



**FPT POLYTECHNIC**



**THỰC HỌC – THỰC NGHIỆP**



**Conceive Design Implement Operate**

## **LẬP TRÌNH FRONT-END FRAMEWORK 1**

### **DATABINDING**


- ⊙ Thao tác Databinding với Property
- ⊙ Thao tác Databinding với event và custom binding
- ⊙ Định dạng hiển thị với pipes hệ thống
- ⊙ Hiểu và ứng dụng tốt ViewChild(), ContentChild() và Lifecycle Hooks



 Gắn kết thuộc tính trong Angular

 Gắn kết sự kiện

 Gắn Kết dữ liệu 2 chiều

 Gắn kết component với @Input, @Output,  
@ViewChild

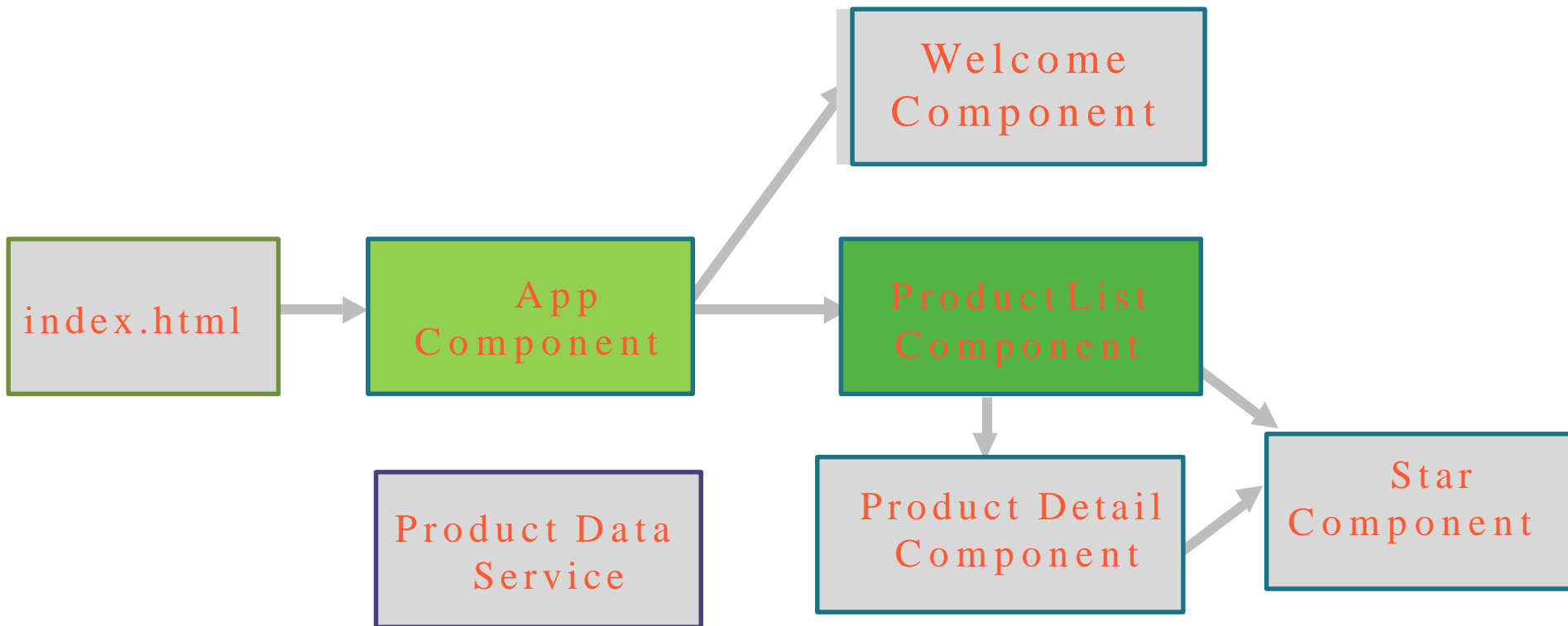
 Hiểu và cài đặt Lifecycle





# PHẦN 1: BINDING

# KIẾN TRÚC CỦA ANGULAR



`<img [src]='product.imageUrl'>`

`<img src={{ product.imageUrl }}>`

Thuộc tính của  
element

Biểu thức trên  
template

Template

Class

```
<h1>{{ pageTitle }}</h1>
```

```
<img [src]='product.imageUrl'>
```

```
<button (click)='toggleImage()'>
```

```
export class ListComponent {
  listFilter: string = 'cart';
  toggleImage(){}
}
```

