What Lies Ahead for AngularJS

Matt Frisbie

Overview

- AngularJS 1.3
  - New modules
  - New features
- AngularJS 2.0
  - ng-europe hubbub
  - It's a whole new ballgame
  - Major change rundown
  - The Good, the Bad, and the Ugly

AngularJS 1.3

Rookies

- ngAria
- ngMessages
- $scope.$watchGroup()
- ngStrictDI
- $touched/$submitted
- Custom form validators
- HTML5 datetime inputs
- One-time/lazy data binding
- ngModelOptions

New 1.3 Modules

ngAria

- Independent module for making applications WAI-ARIA compliant
- Defines a way to make content more accessible to people with disabilities
- Upon inclusion, automatically implemented using aria-* attributes

ngMessages

- Independent module providing an add-on form error message framework
- ngMessages exists as two separate directives: ng-messages, which defines the $error message block and consumes a form $error object; and ng-message, which maps to a specific property within the $error object
- ngMessages allows for error template reuse
- http://jsfiddle.net/msfrisbie/w9x4L3b2/

New 1.3 Features

$scope.$watchGroup()

- Accepts array of expressions that map to the same listener callback
- newVal and oldVal parameters are ordered arrays that correspond to the ordering of the watched expression collection
- http://jsfiddle.net/msfrisbie/e7q9vbxp/

ng-strict-di

- If ng-strict-di is included on an element, functions without minification-safe dependency – ['$scope', function($scope) {}] – injection syntax will fail to execute.
- Useful when checking for minification-vulnerable applications

$touched, $submitted form states

- New states offer better control over form completion flow
- $touched is a stateful version of $pristine that tracks if a focus/blur event pair has occured
- $submitted is a stateful tracker of submission attempts
- http://jsfiddle.net/msfrisbie/7kpzyqjn/

Custom form validators

- Custom form validation no longer requires formatters/validators
- Injecting ng-model into the attribute directive definition allows for custom validation logic
- Synchronous and asynchronous validation is available through ngModel.$validators and ngModel.$asyncValidators, respectively
- http://jsfiddle.net/msfrisbie/4q21ke19/

HTML5 datetime inputs

- You can now bind to these input types and preserve their native data format
  - **<input type="date">**
  - **<input type="dateTimeLocal">**
  - **<input type="time">**
  - **<input type="week">**
  - **<input type="month">**
- If unsupported in the browser, these fields gracefully degrade to ISO datestrings

One time/lazy data binding

- Partially inspired by bindonce
  - https://github.com/Pasvaz/bindonce
- {{ ::user.name }}
- An expression with lazy binding will be watched until its value becomes defined, in which case it will schedule itself for deletion from the watchlist
- http://jsfiddle.net/msfrisbie/3vadvap3/

ngModelOptions

- Provides greater degree of control over how and when the model is updated
- ng-model-options directive takes a stringified configuration option object
  - updateOn
  - debounce
  - allowInvalid
  - getterSetter
  - timezone
  - $rollbackViewValue
- http://jsfiddle.net/msfrisbie/Lp37oLpw/

AngularJS 2.0

The ng-europe hubbub

- Igor Minar and Tobias Bosch dropped some bombshells during the ng-europe Angular 2.0 presentation
- AngularJS 2.0 is a complete rewrite and barely recognizable
- 1.3 will be eventually phased out
- Community response has been “emphatic”

[-] **terror406** **56 points** 12 days ago

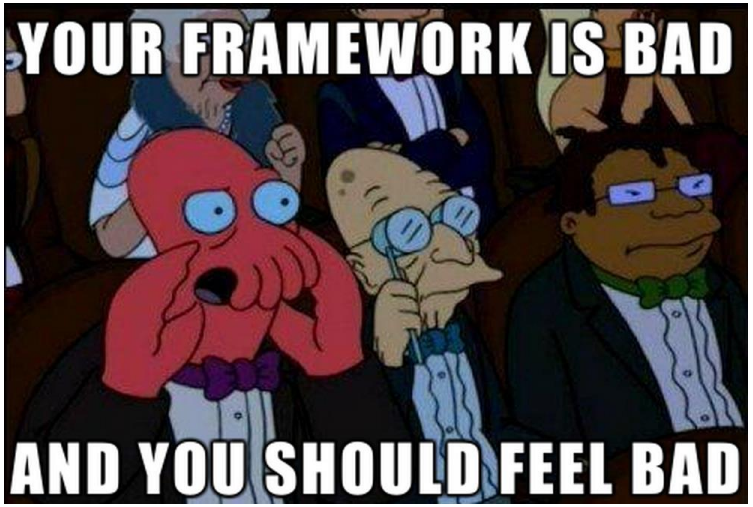
I'm not a front-end or javascript developer, so I have no dog in this fight.

I have however been a developer for over 25 years, and I've seen enough language, frameworks and platforms try to do exactly the same. It's suicide. *It has never worked.* Sure, some projects have "survived", but only in a very small niche. Hell, in some cases they have even been surpassed by forks of the original version.

