

AngularJS

[Classic](#) [Flipcard](#) [Magazine](#) [Mosaic](#) [Sidebar](#) [Snapshot](#) [Timeslide](#)

3 weeks ago

Angular 2 Survey Results



Dynamic Views template. Powered by [Blogger](#).

This is a guest post from Jeff Whelpley [<https://twitter.com/jeffwhelpley>] and Patrick Stapleton [<https://twitter.com/gdi2290>]. Though they're not full time Angular team members, they are incredible contributors to the Angular community. I think this post is a great summary of what users want to see from us. Enjoy!

- Brad Green

A couple weeks ago, Patrick and I sent out a survey to Angular developers [https://docs.google.com/forms/d/1BSHEMqR-1fkG0_edLiOQql7nbNT7Z8aHVxMJok1ljgo/viewform] with questions about how they plan on using Angular 2. We got over **2,100 responses** and a lot of additional feedback. All of the results are posted below with some context and analysis.

Disclaimer

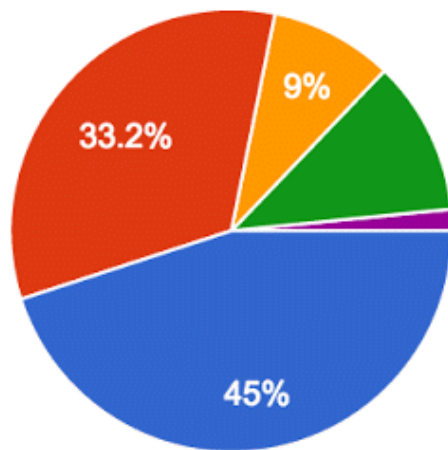
Originally we were only thinking about getting feedback from the small group of developers that are already spending most of their day working on Angular 2. This quickly changed, however, as more and more people started submitting responses. Angular 2 is only in alpha and you cannot build production apps with it yet. It is probably safe to assume that many people who responded are just starting to learn

about Angular 2. Therefore, we now view the results in terms of what developers think they will do in the future rather than what they are actually doing right now.

With that disclaimer out of the way, let's get to the results...

Transpilers

Which approach do you prefer when hacking Angular 2? (choose one)



| | | |
|------------|-------|-----|
| TypeScript | 45.0% | 950 |
| Babel | 33.2% | 700 |
| Not sure | 11.3% | 238 |
| ES5 | 9.0% | 190 |
| Other | 1.6% | 33 |

Analysis

For the first 1,000 results or so, TypeScript was in a dead heat with Babel but then

TypeScript started to pull away. This is not surprising given that the Angular core team has a preference for TypeScript. We had a great discussion about TypeScript vs Babel on Angular Air

[<https://plus.google.com/events/cvo18elq1u8vam3lhnoa4dheo4k>] and we concluded that much of the decision is going to come down to the personal preferences of the team.

Although sticking with ES5 is not a good long term strategy, a number of developers have used it to help with the transition from Angular 1 to Angular 2. If you are thinking about sticking with ES5 just because you don't like transpilers, however, you should give them another shot. Transpilers in general have vastly improved over the past year.

"Other" Transpilers

A couple people wrote in that they use both TypeScript and Babel which may seem a little odd at first, but both teams have discussed ways that they can potentially work together in the future [<https://twitter.com/jntrnr/status/636201907850096640>].

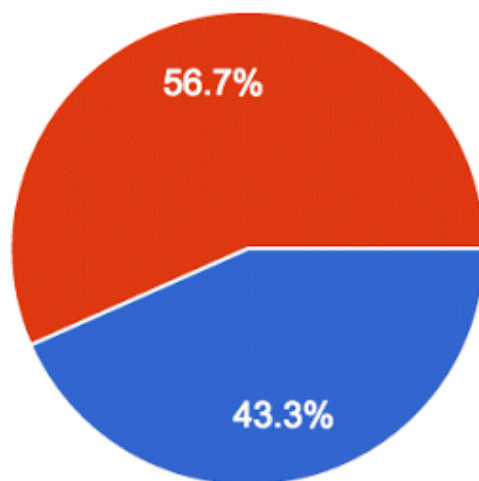
There were about a half dozen responses for Dart and 1 for Closure Compiler.