()

Khai báo sự kiện

“

Biểu thức trong  
template

Template

Class

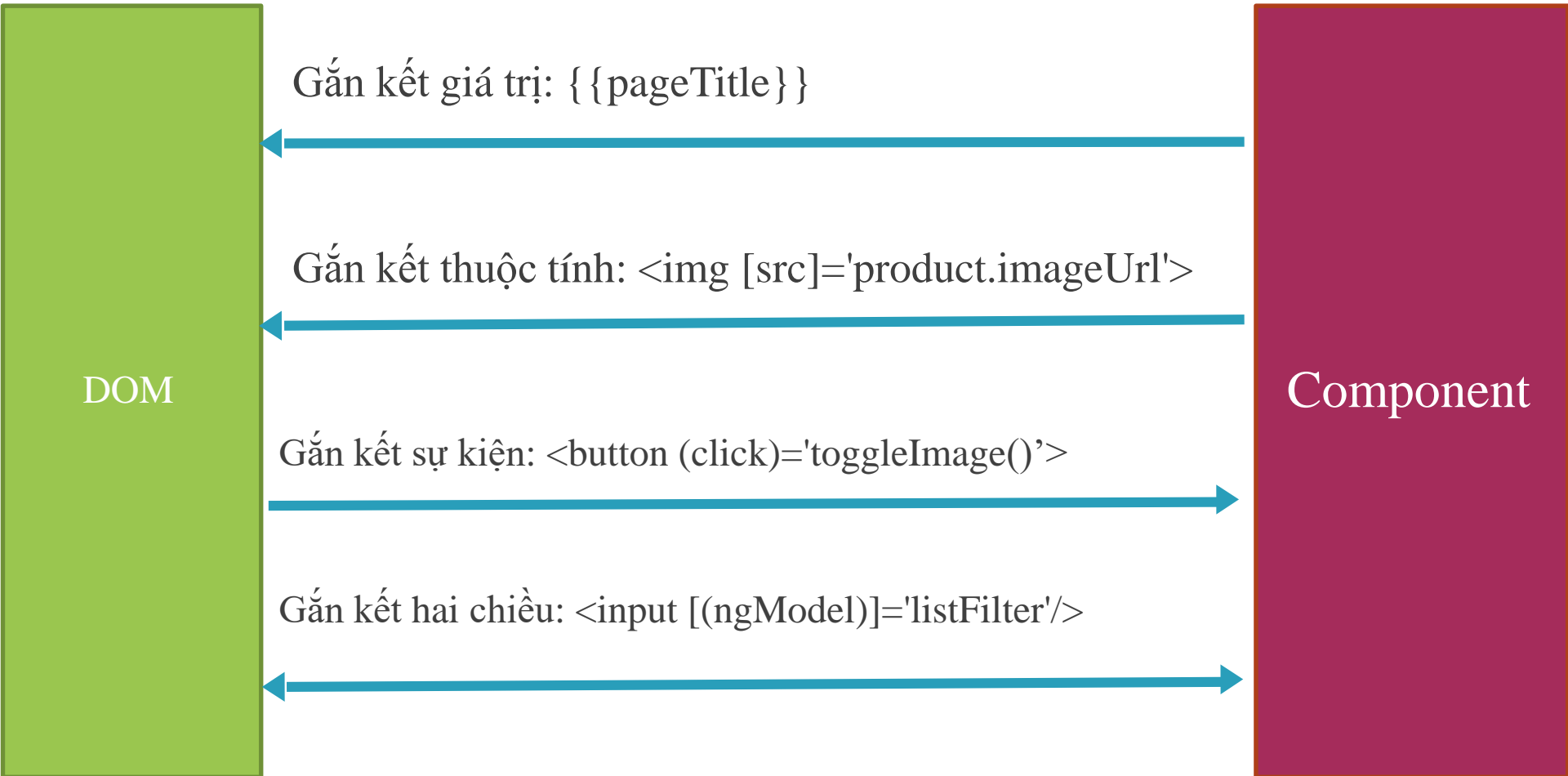
`<input [(ngModel)]='filter'>`

```
export class ProductListComponent {
  filter: string = 'Gadent';
}
```

`[( )]`

Khai báo gắn kết





## Định dạng dữ liệu với Pipes

Thuộc tính định dạng để phía sau thuộc tính hiển thị

Các định dạng sẵn có

- date
- number, decimal, percent, currency
- json, slice
- ...

