



What Makes a Component Nest-able?



Its template only manages a fragment of a larger view

It has a selector

It optionally communicates with its container

Module Overview



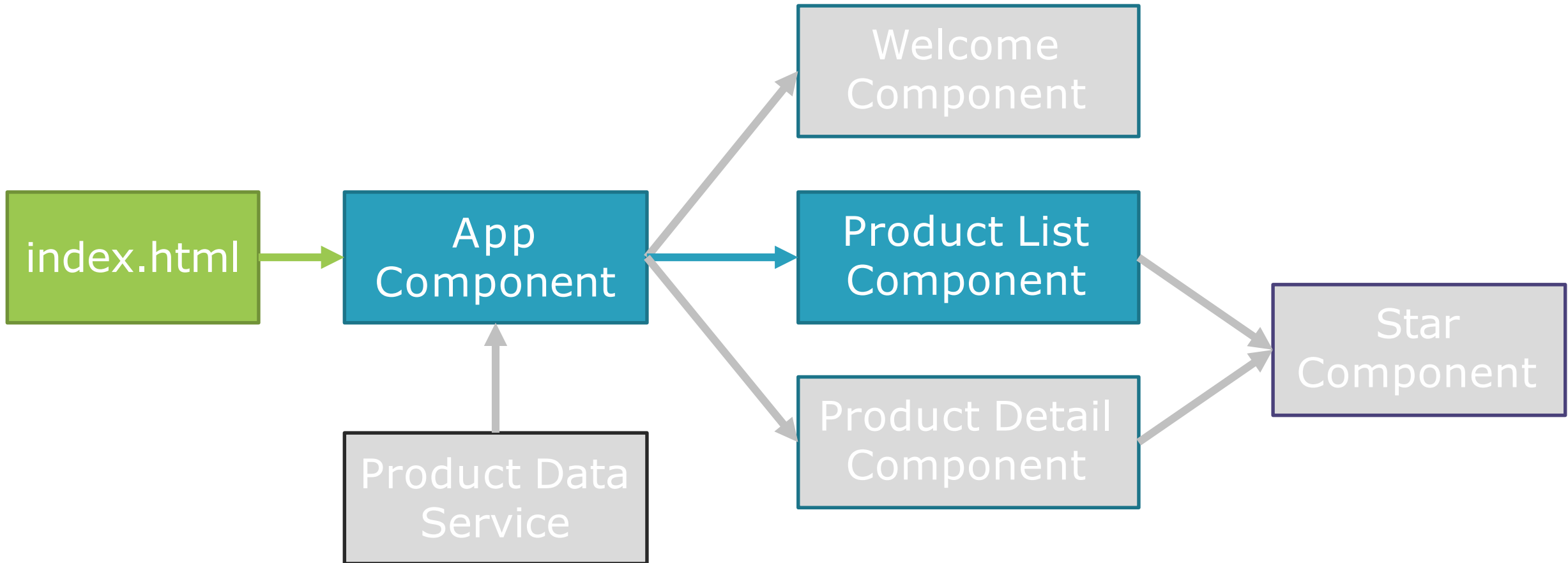
