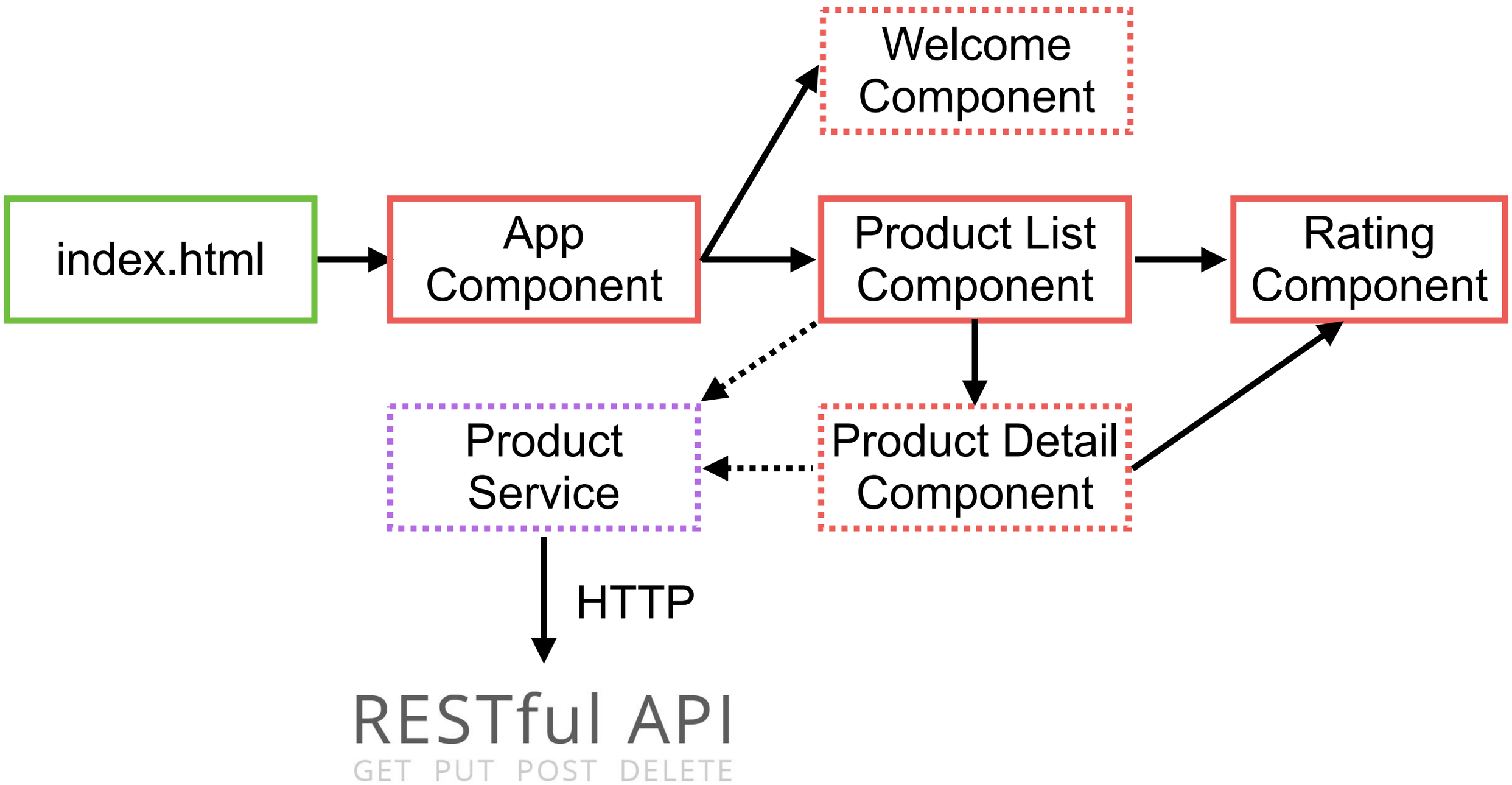
```
export class ProductListComponent implements OnInit {

   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
/products	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'>
```

Use configured routes

```
    <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>
```

Show content from component's view

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

Result

Home | Product List

List of Product Page

Filter by:

Filtered by

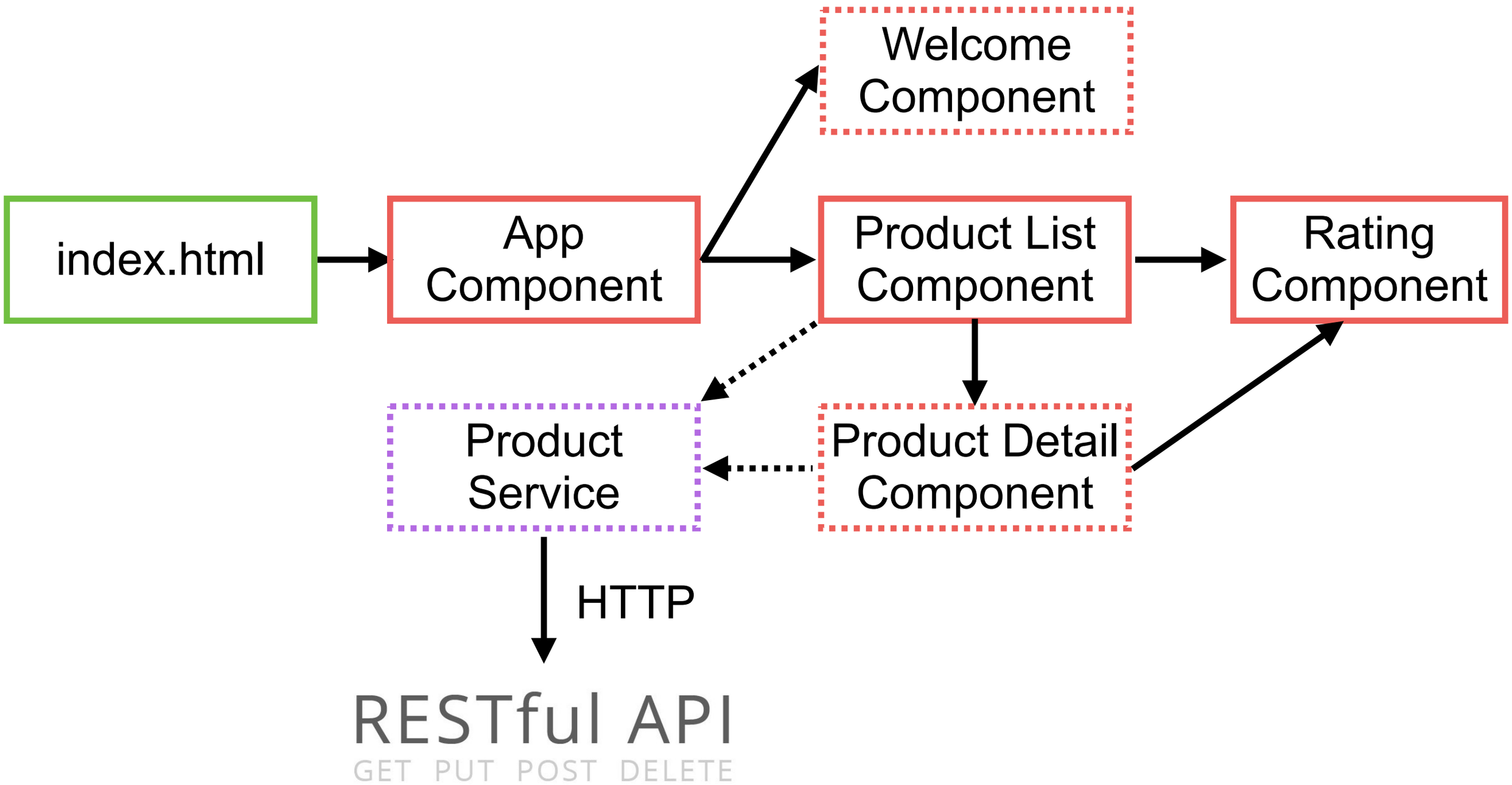
Hide Image	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	★★★★★
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	★★
	BB-0003	PRODUCT NAME 3	THB 2,000.00	Yes	★★★½