Định dạng tự xây dựng

## Ví dụ

```
{{ product.productCode | lowercase }}
```

```
<img [src]='product.imageUrl'  
      [title]='product.productName | uppercase'>
```

```
{{ product.price | currency | lowercase }}
```

```
{{ product.price | currency:'USD':true:'1.2-2' }}
```



**COMPONENT PRODUCTLIST**  
**TẠO COMPONENT PRODUCTDETAIL**



## PHẦN 2: DATABINDING CHUYÊN SÂU

## Product List

Filter by:

Show Image

Product	Code	Available	Price	5 Star Rating
Leaf Rake	GDN-0011	March 19, 2016	\$19.95	3.2
Garden Cart	GDN-0023	March 18, 2016	\$32.99	4.2
Hammer	TBX-0048	May 21, 2016	\$8.9	4.8
Saw	TBX-0022	May 15, 2016	\$11.55	3.7
Video Game Controller	GMG-0042	October 15, 2015	\$35.95	4.6


## Product List

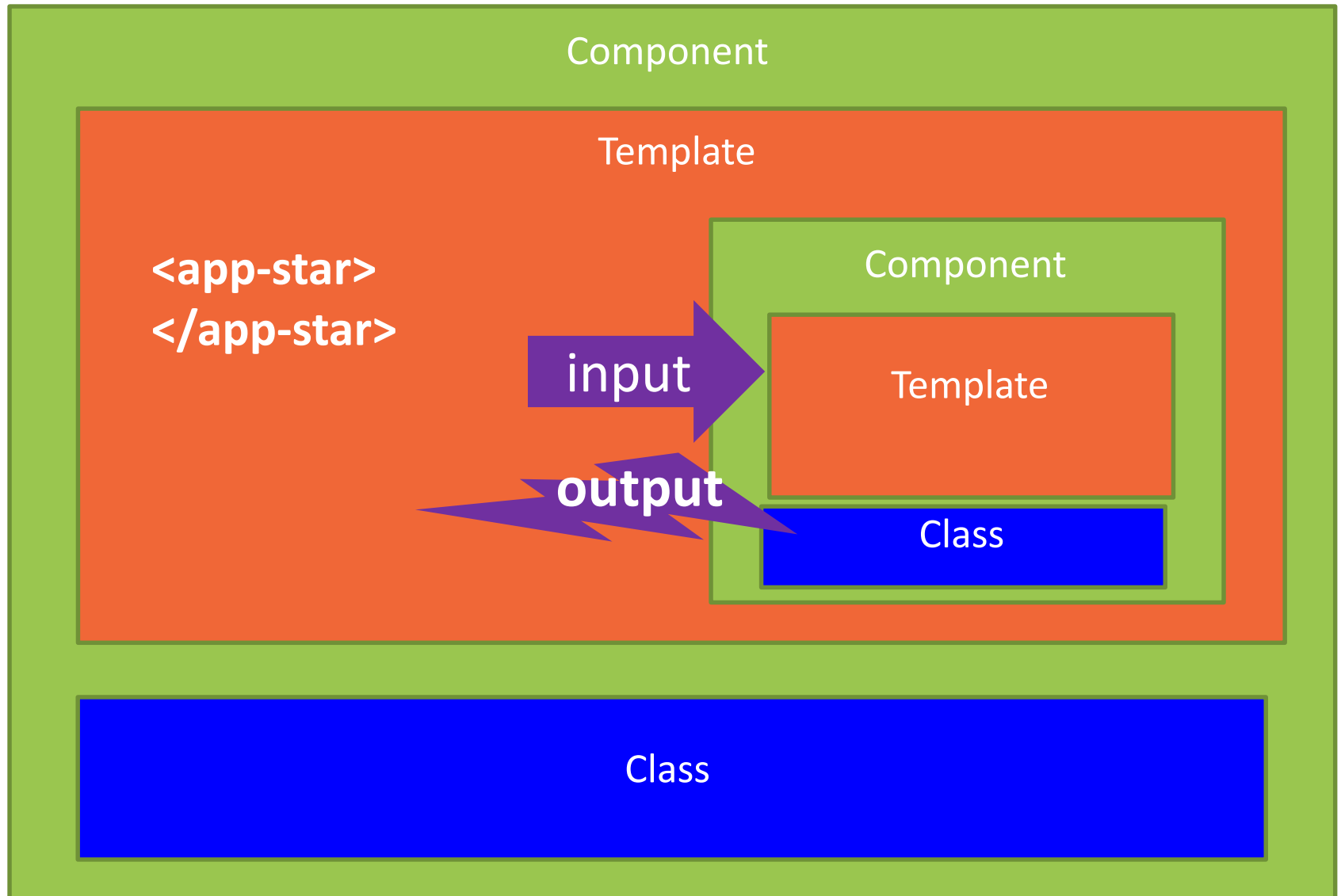
Filter by:

