Building Nested Components

Deborah Kurata
Consultant | Speaker | Author | MVP | GDE

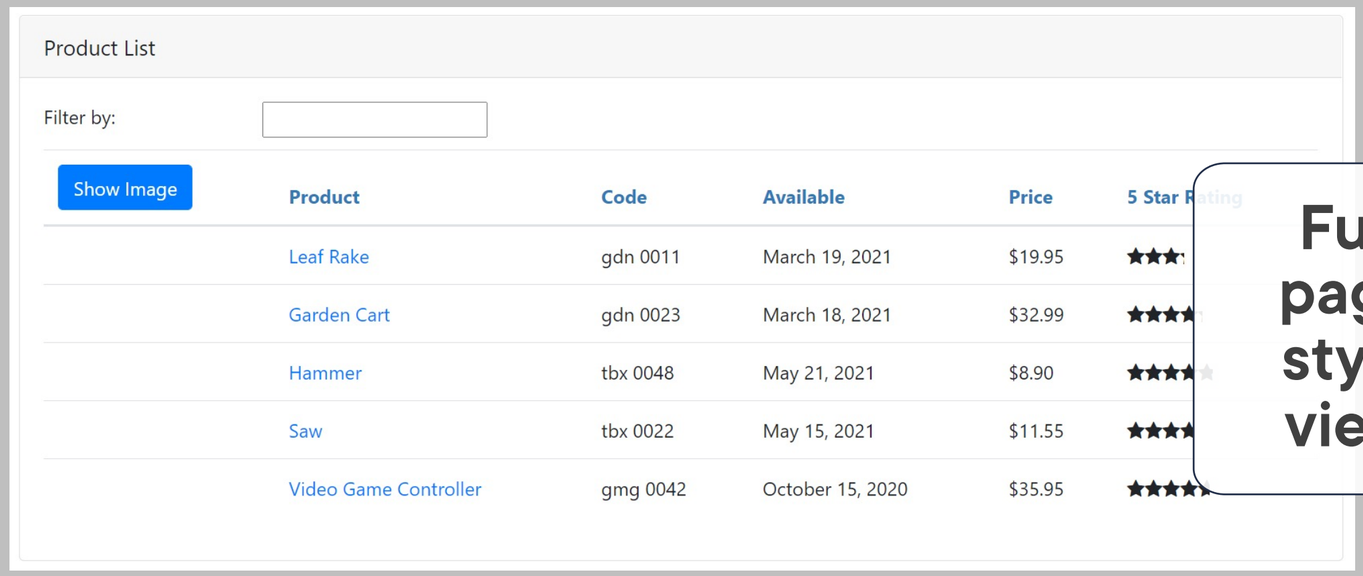
@deborahkurata

Using a Component

As a Directive

App Component OR Nested Component

As a Routing target



Full page style view

```
<body>
  <pm-root></pm-root>
</body>
```

What Makes a Component Nest-able?

