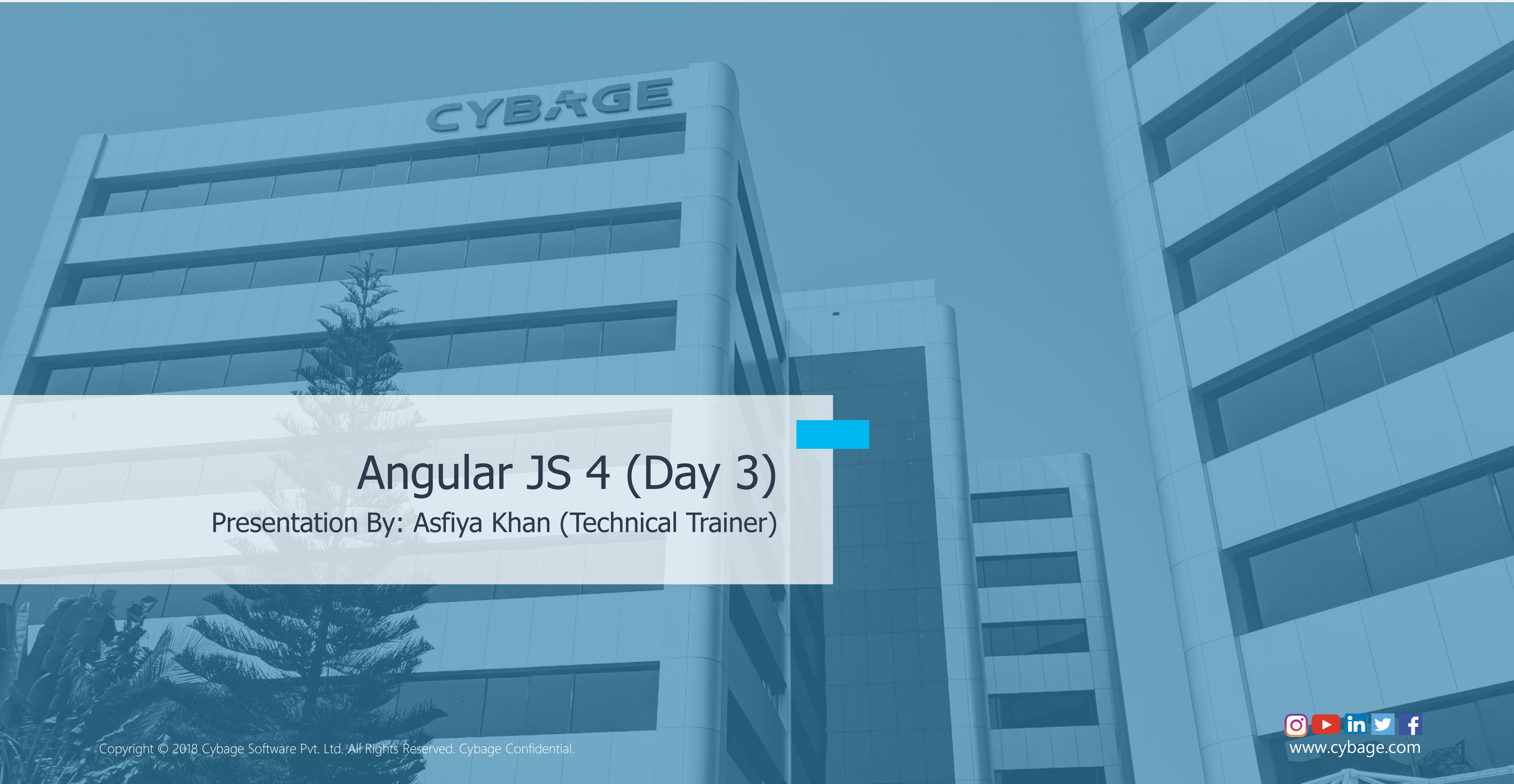


Angular JS 4 (Day 3)

Presentation By: Asfiya Khan (Technical Trainer)

