NG-UPGRADE

Ori Calvo, 2017

oric@trainologic.com

https://trainologic.com

Upgrading Strategies

- Big bang
- Incremental
- Multi Tab/IFrame
- Modern AngularJS
- Don’t touch

Preparations

- The AngularJS side does not require any preparations ☺
- However, running Angular code requires
  - Module loader
  - Typescript
  - Polyfills
- Only certain type of directives can be reused inside the Angular side
  - ng-controller is out of the game ☹

Starting Point


```
<body>
  <div ng-controller="AppCtrl as $ctrl">
    <h1>Hello AngularJS</h1>

    <ul>
      <li ng-repeat="contact in $ctrl.contacts">
        <span>{{contact.name}}</span>
      </li>
    </ul>
  </div>

  <script src="node_modules/angular/angular.js"></script>
  <script src="app.module.js"></script>
  <script src="app.ctrl.js"></script>
</body>
```

```
const appModule = angular.module("myApp", []);
```

```
class AppCtrl {
  contacts: Contact[];

  constructor() {
    this.contacts = [
      {"id": 1, "name": "Ori"},
      {"id": 2, "name": "Roni"},
      {"id": 3, "name": "Udi"},
      {"id": 4, "name": "Tommy"},
    ];
  }
}

interface Contact {
 id: number;
 name: string;
}

appModule.controller("AppCtrl", AppCtrl);
```

Install Angular

- @angular/core
- @angular/platform-browser
- @angular/platform-browser-dynamic
- @angular/common
- @angular/compiler
- @angular/upgrade
- rxjs
- zone.js
- reflect-metadata

tsconfig.json

- Enable commonjs modules
- Enable decorator support

```
{
 "compilerOptions": {
  "module": "commonjs",
  "target": "es5",
  "sourceMap": true,
  "lib": ["dom", "es2015"],
  "experimentalDecorators": true,
  "emitDecoratorMetadata": true
 },
 "exclude": [
  "node_modules"
 ]
}
```

Bootstrapping with Angular

- Remove ng-app and use the following

```
import {NgModule} from '@angular/core';
import {BrowserModule} from '@angular/platform-browser';
import {UpgradeModule} from '@angular/upgrade/static';
import {platformBrowserDynamic} from '@angular/platform-browser-dynamic';

@NgModule({
  imports: [
    BrowserModule,
    UpgradeModule,
  ],
})
export class AppModule {
  constructor(private upgrade: UpgradeModule) {
  }

  ngDoBootstrap() {
    this.upgrade.bootstrap(document.body, ['myApp'], { strictDi: true });
  }
}

platformBrowserDynamic().bootstrapModule(AppModule);
```

Module Loader - Webpack


```
const path = require('path');

module.exports = {
   devtool: 'inline-source-map',
   entry: './main.ts',
   resolve: {  extensions: ['.ts', '.tsx', '.js']  },
   module: {
      rules: [
         {
            test: /\.ts$/,
            use: [{
                  loader: 'ts-loader',
                  options: {
                     compilerOptions: {
                        noEmit: false,
                     }
                  }
               },
               {  loader: "angular2-template-loader"  }]
         },
         {  test: /\.html$/, use: 'raw-loader'  }
      ]
   },
   output: {
      filename: 'bundle.js',
      path: path.resolve(__dirname, 'dist')
   }
};
```

Polyfills & Bundle


```
<head>
   <meta charset="UTF-8">
   <title>Title</title>

   <script src="node_modules/reflect-metadata/Reflect.js"></script>
   <script src="node_modules/zone.js/dist/zone.js"></script>
</head>
```

```
<body>
   <script src="node_modules/angular/angular.js"></script>
   <script src="app.module.js"></script>
   <script src="app.ctrl.js"></script>

   <script src="dist/bundle.js"></script>
</body>
```

