
Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS
San Diego
Angular 101
Deborah Kurata
Developer | Author | MVP | GDE
Level: Beginner
Code Again for the First Time!
Visual Studio Live!
25
YEARS OF CODING INNOVATION

Deborah Kurata

Developer

Angular v2 Documentation Team Member (angular.io)

Pluralsight Author

- Angular Getting Started
- Angular Reactive Forms
- Angular Routing
- Angular Component Communication
- Angular NgRx: Getting Started
- C# Best Practices

Microsoft Most Valuable Professional (MVP)

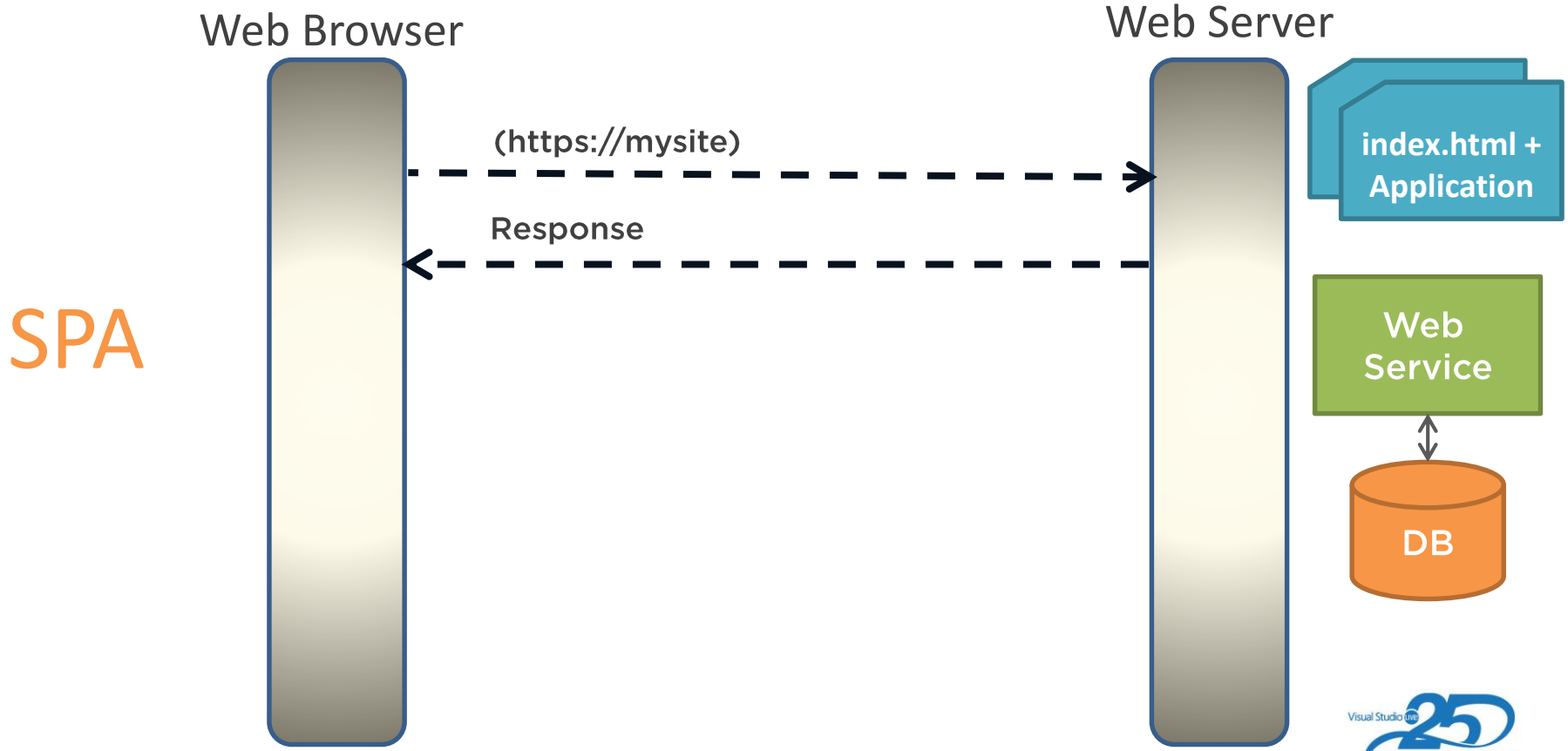
Google Developer Expert (GDE)

@deborahkurata

https://github.com/DeborahK/MovieHunter

Rate Yourself on Angular AngularJS

What is Angular?


Web Browser
Web Server
(https://mysite)
Response
index.html +
Application
Web
Service
DB

SPA

Web Browser

index.html + Application

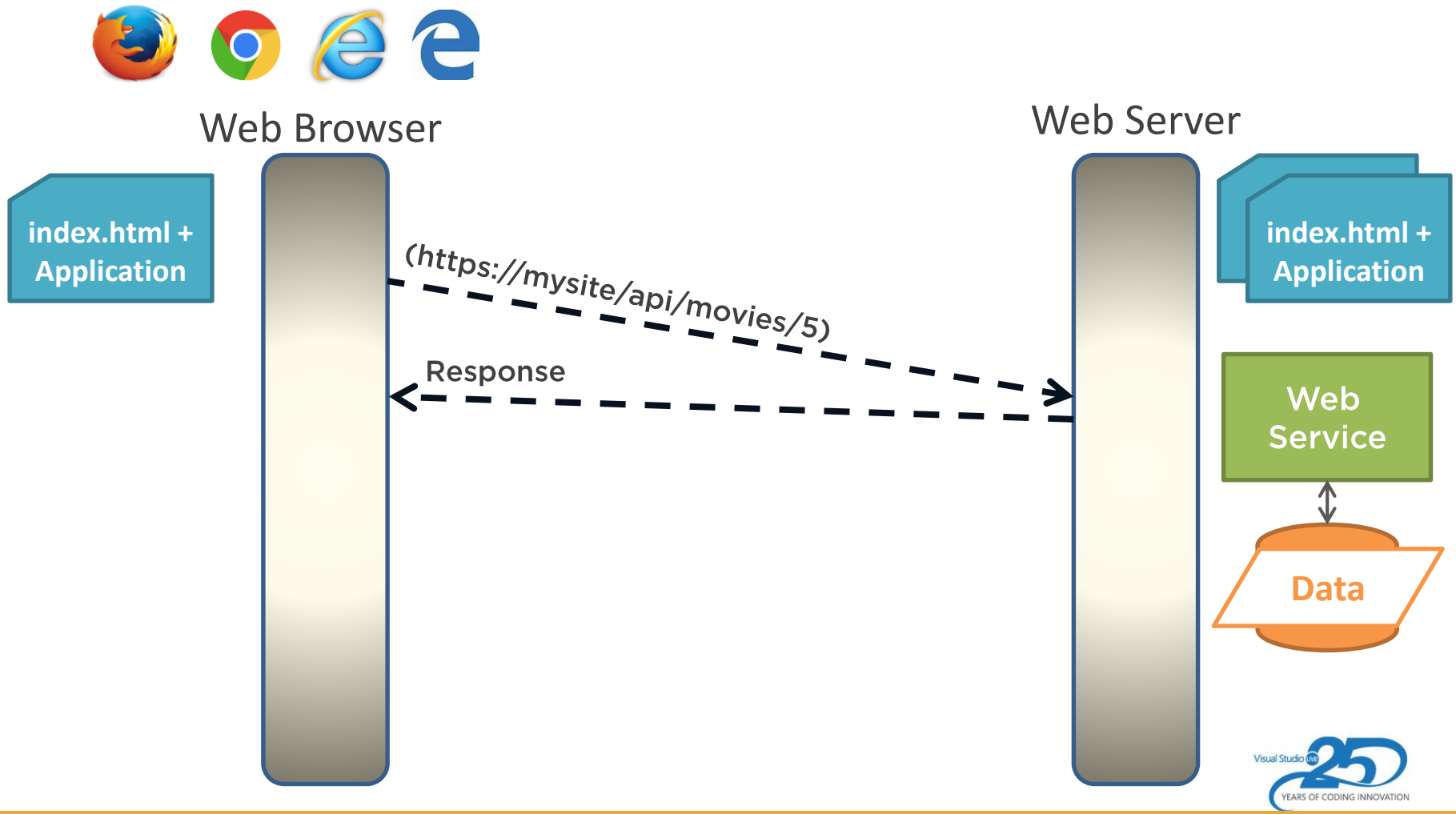
(https://mysite/movieList)

