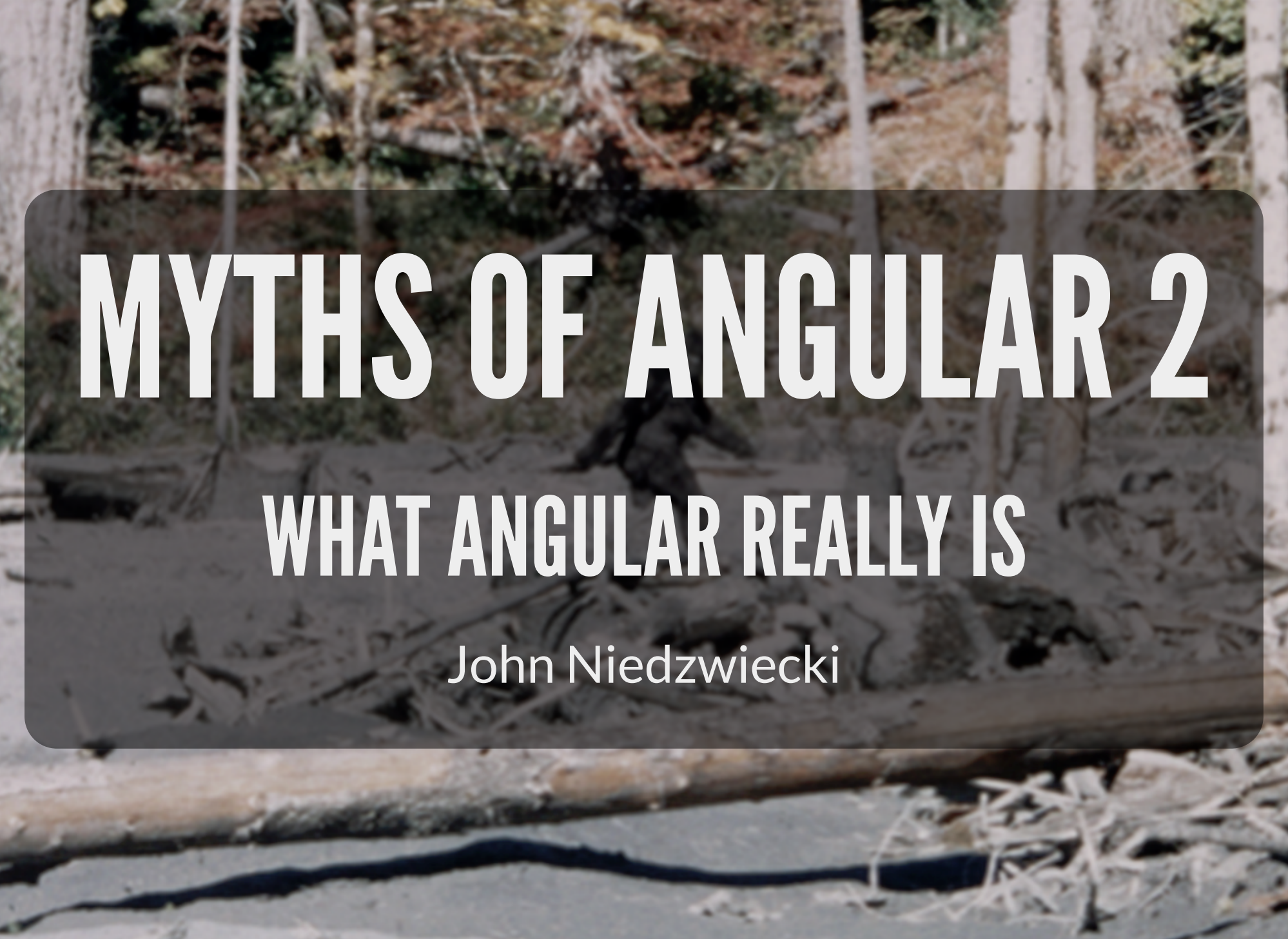

MYTHS OF ANGULAR 2
WHAT ANGULAR REALLY IS
John Niedzwiecki

ABOUT THE GEEK

```
function getDeveloperInfo() {
    let dev = {
        name: 'John Niedzwiecki',
        company: 'ThreatTrack',
        role: 'Lead Software Engineer - UI',
        twitter: '@rhgeek',
        github: 'rhgeek',
        web: 'https://rhgeek.github.io',
        currentDevLoves: ['Angular', 'JavaScript Everywhere', 'd3', 'CSS instead of JS'],
        nonDevAttr : {
            isHusband: true,
            isFather: true,
            hobbies: ['running', 'disney', 'cooking', 'video games', 'photography'],
            twitter: '@rgrdisney',
            web: 'http://www.rungeekrundisney.com',
        }
    };
    return dev;
}
```

