

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 Liked Following Share ...

+ Add a Button



Agenda

- Introduction to Angular 13
- Installation and configuration
- Structure of Angular project
- Introduction to TypeScript
- Design and develop component/service
- Routing management
- Working with RESTful APIs



Angular



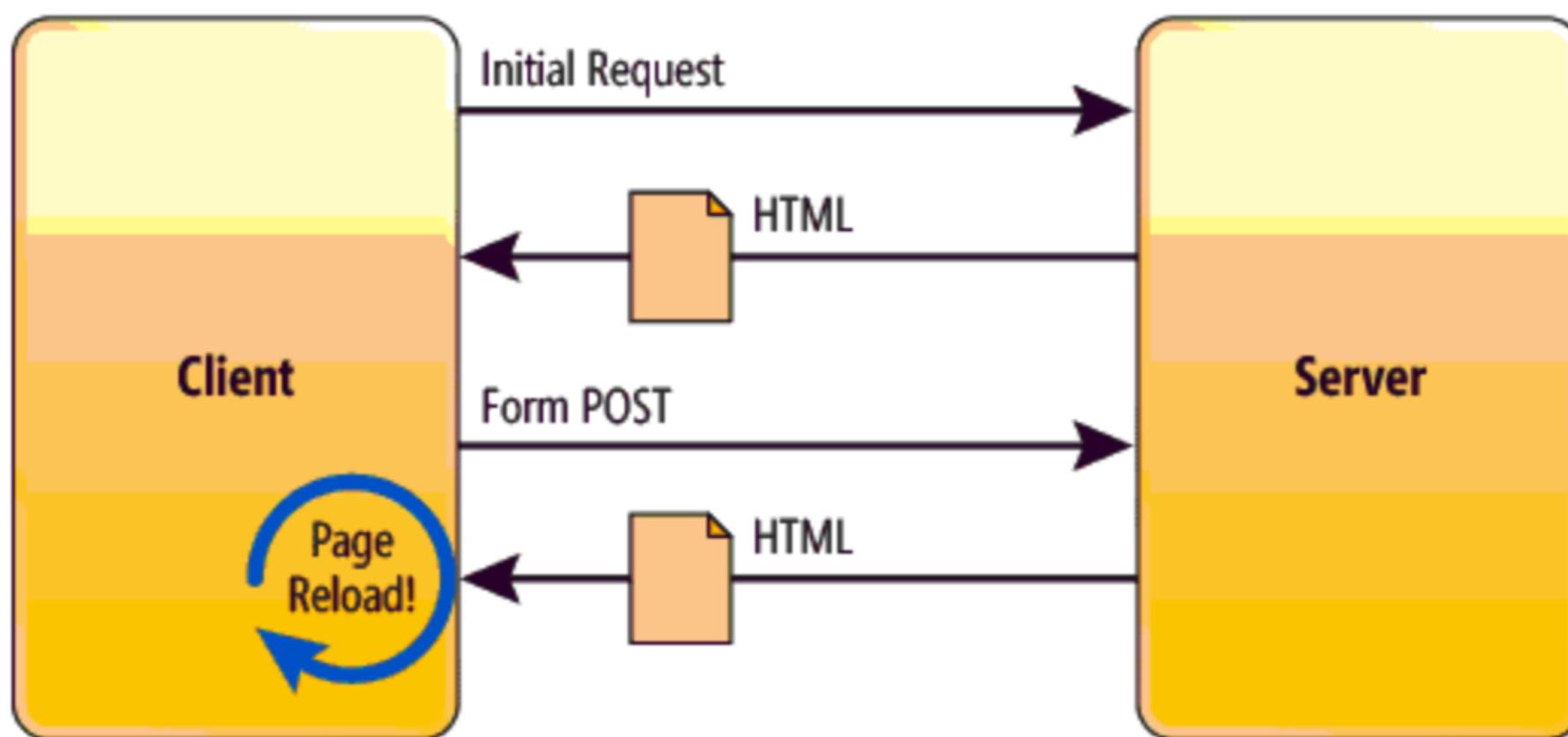
What is Angular ?

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

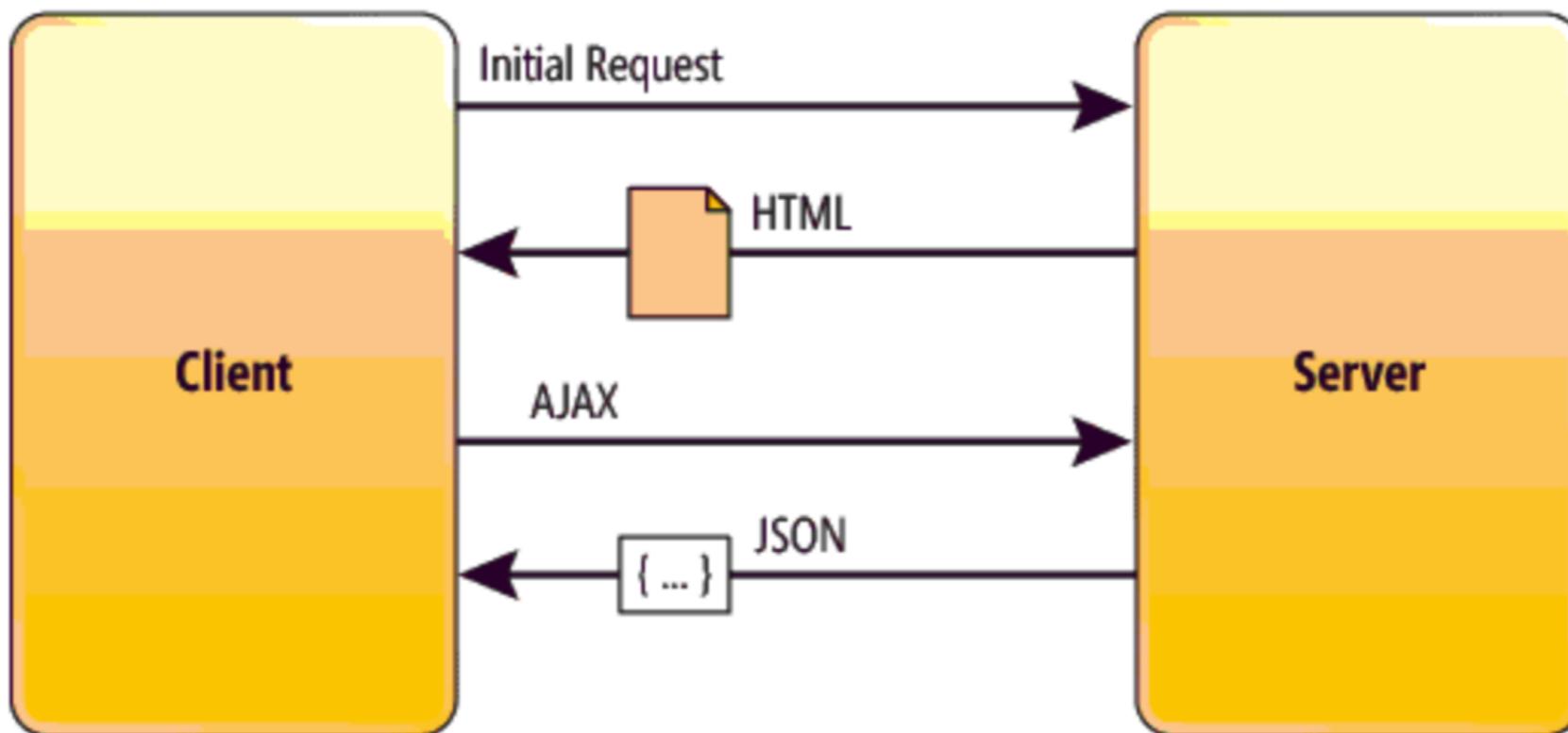
<https://angular.io/>



Traditional



Single Page Application

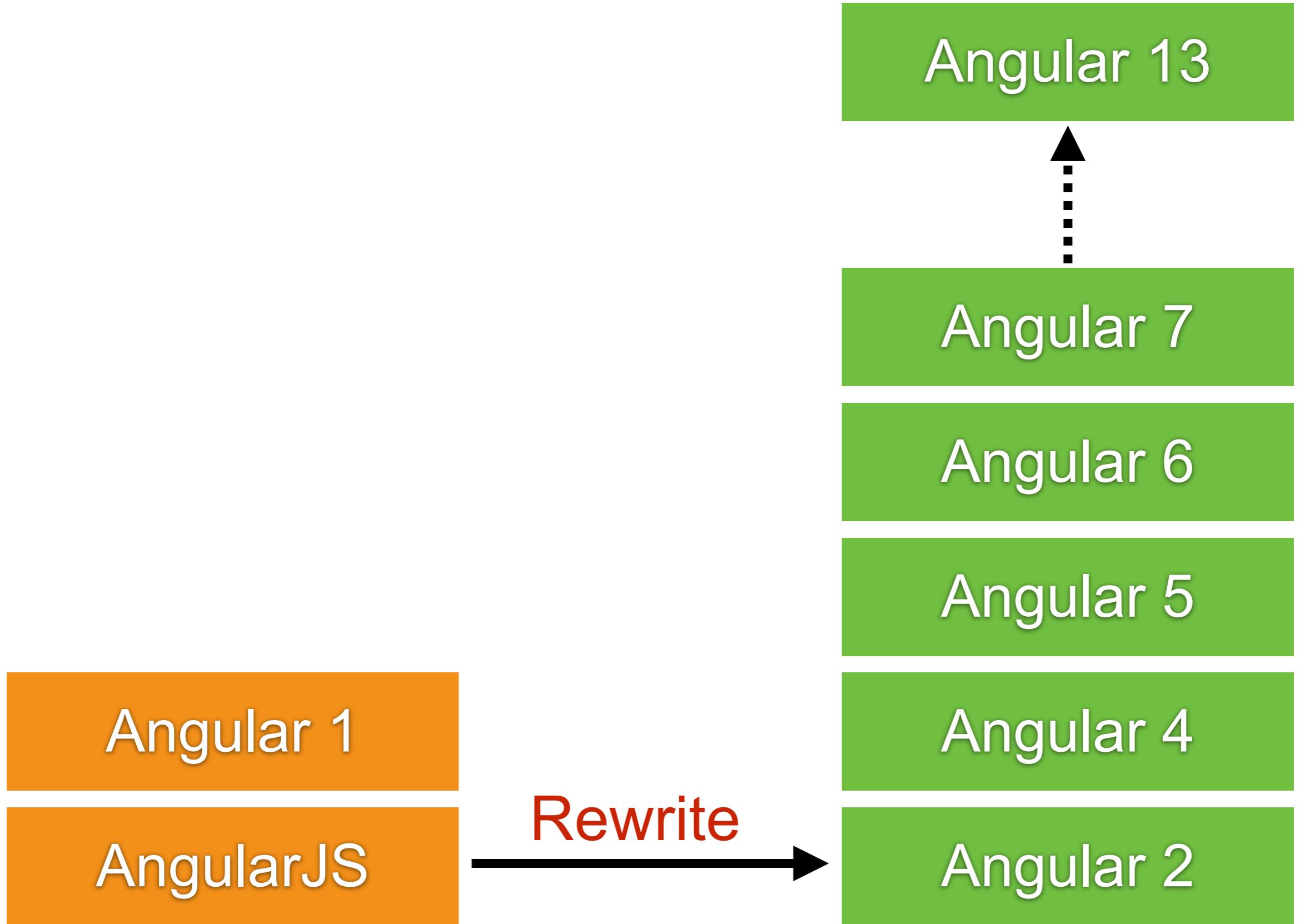


Angular 1

AngularJS





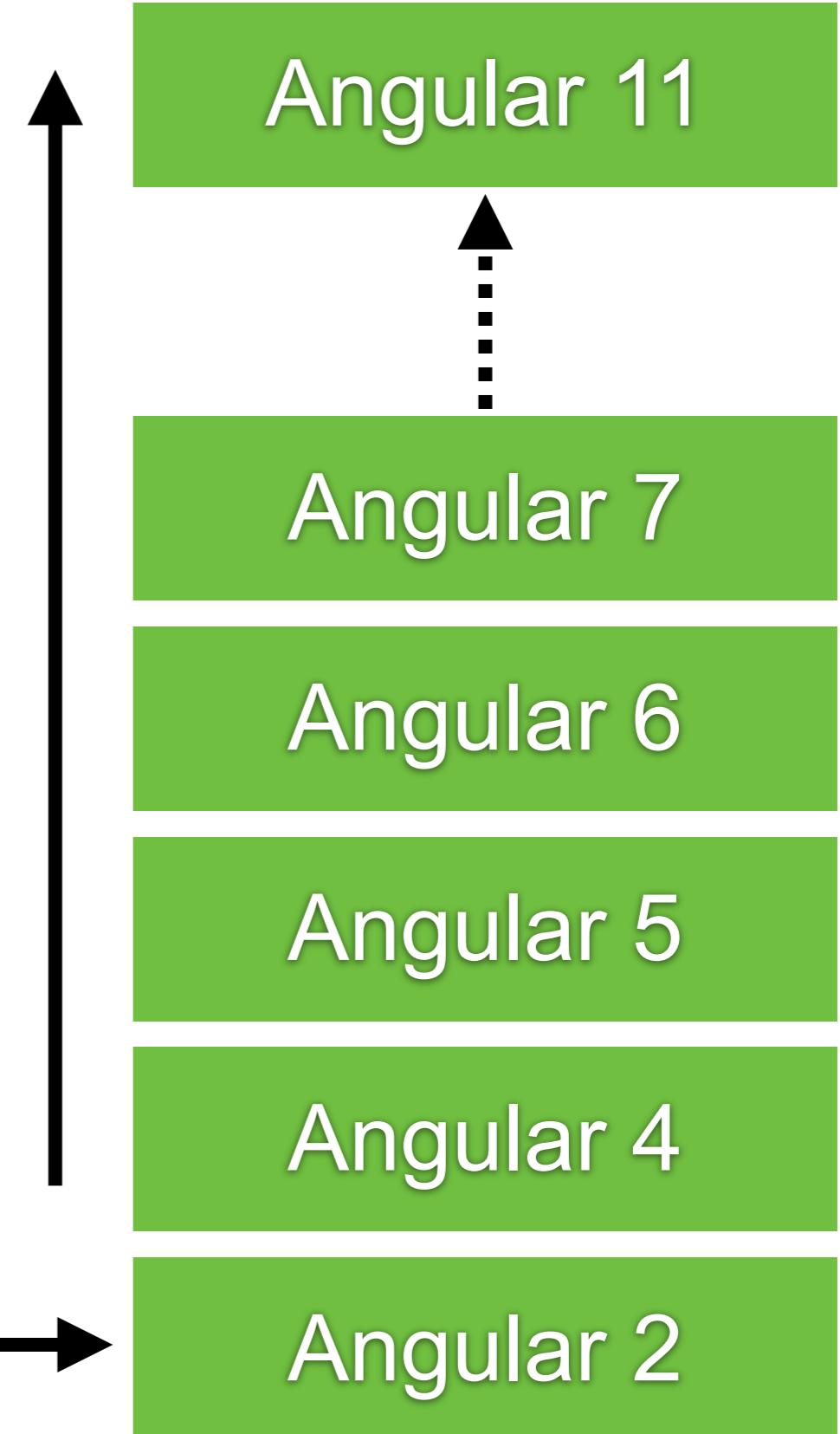


**Incremental improvement
(Every 6 months)**

Angular 1

AngularJS

Rewrite



Software requirement

Install NodeJS

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

Download for macOS (x64)

[16.15.0 LTS](#)

Recommended For Most Users

[18.0.0 Current](#)

Latest Features

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

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

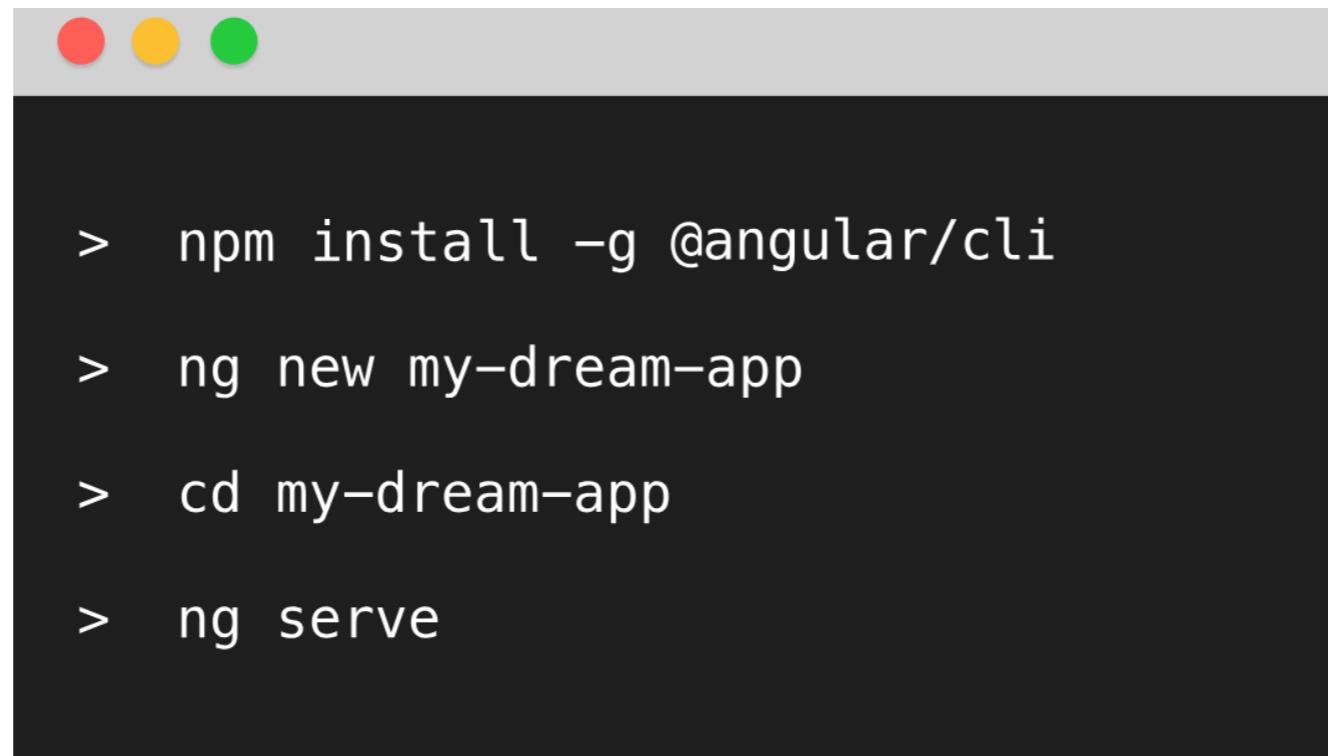
Or have a look at the [Long Term Support \(LTS\) schedule](#)

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



Angular CLI

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

A screenshot of a macOS terminal window. The title bar has three colored window control buttons (red, yellow, green). The main window is black with white text. It contains four command-line entries, each starting with a blue right-pointing arrow: "npm install -g @angular/cli", "ng new my-dream-app", "cd my-dream-app", and "ng serve".

```
> 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
```



Try after installed

\$ng version

```
Angular CLI: 13.3.4
Node: 18.0.0 (Unsupported)
Package Manager: npm 8.6.0
OS: darwin x64

Angular:
...
Package          Version
-----
@angular-devkit/architect    0.1303.4 (cli-only)
@angular-devkit/core         13.3.4 (cli-only)
@angular-devkit/schematics   13.3.4 (cli-only)
@schematics/angular          13.3.4 (cli-only)
```



TypeScript

TS TypeScript

Download Docs Handbook Community Playground Tools

Search Docs

TypeScript 4.2 is now available, 4.3 is currently in beta.

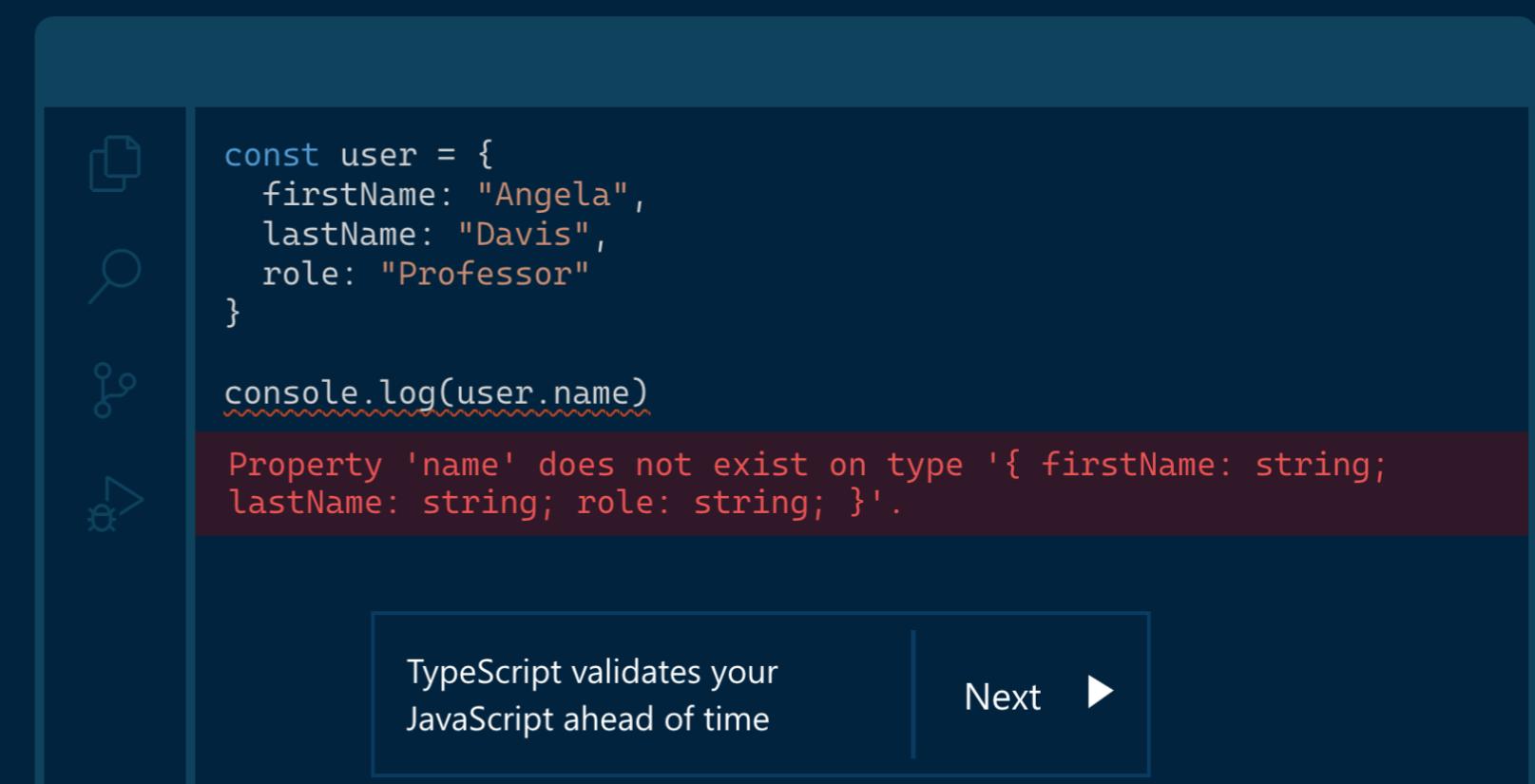
Typed JavaScript at Any Scale.