Also, there were 7 responses for CoffeeScript, presumably from the Ruby/Python

crowd.

Template Syntax

Which template binding syntax will you use? (choose one)



`bind-prop="val"` 56.7% 1192

`[prop]="val"` 43.3% 912

Analysis


This was initially one of the most surprising results, but the more we thought about it, the more it made a lot of sense. Although the Angular core team prefers the second option (i.e. `[prop]=val` which is the canonical syntax), most developers seem have the same initial reaction. It feels weird and unfamiliar (is it even valid

<http://blog.thoughttram.io/angular/2015/08/11/angular-2-template-syntax-demystified-part-1.html>). The first option (i.e. `bind-prop="val"`) looks and feels like Angular 1.x template code.

Despite this, it may be interesting to note that all developers we know who hack on Angular 2 every day use the canonical syntax in most cases. It was weird at first, but we all got used to it. Remember that when Angular 1 first came out, everyone was freaking out about using non-standard attributes that start with the prefix **ng-**. Initially many Angular 1 developers thought about either prefixing attributes with **data-** (ex. `data-ng-bind`) or using the xmlns notation of `ng:*`. Those ideas quickly faded away, however, once people realized they didn't provide any real benefits.

"Other"


I thought this was sort of interesting:



Jeff Whelpley
@jeffwhelpley

19 Aug

@gbziel @bradlygreen @gdi2290 you could...but why?



Grzegorz Biziel
@gbziel

Follow

@jeffwhelpley I like simplicity of [prop]="val" for properties and a strong distinguish of events by "on-" prefix.

6:44 PM - 19 Aug 2015

1

Some of this particular concern will go away once text editors properly support Angular 2 syntax highlighting, but it is nice that you can mix and match the canonical syntax with the alternate syntax as you please. For example, even though we use the canonical syntax for binding and events, we prefer the alternate syntax for vars:

*ng-for="**var item** of collection"

instead of:

*ng-for="**#item** of collection"

We suggest trying out different variations of syntax yourself. Don't shy away from the canonical syntax just because it feels weird. Finally, keep an eye on out for what others are using in examples online and at conferences. More than likely, the community will converge on a set of standards over time.

Template Location

Where do you prefer to keep your templates? (choose one)

| | | |
|---------------|-------|------|
| Both | 47.6% | 1008 |
| External file | 46.5% | 986 |
| Inline | 3.4% | 73 |
| Not sure | 1.8% | 39 |
| Other | 0.6% | 13 |

Analysis

We have a strong preference for inline templates, but there are a number of use cases where separate template files make more sense.

Advantages of inline templates:

1. All code for a given component in one file
2. Encourages developers to keep templates small and refactor when they get too big

Advantages of separate template files:

1. Angular 1 developers familiar with this approach
2. Easier to share with designers and other non-developers
3. Generally better intellisense support in editors today
4. More natural home for large templates that can't be broken down into smaller pieces (ex. forms and layouts)

We think that the numbers in favor of inline templates will increase over the next year as developers get familiar with Angular 2 and intellisense support increases.

As far as sharing templates with designers, it should be noted that in the React world, there has been some success getting designers to modify inline templates [https://twitter.com/dan_abramov/status/577995829195210752].

Routing

Which routing mechanism do you prefer? (choose one)

| | | |
|------------------|-------|-----|
| Component Router | 36.7% | 776 |
| UI Router | 33.0% | 698 |
| Not sure | 28.1% | 594 |
| Custom | 1.7% | 35 |
| Other | 0.4% | 9 |

Analysis

There are a few things to keep in mind while looking at these results:

1. There is a lot of hype around the new router and it has the support of the core team which is why it is in the lead.
2. That said, many of the developers that have been working with Angular 2 so

far have experienced frustrations (i.e. things not working or features missing) which is why it doesn't have an overwhelming lead.

3. The UI Router doesn't yet have any Angular 2 support, but it is what many developers use with Angular 1 apps and what they expect.

This is not an ideal situation, but we believe that this is just a short term issue. Many of the problems developers have encountered with the Component Router have either already been fixed or are in the process of being addressed. There are also plans to integrate the UI Router with Angular 2 in the near future. So, by the time Angular 2 is released, developers should have two great options for routing.

