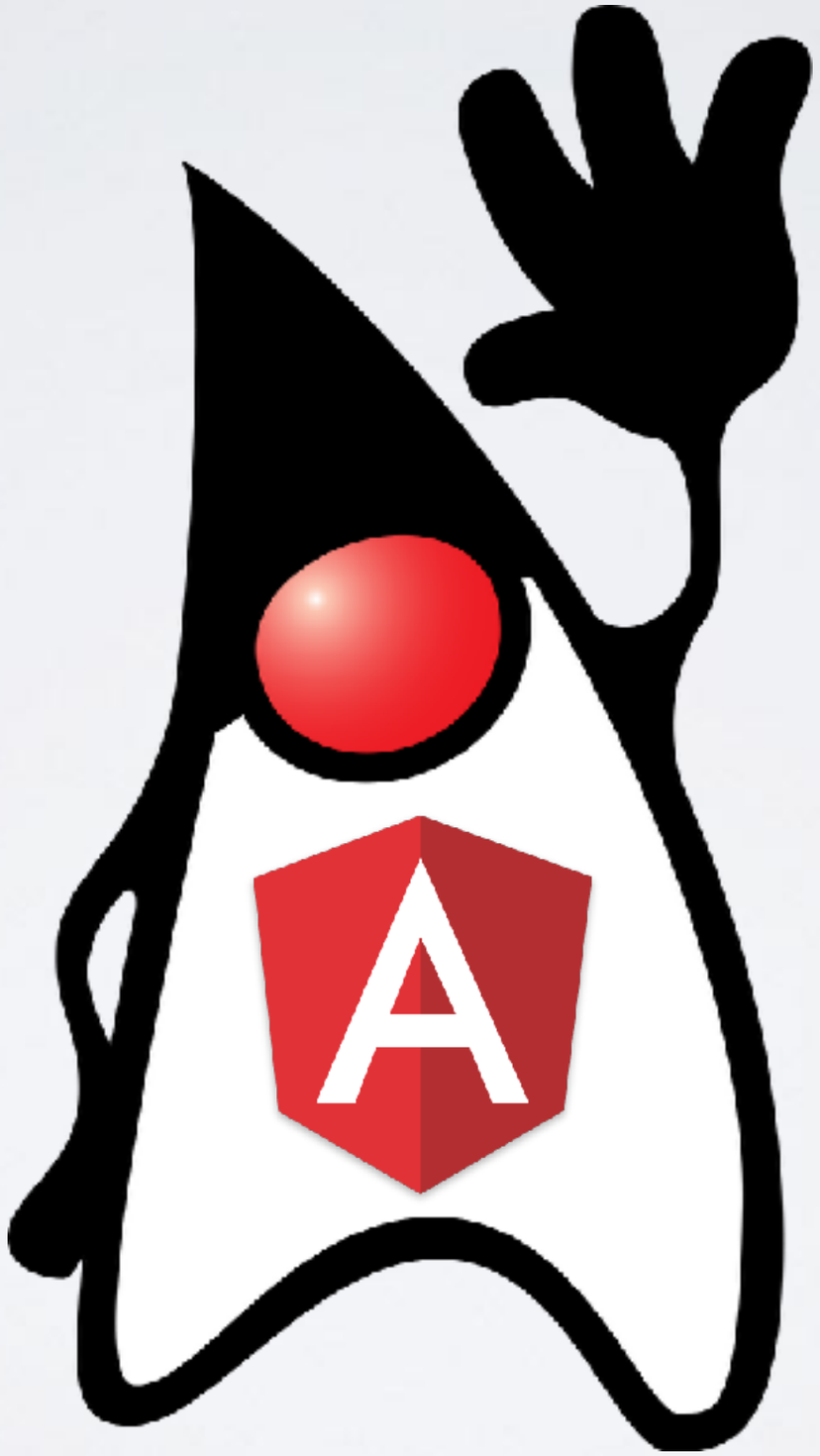
ANGULAR IN LEGACY JAVA ENVIRONMENTS

Ben Ellingson
ben@nofluffjuststuff.com
https://github.com/bellingson

Topics

- Why Discuss Angular and Java
- Deployment Architecture
- Security Options
- Build Tools
- TypeScript

Why “For Java Developers”?



Why Not Just Drop Java And Use Node?