By understanding JavaScript, TypeScript saves you time catching errors and providing fixes before you run code.

Any browser, any OS, anywhere JavaScript runs.
Entirely Open Source.

Try TypeScript Now
Online or via npm

...



The screenshot shows a code editor with the following TypeScript code:

```
const user = {
  firstName: "Angela",
  lastName: "Davis",
  role: "Professor"
}

console.log(user.name)
```

An error message is displayed below the code:

Property 'name' does not exist on type '{ firstName: string; lastName: string; role: string; }'.

On the left side of the editor, there are four icons: a file, a magnifying glass, a gear, and a right-pointing arrow. On the right side, there is a button labeled "Next" with a right-pointing arrow icon.

<https://www.typescriptlang.org/>



RxJS



RxJS

Reactive Extensions Library for JavaScript

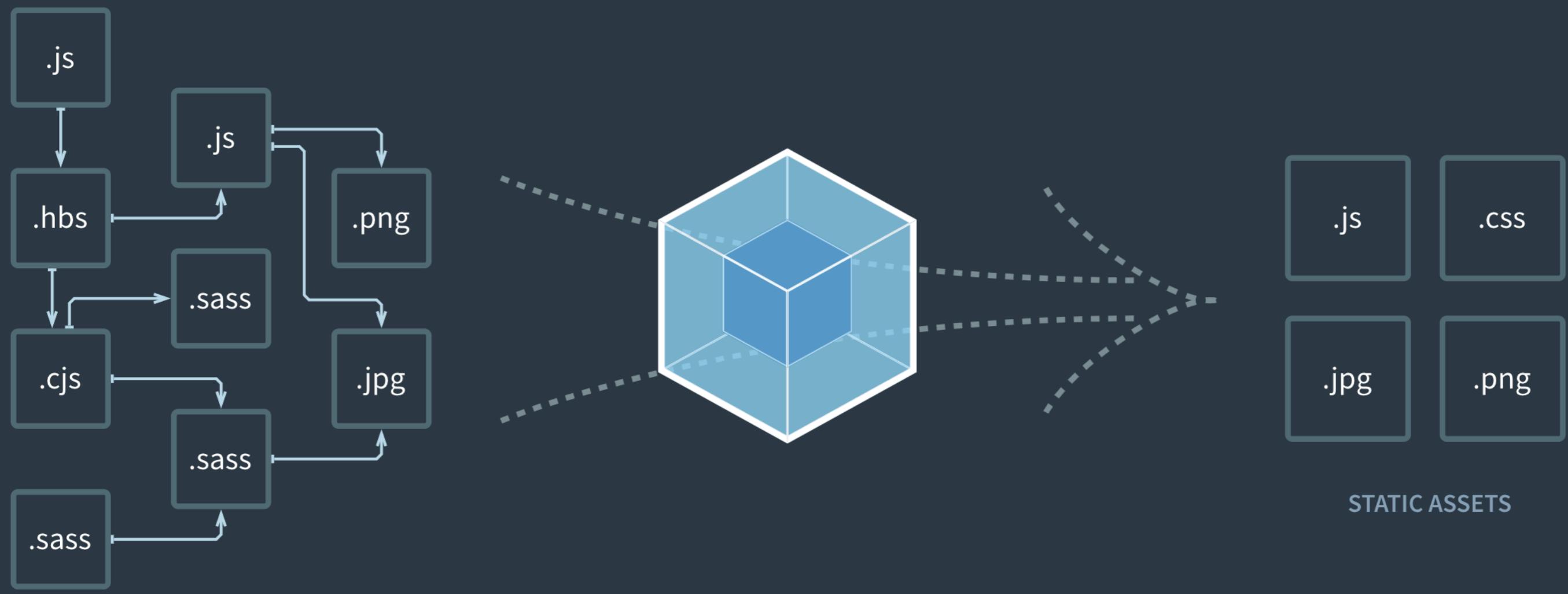
GET STARTED

API DOCS

<https://rxjs.dev/>



Webpack



<https://webpack.js.org>



Create first project



Try to create first project

\$ng new hello-app

```
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? CSS
CREATE hello-app/README.md (1062 bytes)
CREATE hello-app/.editorconfig (274 bytes)
CREATE hello-app/.gitignore (548 bytes)
CREATE hello-app/angular.json (3057 bytes)
CREATE hello-app/package.json (1072 bytes)
CREATE hello-app/tsconfig.json (863 bytes)
CREATE hello-app/.browserslistrc (600 bytes)
CREATE hello-app/karma.conf.js (1426 bytes)
CREATE hello-app/tsconfig.app.json (287 bytes)
CREATE hello-app/tsconfig.spec.json (333 bytes)
CREATE hello-app/.vscode/extensions.json (130 bytes)
CREATE hello-app/.vscode/launch.json (474 bytes)
CREATE hello-app/.vscode/tasks.json (938 bytes)
CREATE hello-app/src/favicon.ico (948 bytes)
CREATE hello-app/src/index.html (294 bytes)
CREATE hello-app/src/main.ts (372 bytes)
CREATE hello-app/src/polyfills.ts (2338 bytes)
CREATE hello-app/src/styles.css (80 bytes)
```



Run your app

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

Initial Chunk Files	Names	Raw Size
vendor.js	vendor	1.70 MB
polyfills.js	polyfills	294.80 kB
styles.css, styles.js	styles	173.22 kB
main.js	main	47.67 kB
runtime.js	runtime	6.52 kB

| Initial Total | 2.21 MB

Build at: 2022-04-28T15:18:36.059Z - Hash: c56d3e6aa0b191db - Time: 6313ms

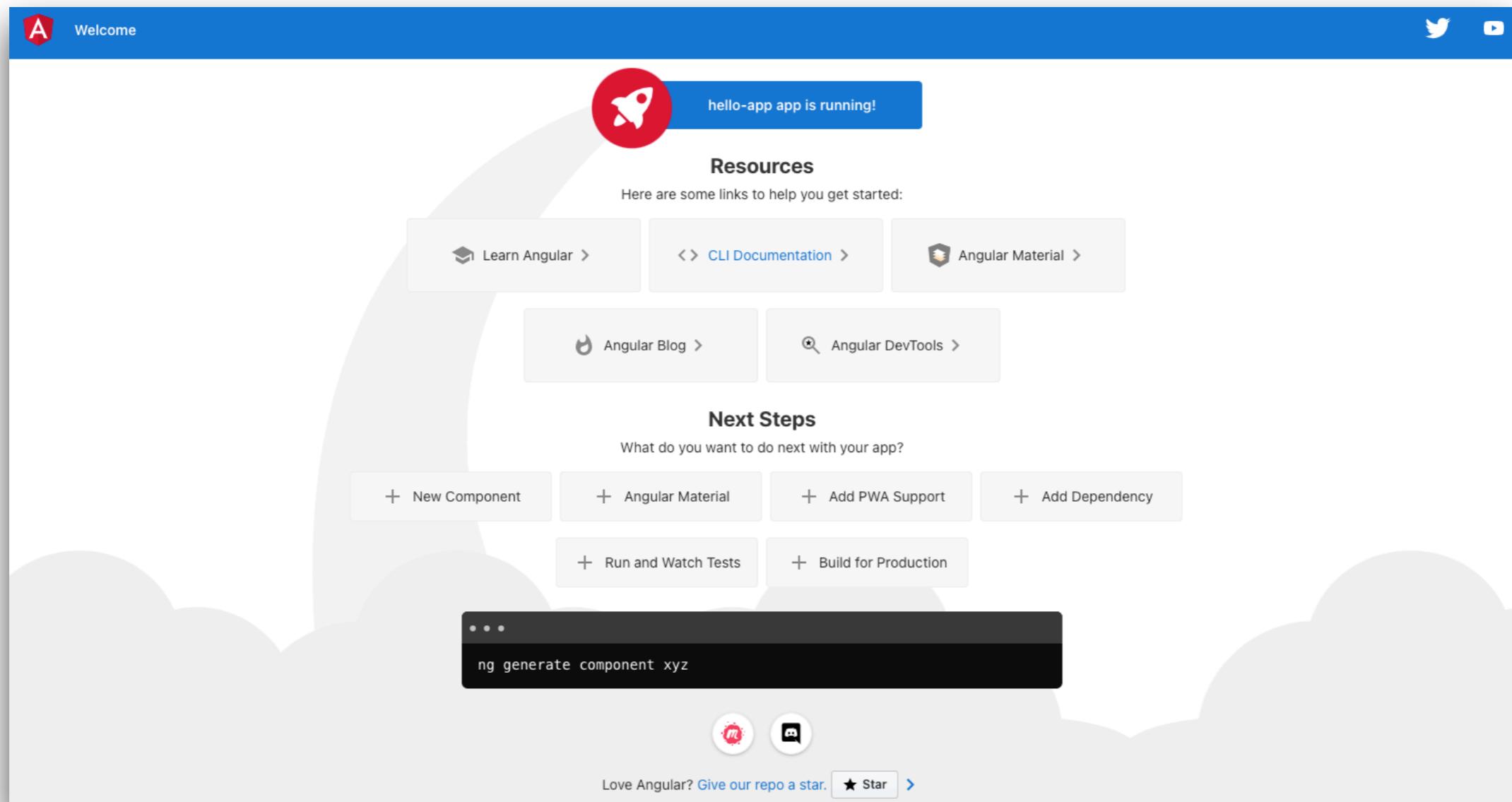
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.

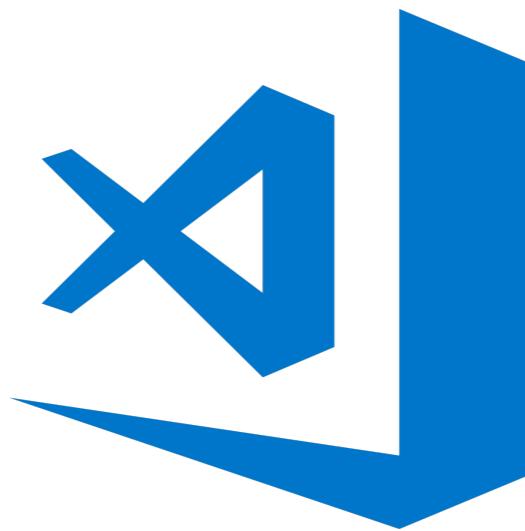


Open in browser

<http://localhost:4200/>



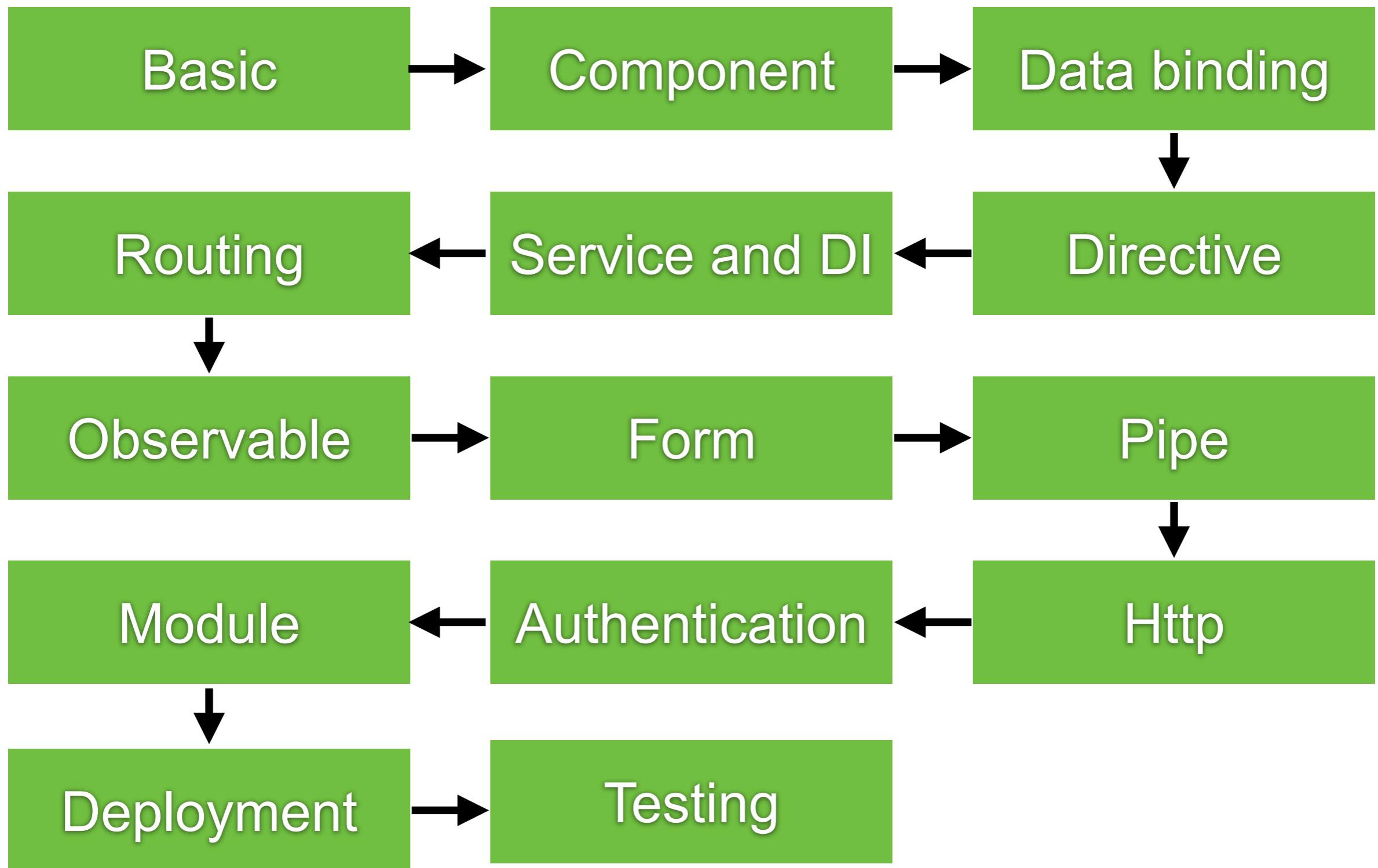
Install Text Editor



Learning path



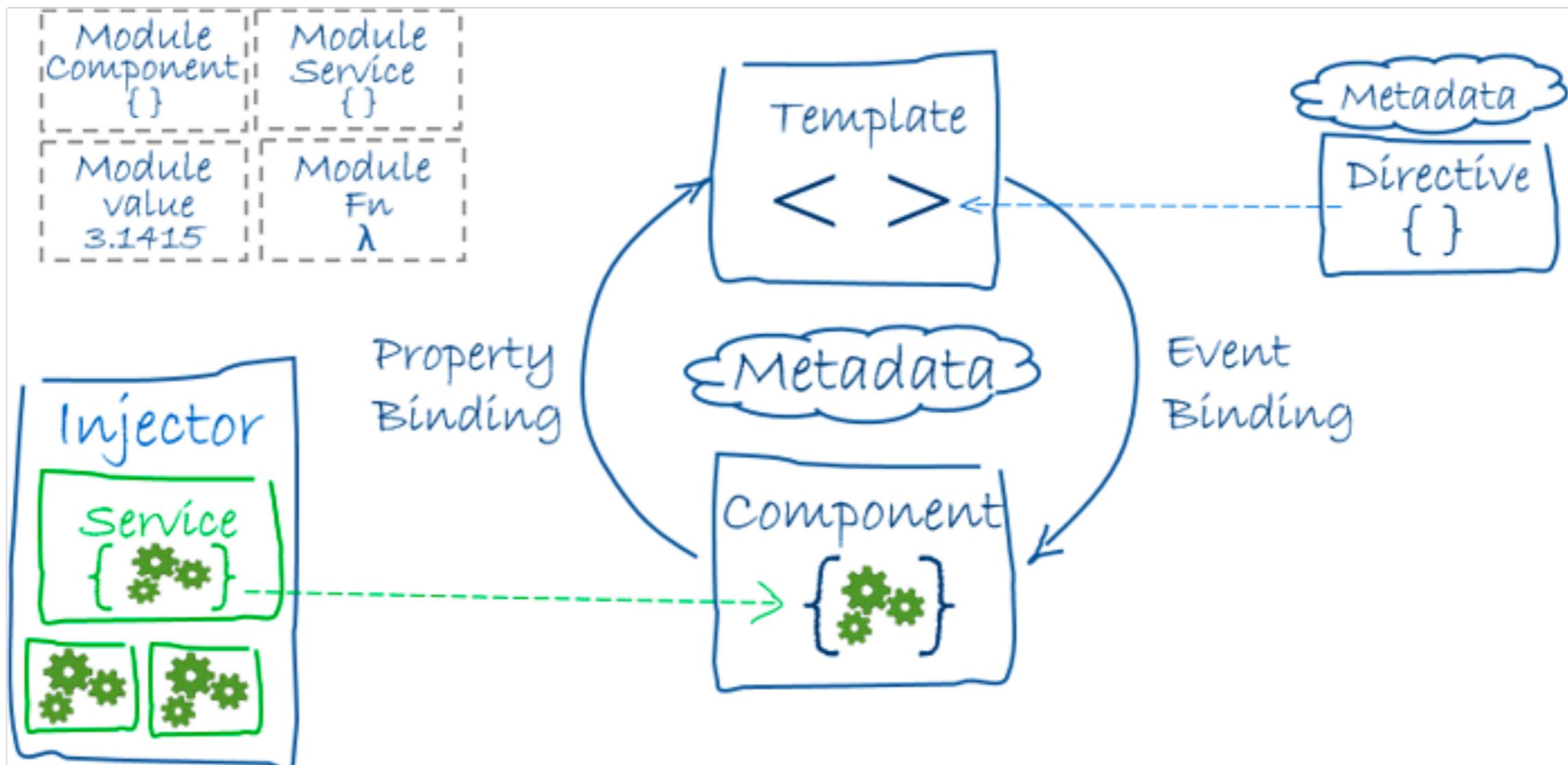
Learning path



Basic of Angular



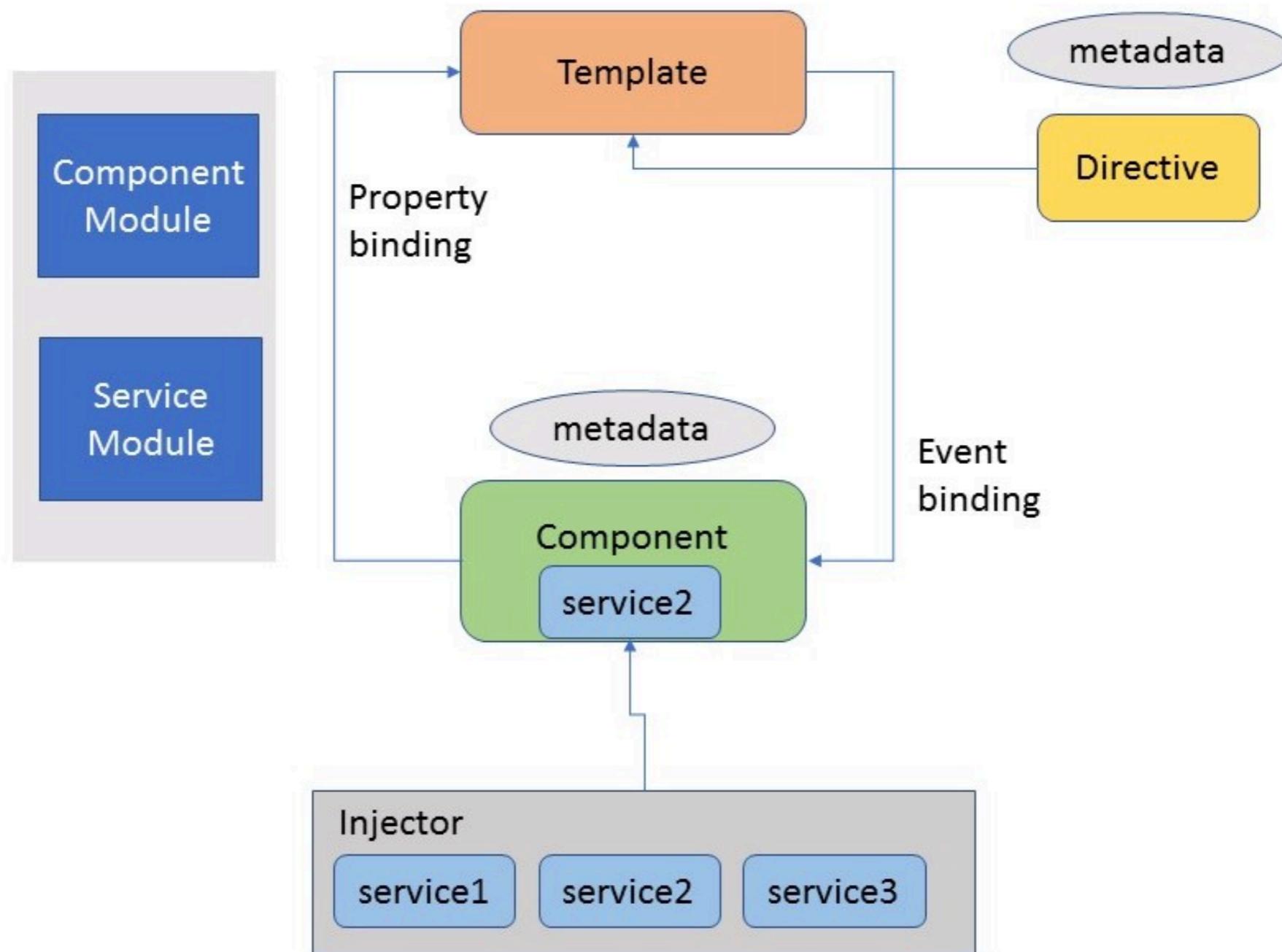
Angular Architecture



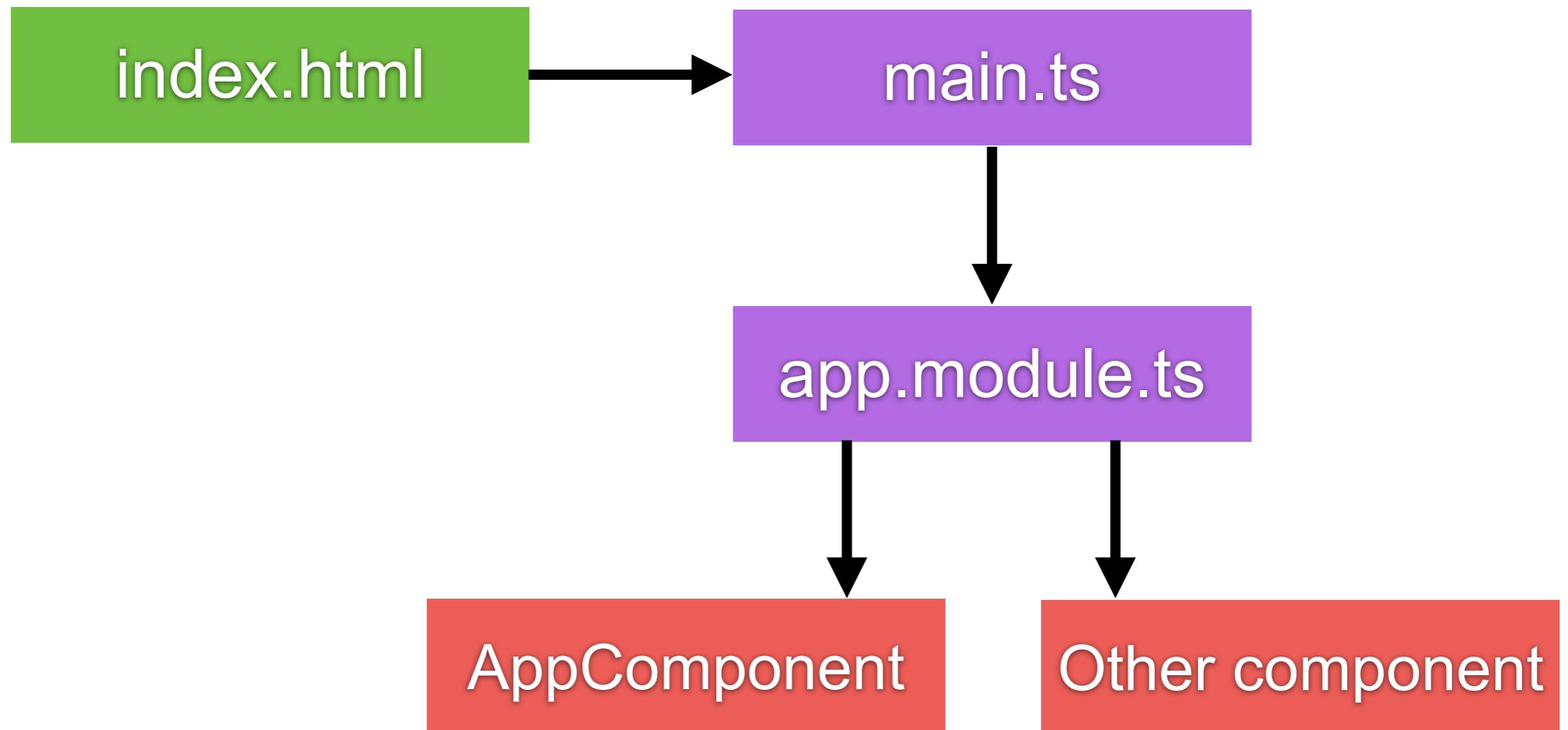
<https://angular.io/guide/architecture>



Angular Architecture



How Angular working ?