WHY WE'RE HERE

At ng-Europe in 2014, the Angular team showed off Angular 2. Unfortunately, the drastic changes were not received well by many. While things have changed since that announcement during the development of Angular 2, many people remember the things said there instead of what was actually done (damn first impressions).

WHAT WE'LL DO

- What is Angular?
- Version Soup
- Mythbusting
- Why Angular?

WHAT IS ANGULAR?

WHAT IS ANGULAR?

"Angular is a development platform for building mobile and desktop web applications."

https://github.com/angular/angular

COMPONENT

```
/* /theme-parks/theme-parks-list/theme-parks-list.component.ts */
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs/Rx';
import { ParksService, ThemeParkGroup } from '../../shared';

@Component({
  selector: 'app-theme-parks-list',
  templateUrl: './theme-parks-list.component.html',
  styleUrls: ['./theme-parks-list.component.css']
})
export class ThemeParksListComponent implements OnInit {

  themeParks: Observable< ThemeParkGroup[] >;

  constructor(
    private parksService: ParksService
```

Code: https://github.com/RHGeek/mythbusters-angular

COMPONENT TEMPLATE

```
<div *ngfor="let group of themeParks | async">
  <h2><a routerlink="../details/{{group.id}}">{{group.company}}</
  <div>
    Location: {{group.location}}
  </div>
  <div>
    # of Parks: {{group.parks.length}}
  </div>
</div>
```

COMPONENT

```
/* /theme-parks/theme-park-details/theme-park-details.component.t
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Params, Router } from '@angular/router';
import { ParksService, ThemePark } from '../../shared';

@Component({
  selector: 'app-theme-park-details',
  templateUrl: './theme-park-details.component.html',
  styleUrls: ['./theme-park-details.component.css']
})
export class ThemeParkDetailsComponent implements OnInit {

  themePark: ThemePark;

  constructor(
    private route: ActivatedRoute
```

COMPONENT TEMPLATE

```
<a routerlink="/parks/list" class="link--back"><small>back</small>
<div class="park-card">
  <h2>{{themePark?.company}}</h2>
  <div>
    Location: {{themePark?.location}}
  </div>
  <div>
    Parks:
    <ul>
      <li *ngfor="let park of themePark?.parks">{{park.name}}</li>
    </ul>

  </div>
</div>
```

SERVICES

```
/* /shared/parks.service.ts */
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import { Observable } from 'rxjs/Rx';
import { ThemePark } from './theme-park.model';
import { ThemeParkGroup } from './theme-park-group.model';

@Injectable()
export class ParksService {

  constructor(private http: Http) { }

  getParks(): Observable< ThemeParkGroup[] > {
    let url: string = `/api/parks/list`;
    return this.http.get(url)
              .map(response => response.json() as ThemeParkGroup[])
```

IMPROVED ROUTING

```
/* /app-routing.module.ts */
import { NgModule }             from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { CanDeactivateGuard } from './guards/';
import { ThemeParksComponent } from './theme-parks/theme-parks.co
import { ThemeParksListComponent } from './theme-parks/theme-park
import { ThemeParkDetailsComponent } from './theme-parks/theme-pa

const routes: Routes = [
    { path: '', redirectTo: 'parks', pathMatch: 'full' },
    { path: 'parks', component: ThemeParksComponent,
        children: [
            { path: '', redirectTo: 'list', pathMatch: 'full' },
            { path: 'list', component: ThemeParksListComponent },
            { path: 'details/:parkId', component: ThemeParkDetail
        ]
```