Working with route

Passing parameter
Activate route with code
Protect routes with **guards**

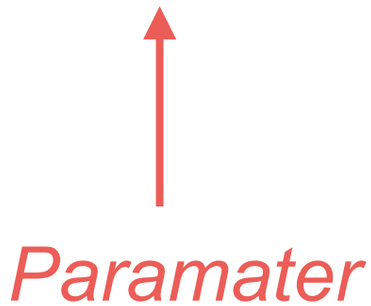
Passing parameters with route

Application Architecture



Product detail

URL = /product/:id



Create product detail component

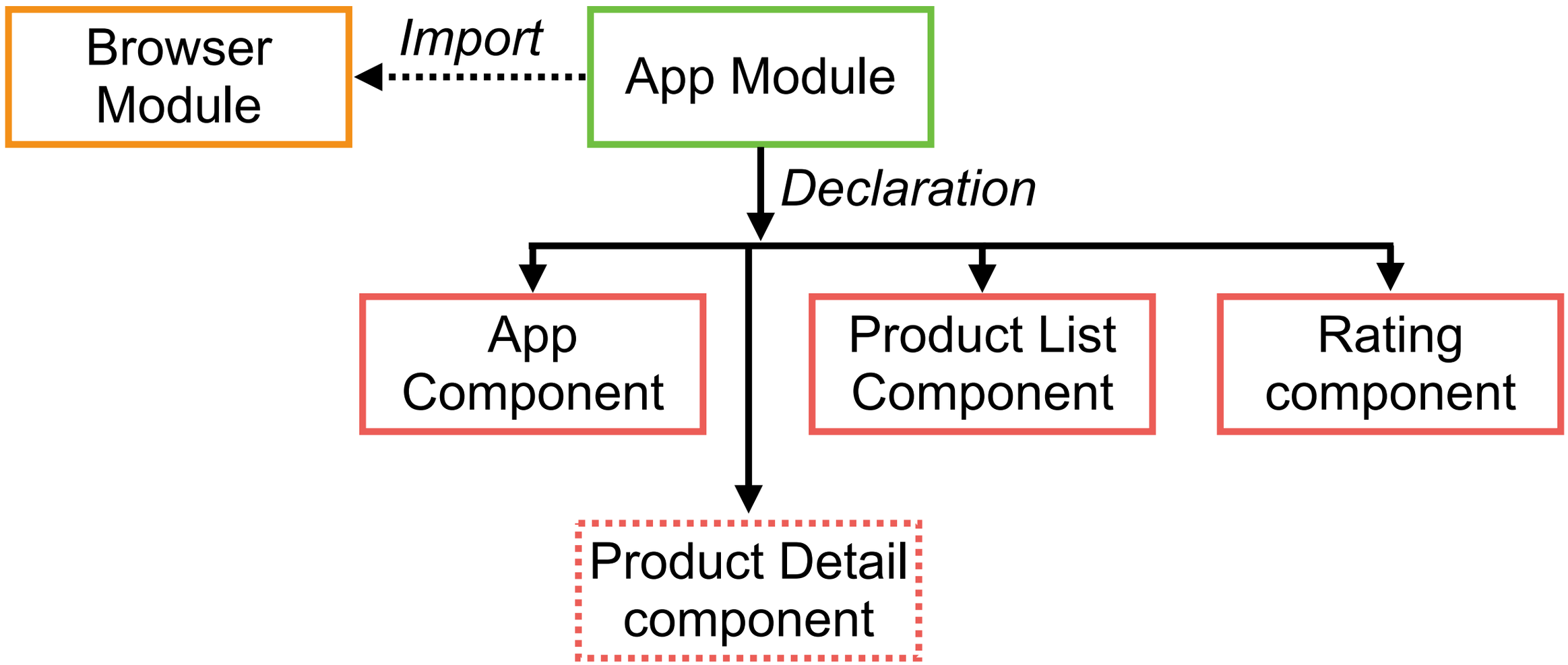
$ng **g**enerate **c**omponent productDetail

$ng **g**enerate **c**omponent product/productDetail --flat

Generate component with existing directory

Create product detail component

$ng **g**enerate **c**omponent 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:

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	★★★½

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

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	★★★½

Link to /product/:id

Show product detail

Mock data to show product detail

product-detail.component.ts

```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

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

export class ProductDetailComponent {


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

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

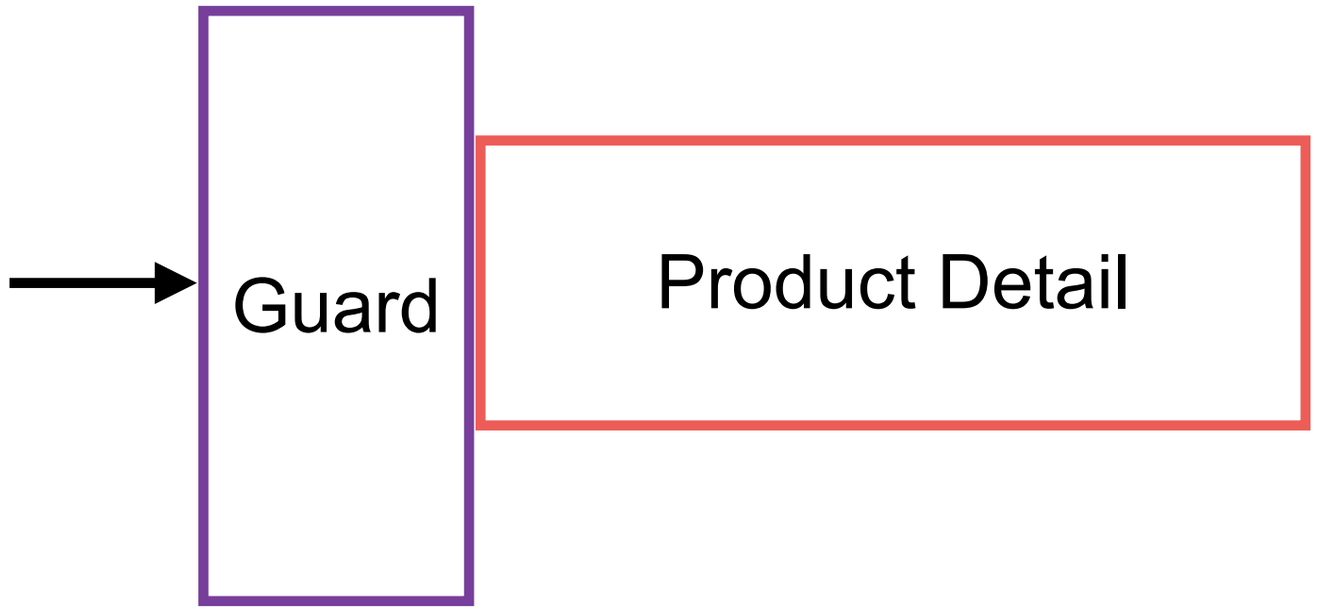
Protect route with guard

Protect route with guard

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

```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

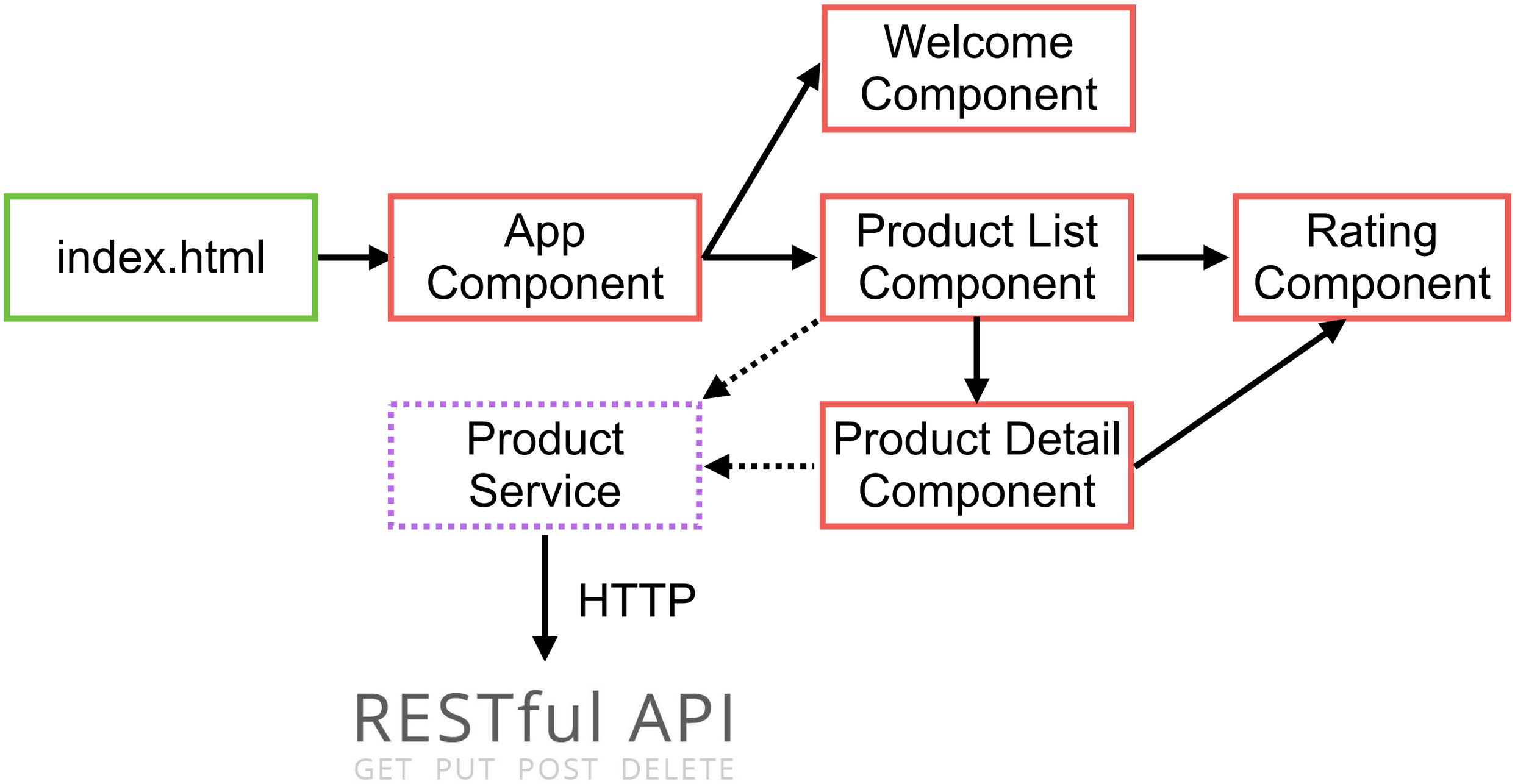
Service

Class with specified purpose
Independent from any component
Provide shared data and logic across components
Encapsulate external interactions

