From Gulp to Angular CLI and beyond

Angular Athens Meetup - Aristeidis Bampakos

@abampakos

Who am I

- Front End Web Developer **@Plexscape**.
- Early adopter of AngularJS (2012).
- Active **@github** (Angular projects, map visualization libraries).
- Blogging **@medium**.
- Gaming console enthusiast, novice farmer, free diving.

Introduction

- Real world problem.
- Licensing application in AngularJS (codebase ~ 3000 LOC).
- Official upgrade guide with SystemJS.
- First beta versions of Angular CLI.
- Focus on tooling.
- Move closer to the Angular ecosystem.

Agenda

- Review the AngularJS application.
- Integrate with the CLI.
- Introduce Typescript.
- Bootstrap in hybrid mode.
- Upgrade artifacts from bottom to top (services, components, router).
- Remove AngularJS completely.

Step 0: AngularJS application

- AngularJS - Angular Material - Gulp - Bower
- Built with AngularJS style guide by John Papa.
- Inspired by modular (ng-demos) and gulp-patterns.
- Lite-server for development.
- Marvel API

Step 1: Move into Angular CLI

- Install Angular CLI and copy files from a blank application.
- Remove gulp build tasks (except **templatecache**).
- Remove **lite-server** configuration.
- Move images to the **assets** folder.
- Copy CSS styles to **styles.css**
- Move JS and CSS references to CLI configuration file.
- Set **main** property in CLI configuration file.

Step 2: Move into Typescript

- Rename JS files to ***.ts**
- Install types for AngularJS and 3rd party libraries.
- Add **typeRoots** in TS configuration file(**tsconfig.app.json**)
- Use let, const and arrow functions.
- Correct any **tslint** errors.

Step 2: Move into Typescript (cont'd)

- Turn services and controllers into classes and make the exportable.
- Install AngularJS and 3rd party libraries from **node**.
- Use **requirejs** to load application files and libraries.
- Add types to services and controllers.

Step 3: Bootstrap as hybrid app

- Install **UpgradeModule**.
- Create **AppModule**.
- Implement **ngDoBootstrap** method.
- Remove **ng-app** attribute from index.html
- Create **main.ts** and update the CLI configuration file.

Step 4: Upgrade helper classes

- Use **environment** files for constants.
- Create global error handler in Angular.
- Upgrade the AngularJS version of **logger** service.
- **ajs-upgraded-providers.ts** file.

Step 5: Upgrade services

- Create **CoreModule**.
- Convert **logger** service to Angular.
- Create feature modules.
- Upgrade data access services using **HttpClientModule**.
- Upgrade http interceptor.
- All services are downgraded.

Step 6: Upgrade components

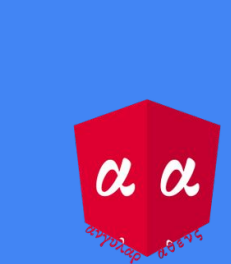
- Install **Angular Material** and create AppMaterial module.
- Upgrade **widgets** module (rename it to **shared**).
- Upgrade components in feature modules (some are downgraded).
- Upgrade the AngularJS version of **$state** service.
- Create a custom **ui-sref** directive.
- Refactor **layout** module to a single component.

Step 7: Upgrade router

- Add downgraded artifacts to the main AngularJS module.
- Create routing modules.
- Upgrade components to use **routerLink** directive.
- Upgrade **AppComponent** and bootstrap it in Angular.

Goodbye AngularJS

- Remove **gulp** completely.
- Remove upgraded AngularJS providers file.
- Remove **UpgradeModule**.
- Remove **ngDoBootstrap** implementation.
- Remove downgraded artifacts.
- Do not set AngularJS in **main.ts**



Questions

Thank you!

- Angular Athens demo: https://github.com/bampakoa/angular-athens-demo
- AngularJS style guide:

  https://github.com/johnpapa/angular-styleguide/blob/master/a1/README.md
- modular: https://github.com/johnpapa/ng-demos/tree/master/modular
- gulp-patterns: https://github.com/johnpapa/gulp-patterns
- Marvel API: https://developer.marvel.com/
- Angular upgrade guide: https://angular.io/guide/upgrade