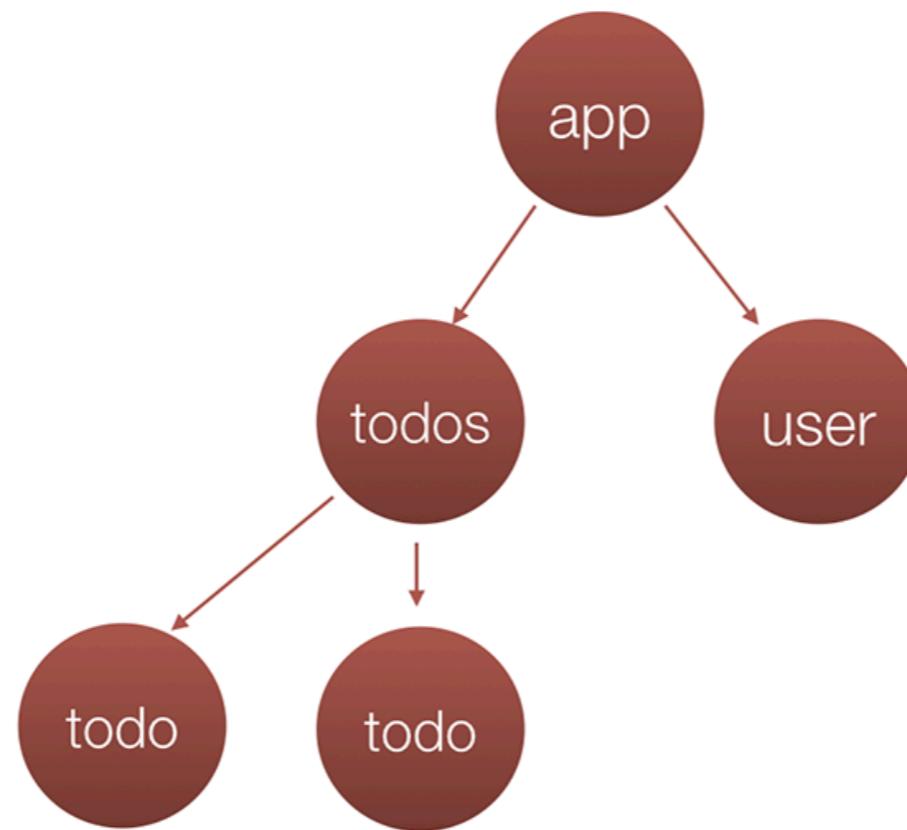


Components

Angular application is a tree of **Components**

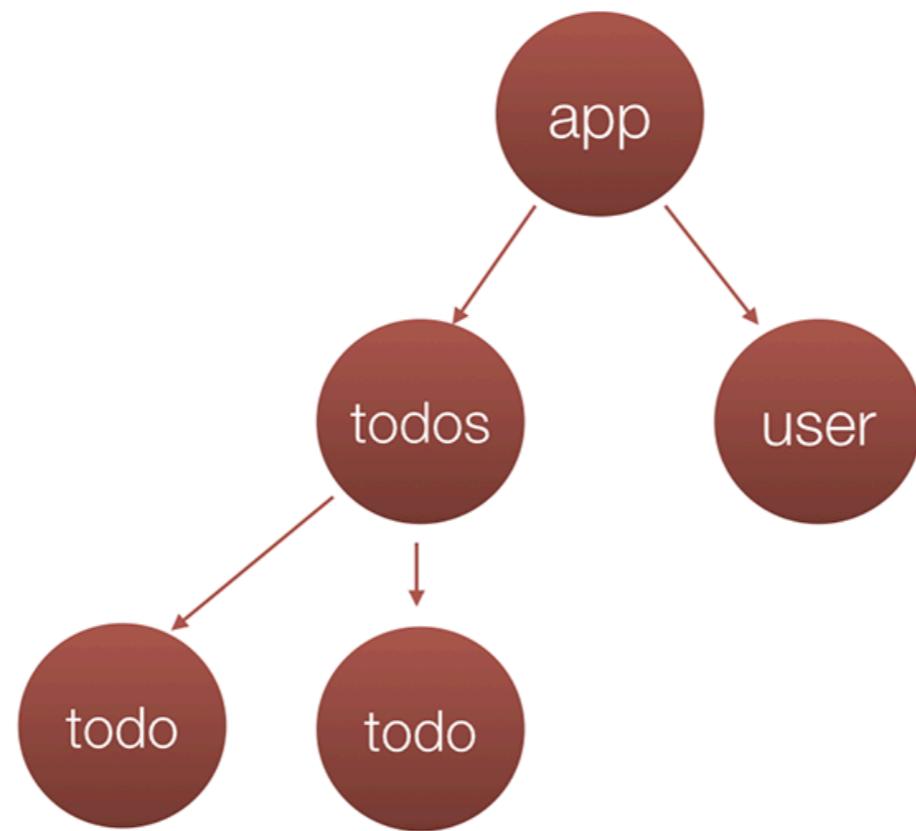
Top level component is the application itself

Component is rendered by the browser



Components

Composable
Reusable
Hierarchical



Components

Own template

Own style

Own business logic

Split complex app to small component !!



Design components

Home

About

Contact

Menu

Item 1

Item 2

Item 3



Header component

Home

About

Contact

Menu

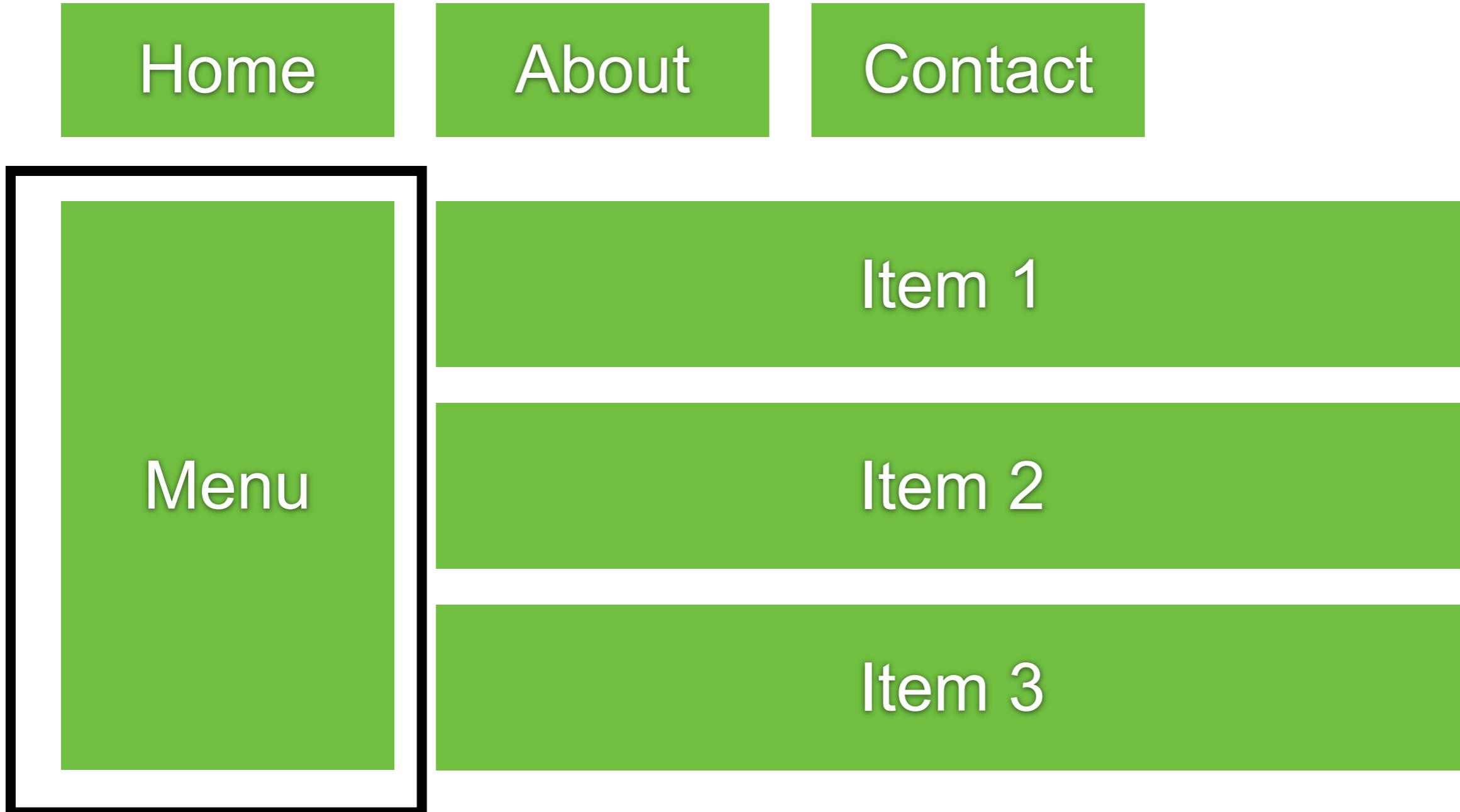
Item 1

Item 2

Item 3



Sidebar/Menu component



Item component

Home

About

Contact

Menu

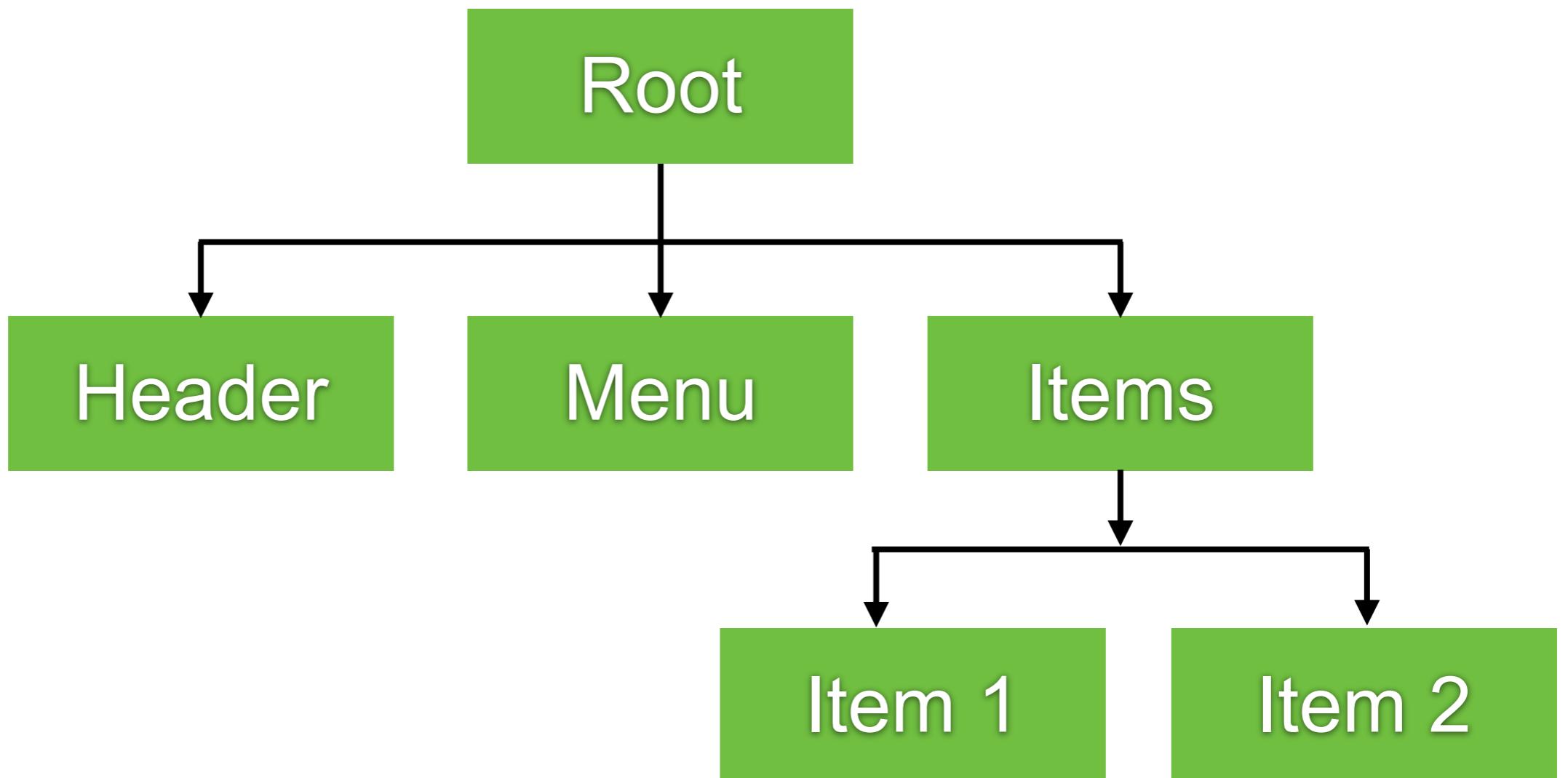
Item 1

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';  
}
```



Angular is Modular

@angular/
core

@angular/
http

@angular/
router



Hello



Hello World

Textfield

Output=



Types of data binding

Interpolation

Properties/ attributes

Event

Two-way



Types of data binding

Type name	Example
Interpolation	<code>{{ expression }}</code>
Property/Attribute/Class/Style	<code>[target] = “expression”</code>
Event	<code>(target)= “statement”</code>
Two-way	<code>[(target)] = “expression”</code>

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



Hello World

Name 1 : <input type="text" (keyup)="onTypedName2(\$event)" />

Name 2 : <input type="text" #nameText (keyup)="onTypedName(nameText.value)" />

Name 3 : <input type="text" [value]="name" (input)="onTypedName2(\$event)" />

<div *ngIf="name; else noName" >Hi {{name}}</div>

<ng-template #noName>Please enter your name !!</ng-template>



Interpolation

Name 1 : <input type="text" (keyup)="onTypedName2(\$event)" />

Name 2 : <input type="text" #nameText (keyup)="onTypedName(nameText.value)" />

Name 3 : <input type="text" [value]="name" (input)="onTypedName2(\$event)" />

<div *ngIf="name; else noName" >Hi {{name}}</div>

<ng-template #noName>Please enter your name !!</ng-template>

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



Binding to event

Name 1 : <input type="text" (keyup)="onTypedName2(\$event)" />

Name 2 : <input type="text" #nameText (keyup)="onTypedName(nameText.value)" />

Name 3 : <input type="text" [value]="name" (input)="onTypedName2(\$event)" />

```
<div *ngIf="name; else noName" >Hi {{name}}</div>
```

```
<ng-template #noName>Please enter your name !!</ng-template>
```

<https://angular.io/guide/event-binding>



Property binding

Name 1 : <input type="text" (keyup)="onTypedName2(\$event)" />

Name 2 : <input type="text" #nameText (keyup)="onTypedName(nameText.value)" />

Name 3 : <input type="text" [value]="name" (input)="onTypedName2(\$event)" />

<div *ngIf="name; else noName" >Hi {{name}}</div>

<ng-template #noName>Please enter your name !!</ng-template>

<https://angular.io/guide/property-binding>



Function in Component

```
export class AppComponent {  
  name = '';  
  
  onTypedName(updateName: string) {  
    this.name = updateName;  
  }  
  
  onTypedName2(event: Event) {  
    this.name = (event.target as HTMLInputElement).value;  
  }  
}
```



Structural directive (*ngIf)

Name 1 : <input type="text" (keyup)="onTypedName2(\$event)" />

Name 2 : <input type="text" #nameText (keyup)="onTypedName(nameText.value)" />

Name 3 : <input type="text" [value]="name" (input)="onTypedName2(\$event)" />

```
<div *ngIf="name; else noName" >Hi {{name}}</div>
<ng-template #noName>Please enter your name !! </ng-template>
```

<https://angular.io/api/common/NgIf>



Two-way binding in Form

1. Template in HTML file

```
<input [(ngModel)]="name">
<input [ngModel]="name" (ngModelChange)="name = $event">
```

2. Add FormModule in app.module.ts

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

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

<https://angular.io/guide/two-way-binding>



Workshop

Textfield

Button

Output=



Click button

Add

```
<button type="button"  
        (click)="onAdd( )">Add</button>
```



Show data (interpolation)

Add

Output=

```
<button type="button"  
        (click)="onAdd()>Add</>  
<p>Output={{output}}</p>
```



Show array data (*ngFor)

```
tasks = ['Task 1', 'Task 2'];
```

```
<p>My Tasks:</p>
<ul>
  <li *ngFor="let task of tasks">
    {{ task }}
  </li>
</ul>
```

<https://angular.io/guide/structural-directives>



Get data from input (template reference)

Textfield

Button

Output=

