Angular 2: A Comparison

Deborah Kurata
Developer | Author | Consultant
InStep Technologies, Inc

Level: Intermediate

Deborah Kurata

- Developer
- Angular 2 Documentation Team Member
- Pluralsight Author
  - Angular 2: Getting Started
  - Angular with TypeScript
  - Angular Front to Back with Web API
  - C# Best Practices
- Microsoft MVP

https://github.com/DeborahK/VSLive2016-LV-2VS1

Rate Yourself on Angular 2

- Tried out Angular 2 - Limited proficiency
- New to Angular 2 - Angular 1 proficiency
- New to Angular - no proficiency

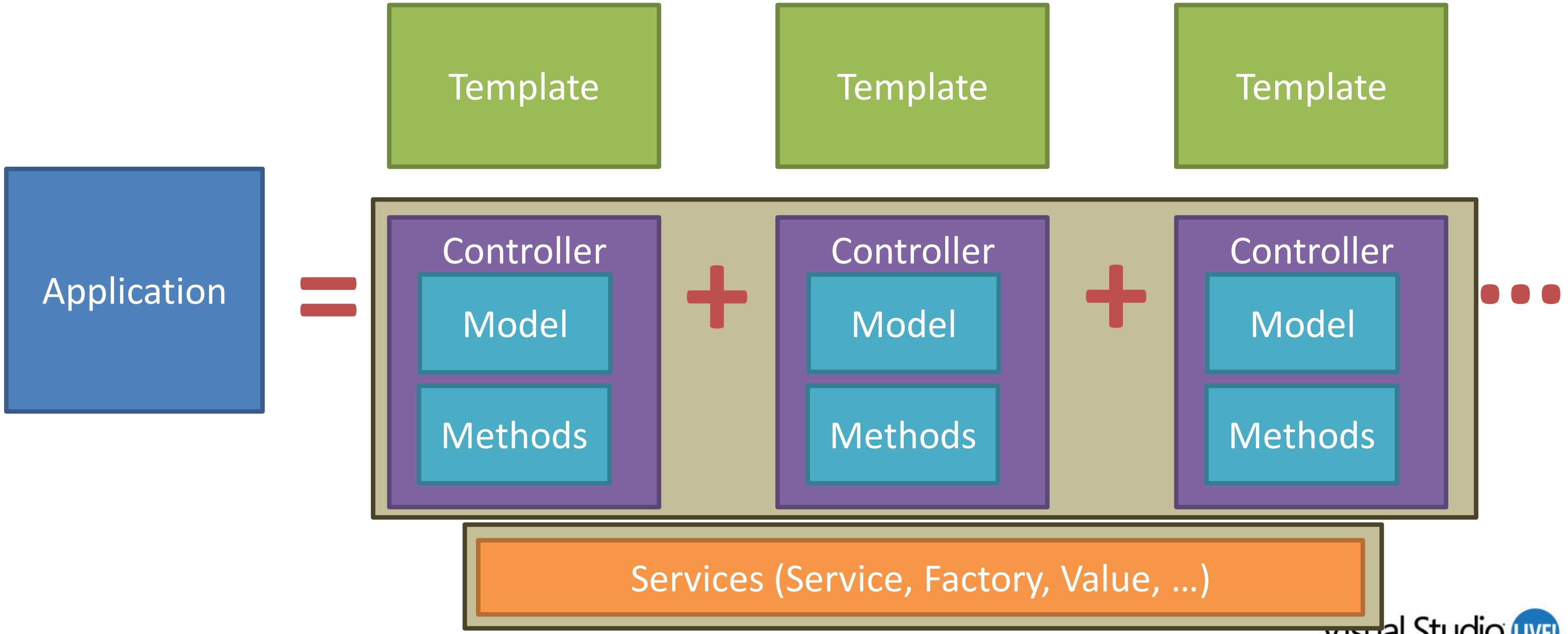
- TypeScript*
- ES 2015
- SystemJS
- Reactive Extensions (RxJS)
- …