Data Libraries

Select any of the following data-related libraries that you will likely utilize on a majority of your Angular 2 projects: (choose multiple)

| | | |
|--------------|-------|-----|
| RxJS | 38.7% | 522 |
| Immutable.js | 34.7% | 469 |
| Firebase | 33.3% | 449 |

| | | |
|----------------|-------|-----|
| angular-meteor | 23.8% | 321 |
| Falcor | 14.5% | 196 |
| Other | 9.1% | 123 |
| BaconJS | 6.6% | 89 |

Analysis

There are a lot of libraries out there that can help you manage your data both in Angular 1 and Angular 2. Some interesting notes for each library:

1. RxJS

This library implements observables [<https://github.com/zenparsing/es-observable>], which are objects that model push-based data sources. We think the popularity of RxJS in Angular 2 boils down to two primary things. First, a great API [<http://reactivex.io/documentation/observable.html>] for handling one or more asynchronous operations (this is why RxJS is used in the Angular 2 core http module). Second, performance gains from using observables with Angular 2 change detection [<http://victorsavkin.com/post/110170125256/change-detection-in-angular-2>] (reducing complexity from $O(N)$ to $O(\log N)$). For more information on

RxJS, check out our [discussion with the creator of RxJS on Angular Air](#) [https://www.youtube.com/watch?feature=player_embedded&v=fV5G9lXRBvA]. For more information on observables, take a look at the proposed implementation of [observables in ES2016](#) [<https://github.com/zenparsing/es-observable>].

2. Immutable.js

It is probably safe to attribute the popularity of [Immutable.js](#) [<https://facebook.github.io/immutable-js/>] to React. As mentioned further below, a significant portion of Angular developers also use React. An even bigger portion, including many of the Angular core team members, are aware of the concepts and ideas coming out of the React community and have started to use those [concepts in their Angular development](#) [<http://victorsavkin.com/post/114168430846/two-phases-of-angular-2-applications>]. Lee Bryon from the React team gave [a great talk](#) [<https://www.youtube.com/watch?v=I7IdS-PbEgI>] earlier this year about the benefits of immutability in general. One of the biggest benefits of using immutable objects in Angular 2, is that it reduces change detection complexity from [\$O\(N\)\$ to \$O\(1\)\$](#) [<https://www.youtube.com/watch?v=QHulaj5ZxbI&feature=youtu.be&t=2352>]. In other words, change detection occurs in constant time that is not linked to the number of bindings on the page.

3. Firebase

Firebase [<https://www.firebase.com/>] has always been extremely popular in the Angular community and I wouldn't expect anything to change with Angular 2. In fact, the numbers here can and should be much higher because we didn't include Firebase as an option in the survey when it was originally sent out. We added Firebase and angular-meteor after 50 responses were already submitted.

4. angular-meteor

Uri, the creator of angular-meteor, hit me over the head about five minutes after I posted the survey and I quickly added it as an option. There is a strong, vibrant community around this library and it supports both Angular 1 [<https://github.com/Urigo/angular-meteor>] and Angular 2 [<http://angular-meteor.com/angular2>].

5. Falcor

The new kid on the block (just recently released [<http://netflix.github.io/falcor/>]) has been getting a lot of hype since it is backed by Netflix and has been frequently touted by Jafar Husain [<https://plus.google.com/events/ca3l6qalpu0uqcce58a379006m0>] over the past couple months. Falcor leverages asynchronous data binding in

Angular 2 and can be extremely powerful if you have a mostly read-only application.

6. Bacon.js

Bacon.js [<https://baconjs.github.io>] is another observables library, but not as popular as RxJS.

"Other" Data Libraries

There were a number of other libraries in responses including:

Angular 2 Data

This library isn't available yet (which is why it wasn't in the survey), but is being modeled after Ember Data

[<https://docs.google.com/document/d/1DMacL7iwjSMPP0ytZfugpU4v0PWUK0BT6lhyaVEmlBQ/edit>] to provide a higher level interface for your data in Angular 2. Other data libraries may be used in conjunction at lower levels. We talked with the Angular core team about Angular 2 Data on Angular Air [https://www.youtube.com/watch?feature=player_embedded&v=QAPK51nE3Cc].