```
<input #box2 (keyup.enter)="onEnter(box2.value)">
```

```
<button (click)="onAdd(box3.value)">Add</button>
```



Workshop



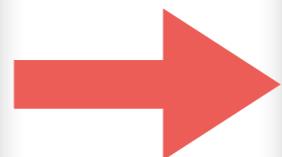
Workshop

Login Page

Email address:

Password:

Login

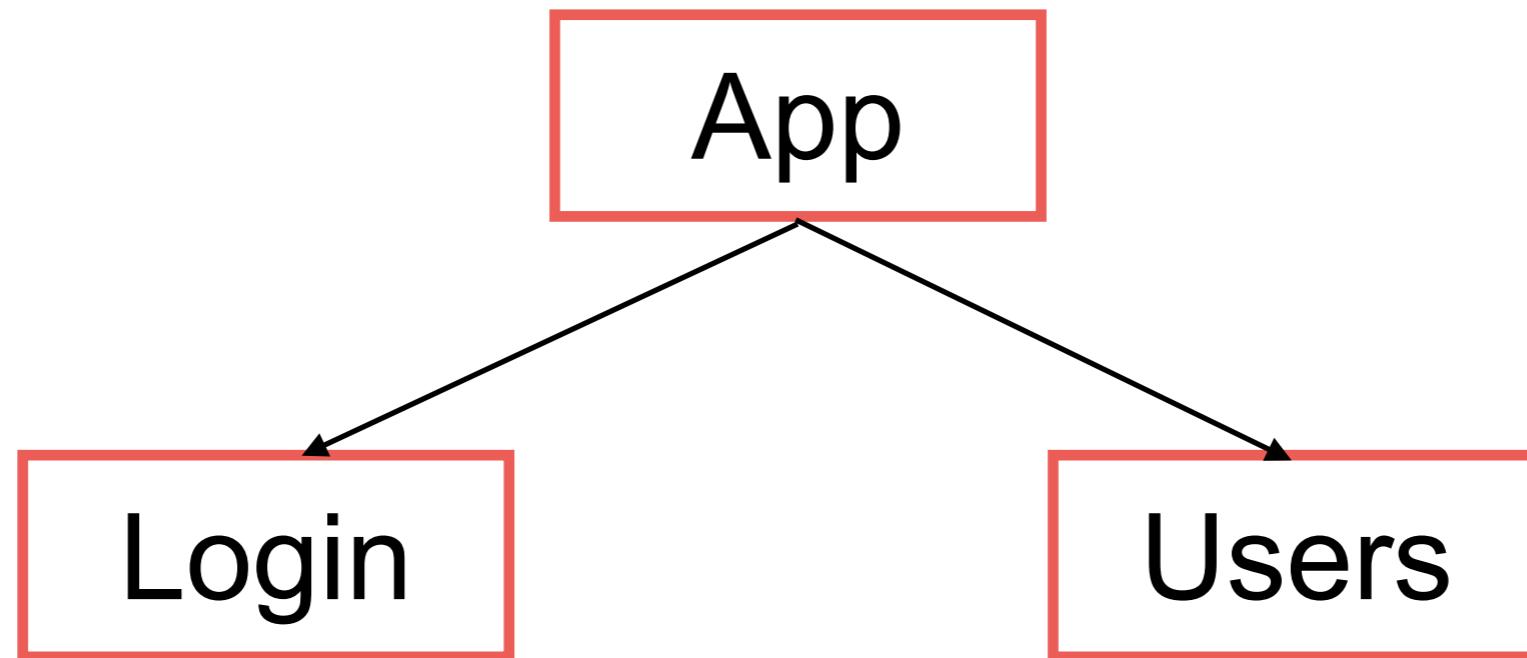


List of users

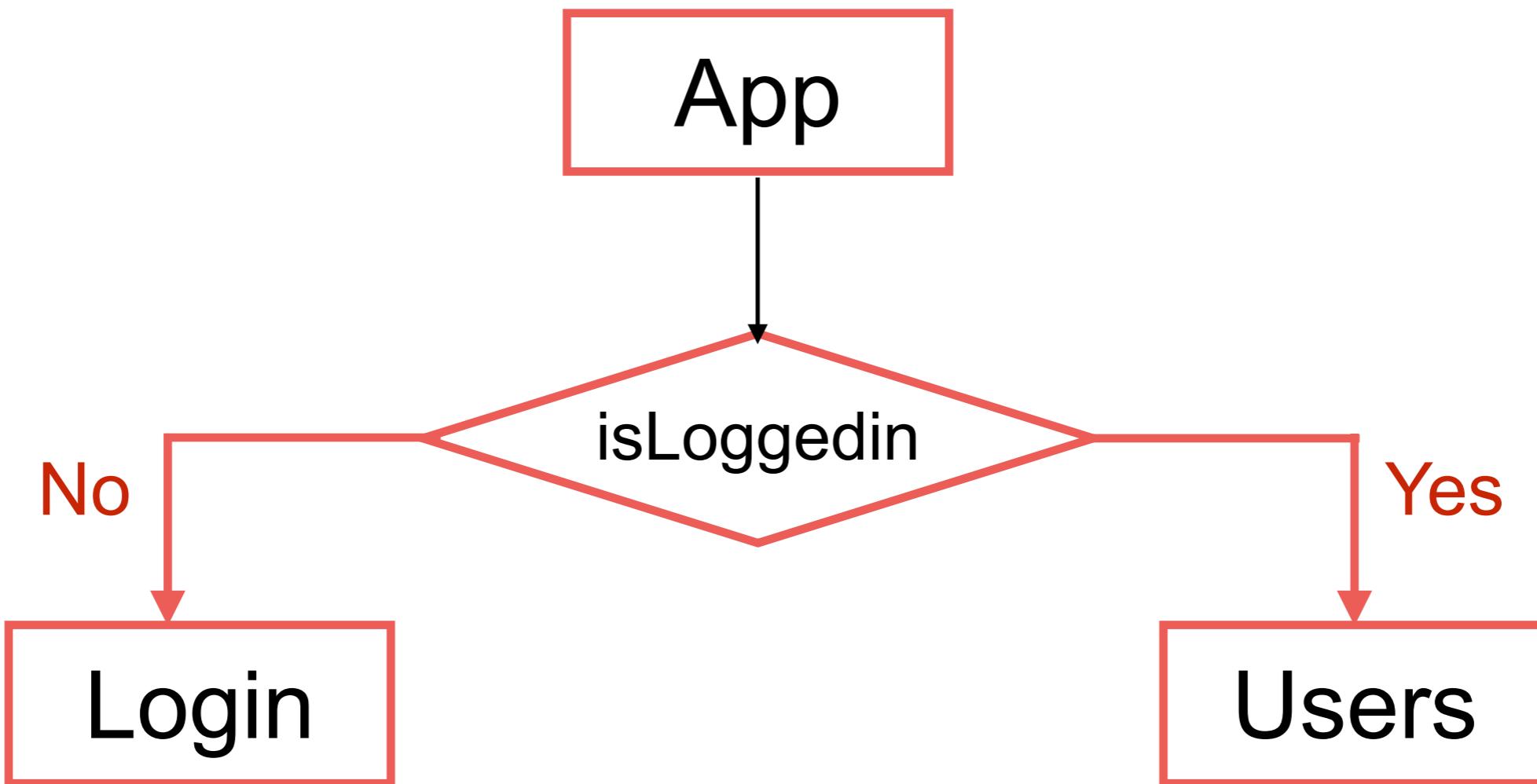
Id	First Name	Last Name	Email
1	User 1	Last name 1	user1@gmail.com
2	User 2	Last name 2	user2@gmail.com
3	User 3	Last name 3	user3@gmail.com



Components



Components



Login Page



Login Page

Login Page

Email address:

Password:

Login



Create new component with CLI

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



Create Login Component

\$ng generate 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 (619 bytes)
CREATE src/app/login/login.component.ts (271 bytes)
UPDATE src/app/app.module.ts (455 bytes)
```



Edit file login.component.ts

Add fields to keep email and password

```
export class LoginComponent implements OnInit {  
  
  form: any = {  
    email: null,  
    password: null  
};  
  
constructor() { }  
  
ngOnInit(): void {  
}  
  
onLogin() {  
  const {email, password} = this.form;  
  console.table(this.form);  
}  
}
```



Edit file login.component.html

Use two-way binding with Form

```
<p>Login page</p>

<form (ngSubmit)="onLogin()" >

  Email: <input type="text" name="email"
                [(ngModel)]="form.email" />

  Password: <input type="password" name="password"
                 [(ngModel)]="form.password" />

  <button>Login</button>

</form>
```



Run and see result

Login Page

Email address:

Password:

Login



List of User Page



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 (612 bytes)
CREATE src/app/user/user.component.ts (267 bytes)
UPDATE src/app/app.module.ts (529 bytes)
```



Create Model of User

\$ng generate class models/user

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



Edit file user.component.ts

Create **fake data** of users

```
import { User } from './models/user';

export class UserComponent implements OnInit {

  users: User[] = [];

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

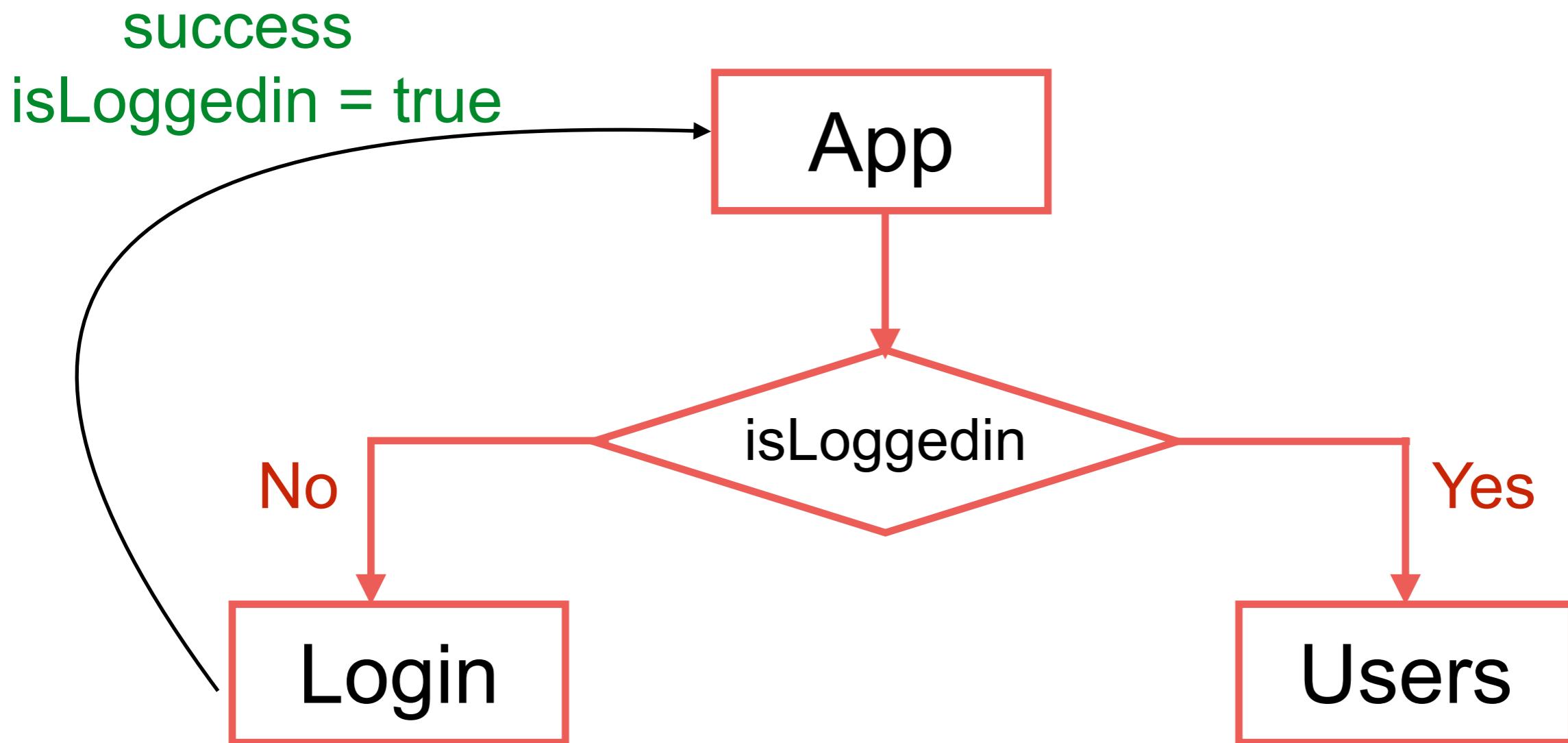


Edit file user.component.html

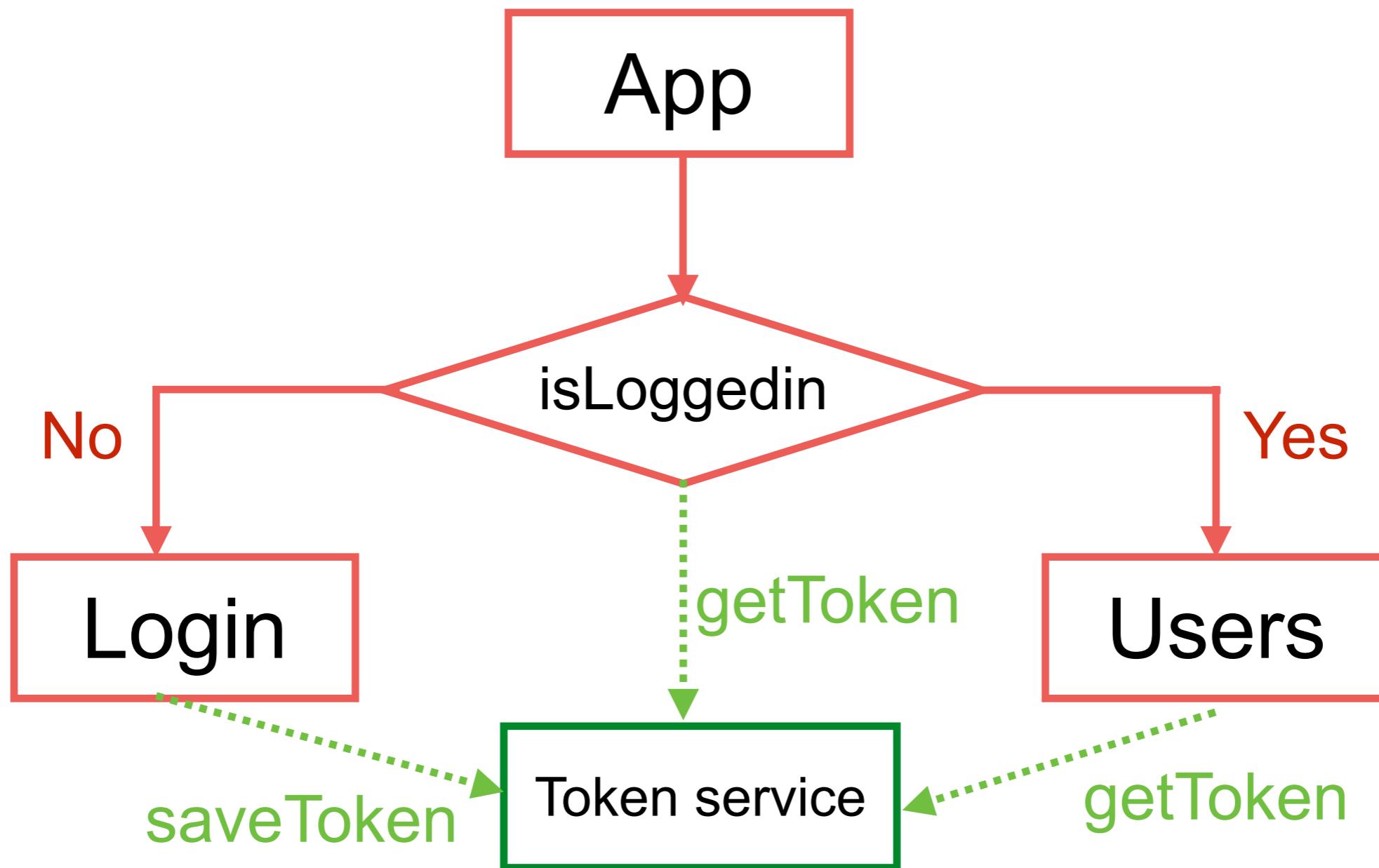
```
<tr *ngFor="let user of users">
  <td>{{user.id}}</td>
  <td>{{user.firstName}}</td>
  <td>{{user.lastName}}</td>
  <td>{{user.email}}</td>
</tr>
```



Components

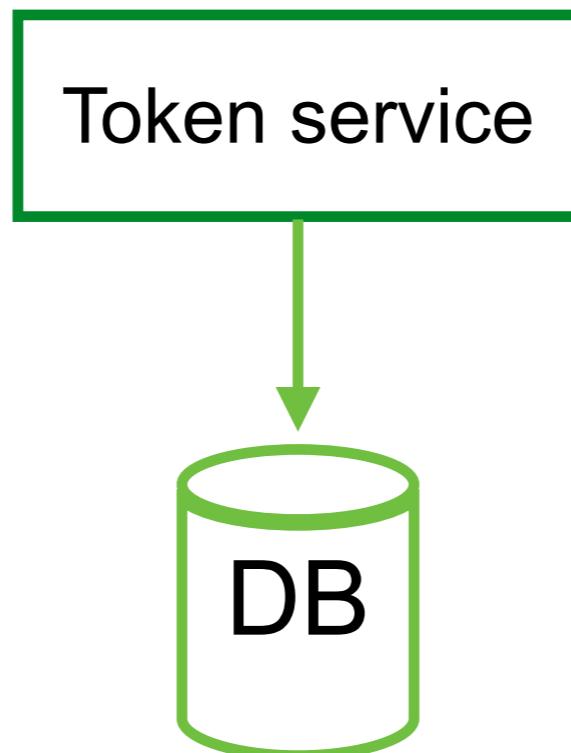


Components



Create Token service

\$ng generate service services/token

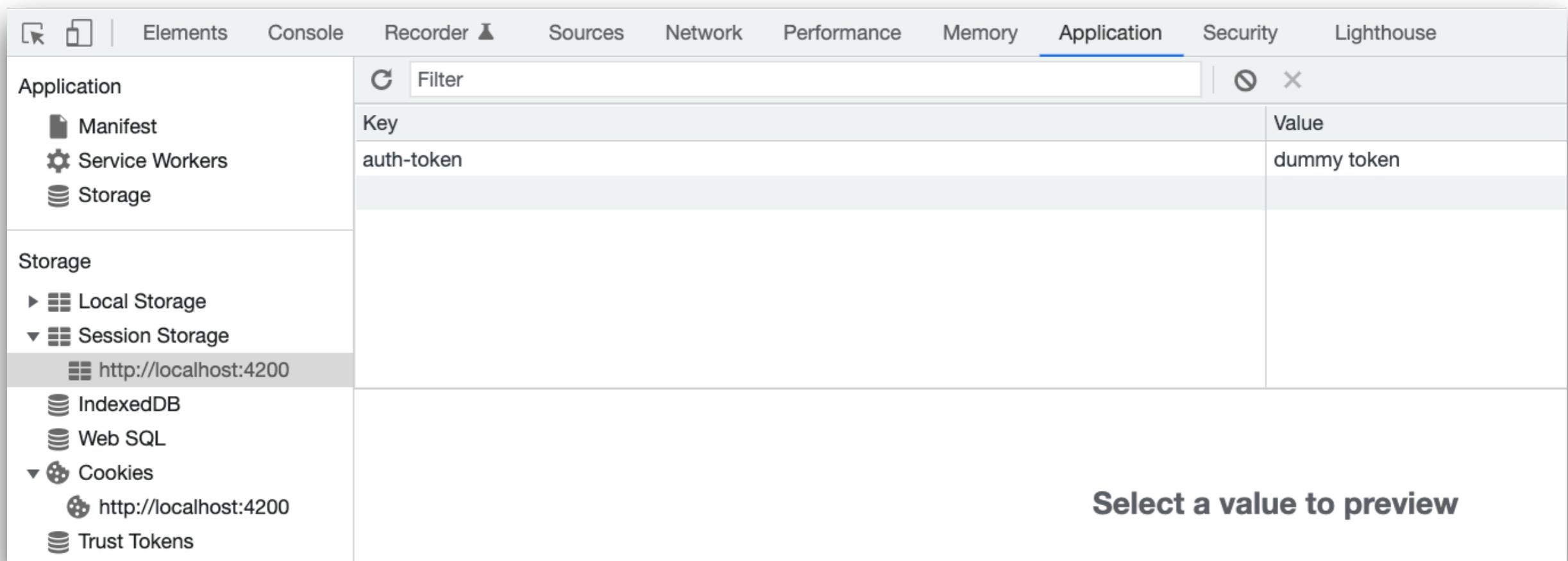


<https://github.com/up1/course-javascript/wiki/Token-service>



Create Token service

Keep data in session storage



The screenshot shows the Chrome DevTools Application tab open. On the left, there's a sidebar with 'Application' (Manifest, Service Workers, Storage), 'Storage' (Local Storage, Session Storage, IndexedDB, Web SQL, Cookies, Trust Tokens), and a preview area for selected items. The 'Session Storage' section is expanded, and 'http://localhost:4200' is selected. In the main pane, a table shows session storage data:

Key	Value
auth-token	dummy token

A message at the bottom right says 'Select a value to preview'.

<https://github.com/up1/course-javascript/wiki/Token-service>



App component

Get data from token service

```
export class AppComponent {  
  isLoggedIn = true;  
  
  constructor(private tokenService: TokenService) { }  
  
  ngOnInit(): void {  
    this.isLoggedIn = !!this.tokenService.getToken();  
    if (this.isLoggedIn) {  
      // TODO  
    }  
  }  
}
```



Login component

Save token when login succeed

```
export class LoginComponent implements OnInit {  
  constructor(private tokenService: TokenService) { }  
  
  onLogin() {  
    const {email, password} = this.form;  
  
    this.tokenService.saveToken("dummy token");  
    window.location.reload();  
  }  
}
```



Add rating for user



Add rating for user

List of users

#	First name	Last name	Email	Rating
1	User 1	Last name 1	user1@xxx.com	★
2	User 2	Last name 2	user2@xxx.com	★★★★★

<https://fontawesome.com/v4/get-started/>



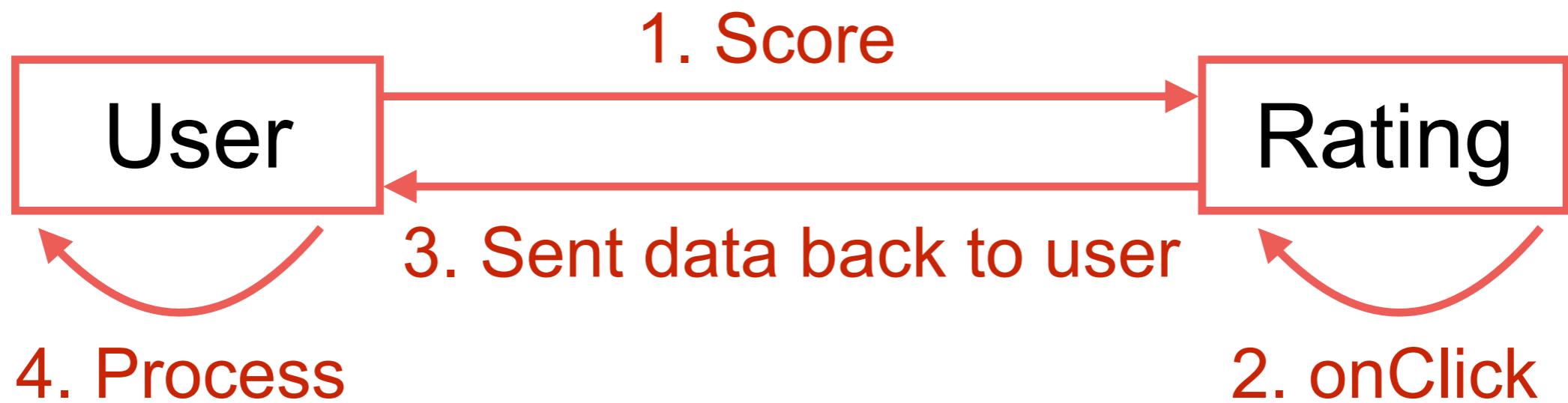
Create User Component

\$ng generate component rating

```
CREATE  src/app/rating/rating.component.css (0 bytes)
CREATE  src/app/rating/rating.component.html (21 bytes)
CREATE  src/app/rating/rating.component.spec.ts (626 bytes)
CREATE  src/app/rating/rating.component.ts (275 bytes)
UPDATE  src/app/app.module.ts (611 bytes)
```

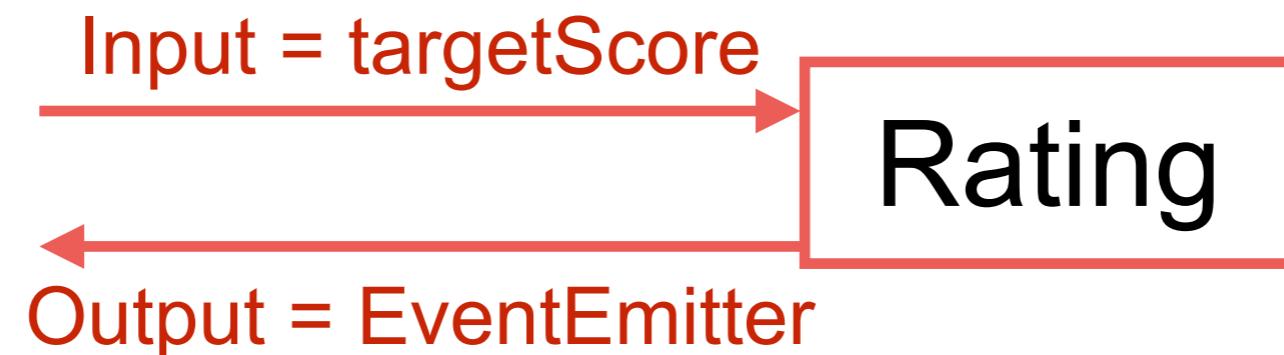


Communication



Rating component

```
export class RatingComponent implements OnInit {  
  constructor() {}  
  
  @Input() targetScore = 0;  
  starWidth = 0;  
  
  @Output() ratingClicked: EventEmitter<string> = new EventEmitter<string>();
```



