# Switching Enterprise-level AngularJS 1.x Apps to 2.0

**Souvik Basu**, Freelance JavaScript Developer and Trainer, MeanHub





## What defines an Enterprise app?

- Larger code base
  - Created over a period of time
- Big Feature List
  - With hundreds of Routes
- Distributed team
  - Typically Geographically separated Varied skill levels
- Existing User base
  - Hundred thousand or even million
- High amount of Transactions per second
  - With high availability of Service



## What are the needs of an Enterprise app?

Backward compatibility

Cannot afford to break older deployments in use

Frequent Releases

Time to market is important Feature list is ever growing

Solid CI pipeline

Automation for regression Continuous delivery (preferably)

QA across a big site

Large QA teams who need faster releases of features



## What parameters are we interested in?

Debugability

While upgrading from 1.x to 2, all code cannot be debugged

Performance

Performance of application should be acceptable during the upgrade

Understandability

Clean code for upgrade Ramp up of team on new technologies

Refactoring and tooling

IDE should work with both versions for refeactoring Auto completion



## Is it worth upgrading to Angular >= 2.0 ?

```
Faster bootstrap
Future proof
ES6
Active development, Community support
Can handle large number of objects in Model
Better handling of Application complexity
Component based architecture
Emulated - Shadow DOM
Module based DI
Better Change Detection
```



**AOT** 

## Is it worth upgrading to Angular >= 2.0?

#### Downside

```
Learning Curve
```

TypeScript, ES6

JavaScript Modules, classes, arrow function, event handling

Obervables, RxJS

Module loaders

Styling binding to DOM properties rather than attributes

Input, Output, EventEmitter, Router, Providers.....

**Existing Team Skills** 

Unpredictability during Upgrade

Support by 3 Part libs



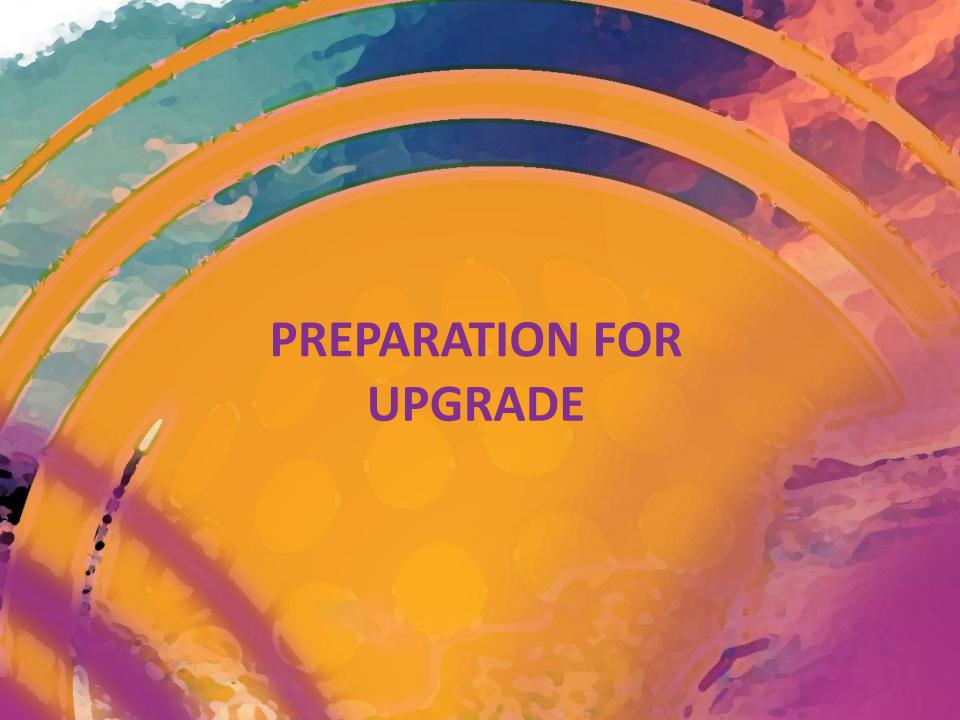


# Is it worth upgrading to Angular >= 2.0 ?

#### Downside

Size of the application
Actually this is the MOST important point!