Web Server

index.html + Application

Web Service

DB


Web Browser
Web Server
index.html + Application
(https://mysite/api/movies/5)
Response
index.html + Application
Web Service
Data
Visual Studio Live!
25
YEARS OF CODING INNOVATION

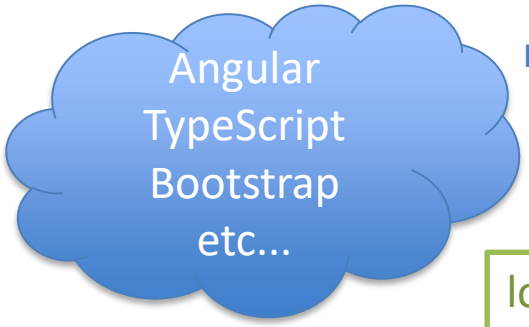
Prerequisites

npm

node package manager

Angular CLI

Angular CLI != Angular



npm repository

Installing the Angular CLI

npm install -g <package>

local machine

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
PS C:\Users\Deborah\MovieHunter> npm install -g @angular/cli
```

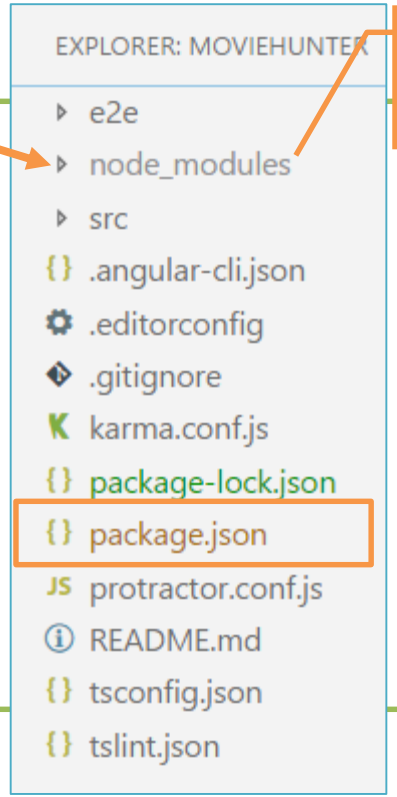
AppData/Roaming/npm

Creating a project with the Angular CLI

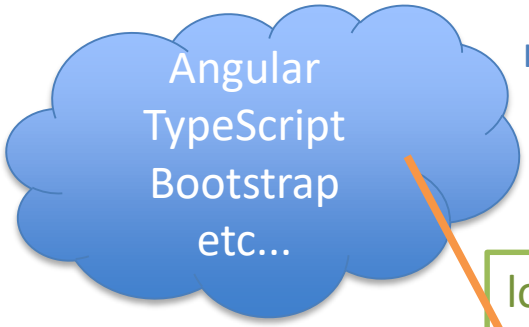
Angular
TypeScript
Bootstrap
etc...

npm repository

local machine



Exclude from source control



npm repository

package.json

local machine

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Deborah\MovieHunter> npm install

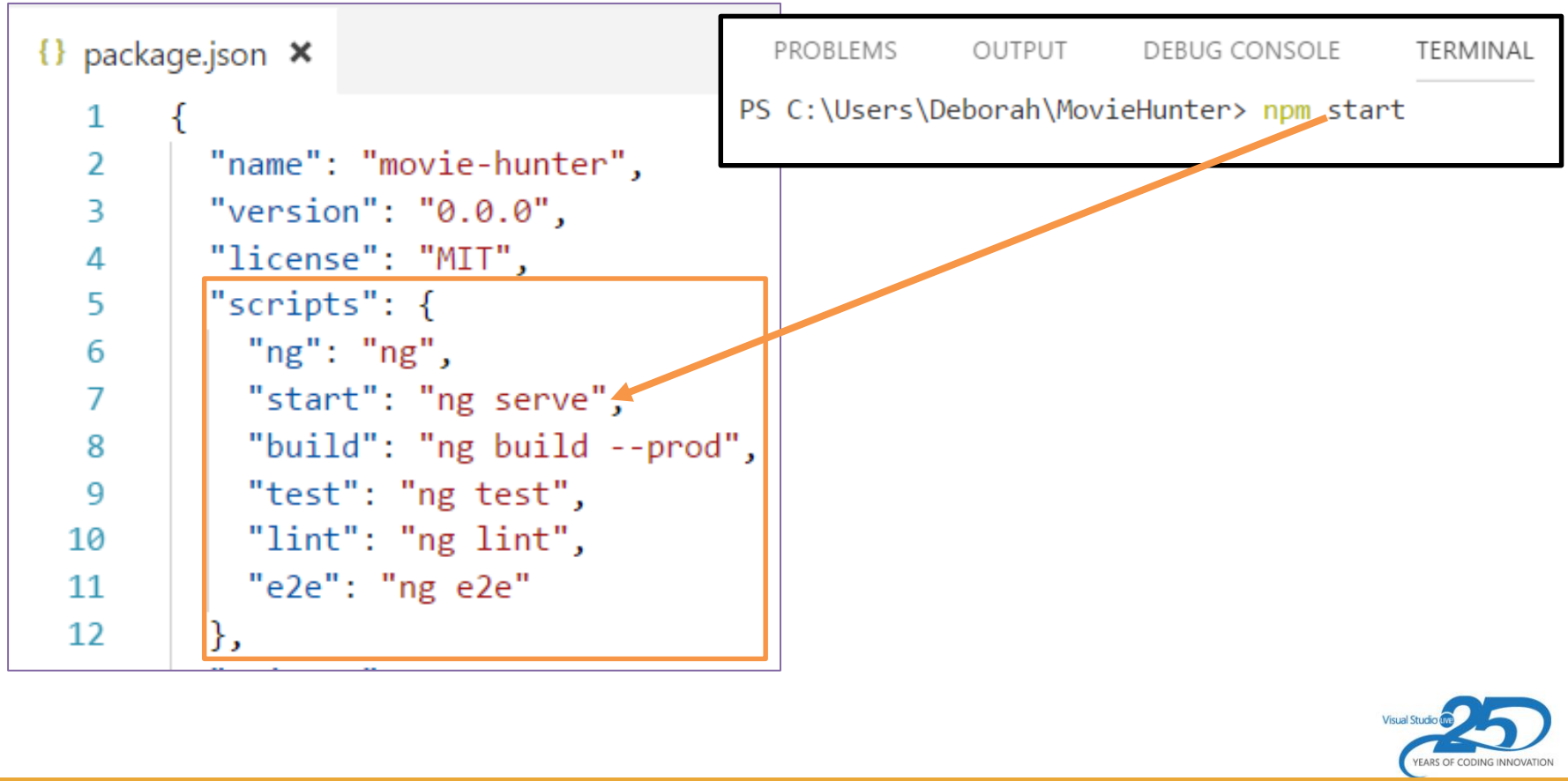
EXPLORER: MOVIEHUNTER

- e2e
- node_modules
- src
- .angular-cli.json
- .editorconfig
- .gitignore
- karma.conf.js
- package-lock.json
- package.json
- protractor.conf.js
- README.md
- tsconfig.json
- tslint.json

package.json