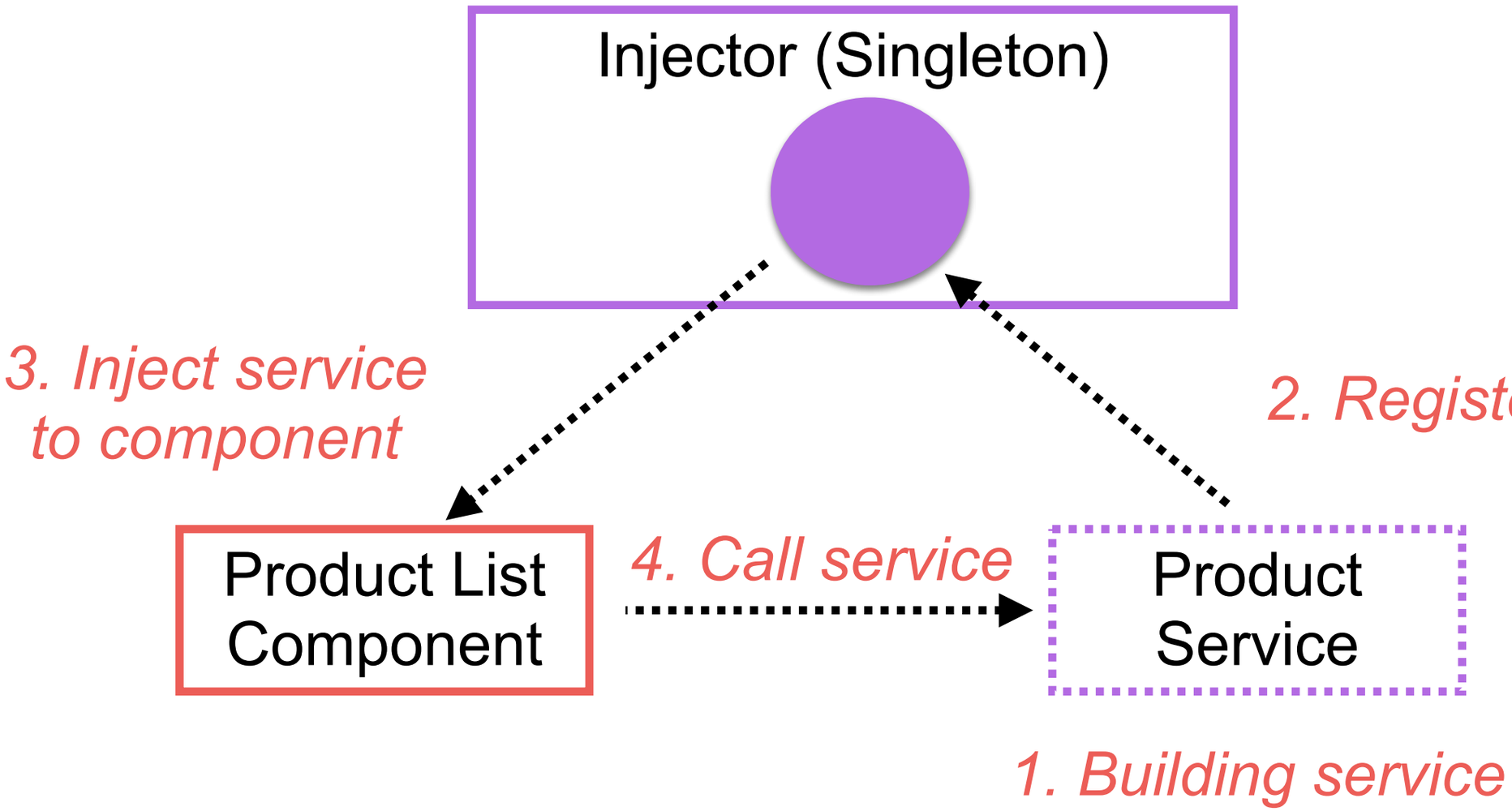
Service

How service working ?
Building service
Register service
Inject service

Application Architecture



How service working ?



Building service

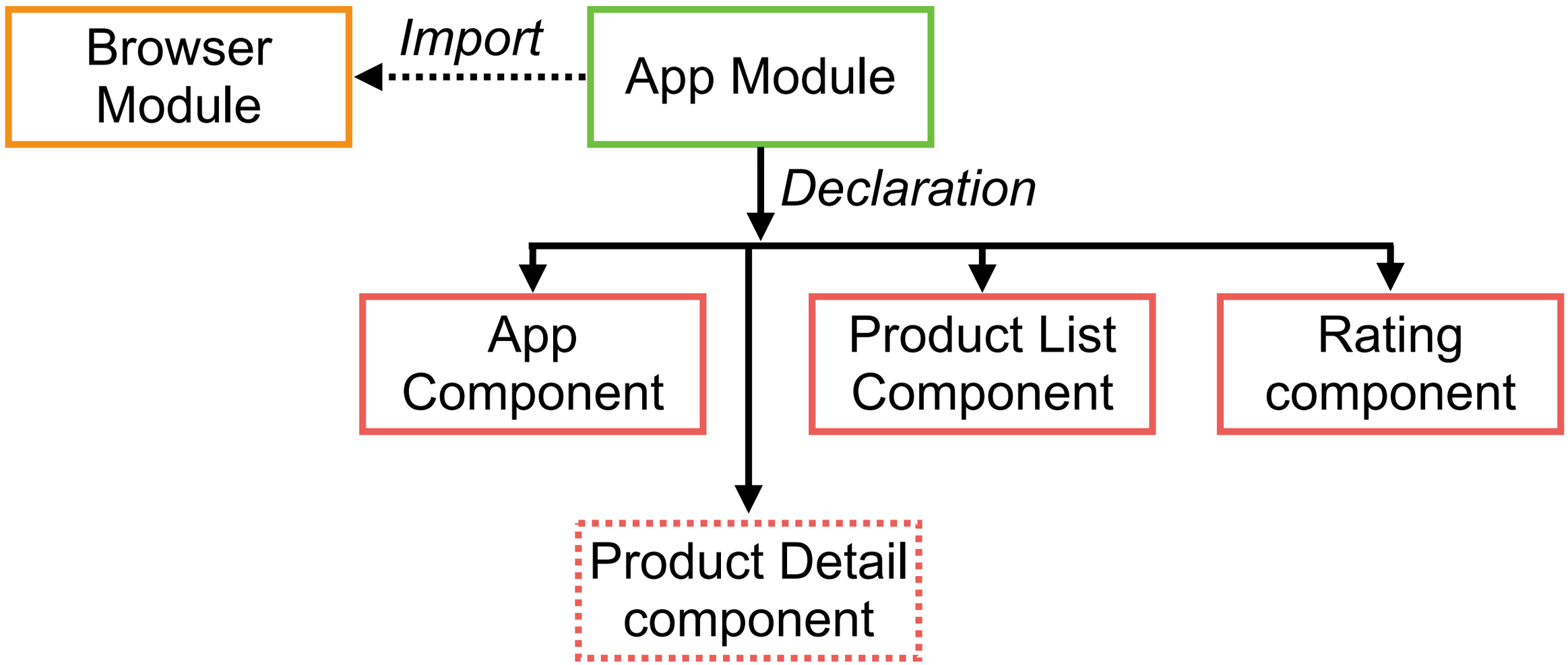
$ng generate service products/product

product.service.ts

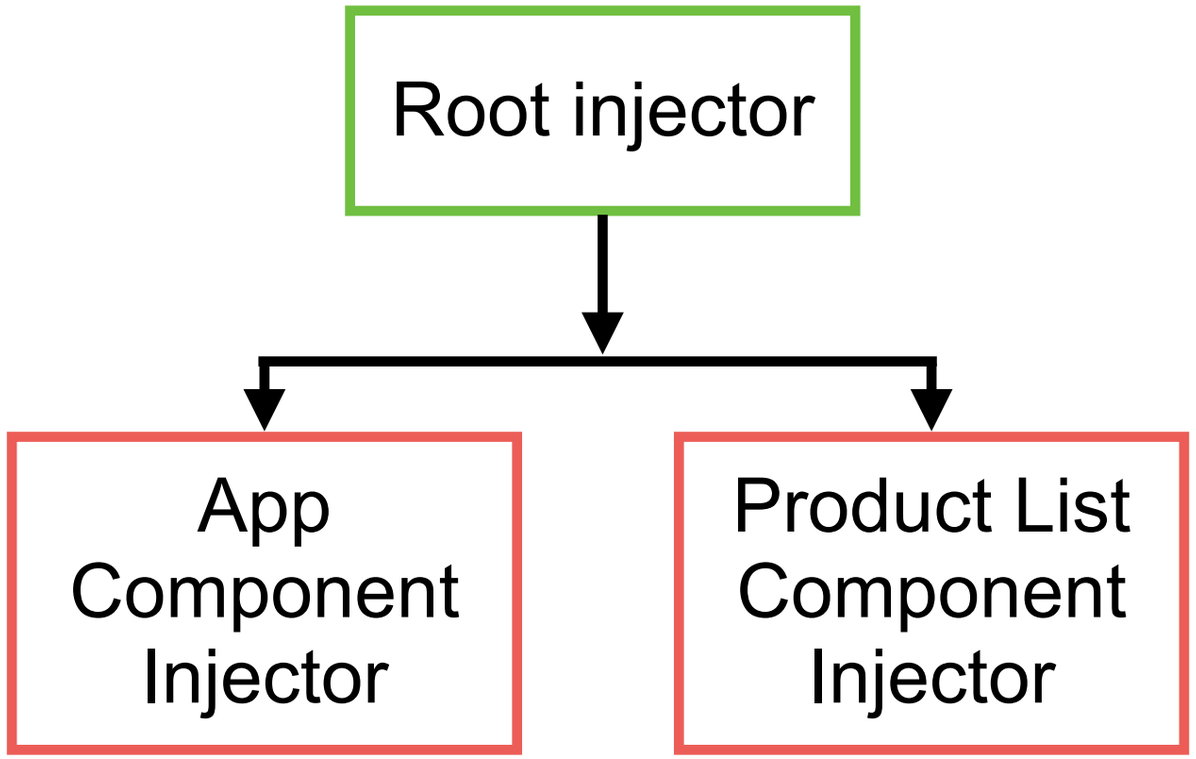
```
import { Injectable } from '@angular/core';
import {ProductDataModel} from './product';

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

  constructor() { }

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

Create product detail component

$ng **g**enerate **c**omponent productDetail



Register service



Register service

Root injector
Component injector

product.service.ts