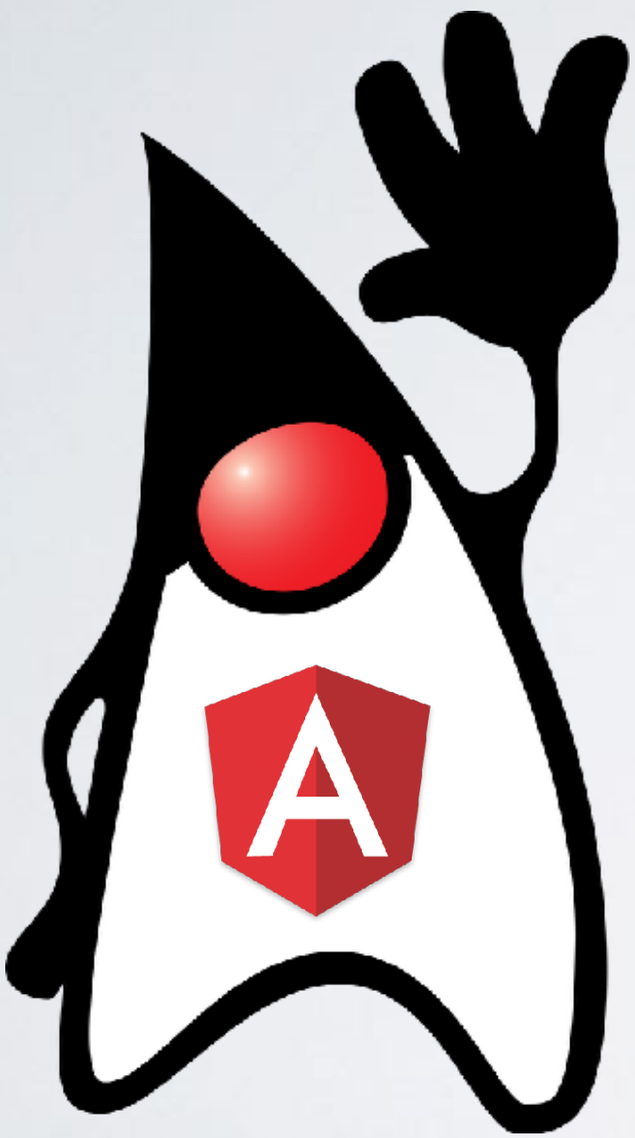
Toibe Index - Java Is Still # 1

Sep 2017	Sep 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.687%	-5.55%
2	2		C	7.382%	-3.57%
3	3		C++	5.565%	-1.09%
4	4		C#	4.779%	-0.71%
5	5		Python	2.983%	-1.32%
6	7	^	PHP	2.210%	-0.64%
7	6	v	JavaScript	2.017%	-0.91%
8	9	^	Visual Basic .NET	1.982%	-0.36%
9	10	^	Perl	1.952%	-0.38%
10	12	^	Ruby	1.933%	-0.03%
11	18	^^	R	1.816%	+0.13%
12	11	v	Delphi/Object Pascal	1.782%	-0.39%
13	13		Swift	1.765%	-0.17%
14	17	^	Visual Basic	1.751%	-0.01%
15	8	vv	Assembly language	1.639%	-0.78%
16	15	v	MATLAB	1.630%	-0.20%
17	19	^	Go	1.567%	-0.06%

What is different about doing Angular with Java?

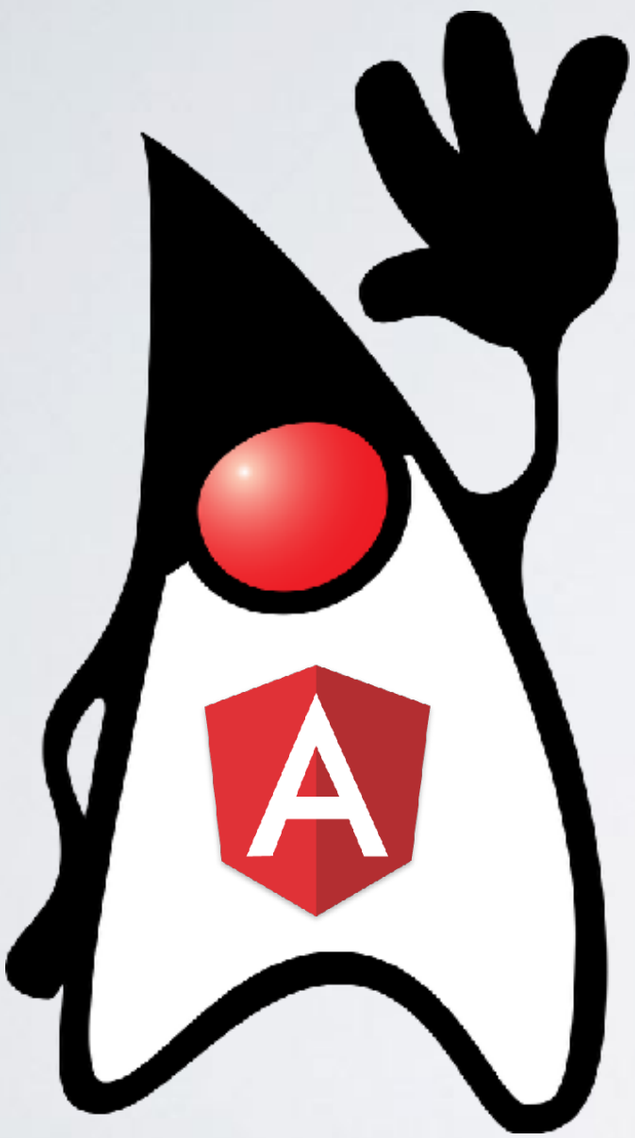
Tooling and Deployment

Deployment Options



- Deploy in WAR file
- Standalone (Microservice??)

Deploy In War File



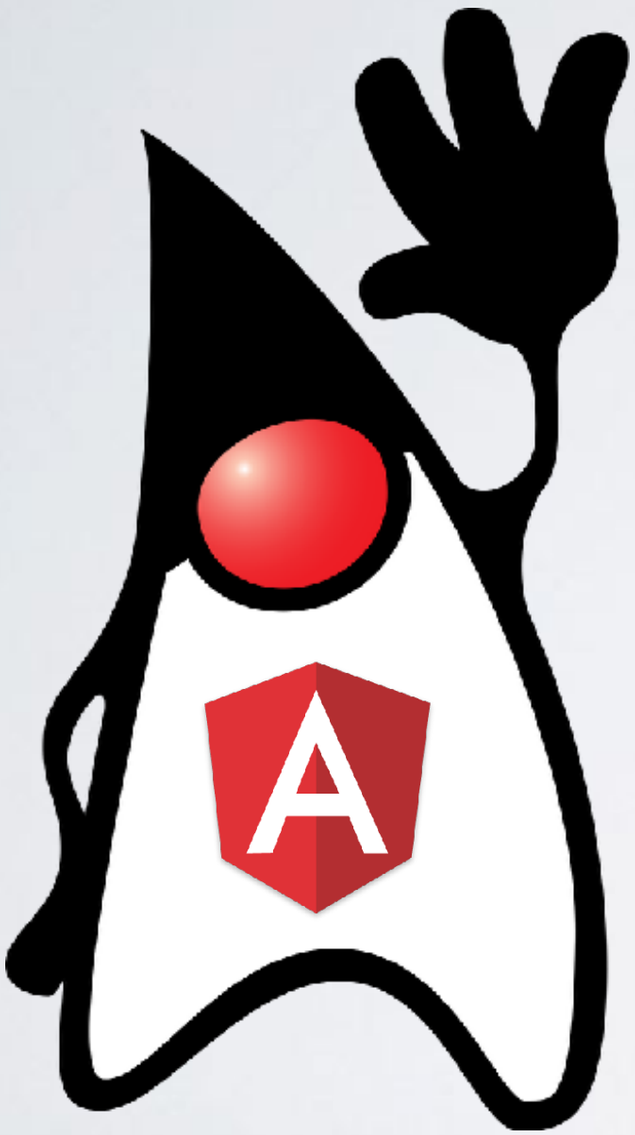
Pros

- Single Unit of Deployment
- Unified Development Environment
- Integrate with existing Security
- Integrate with existing infrastructure
- Easier hand off to operations

Cons

- Single Unit of Deployment
- Immature Tooling

Standalone (Microservice??)