Building a Nested Component

Using a Nested Component

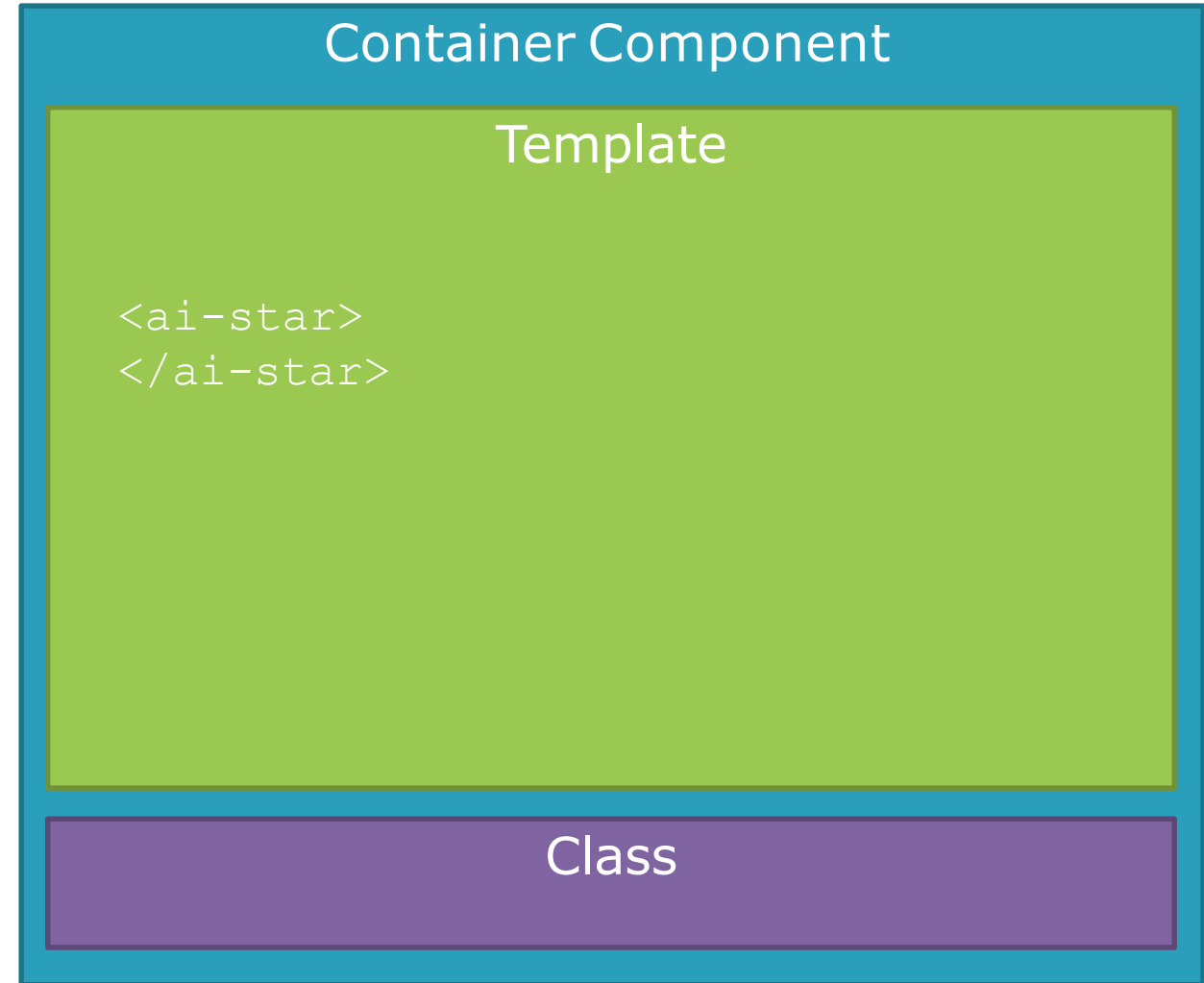
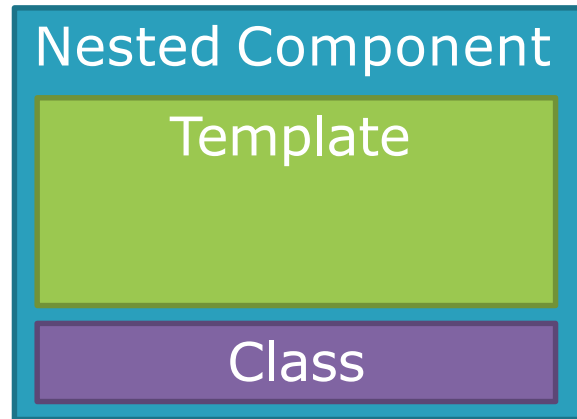
Passing Data to a Nested Component
Using @Input