Rating event handling

```
<div  
  [style.width.px] = "starWidth"  
  style = "overflow: hidden;"  
  (click) = "onClicked()"  
>  
  <div style = "width: 75px">  
    <span class = "fa fa-star"></span>  
    <span class = "fa fa-star"></span>  
  </div>  
</div>
```

```
onClicked(): void {  
  console.log('some data at ' + this.targetScore);  
  this.ratingClicked.emit('some data at ' + this.targetScore);  
}
```



User component

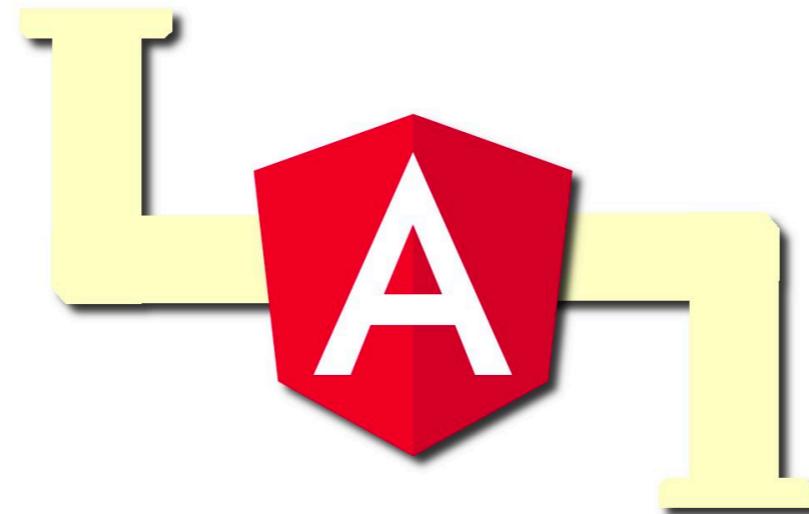
Send and receive data from rating component

```
<td>
  <app-rating
    [targetScore]=u.score
    (ratingClicked)="onClickedFromRating($event)"
  ></app-rating>
</td>
```

```
onClickedFromRating(event: string): void {
  console.log('Received data from rating= ' + event);
  this.result = 'Received data from rating= ' + event;
}
```



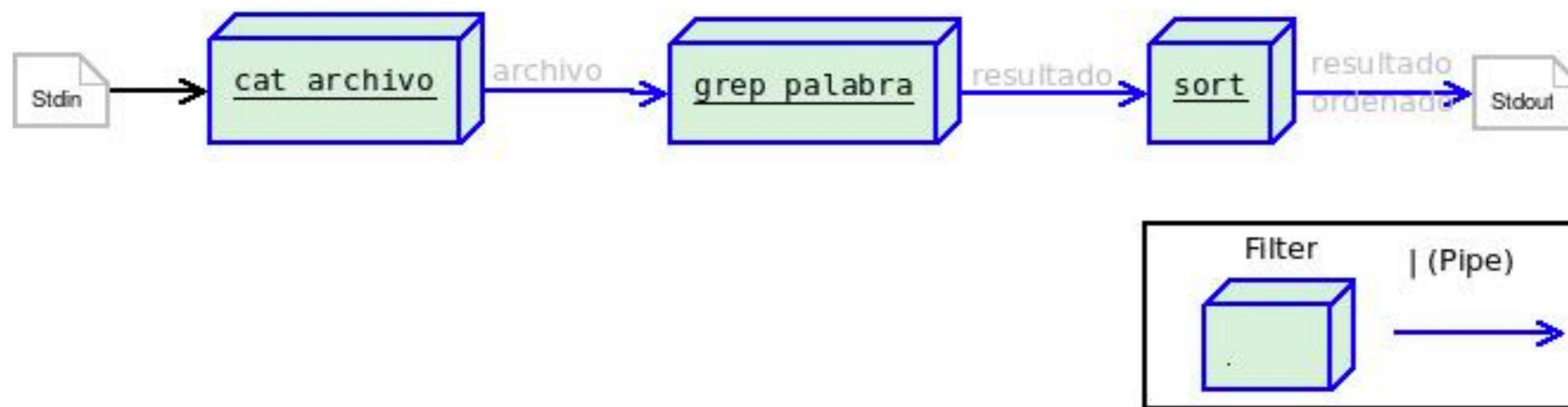
Transform data with Pipe



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



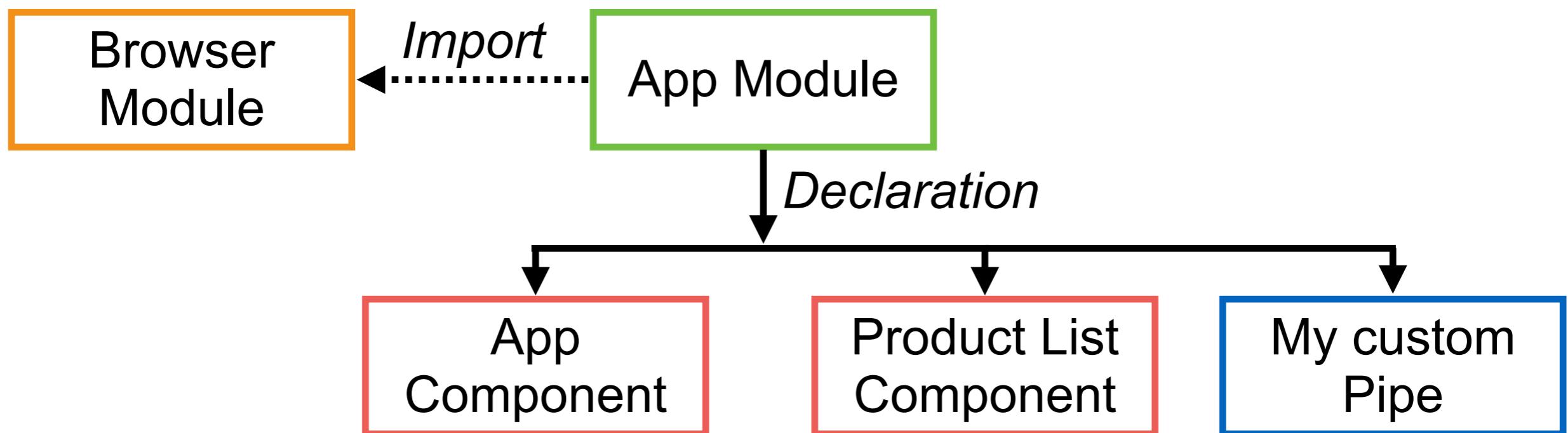
Build-in pipes

```
<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:	Product Code	Product Name
<input type="text"/>	AA-0001	PRODUCT NAME 1
<input type="text"/>	BB-0002	PRODUCT NAME 2



Routing ?



We need routing ?

Angular have @angular/router

/login

Login Page

Email address:

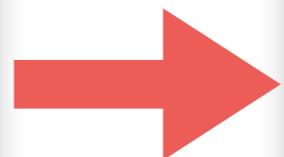
Password:

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



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]
})
```



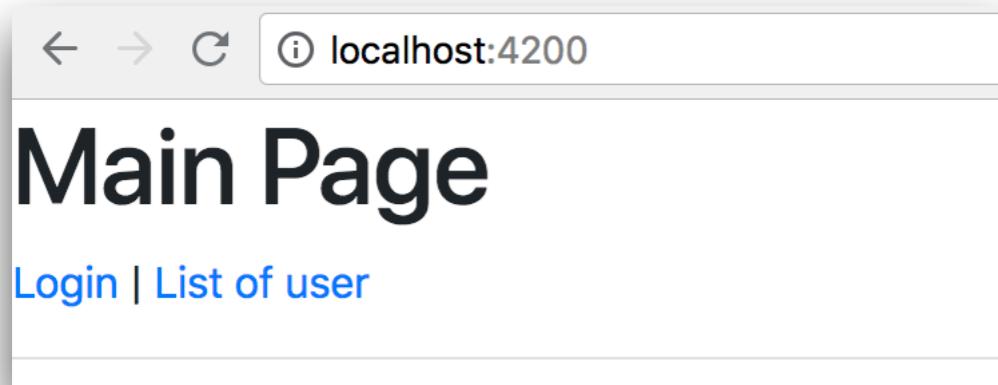
Edit app.component.html

Create template of app component

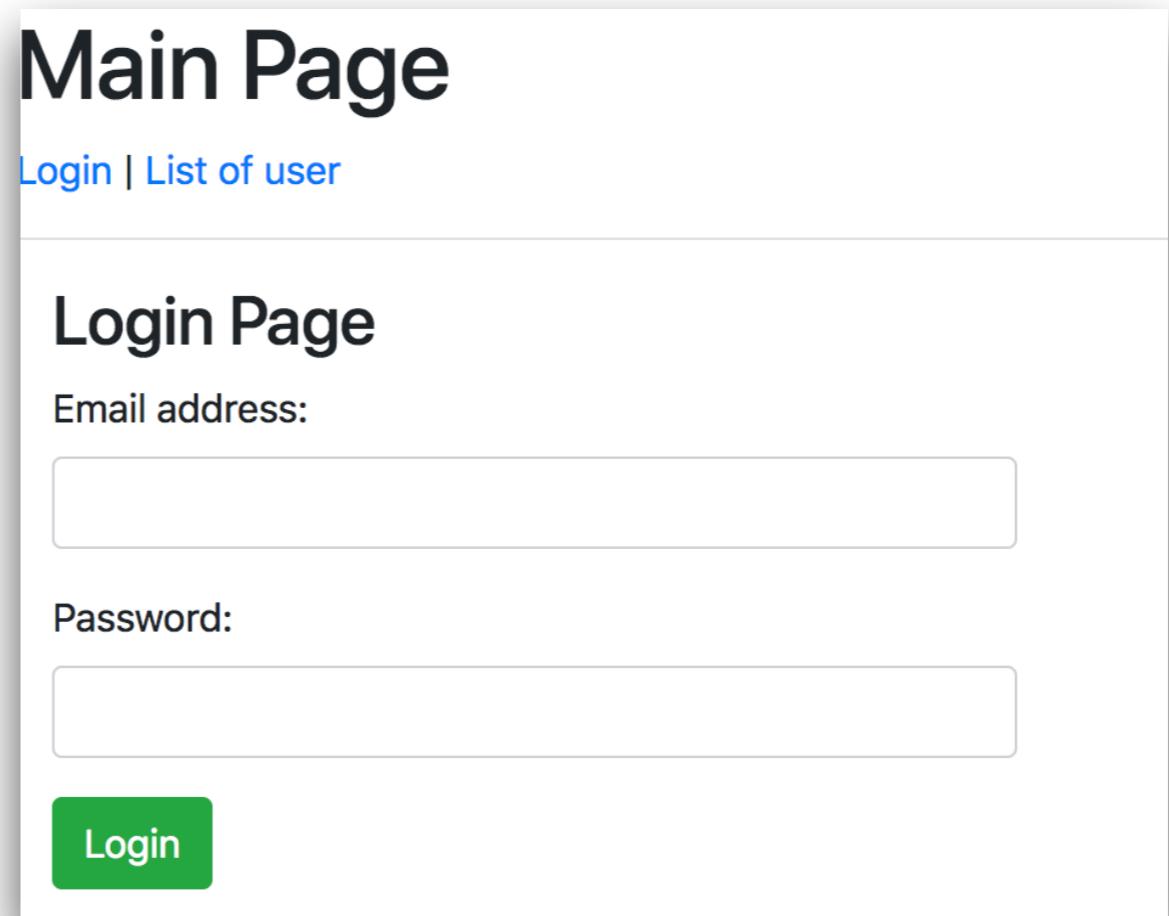
```
<h1 class="title">Main Page</h1>
<nav>
  <a routerLink="/login" routerLinkActive="active">Login</a> |
  <a routerLink="/user" routerLinkActive="active">List of user</a>
</nav>
<hr/>
<router-outlet></router-outlet>
```



See result in first page



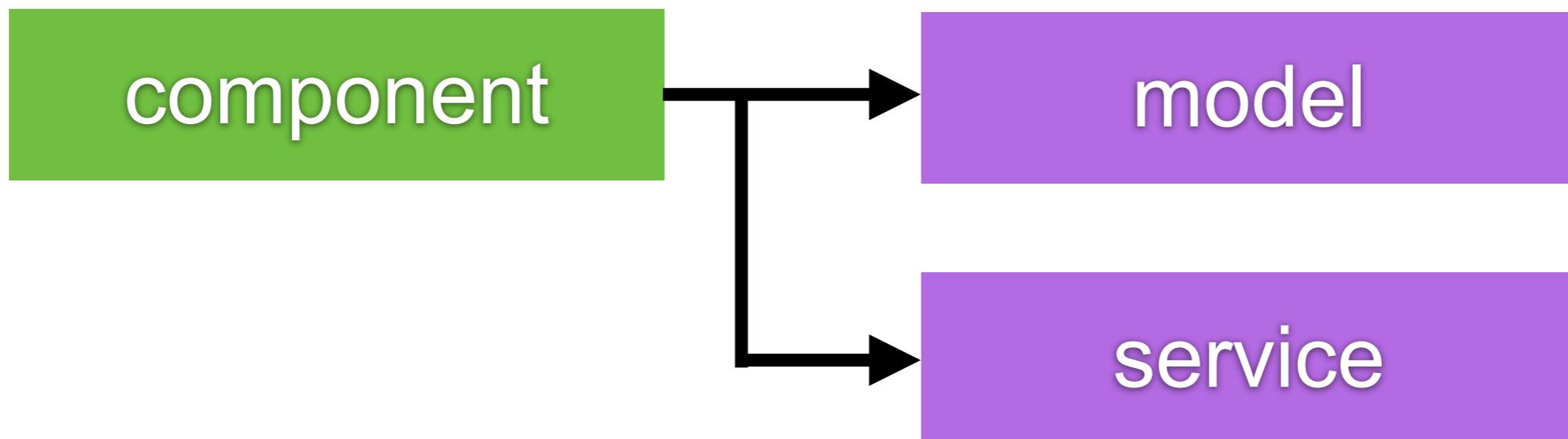
A screenshot of a web browser window. The address bar shows 'localhost:4200'. The main content area displays the text 'Main Page' in large bold letters, followed by a link 'Login | List of user'.



A screenshot of a web browser window. The title bar says 'Main Page'. Below it, there are links 'Login | List of user'. The main content area is titled 'Login Page'. It contains two input fields: one for 'Email address:' and one for 'Password:', both represented by empty text boxes. Below the password field is a green rectangular button labeled 'Login'.