Its template manages a fragment of a larger view

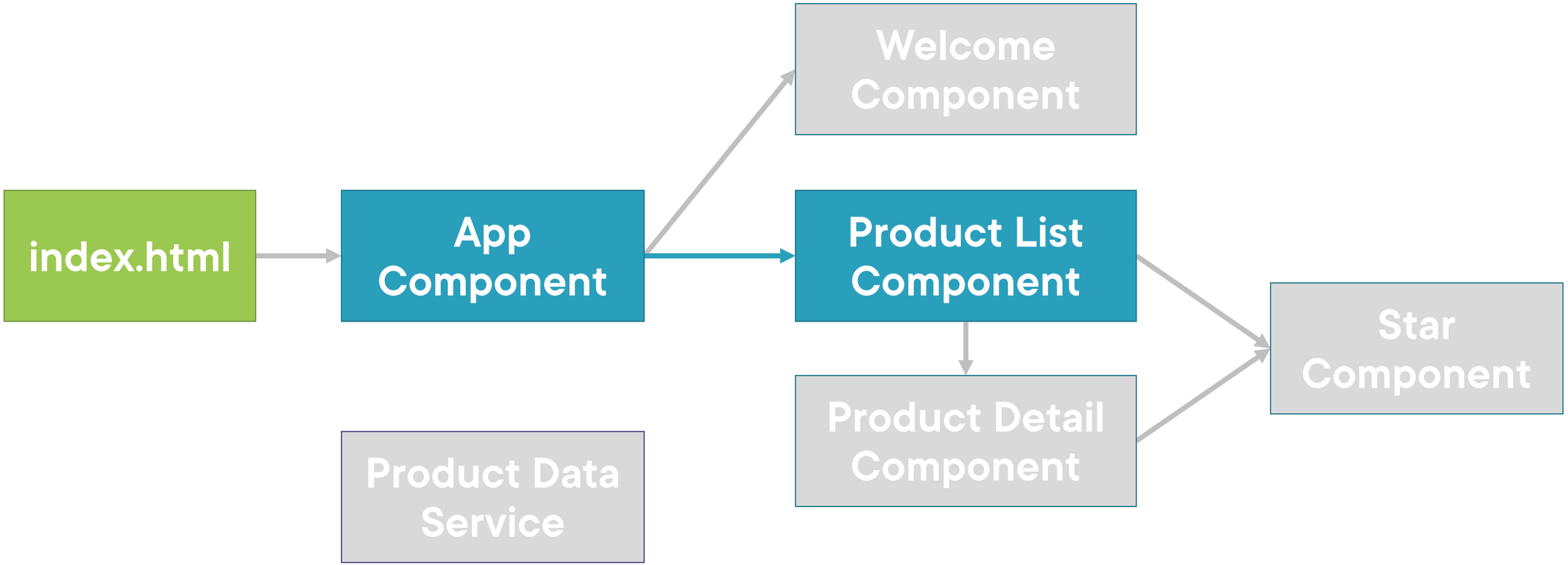
It has a selector

It optionally communicates with its container

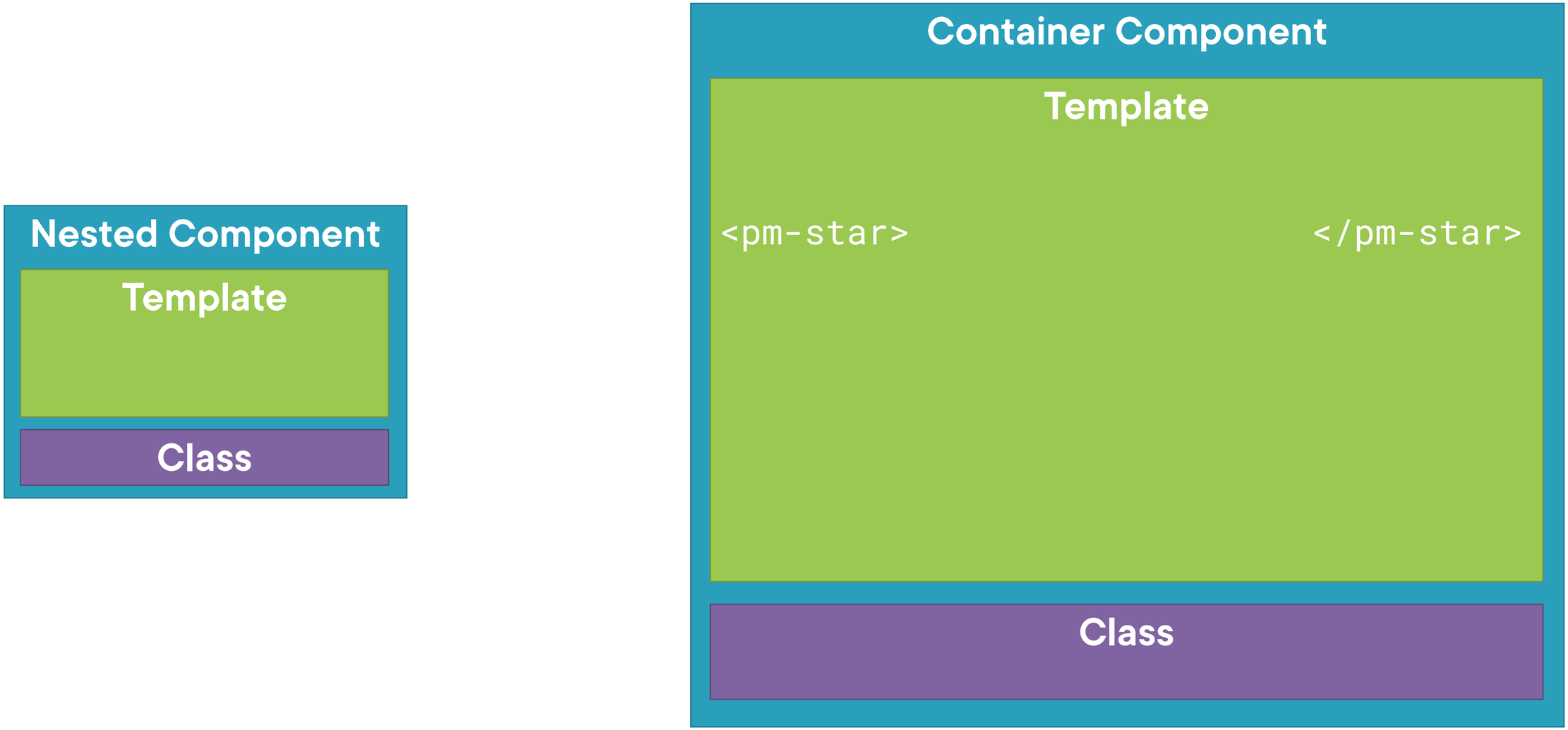
Module Overview

- Building a nested component
- Using a nested component
- Passing data to a nested component using @Input
- Raising an event from a nested component using @Output

Application Architecture



Building a Nested Component



Product List View

Product List

Filter by:

Show Image

Product	Code	Available	Price	5 Star Rating
Leaf Rake	gdn 0011	March 19, 2021	$19.95	3.2
Garden Cart	gdn 0023	March 18, 2021	$32.99	4.2
Hammer	tbx 0048	May 21, 2021	$8.90	4.8
Saw	tbx 0022	May 15, 2021	$11.55	3.7
Video Game Controller	gmg 0042	October 15, 2020	$35.95	4.6

Product List View

Product List

Filter by:

Show Image

Product	Code	Available	Price	5 Star Rating
Leaf Rake	gdn 0011	March 19, 2021	$19.95	
Garden Cart	gdn 0023	March 18, 2021	$32.99	
Hammer	tbx 0048	May 21, 2021	$8.90	
Saw	tbx 0022	May 15, 2021	$11.55	
Video Game Controller	gmg 0042	October 15, 2020	$35.95	

Using a Nested Component as a Directive

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

product-list.component.html

```
<td>
   {{ product.starRating }}
</td>
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
  rating: number;
  cropWidth: number;
}
```