Raising an Event from a Nested
Component Using @Output

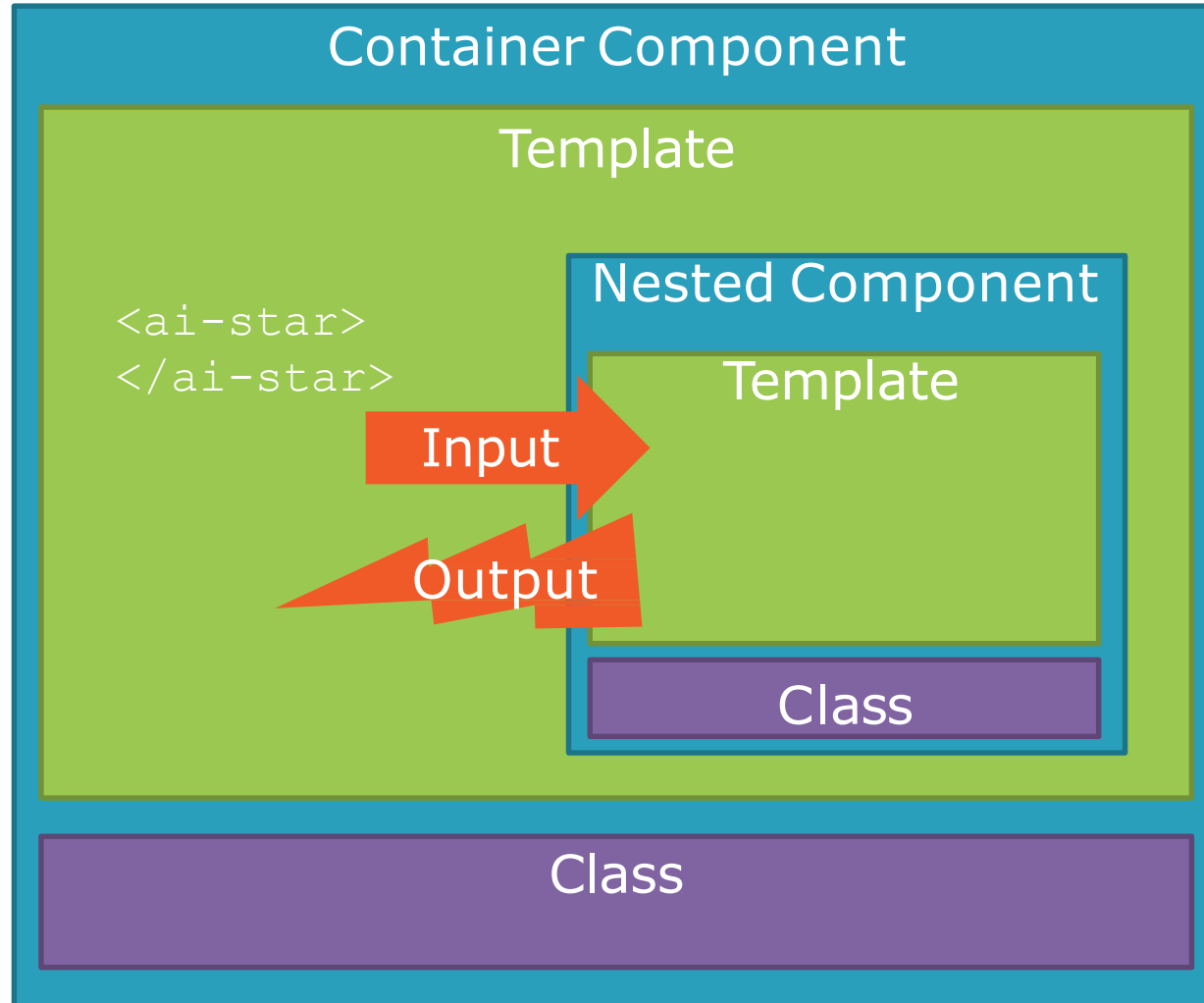
Application Architecture



Building a Nested Component



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, 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 View

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

★★★★★

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 | number }}
</td>
```

star.component.ts

```
@Component ({
  selector: 'ai-star',
  templateUrl: 'star.component.html'
})
export class StarComponent {
  rating: number;
  starWidth: number;
}
```

Using a Nested Component as a Directive

product-list.component.ts