Pros

- Independent Projects
- Simplified Build??
- Polyglot Architectures

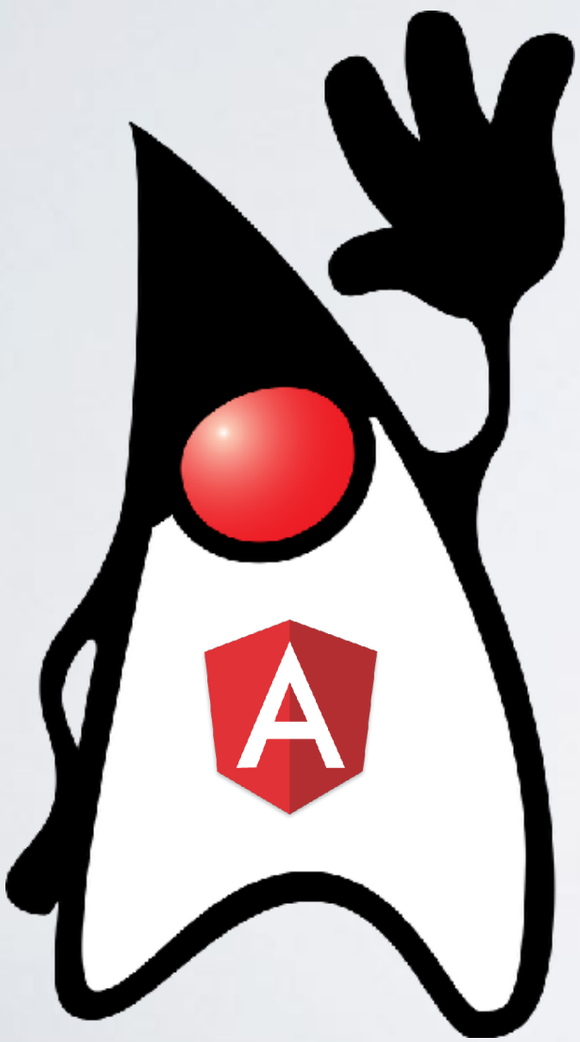
Cons

- complicates page templates / decorators / navigation
- Security is more complex
- Multiple Deployment Steps
- Requires a Mature Continuous Delivery Process
- Multiple Development Workspaces
- requires higher skill level across entire team

Binary Repositories

(nexus, npm, etc)

https://binary-repositories-comparison.github.io/



Pros

- create official builds
- reduce build times
- reduce tooling complexity

Cons

- private repositories cost $$
- another server / service to maintain

Binary Repository

Sinopia

https://github.com/rlidwka/sinopia

Deploy In War File (Java Webapp)

Java Web App with Multiple Angular SPAs

Source

```
src/
   main/
      java/
      webapp/
         style/
         script/
         WEB-INF/
            jsp/
               member/
               admin/
      js/
         member-profile/
         member-admin/
```

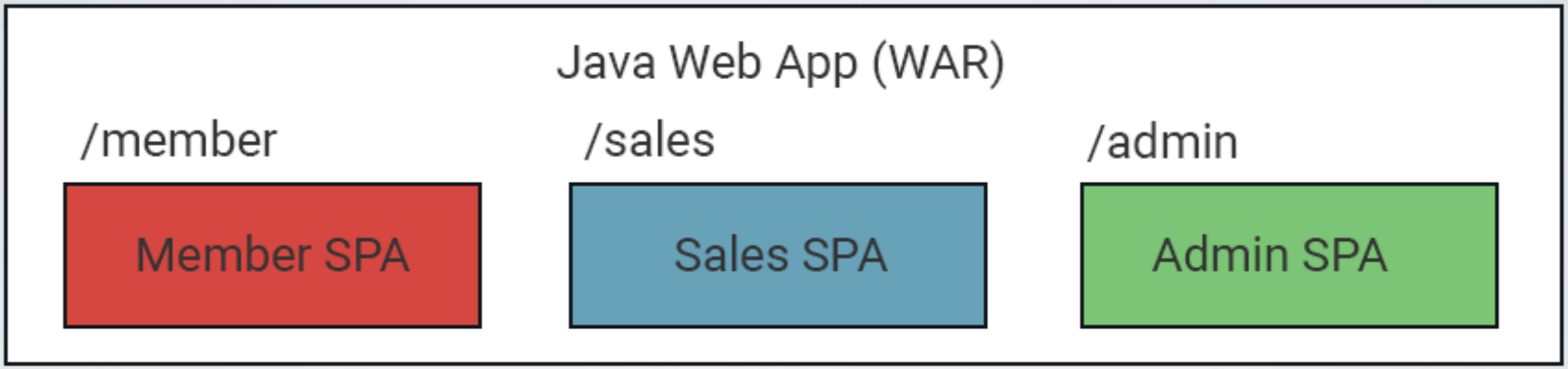
Build

```
build/
   webapp/
      member/
         profile/
      admin/
         member/
      style/
      script/
      WEB-INF/
         jsp/
            member/
            admin/
```