Using a Nested Component as a Directive

product-list.component.ts

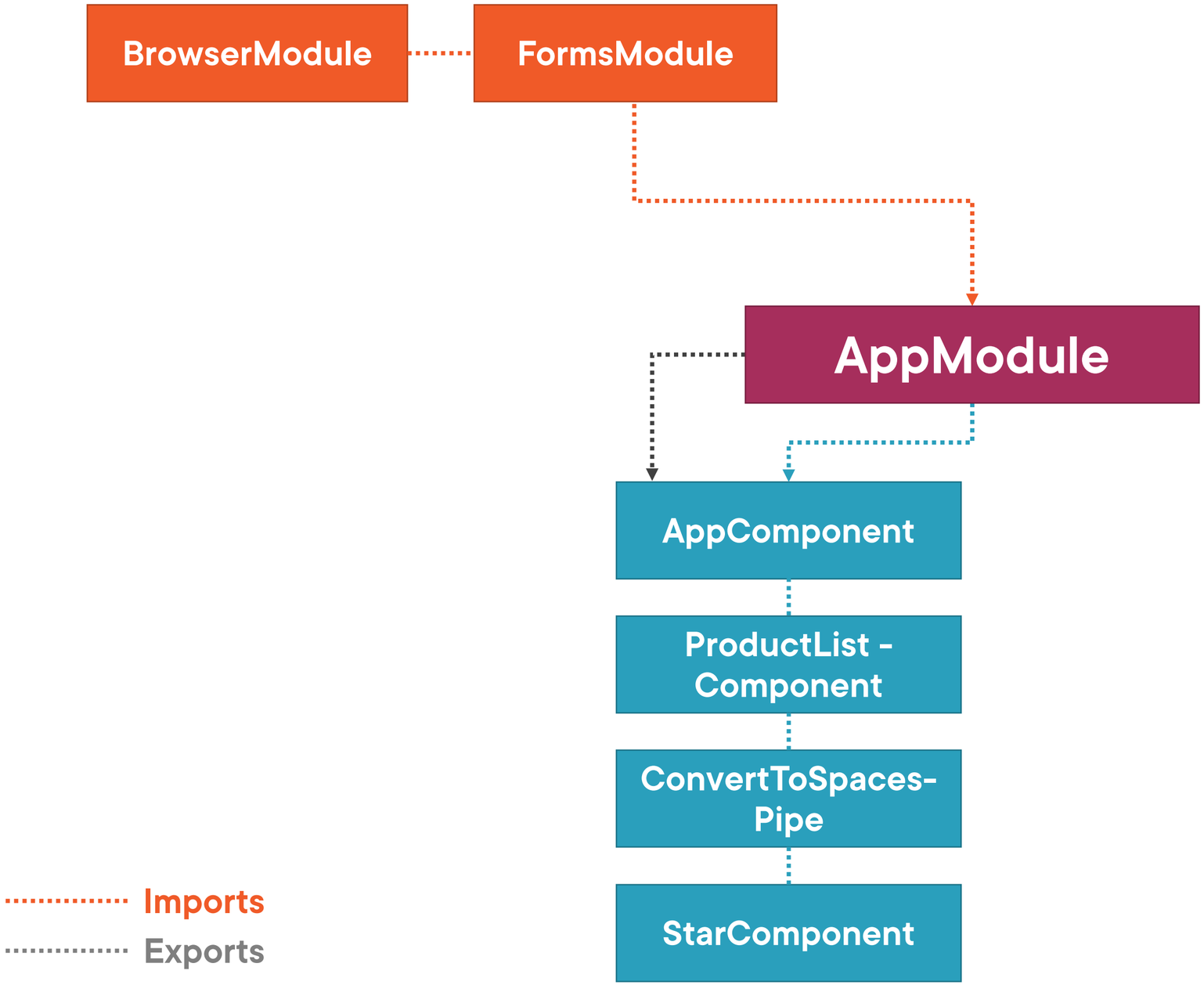
```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

product-list.component.html