10 KEY CONCEPT DIFFERENCES BETWEEN ANGULAR 1 AND ANGULAR 2

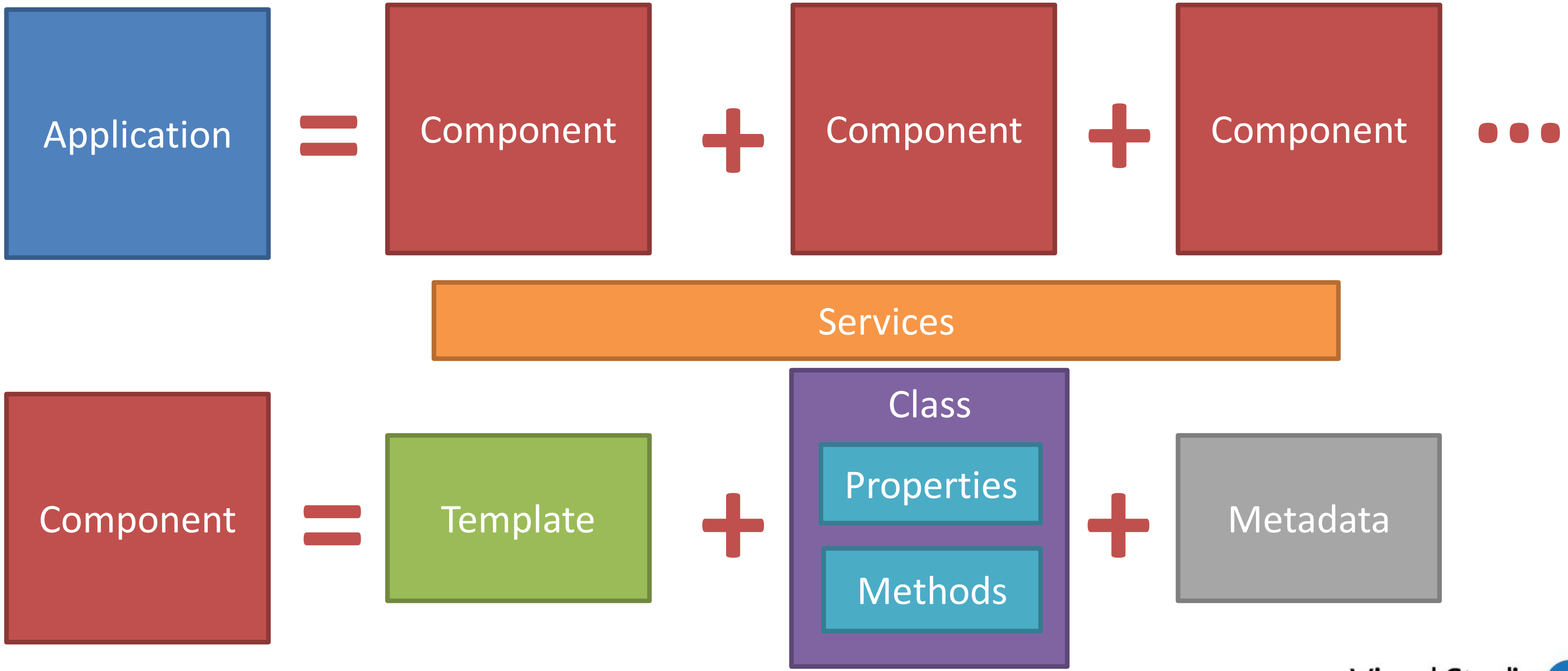
Demo

Movie Hunter Sample Application

Angular 1 Application



Angular 2 Application



1 Angular Module => ES 2015 Module

Angular 1

app.js

```
(function () {
  angular
    .module("movieHunter",
            ["common.services"]);
}());
```

Angular 2

app.component.ts

movie-list.component.ts

movie.service.ts

module == file

file == module

One or more definitions in a file

Only exported definitions can be imported

2

Controllers => Components

Angular 1

movieListView.html
```
<div ng-controller=
                "MovieListCtrl as vm">
```