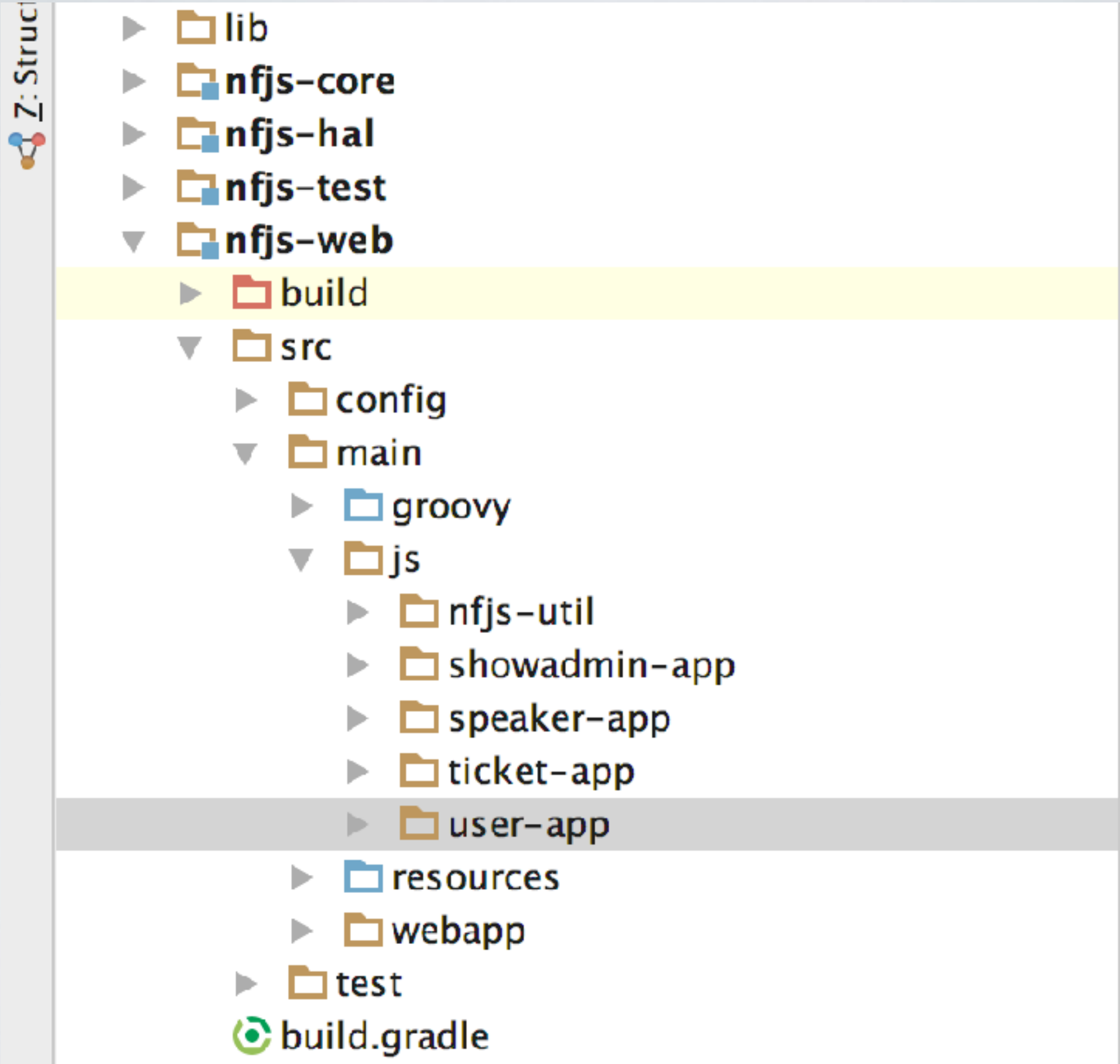
Create Angular SPAs in a 'src/main/js' level directory and integrate with webapp in the build process

Multiple SPA vs Single SPA

Java Web App with Multiple Angular SPAs



Composing application with role-based SPAs enables easy integration with Spring Security and other security packages. SPAs can be further split by feature area if necessary.


7: Struct
lib
nfjs-core
nfjs-hal
nfjs-test
nfjs-web
build
src
config
main
groovy
js
nfjs-util
showadmin-app
speaker-app
ticket-app
user-app
resources
webapp
test
build.gradle

Java Web App with Multiple Angular SPAs

Pros

- index.html is customized for each SPA
- Uses existing SpringSecurity configuration
- Feature based apps are independent of each-other

