

AngularJS

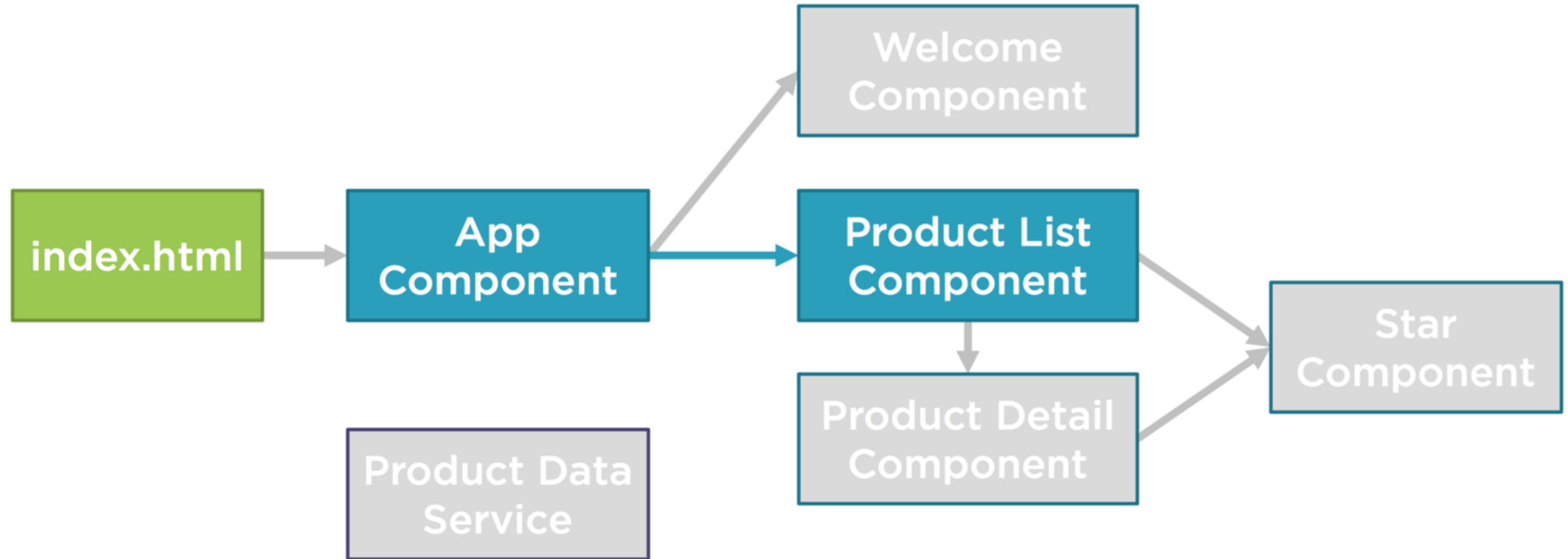
More on components



Module Overview

- Defining an interface
- Encapsulating component styles
- Using Lifecycle hooks
- Building a custom pipe
- Defining relative paths with module id

Application Architecture



Strong typing

```
export class ProductListComponent {  
    pageTitle: string = 'Title';  
    showImage: boolean = false;  
    listFilter: string = 'cart';  
    message: string;  
    products: Array<any> = [...];  
  
    toggleImage(): void {  
        this.showImage = !this.showImage;  
    }  
    onRatingClicked(message: string): void {  
        this.message = message;  
    }  
}
```

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!!!