Show Image

Product	Code	Available	Price	5 Star Rating
Leaf Rake	GDN-0011	Mar 19, 2016	\$19.95	★★★★
Garden Cart	GDN-0023	Mar 18, 2016	\$32.99	★★★★
Hammer	TBX-0048	May 21, 2016	\$8.99	★★★★
Saw	TBX-0022	May 15, 2016	\$11.55	★★★★
Video Game Controller	GMG-0042	Oct 15, 2015	\$35.95	★★★★

Tách phần  
star thành 1  
component





# GẮN KẾT COMPONENT -TRUYỀN DỮ LIỆU

Truyền dữ liệu tới component được chèn với @Input

Productlist.component.ts

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

Productlist.component.html

```
<td>
<app-star [rating]='p.starRating'></app-star>
</td>
```

Star.component.ts

```
@Component({
  selector: 'app-star',
  templateUrl: './star.component.html',
  styleUrls: ['./star.component.css']
})
export class StarComponent implements OnInit {
  @Input() rating:number;
  constructor() {
    this.rating=0;
  }
}
```



## Xây dựng sự kiện với @Output

### Productlist.component.ts

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

### Productlist.component.html

```
@Component({
  selector: 'app-star',
  templateUrl: './star.component.html',
  styleUrls: ['./star.component.css']
})
export class StarComponent implements OnInit {
  @Input() rating:number;
  starWidth:number;
  @Output() ratingClicked: EventEmitter<string> =
    new EventEmitter<string>();
  constructor() {
    this.rating=0;
    this.starWidth=this.rating*86/5;
  }
}
```

Productlist.component.html

```
<td>
  <app-star [rating]='p.starRating'></app-star>
</td>
```

# GẮN KẾT COMPONENT - TRUYỀN DỮ LIỆU

## Xây dựng sự kiện với @Output

### Productlist.component.html

```
<td>
<app-star [rating]='p.starRating'
(ratingClicked)= 'onRatingClicked($event)'>
</app-star>
</td>
```

### Productlist.component.ts

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

### star.component.ts

```
@Component({
selector: 'app-star',
templateUrl: './star.component.html',
styleUrls: ['./star.component.css']
})
export class StarComponent implements OnInit {
@Input() rating:number;
starWidth:number;
@Output() ratingClicked: EventEmitter<string> =
new EventEmitter<string>();
onClick(): void {
this.ratingClicked.emit(`The rating ${this.rating} was
clicked!`);
}
```

### star.component.html

```
<div (click)='onClick()'> ... stars ... </div>
```

- Thường ta sử dụng ViewChild, ViewContent để truy xuất dữ liệu của thành phần con.
- Sử dụng ViewChild , chúng ta có thể nhận được tham chiếu của child-element và truy cập vào tất cả các thuộc tính và phương thức của child-element

Đoạn code bên dưới là một child-element chứa thuộc tính password và hàm cập nhật thuộc tính này

```
@Component({
  selector: 'child-component',
  template: `
    <input type="text" [(ngModel)]="passWord" />`
})
export class ChildComponent implements OnInit {
  public passWord : string;
  updatePassWord(): void {
    this. passWord = "FPTHCM"
  }
}
```

```
export class ParentComponent implements OnInit {
  @ViewChild("userInformation") childComponentReference: any;
  updateUserData() {
    // truy cập thuộc tính của Child Component
    this.childComponentReference.userName = "Updated Name";
    // Truy cập tới hàm của Child Component
    this.childComponentReference.updateUserName();
  }.
}
```

- Đoạn code trên đại diện cho ParentComponent.
- Trong component cha, chúng ta có một thành phần con được thêm vào (tham chiếu).
- Chúng ta có thể sử dụng tham chiếu này để truy cập các thuộc tính và biến của thành phần con

**ngOnChanges**

Thực thi bất cứ khi nào thuộc tính đầu vào dữ liệu thay đổi.

