Angular Fundamentals

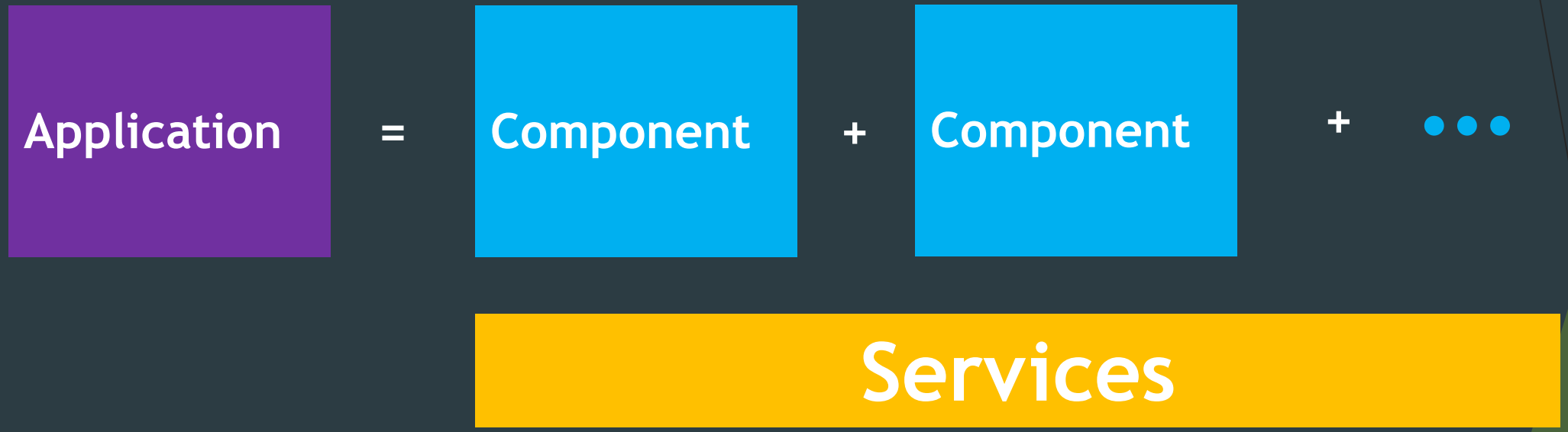
What is Angular

- A JavaScript framework
- For building client-side applications
- Using HTML, CSS and TypeScript

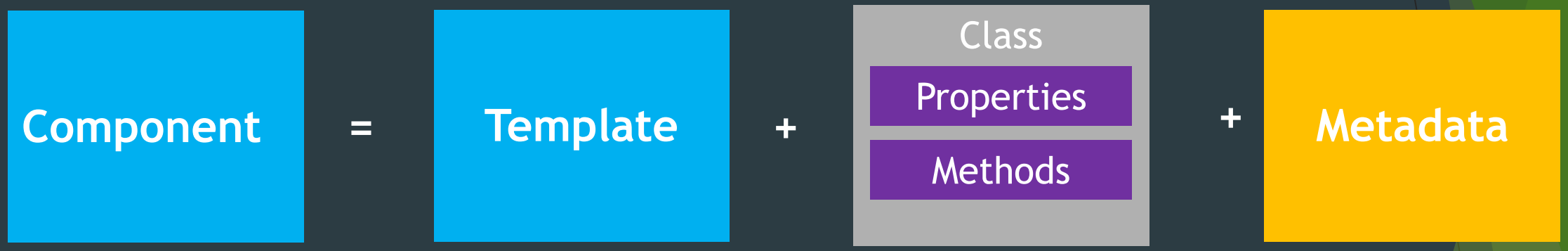
Why Angular?

- Expressive HTML
- Powerful Data binding
- Modular by Design
- Built-in Back-end integration

Structure of Angular Application



What is a Component?



- HTML Layout, Includes binding & directives

- Typescript code supporting View
- Properties representing data
- Methods representing logic

- Extra data for angular
- Defined with decorators

Prerequisites

- **Required**
  - ✓ JavaScript basics
  - ✓ HTML basics
  - ✓ CSS
- **Helpful**
  - ✓ OOP Concepts
  - ✓ C++, C#, JAVA
- **Not Required**
  - ✓ Angular Knowledge
  - ✓ Typescript Knowledge

Typescript

- TypeScript is the programming language we use when building Angular applications
  - ✓ Open-source language
  - ✓ Superset of JavaScript
  - ✓ Transpiles to plain JavaScript
  - ✓ Strongly typed
  - ✓ Class-based object-orientation

What component contains?

