Navigation and Routing
Additional Techniques

Deborah Kurata
Consultant | Speaker | Author | MVP | GDE

@deborahkurata


MAR
WESTERN
INDIAN
AIRE

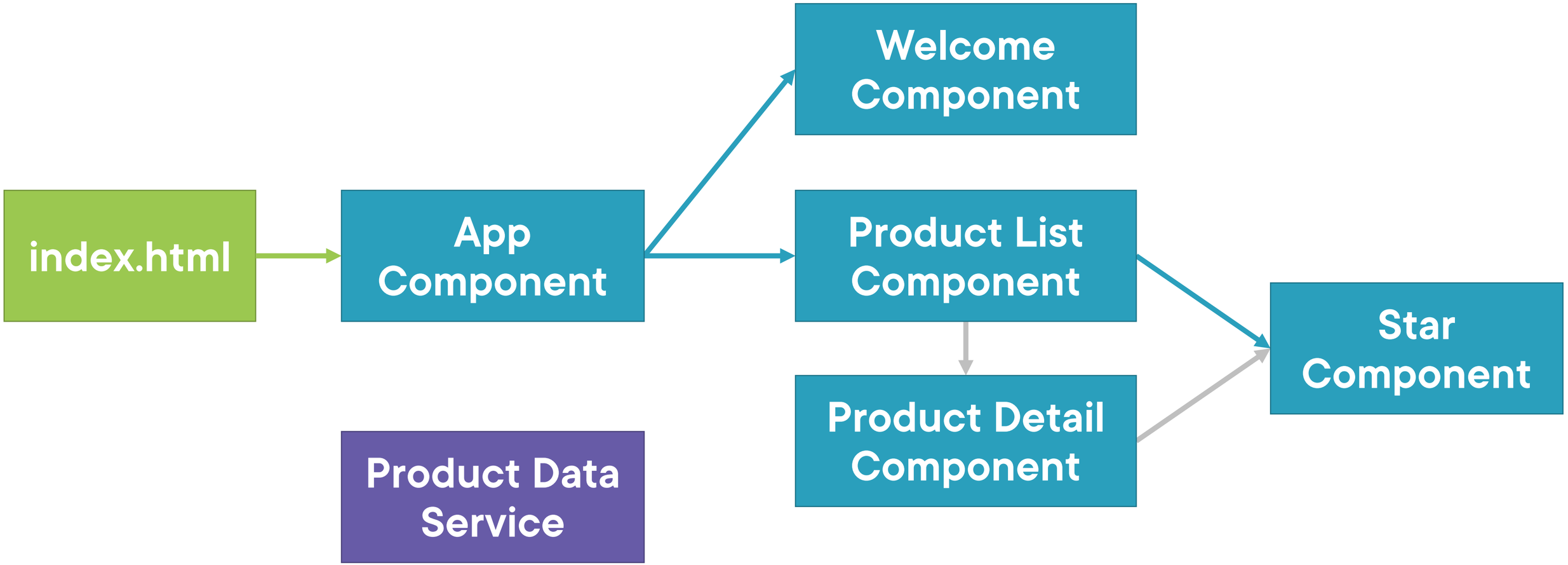
Module Overview

Passing parameters to a route

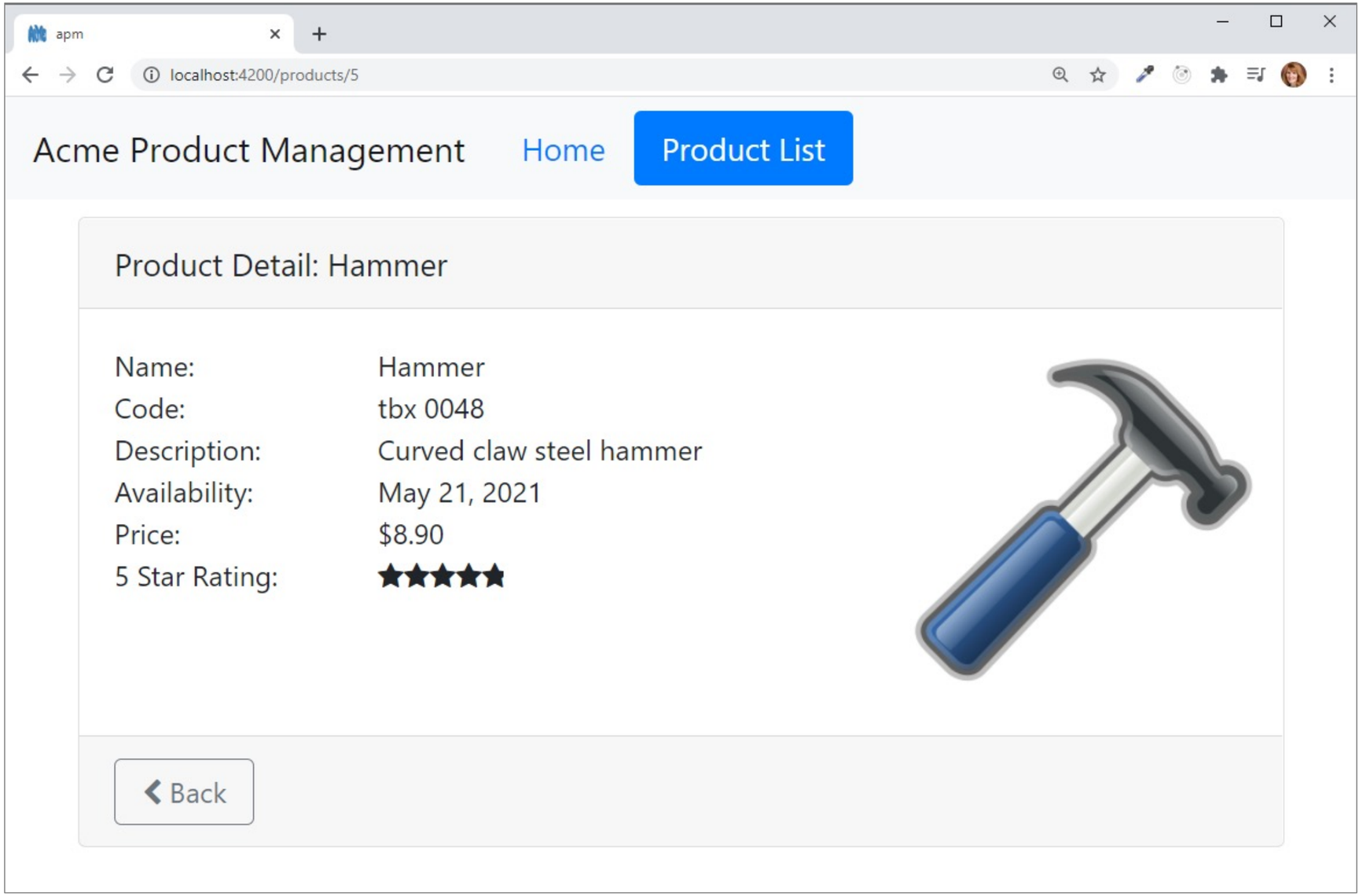
Activating a route with code

Protecting routes with guards

Application Architecture



Passing Parameters to a Route