Relay/GraphQL

Created by Facebook and part of the “React stack”, this library [<http://facebook.github.io/react/blog/2015/02/20/introducing-relay-and-graphql.html>] is similar in concept to Falcor, but with more features and more complicated (Jafar compares the two here [<https://www.youtube.com/watch?v=WL54eYbTJUw&feature=youtu.be&t=53m55s>]).

Breeze.js

Works well with Angular 1 for caching, model validation, offline support and more [<http://www.getbreezenow.com>].

Server Rendering

Will you use the server rendering features when they are available in Angular 2?
(choose one)

| | | |
|-----------|-------|-----|
| Yes | 27.2% | 566 |
| Sometimes | 25.9% | 539 |

| | | |
|----------|-------|-----|
| No | 25.7% | 534 |
| Not sure | 21.2% | 442 |

Analysis

We split this question out from the general Angular 2 features question below because Patrick and I are heavily involved in the effort to get Angular 2 rendering on the server. As we discussed on [Adventures in Angular](https://devchat.tv/adventures-in-angular/056-aia-server-rendering-with-angular-with-jeff-whelpley-and-patrick-stapleton) [<https://devchat.tv/adventures-in-angular/056-aia-server-rendering-with-angular-with-jeff-whelpley-and-patrick-stapleton>], universal rendering is fundamentally better than client-only rendering so the only question is about how much effort it takes. We view this result as very positive since we haven't released the feature yet and only a handful of developers have used it yet. So, if 52.7% of developers think they will use server rendering on at least some of their projects now, we feel really good

about gaining a majority of the mind share by the time we release this feature.

Track our progress on the [Universal Angular repo](https://github.com/angular/universal)

[\[https://github.com/angular/universal\]](https://github.com/angular/universal) and/or follow us on Twitter ([@jeffwhelpley](https://twitter.com/jeffwhelpley) [\[https://twitter.com/jeffwhelpley\]](https://twitter.com/jeffwhelpley) and [@gdi2290](https://twitter.com/gdi2290) [\[https://twitter.com/gdi2290\]](https://twitter.com/gdi2290)). Also be sure to tune into [Angular Connect](http://angularconnect.com/sessions#full-stack-angular-2) [\[http://angularconnect.com/sessions#full-stack-angular-2\]](http://angularconnect.com/sessions#full-stack-angular-2) in October for an update.

Editors

Which editors will you use most often when hacking on Angular 2? (choose multiple)

| | | |
|----------|-------|-----|
| Webstorm | 43.0% | 907 |
| Sublime | 39.0% | 822 |
| VS Code | 30.1% | 634 |
| Atom | 25.0% | 528 |
| Other | 11.2% | 237 |
| VIM | 10.7% | 226 |

Emacs 1.6% 33

Analysis

It is interesting to see the most feature packed IDE, Webstorm, in the lead. There are many “enterprise” developers that use Java or .NET along with Angular and they are used to heavier weight development environments.

On the other end of the spectrum, VIM actually got more responses than I thought it would. Our friend Mike Hartington from Ionic is an avid VIM user and thinks it works really well with Angular 2 and TypeScript.

Most of the Angular core team has started to use Visual Studio Code. If you haven’t tried it yet, Visual Studio Code has a Sublime-like feel and it works really well right out of the box with TypeScript.

"Other" Editors

The numbers for Webstorm and Visual Studio Code would even be bigger if you combined them with their “sister” products that people wrote in (ex. IntelliJ and RubyMine with Webstorm and VS.NET with Visual Studio Code).

A half dozen developers also strongly recommended Brackets [<http://brackets.io>].

Build Tools

Which build tools do you plan on using when hacking on Angular 2? (choose multiple)

| | | |
|------------|-------|------|
| Gulp | 72.4% | 1464 |
| Grunt | 39.5% | 799 |
| Webpack | 23.6% | 478 |
| SystemJS | 17.0% | 344 |
| Browserify | 16.3% | 330 |
| JSPM | 12.8% | 258 |
| Other | 3.1% | 63 |
| Broccoli | 2.3% | 46 |

Analysis

Clearly Gulp [<http://gulpjs.com>] is the winner here, but it should be noted that most of these tools can be used at the same time and are not mutually exclusive. We were not surprised to see Webpack [<http://webpack.github.io>] have such a strong showing.