CHILD OUTLETS

```
/* /theme-parks.component.html */
<h1>{{greeting}}</h1>
<router-outlet></router-outlet>
```

LAZY LOADING MODULES

```
/* from previous routing code */
{ path: 'about', loadChildren: 'app/about/about.module#AboutModul

/* /about/about.module.ts */
import { NgModule } from '@angular/core';
import { CommonModule }   from '@angular/common';
import { AboutComponent } from './about.component';
import { AboutRoutingModule } from './about-routing.module';

@NgModule({
  declarations: [
    AboutComponent
  ],
  imports: [
    CommonModule,
    AboutRoutingModule
```

LET'S SEE IT

http://localhost:4200/

ANGULAR OR ANGULARJS?

AngularJS == 1.x

Angular == 2, 4, 6, 8...
who do we appreciate?
also may see things like ng2 or ngx

speaking of Angular 2 and Angular 4...

SEMANTIC VERSIONING

SEMANTIC VERSIONING

With new Angular release, the team announced the switch to Semantic Versioning (SEMVER).

WHAT DOES IT MEAN FOR YOU?

- Future changes in a predictable way.
- Time based release cycles.
- Patch release each week, ~3 minor updates, and one major update every 6 months.

BREAKING CHANGES?

Angular 1 to Angular 2 was a complete rewrite. Angular jumped from version 2 to 4 to align the core packages versions (Angular's router was already version 3). Future major updates will be more standard "breaking changes" with proper documented deprecation phases.

JUST ANGULAR

From here on, it's just #angular.

MYTH: EVERGREEN BROWSERS ONLY

"DESIGNED FOR THE FUTURE"

Angular 2 was designed "targeting modern browsers and using ECMAScript 6". This meant only evergreen, auto-updating browswers.

Source

BROWSER SUPPORT AND POLYFILLS

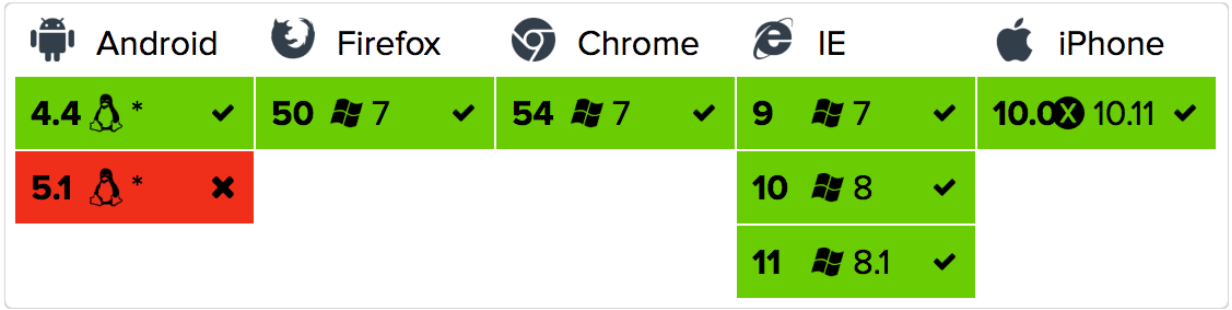
Angular supports most recent browsers, on both desktop and mobile.

Chrome	Firefox	Edge	IE	Safari	iOS	Android	IE mobile
latest	latest	14	11	10	10	Marshmallow (6.0)	11
		13	10	9	9	Lollipop (5.0, 5.1)	
			9	8	8	KitKat (4.4)	
				7	7	Jelly Bean (4.1, 4.2, 4.3)	

Official Browser Support

BROWSER SUPPORT AND POLYFILLS

build passing | circleci passing | gitter join chat | pull requests closed in about 22 hours | issues closed in 7 days | npm package 4.1.1



Safari (7+), iOS (7+), Edge (14) and IE mobile (11) are tested on BrowserStack.

https://github.com/angular/angular

BROWSER SUPPORT AND POLYFILLS

There's a challenge in supporting the larger range of browsers. They offer mandatory and optional polyfills, depending on the browsers you want to support.

"Note that polyfills cannot magically transform an old, slow browser into a modern, fast one."