**ngOnInit**

Được gọi một lần khi khởi tạo component

**ngDoCheck**

Được gọi trong mỗi lần chạy phát hiện thay đổi

**ngAfterContentInit**

Thực thi sau khi Angular thêm nội dung bên ngoài vào view của component

**ngAfterContentChecked**

Thực thi sau khi Angular đã kiểm tra nội dung bên ngoài đã được đưa vào view của component.

**ngAfterViewInit**

Thực thi sau khi Angular khởi tạo các view của component và các view con

**ngAfterViewChecked**

Thực thi sau khi Angular kiểm tra các view của component và các view con

**ngOnDestroy**

Dọn dẹp ngay trước khi Angular phá hủy directive / component

```
constructor() {  
  console.log('constructor is call!!');  
}  
ngOnInit(): void {  
  console.log('onInit is call!!');  
}
```



constructor is call!!	<u>productlist.component.ts:70</u>
onInit is call!!	<u>productlist.component.ts:75</u>
Angular is running in development mode. Call enableProdMode() to enable production mode.	<u>core.js:27701</u>
[WDS] Live Reloading enabled.	<u>client:52</u>

```
import { Component, OnInit, Input, EventEmitter, Output,
  OnChanges, SimpleChange } from '@angular/core';
```

```
constructor() {
  console.log('constructor is call!!!');
}

ngOnChanges(changes: SimpleChange){
  console.log('OnChange is call!!!');
}

ngOnInit(): void {
  console.log('onInit is call!!!');
}
```



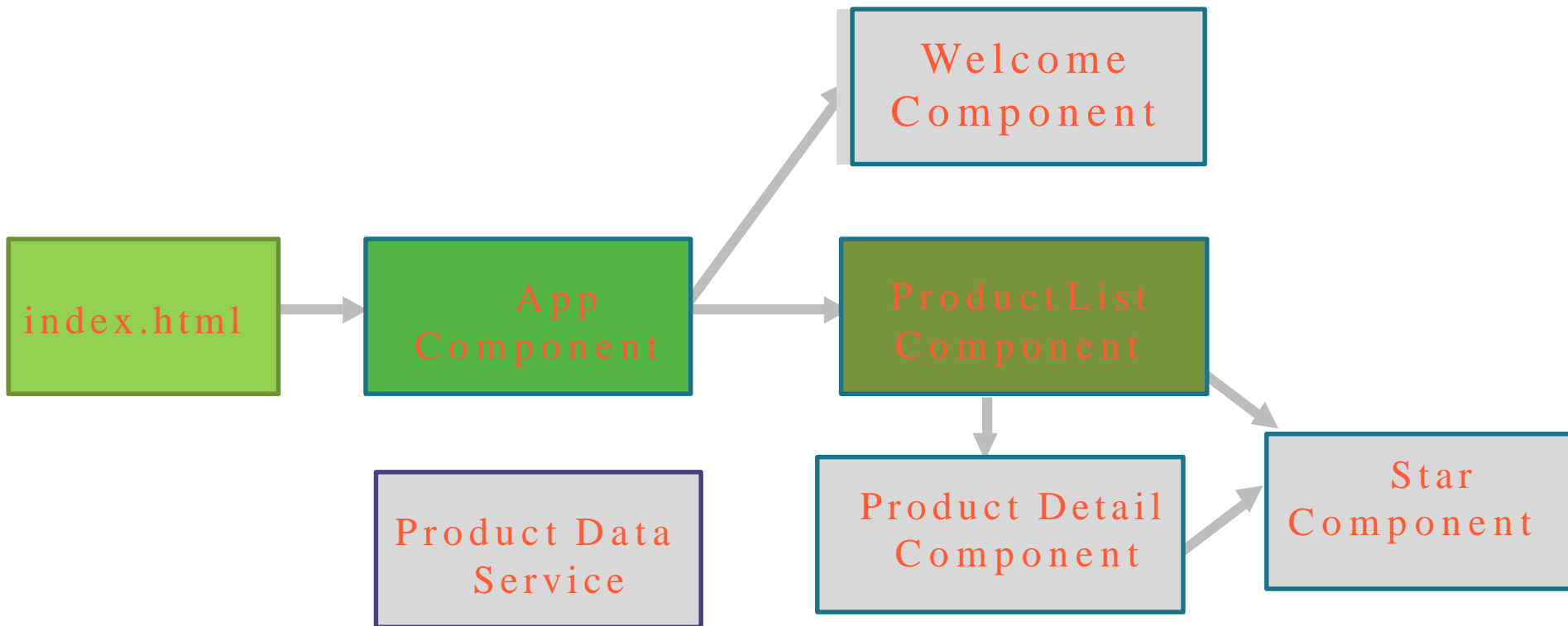
constructor is call!!	<u>productlist.component.ts:70</u>
onInit is call!!	<u>productlist.component.ts:75</u>
Angular is running in development mode. Call enableProdMode() to enable production mode.	<u>core.js:27701</u>
[WDS] Live Reloading enabled.	<u>client:52</u>