Interface is a Specification

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

Using an interface as a Data Type

```
import { IProduct } from './product';
export class ProductListComponent implements IProduct {
  pageTitle: string = 'Title';
  showImage: boolean = false;
  listFilter: string = 'cart';
  products: Array<any> = [...];

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


Handling Unique Component Styles

- Templates sometimes require unique styles
- Different ways to do it:
 - Inline directly the style into the HTML
 - Use a specific stylesheet and link it into the index.html
 - Or better:
use a specific stylesheet and link it to the component

Encapsulating Component Styles

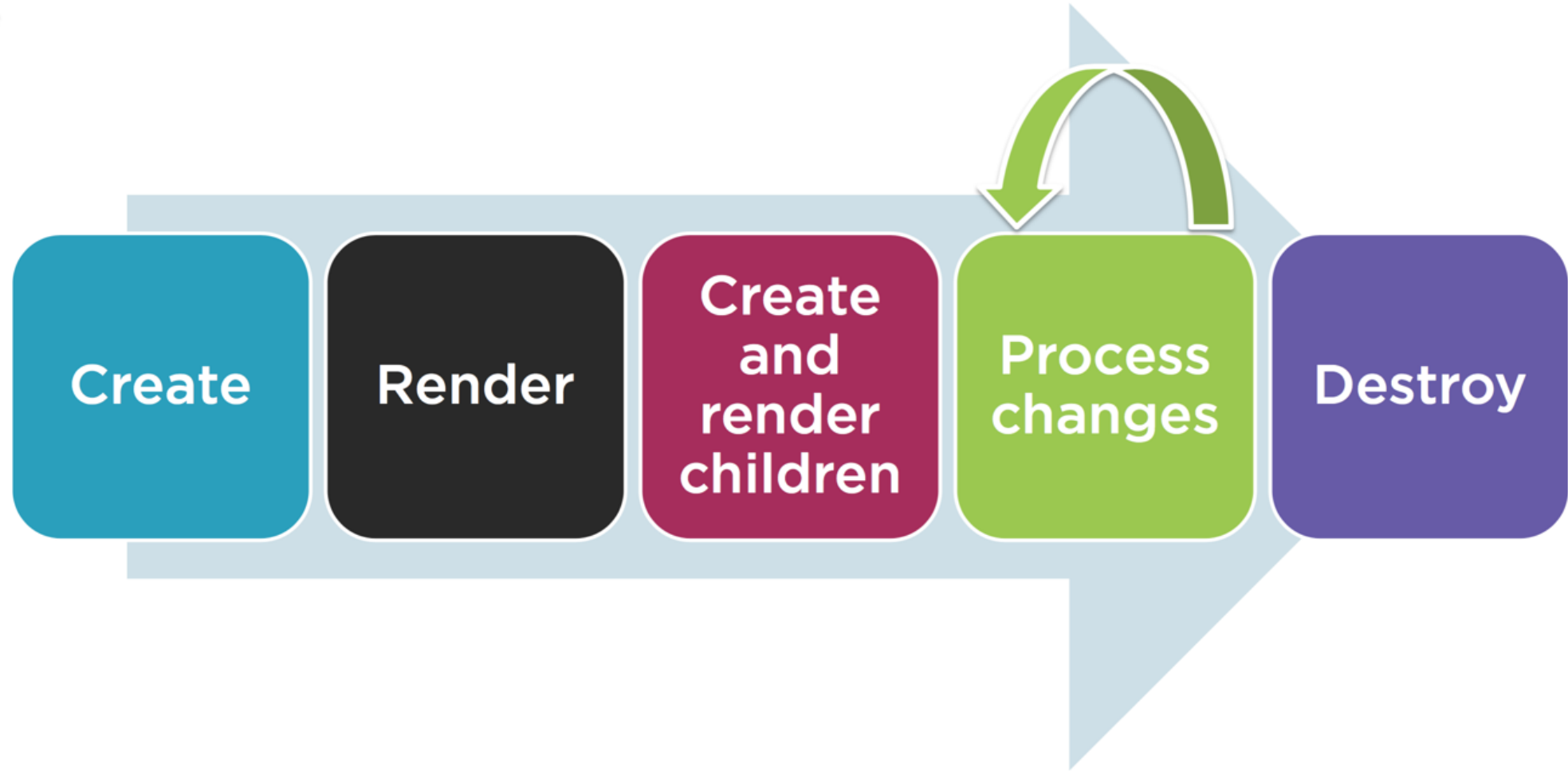
styles

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

styleUrls

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

Component Lifecycle



Component Lifecycle Hooks

- **OnInit**

Perform component initialization, retrieve data

- **OnChange**

Perform action after change to input properties

- **OnDestroy**

Perform cleanup

Using a Lifecycle hook