## **Did you follow Best Practices?**

### Test Coverage

Unit, Integration, Screenshot, Compilation tests Tests Automation Enough Test Coverage

## Angular Best Practices

Component based design
Single component per file
Divide application into modules
Avoid using private variables in AngularJS lib (\$\$)



## **Did you follow Best Practices?**

- Logging Error Handling
- Abstraction Layers

Reuse components

DRY. Avoid copy paste of code

- Full SPA
- Install dependencies using npm
- Use .service() instead of .factory()





## **Big Bang or Incremental?**

## Big Bang

Small and Medium size apps
Messed up Enterprise Apps
Enough Time to Market and Budget
All New Projects
Autonomous teams

#### Incremental

For Enterprise apps currently deployed in production Small Agile teams: Small iterative fixes Distributed teams



## **Big Bang or Incremental?**

Bangalore 1.0

→ Bangalore 2.0



Already in Production Slow Infra



High Performance
Handle more features and complexities
Better UX. More green cover





# **Big Bang or Incremental?**

Problems with creating a New Bangalore 2.0 city from scratch

Very High Cost of Creation

Long time of Development and Completion

Complexity in migrating existing population to new city

Where to create new Multiplexes and Shopping Malls?

New city or Old city?

Advantages of Creating a New Bangalore 2.0 city (e.g. Amaravati)

Very well planned

10 lane roads throughout the city

Well created using latest tools and machinery

Lesser bugs (and potholes!)







## Ways to do Incremental Upgrade

 Use Existing AngularJS app and upgrade components to Angular one by one

> Component by Component Feature by Feature (Route by Route) Module by Module

ngUpgrade

Dependency Injection
Component Nesting
Content Projection / Transclusion

**Change Detection** 



## Step 0

## Bridging the gap

JavaScript → TypeScript

Concatenated all.js → Modules

Module loaders

Webpack

System.js

Browserify

Bootstrapping AngularJS and Angular side by side Angular inside AngularJS or AngularJS inside Angular?



# **Top Down or Bottom Up?**

https://github.com/souvikbasu/angular-upgrade



# **Migrating Services**

Rewrite if possible

Since there is no hierarchy of services Services are plain classes



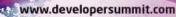
## **Migrating Routes**

Use AngularJS as well as Angular Routes together

```
<router-outlet></router-outlet></div>
```

- Routes per module
- Separate Modules for AngularJS and Angular





# Migrating Filters → Pipes



## **Migrating Tests**

- Significant changes in Unit tests to match current code
- Protractor tests using angular matchers will need minor modifications

```
ng-repeat
ng-model
protractor-conf.js → ng12Hybrid: true
```



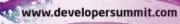
## Watch out

- Finish off the Upgrade as soon as possible
- Certain things do not work in Upgrade

templateUrl styleUrls

- You need to add the upgraded Angular component in entryComponents to use in AngularJS
- Do not try to include AngularJS component into Angular.. it gets messy soon





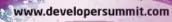
# What about 4.0 and beyond?

- Releases every 6 months
- Only upgrade npm packages

npm i @angular/common@next @angular/compiler@next @angular/core@next @angular/forms@next @angular/http@next @angular/platform-browser-dynamic@next @angular/platform-server@next @angular/router@next typescript@latest --S -E

npm i @angular/compiler-cli@next --D -E







## References

https://angular.io/docs/ts/latest/guide/upgrade.html

https://blog.thoughtram.io/angular/2015/10/24/upgrading-apps-to-angular-2-using-ngupgrade.html

https://docs.google.com/document/d/1xvBZoFuNq9hsgRhPPZOJC-Z48AHEbIBPlOCBTSD8m0Y/edit

https://vsavkin.com/migrating-angular-1-applications-to-angular-2-in-5-simple-steps-40621800a25b