- ✓ View layout
- ✓ Created with HTML
- ✓ Includes binding and directives

- ✓ Code supporting the view
- ✓ Created with TypeScript
- ✓ Properties: data
- ✓ Methods: logic

- ✓ Extra data for Angular
- ✓ Defined with a decorator

Service

- Class with focused purpose
  - Independent from any component
  - Provide shared data or logic across component
  - Encapsulate external interactions

How Service Works

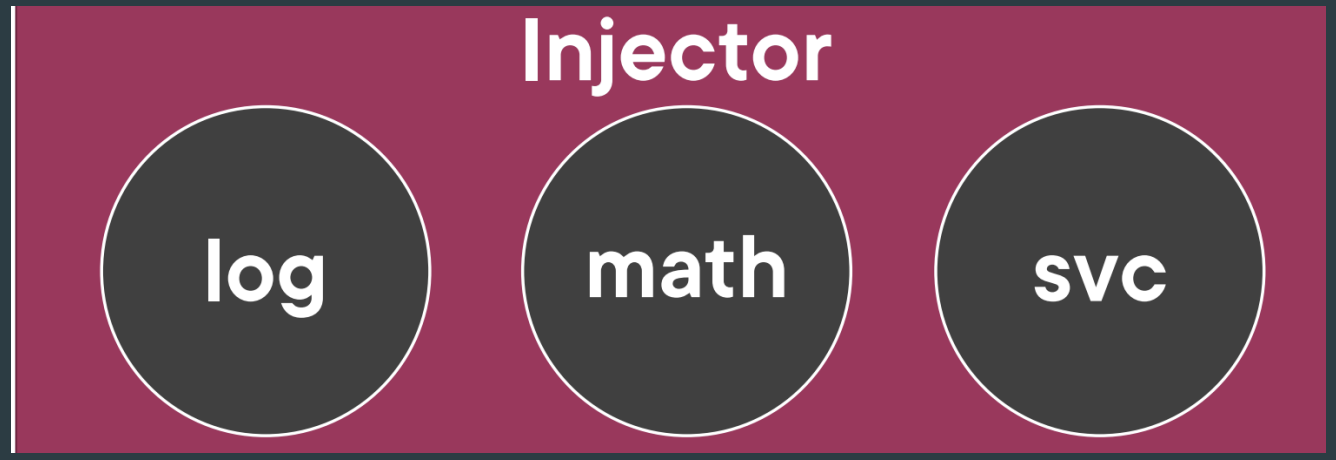
Service

```
export class myService {}
```

Component

```
let svc = new myService();
```

How Service Works cont...