```
import { OnInit } from '@angular/core';
export class ProductListComponent implements OnInit {
  pageTitle: string = 'Title';
  showImage: boolean = false;
  listFilter: string = 'cart';
  products: Array<any> = [...];

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

  ngOnInit(): void {}
}
```

Transforming Data with Pipes

- Pure functions which transform properties before display
- Built-in pipes
 - date
 - number, decimal, percent, currency
 - json, slice,
 - More on the angular.io documentation
<http://>
- Custom pipes

Pipe Examples

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

```

```

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

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

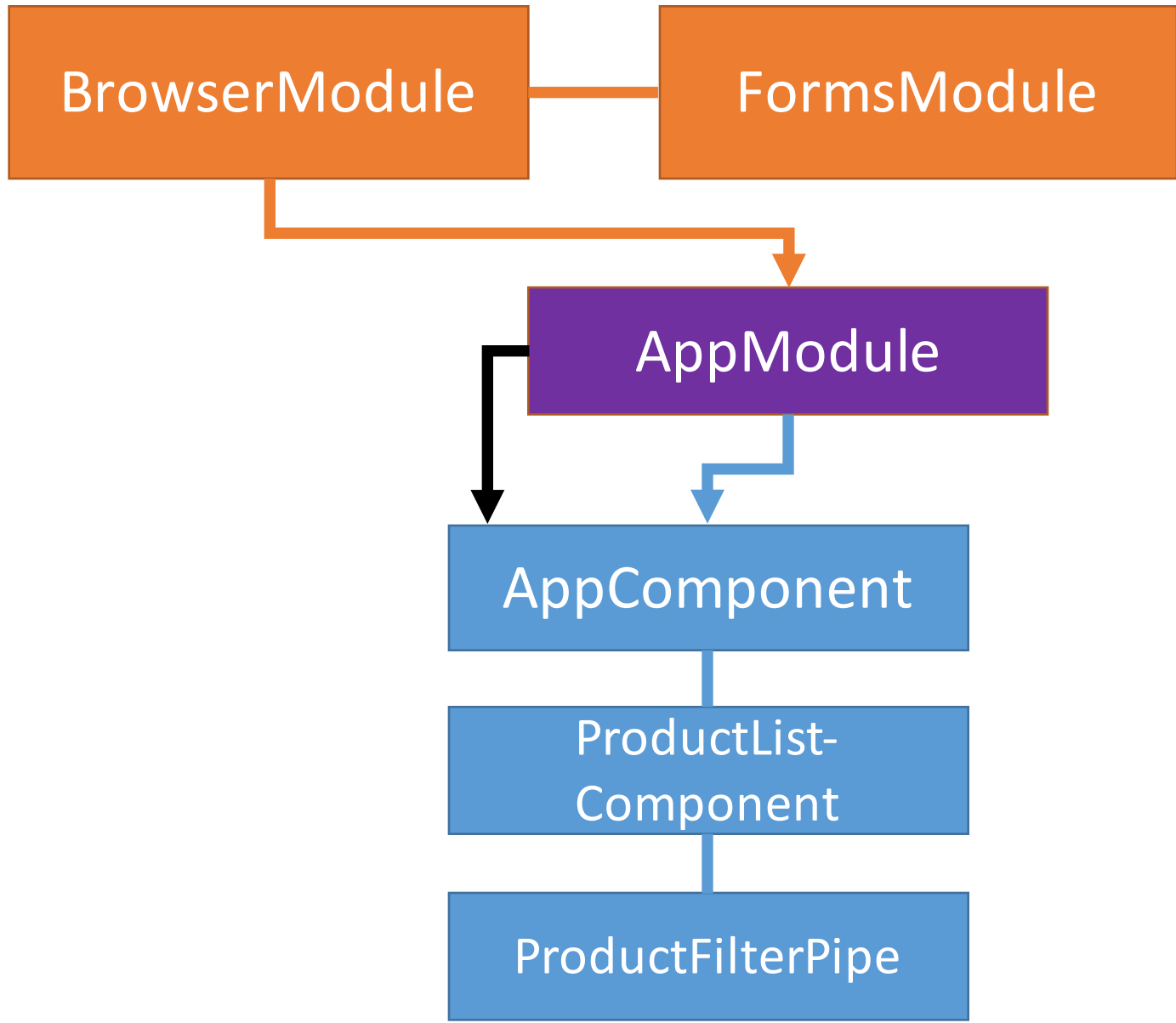
Building a custom pipe

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

@Pipe({
  name: 'productFilter'
})
export class ProductFilterPipe implements PipeTransform {
  transform(value: Iproduct[], filterBy: string): Iproduct[] {
    ...
  }
}
```