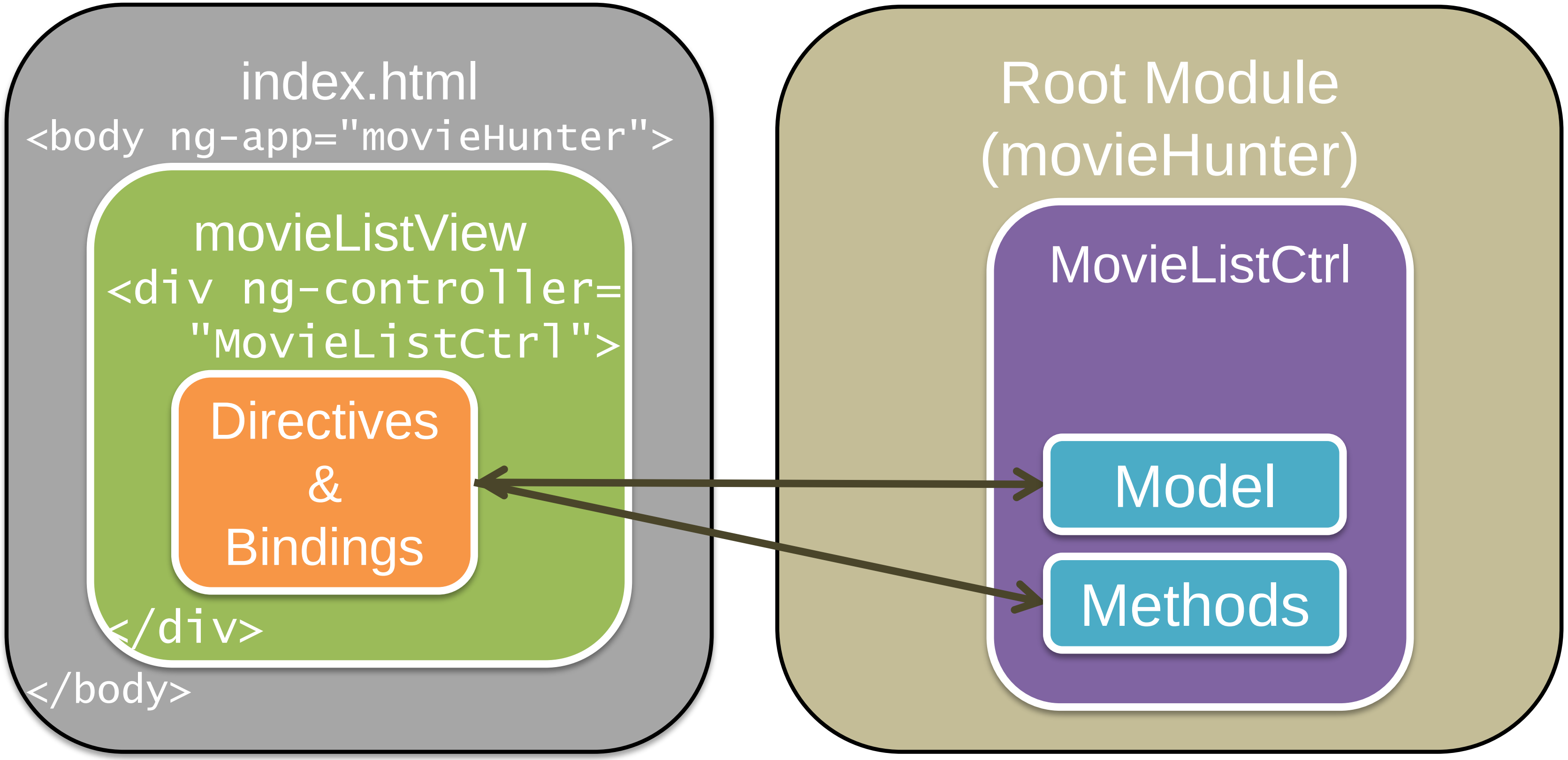
movieListCtrl.js
```
(function () {
  angular
    .module("movieHunter")
    .controller("MovieListCtrl",
                ["movieResource",
                 MovieListCtrl]);

  function MovieListCtrl(movieResource)
  { }
}());
```