Service

```
export class myService {}
```

Component

```
constructor(private myService) {}
```

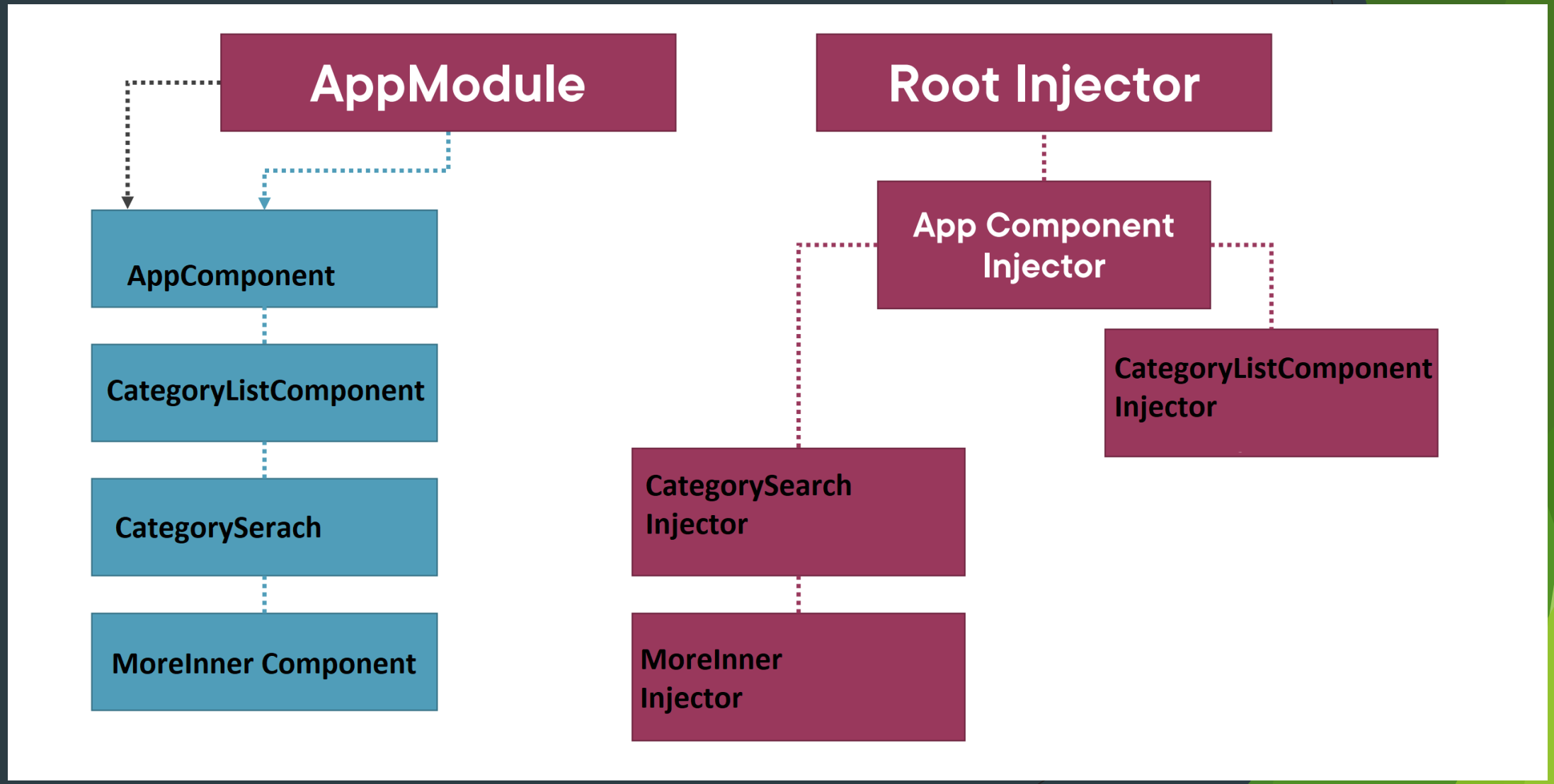
Dependency Injection

- A coding pattern in which a class receives the instances of objects it needs (called dependencies) from an external source rather than creating them itself