Also note that since JSPM [<http://jspm.io>] uses SystemJS [<https://github.com/systemjs/systemjs>] behind the scenes, we can effectively add it to all responses with JSPM that didn't already have SystemJS. That would put SystemJS usage on par with Webpack.

"Other" Build Tools

There were many write in submissions for simply using npm scripts. Other options submitted by more than five people included Gradle [<https://gradle.org>], Meteor's build system [<https://www.meteor.com/isobuild>], RequireJS [<http://requirejs.org>] and Bower [<http://bower.io>].

Frameworks

What frameworks other than Angular 1.x do you use regularly today? (choose multiple)

| | | |
|--------------|-------|------|
| jQuery | 54.7% | 1111 |
| React | 26.8% | 545 |
| Only Angular | 24.2% | 491 |
| Backbone | 12.6% | 255 |
| Meteor | 9.7% | 196 |
| Other | 8.3% | 168 |
| Sails | 3.8% | 77 |
| Ember | 3.8% | 77 |

Analysis

As much as we like to think about the future of the web, the fact is that everyone still has to support older browsers and no library does that better than jQuery. Even so, it is amazing to see how predominant jQuery still is with Angular developers today.

Also interesting to see more than 1/4th of Angular developers also using React. There is a lot of hype around React, but it is based on some really good ideas. You

could argue that number is going to increase over the next year as React becomes even more popular or you could argue it will decrease since Angular 2 has adopted many of the best concepts from React (and thus less motivation for Angular developers to go outside Angular).

"Other" Frameworks

If we could have changed one question on this survey this would have been it. We forgot to include 4 frameworks that a number of people wrote in about:

- Ionic [<http://ionicframework.com>] — An extremely popular library for building hybrid mobile/web apps that uses Angular. The Ionic team are huge fans of Angular 2 and are one of the only groups writing production-level code on top of Angular 2 today.
- Dart [<https://www.dartlang.org>] — I am sure someone outside Google uses Dart, but I just haven't met that person yet. In any case, it will be interesting to see if Dart remains popular inside Google given the decision not to include the Dart VM in Chrome [<http://arc.applause.com/2015/03/27/google-dart-virtual-machine-chrome/>] and the fact that Angular 2 with TypeScript addresses some of the problems that Dart is trying to solve.
- Polymer [<https://www.polymer-project.org/1.0>] — Unlike Dart, I have actually

met a number of groups that are using Polymer in the wild. Polymer will likely never be as popular as Angular, but as modern browsers become more prevalent and web component standards solidify, there will be a niche for Polymer.

- Aurelia [<http://aurelia.io>] — A framework similar to Angular created by a former Angular core team member.

Features

Which Angular 2 features are you most looking forward to? (choose multiple)

| | | |
|------------------|-------|------|
| Change Detection | 65.0% | 1298 |
| Web Components | 57.9% | 1157 |
| Zone.js | 53.1% | 1060 |
| Component Router | 42.3% | 845 |
| DI updates | 39.4% | 788 |
| Server Rendering | 37.1% | 742 |
| i18n | 29.2% | 583 |

Animation updates 25.7% 513

Other 3.6% 72

Analysis

As I wrote about last year [<https://medium.com/@jeffwhelpley/screw-you-angular-62b3889fd678>], many developers love to hate Angular. In that article I wrote:

The typical anti-Angular rant usually includes one or more of the following gripes:

- 1. Subpar features (ex. routing, model layer)*
- 2. Missing features (ex. server rendering, async loading)*
- 3. Cumbersome and/or confusing APIs (ex. injection arrays, service/factory/provider, directives)*
- 4. Fundamental design decisions (ex. working off the DOM, dirty checking vs KVO vs Virtual DOM, mutable data vs immutable data).*

Let's match up these Angular 1 issues with the top Angular 2 features that people are excited about:

1. Change Detection

Change detection in Angular 2 completely eliminates most of the fundamental design decision issues from Angular 1 (issue #4 above). Right out of the box, Angular 2 change detection is faster than Angular 1 [<https://www.youtube.com/watch?v=jvKGQSfQf10&feature=youtu.be&t=330>]. You can see evidence of this in a number of experiments that developers are starting to come out with like this recent blog post from the Meteor team [<http://info.meteor.com/blog/comparing-performance-of-blaze-react-angular-meteor-and-angular-2-with-meteor>]. You can also use immutable or observable data as you wish to make your app even faster. In addition, the design of change detection allows for unidirectional data flow which makes your app easier to reason about. In short, the developer has full control over how change detection works in Angular 2 and can optimize it for their specific situation. It's no wonder this is the top feature people are excited about.