Angular 2

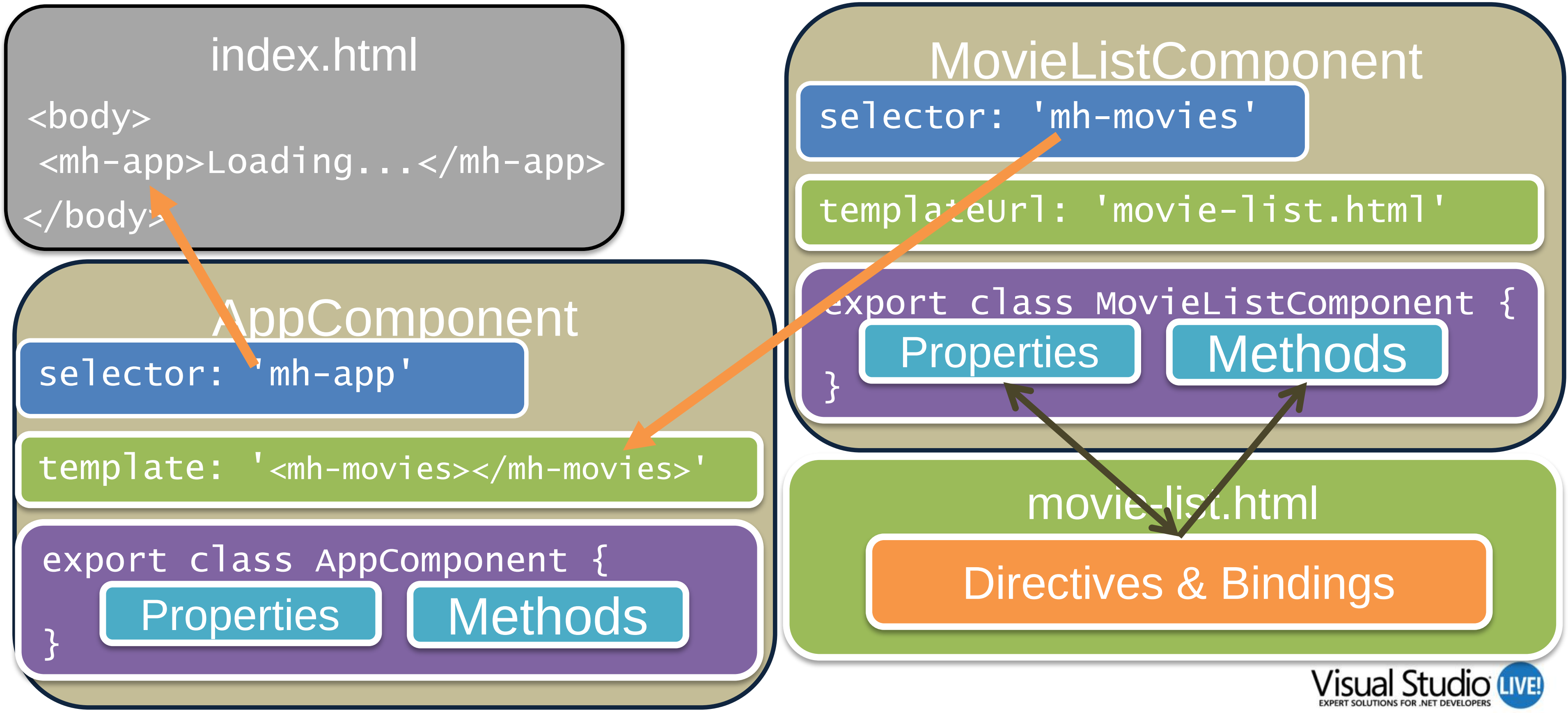
movie-list.component.ts
```
import {Component} from 'angular2/core';

@Component({
    selector: 'mh-movies',
    templateUrl:
        'app/movies/movie-list.component.html'
})
export class MovieListComponent
{ }
```

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Angular 1: Modules & Controllers



Angular 2: Components



index.html

```
<body>
 <mh-app>Loading...</mh-app>
</body>
```

AppComponent

```
selector: 'mh-app'
template: '<mh-movies></mh-movies>'
export class AppComponent {
```

```
}
```