```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 productSerive: ProductService) {
    console.log(this.products);
  }

  ngOnInit(): void {
    this.products = this.productService.getProducts();
    this.filteredProducts = this.products;
  }
```

Workshop with HTTP client

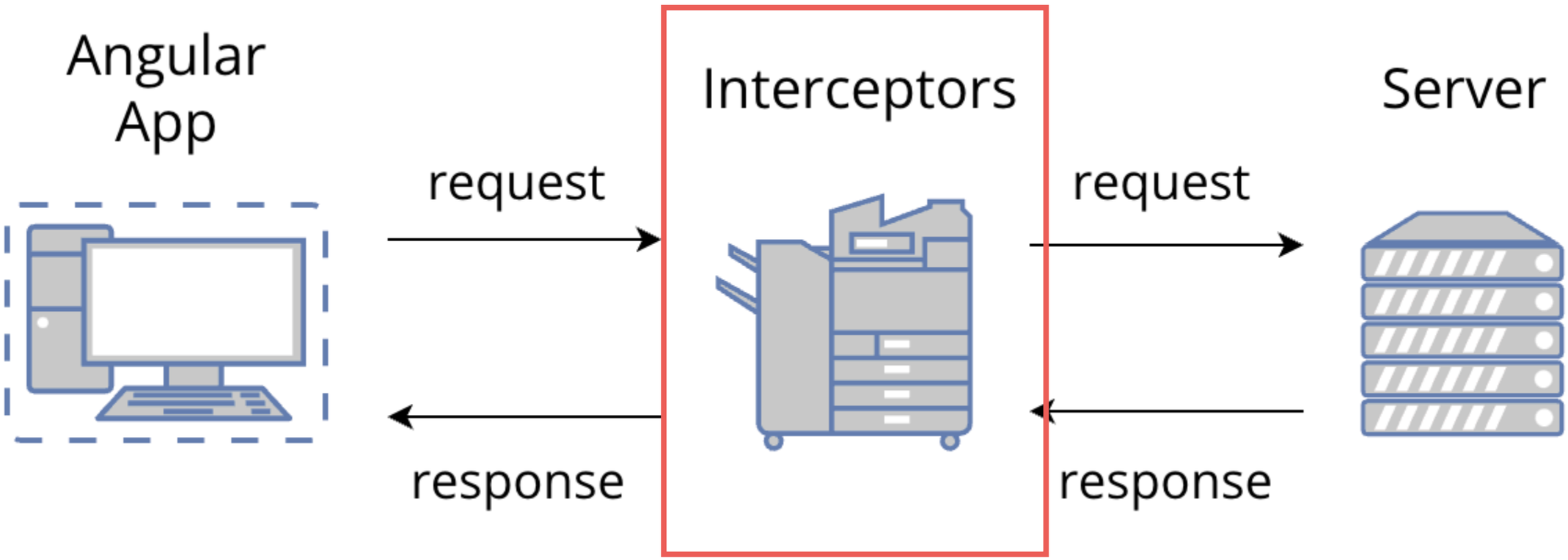
Dependency Injection

Interceptor

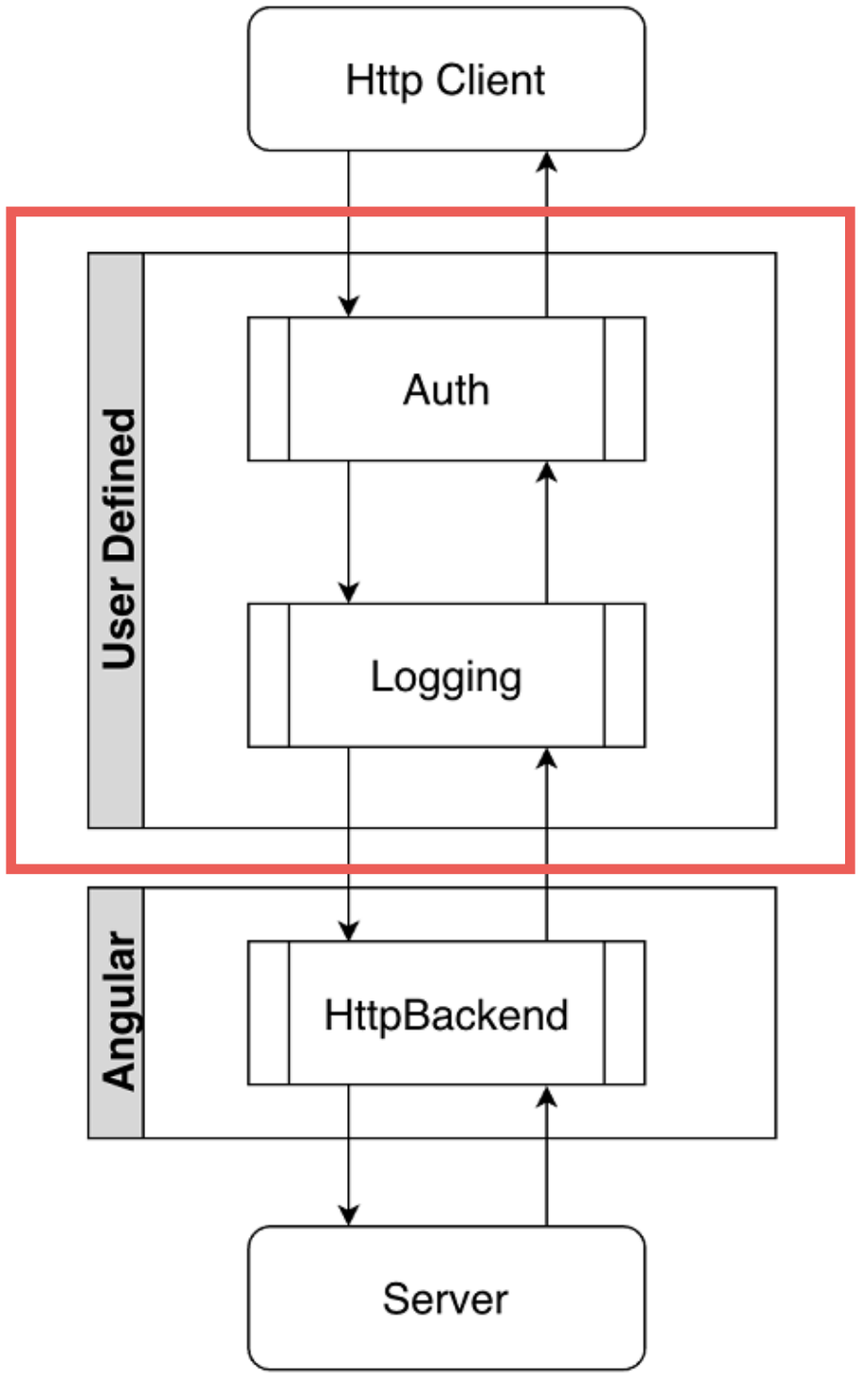
Error handling

https://angular.io/guide/understanding-communicating-with-http

Structure



Interceptor



Gernerate Interceptor

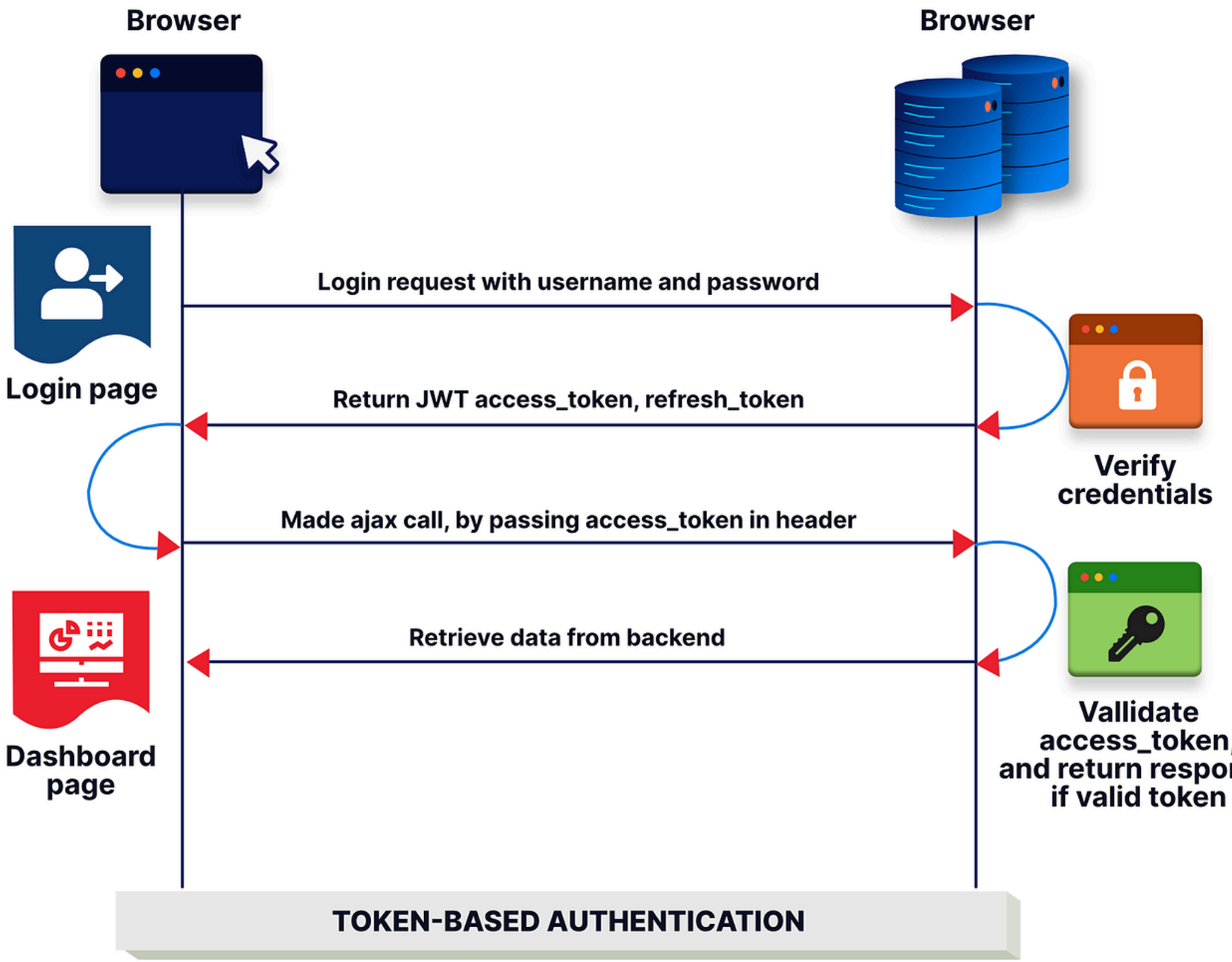
$ng generate interceptor