Use a custom pipe

```
<tr *ngFor="let product of products | productFilter:  
listFilter">
```



Use a custom pipe

```
// product-list.component.html
<tr *ngFor="let product of products | productFilter: listFilter
```

Relative Paths and Module id

```
...  
@Component({  
  selector: 'nat-products',  
  templateUrl: 'app/products/product-list.component.html',  
  styles: [ 'app/products/product-list.component.css' ]  
})  
export class ProductListComponent {  
  pageTitle: string = 'Product List';  
  ...  
}
```

Relative Paths and Module id

```
...  
@Component({  
  selector: 'nat-products',  
  moduleId: module.id,  
  templateUrl: 'app/products/product-list.component.html',  
  styles: [ 'app/products/product-list.component.css' ]  
})  
export class ProductListComponent {  
  pageTitle: string = 'Product List';  
  ...  
}
```

Module.id

Variable

- Available when using the CommonJS module format

Contains

- The absolute URL of the component class module files

Requires

- Writing modules in CommonJS format
- Using a module loader, such as SystemJS

More on Modules!

CommonJS vs AMD

- <http://requirejs.org/docs/whyamd.html#commonjs>

Typescript output into CommonJS

- Can be customizable into the tsconfig.json

Require a module loader

- Like **SystemJS**

Checklist: Interfaces

- ✓ Defines custom types
- ✓ Creating interfaces:
 - ✓ **interface** keyword
 - ✓ export it
- ✓ Implementing interfaces:
 - ✓ **implements** keyword & interface name
 - ✓ write code for each property & method

Checklist: Encapsulating styles

- ✓ styles property
 - ✓ Specify an array of style strings
- ✓ styleUrls property
 - ✓ Specify an array of stylesheet paths

Checklist: Using Lifecycle Hooks

- ✓ Import the lifecycle hook interface
- ✓ Implement the lifecycle hook interface
- ✓ Write code for the hook method

Checklist: Building a custom pipe

- ✓ Import Pipe and Pipe Transform
- ✓ Create a class that implements PipeTransform
 - ✓ export the class
- ✓ Write code for the Transform method
- ✓ Decorate the class with the Pipe decorator

Checklist: Using a custom pipe

- ✓ Import the custom pipe
- ✓ Add the pipe to the declarations array of an Angular Module
- ✓ Any template associated with a component that is also declared in that Angular module can use that pipe
- ✓ Use the Pipe in the template
 - ✓ Pipe character
 - ✓ Pipe name
 - ✓ Pipe arguments (colon-separated)

Checklist: Relative Paths with Module Id

- ✓ Set the moduleId property of the component decorator to module.id
- ✓ Change the Url to a component-relative path:
 - ✓ templateUrl
 - ✓ styleUrls

Module Overview

- Defining an interface
- Encapsulating component styles
- Using Lifecycle hooks
- Building a custom pipe
- Defining relative paths with module id

Application Architecture