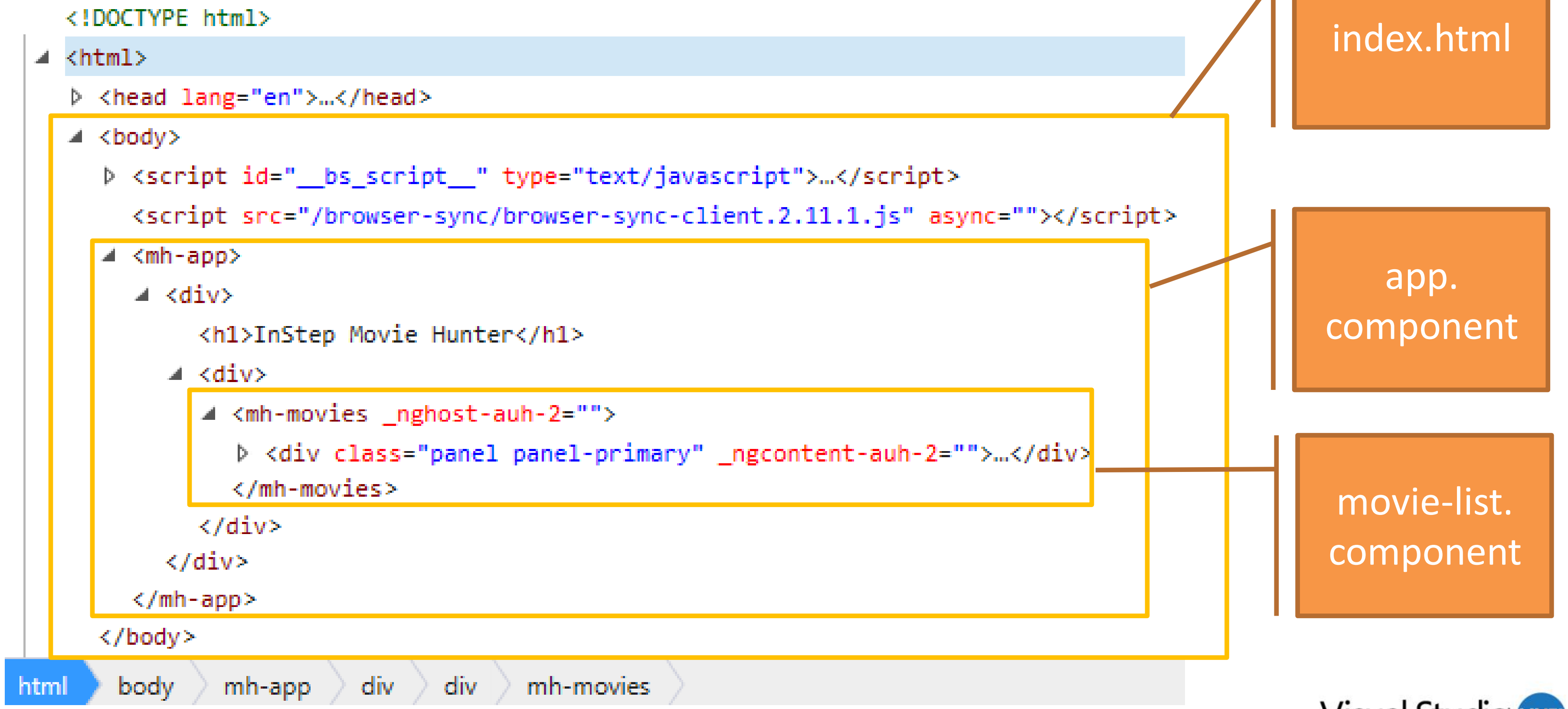
MovieListComponent

```
selector: 'mh-movies'
templateUrl: 'movie-list.html'
export class MovieListComponent {
```

```
}
```

movie-list.html

Resulting DOM



```
<!DOCTYPE html>
<html>
  <head lang="en">…</head>
  <body>
    <script id="__bs_script__" type="text/javascript">…</script>
    <script src="/browser-sync/browser-sync-client.2.11.1.js" async=""></script>
    <mh-app>
      <div>
        <h1>InStep Movie Hunter</h1>
        <div>
          <mh-movies _nghost-auh-2="">
            <div class="panel panel-primary" _ngcontent-auh-2="">…</div>
          </mh-movies>
        </div>
      </div>
    </mh-app>
  </body>
```

Bootstrapping

To pull oneself up by one's bootstraps: Better oneself by one's own unaided efforts.

A self-starting process that proceeds without external input. An entry point for an application.

3

Bootstrapping

systemJS

Imports the main file

Angular 1

index.html

```
<body ng-app="movieHunter">

  <h1>InStep Movie Hunter</h1>
  <div>
    <ng-include
        src="'movieListView.html'">
    </ng-include>
  </div>

  <script src="angular.js"></script>

  <script src="app.js"></script>
  <script src="movieListCtrl.js">
  </script>
</body>
```

Angular 2

index.html

```
<head>
<script>
    System.config({
    packages: {
      app: {
        format: 'register ,
        defaultExtension: '
      }
    }
  });
  System.import('app/main')
        .then(null, console.error.bind(console));
</script>
</head>
<body>
  <mh-app>Loading...</mh-app>
</body>
```