```
<td>
   <pm-star></pm-star>
</td>
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
  rating: number;
  cropWidth: number;
}
```


BrowserModule
FormsModule
AppModule
AppComponent
ProductList - Component
ConvertToSpaces-Pipe
StarComponent
Imports
Exports
Declarations
Bootstrap

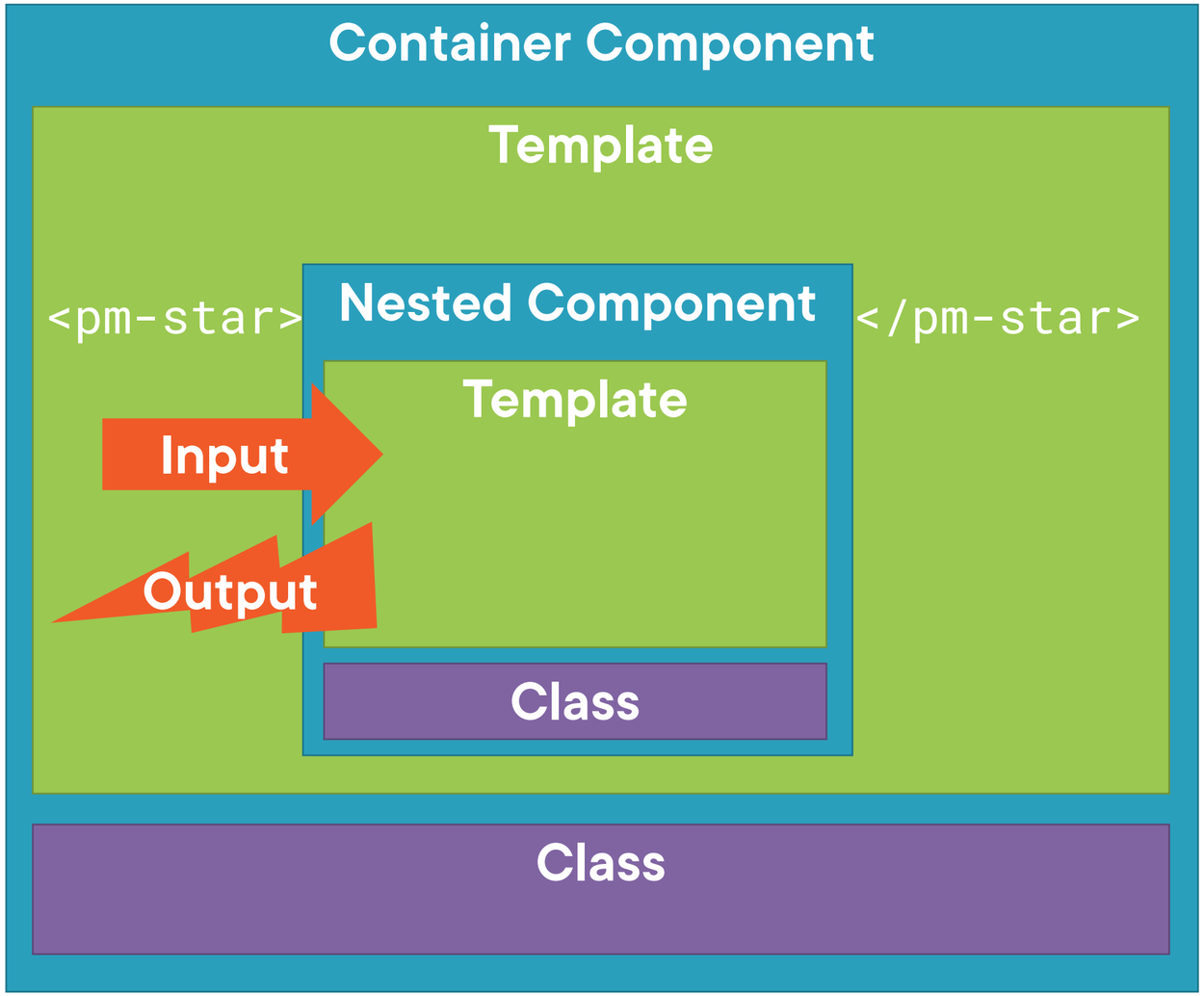
Telling Angular About Our Component

app.module.ts

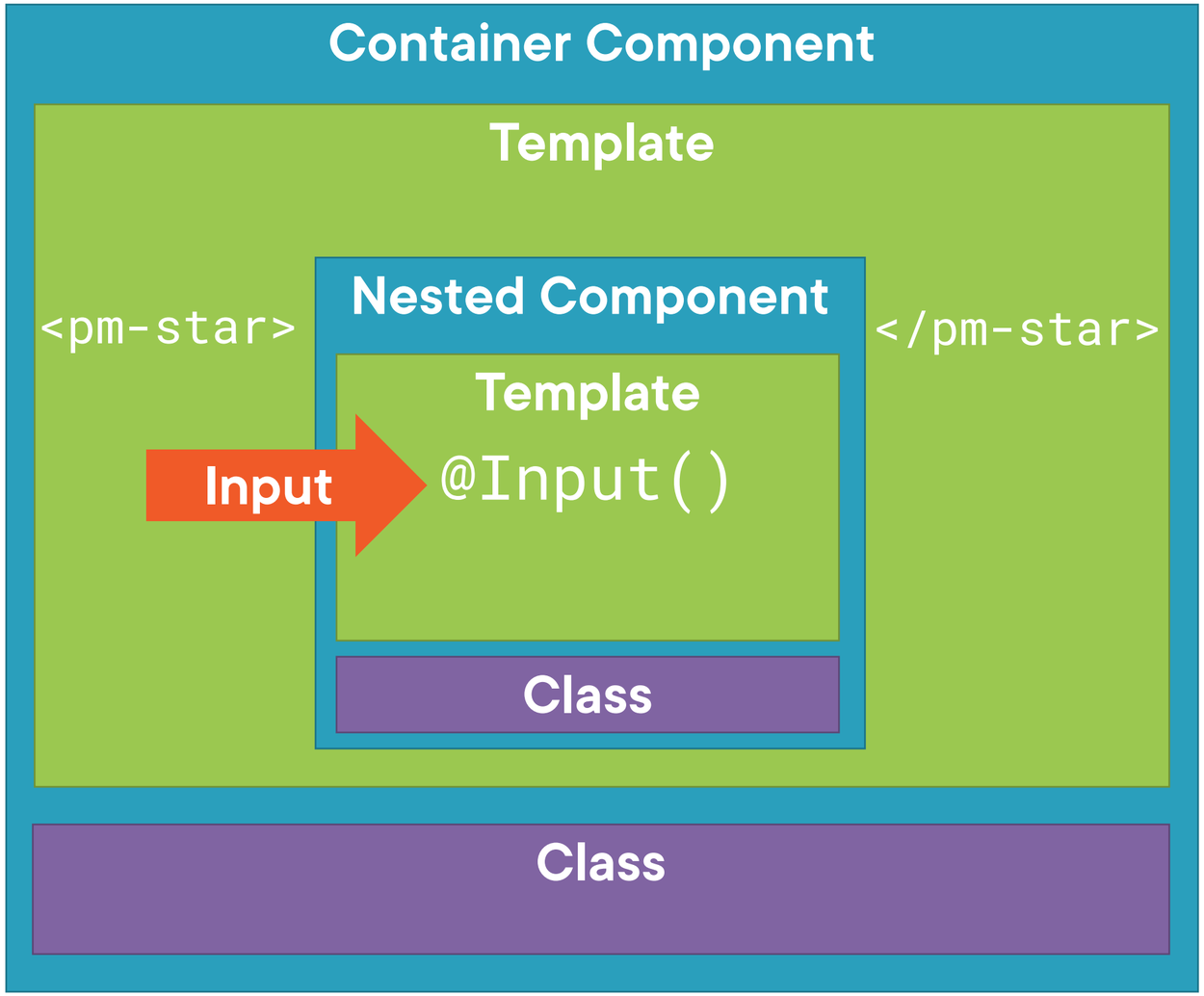
```
...
import { StarComponent } from './shared/star.component';