```ts
import { HttpInterceptorFn } from '@angular/common/http';

export const demoInterceptor: HttpInterceptorFn = (req, next) => {
  return next(req);
};
```

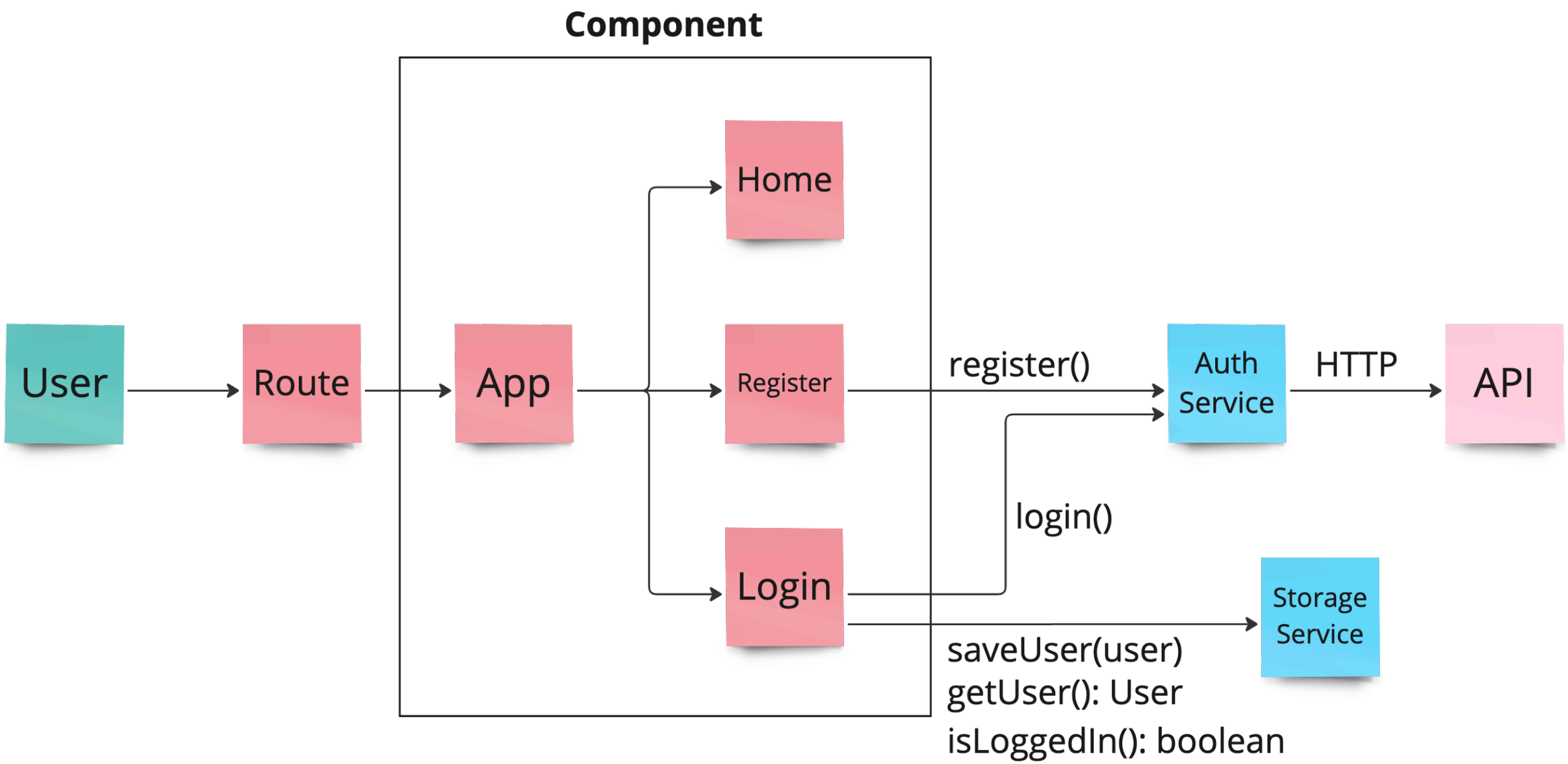
Workshop

Authentication and Authorization

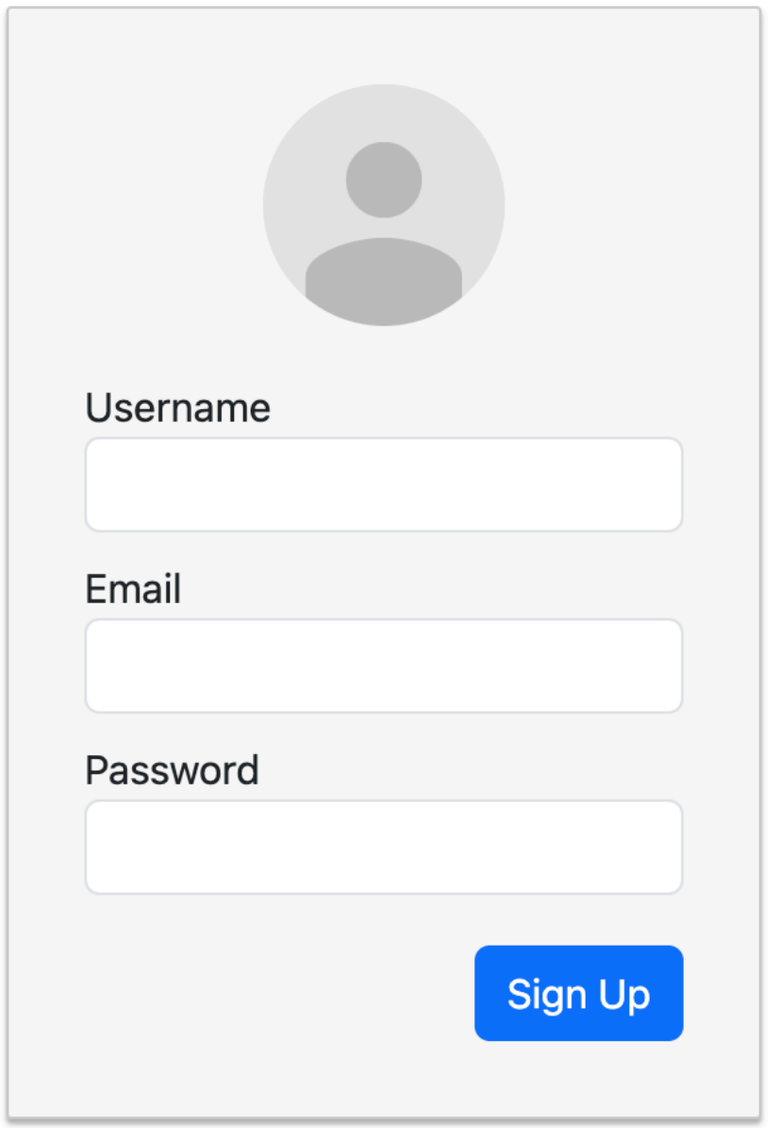
Structure



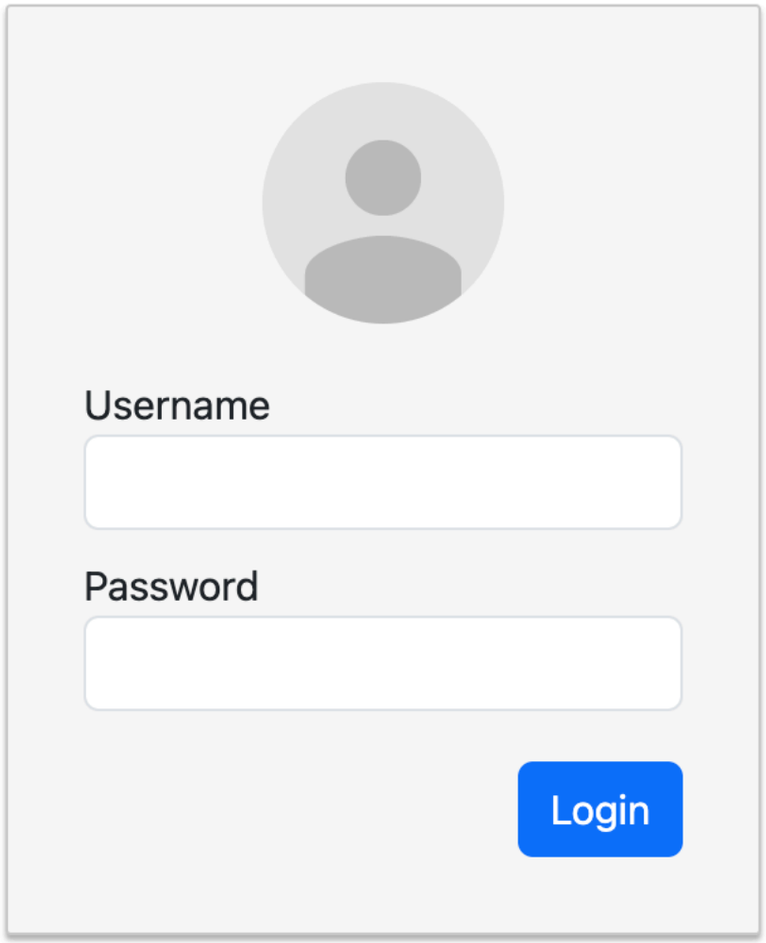
Flow of system



Register form



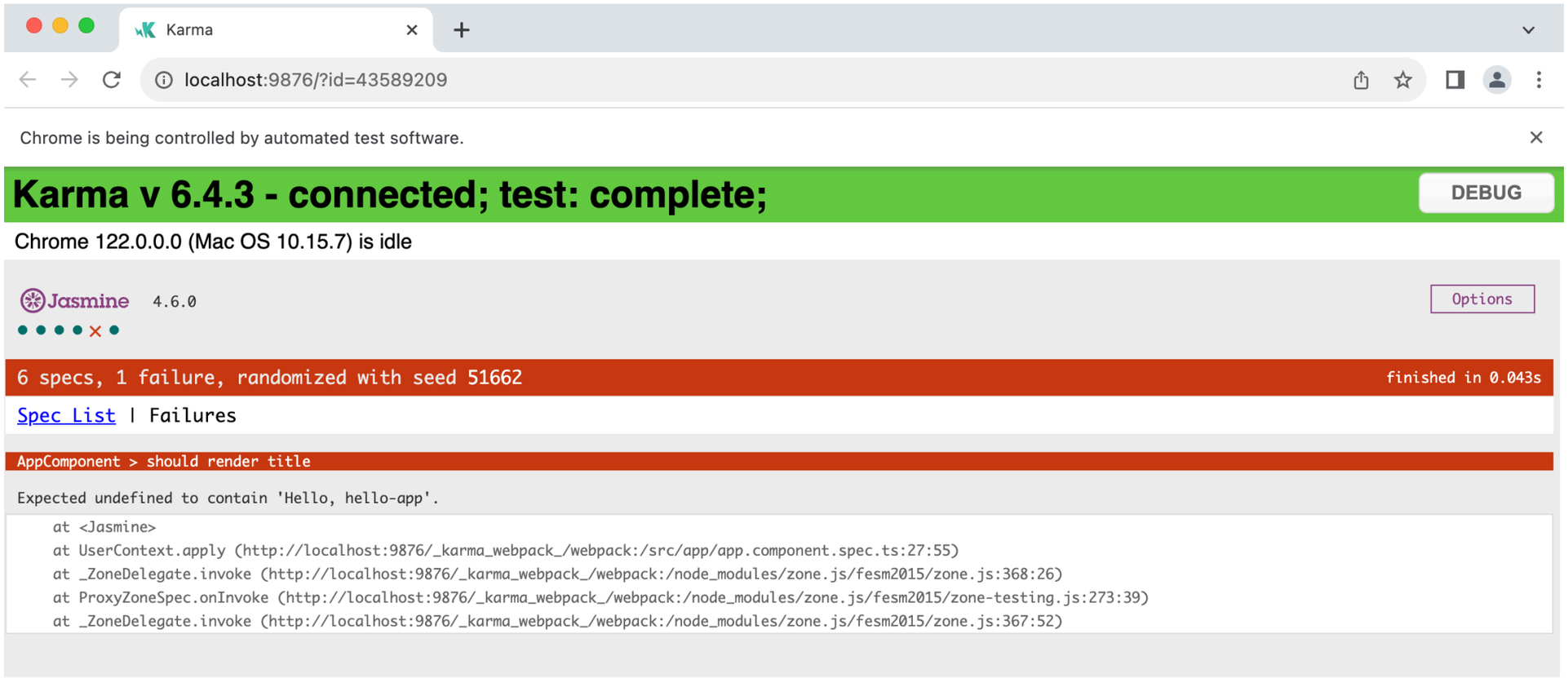
Login form



Testing

Default of Angular

$ng test



https://angular.io/guide/testing

TestBed

Manage environment for unit testing

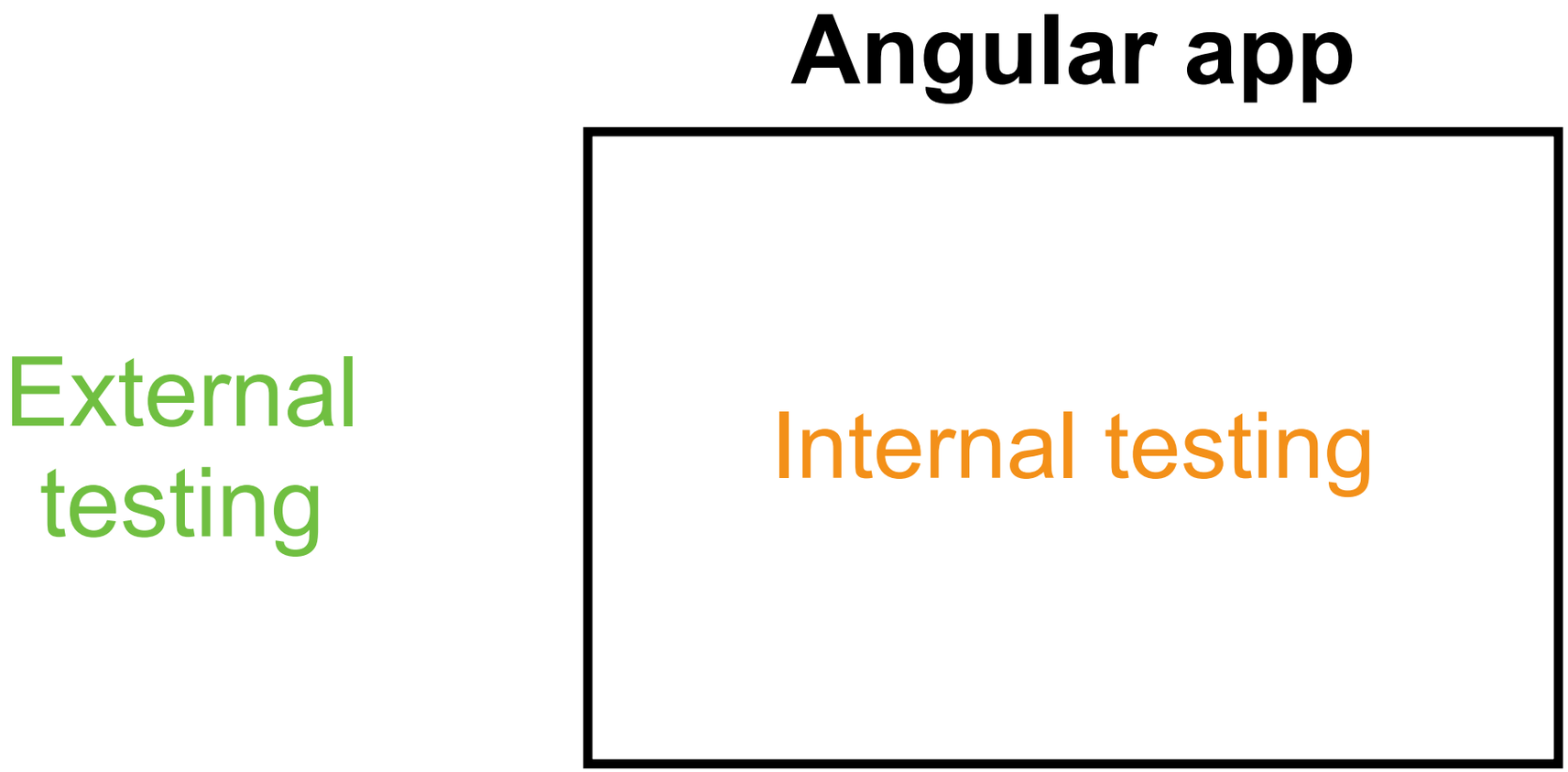
```ts
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [AppComponent],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });
```

https://angular.io/api/core/testing/TestBed

Testing

Component testing
Service testing
Pipe testing
Utility => mock, fakeAsync, tick
Unit testing
UI testing

Types of Testing



External Testing

External testing

Angular app

Internal testing

Internal Testing

External testing

Angular app

Internal testing

TestBed
Karma
Component
Service
Asynchronous

Angular Best Practices

Consistent Folder Structure

Easier to working as a team
Separate component/service/module
Separate component/service/module