Cons

- More build overhead
- More time to upgrade
- More Angular Boilerplate code

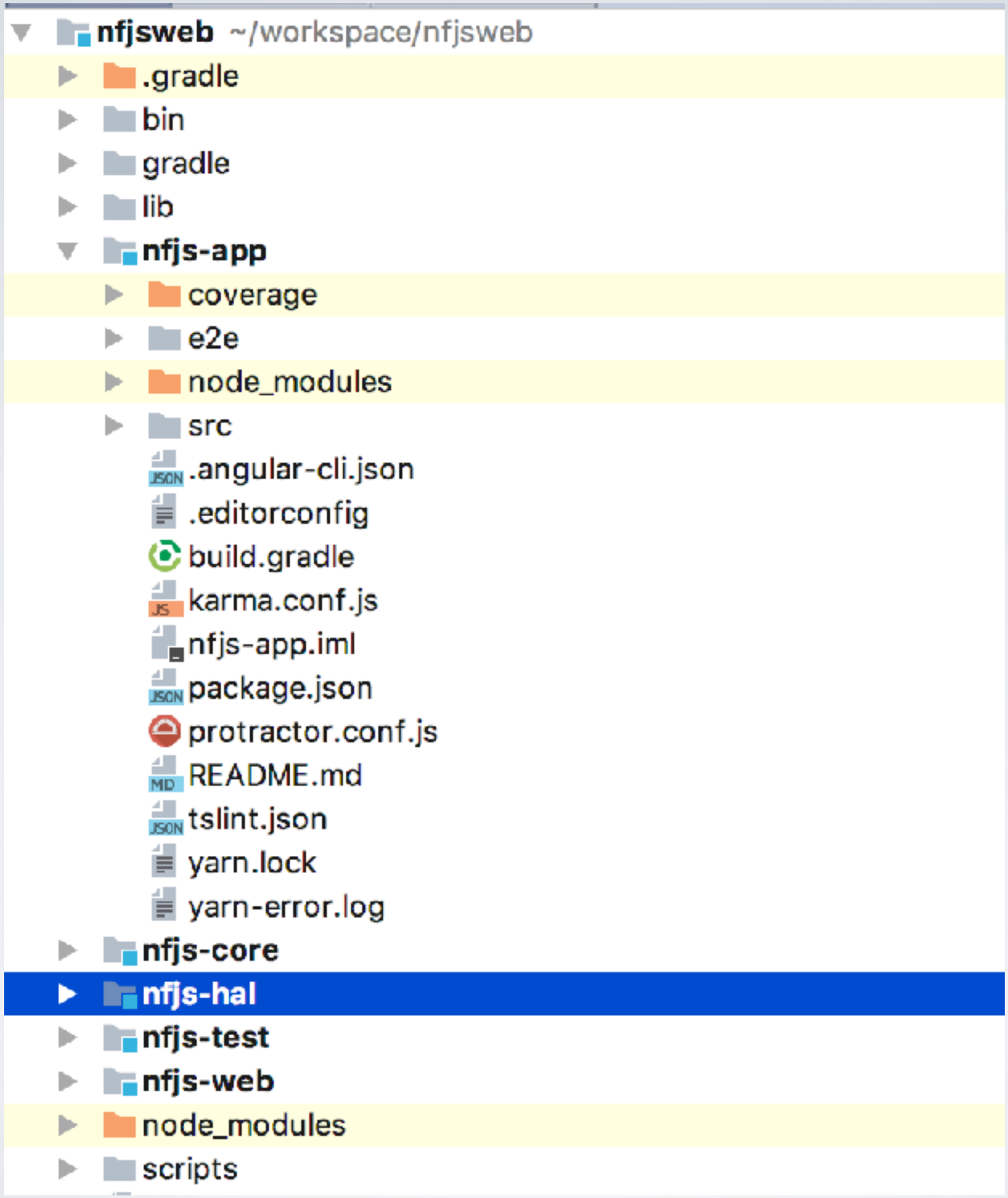
Java Web App with Single Angular SPA

Pros

- Less build overhead
- easier upgrades
- Less Angular Boilerplate code

Cons

- Can’t apply fine-grained SpringSecurity rules
- Must upgrade whole app when upgrading
- No incremental compilation


nfjsweb ~/workspace/nfjsweb
.gradle
bin
gradle
lib
nfjs-app
coverage
e2e
node_modules
src
.angular-cli.json
.editorconfig
build.gradle
karma.conf.js
nfjs-app.iml
package.json
protractor.conf.js
README.md
tslint.json
yarn.lock
yarn-error.log
nfjs-core
nfjs-hal
nfjs-test
nfjs-web
node_modules
scripts

Security Options

Only Spring Security

- Easy way to start
- Use existing configuration

Spring Security with Angular Guards

- Spring Security protects API endpoints
- Angular protects routes with Guards (CanActivate Interface)

JWT Tokens

- stateless authentication (not session based)
- scale across server cluster
- polyglot systems
- services (AuthZero and Okta)

Security Options

OAuth

- more complex

SpringSecurity Config

```
@Configuration
@EnableWebSecurity
class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()
        // admin
                .antMatchers('/admin/**').access("hasRole('ADMIN')")
                .antMatchers('/data/admin/**').access("hasRole('ADMIN')")

        // sales
                .antMatchers('/sales/**').access("hasRole('SALES')")
                .antMatchers('/data/sales/**').access("hasRole('SALES')")

        // member
                .antMatchers('/member/**').access("hasRole('MEMBER')")
                .antMatchers('/data/member/**').access("hasRole('MEMBER')")

        // more........

    }

}
```

Guarding and Angular Route

```
// app.routing.ts
const routes: Routes = [
    { path: 'ticket', children: [
        { path: 'order-admin', loadChildren: 'app/ticket/order-admin/order-
admin.module#OrderAdminModule' },
    ] },
];

export const appRouting = RouterModule.forRoot(routes);

// order-admin routing specifies guards in the canActivate properties

const routes: Routes = [

    {
        path: '', component: OrderAdminComponent, canActivate: [OrderAdminService],
        children: [
            {path: '', redirectTo: 'list', pathMatch: 'full'},
            {path: 'list', component: OrderListComponent},
        ]
    }
];

export const routing = RouterModule.forChild(routes);
```

Implementing CanActivate Interface

```
// OrderAdminService implements CanActivate interface

canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Promise<boolean> {

    return new Promise((resolve, reject) => {

        if(_.includes(this.user.roles,'ROLE_FINANCE')) {
            resolve(true);
            return;
        }

        resolve(false);
    });

}
```

JSON Web Tokens (JWT) are an open, industry standard **RFC 7519** method for representing claims securely between two parties.

Example JWT Token

Encoded

PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Decoded

EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) secret base64 encoded
```

Generate JWT Token

```
@RequestMapping(value="/n/data/member/token", method = RequestMethod.GET)
@ResponseBody Map token(HttpServletRequest request) {

    User user = data.getActiveUser(request)

    Date expires = new Date().plus(180)

    String token = JWT.create()
            .withIssuer('nfjs')
            .withClaim('userId', user.id.toString())
            .withClaim('firstName',user.firstName)
            .withClaim('lastName',user.lastName)
            .withClaim('email', user.email)
            .withClaim('roles',user.roleString)
            .withExpiresAt(expires)
    \       .sign(Algorithm.HMAC256(jwtSecret))

    return [token: token ]
}
```

Verify JWT Token

```
void checkMemberToken(HttpServletRequest request, HttpServletResponse response) {

    JWTVerifier verifier = JWT.require(Algorithm.HMAC256(LoginJSONController.jwtSecret))
            .withIssuer('nfjs')
            .build(); //Reusable verifier instance
    JWT jwt = (JWT) verifier.verify(token);

    Long userId = Long.valueOf(jwt.getClaim('userId').asString())
    String firstName = jwt.getClaim('firstName').asString()
    String lastName = jwt.getClaim('lastName').asString()
    String email = jwt.getClaim('email').asString()

    List<String> roles = jwt.getClaim('roles').asString().tokenize(',')

    UserShort userS = new UserShort(id: userId, email: email, firstName: firstName, lastName: lastNam
    userS.roles =  roles as String []
    UsernamePasswordAuthenticationToken auth = new UsernamePasswordAuthenticationToken(userS,
userS.password, userS.authorities )


    SecurityContext securityContext = SecurityContextHolder.getContext()


    securityContext.setAuthentication(auth)
    request.setAttribute(NfjsData.SES_KEY_USER, userS)

    request.session.setAttribute('SPRING_SECURITY_CONTEXT', securityContext)

}
```

Build Tools

IntelliJ IDEA the Java IDE

Ben

JetBrains s.r.o. [CZ] https://www.jetbrains.com/idea/

JET BRAINS

ALL TOOLS IDEs .NET & VISUAL STUDIO TEAM TOOLS LANGUAGES STORE SUPPORT WE ARE JETBRAINS

IntelliJ IDEA

What's New Features Learn Buy Download

IJ IntelliJ IDEA

Capable and Ergonomic
Java * IDE

DOWNLOAD

*Actually, much more than just Java

Javascript Tool Fatigue

YEOMAN

GRUNT

webpack
MODULE BUNDLER

jspm.io

Angular CLI

https://cli.angular.io

ANGULAR CLI CLI COMMANDS GITHUB