Adding Angular Component


```
<div ng-controller="AppCtrl as $ctrl">
  <h1>Hello AngularJS</h1>

  <app-new-contact (on-added)="$ctrl.onAdded($event)"></app-new-contact>

  <ul>
    <li ng-repeat="contact in $ctrl.contacts">
      <span>{{contact.name}}</span>
    </li>
  </ul>
</div>
```

Angular Component

Note the unsual kebab-case

Implementation


```
import {Component, EventEmitter, Output} from "@angular/core";

@Component({
  selector: "app-new-contact",
  templateUrl: "./newContact.component.html",
})
export class NewContactComponent {
  name: string;

  @Output() onAdded: EventEmitter<Contact> = new EventEmitter<Contact>();

  constructor() {
  }

  add() {
    const contact: Contact = {
      id: -1,
      name: this.name,
    };

    this.onAdded.emit(contact);
  }
}
```

```
<input [(ngModel)]="name">
<button (click)="add()">Add</button>
```

Register the Component

- Must add to the declarations section
- Also need to add to the entryComponents

```
@NgModule({
  imports: [
    BrowserModule,
    UpgradeModule,
    FormsModule,
  ],
  declarations: [
    NewContactComponent,
  ],
  entryComponents: [
    NewContactComponent,
  ]
})
```

Downgrading a Component

- Bu default Angular component is not visible to the AngularJS part
- We need to upgrade the component manually

```
appModule.directive('appNewContact', downgradeComponent({
  component: NewContactComponent }) as angular.IDirectiveFactory);
```

Upgrading a Service


```
class AppService {
}

appModule.service("appService", AppService);
```

```
const AppServiceProvider = {
   provide: AppService,
   useFactory: function($injector) {
      return $injector.get("appService");
   },
   deps: ['$injector']
};
```

```
@NgModule({
   providers: [
      AppServiceProvider,
   ]
})
export class AppModule {
}
```

Downgrade a Service


```
@Injectable()
export class AppService {
  contacts: Contact[];

  constructor() {
    this.contacts = [
      {"id": 1, "name": "Ori"},
      {"id": 2, "name": "Roni"},
      {"id": 3, "name": "Udi"},
      {"id": 4, "name": "Tommy"},
    ];
  }

  add(contact: Contact) {
    this.contacts.push(contact);
  }
}
```

```
appModule
  .factory('appService', downgradeInjectable(AppService));
```

```
class AppCtrl {
  static $inject = ["appService"];

  constructor(private appService) {
  }

  onAdded(contact) {
    this.appService.add(contact);
  }
}
```

Upgrading a Component

- Only the following directive can be upgraded
  - restrict: 'E'
  - scope: {}
  - bindToController: {}
  - controller & controllerAs
  - template or templateUrl
- The following is restricted
  - compile
  - replace

AngularJS 1.5 component API

- To allow easier definition of AngularJS that is upgradable you should use the component API

```
class ContactIndexComponent {
   onRefresh: ()=>void;

   constructor() {
   }

   refresh() {
      this.onRefresh();
   }
}
```

```
appModule.component("appContactIndex", {
   controller: ContactIndexComponent,
   template: `<div>
      <button ng-click="$ctrl.refresh()">Refresh</button>
   </div>`,
   bindings: {
      "onRefresh": "&",
   }
});
```

UpgradeComponent class


```
@Directive({
   selector: 'app-contact-index'
})
export class ContactIndexUpgraded extends UpgradeComponent {
   @Output() onRefresh: EventEmitter<any> = new EventEmitter();

   constructor(elementRef: ElementRef, injector: Injector) {
      super('appContactIndex', elementRef, injector);
   }
}
```

```
<app-contact-index (onRefresh)="onRefresh()"></app-contact-index>
```

Routing

- No special support
- The trick is to use both routers
- Let each router handle only its own routes
- Let each router has its own placeholder
  - ng-view
  - router-outlet

Summary

- Running both AngularJS and Angular in the same application is challenging
- “Well written” AngularJS directives can be upgraded
- Angular component can be downgraded
- Services can be upgraded/downgraded