Document History

Version No.	Authored/ Modified by	Remarks/ Change History	Date <dd- mon-yy >
1.0	Asfiya Khan	First version of Angular 4	13 March 2018

Course Structure

Target audience	Trainee,SE,SSE
Level	1,2,3
Pre-requisites	Javascript,TypeScript,HTML,CSS
Training methods	Presentation , Demos, Hands-on
Evaluation	Multiple Choice Question

CYBAGE

Agenda

- Architecture and Components
- Data Binding and Pipes
- Routing and Navigation
- Templates ,Interpolation and Directives
- Angular Modules
- Services and Dependency Injection
- Ng-Forms
- Retrieving data using HTTP

Improving Our Components

- Strong typing & interfaces
- Encapsulating styles
- Lifecycle hooks
- Custom pipes
- Nested components

Interface

A specification identifying a related set of properties and methods.

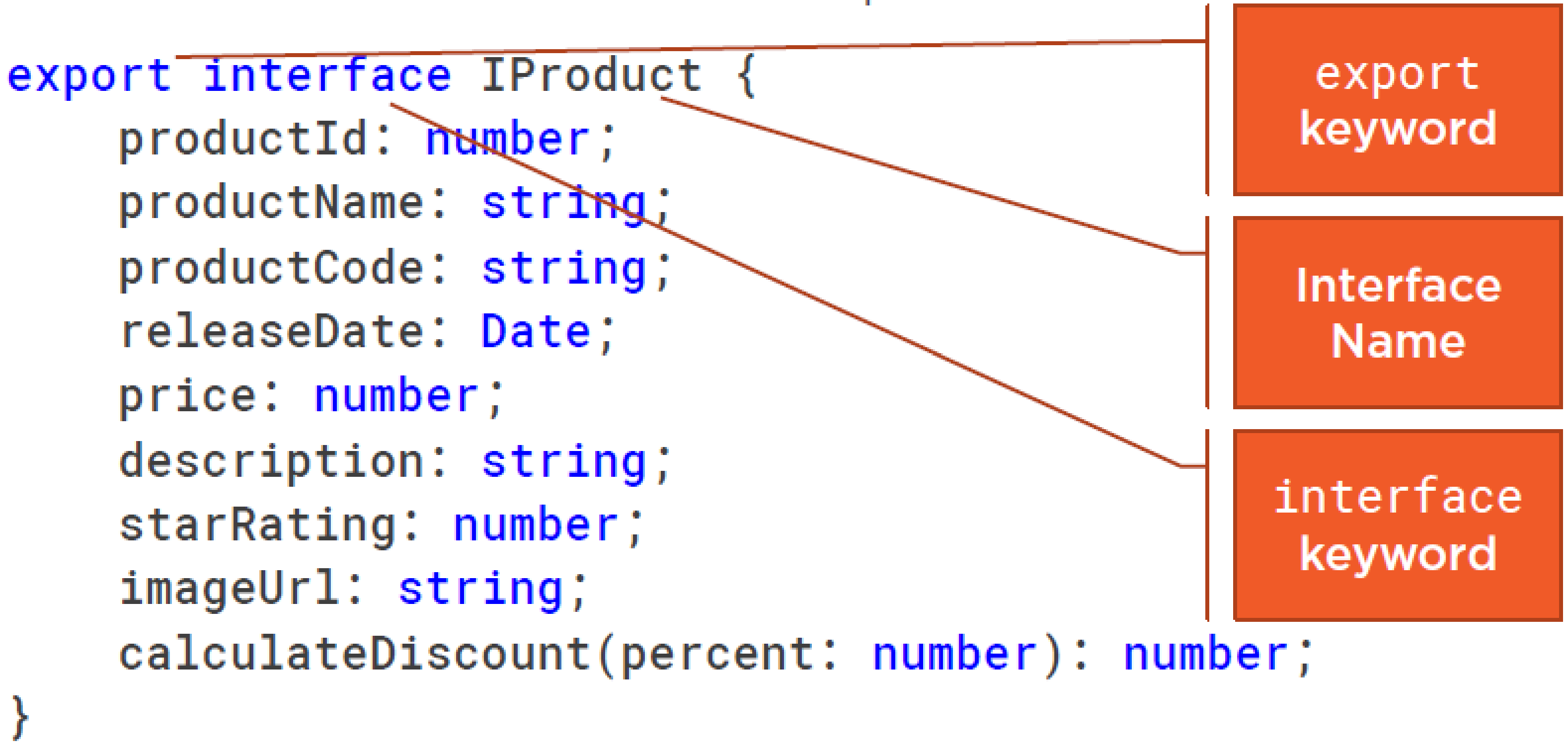
A class commits to supporting the specification by implementing the interface.