```
> npm install -g angular-cli
> ng new my-dream-app
> cd my-dream-app
> ng serve
```

Angular CLI

A command line interface for Angular

GET STARTED

ng new

The Angular2 CLI makes it easy to create an application that already works, right out of the box. It already follows our best practices!

ng generate

Generate components, routes, services and pipes with a simple command. The CLI will also create

yarn Getting Started Docs Packages Blog

Aあ English

Search packages (i.e. babel, webpack, react...)

FAST, RELIABLE, AND SECURE DEPENDENCY MANAGEMENT.

GET STARTED INSTALL YARN Star 24,409

Stable: v0.23.2

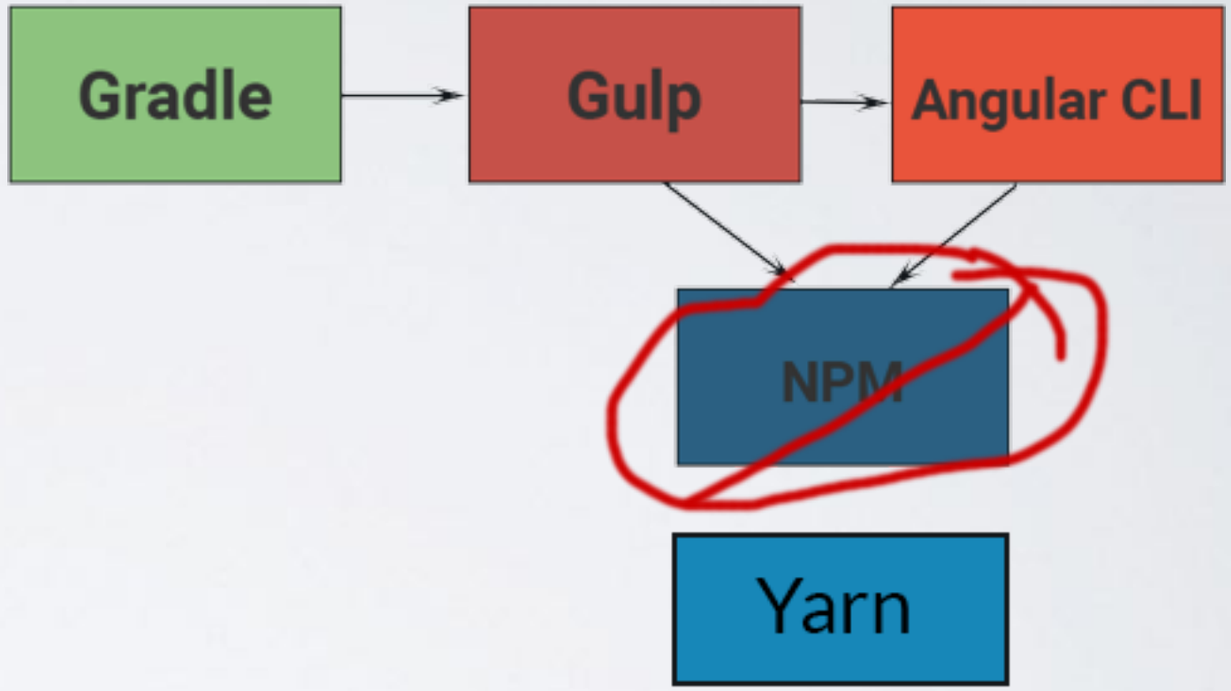
Ultra Fast.

Yarn caches every package it downloads so it never needs to

WebPack
vs
SystemJS

Recommended Tool Stack

- Gradle
  - manage Java dependencies
  - exec Gulp
  - build war
- Angular CLI
  - build Angular App(s)
- Gulp
  - exec Angular CLI
  - front-end processing (SASS, Minification, etc)
- NPM / Yarn
  - manage JavaScript dependencies



Gradle

```
// build.gradle

task gulp(type: Exec) {

    workingDir '.'

    if (System.getProperty('os.name').toLowerCase().contains('windows')) {
        commandLine 'cmd', '/c', 'gulp'
    } else {
        commandLine 'gulp'
    }

    standardOutput = System.out
    errorOutput = System.out

}
```

Gulp

```
/*
 *  exec build via angular-cli
 *  ng build --prod --aot
 *
 */

function buildNgApps(isProd, watch) {

    var execConfig = ng2Apps.map(function(app) {
    var cmd = 'ng build ' + (isProd ? ' -prod --aot ' : ' ') +
              (watch ? ' --watch ':'');
        return { cmd: cmd, cwd: app.src };
    });

    return execV.execWithVerify(execConfig);
}
```

Npm & Yarn

```
// install a package to node_modules in current directory
npm install mypackage
yarn add mypackage

// install a package globally
npm install mypackage -g
yarn global add mypackage

// install a package and save as dependency in package.json
npm install mypackage -save
yarn add mypackage -save

// install a package and save as dev dependency in package.json
npm install mypackage -save-dev
yarn add mypackage -d

// install all packages defined in the project's package.json
npm install
yarn

// run the start script defined in the package.json
npm start
yarn start
```

Angular CLI / Build Watch

```
product-mgr$ ng build --watch
Hash: 8080d5d3df175e433708
Time: 6111ms
chunk    {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 158 kB {4} [initial
[rendered]
chunk    {1} main.bundle.js, main.bundle.js.map (main) 17.1 kB {3} [initial] [rendered]
chunk    {2} styles.bundle.js, styles.bundle.js.map (styles) 9.77 kB {4} [initial] [rende
chunk    {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.36 MB [initial] [rendered]
chunk    {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]
```

TypeScript

- Looks more familiar to Java Developers
- Superset of JavaScript ES6
  - adds classes and modularity to JavaScript
  - syntactic sugar and many other features
- TypeScript adds Static Typing to JavaScript
  - helps prevent type related bugs
  - enables contextual help in development tools
- TypeScript is transpiled to JavaScript
- Defacto Standard for Angular
- Angular framework is written in TypeScript

Classes

