Benjamin Longearet

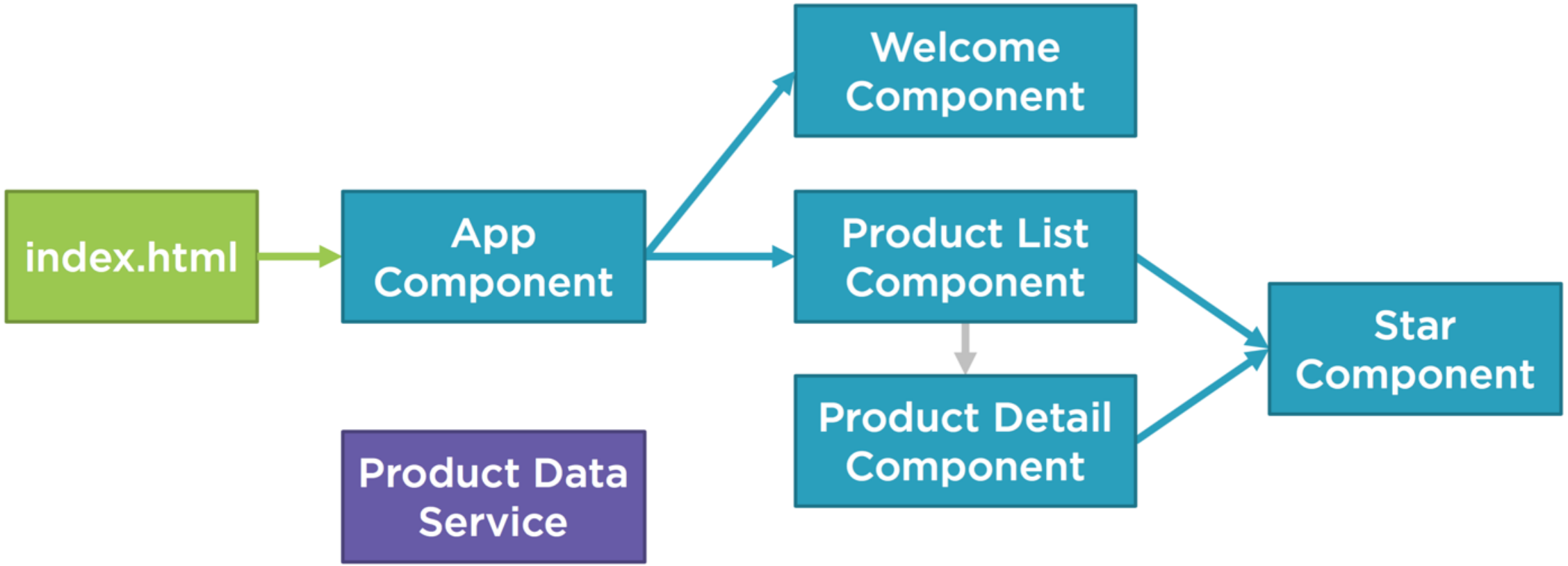
Angular

Navigation and Routing Additional Techniques

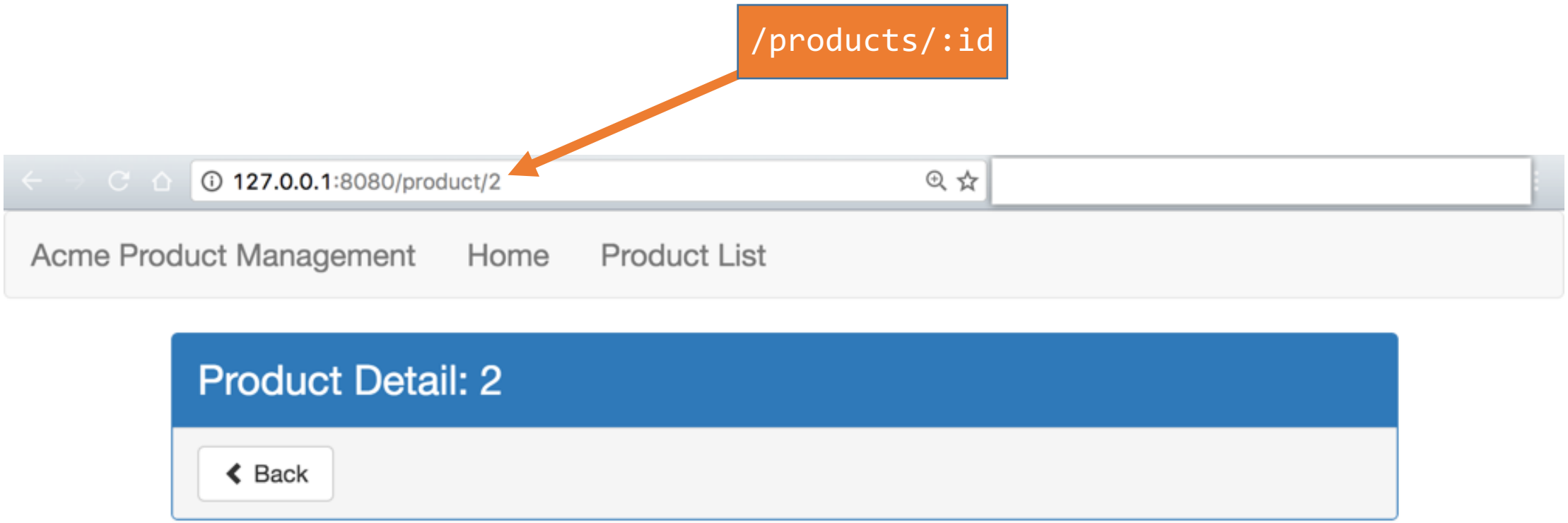
Module Overview

- Passing Parameters to a Route
- Activating a Routes with Code
- Use Resolvers
- Protecting Routes with Guards

Application Architecture



Passing parameters to a Route



Passing parameters to a Route

```
// app.module.ts
import { RouterModule } from '@angular/router';
@NgModule({
  imports: [
    …
    RouterModule.forRoot([
      { path: 'products', component: ProductComponent, children: [
        { path: '', component: ProductListComponent },