```
29    "devDependencies": {
30      "@angular/cli": "~1.7.3",
31      "@angular/compiler-cli": "^5.2.0",
32      "@angular/language-service": "^5.2.0",
33      "@types/jasmine": "~2.8.3",
34      "@types/jasminewd2": "~2.0.2",
35      "@types/node": "~6.0.60",
36      "codelyzer": "^4.0.1",
37      "jasmine-core": "~2.8.0",
38      "jasmine-spec-reporter": "~4.2.1",
39      "karma": "~2.0.0",
40      "karma-chrome-launcher": "~2.2.0",
41      "karma-coverage-istanbul-reporter": "^1.2.1",
42      "karma-jasmine": "~1.1.0",
43      "karma-jasmine-html-reporter": "^0.2.2",
44      "protractor": "~5.1.2",
45      "ts-node": "~4.1.0",
46      "tslint": "~5.9.1",
47      "typescript": "~2.5.3"
48    }
```

package.json Scripts

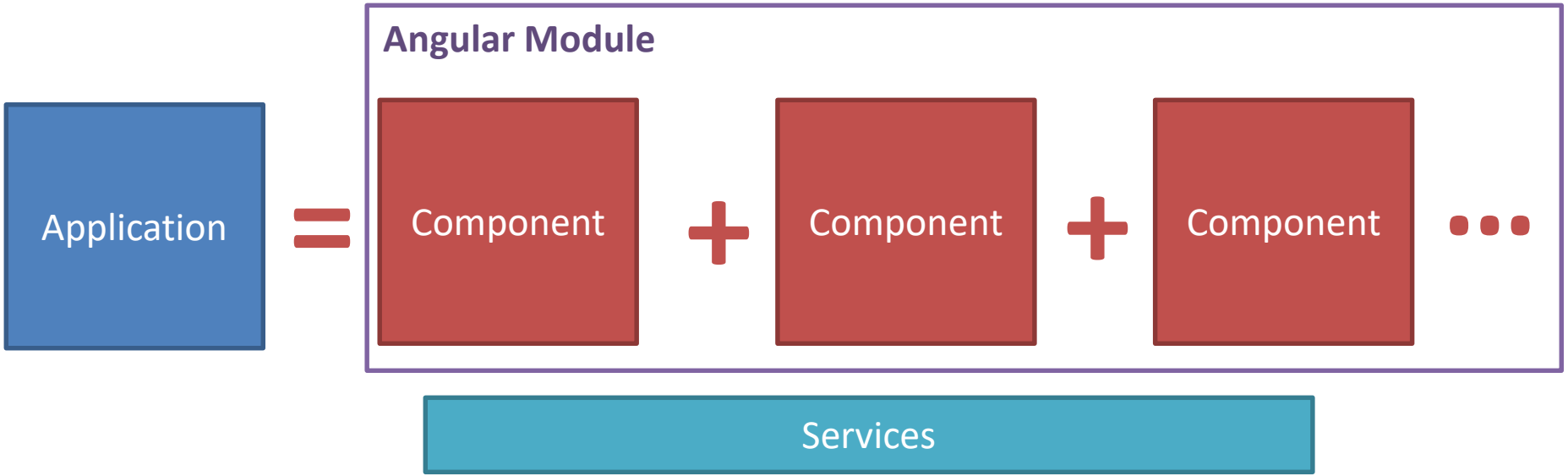


Demo

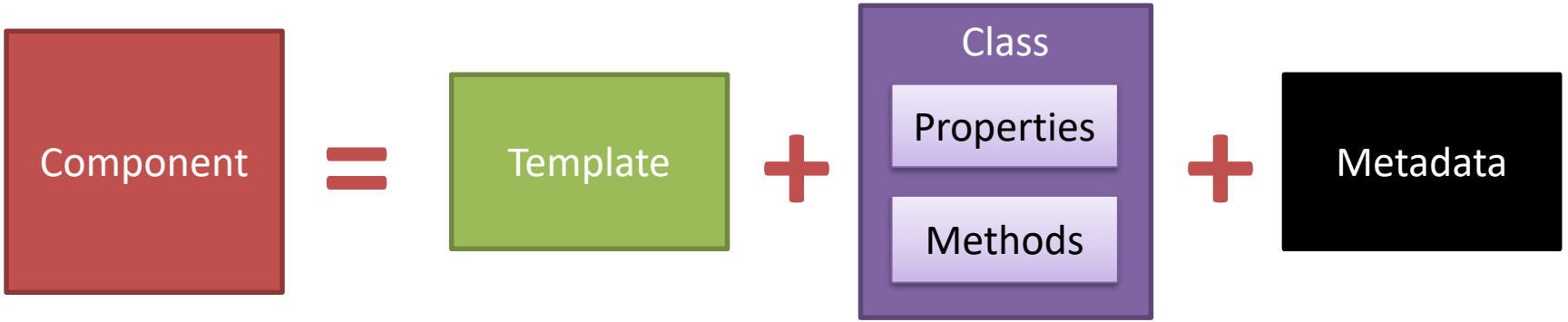
Angular CLI

```
ng new MovieHunter
```

Angular Application



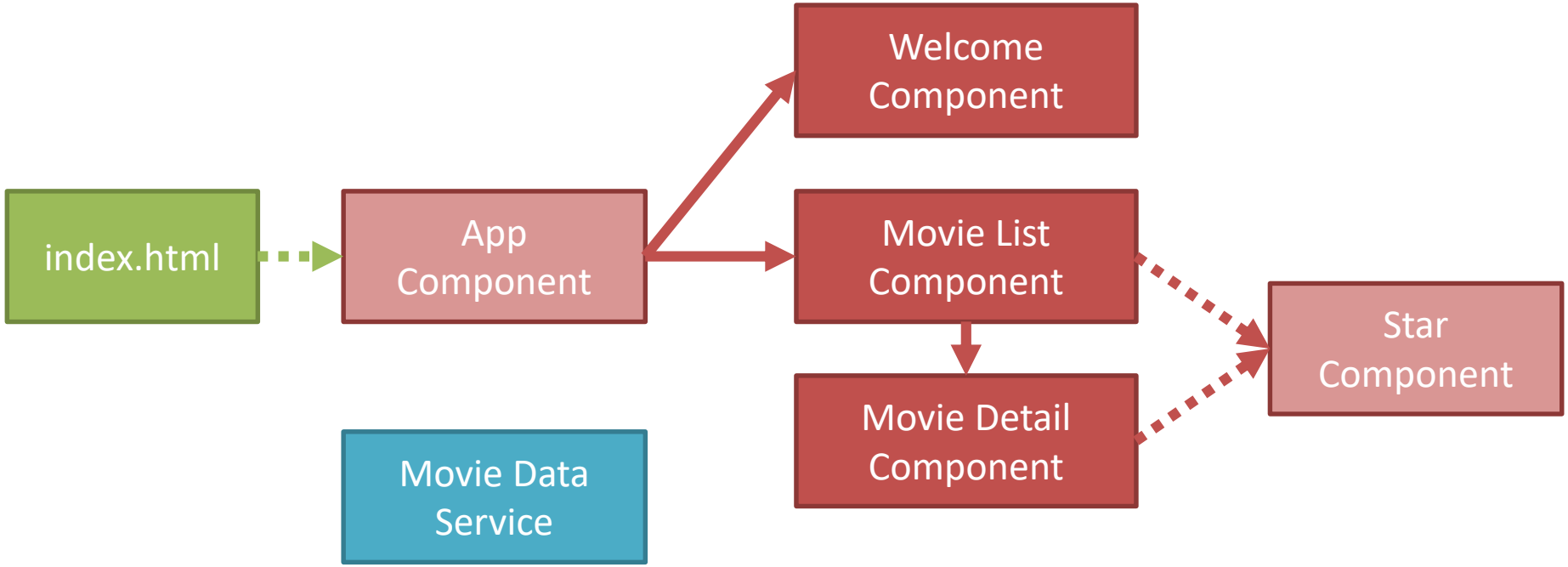
Component



Demo

MovieHunter
Sample Application

Sample Application Architecture



Component

app.component.ts