2. Web Components

To be quite frank, we don't think most developers care about Web Components. When we talk to developers about this, they talk more about the component-driven design of Angular 2 rather than integration with actual Web Components. Building your web app with a highly structured component tree in Angular 2 solves many of

the issues with cumbersome and/or confusing APIs in Angular 1 (issue #3).

3. Zone.js

This library has many uses in Angular 2 core, but the most notable benefit is that it eliminates the need for developers to use \$scope.\$apply()

[https://www.youtube.com/watch?v=3IqtmUscE_U&feature=youtu.be&t=934] whenever they change data outside the context of Angular. This is huge.

4. Component Router

When it is ready, the Component Router will address one of the biggest subpar features of Angular 1 (issue #1). It will also enable the asynchronous loading of different components (issue #2). We talked with Brian Ford about the Component Router on Angular Air [https://www.youtube.com/watch?feature=player_embedded&v=ZgXbk9NawJw].

5. Dependency Injection

One of the biggest noticeable impacts of the Angular 2 DI system is that Angular no longer needs to call `toString()` on a function to get the arguments as string tokens. This means no more minification issues with Angular and no need to

manually declare the string tokens with an array (i.e. issue #3). Overall, the DI interface is much cleaner and more flexible. Pascal wrote a great blog post about the Angular 2 DI system [<http://blog.thoughttram.io/angular/2015/05/18/dependency-injection-in-angular-2.html>] that explains more.

6. Server Rendering

This is one of the biggest missing features from Angular 1 (issue #2) and it is going to be awesome. We explain the motivations behind this feature and how we are building it into Angular 2 in our recent AngularU talk [<https://www.youtube.com/watch?v=0wvZ7gakqV4>].

"Other" Features

There were a couple other notable features submitted including:

- Native integration (both NativeScript [<https://www.nativescript.org/blog/details/angular-2.0-running-in-a-native-mobile-app-using-nativescript>] and React Native [<https://github.com/angular/react-native-renderer>])
- Material Design integration

Demographics

What is your Angular 1.x experience level? (choose one)

| | | |
|------------------------------|-------|-----|
| Novice / beginner | 13.9% | 292 |
| Experienced (simple apps) | 27.6% | 581 |
| Advanced (1+ large prod app) | 37.2% | 784 |
| Expert | 21.3% | 448 |

Analysis

This survey was originally intended for more advanced Angular developers, so we were happy to see that a majority of the responders (58.7%) came from the Advanced or Expert buckets. However, since we changed the focus from “what are people using” to “what do people think they will use”, it was also great to get a healthy showing from less experienced Angular developers. As a result, we feel

that this survey is a pretty good representation of the Angular community as a whole.

Final Word

A really big thank you to the thousands of Angular developers that participated in this survey. Also thanks to the dozens of friends that reviewed this post and provided feedback. In particular, thanks to Brad Green for giving us the chance to do a guest post on the official Angular blog.

We found the results extremely interesting and we hope that this blog post helps everyone in the community get a sense of what other developers are thinking about Angular 2 so far. That said, keep in mind that the responses to these same questions will likely be quite different a year from now after Angular 2 has been released and

developers are using it in production. We will have to do a follow up to this survey at that time to see how sentiment has changed.

Jeff Whelpley | [GetHuman \[https://gethuman.com\]](https://gethuman.com) | [@jeffwhelpley](https://twitter.com/jeffwhelpley)
[\[https://twitter.com/jeffwhelpley\]](https://twitter.com/jeffwhelpley)

Patrick Stapleton | [Angular Class \[https://angularclass.com\]](https://angularclass.com) | [@gdi2290](https://twitter.com/gdi2290)
[\[https://twitter.com/gdi2290\]](https://twitter.com/gdi2290)

Posted 3 weeks ago by [Jeff Whelpley](#)



Add a comment

4 weeks ago **Angular 1 and Angular 2 integration: the path to seamless upgrade**

Have an existing Angular 1 application and are wondering about upgrading to Angular 2? Well, read on and learn about our plans to support incremental upgrades.