Passing Parameters to a Route

app.module.ts

```
@NgModule({
  imports: [
    ...,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id', component: ProductDetailComponent },
      { path: 'welcome', component: WelcomeComponent },
      { path: '', redirectTo: 'welcome', pathMatch: 'full' },
      { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
    ])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Passing Parameters to a Route

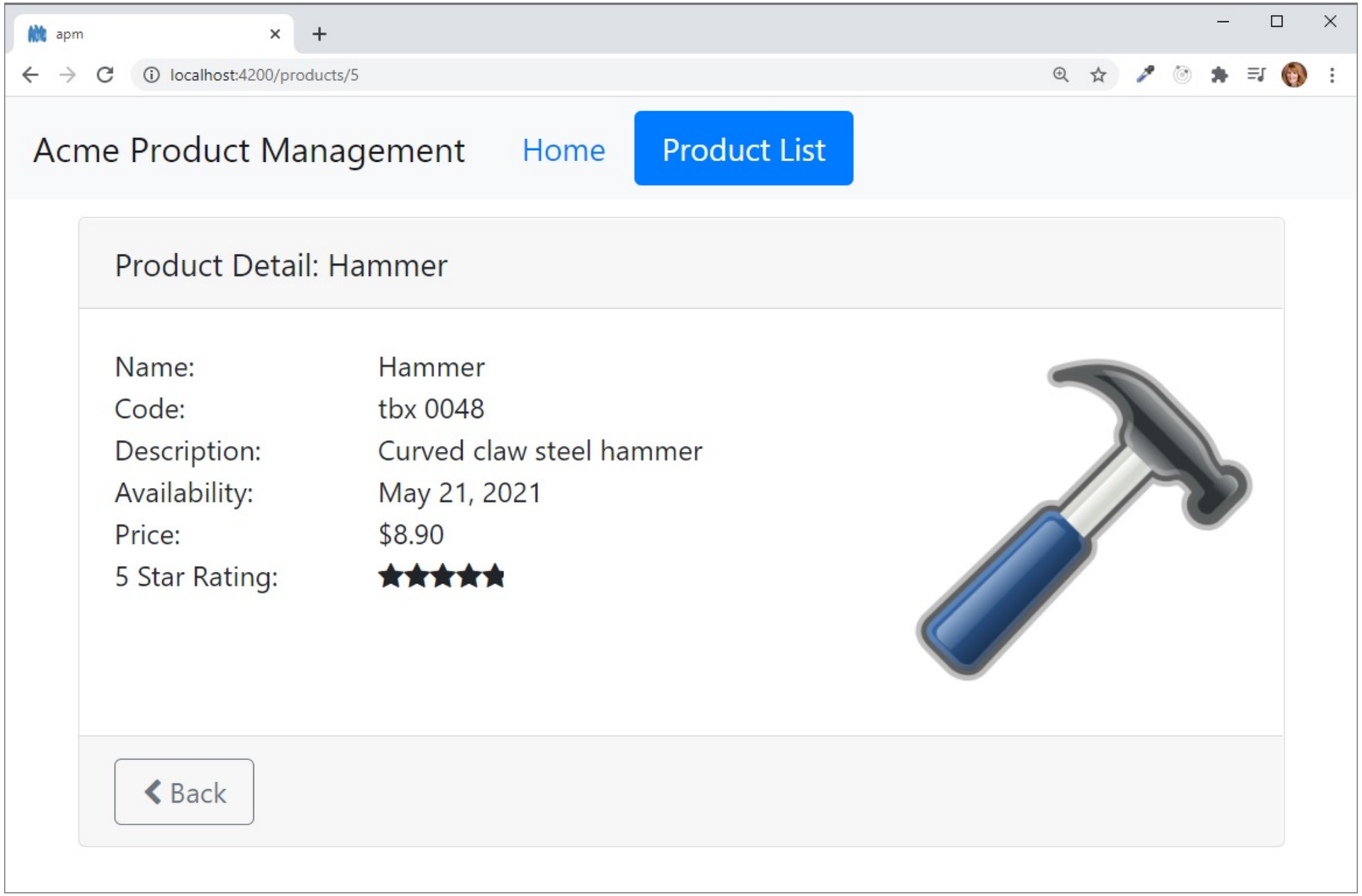
product-list.component.html

```
<td>
  <a [routerLink]="['/products', product.productId]">
     {{product.productName}}
  </a>
</td>
```

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

Reading Parameters from a Route



Reading Parameters from a Route

product-detail.component.ts