```
@Component ({
  selector: 'pm-products',
  templateUrl: 'product-list.component.html',
  directives: [StarComponent]
})
export class ProductListComponent { }
```

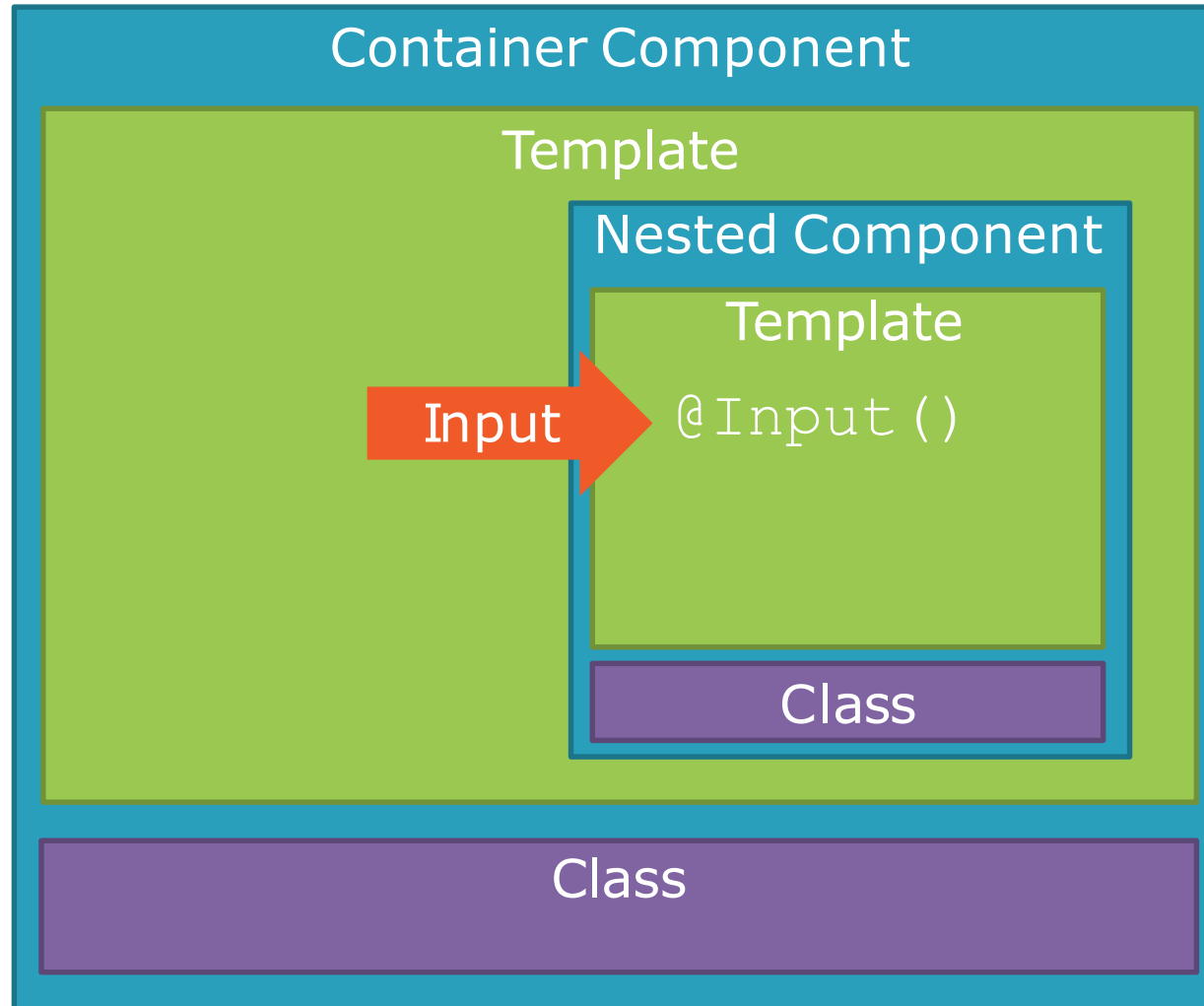
product-list.component.html

```
<td>
  <ai-star></ai-star>
</td>
```

star.component.ts

```
@Component ({
  selector: 'ai-star',
  templateUrl: 'star.component.html'
})
export class StarComponent {
  rating: number;
  starWidth: number;
}
```

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',  
  directives: [StarComponent]  
})  
export class ProductListComponent { }
```

product-list.component.html

```
<td>  
  <ai-star></ai-star>  
</td>
```

star.component.ts

```
@Component ({  
  selector: 'ai-star',  
  templateUrl: 'star.component.html'  
})  
export class StarComponent {  
  @Input rating: number;  
  starWidth: number;  
}
```

Passing Data to a Nested Component (@Input)

product-list.component.ts

```
@Component ({
  selector: 'pm-products',
  templateUrl: 'product-list.component.html',
  directives: [StarComponent]
})
export class ProductListComponent { }
```