        { path: ':id', component: ProductDetailComponent },
      ]},
      { path: 'welcome', component: WelcomeComponent },
      { path: '', redirectTo: 'welcome', pathMatch: 'full' },
      { path: '**', component: PageNotFoundComponent }
    ])
  ],
  declarations: [
    …
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

Passing parameters to a Route

```
// product-list.component.html
<td>
  <a [routerLink]="[product.id]">
     {{product.productName}}
  </a>
</td>
```

```
// app.module.ts
{ path: ':id', component: ProductDetailComponent }
```

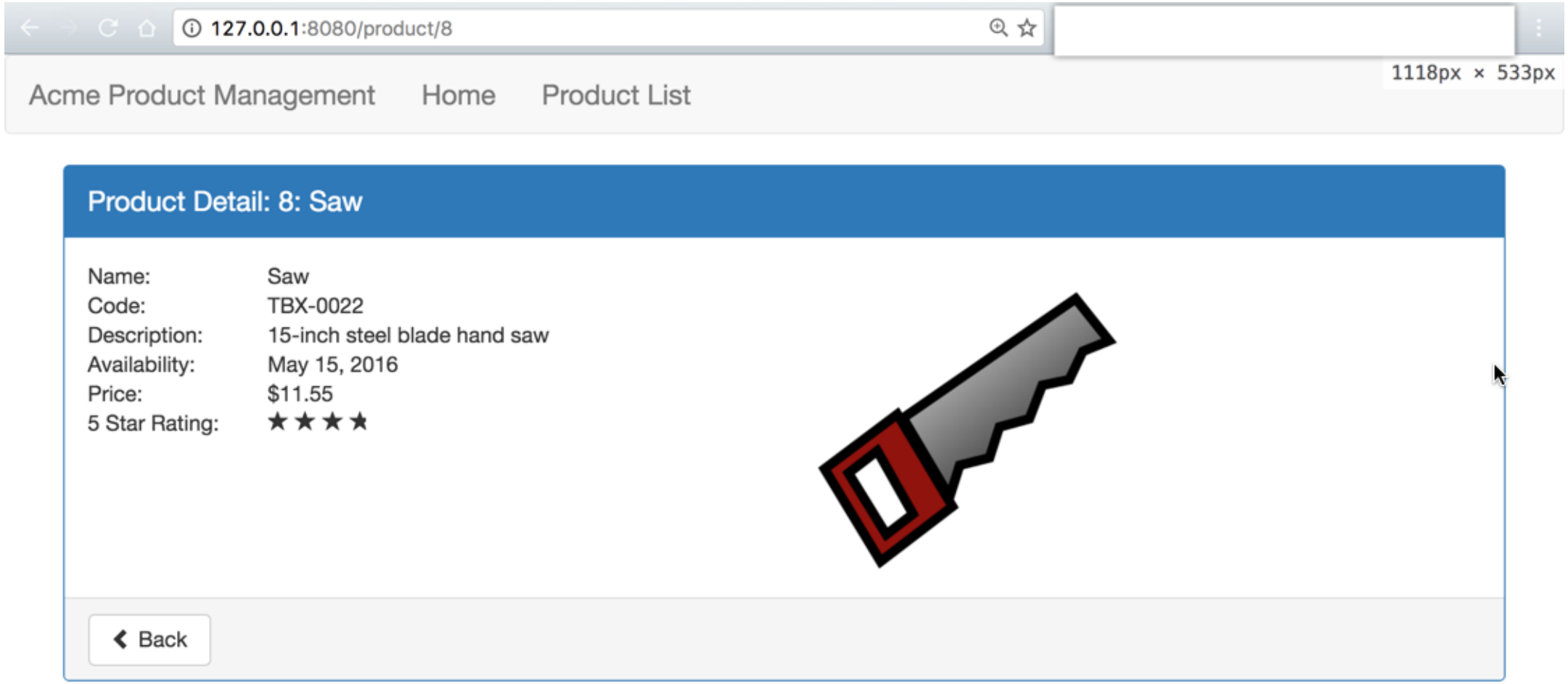
Reading parameters from a Route