```
import { ActivatedRoute } from '@angular/router';

    constructor(private route: ActivatedRoute) { }
```

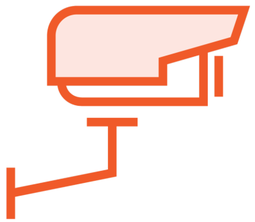
app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

Reading Parameters from a Route

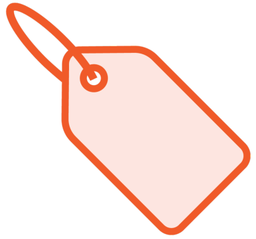
Snapshot : Read the parameter one time

```
this.route.snapshot.paramMap.get('id');
```



Observable: Read emitted parameters as they change

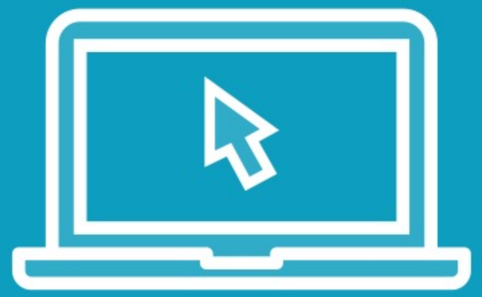
```
this.route.paramMap.subscribe(
  params => console.log(params.get('id'))
);
```



Specified string is the route parameter name

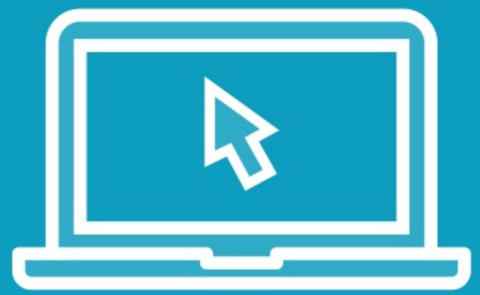
```
{ path: 'products/:id',
  component: ProductDetailComponent }
```

Demo



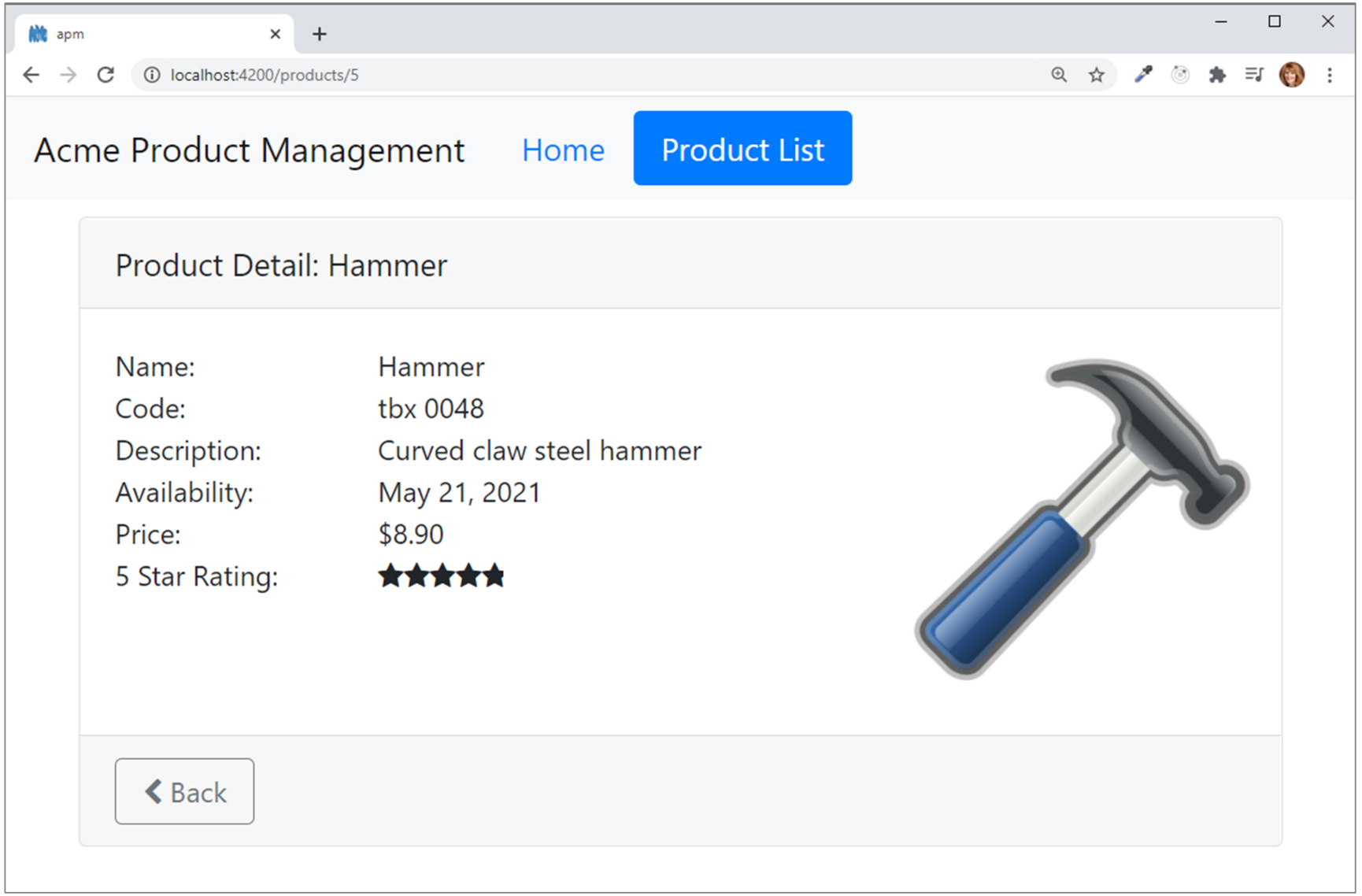
Passing parameters to a route

Demo



Handling null and undefined

Activating a Route with Code



Activating a Route with Code

product-detail.component.ts