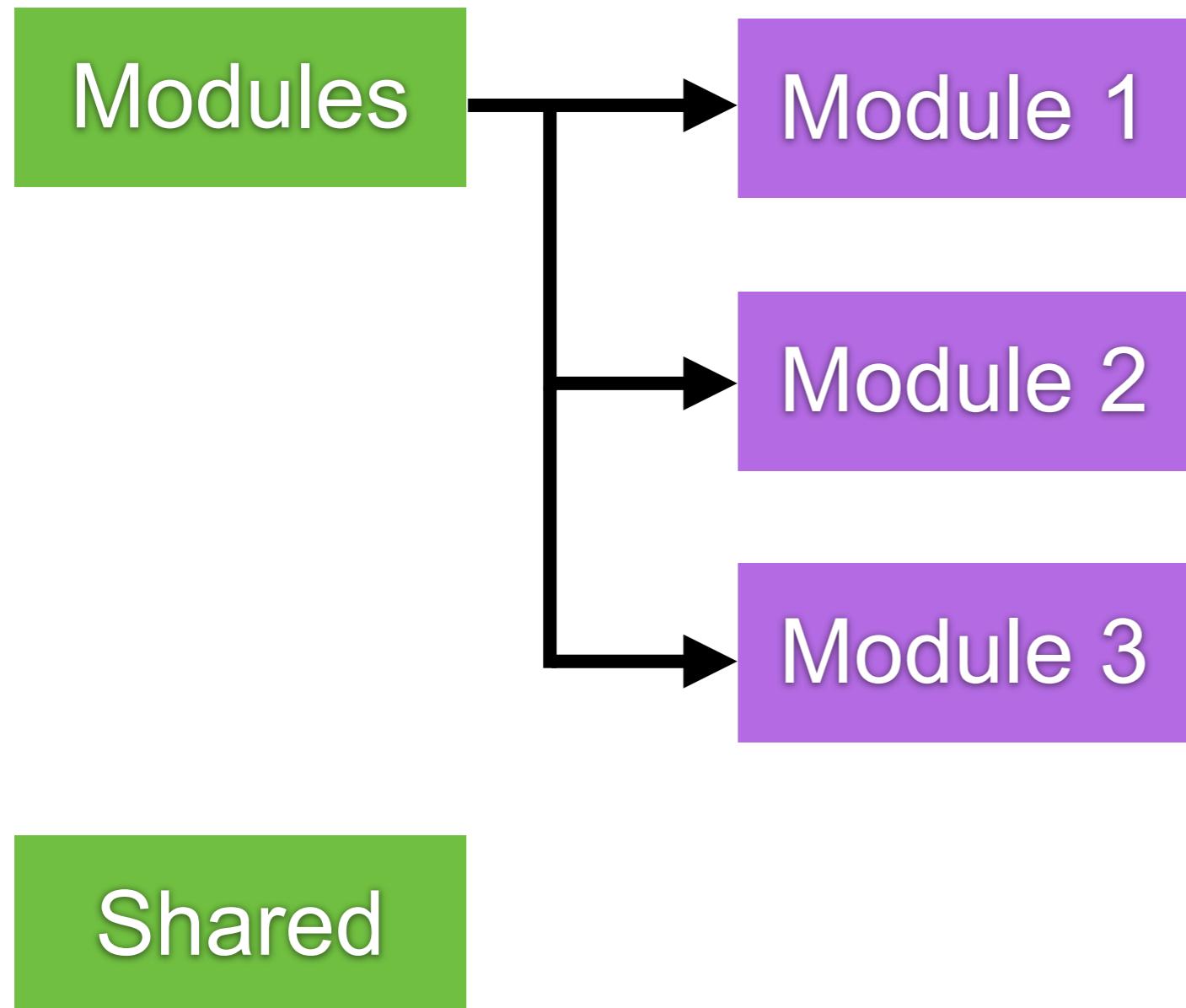
Better Structure of project



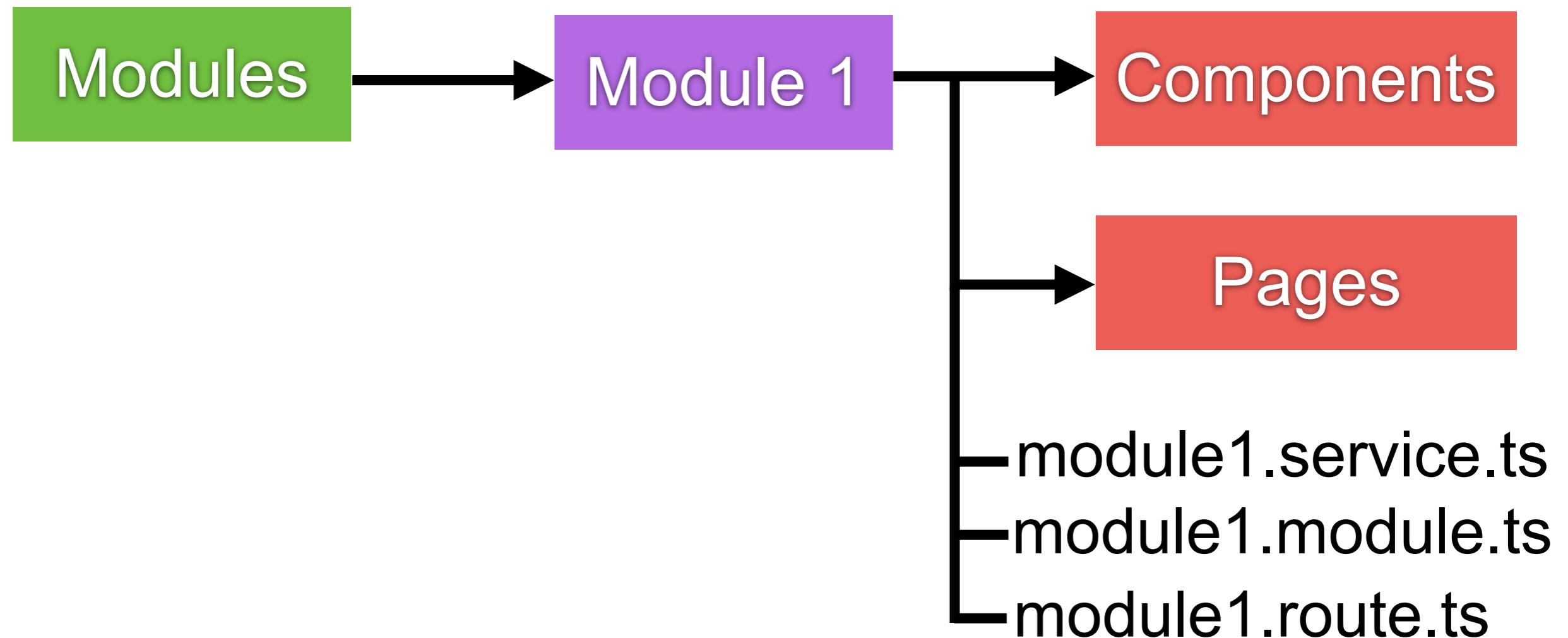
Large Project Structure



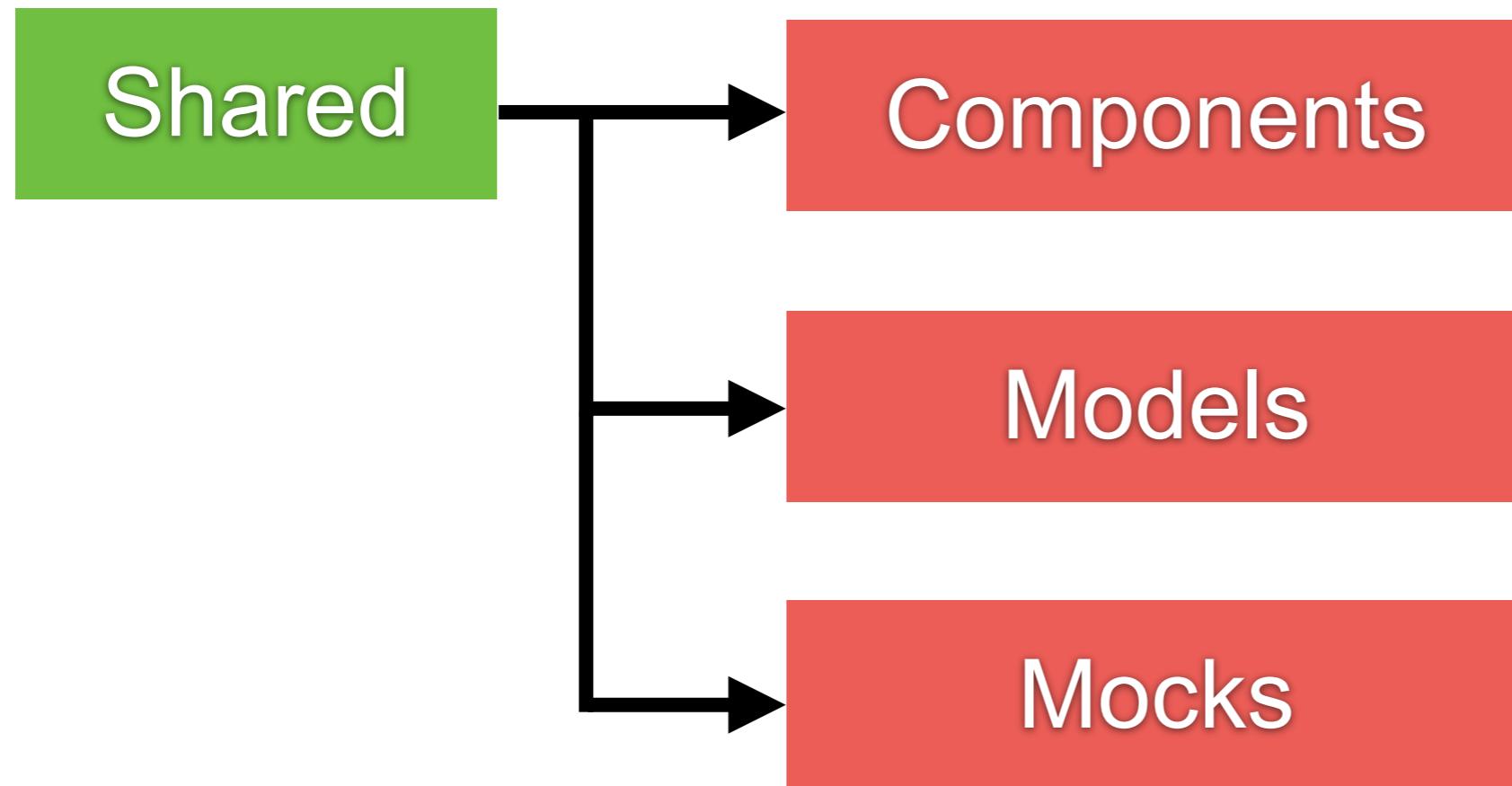
Better Structure of project



Module structure



Shared



BENWARE

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



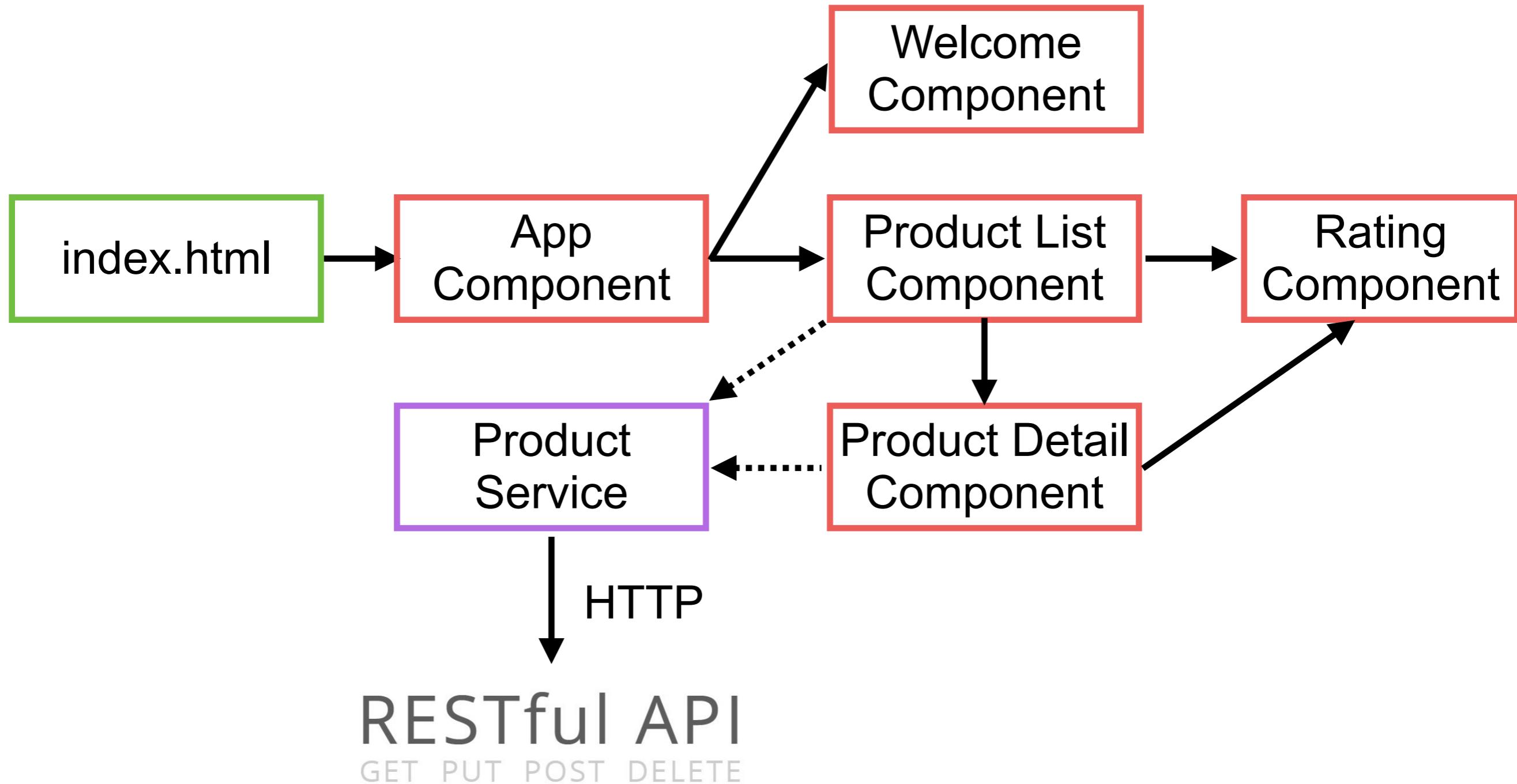
Environment with build



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



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 ...

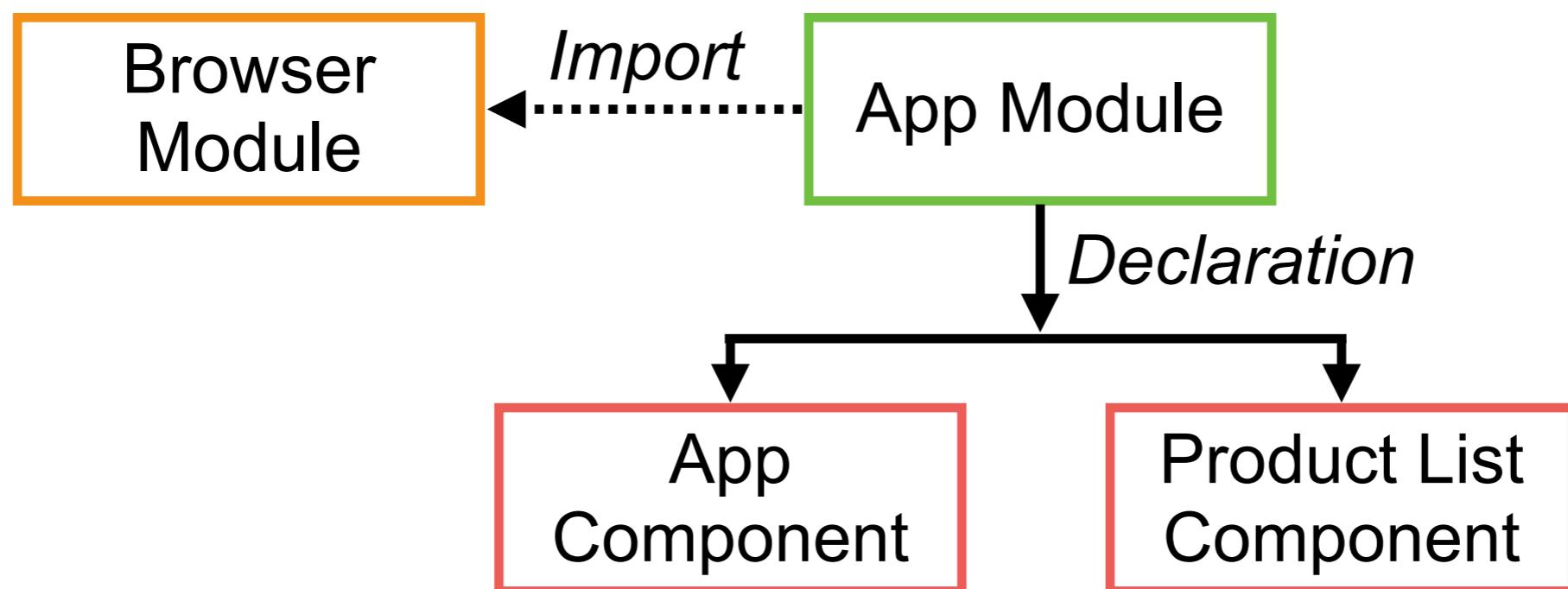
Product Image	Product Code	Product Name	Price	Available	Rating
XXX	0001	Name 01	1000.00	Yes	****
XXX	0002	Name 02	1000.00	Yes	****
XXX	0003	Name 03	1000.00	Yes	****

<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



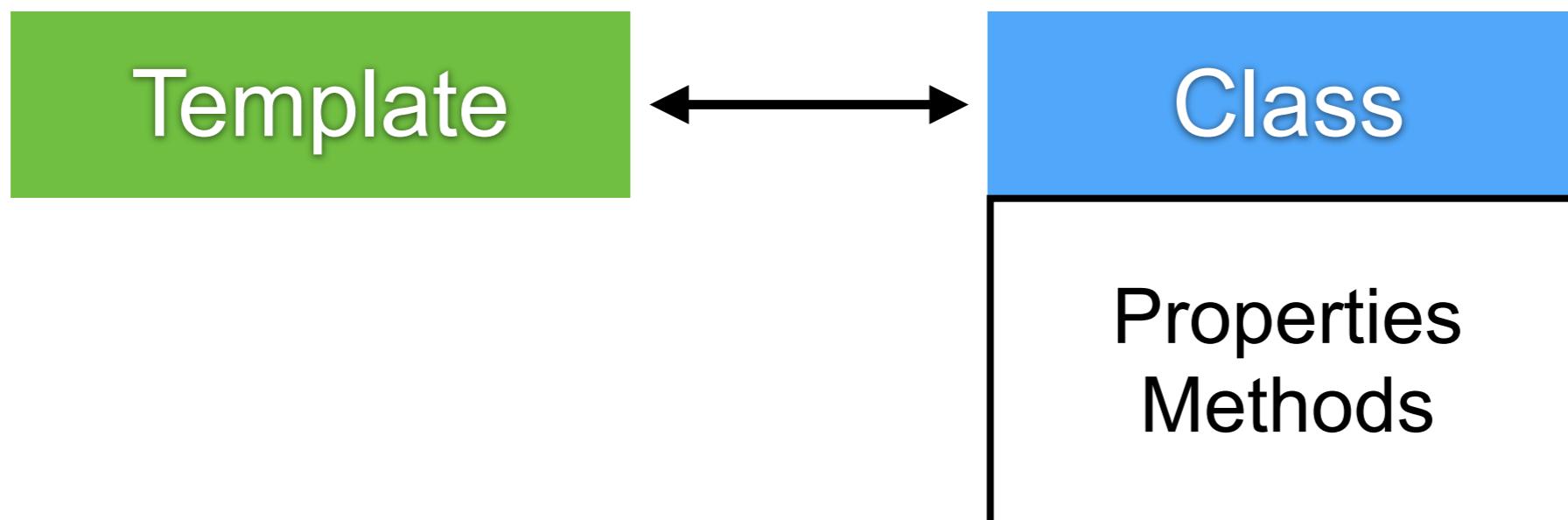
Types of directives

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'  
      *ngIf="products && products.length"  
      v>
```



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 = '';
```



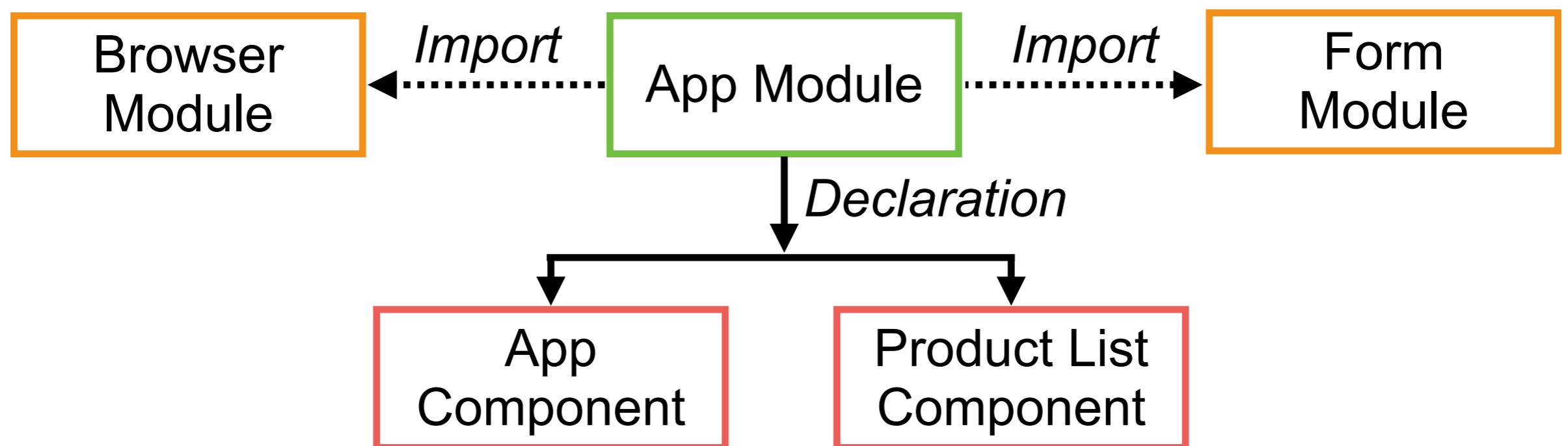
Error !!

✖ 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';

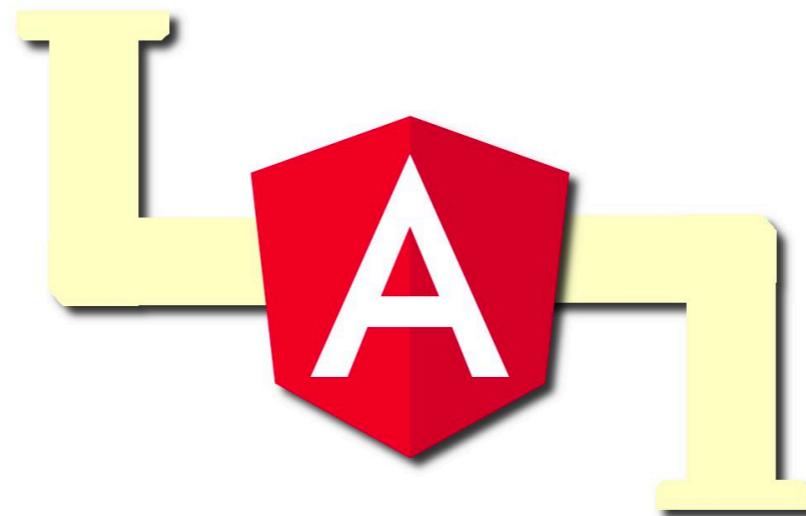
import { FormsModule } from '@angular/forms';

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

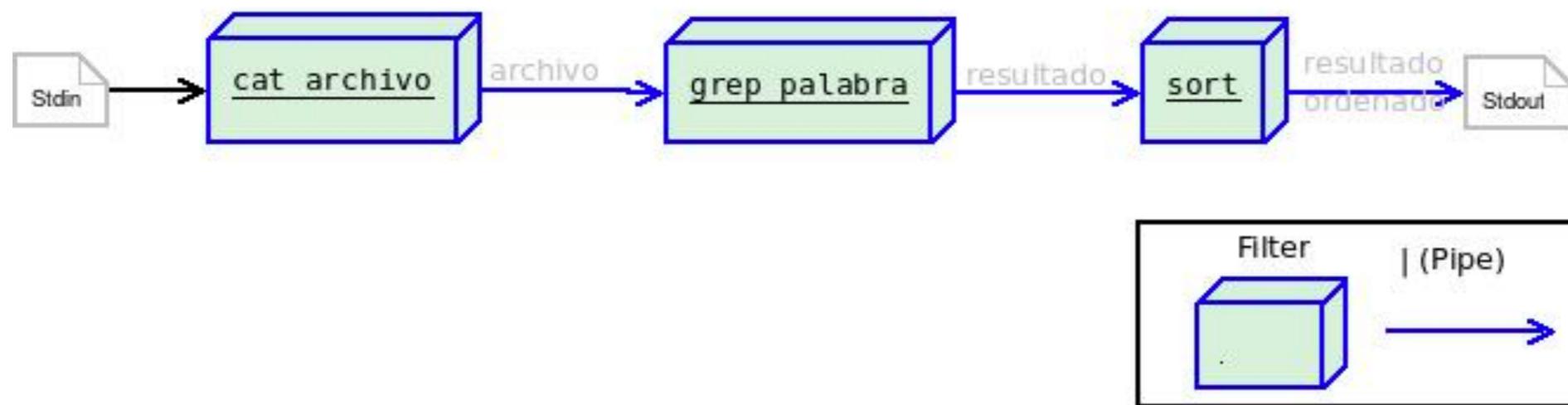
@NgModule({
  declarations: [
    AppComponent,
    ProductListComponent,
  ],
  imports: [
    BrowserModule,
    FormsModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



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>



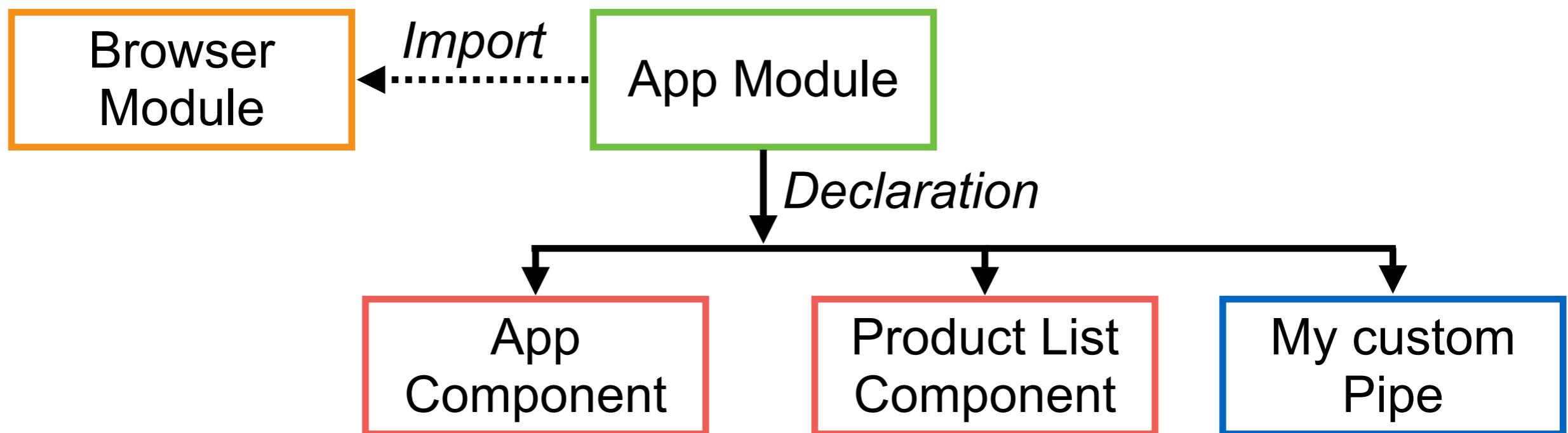
Build-in pipes

```
<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:		
Product 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

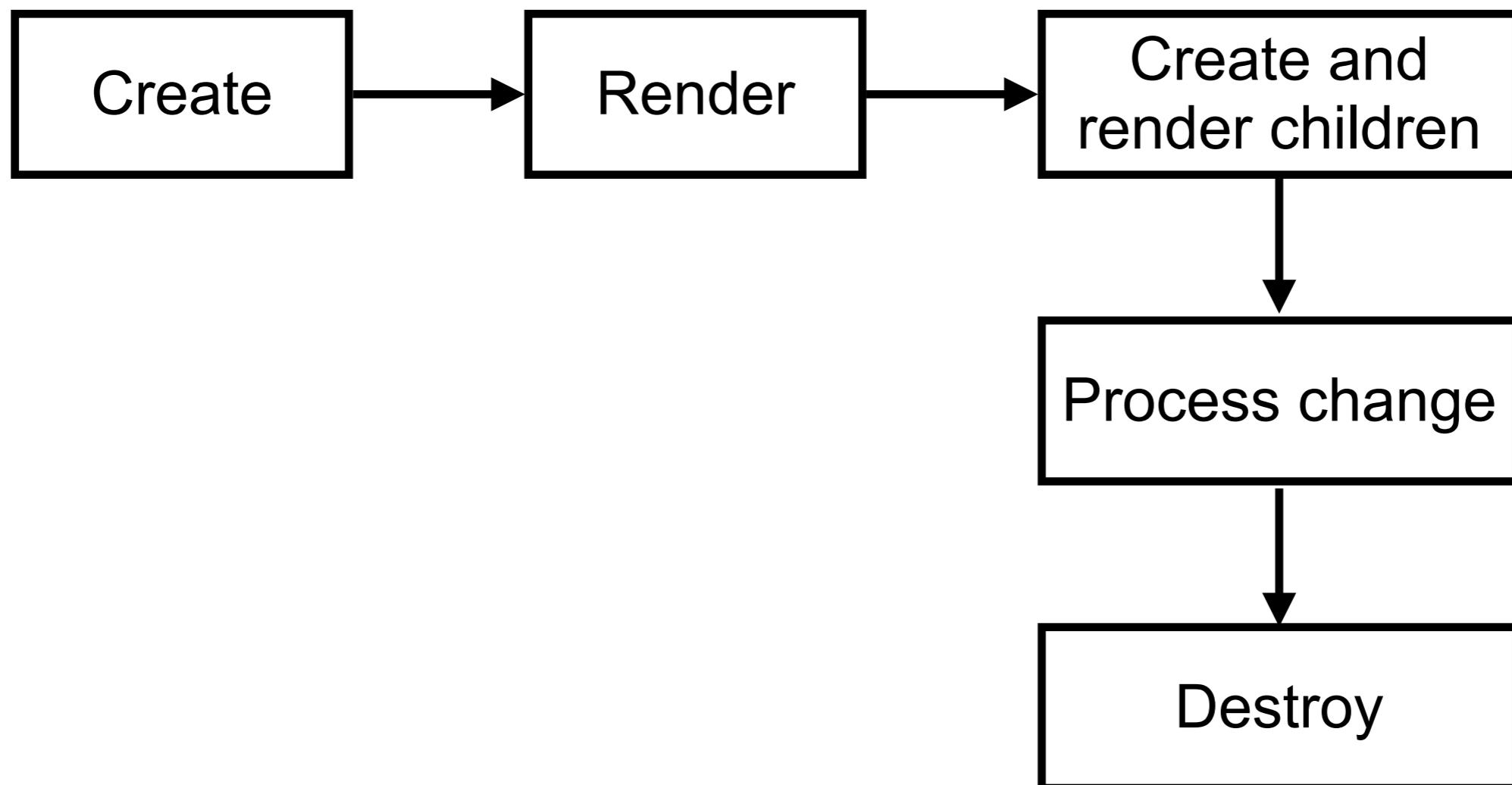
```
@Component({  
  selector: 'app-product-list',  
  templateUrl: './product-list.component.html',  
  