```
import { Component } from '@angular/core';

@Component({
    selector: 'app-root',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
    pageTitle: string = 'InStep Movie Hunter';
}
```

ES 2015
Import Syntax

Metadata:
Component
Decorator

ES 2015
Class Syntax

Hosting the Application

index.html

```
<body>
  <app-root></app-root>
</body>
```

app.component.ts

```
import { Component } from '@angular/core';

@Component({
    selector: 'app-root',
    template:
    `<div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>`

})
export class AppComponent {
 pageTitle: string = 'InStep Movie Hunter';
}
```

Template

app.component.ts

```
import { Component } from '@angular/core';

@Component({
    selector: 'app-root',
    template: `
    <div><h1>{{pageTitle}}</h1>
        <div>My First Component</div>
    </div>
    `
})
export class AppComponent {
    pageTitle: string = 'InStep Movie Hunter';
}
```

Template

Simple HTML

```
template:
"<h1>{{pageTitle}}</h1>"
```

Multi-line HTML

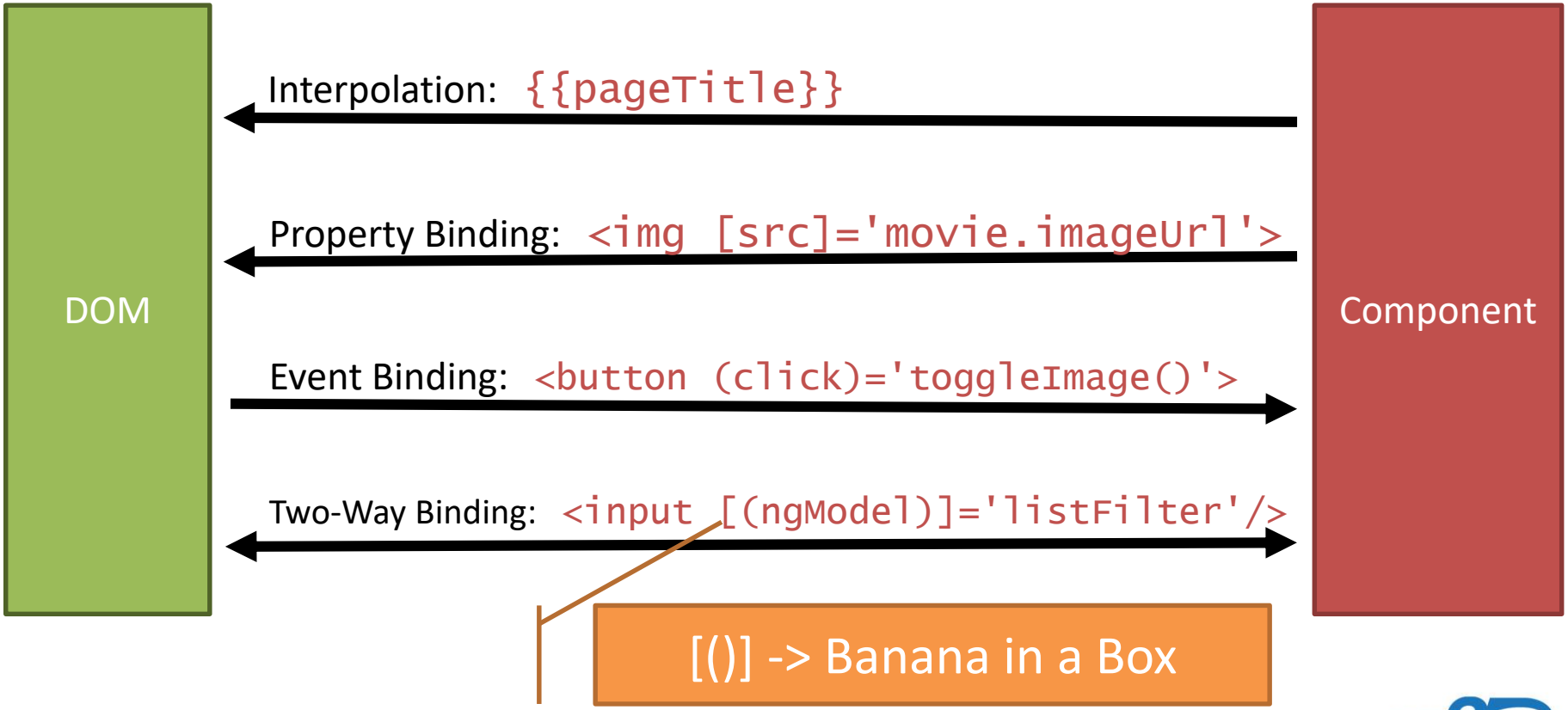
```
template: `
<div>
 <h1>{{pageTitle}}</h1>
 <div>
    My First Component
 </div>
</div>
`
```

ES 2015
Back Ticks
Template Literal

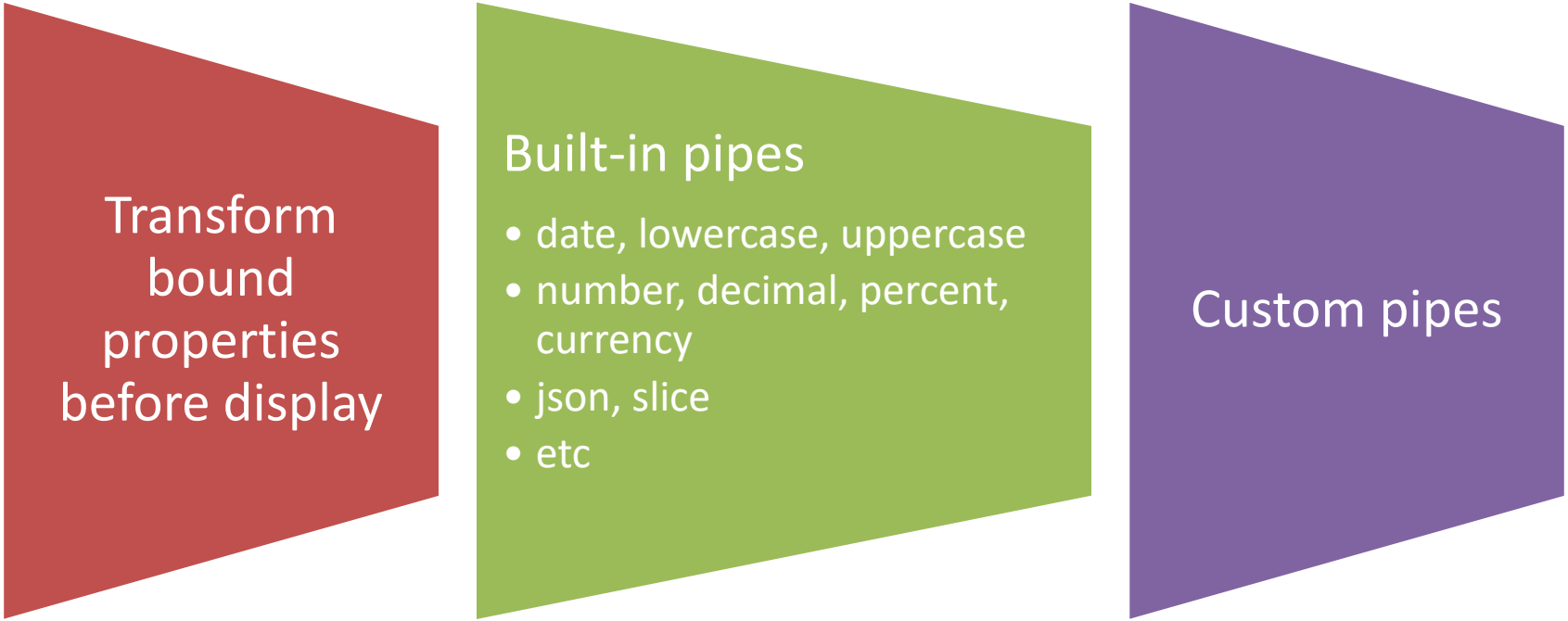
URL