```
/src
  /app
    /components
      /header
        header.component.ts
        header.component.html
        header.component.css
    /services
      data.service.ts
    /modules
      main.module.ts
```

Use Angular CLI

Use to generate project/component/model
Don’t copy and paste
Reduce human error

Modular Architecture

Breaking your app into smaller modules
Reusable
Easier to maintenance

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

Single Responsibility Principle

Use when design components and services
Easier to maintain and test

Bad

```
@Component({
    selector: 'app-user',
    templateUrl: './user.component.html'
  })
  export class UserComponent {
    // Contains multiple responsibilities
    fetchUserData() { ... }
    displayUserData() { ... }
    updateUserData() { ... }
  }
```

Good

```
@Component({
    selector: 'app-user',
    templateUrl: './user.component.html'
})
export class UserComponent {
    // Handles user display logic
    displayUserData() { ... }
}
```

Use Dependency Injection

Manage dependency in services
Improve testability

```
// Bad
@Injectable()
export class UserService {
  constructor() {
    // Avoid creating instances like this
    this.httpClient = new HttpClient();
  }
}

// Good
@Injectable()
export class UserService {
  constructor(private httpClient: HttpClient) {}
}
```

Lazy loading modules

Load modules only when needed
Improve initial load time

```
<!-- Bad -->
const routes: Routes = [
  { path: 'dashboard', component: DashboardComponent },
  // ...
];

<!-- Good -->
const routes: Routes = [
  { path: 'dashboard', loadChildren: ()
        => import('./dashboard/dashboard.module').then(m => m.DashboardModule) },
  // ...
];
```

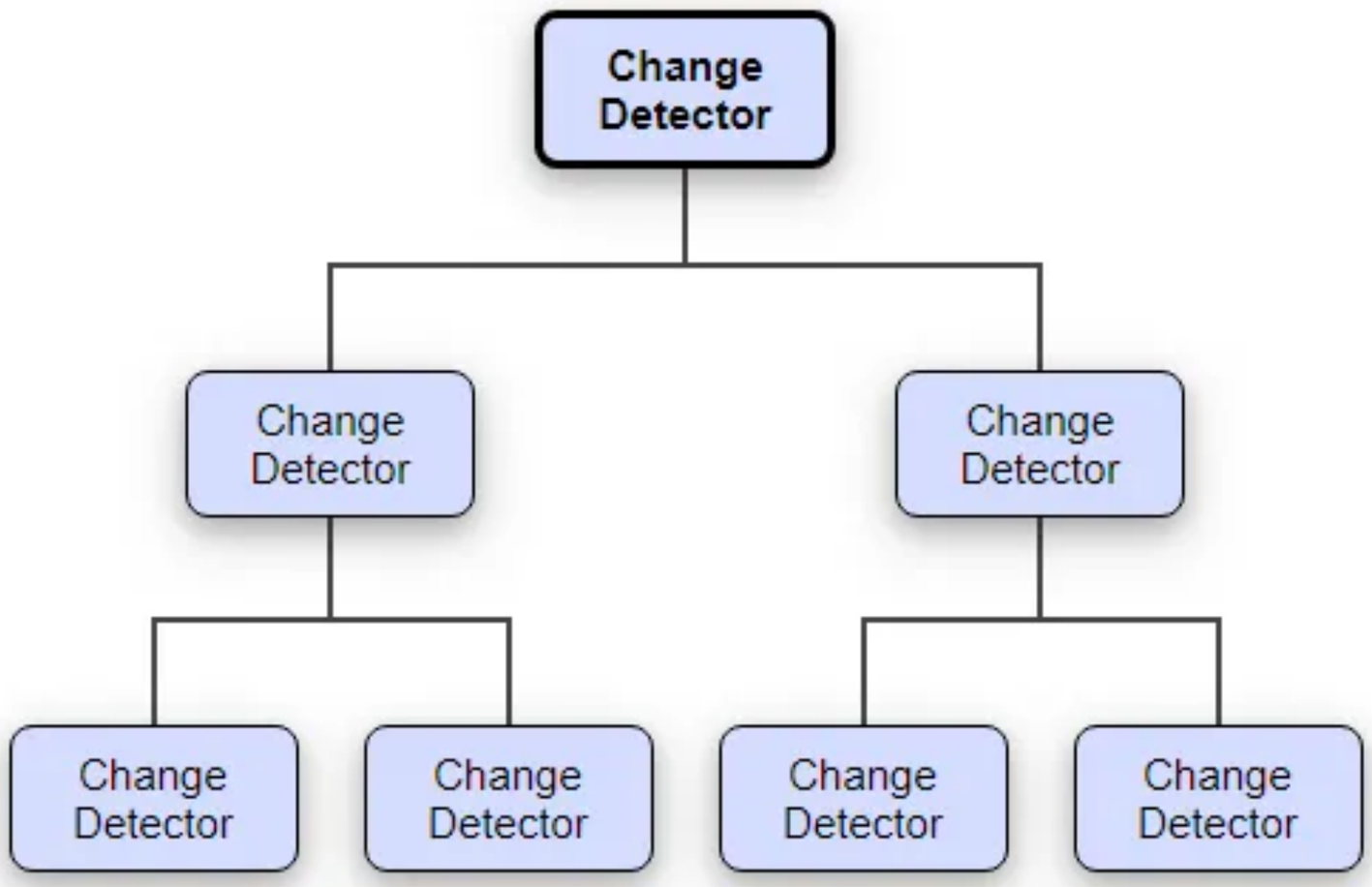
Optimize Angular Performance

Implement change detection strategies
When component should be updated ?

```
@Component({
    selector: 'app-list',
    templateUrl: './list.component.html',
    changeDetection: ChangeDetectionStrategy.OnPush
})
export class ListComponent {
    // ...
}
```

Change Detection

Keep UI in sync with app's data model



Optimize Angular Performance

Use memoization

Technique of caching the result of function call

```
@Component({
    selector: 'app-example',
    template: `
      <p>{{ getComplexValue(item) }}</p>
    `,
    changeDetection: ChangeDetectionStrategy.OnPush
})
export class ExampleComponent {
    private memo = new Map<Item, any>();

    getComplexValue(item: Item): any {
        if (!this.memo.has(item)) {
        this.memo.set(item,
this.calculateComplexValue(item));
        }

        return this.memo.get(item);
    }

    private calculateComplexValue(item: Item): any {
        // Do some fancy complex calculation
    }
}
```

Monitoring project build time !!

Q/A

Problems !!

$npm build

```
▲ [WARNING] bundle initial exceeded maximum budget.
Budget 500.00 kB was not met by 25.46 kB with a total of 525.46 kB.
```

How to solve this problem ?

What is budget ?
How can I disable this warning ?
What is making app bundle so big ?

Angular Budget

Config max size of app (acceptable)
Config in file angular.json