Use the interface as a data type.

Development time only!

CYBAGE

Interface Is a Specification



```
export interface IProduct {
    productId: number;
    productName: string;
    productCode: string;
    releaseDate: Date;
    price: number;
    description: string;
    starRating: number;
    imageUrl: string;
    calculateDiscount(percent: number): number;
}
```

Using an Interface as a Data Type

```
import { IProduct } from './product';

export class ProductListComponent {
  pageTitle: string = 'Product List';
  showImage: boolean = false;
  listFilter: string = 'cart';

  products: IProduct[] = […];

  toggleImage(): void {
      this.showImage = !this.showImage;
   }
}
```

Encapsulating Component Styles

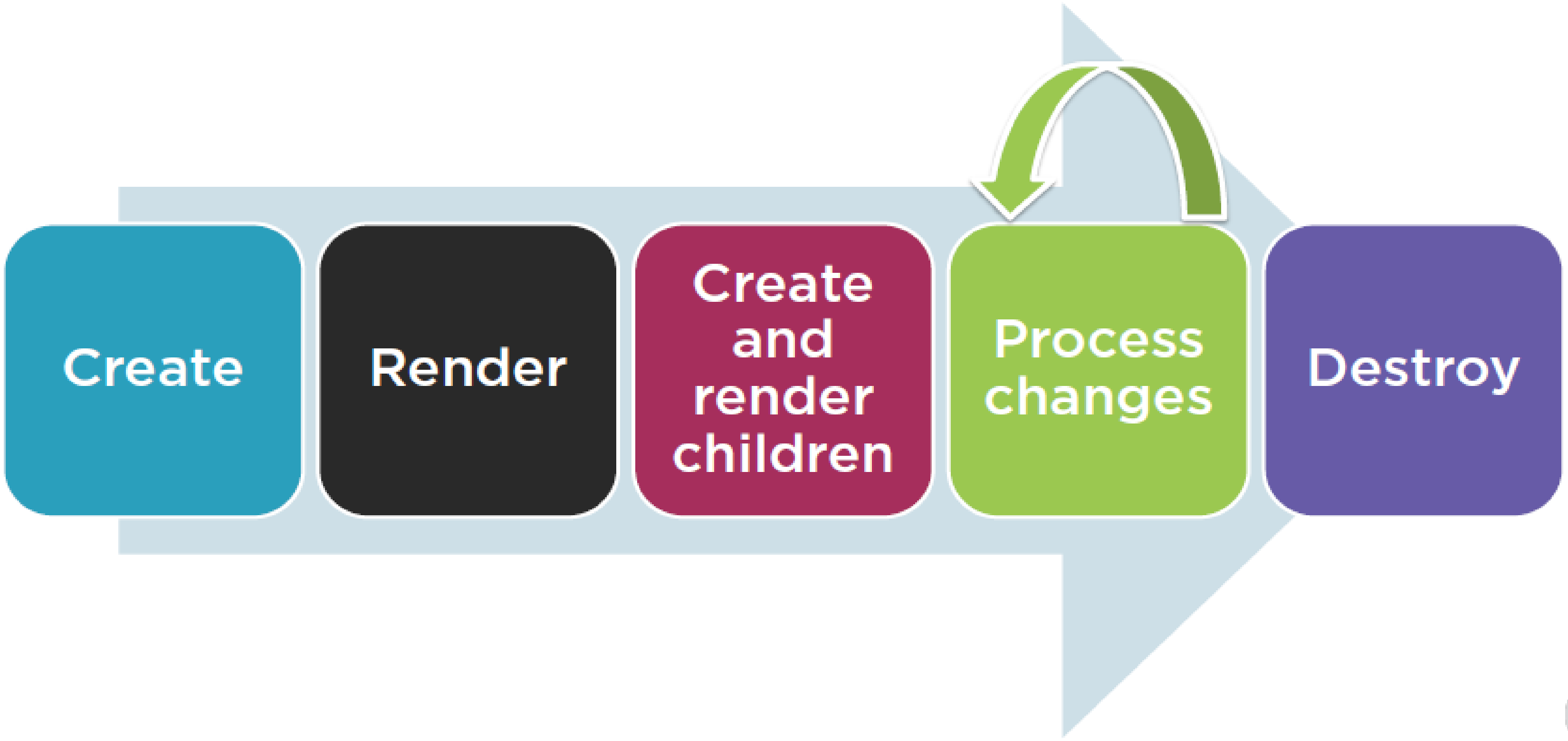
styles

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

styleUrls

```
@Component({
    selector: 'pm-products',
    templateUrl: './product-list.component.html',
    styleUrls: ['./product-list.component.css']})
```

Component Lifecycle



Component Lifecycle Hooks

OnInit: Perform component initialization, retrieve data

OnChanges: Perform action after change to input properties

OnDestroy: Perform cleanup

Using a Lifecycle Hook

```
2

   export class ProductListComponent
1                           implements OnInit {
     pageTitle: string = 'Product List';
     showImage: boolean = false;
     listFilter: string = 'cart';
     products: IProduct[] = […];

3



   }
```

Building a Custom Pipe

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
    name: 'convertToSpaces'
})
export class ConvertToSpacesPipe
                  implements PipeTransform {

  transform(value: string,
            character: string): string {
  }
}
```

Using a Custom Pipe

Template