```
// product-detail.component.ts
import { ActivatedRoute } from ’@angular/router';

class ProductDetailComponent {
  constructor(private _route: ActivatedRoute) {
    console.log(this._route.snapshot.params['id']);
  }
}


// app.module.ts
{ path: ':id', component: ProductDetailComponent }
```

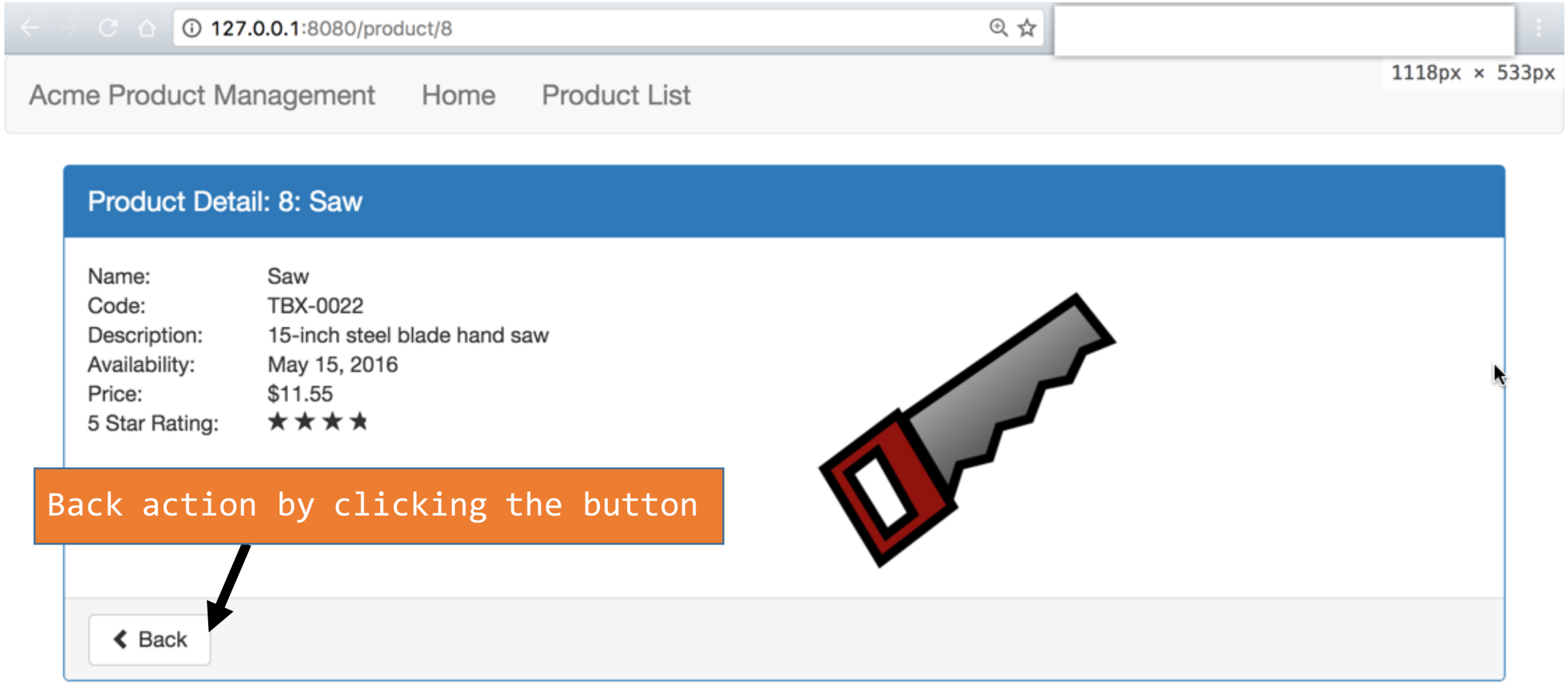
Reading parameters from a Route