```
templateUrl:
'./movie-list.component.
html'
```

Data Binding



Transforming Data with Pipes




```
{{ movie.mpaa | uppercase }}
```

Directives

Structural Directives

```
<table *ngIf="movies.length">

<tr *ngFor="let movie of movies">
```

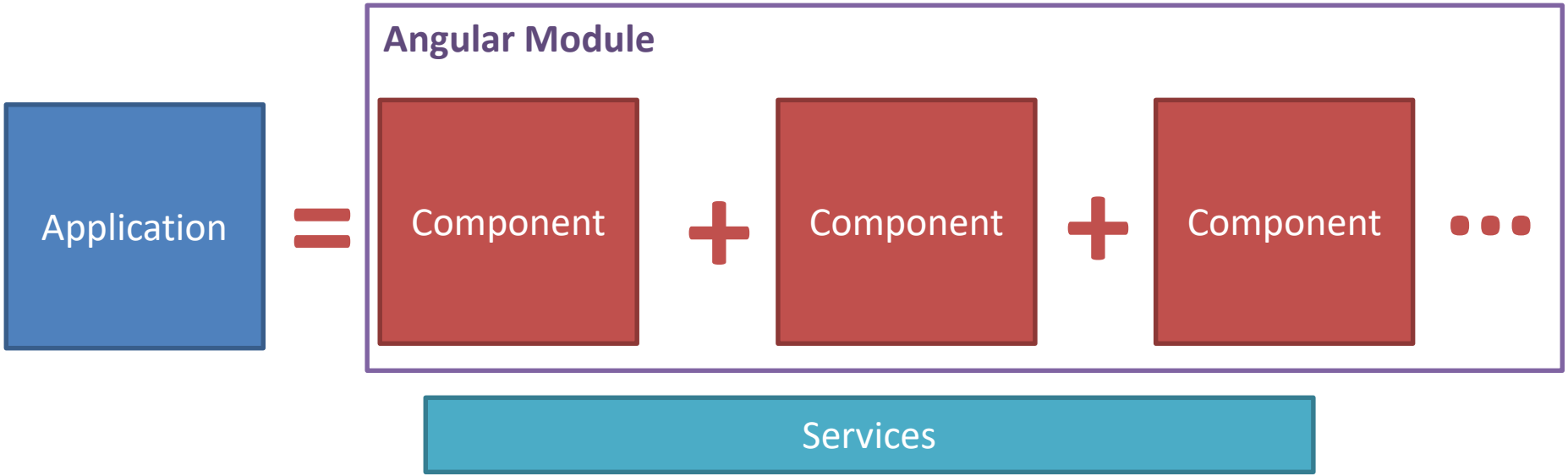
Demo

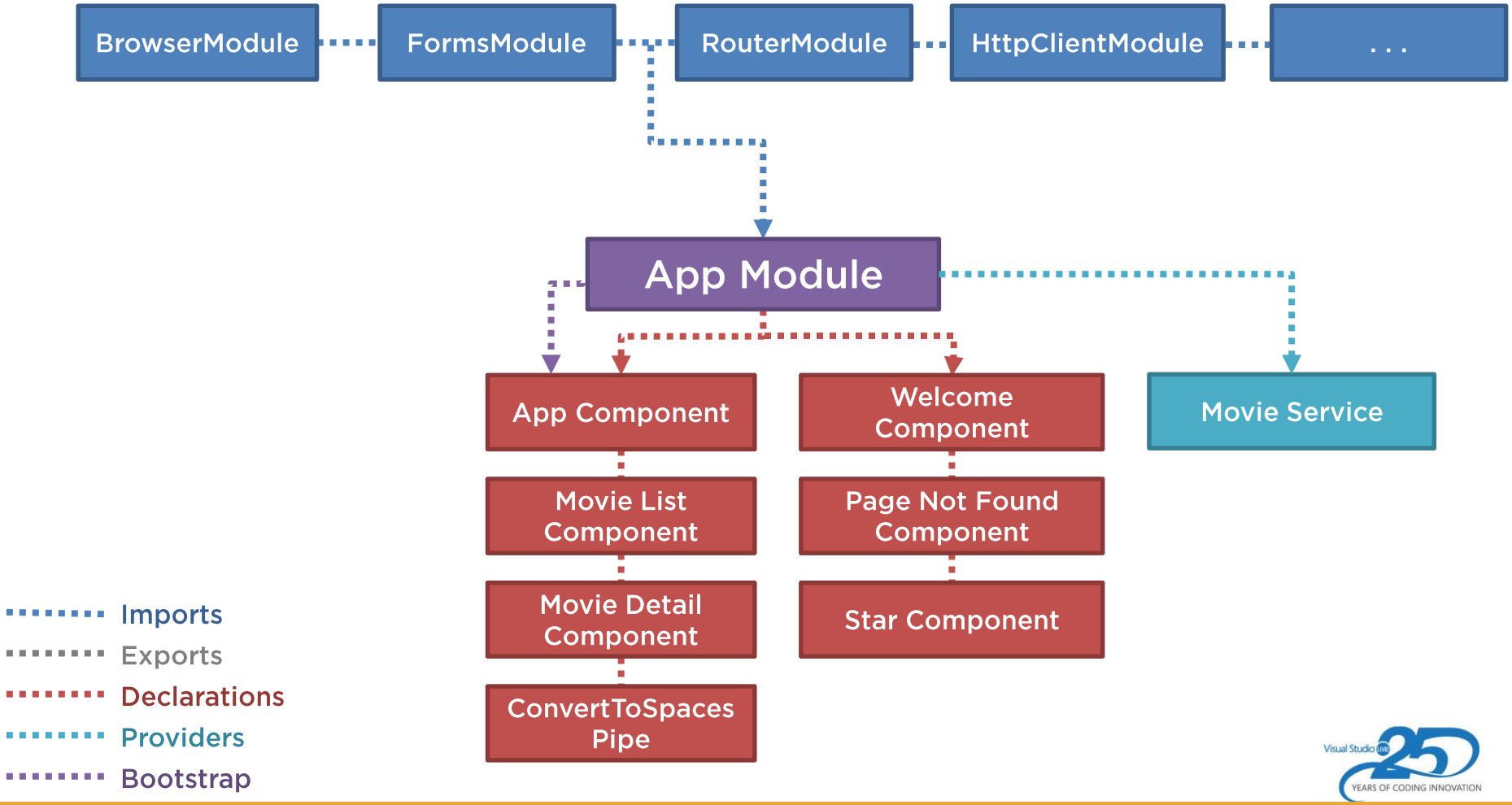
Movie List Template

Component Best Practices

- Put presentation logic in the component class
- Apply the single responsibility principle
- Define one component per file
- Use a separate template file
- Leverage Angular's change detection

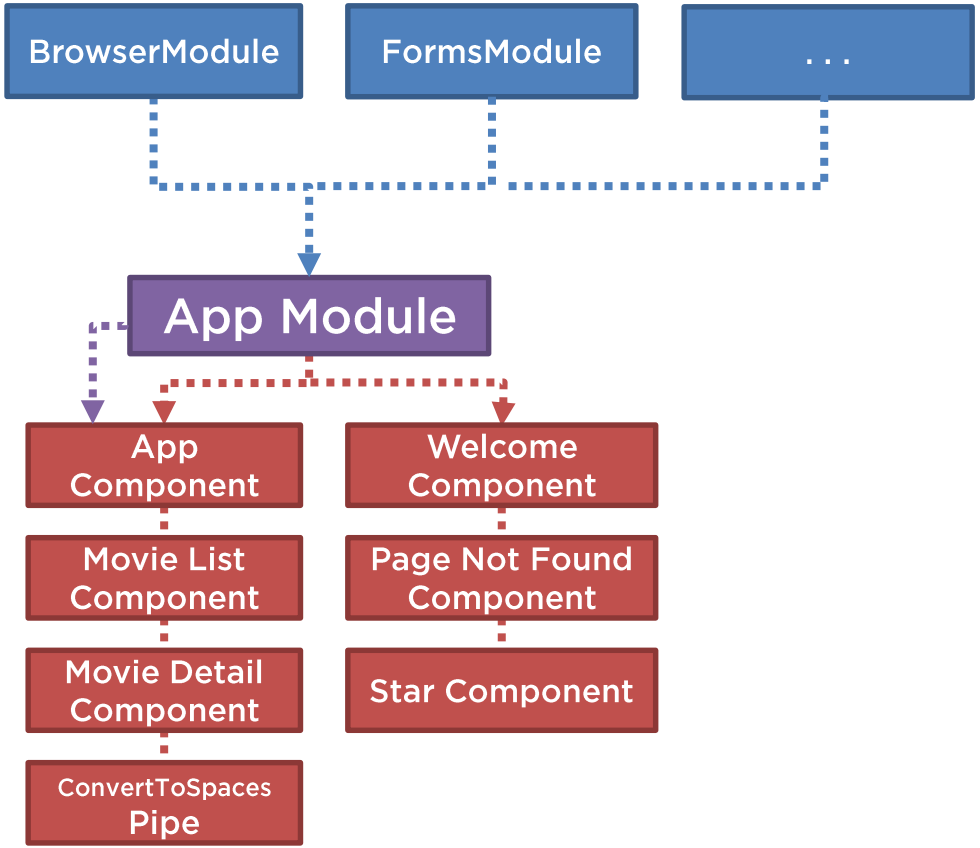
Angular Application

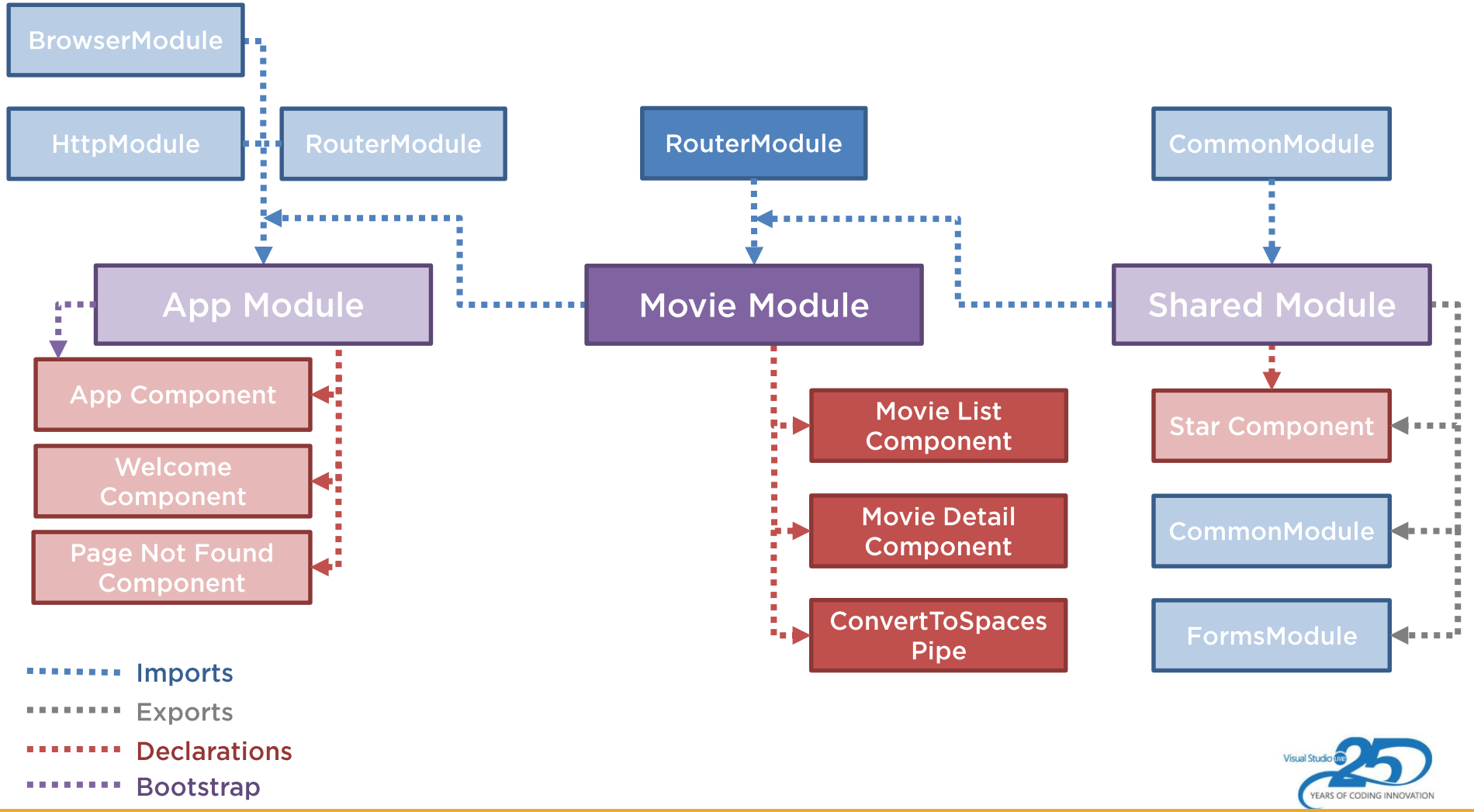


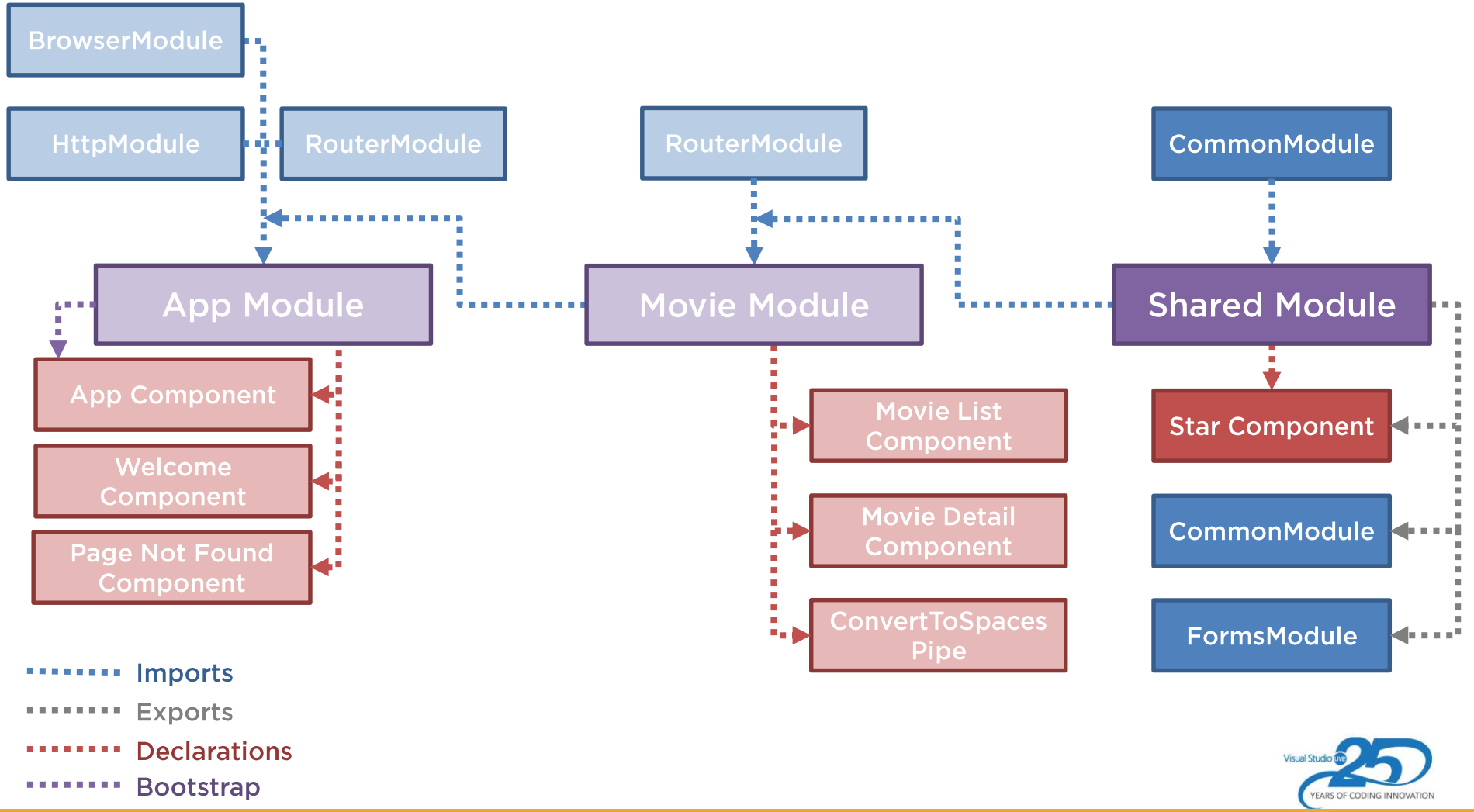