Angular 2 Application Startup

index.html

main.ts
(bootstrapper)

app.component.ts

```
System.import('app/main');
```

```
<body>
    <mh-app>
     Loading...
   </mh-app>
</body>
```

```
import {bootstrap}
 from 'angular2/
   platform/browser';

import {AppComponent}
 from './app.component';

bootstrap(AppComponent);
```

```
@Component({
 selector: 'mh-app',
 template: `
 <div>{{pageTitle}}</div>
 `
})
export class AppComponent
{ }
```

Demo

Bootstrapping the Application

4

Templates

Angular 1

External file

Identified associated controller

```
<div ng-controller=
            "MovieListCtrl as vm">

  <h1>{{vm.pageTitle}}</h1>
  <div>
    <div>Filter by:</div>
    ...
  </div>
</div>
```

Angular 2

External file or inline

Defined within the component

```
@Component({
 selector: 'mh-app',
 template: `
   <div>
    <h1>{{pageTitle}}</h1>
    <div>
      <mh-movies></mh-movies>
    </div>
  </div>
 `
})
```

Templates

Simple HTML

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

Multi-line HTML

```
template: `
<div>
 <h1>{{pageTitle}}</h1>
 <div>
 <mh-movies></mh-movies>
 </div>
</div>
`
```

ES 2015
back ticks

URL

```
templateUrl:
       "movie-list.html"
```

5

Binding

Angular 1

Interpolation (one way)

ng-model (two way)

```
<h1>{{vm.pageTitle}}</h1>
<input ng-model="listFilter"/>
<img ng-show="vm.showImage"
 ng-src="{{movie.imageurl}}">
<button ng-click="vm.toggleImage()">
```

Angular 2

Interpolation (one way)

ngModel (two way)

Property binding (one way)

Event binding (one way)

```
<h1>{{pageTitle}}</h1>
<input [(ngModel)]="listFilter"/>
<img [hidden]=
     [src]="mo
<button (click
```

[()]
Banana in a Box

6

Structural Directives

Angular 1

```
<table ng-if="vm.movies.length">

<tr ng-repeat="movie in vm.movies>
```

Angular 2

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

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

#
Local Variable

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

7

Filters => Pipes

Angular 1	Angular 2
`{{ movie.mpaa | uppercase}}`	`{{ movie.mpaa | uppercase}}`
`{{ movie.price | currency:'USD$':2}}`	`{{ movie.price | currency:'USD':true:'1.2'}}`
`{{ movie.approvalRating | number: '1.2'}}`	`{{ movie.approvalRating | percent: '1.0-0'}}`
`<tr ng-repeat="movie in vm.movies | orderBy : 'title'">`	NADA