```
// product-detail.component.ts
import { ActivatedRoute } from '@angular/router';
import { ProductService } from './product.service';

class ProductDetailComponent {
  constructor(private _route: ActivatedRoute,
              private _productService: ProductService) {}
  ngOnInit() {
    let id = this._route.snapshot.params['id'];
    this._productService.getProductById(id)
      .subscribe(
        product => this.products = products,
        error => this.errorMessage = <any>error
      );
  }
}
```

Reading parameters from a Route



Activating a Route with Code



Activating a Route with Code

```
// product-detail.component.ts
import { Router } from '@angular/router';

export class ProductDetailComponent {
  constructor(private _router: Router) {}
  onBack() {
    this._router.navigate(['/products']);
  }
}
```

Building a Resolver

```
// product-detail.resolver.ts
import { Injectable } from '@angular/core';
import { Resolve, RouterStateSnapshot, ActivatedRouteSnapshot } from '@angular/
router';

@Injectable()
export class ProductDetailResolver implements Resolve<IProduct> {
  resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):
  Promise<Crisis> {
    …
  }
}
```

Registering a Resolver

```
// app.module.ts
import { ProductDetailResolver } from ’./core/models/product-
detail.resolver';

@NgModule({[…],
  declarations: […],
  providers:
  [ ProductDetailResolver ],
})bootstrap: [ AppComponent ]
export class AppModule()
```

Using a Resolver

```
// app.module.ts
import { ProductDetailResolver } from ’./core/models/product-detail.resolver';

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

Protecting Routes with Guards

- **CanActivate**
  - Guard navigation **to** a route
- **CanDeactivate**
  - Guard naviation **from** a route
- **Resolve**
  - Pre-fetch data before activating a route
- **CanLoad**
  - Prevent asynchronous routing

Building a Guard

```
// product-guard.service.ts
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';

@Injectable()
export class ProductDetailGuard implements CanActivate {
  canActivate(): boolean {
    …
  }
}
```

Registering a Guard

```
// app.module.ts
import { ProductDetailGuard } from ’./products/product-guard.service';

@NgModule({
  imports: […],
  declarations: […],
  providers: [ ProductDetailGuard ],
  bootstrap: [ AppComponent ]
})
export class AppModule()
```