```
"configurations": {
    "production": {
      "budgets": [
        {
          "type": "initial",
          "maximumWarning": "500kb",
          "maximumError": "1mb"
        },
        {
          "type": "anyComponentStyle",
          "maximumWarning": "2kb",
          "maximumError": "4kb"
        }
      ],
      "outputHashing": "all"
    },
```

https://angular.io/guide/build#configuring-size-budgets

What is making app bundle so big ?

What is budget ?
How can I disable this warning ?
What is making app bundle so big ?

What is making app bundle so big ?

Generate bundle with source maps
Development mode only !!

$ng build --source-map=true --named-chunks=true

Visualize your bundle

Install source-map-explorer

$npm i -D source-map-explorer

https://github.com/danvk/source-map-explorer

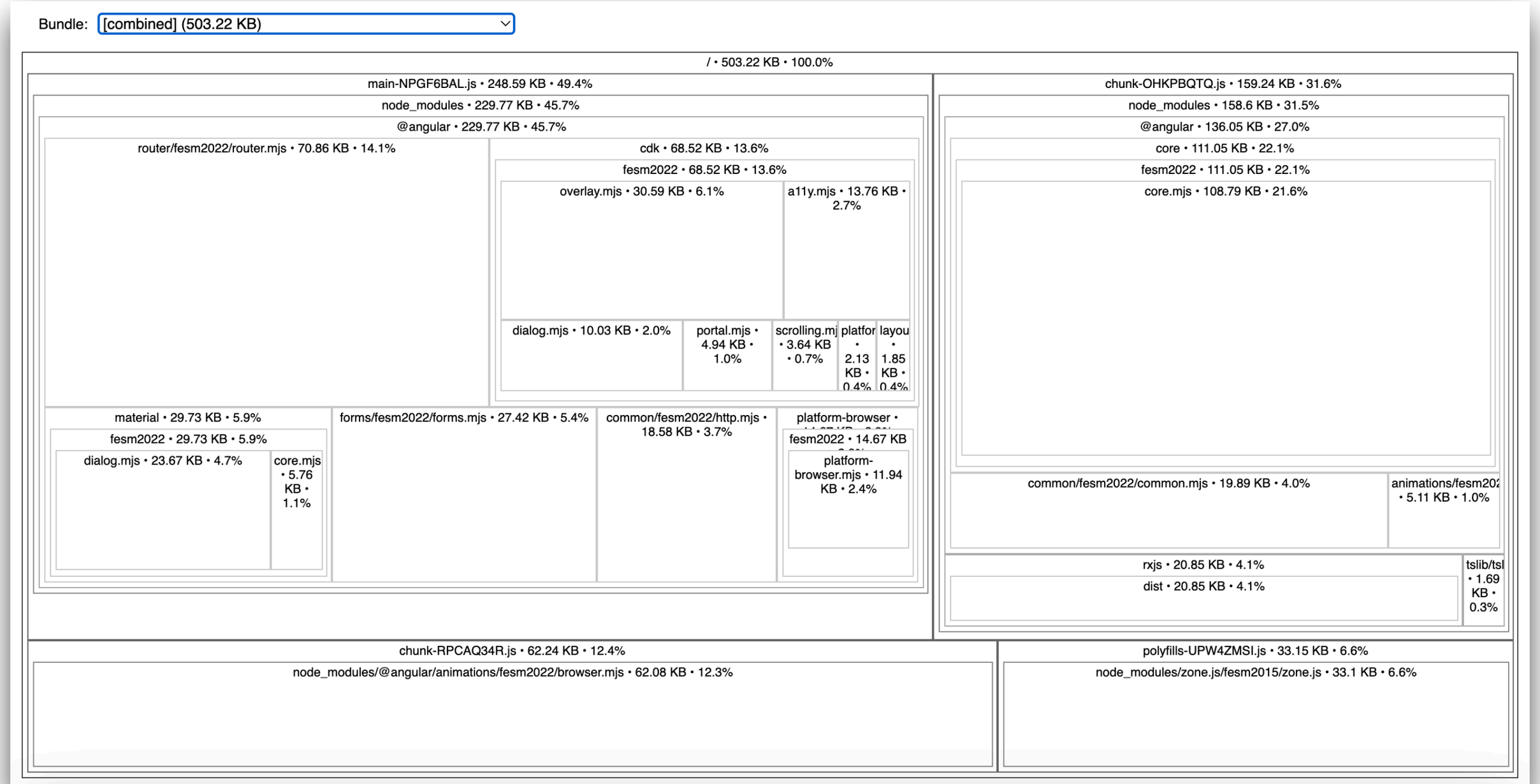
Visualize your bundle

Open in explorer

$ng build --source-map=true --named-chunks==true && source-map-explorer dist/<YOU_APP_NAME>//*.js**

https://github.com/danvk/source-map-explorer

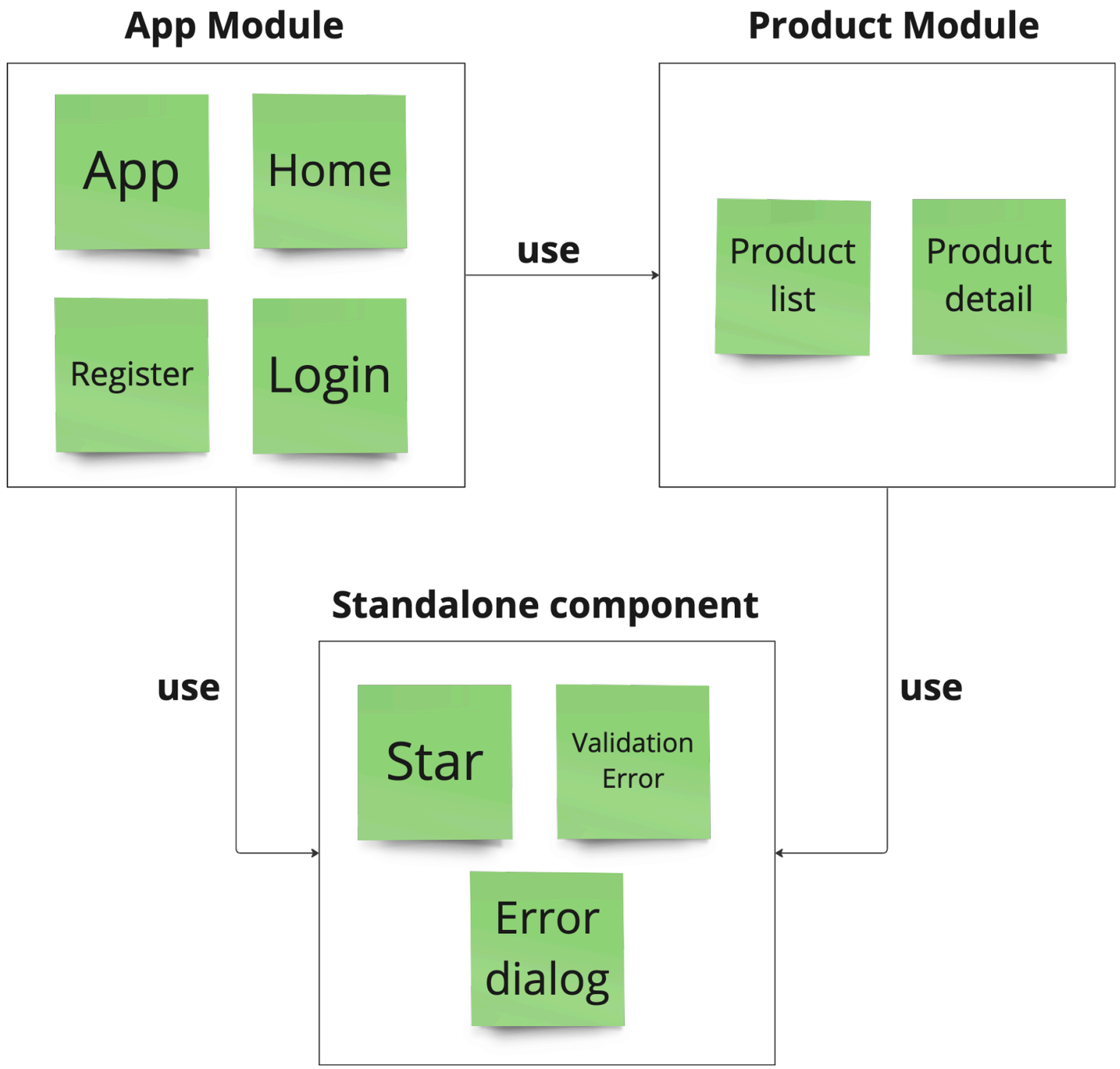
Visualize your bundle



Other solutions

Use lazy load for multi-modules
Standalone component

Project Structure



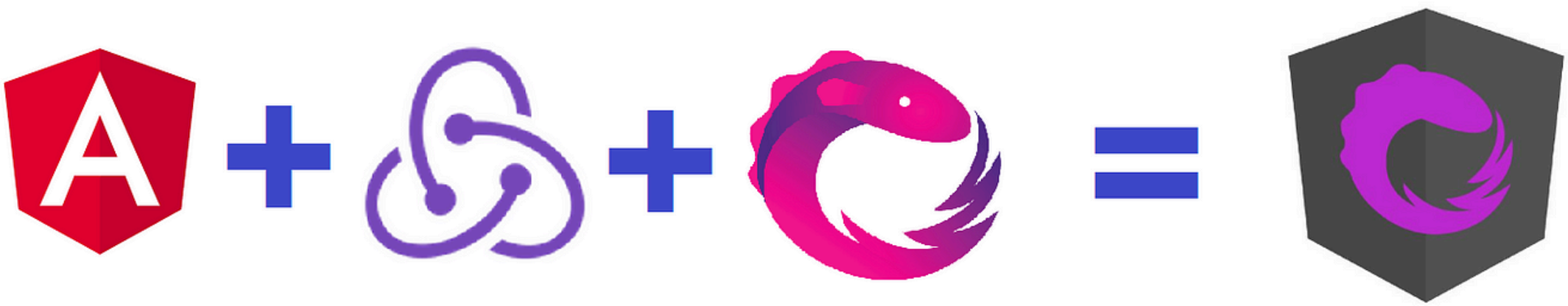
Building

```
Initial chunk files | Names         |  Raw size
styles.css          | styles        |  93.20 kB |
polyfills.js        | polyfills     |  83.60 kB |
main.js             | main          |  26.78 kB |

                    | Initial total | 203.58 kB

Lazy chunk files    | Names         |  Raw size
chunk-4A6TWH6D.js   | products-module |  2.37 kB |
```

Size comparison

Single module	Multi-module + Lazy loading	Standalone component
524.60 kb	?	?

Learning Next ?



Angular	*Redux*	*RxJS*	*NgRX*