  styles: ['thead {background: #337AB7; color: white}']  
})
```

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



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
----------------------------	--------------	--------------	-------	-----------	--------

 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



Try to run

List of Product Page

Filter by:

Filtered by

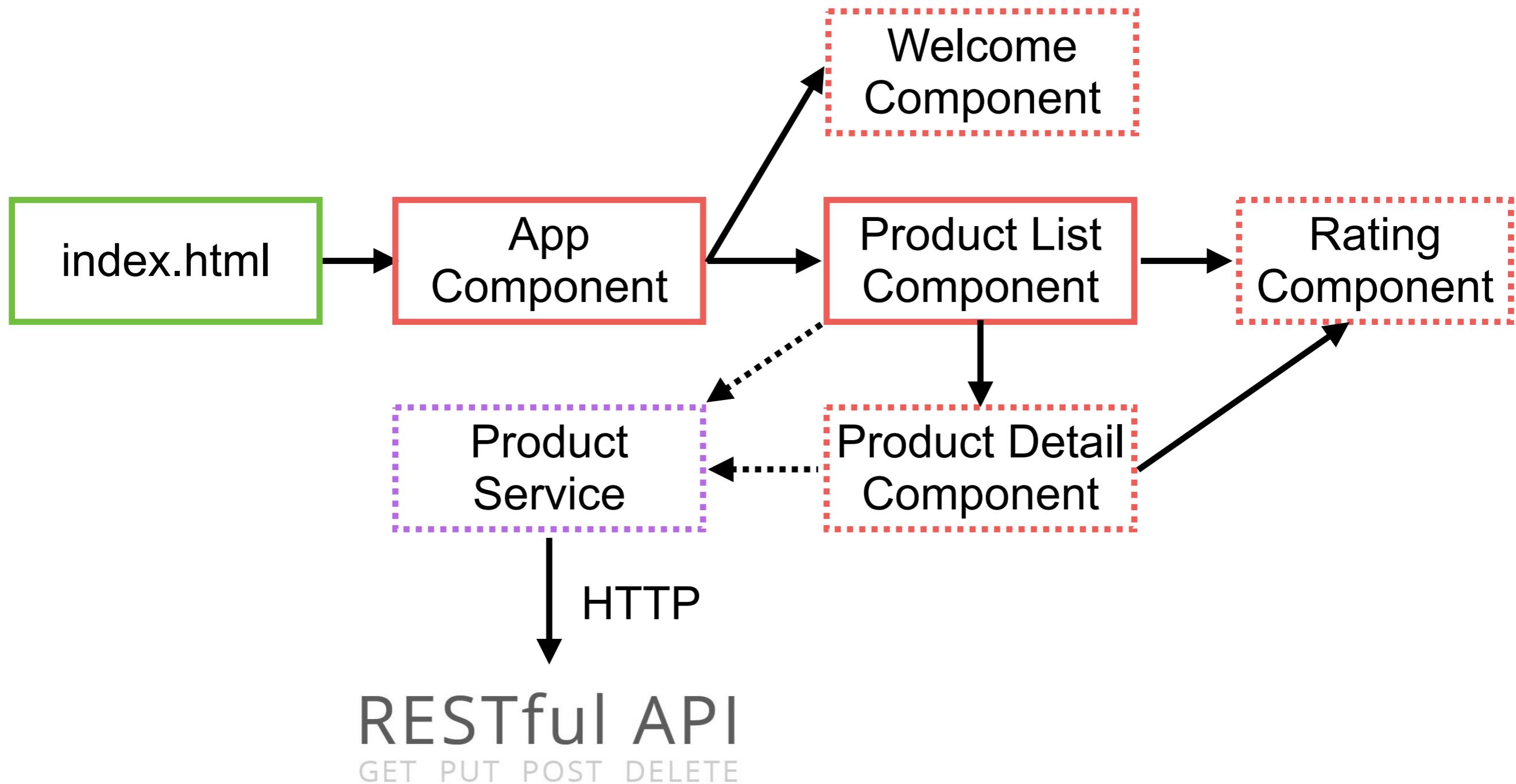
Hide Image	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	****
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	****



Nested components



Application Architecture



Rating component

List of Product Page

Filter by:

Filtered by

<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

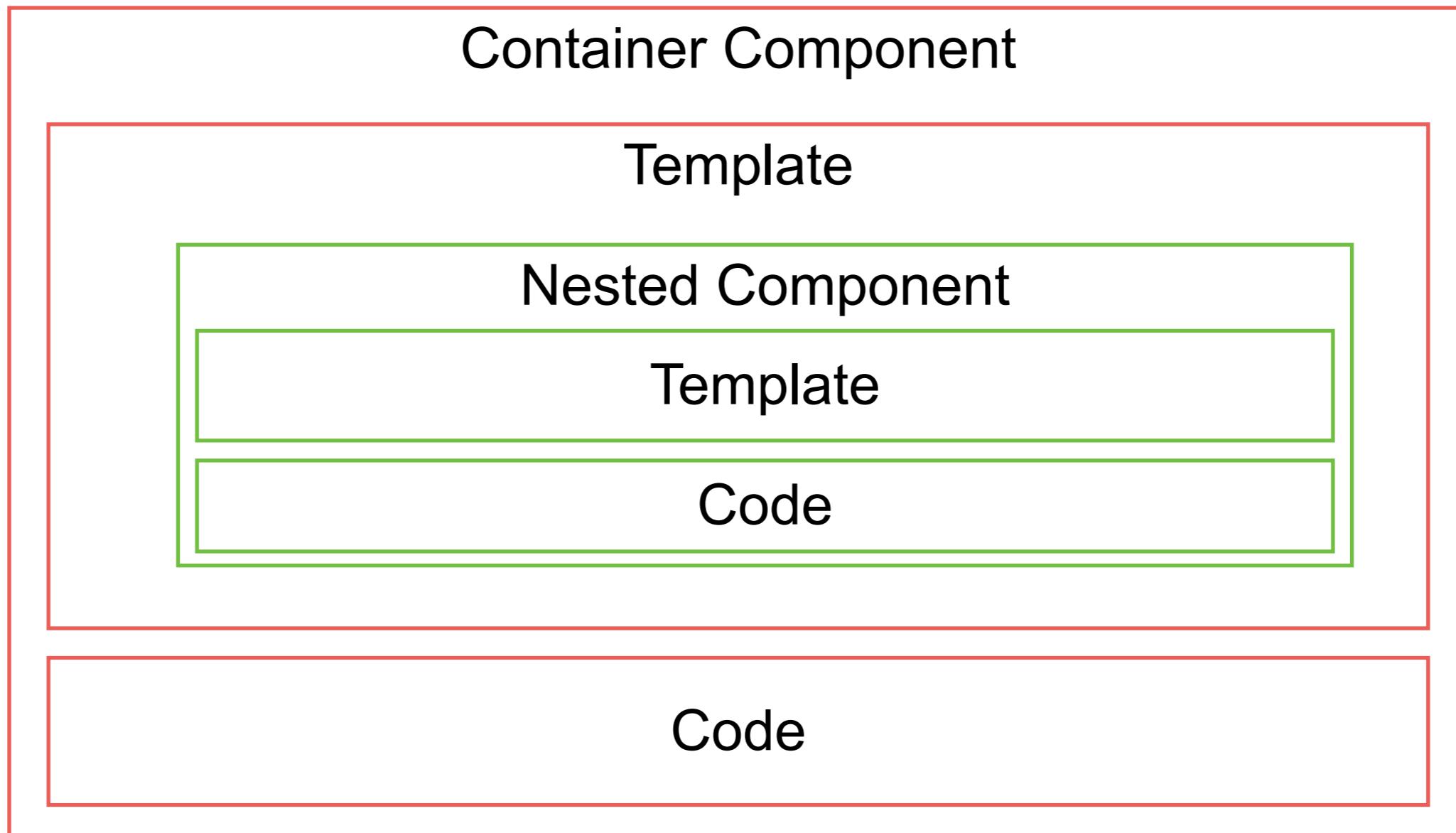
Container Component

Template

Code

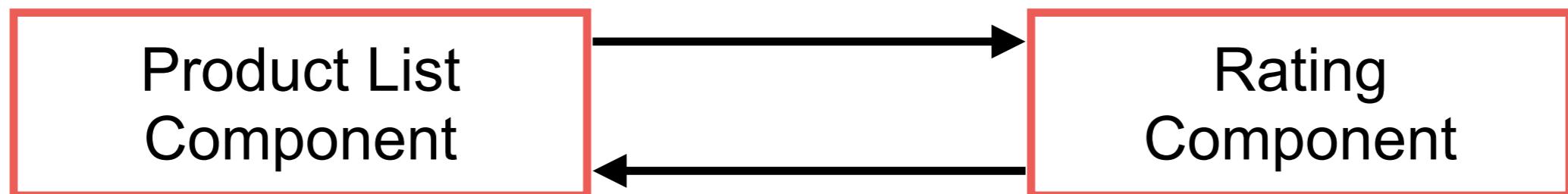


Nested component



Nested component

Passing data using @Input



Raising event using @Output

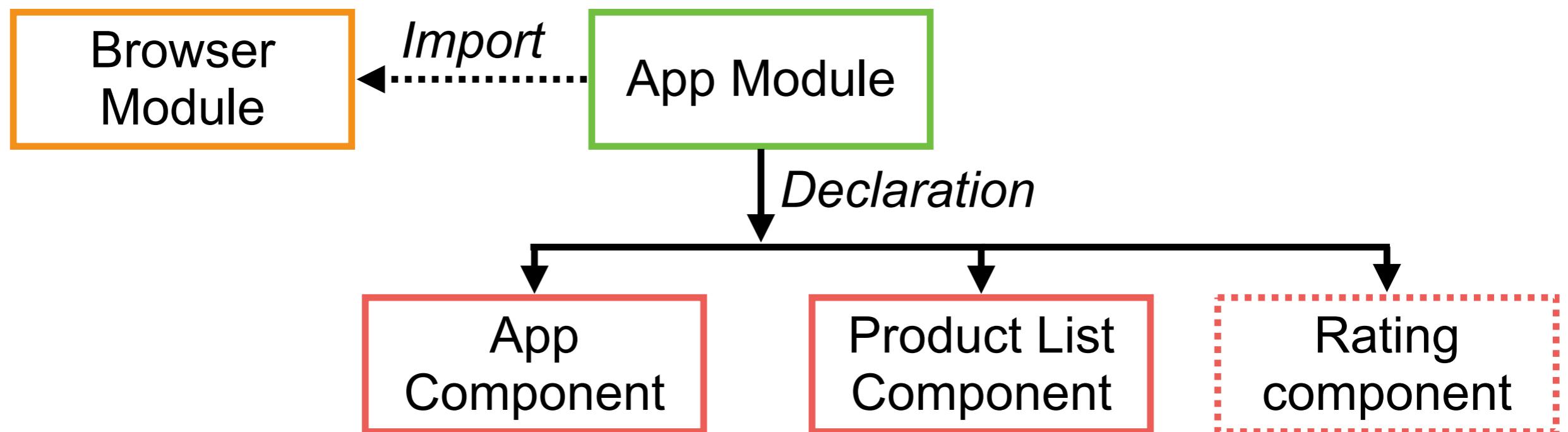


Building component



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,
  ],
})
```

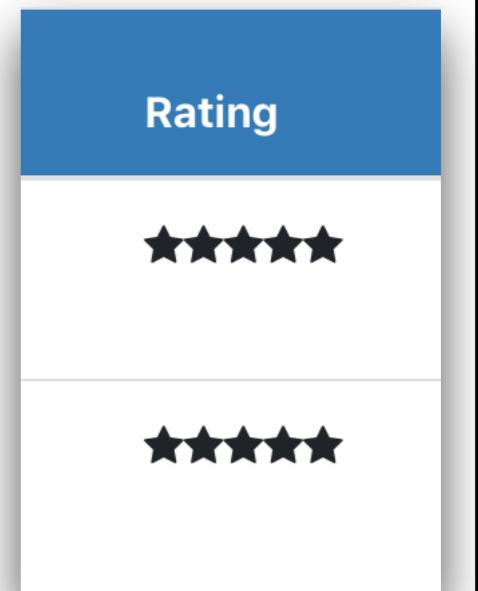


3. Edit template of rating

/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

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	

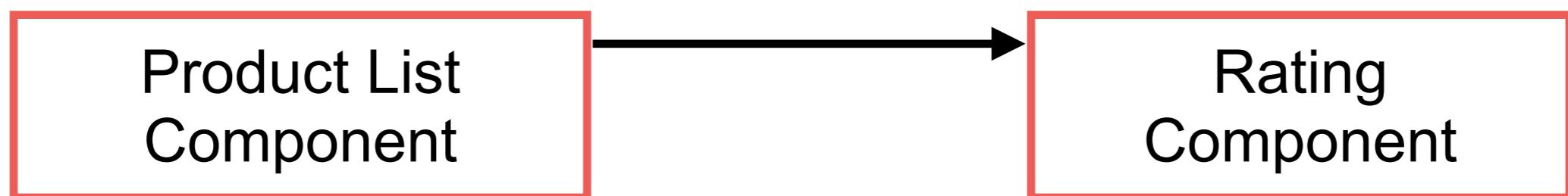


Passing data to component



Passing data to component

Passing data 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

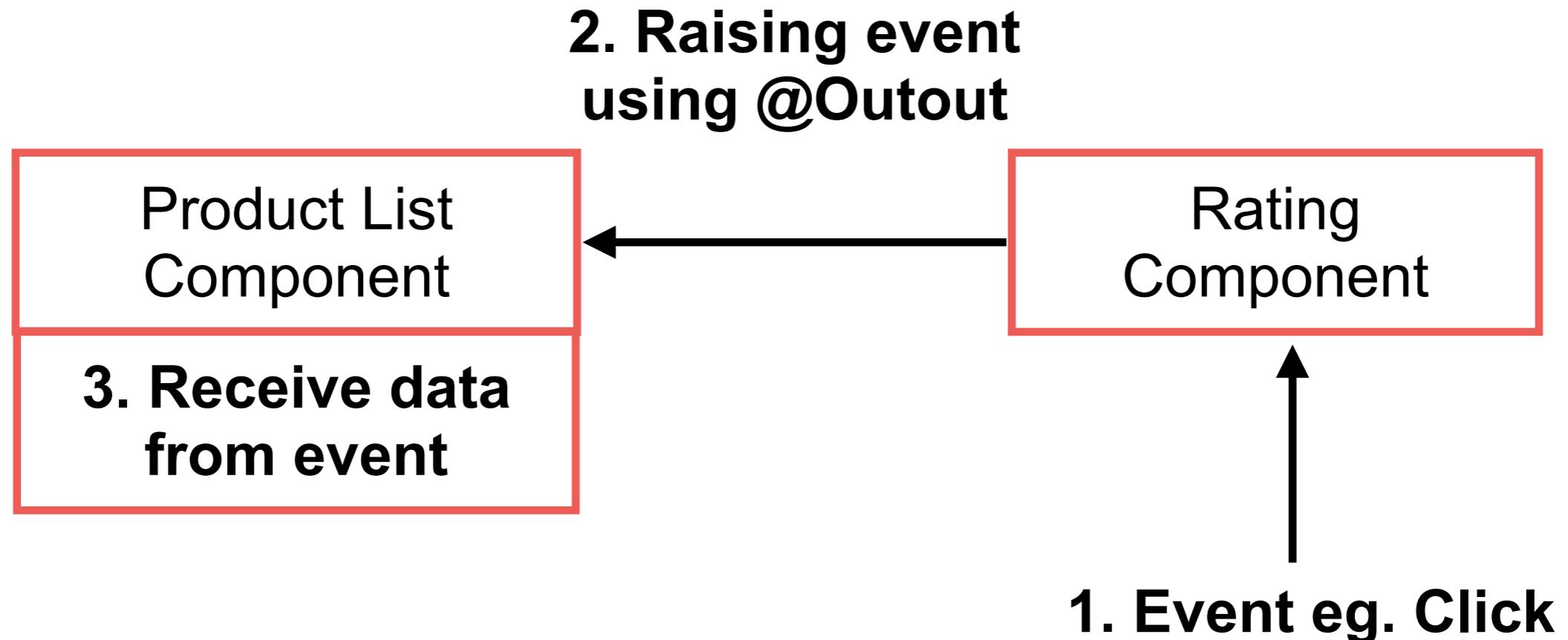
Hide Image	Product Code	Product Name	Price	Available	Rating
	AA-0001	PRODUCT NAME 1	THB 100.00	Yes	
	BB-0002	PRODUCT NAME 2	THB 2,000.00	Yes	
	BB-0003	PRODUCT NAME 3	THB 2,000.00	Yes	



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

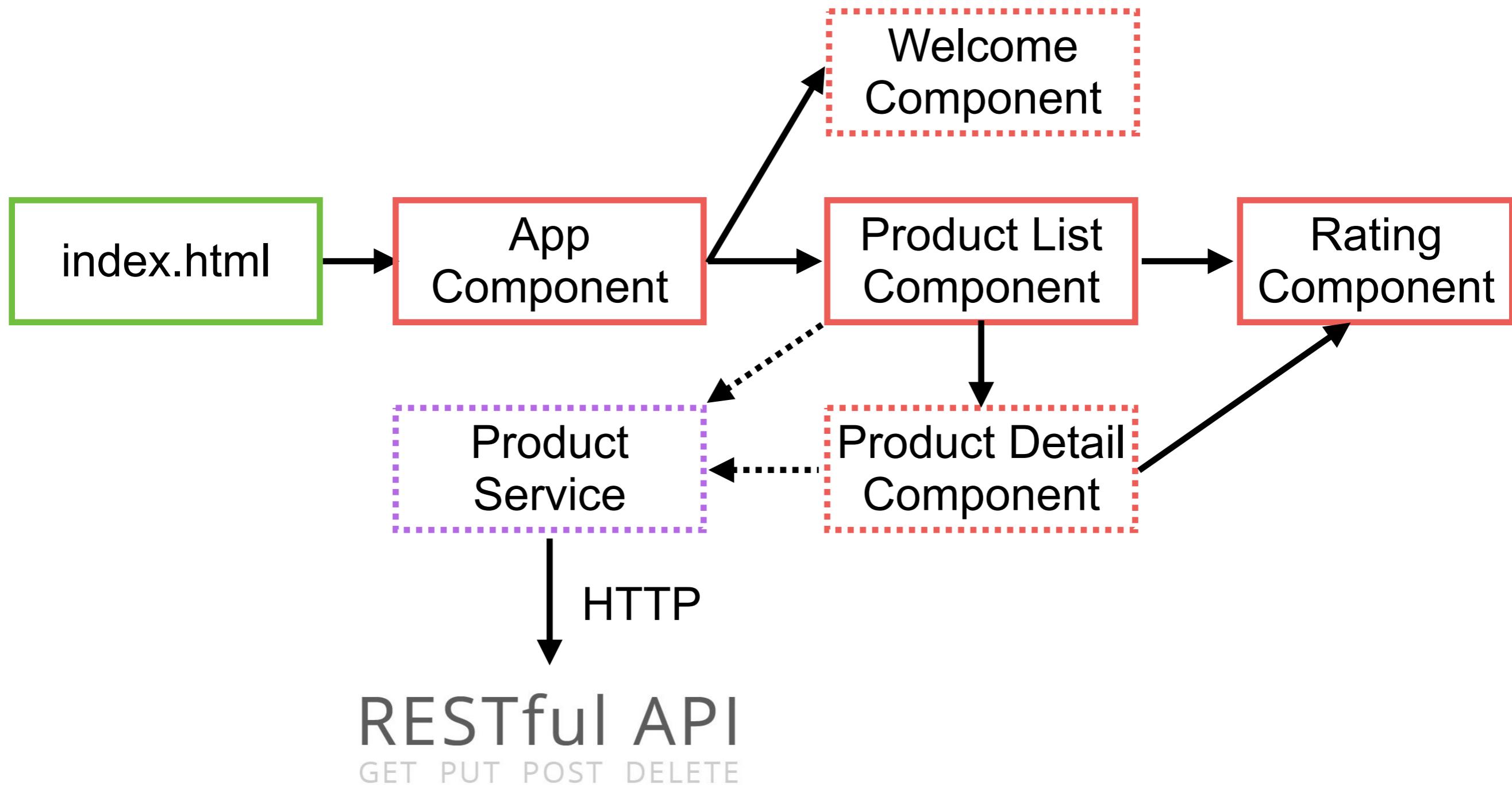
Configure route in each component

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'>
    <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	★★★