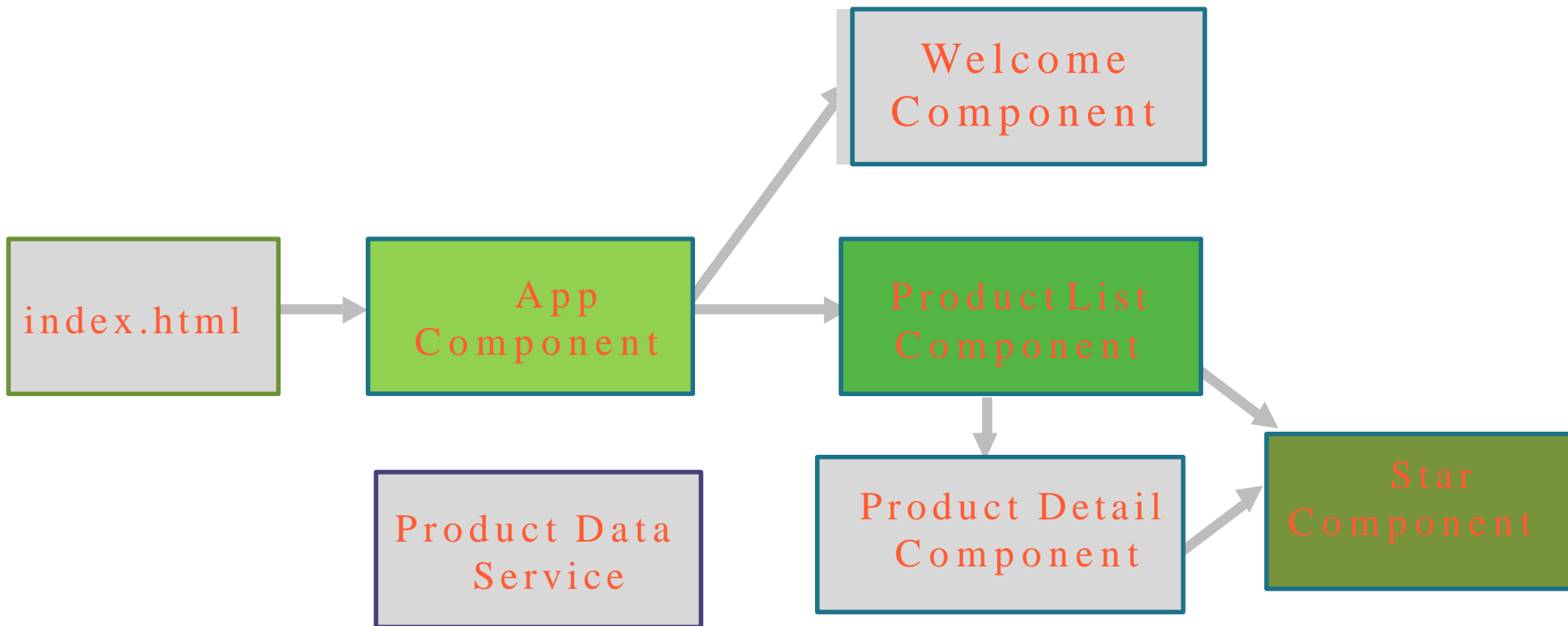


# **BINDING COMPONENT LIFECYCLE**

# KIẾN TRÚC CỦA ANGULAR



# KIẾN TRÚC CỦA ANGULAR



- ☑ Gắn kết thuộc tính trong Angular
- ☑ Gắn kết sự kiện
- ☑ Gắn Kết dữ liệu 2 chiều
- ☑ Gắn kết component với @Input, @Output, @ViewChild
- ☑ Hiểu và cài đặt Lifecycle



thank  
you!