```
import { Router } from '@angular/router';
...
    constructor(private router: Router) { }

    onBack(): void {
      this.router.navigate(['/products']);
    }
```

Protecting Routes with Guards

Limit access to a route

Restrict access to only certain users

Require confirmation before navigating away

Protecting Routes with Guards

CanActivate

- Guard navigation to a route

CanDeactivate

- Guard navigation from a route

Resolve

- Pre-fetch data before activating a route

CanLoad

- Prevent asynchronous routing

Building a Guard

product-detail.guard.ts

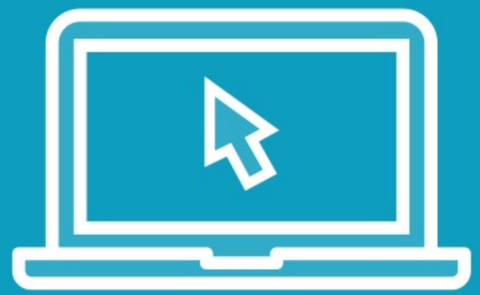
```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
export class ProductDetailGuard implements CanActivate {

    canActivate(): boolean {
      ...
    }
}
```

Using a Guard

app.module.ts

```
@NgModule({
  imports: [
    ...,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id',
        canActivate: [ ProductDetailGuard ],
        component: ProductDetailComponent },
    ...])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Demo



Protecting routes with guards

General Checklist: Null and Undefined

Prevent null or undefined errors in a template

- Safe navigation operator (?)

```
{{product?.productName}}
```

```
{{product?.supplier?.companyName}}
```

- *ngIf directive

```
<div *ngIf='product'>
  <div>Name:</div>
  <div>{{product.productName}}</div>
  <div>Description:</div>
  <div>{{product.description}}</div>
</div>
```

Routing Checklist: Passing Parameters

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

product-list.component.html

```
<a [routerLink]="['/products', product.productId]">
   {{product.productName}}
</a>
```

product-detail.component.ts

```
import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
   console.log(this.route.snapshot.paramMap.get('id'));
}
```