Working with route

Passing parameter

Activate route with code

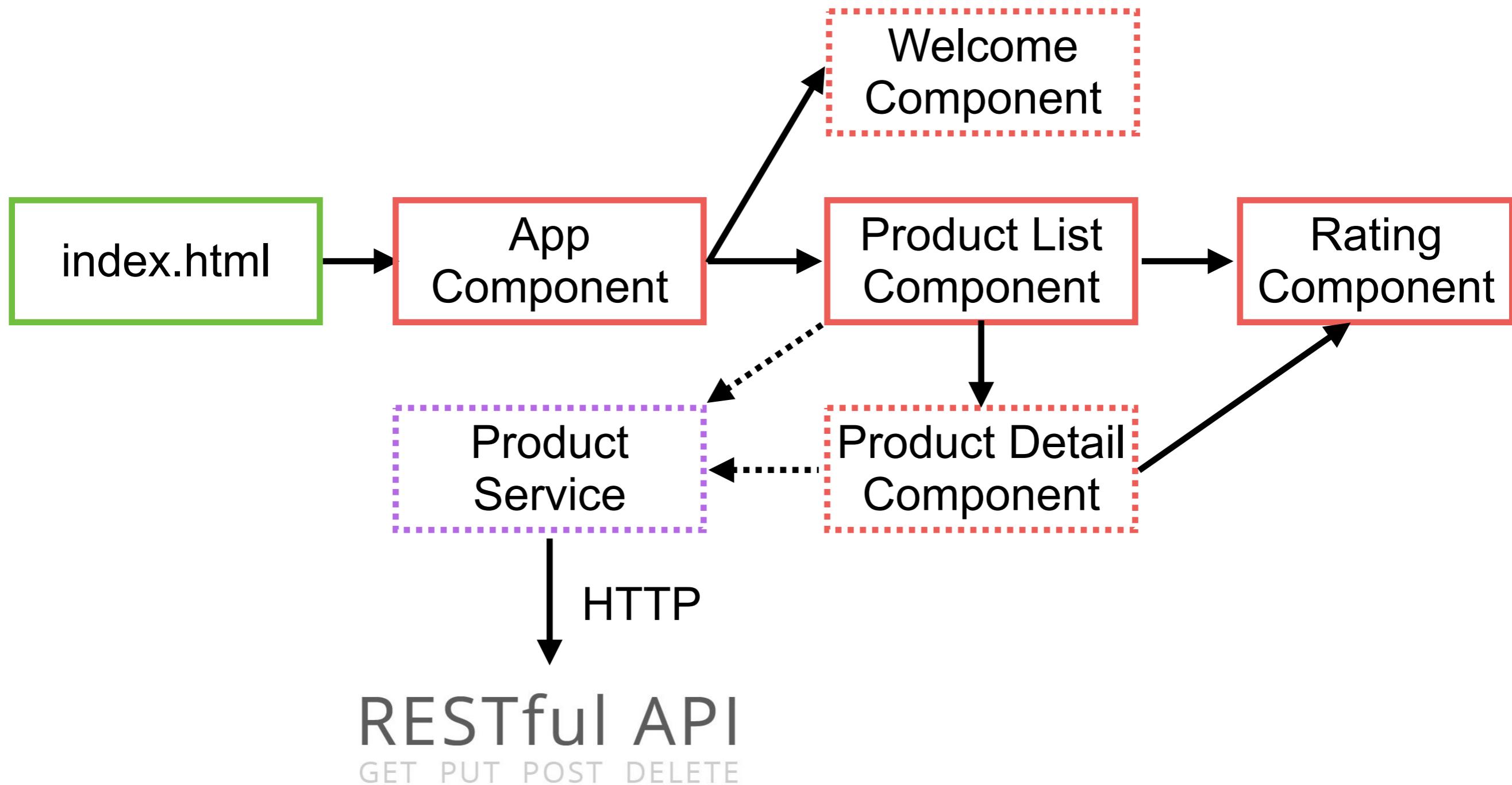
Protect routes with **guards**



Passing parameters with route



Application Architecture



Product detail

URL = /product/:id



Paramater



Create product detail component

```
$ng generate component productDetail
```

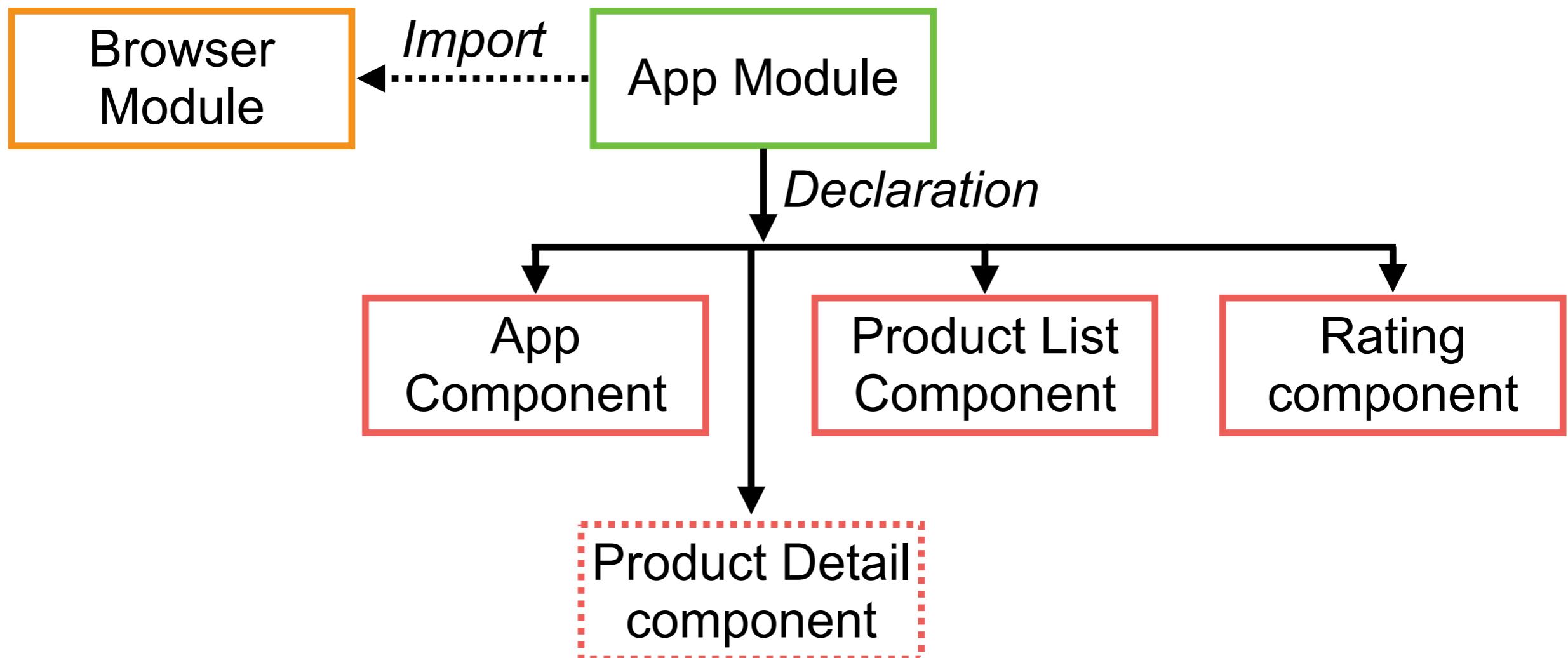
```
$ng generate component product/productDetail --flat
```

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



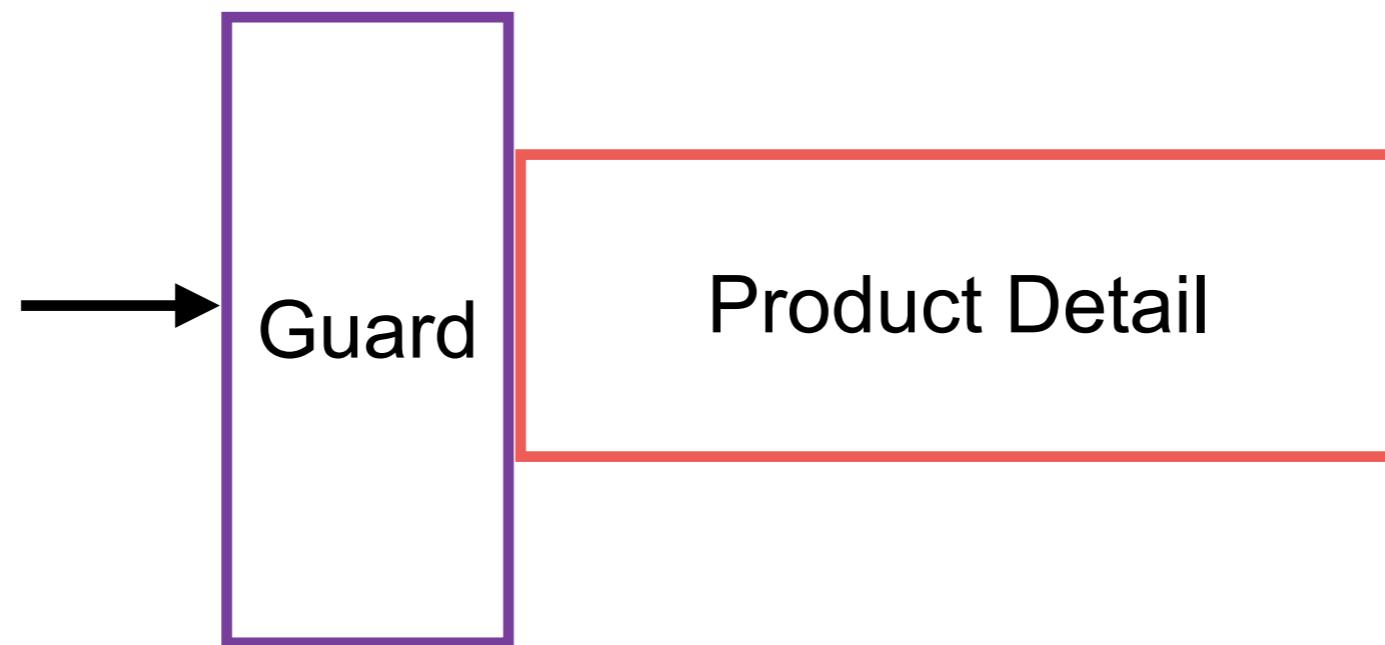
Protect route with guard

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

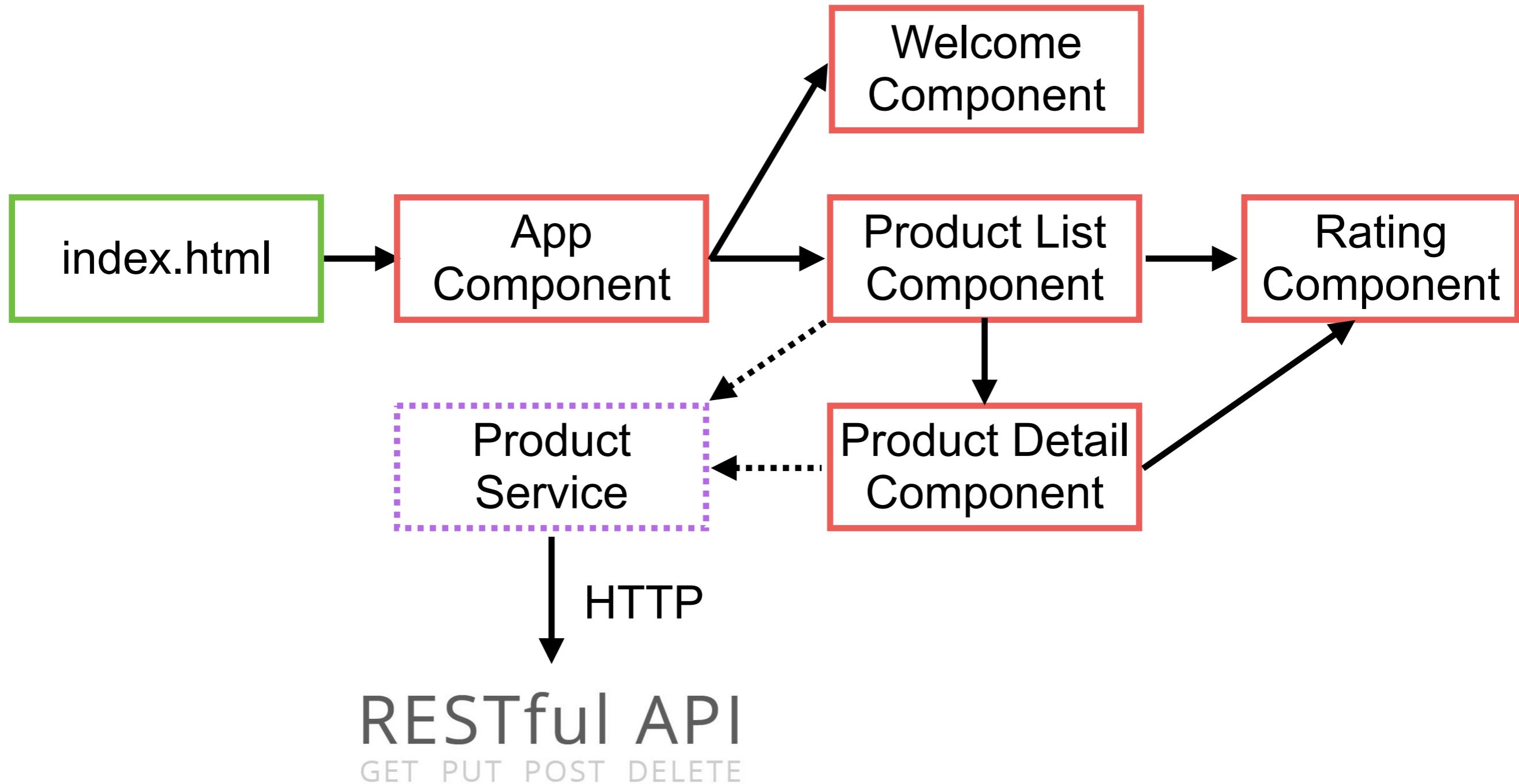


Service

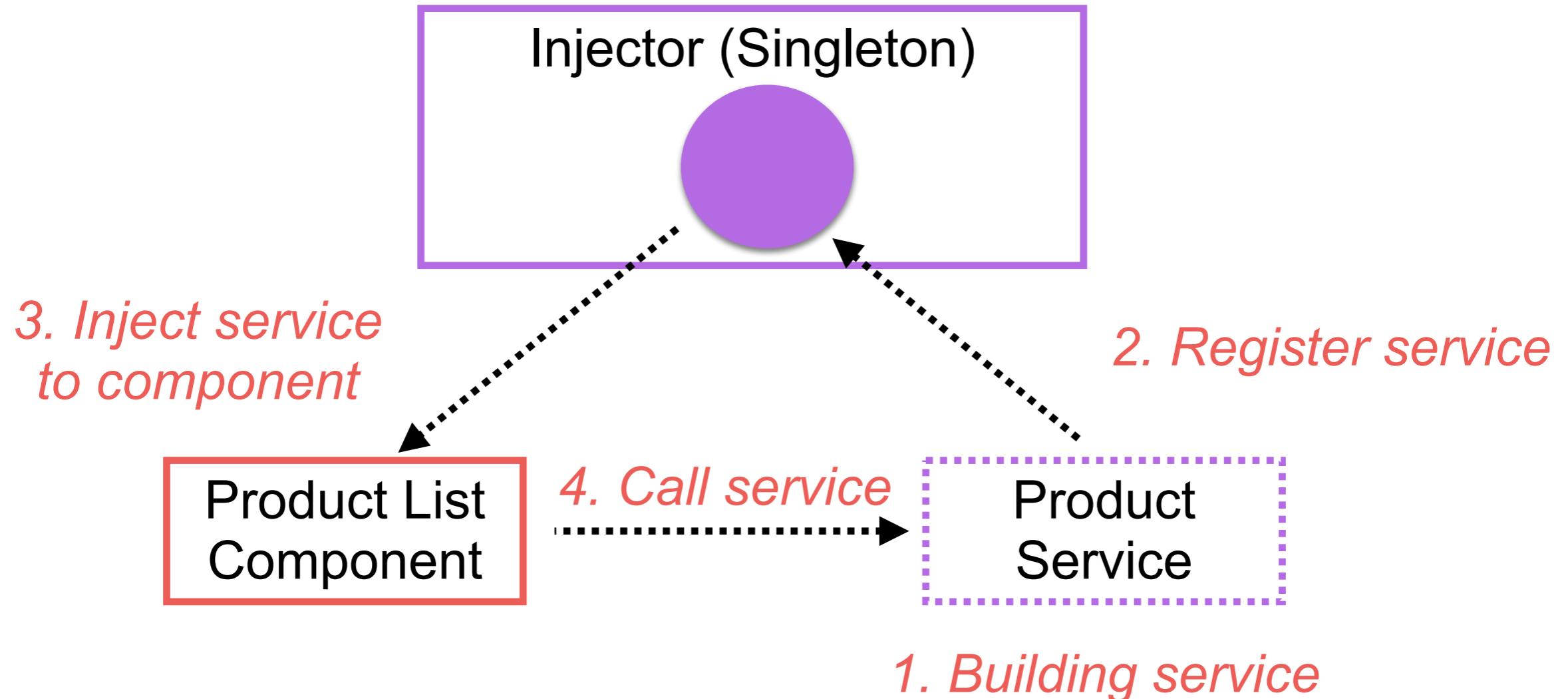
How service working ?
Building service
Register service
Inject service



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 {

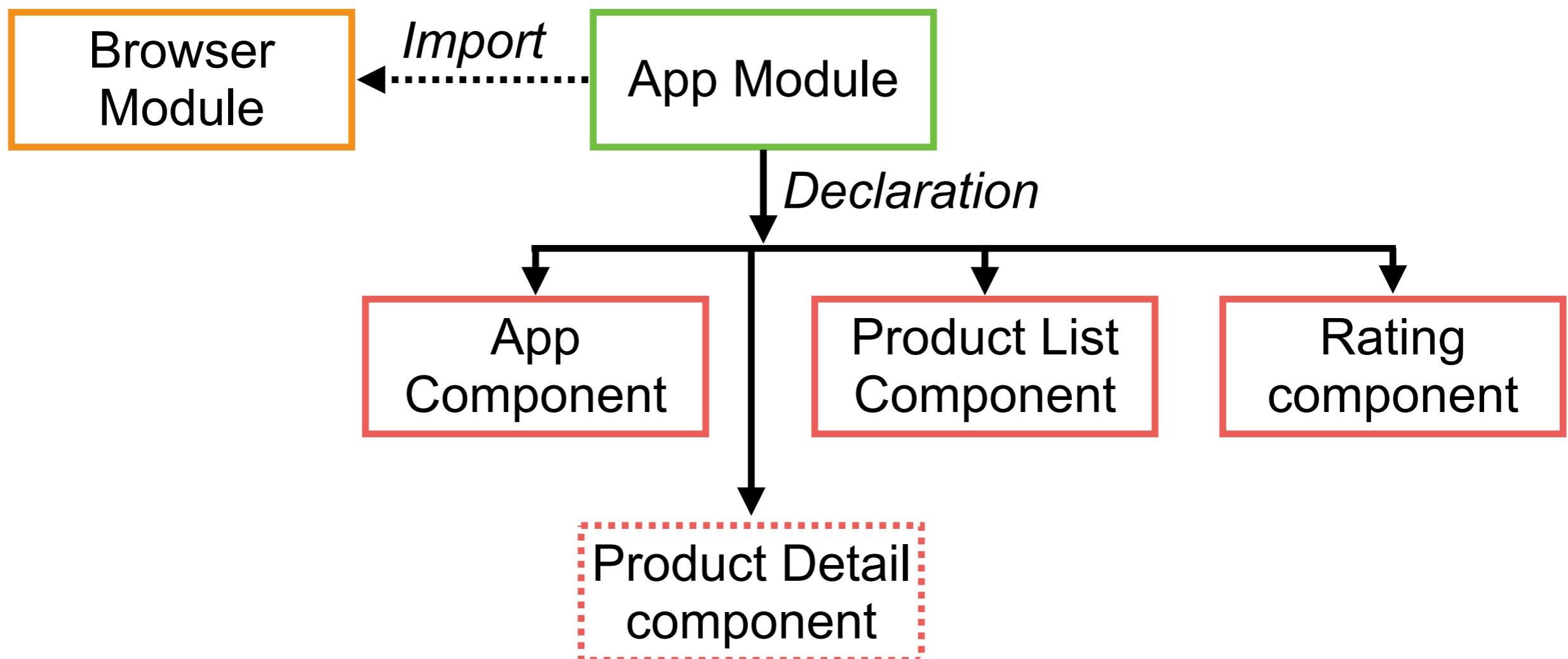
  constructor() { }

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

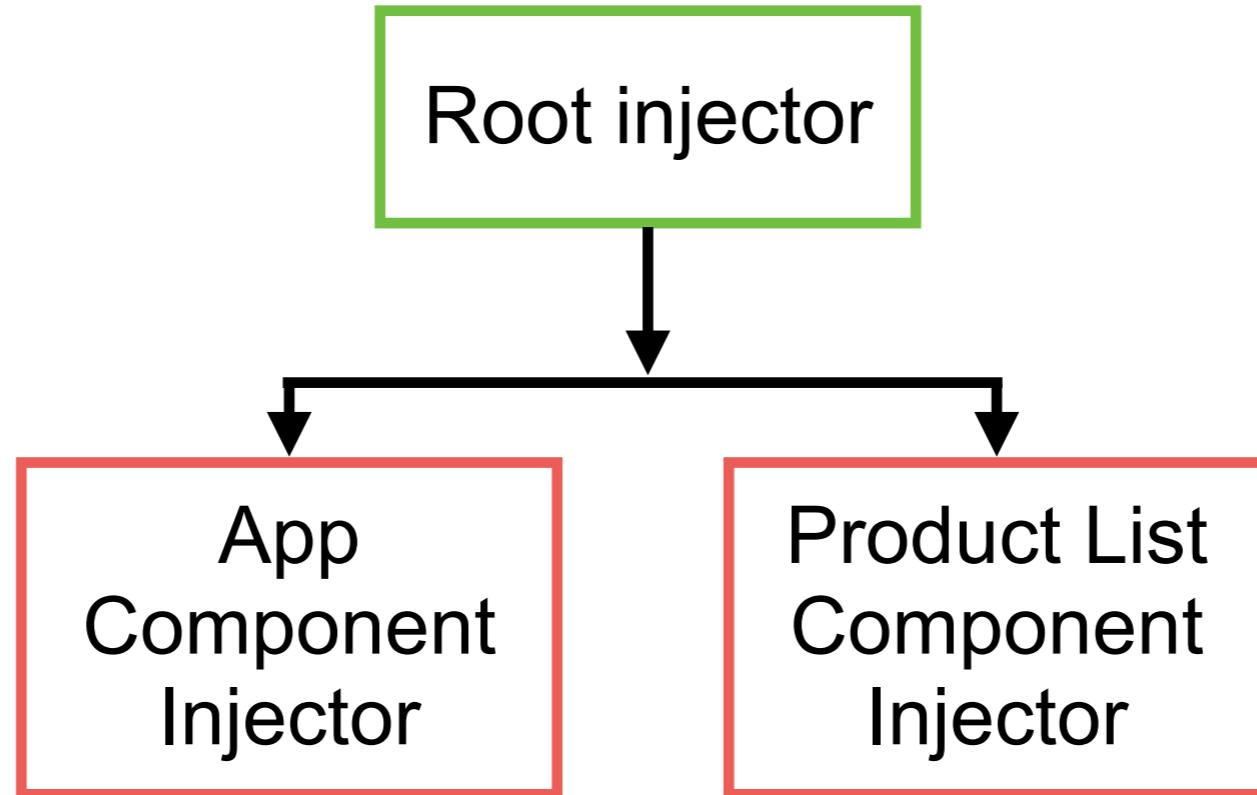


Create product detail component

\$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