Building Service

- Create Service Class
- Define metadata with decorator
- Import what we need

Angular Injectors



Registering Service

Root Injector

Service is available throughout the application

Recommended for most scenarios

Component Injector

Service is available ONLY to that component and its child (nested) components

Isolates a service used by only one component

Provides multiple instances of the service

Injecting Service

product.service.ts

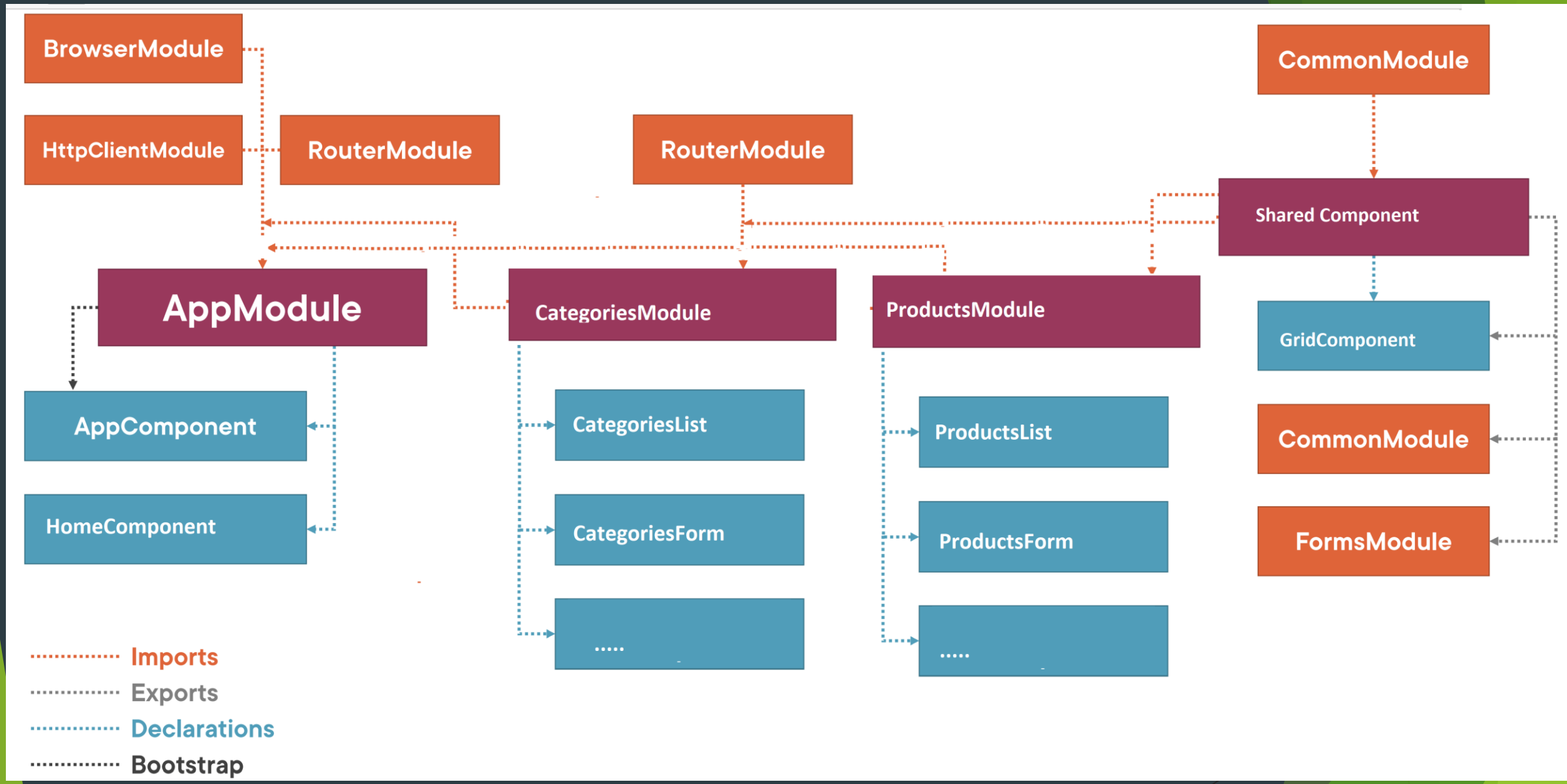
```
@Injectable({
  providedIn: 'root'
})
export class ProductService { }
```

product-list.component.ts

```
@Component({
  templateUrl: './product-list.component.html',
  providers: [ProductService]
})
export class ProductListComponent { }
```

app.module.ts

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


BrowserModule
HttpClientModule
RouterModule
RouterModule
CommonModule
Shared Component
AppModule
CategoriesModule
ProductsModule
GridComponent
AppComponent
HomeComponent
CategoriesList
CategoriesForm
.....
ProductsList
ProductsForm
.....
CommonModule
FormsModule
Imports
Exports
Declarations
Bootstrap

Angular Modules

- A class with an NgModule decorator
- Its purpose:
  - Organize the pieces of our application
  - Arrange them into blocks
  - Extend our application with capabilities from external libraries
  - Provide a template resolution environment
  - Aggregate and re-export

NgModule Decorator

- @NgModule({
  declarations: [CategoriesComponent],
  imports: [CommonModule],
  exports: [CommonModule, FormsModule, CategoriesComponent] })

  bootstrap: [AppComponent] // Only for root module i.e. AppModule
- export class CategoryModule { }

Bootstrap Array

- bootstrap: [AppComponent]
- Every application must bootstrap at least one component, the root application component
- The bootstrap array should only be used in the root application module, AppModule

Declaration Array

- declarations: [AppComponent,
  HomeComponent,
  ProductListComponent,
  ProductFormComponent]
- Every component, directive, and pipe we create must belong to one and only one Angular module
- Only declare components, directives and pipes
- All declared components, directives, and pipes are private by default
- They are only accessible to other components, directives, and pipes declared in the same module
- The Angular module provides the template resolution environment for its component templates.

Export Array

- Export any component, directive, or pipe if other components need it.
- Re-export modules to re-export their components, directives, and pipes.
- We can export something without including it in the imports array

Import Array

- imports: [BrowserModule,
  FormsModule,
  HttpClientModule,
  RouterModule.forRoot([...])]
- Adding a module to the imports array makes available any components, directives, and pipes defined in that module's exports array.
- Only import what this module needs.
- For each component declared in the module, add to the imports array what is needed by the component's template
- Importing a module does NOT provide access to its imported modules
- Use the imports array to register services provided by Angular or third-party modules
- Import the module in the AppModule to ensure its services are registered one time