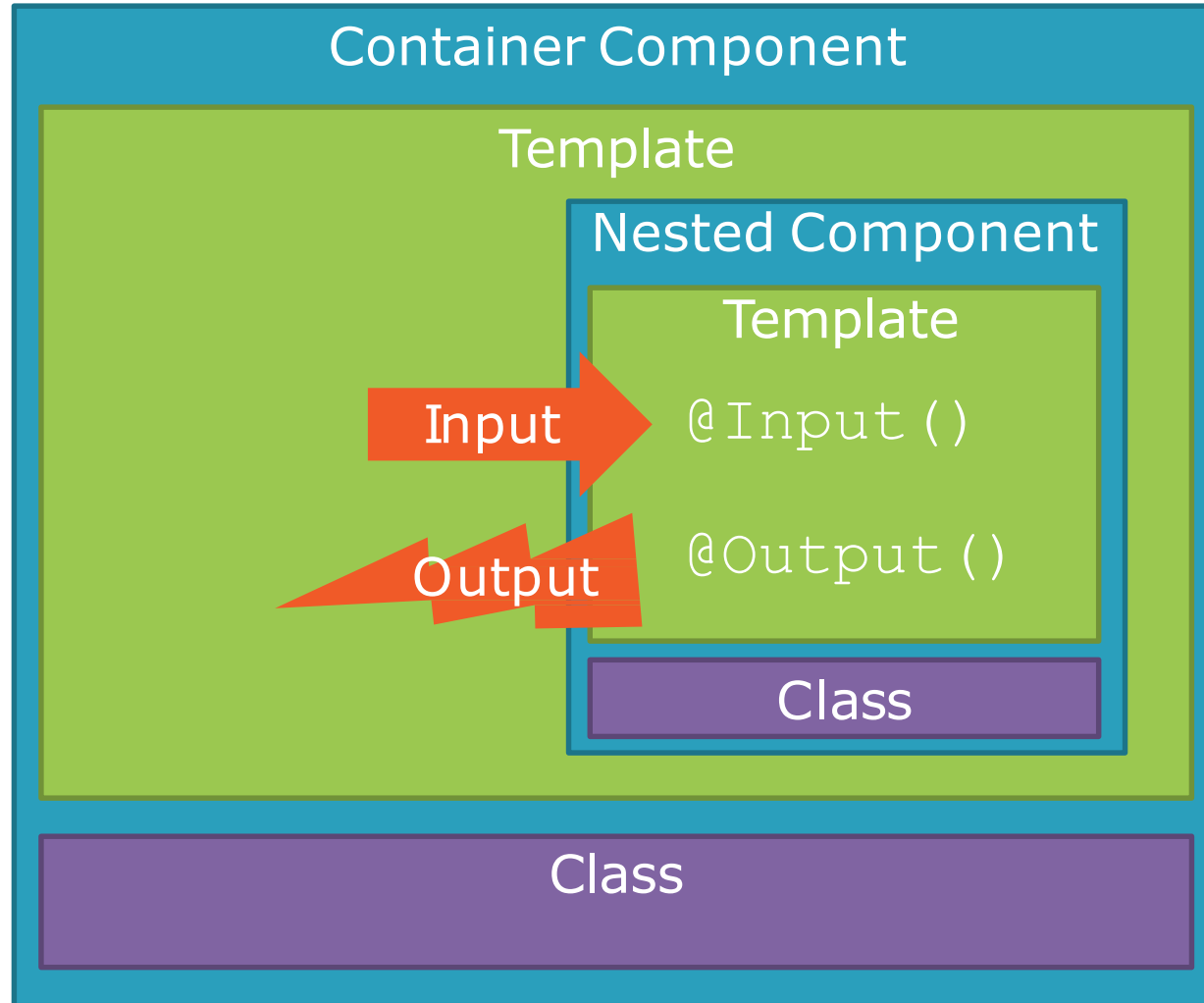
product-list.component.html

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

star.component.ts

```
@Component ({
  selector: 'ai-star',
  templateUrl: 'star.component.html'
})
export class StarComponent {
  @Input rating: number;
  starWidth: number;
}
```

Raising an Event (@Output)