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Demo

Binding and Structural Directives

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

8

Composition

Angular 1

Build custom directives

Angular 2

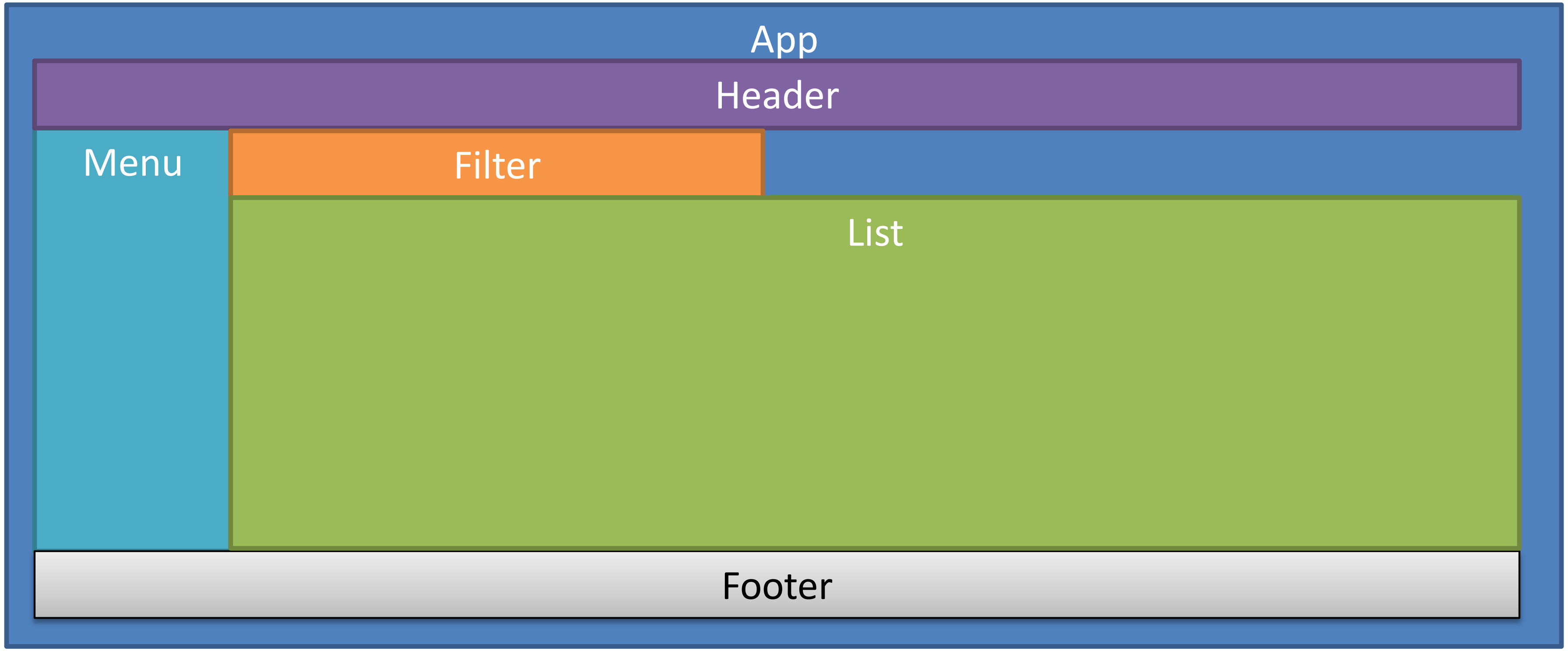
Build components

Template can reference any number of components, forming a component tree

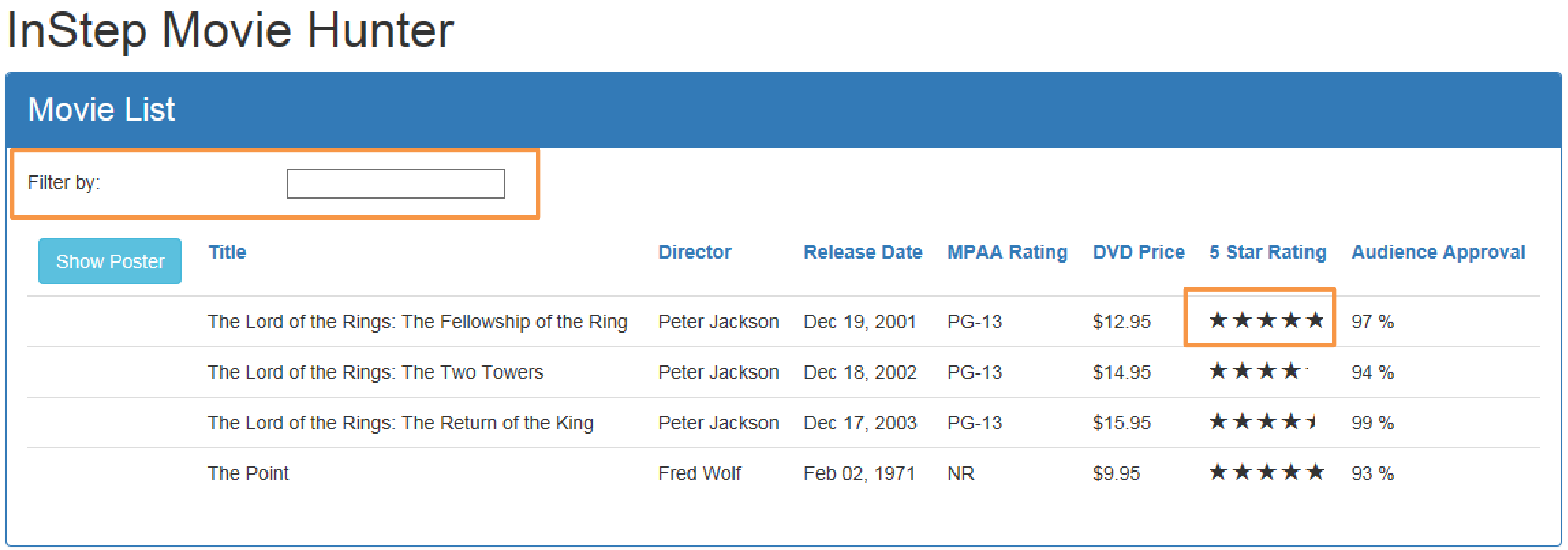
`@Input` decorator to flow data into the child component