```
<td>{{ product.productCode | convertToSpaces:'-'}}</td>
```

Pipe

```
  transform(value: string, character: string): string {

  }
```

Using a Custom Pipe

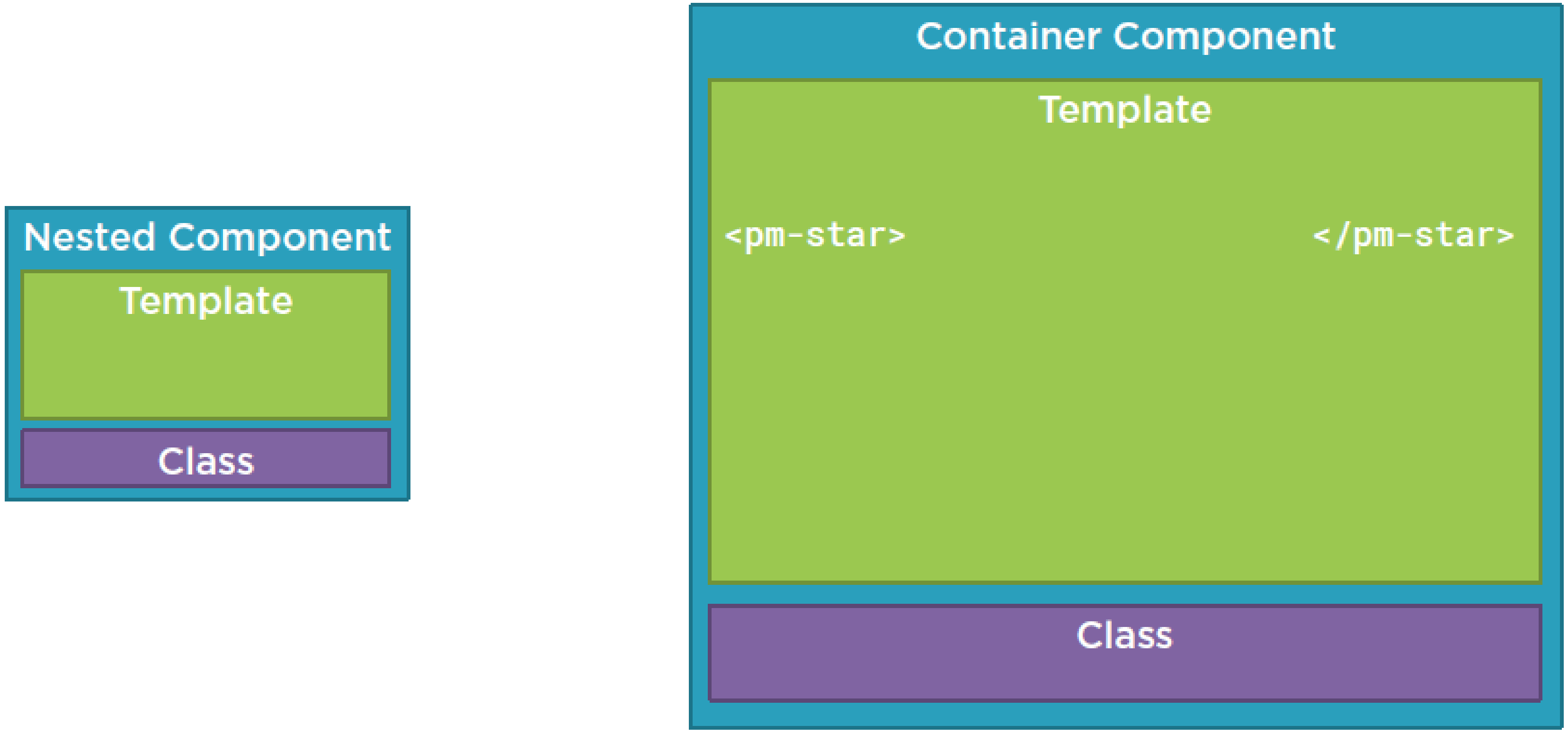
Template

```
<td>{{ product.productCode | convertToSpaces:'-'}}</td>
```

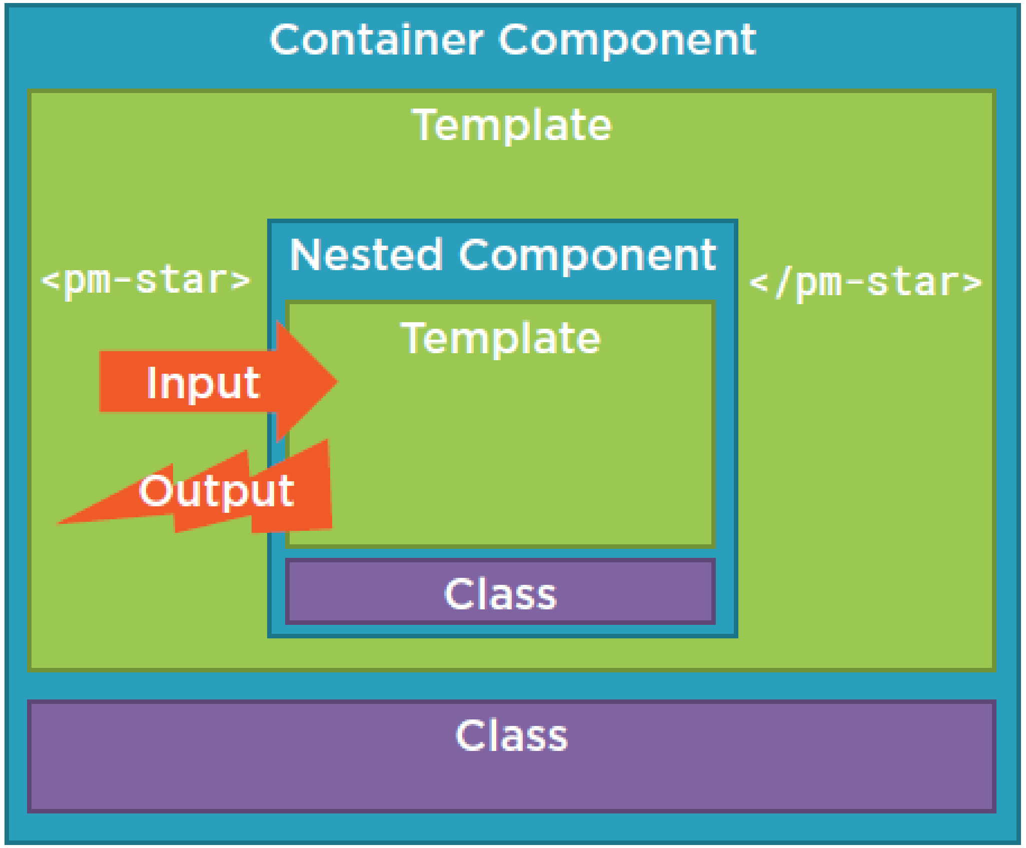
Module

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

Building a Nested Component



Building a Nested Component



Using a Nested Component as a Directive

product-list.component.ts

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

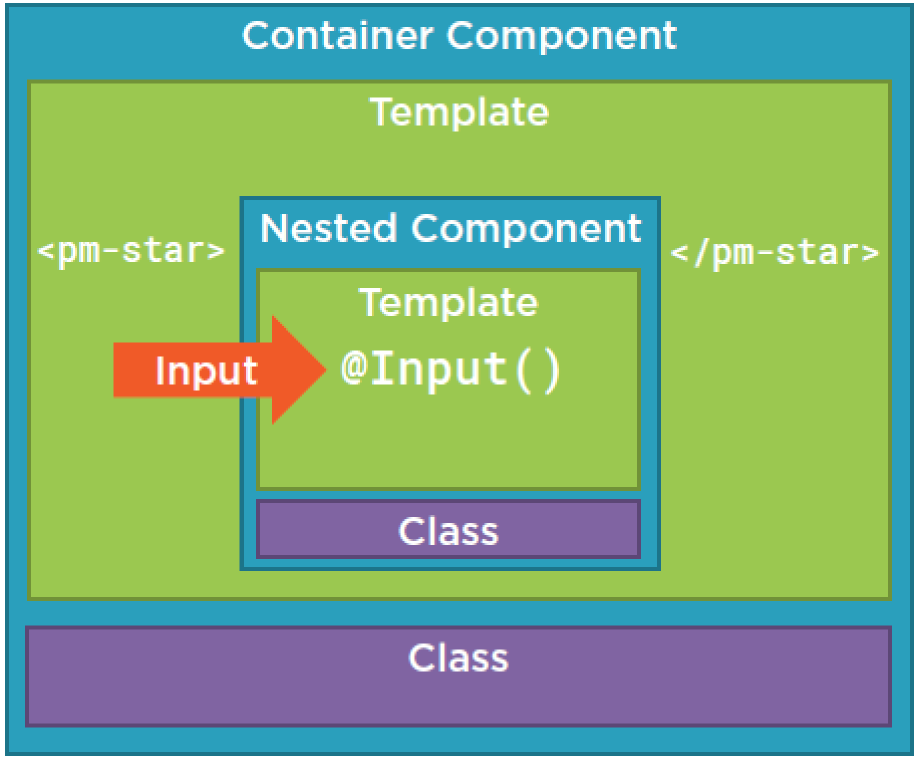
product-list.component.html

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

star.component.ts

```
@Component({
  selector: 'pm-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'
})
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;
  starWidth: number;
}
```

Raising 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;
 starWidth: number;
 @Output() notify: EventEmitter<string> =
             new EventEmitter<string>();
}
```

product-list.component.html

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

Raising an Event (@Output)

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;
 starWidth: number;
 @Output() notify: EventEmitter<string> =
            new EventEmitter<string>();

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

star.component.html

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

Raising an Event (@Output)

product-list.component.ts

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

product-list.component.html

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

star.component.ts

```
@Component({
  selector: 'pm-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!');
 }
}
```

star.component.html

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

DEMO

Any questions?

Thank You!

www.cybage.com