No serious development outfit will now consider adopting Angular. Not for *years*. Those who currently have Angular in production on a scale they cannot just quickly refactor will never, ever use it again. And even those who can still make the jump will consider other frameworks first.

In the ecosystem of frameworks, Angular has now become an evolutionary dead end.

"Is this a joke?"



"Screw You, Angular"

Anthony Short @anthonyshort · Nov 6

I'm really surprised people didn't see all the problems with **Angular** before the **2.0** blow up

1 3

Why all gnashing of teeth?

- AngularJS as we know it has been gutted
  - Controllers, $scope, directive definition objects, angular.module(), jqLite have all been discarded
  - Unification of Angular and AngularDart means 2.0 is written in AtScript, a brand new Google-created language (extending ECMAScript 6)
- Existing AngularJS apps now have a shelf life
  - 1.3 only supported for 1.5 years after 2.0 release
  - No clear upgrade path to 2.0 for existing 1.3 applications, plugins, and third party libraries
- It would appear that all the hard work we have put into learning the framework needs to be redone

Yikes.


DON'T
PANIC

Method to this madness

- ECMAScript 6 and web components are the driving forces behind the radical change
- AtScript and Traceur allow for backwards compatibility with ES5, progressive integration with ES6, Angular/AngularDart compilation from a singular annotated and typed codebase
- Angular core team is (hopefully) skating to where the puck is going to be

It’s a whole new ballgame.

Web Components

- The Web Component W3C specification decomposes into the following general cases:
  - HTML imports
  - Custom elements
  - Template elements
  - Shadow DOM
- Angular 1.x specification has substantial overlap with Web Components
- Polymer allows you to do most of this already

Welcome to the future

Web Components usher in a new era of web development based on encapsulated and interoperable custom elements that extend HTML itself. Built atop these new standards, Polymer makes it easier and faster to create anything from a button to a complete application across desktop, mobile, and beyond.

GET POLYMER VIEW ON GITHUB

Use Elements (30 sec) ➜ Create Elements (5 min) ➜ Build an app (30 min)

```
<!-- Polyfill Web Components support for older browsers -->
<script src="components/webcomponentsjs/webcomponents.js"></script>

<!-- Import element -->
<link rel="import" href="google-map.html">

<!-- Use element -->
<google-map lat="37.790" long="-122.390"></google-map>
```

Using Polymer Elements

Using Polymer elements (like all Web Components) is absurdly simple:

1 Import element.

Find a component and import its definition into your page using an HTML Import (`<link rel="import">`).

Web Component: HTML imports

- Provide a way to include and reuse HTML documents within other HTML documents

```
<link rel="import" href="signup-form.html">

<signup-form></signup-form>
```

- AngularJS analogue: ng-include*

Web Component: Template Elements

- Define reusable HTML using declarative element <template></template>

```
<template id="signupTemplate">
  <h2>Signup</h2>
  <form>
    …
  </form>
</template>
```

- signupTemplate can now be reused throughout the application by id reference
- AngularJS analogue: ng-template

Web Component: Custom Elements

- Enables developers to create new types of fully-featured DOM elements
- http://jsfiddle.net/msfrisbie/8a7pke1n/
- Pairs nicely with template elements
- AngularJS analogue: (element) directives

Web Component: Shadow DOM

- Provides DOM encapsulation
- No global object
- No window object
- <element>.createShadowRoot()
- CSS, JS can be scoped
- AngularJS analogue: directive templates

DOM APIs

- Every DOM element exposes four APIs:
  - Attributes
  - Events
  - Methods
  - Properties
- Everything is DOM. Angular doesn’t know and shouldn’t know about custom elements.

ECMAScript 6

- Lots of changes
  - Classes
  - Modules + Dependency Injection
  - Promises
  - Object.observe()
  - And much, much more!!!!!11one
- Traceur compiles ES6 to ES5
- Lots of overlap with existing Angular 1.x components

Major Change Rundown

AtScript

- Like Microsoft's TypeScript, extends ECMAScript 6
- Enhancements:
  - Type annotations
  - Field annotations
  - Metadata annotations
  - Type introspection
- Required to contribute to the Angular 2.0 codebase
- Use in developing your Angular 2.0 applications is **optional**

AtScript

```
class myClass {
  methodA(name:string):int {
    var length:int = name.length;
    return length;
  }
}
```

ECMAScript 6

```
Class MyClass {
  methodA(name) {
    var length = name.length;
    return length;
  }
}
```

AtScript

```
import {Component} from 'angular';
import {Server} from './server';

@Component({selector: 'foo'})
export class MyComponent {
  constructor(server:Server) {
      this.server = server;
  }
}
```

ECMAScript 6

```
import * as rtts from 'rtts';
import {Component} from 'angular';
import {Server} from './server';

export class MyComponent {
  constructor(server) {
      rtts.types(server, Server);
      this.server = server;
  }
}

MyComponent.parameters = [{is:Server}];
MyComponent.annotate = [
  new Component({selector: 'foo'})
];
```