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

Building a Nested Component



Passing Data to a Nested Component (`@Input`)



Passing Data to a Nested Component (@Input)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

product-list.component.html

```
<td>
   <pm-star></pm-star>
</td>
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  cropWidth: number;
}
```

Passing Data to a Nested Component (@Input)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

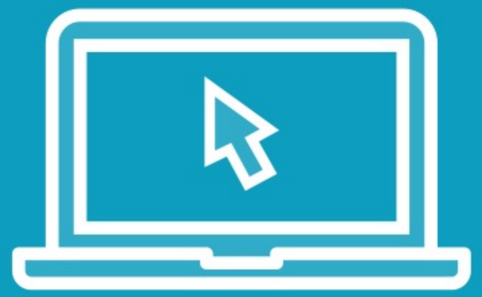
product-list.component.html

```
<td>
 <pm-star [rating]='product.starRating'>
 </pm-star>
</td>
```

star.component.ts

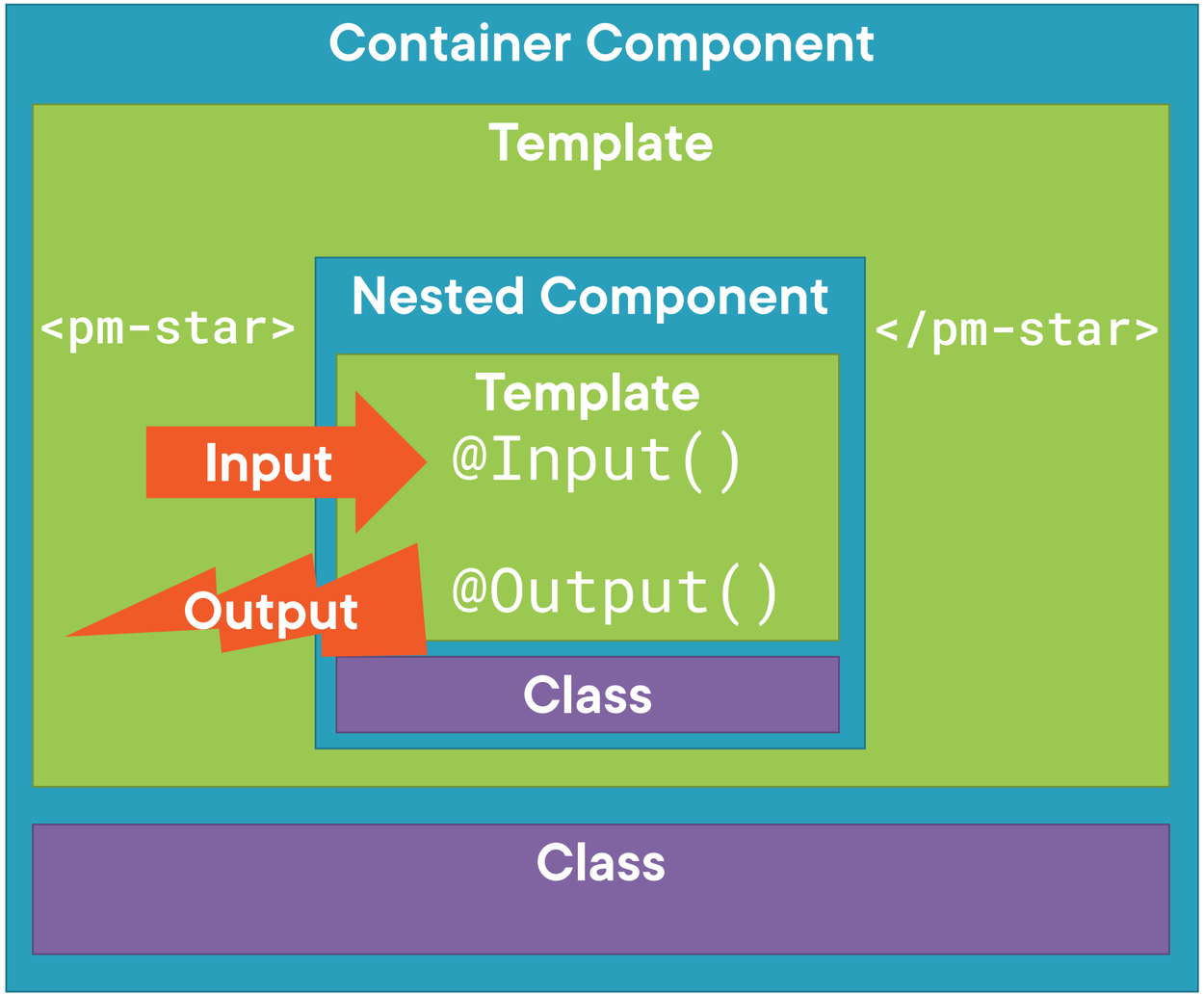
```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  cropWidth: number;
}
```