`@Output` decorator to flow events out back to the parent component

Composition



Composition



InStep Movie Hunter

Movie List

Filter by:

Show Poster	Title	Director	Release Date	MPAA Rating	DVD Price	5 Star Rating	Audience Approval
	The Lord of the Rings: The Fellowship of the Ring	Peter Jackson	Dec 19, 2001	PG-13	$12.95	★★★★★	97 %
	The Lord of the Rings: The Two Towers	Peter Jackson	Dec 18, 2002	PG-13	$14.95	★★★★	94 %
	The Lord of the Rings: The Return of the King	Peter Jackson	Dec 17, 2003	PG-13	$15.95	★★★★½	99 %
	The Point	Fred Wolf	Feb 02, 1971	NR	$9.95	★★★★★	93 %

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

@Input

```
import {Component, Input, OnChanges } from 'angular2/core';

@Component({
    selector: 'is-star',
    templateUrl: 'star.component.html',
    styleUrls: ['star.component.css']
})
export class StarComponent implements OnChanges {
    @Input() rating: number;
    starWidth: string;

    ngOnChanges() { ... }
}
```

```
<is-star [rating]="movie.starRating"></is-star>
```

@Output

```
import {Component, Output, EventEmitter}  from 'angular2/core';

@Component({
    selector: 'is-filter-entry',
    templateUrl: 'filter-entry.component.html',
})
export class FilterEntryComponent {
    filterText: string = "";
    @Output() filterChanged : EventEmitter<string> =
                                       new EventEmitter<string>();
    onChanged() {
        this.filterChanged.emit(this.filterText);
    }
}
```

```
<is-filter-entry (filterChanged)="onFilterChanged($event)">
</is-filter-entry>
```

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Using Child Components

```
import {Component} from 'angular2/core';

import {FilterEntryComponent} from '../shared/filter-entry.component';
import {StarComponent} from "../shared/star.component";

@Component({
    selector: 'mh-movies',
    templateUrl: 'app/movies/movie-list.component.html',
    styleUrls: ['app/movies/movie-list.component.css'],
    directives: [FilterEntryComponent,
                 StarComponent]
})
```

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Demo

Composition

9

Services

Angular 1

```
(function () {
    angular
        .module("common.services")
        .factory("movieResource",
                ["$resource",
                 movieResource]);

    function movieResource($resource) {
        ...
    }
}());
```

Angular 2

```
import {Injectable} from 'angular2/core';
import {IMovie} from './movie';

@Injectable()
export class MovieService {
    constructor() { }

    getMovies() : IMovie[] {
        . . .
    }
}
```

10 Dependency Injection

Angular 1

```
(function () {
  angular
    .module("movieHunter")
    .controller("MovieListCtrl",
                ["movieResource",
                  MovieListCtrl]);

  function MovieListCtrl(movieResource) {
        var vm = this;
        vm.movies = [];

        movieResource.query(
            ...
            vm.movies = ...
        );
    }
}());
```

Angular 2

```
import {Component, OnInit}  from 'angular2/core';
import {IMovie} from './movie';
import {MovieService} from "./movie.service";

@Component({
  selector: 'mh-movies',
  templateUrl: 'movie-list.component.html',
  providers: [MovieService]
})
export class MovieListComponent
              implements OnInit {
  movies: IMovie[];

  constructor(private _movieSvc: MovieService) {
  }

  ngOnInit() {
   this.movies = this._movieSvc.getMovies();
  }
}
```

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

Demo

Services

Take Aways

- Modules
- Controllers => Components
- Bootstrapping
- Templates
- Binding
- Structural Directives
- Filters => Pipes
- Composition
- Services
- Dependency Injection

Thank You!

@deborahkurata

deborahk@insteptech.com

http://blogs.msmvps.com/deborahk

https://github.com/DeborahK/VSLive2016-LV-2VS1