Source

ANGULAR-CLI PROVIDED POLYFILLS.TS

```
/**
 * This file includes polyfills needed by Angular and is loaded before the app.
 * You can add your own extra polyfills to this file.
 *
 * This file is divided into 2 sections:
 *   1. Browser polyfills. These are applied before loading ZoneJS and are sorted by brows
 *   2. Application imports. Files imported after ZoneJS that should be loaded before your
 *      file.
 *
 * The current setup is for so-called "evergreen" browsers; the last versions of browsers
 * automatically update themselves. This includes Safari >= 10, Chrome >= 55 (including Op
 * Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.
 *
 * Learn more in https://angular.io/docs/ts/latest/guide/browser-support.html
 */

/***************************************************************************************
 * BROWSER POLYFILLS
 */
```

MYTH: EVERGREEN BROWSERS ONLY

BUSTED

MYTH: NO MIGRATION PATH

"WE WILL SEE"

Initially, no path to migration was announced. The team's goal was the best framework to build a web app without worrying backwards compatibility with existing APIs.

Obviously, people were less than happy for no concrete migration path.

UPGRADING FROM ANGULARJS

Not only can you migrate an app, you can upgrade *incrementally*.

- Follow Angular Style Guide
- Use a module loader
- Migrate to TypeScript
- Use Component Directives

THE UPGRADE MODULE

This module allows you to run AngularJS and Angular components in the same application and interoperate with each other seemlessly.

Actually runs both versions of Angular side by side, no emulation. Interoperate via dependency injection, the DOM, and change detection.

More information: NgUpgrade

MYTH: NO MIGRATION PATH

BUSTED

MYTH: BASE ANGULAR APP IS HUGE

BASE ANGULAR LIBRARIES MAKE A SIMPLE APP HUGE

After the release, I've heard many arguments about the size of Angular files. It often pops up in comparisions of Angular vs React. You would see the comparison of Angular 2's minified version (566K) to React JS with add-ons, minified version (144K).

Min size comparisions

AHEAD OF TIME (AOT) COMPILATION

Non-issue for an Angular app thanks to AOT compilation. Angular built to take advantage of ES2015 module and tree shaking.

Proof of concept Hello World can be made in just 29KB

POC

AHEAD OF TIME (AOT) COMPILATION

Angular 4 has made even larger improvements for AOT. The team has implemented a new View Engine to generate less code.

Two medium size apps improvement:
from 499Kb to 187Kb (68Kb to 34Kb after gzip)
from 192Kb to 82Kb (27Kb to 16Kb after gzip)

App stats

MYTH: BASE ANGULAR APP IS HUGE

BUSTED

WHY ANGULAR?

SPEED AND PERFORMANCE

I loved AngularJS, but there were performance issues (I'm looking at you dirty checking and digest cycles). Part of the reason for the rewrite for Angular 2 was to get things right, from the ground up.

Faster checking of single bindings, improved change detection strategies, Immutables, Observables, use of zone.js, AOT compiling, lazy loading modules

ANGULAR ANYWHERE

- Mobile Web
- Desktop Web
- Mobile Native (Angular + NativeScript)
- Internet of Things (IoT)


UNIVERSAL
Universal is more than just a theme park with some awesome Harry Potter lands. It's also the name of the *official* package for server side rendering in Angular.
Server side rendering can be important to single page apps (SPA). Everyone knows it matters for search engine optimization. It's also great to ensure you get that nifty little preview on social media.

ANGULAR UNIVERSAL

It's also great for *perceived* performance.

Angular Universal lets you render your app on the server, give it to the user, preboot a div to record their actions, and then replay and swap out once the app is bootstrapped.

https://universal.angular.io/

TYPESCRIPT: THE THING YOU MAY BE SURPRISED TO LOVE

Frankly, I was worried about *needing* to use TypeScript going into Angular 2.

Then I used it.

Now, I  it.

FANTASTIC TOOLING

- Great IDE Integration
- Great TypeScript tooling
- AOT Compiling
- Angular-CLI

ANGULAR-CLI

Amazing command line tool for Angular applications.

https://cli.angular.io/

COMMUNITY

Established community that has grown over many years.

Find me: @rhgeek

https://rhgeek.github.io/

Code: https://github.com/RHGeek/mythbusters-angular