```
export class Shape {

    name: string;
    sides: number;

    constructor(name: string, sides: number) {
        this.name = name;
        this.sides = sides;
    }

    describe() {
        console.log(`A ${this.name} has ${this.sides} sides`);
    }
}
```

Inheritance

```
export class Square extends Shape {

    constructor() {
        super('square', 4)
    }
}

let square = new Square();
square.describe();

// A square has 4 sides
```

Constructors with Member Variables

```
export class Shape {

    constructor(public name: string,
                public sides: number) { }

}
```

Data Types

```
export class Author {

    id: number;
    name: string;
    age: number;
    isAlive: boolean;

    books: Array<Book>;

    articles: Array<any>;

}
```

Methods (Functions)

```
export class Author {

    constructor(public id: number,
                public name: string,
                public books: Array<any>) { }

    description() : string {
        return `${this.name} wrote ${this.bookCount()} book(s)`;
    }

    bookCount() : number {
        return this.books.length;
    }
}

let a = new Author(1, 'Bob Smith', ['JavaScript for Dummies']);

console.log(a.description());

// Bob Smith wrote 1 book(s)
```

let and const

```
export const MAX_VALUE = 1000;

export class MyClass {

    doSomething() {

        let x = 100;
        var y = 200;

        console.log(`x is ${x}`);
        console.log(`y is ${y}`);
        console.log(`max is ${MAX_VALUE}`);

    }
}

let c = new MyClass();
c.doSomething();

// x is 100
// y is 200
// max is 1000
```

Interfaces

```
export interface Shape {

    name: string;
    sides: number;

    describe(): string;
}

export class Square implements Shape {

    constructor(public name: string = 'square', public sides: number = 4) { }

    describe() : string {
        return `A ${this.name} has ${this.sides} sides`;
    }
}

let square = new Square();
console.log(square.describe());

// A square has 4 sides
```

import

```
import { Square } from './square';

let square = new Square();
console.log(square.describe());

// A square has 4 sides
```

Fat Arrow Functions

```
// 'regular' function assignment

let myFunction = function(message) {
    console.log(message);
};

// function assignment with fat arrow function

let myOtherFunction = (message) => {
    console.log(message);
};

myFunction('Hello World');

myOtherFunction('Hello World');
```

Fat Arrow Functions

```
class MyClass {

    doSomething(callback: any) {
        // do it
        callback();
    }
}

let c = new MyClass();

c.doSomething(() => {
    console.log('something is complete');
});
```

Transpilation

```
export class Person implements Mammal {

   constructor(public id: number,
               public name: string,
               public age: number) {
    }