Routing Checklist: Activate a Route with Code

Use the Router service

- Define it as a dependency

Create a method that calls the navigate method of the Router service

- Pass in the link parameters array

```
import { Router } from '@angular/router';
...
    constructor(private router: Router) { }

    onBack(): void {
       this.router.navigate(['/products']);
    }
```

Add a user interface element

- Use event binding to bind to the created method

```
<button (click)='onBack()'>Back</button>
```

Routing Checklist: Protecting Routes with Guards

Build a guard service

- Implement the guard type (CanActivate)
- Create the method (canActivate())

Register the guard service provider

- Use the providedIn property

```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';

@Injectable({ providedIn: 'root' })
export class ProductDetailGuard implements CanActivate {
    canActivate(): boolean { ... }
}
```

Add the guard to the desired route

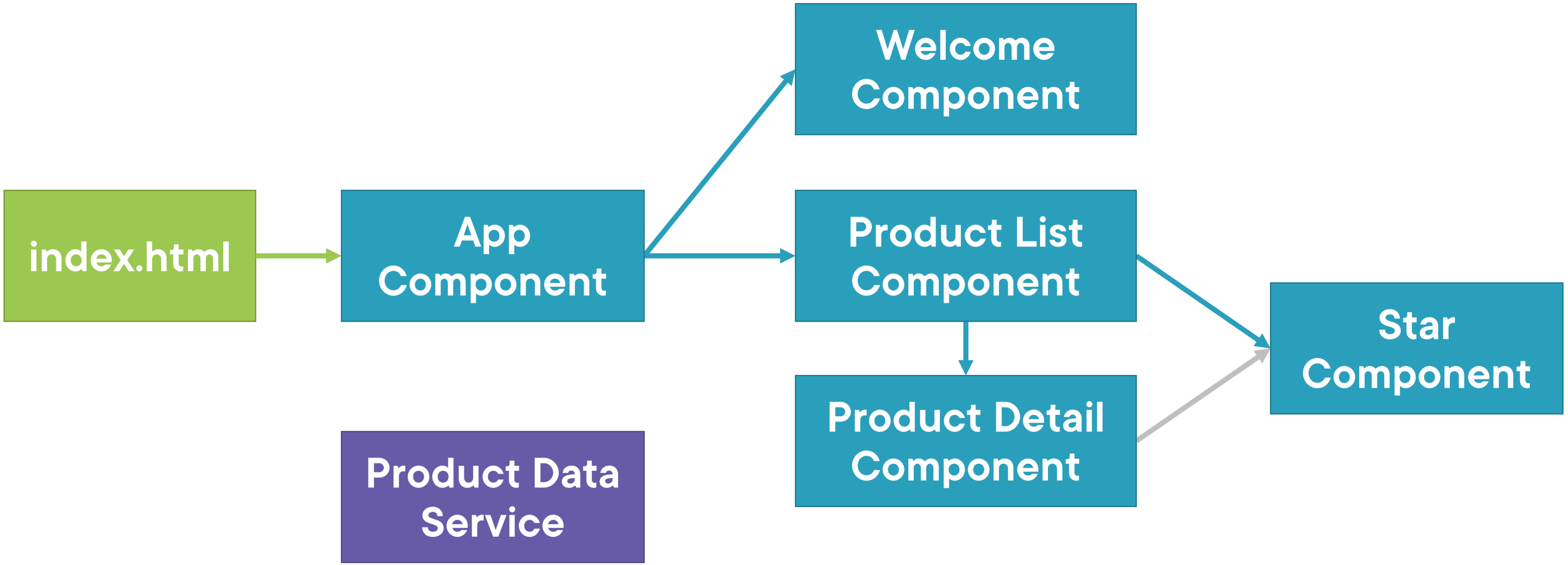
```
{ path: 'products/:id', canActivate: [ ProductDetailGuard ],
  component: ProductDetailComponent },
```