Dependency Injection

- Old-style dependency injected modules (angular.module) are replaced with ES6 modules and constructor arguments.
- Instance scope
- Child injectors
- AtScript annotators
- Promise-based and asynchronous injection
- https://gist.github.com/msfrisbie/23ed5df17ce8eaf506a7

Templating and Data Binding

- Dynamic Loading
- Directives
  - Component
  - Decorator
  - Template
- https://gist.github.com/msfrisbie/429a0643e44f3bf166bf

Angular 2.0 Template Syntax

- Binding and template syntax is still very much in the air
- It appears that there is a good deal of dissent in the community as well as within the core team

Angular 2.0 Template Syntax

Standard binding:

```
<date-picker [value]="expression"
             (change)="expression">
```

Template binding:

```
<div [ng-repeat|person]="people">person</div>
```

- https://gist.github.com/msfrisbie/8076ccb20956c81fb0

Routing

- Sibling view ports
- Nested routing
- Navigation model
- Component lazy-loading
- Screen activator
- Customizable internal asynchronous pipeline
- https://gist.github.com/msfrisbie/5dfd1b455e2dec7865e3

Timeline

- HTML 5.0 standardization: Oct 28, 2014
- ECMAScript 6 release: June 2015
- Web Components standardization: ???
- Angular 2.0 tentative release:
  - Angular 1.0: June 2012
  - Angular 1.1: August 2012
  - Angular 1.2: November 2013
  - Angular 1.3: October 2014
  - Angular 2.0: late 2015/early 2016

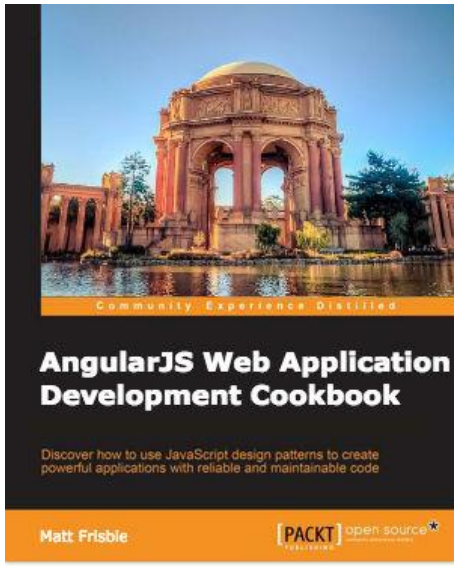
The Good, the Bad, and the Ugly

The Good

- Don’t panic
  - The same team that brought you 1.x is still calling the shots for 2.0
  - You can bet that ES6 is going to rip apart a lot of frameworks and libraries other than AngularJS
  - There’s plenty of time before the 2.0 release
  - Nothing is set in stone – it’s still being written!
- Change is good
  - The team is trying to push AngularJS to match what frameworks and web development will be in the future
  - Web dev is as turbulent of a profession as it gets – this is par for the course
  - The future of web client technologies is an awesome one

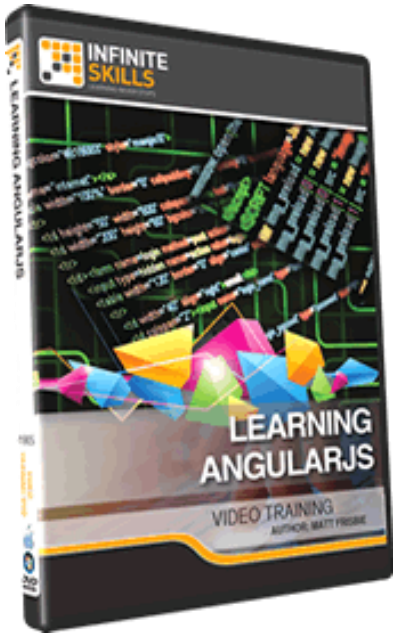
The Bad, and the Ugly

- Merge of Angular and AngularDart only marginally benefits the community, but at great cost
- Everything that exists now must be upgraded or rebuilt
- The learning curve just got a little bit steeper
- Other frameworks await with open arms
- Companies will not be enticed to adopt a framework that is effectively already deprecated



Coming soon:

AngularJS Web Application Development Cookbook (Packt Publishing)



Available now:

Learning AngularJS (Infinite Skills)

Further reading

- AtScript Primer: https://docs.google.com/document/d/11YUzC-1d0V1-Q3V0fQ7KSit97HnZoKVygDxpWzEYW0U/edit
- All About Angular 2.0: http://eisenbergeffect.bluespire.com/all-about-angular-2-0/?utm_source=javascriptweekly
- Data Binding with Web Componentshttps://docs.google.com/document/d/1kpuR512G1b0D8egl9245OHaG0cFh0ST0ekhD_g8sxtI/edit#heading=h.xgjl2srtytjt
- Web Components and Concepts:http://toddmotto.com/web-components-concepts-shadow-dom-imports-templates-custom-elements/
- Gist of links and references: https://gist.github.com/msfrisbie/32d4d093bb333aaa1e66