Demo



Handling events

Emitting an Event (`@Output`)



Emitting an Event (@Output)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
 @Input() rating: number;
 cropWidth: number;
 @Output() notify: EventEmitter<string> =
                   new EventEmitter<string>();
}
```

product-list.component.html

```
<td>
 <pm-star [rating]='product.starRating'>
 </pm-star>
</td>
```

Emitting an Event (@Output)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
 @Input() rating: number;
 cropWidth: number;
 @Output() notify: EventEmitter<string> =
                   new EventEmitter<string>();
 onClick() {
   this.notify.emit('clicked!');
 }
}
```

product-list.component.html

```
<td>
 <pm-star [rating]='product.starRating'>
 </pm-star>
</td>
```

star.component.html

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

Emitting an Event (@Output)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent { }
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
 @Input() rating: number;
 cropWidth: number;
 @Output() notify: EventEmitter<string> =
                   new EventEmitter<string>();
 onClick() {
   this.notify.emit('clicked!');
 }
}
```

product-list.component.html

```
<td>
 <pm-star [rating]='product.starRating'
          (notify)='onNotify($event)'>
 </pm-star>
</td>
```

star.component.html

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

Emitting an Event (@Output)

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateURL: './product-list.component.html'
})
export class ProductListComponent {
  onNotify(message: string): void { }
}
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateURL: './star.component.html'
})
export class StarComponent {
 @Input() rating: number;
 cropWidth: number;
 @Output() notify: EventEmitter<string> =
                   new EventEmitter<string>();
 onClick() {
   this.notify.emit('clicked!');
 }
}
```

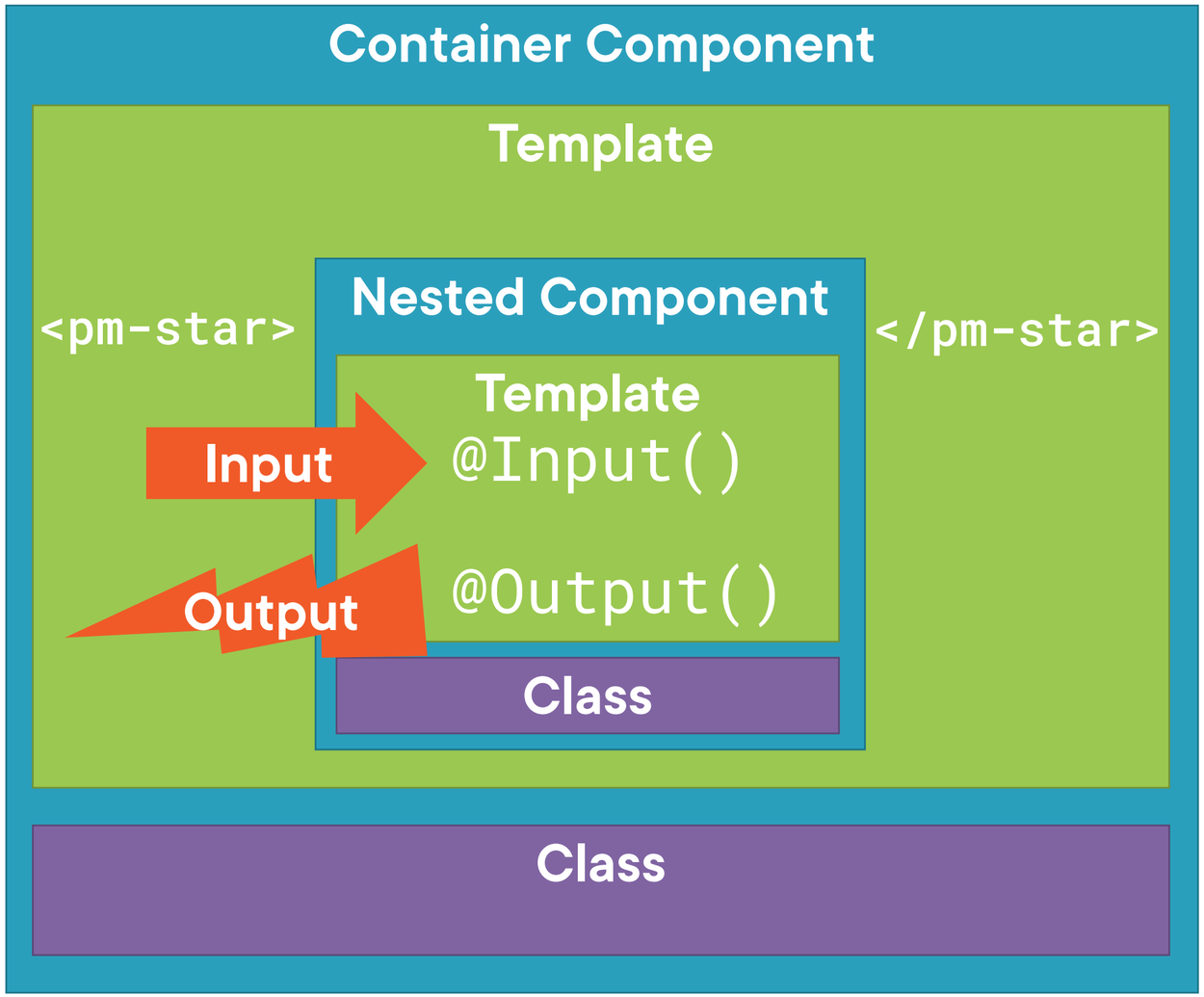
product-list.component.html

```
<td>
 <pm-star [rating]='product.starRating'
          (notify)='onNotify($event)'>
 </pm-star>
</td>
```

star.component.html

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

Nest-able Component's Public API



Nesting Checklist: Input Properties

Input decorator

Attach to a property of any type

Prefix with @; Suffix with ()

```
export class StarComponent {
 @Input() rating: number;
}
```