BrowserModule
FormsModule
RouterModule
HttpClientModule
. . .
App Module
App Component
Welcome Component
Movie Service
Movie List Component
Page Not Found Component
Movie Detail Component
Star Component
ConvertToSpaces Pipe
Imports
Exports
Declarations
Providers
Bootstrap

app.module.ts

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



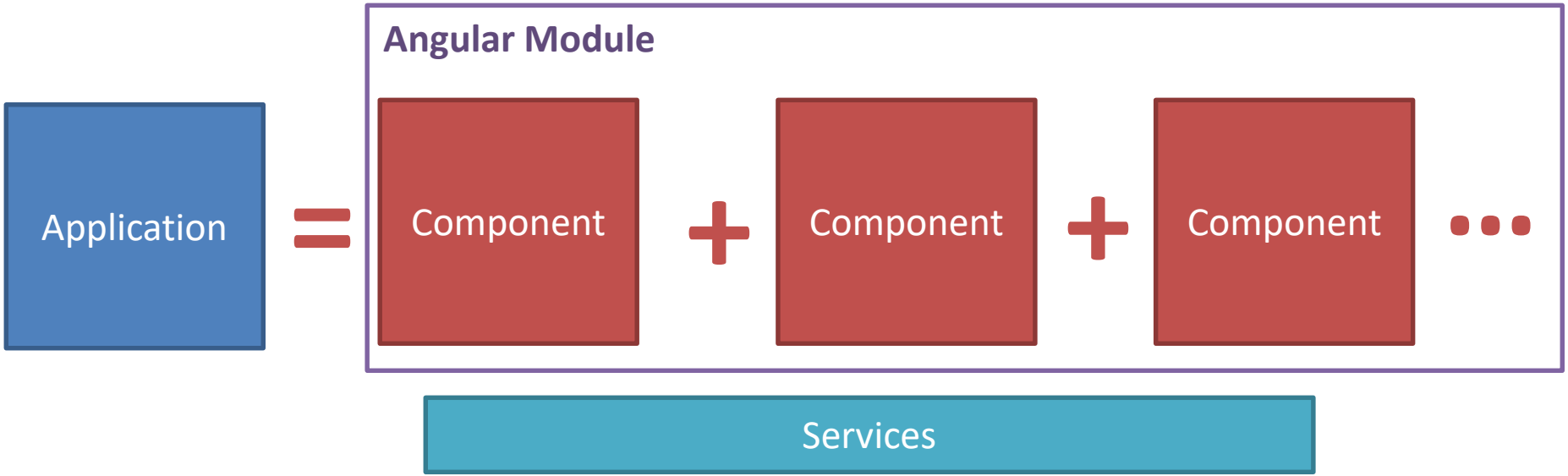

BrowserModule
HttpModule
RouterModule
App Module
App Component
Welcome Component
Page Not Found Component
RouterModule
Movie Module
Movie List Component
Movie Detail Component
ConvertToSpaces Pipe
CommonModule
Shared Module
Star Component
CommonModule
FormsModule
Imports
Exports
Declarations
Bootstrap