    sayHello() {
        console.log(`Hello ${this.name}`);
    }

}
```

Transpilation Result

```
"use strict";
var Person = (function () {
    function Person(id, name, age) {
        this.id = id;
        this.name = name;
        this.age = age;
    }
    Person.prototype.sayHello = function () {
        console.log("Hello " + this.name);
    };
    return Person;
}());
exports.Person = Person;
```

Topic Review

- Why Discuss Angular and Java
- Deployment Architecture
- Security Options
- Build Tools
- TypeScript

Questions?

Podcasts

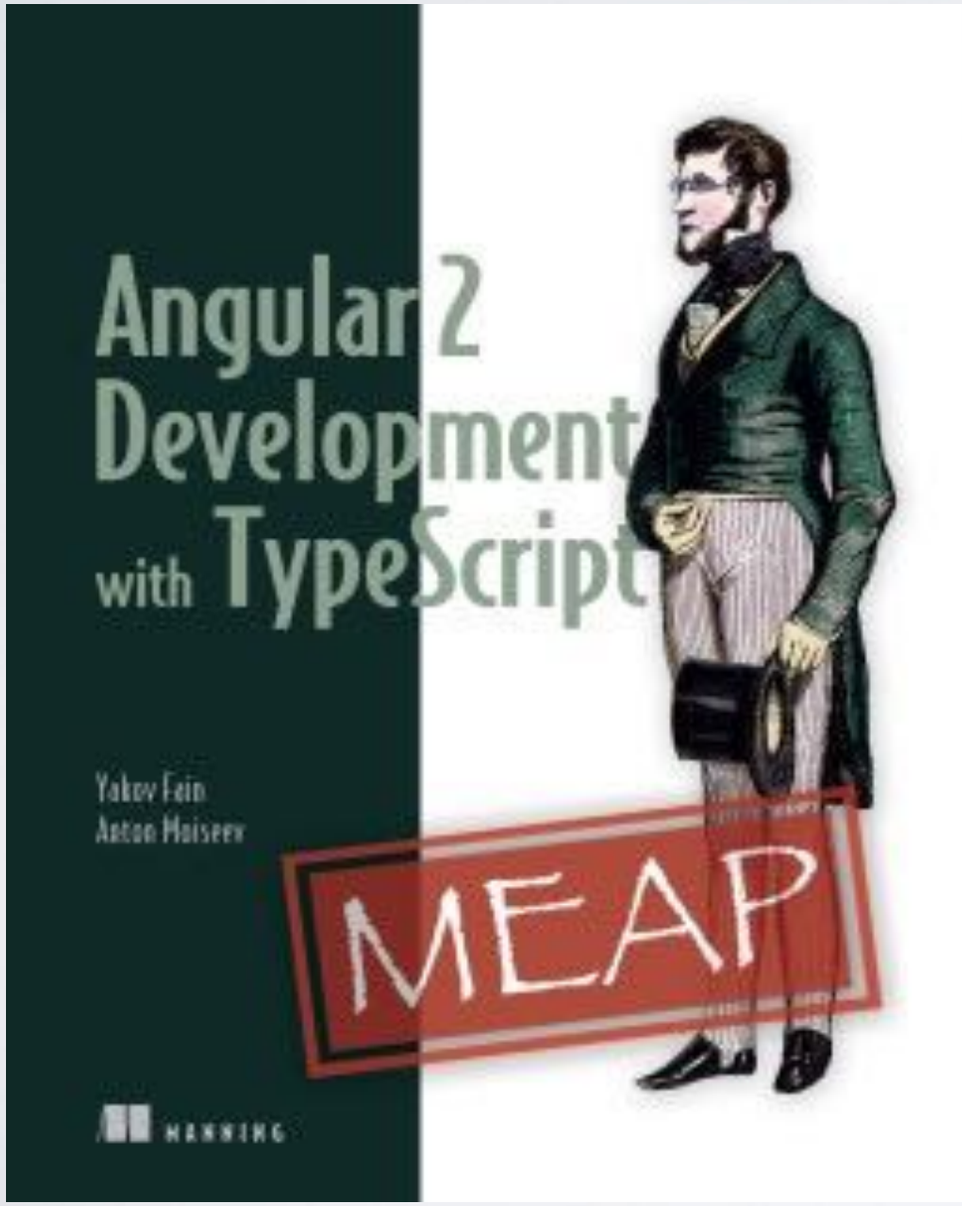

ADVENTURES IN
ANGULAR

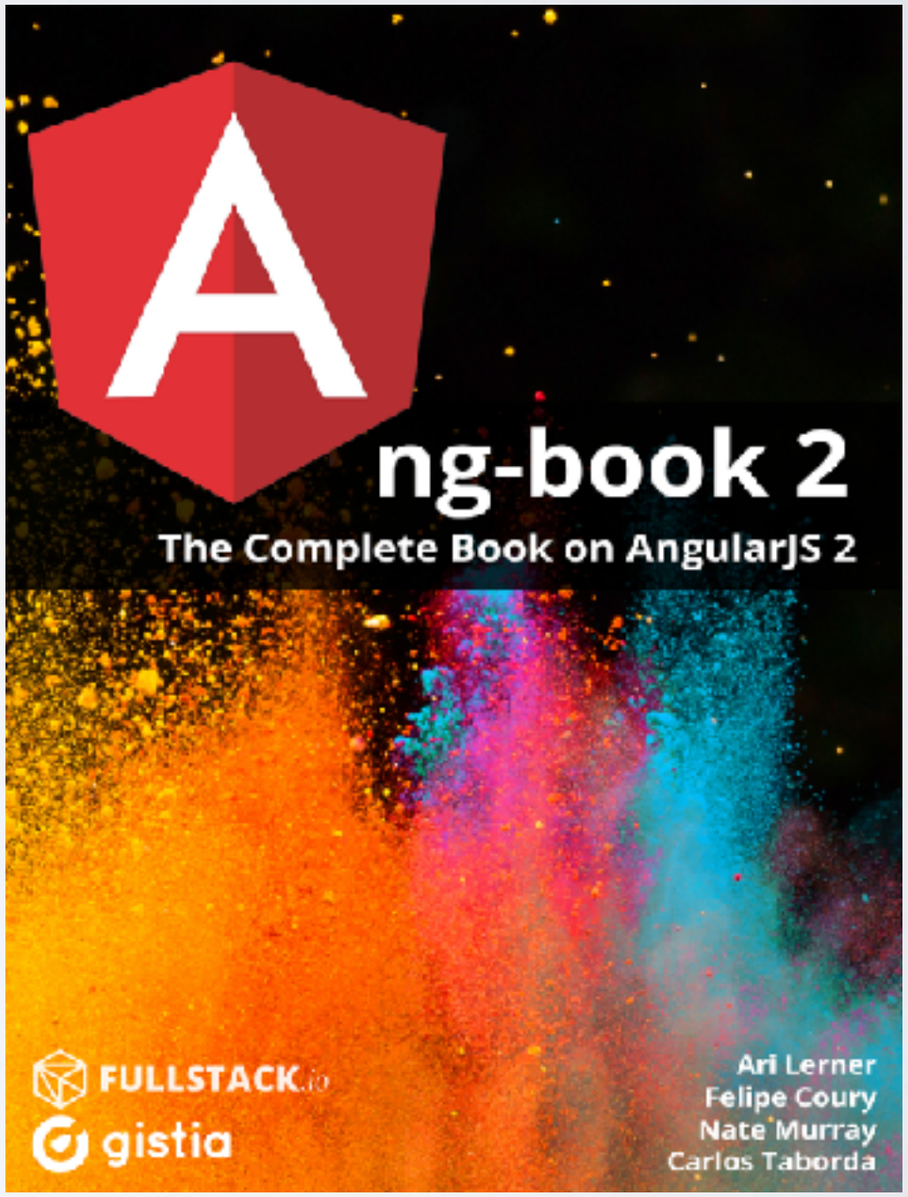

@AngularAir


JAVASCRIPT
JABBER
‹DevChat.tv


JS

Books


Angular 2
Development
with TypeScript
Yakov Fain
Anton Moiseev
MEAP
MANNING


ng-book 2
The Complete Book on AngularJS 2
FULLSTACK.io
gistia
Ari Lerner
Felipe Coury
Nate Murray
Carlos Taborda

ANGULAR SUMMIT

2:45 - 3:00 PM - BREAK

3:00 - 4:30 PM

UI Router: The ultimate routing solution for every Angular app
Peter Pavlovich

Mastering Angular Animations (v4+)
Gerard Sans

4:30 - 5:00 PM - BREAK

5:00 - 6:30 PM

Angular Architectures: A roadmap for the hearty traveller
Peter Pavlovich

Angular 2 in Legacy Java Environments
Ben Ellingson

6:30 - 7:30 PM - DINNER

7:30 - 8:30 PM - Keynote

The pursuit of perfection in engineering and art
Michael Carducci

Angular 2 in Legacy Java Environments

Ben Ellingson
Lead Developer
NoFluffJustStuff.com

Thursday, May 11 - 5:00 PM

There are several Angular 2 books and many tutorials available. However, they all tend to assume you are starting an application from scratch or that you are in a pure JavaScript / Node development environment. In reality many of us are adopting Angular 2 into legacy environments. This presents a lot of problems. Which tools should you use and how do you integrate with a legacy application? How to you create an efficient workflow? How do you deploy to production?

This talk will review Angular 2 tooling challenges and show efficient solutions for adopting Angular 2 in a legacy Java web application. Tools covered include: gradle, npm, gulp, and angular-cli.

Please Submit an Eval

Add to Itinerary | Slides | Video | Eval | Ask Question | Audio

ANGULAR IN LEGACY JAVA ENVIRONMENTS

Ben Ellingson
ben@nofluffjuststuff.com
https://github.com/bellingson