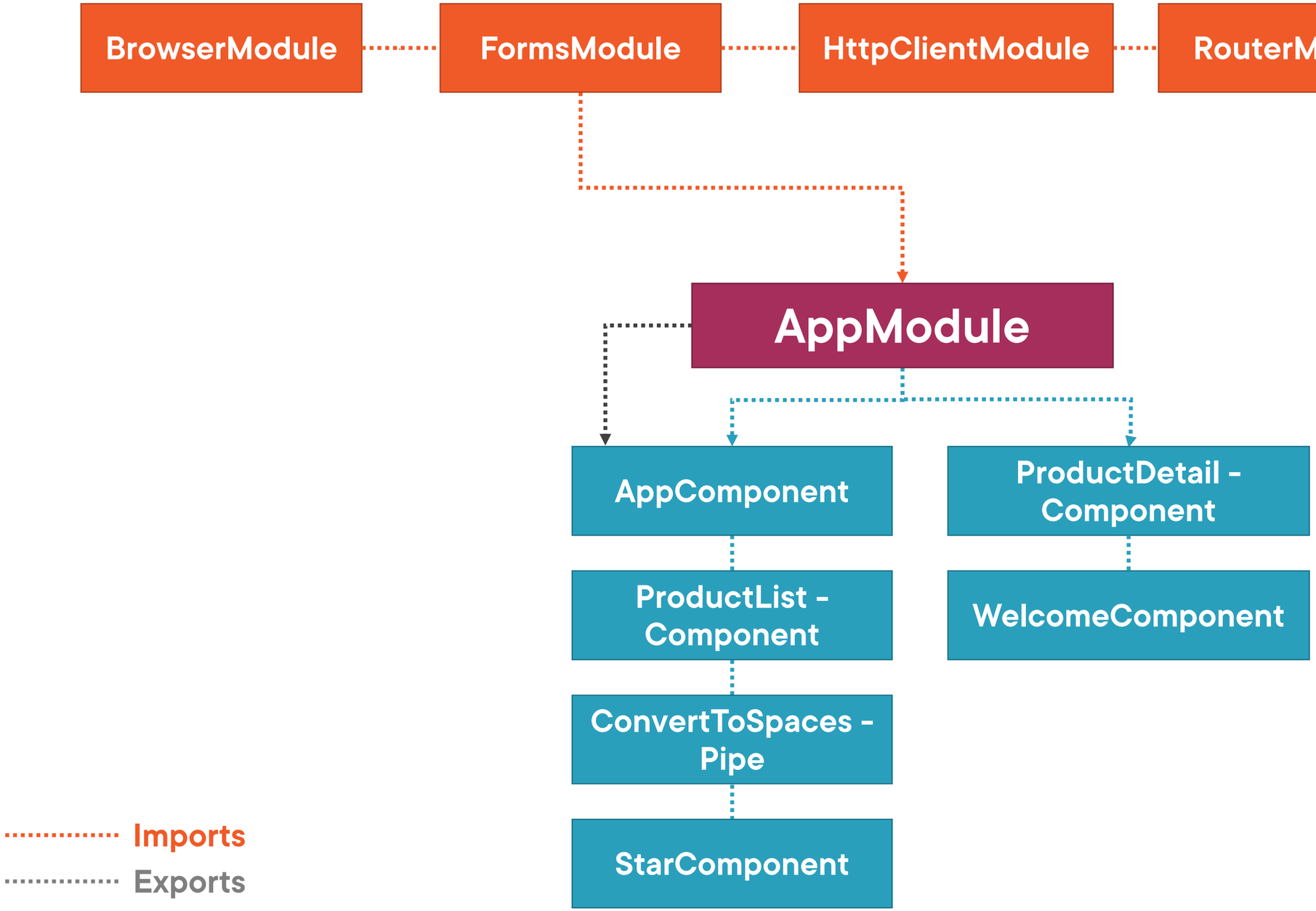
Learning More

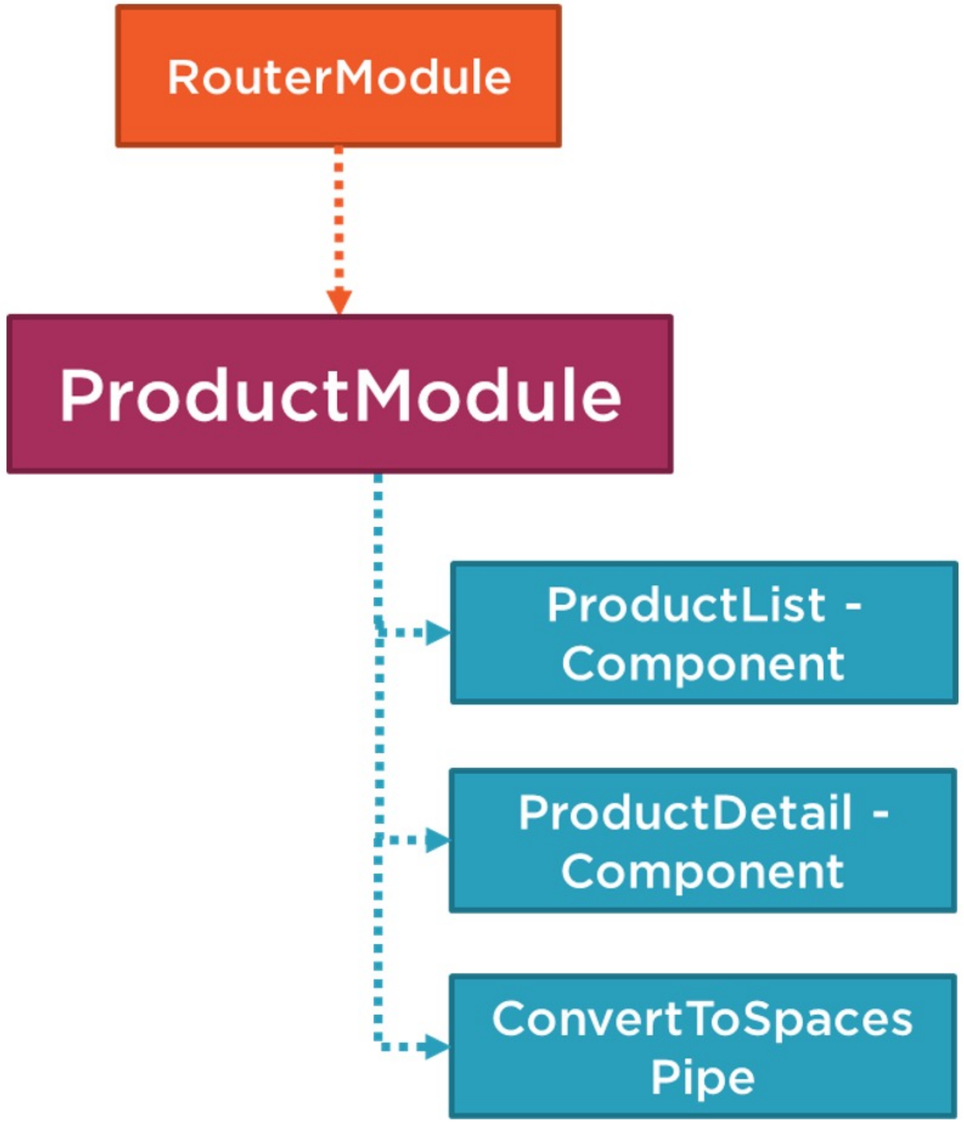
"Angular Routing"

- **Required, optional, & query parameters**
- **Prefetching with route resolvers**
- **Child and secondary router outlets**
- **Router guards**
- **Lazy loading**

Application Architecture




BrowserModule
FormsModule
HttpClientModule
RouterModule
AppModule
AppComponent
ProductDetail - Component
ProductList - Component
WelcomeComponent
ConvertToSpaces - Pipe
StarComponent
Imports
Exports
Declarations
Bootstrap



Coming up next ...

Angular Modules