BrowserModule
HttpModule
RouterModule
App Module
App Component
Welcome Component
Page Not Found Component
RouterModule
Movie Module
Movie List Component
Movie Detail Component
ConvertToSpaces Pipe
CommonModule
Shared Module
Star Component
CommonModule
FormsModule
Imports
Exports
Declarations
Bootstrap

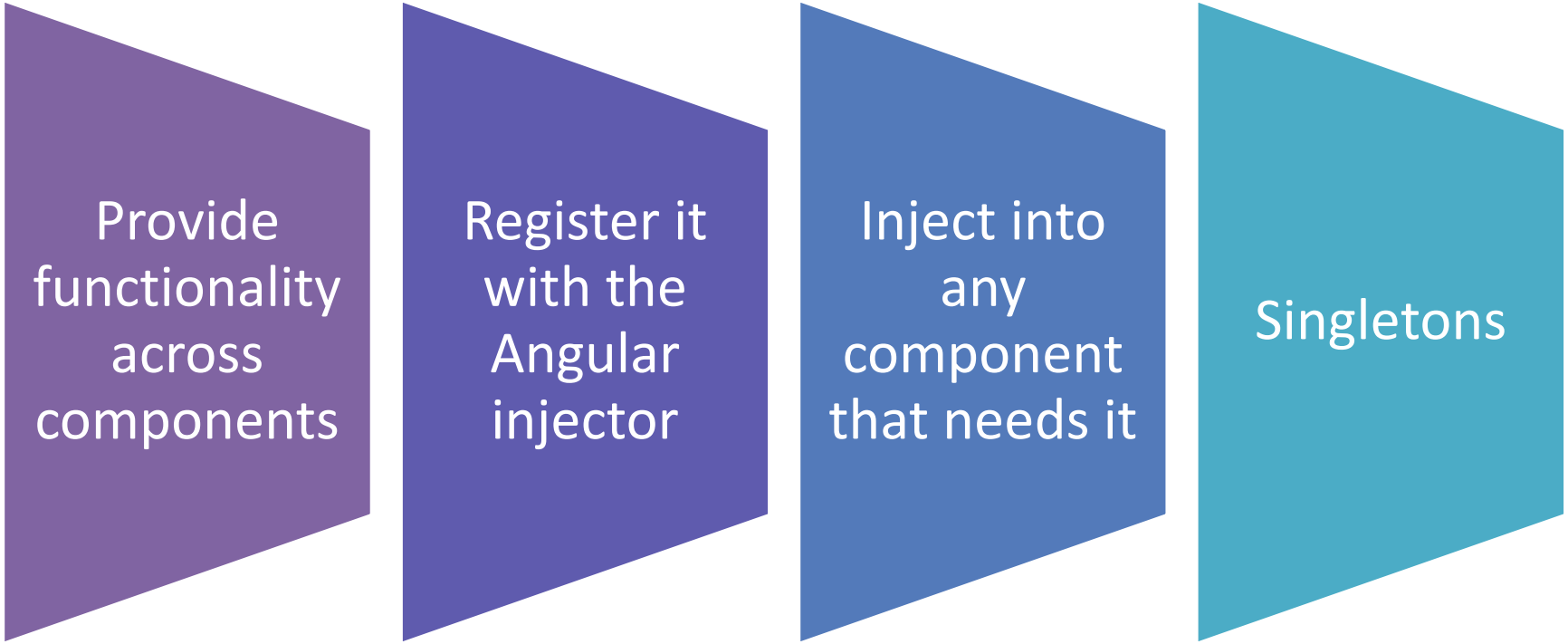
Module Best Practices

- Break your application into features
- Define one module per feature
- Build a shared module for shared pieces
- Don't use `providers[]`

Angular Application



Services



Building a Service

movie.service.ts

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

@Injectable()
export class MovieService {
  getMovies() { }
}
```