Raising an Event (@Output)

product-list.component.ts

```
@Component ({
  selector: 'pm-products',
  templateUrl: 'product-list.component.html',
  directives: [StarComponent]
})
export class ProductListComponent { }
```

star.component.ts

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

product-list.component.html

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

Raising an Event (@Output)

product-list.component.ts

```
@Component ({
  selector: 'pm-products',
  templateUrl: 'product-list.component.html',
  directives: [StarComponent]
})
export class ProductListComponent { }
```

product-list.component.html

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

star.component.ts

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

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

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


Raising an Event (@Output)

product-list.component.ts

```
@Component ({
  selector: 'pm-products',
  templateUrl: 'product-list.component.html',
  directives: [StarComponent]
})
export class ProductListComponent { }
```

product-list.component.html

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

star.component.ts

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

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

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

Raising an Event (@Output)

product-list.component.ts

```
@Component ({
  selector: 'pm-products',
  templateUrl: 'product-list.component.html',
  directives: [StarComponent]
})
export class ProductListComponent {
  onNotify(message: string): void { }
}
```

product-list.component.html

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

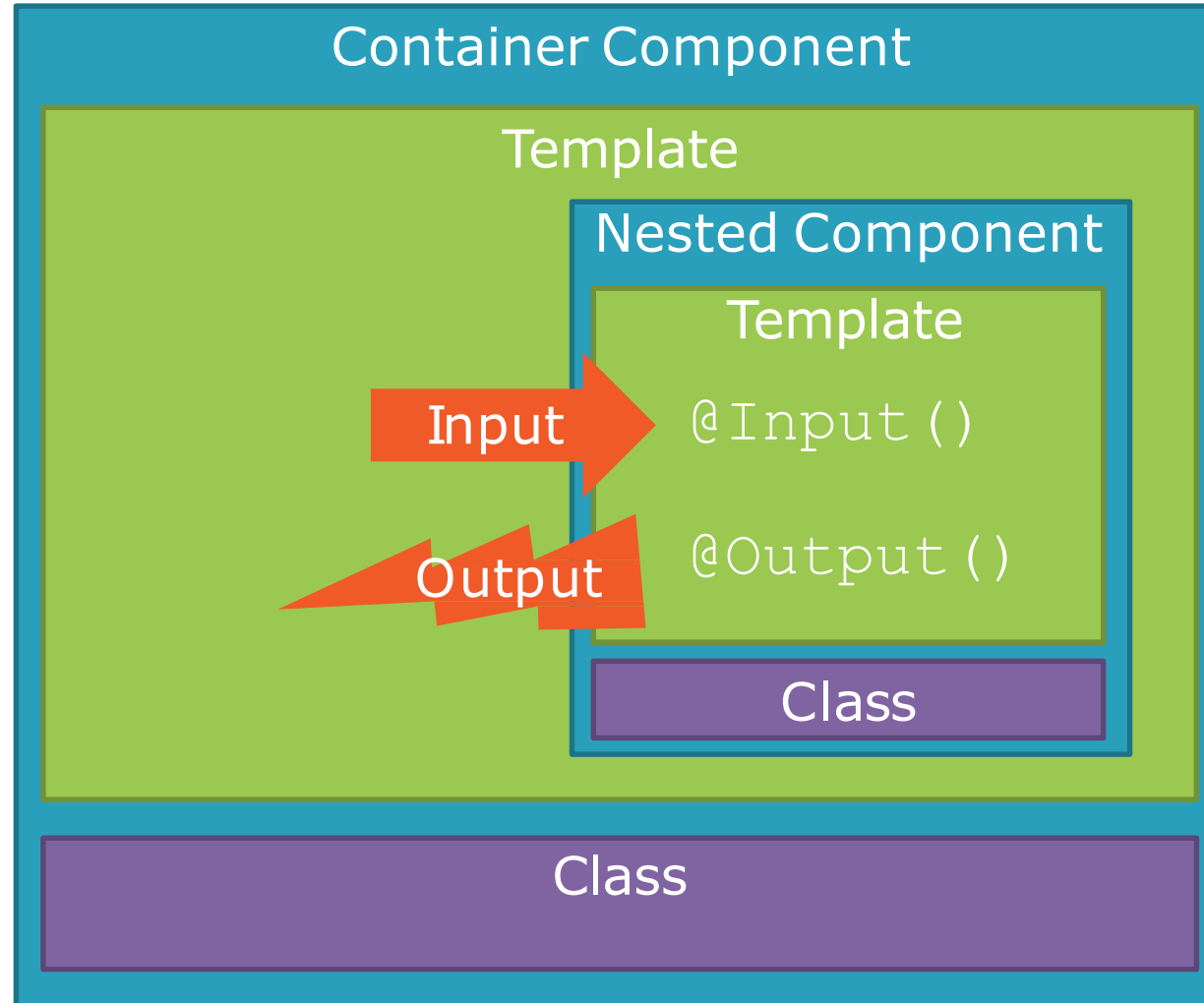
star.component.ts

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

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

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

Nest-able Component's Public API



Checklist: Nested Component



Input decorator

- Attached to a property of any type
- Prefix with @; Suffix with ()

Output decorator

- Attached to a property declared as an EventEmitter
- Use the generic argument to define the event payload type
- Use the new keyword to create an instance of the EventEmitter
- Prefix with @; Suffix with ()

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 payload passed from the nested component

Summary



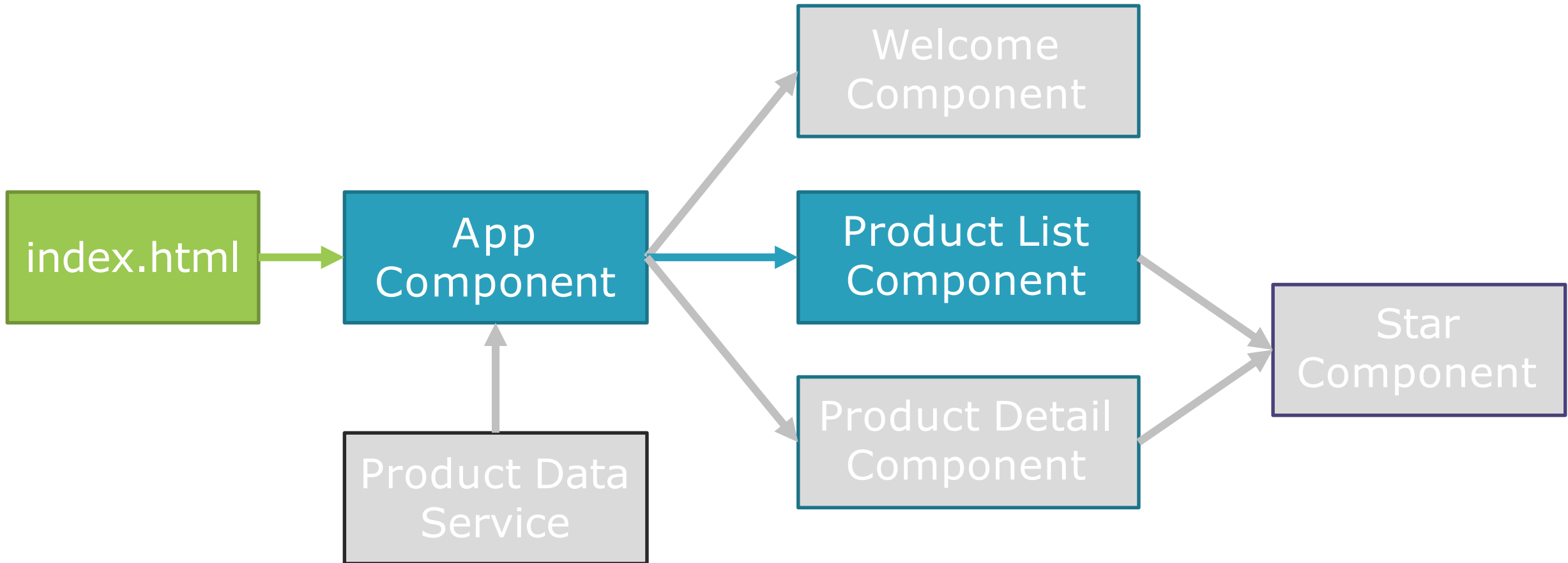
Building a Nested Component

Using a Nested Component

Passing Data to a Nested Component
Using @Input

Raising an Event from a Nested
Component Using @Output

Application Architecture



Application Architecture