Nesting Checklist: Output Properties

`Output` decorator

Attached to a property declared as an EventEmitter

Use the generic argument to define the event data type

Use the new keyword to create an instance of the EventEmitter

Prefix with @; Suffix with ()

```
export class StarComponent {
 @Output() notify: EventEmitter<string> =
                                new EventEmitter<string>();
}
```

Nesting Checklist: Container Component

Use the directive
- Directive name -> nested component's selector

Use property binding to pass data to the nested component

Use event binding to respond to events from the nested component
- Use $event to access the event data passed from the nested component

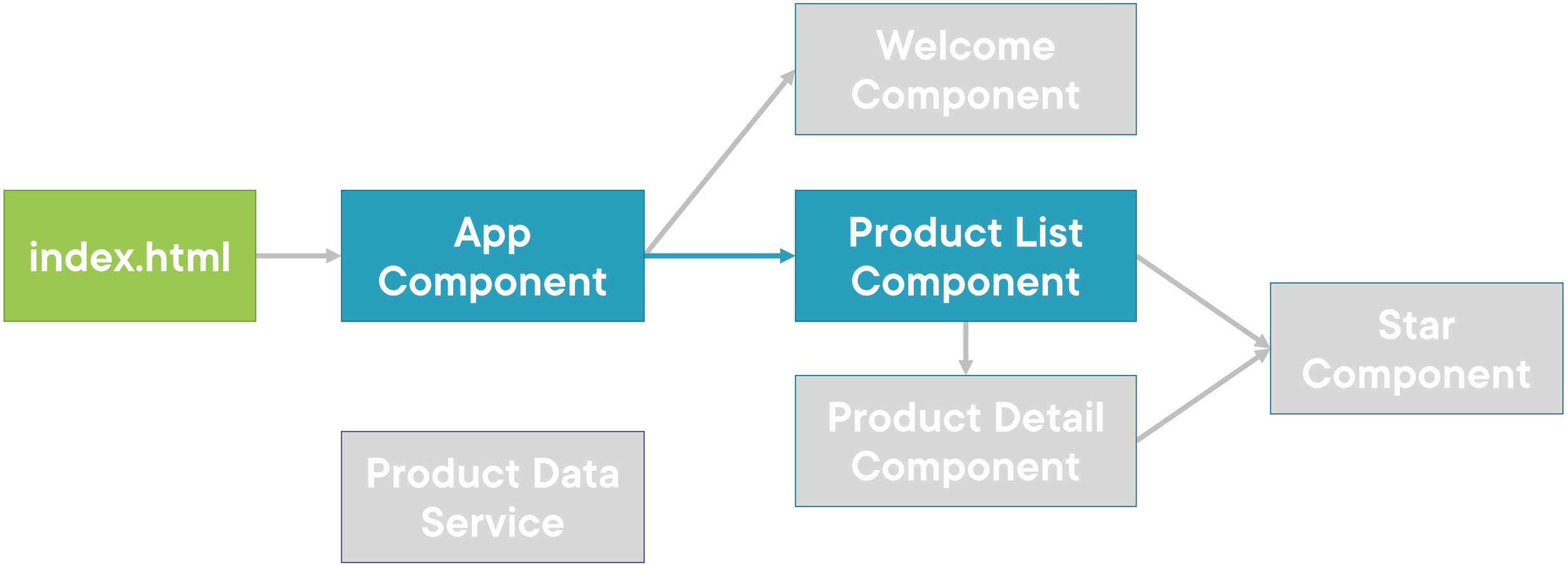
```
<pm-star [rating]='product.starRating'
         (notify)='onNotify($event)'>
</pm-star>
```