Registering a Service

movie.service.ts

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

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

Service Registration Best Practices

- Use `providedIn = 'root'` in the service
  - Registers with the root application injector, tree shaken
- Do **not** use `providers[]` in a module
- Use `providers[]` in a component
  - To limit access to a component and its children
  - For service isolation or multiple instances of the service
- Use `providedIn = 'lazymodule'` in the service
  - To limit access to a particular lazy loaded module
  - Requires an additional module to prevent the circular dependency

Using a Service

movie-list.component.ts

```
import { Component } from '@angular/core';
import { MovieService } from './movie.service';

@Component({
  templateURL: './movie-list.component.html'
})
export class MovieListComponent {
 constructor(private movieService: MovieService) { }
}
```

movie.service.ts

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

@Injectable()
export class MovieService {
  getMovies() { }
}
```

Demo

Movie Parameter Service

Service Best Practices

- Build a service for
  - Shared data
  - Reusable operations
  - Complex logic
  - Cross cutting functionality
  - Data access

Data Access: Build an Interface

movie.ts

```
export interface IMovie {
    id: number;
    approvalRating: number;
    description: string;
    director: string;
    imageurl: string;
    mpaa: string;
    price: number;
    releaseDate: string;
    starRating: number;
    title: string;
}
```

Data Access: Build a Service

movie.service.ts

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

import { IMovie } from './movie';
```

```
@Injectable({
 providedIn: 'root'
})
export class MovieService {
 private moviesUrl = './api/movies/movies.json';
 ...
}
```

Data Access: Inject HttpClient

movie.service.ts

```
import { HttpClient } from '@angular/common/http';
...

export class MovieService {
 private moviesUrl = './api/movies/movies.json';

 constructor(private http: HttpClient) { }

 ...

}
```

Data Access: Call http.get

movie.service.ts

```
getMovies(): Observable<IMovie[]> {
  return this.http.get<IMovie[]>(this.moviesUrl)
    .pipe(
      tap(data => console.log(JSON.stringify(data))),
      catchError(this.handleError)
    );
}
```

Returns RxJS

Http returns a response mapped to an Observable

Ta
str

Catches any errors

Pipes the stream through a set of operators

Data Access: Subscribe (Component)

movie-list.component.ts

```
movies: IMovie[];

constructor(private movieService: MovieService) { }

ngOnInit(): void {
  this.movieService.getMovies()
    .subscribe(
      (movies: IMovie[]) => this.movies = movies,
      (error: any) => this.errorMessage = <any>error);
    );
}
```

Data Access: Subscribe (Component)

movie-list.component.ts

```
movies: IMovie[];

constructor(private movieService: MovieService) { }

ngOnInit(): void {
  this.movieService.getMovies()
    .subscribe(
      (movies: IMovie[]) => this.movies = movies,
      (error: any) => this.errorMessage = <any>error);
    );
  console.log(this.movies);
}
```

Data Access: Subscribe (Component)

movie-list.component.ts

```
movies: IMovie[];

constructor(private movieService: MovieService) { }

ngOnInit(): void {
  this.movieService.getMovies()
    .subscribe(
      (movies: IMovie[]) => {
           this.movies = movies;
           console.log(this.movies);
       },
      (error: any) => this.errorMessage = <any>error);
    );
}
```

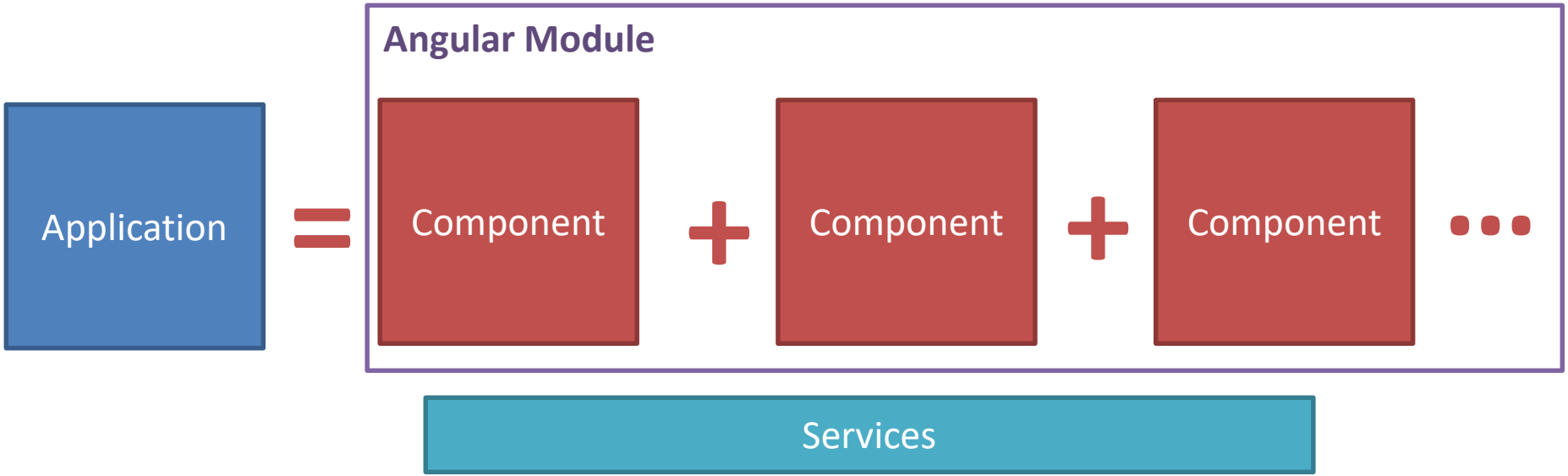
Demo

Movie Data Service

Data Access Best Practices

- Build interfaces to define your types
- Encapsulate data access in a service
- Return an Observable
- Subscribe as close as possible to the UI

Angular Application



Angular CLI

- Command line tool for generating, testing, and deploying an Angular application

```
ng new MovieHunter
ng generate component movie-list
ng build --prod
...
```

Demo

Angular CLI

Thank You!

@deborahkurata

https://github.com/DeborahK/MovieHunter

http://bit.ly/Angular-GettingStarted