Using a Guard

```
// app.module.ts
import { ProductDetailGuard } from ’./products/product-guard.service';

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

Checklist: Activate a Route With Code

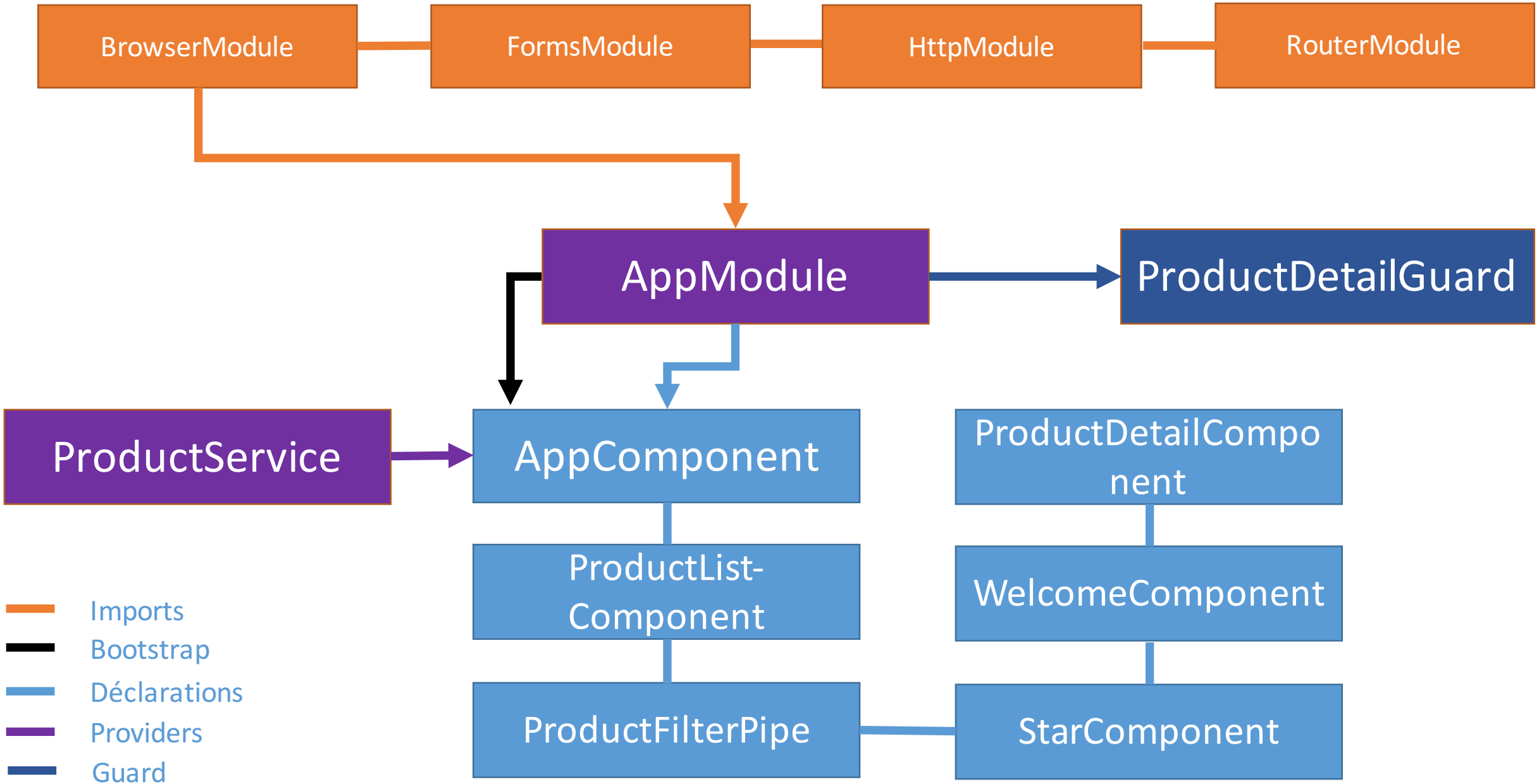
- ✓ Use the Router service
  - ✓ Import the service
  - ✓ Define it as a dependency
- ✓ Create a method that calls the navigate method of the Router service
  - ✓ Pass in the link parameters array (as in a template!)
- ✓ Add a user interface element
  - ✓ Use event binding to bin to the created method

Checklist: Protecting Routes with Guards

- ✓ Build a guard Service
  - ✓ Implement the guard type (CanActivate)
  - ✓ Create the method (CanActivate)
- ✓ Register the guard service provider
  - ✓ Must be in an Angular Module
- ✓ Add the guard to the desired route

Module Overview

- Passing Parameters to a Route
- Activating a Routes with Code
- Using a Resolver
- Protecting Routes with Guards


BrowserModule
FormsModule
HttpModule
RouterModule
AppModule
ProductDetailGuard
ProductService
AppComponent
ProductDetailCompo
nent
ProductList-
Component
WelcomeComponent
ProductFilterPipe
StarComponent
Imports
Bootstrap
Déclarations
Providers
Guard

Application Architecture