Learning More

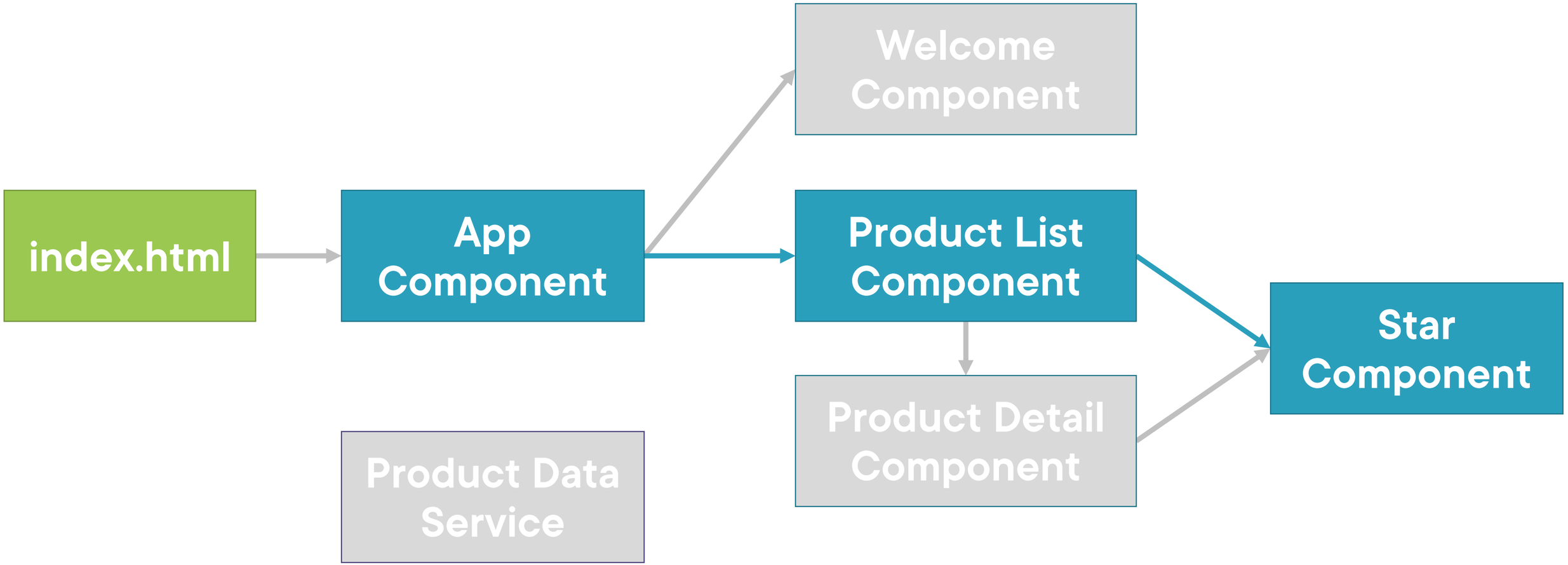
Pluralsight Course

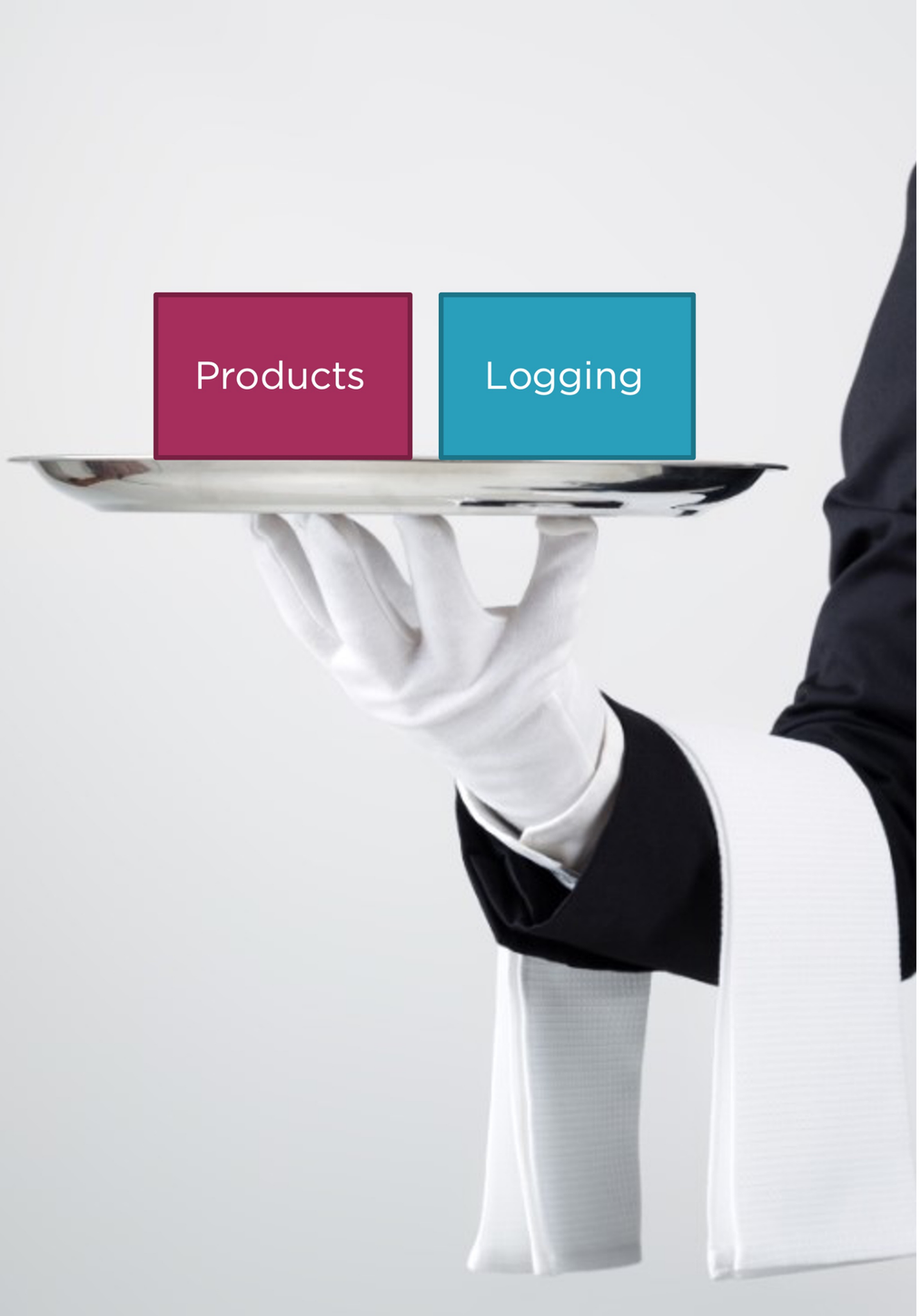
- **"Angular Component Communication"**

Application Architecture



Application Architecture





Coming up next ...

Services and Dependency Injection