Summary

Good news!

- We're enabling mixing of Angular 1 and Angular 2 in the same application.
- You can mix Angular 1 and Angular 2 components in the same view.
- Angular 1 and Angular 2 can inject services across frameworks.

- Data binding works across frameworks.

Why Upgrade?

Angular 2 provides many benefits over Angular 1 including dramatically better performance, more powerful templating, lazy loading, simpler APIs, easier debugging, even more testable and much more. Here are a few of the highlights:

Better performance

We've focused across many scenarios to make your apps snappy. 3x to 5x faster on initial render and re-render scenarios.

- Faster change detection through monomorphic JS calls
- Template precompilation and reuse
- View caching
- Lower memory usage / VM pressure
- Linear (blindingly-fast) scalability with observable or immutable data structures
- Dependency injection supports incremental loading

More powerful templating

- Removes need for many directives
- Statically analyzable - future tools and IDEs can discover errors at development time instead of run time
- Allows template writers to determine binding usage rather than hard-wiring in the directive definition

Future work

We've decoupled Angular 2's rendering from the DOM. We are actively working on supporting the following other capabilities that this decoupling enables:

- **Server-side rendering.** Enables super-fast initial render and web-crawler support.
- **Web Workers.** Move your app and most of Angular to a Web Worker thread to keep the UI smooth and responsive at all times.
- **Native mobile UI.** We're enthusiastic about supporting the Web Platform in mobile apps. At the same time, some teams want to deliver fully native UIs on their iOS and Android mobile apps.
- **Compile as build step.** Angular apps parse and compile their HTML templates. We're working to speed up initial rendering by moving the compile step into your build process.

Angular 1 and 2 running together

Angular 2 offers dramatic advantages over Angular 1 in performance, simplicity, and flexibility. We're making it easy for you to take advantage of these benefits in your existing Angular 1 applications by letting you seamlessly mix in components and services from Angular 2 into a single app. By doing so, you'll be able to upgrade an application one service or component at a time over many small commits.

For example, you may have an app that looks something like the diagram below. To get your feet wet with Angular 2, you decide to upgrade the left nav to an Angular 2 component. Once you're more confident, you decide to take advantage of Angular 2's rendering speed for the scrolling area in your main content area.

[\[http://2.bp.blogspot.com/-fHqaW0OSpkc/VdyH89acxDI/AAAAAAB0Ik/cfoXHUMDa14/s1600/Screen%2BShot%2B2015-08-25%2Bat%2B8.20.05%2BAM.png\]](http://2.bp.blogspot.com/-fHqaW0OSpkc/VdyH89acxDI/AAAAAAB0Ik/cfoXHUMDa14/s1600/Screen%2BShot%2B2015-08-25%2Bat%2B8.20.05%2BAM.png)

For this to work, four things need to interoperate between Angular 1 and Angular 2:

- Dependency injection
- Component nesting
- Transclusion
- Change detection

To make all this possible, we're building a library named ng-upgrade. You'll include ng-upgrade and Angular 2 in your existing Angular 1 app, and you'll be able to mix and match at will.

You can find full details and pseudocode in the original [upgrade design doc](#) [\[https://docs.google.com/document/d/1xvBZoFuNq9hsgRhPPZOJC-Z48AHEbIBPIOCBTSD8m0Y\]](https://docs.google.com/document/d/1xvBZoFuNq9hsgRhPPZOJC-Z48AHEbIBPIOCBTSD8m0Y) or read on for an overview of the details on how this works. In future posts, we'll walk through specific examples of upgrading Angular 1 code to Angular 2.

Dependency Injection

First, we need to solve for communication between parts of your application. In Angular, the most common pattern for calling another class or function is through dependency injection. Angular 1 has a single root injector, while Angular 2 has a [hierarchical injector](#) [\[http://victorsavkin.com/post/126514197956/dependency-injection-in-angular-1-and-angular-2\]](http://victorsavkin.com/post/126514197956/dependency-injection-in-angular-1-and-angular-2) . Upgrading services one at a time implies that the two injectors need to be able to provide instances from each other.

The ng-upgrade library will automatically make all of the Angular 1 injectables available in Angular 2. This means that your Angular 1 application services can now be injected anywhere in Angular 2 components or services.

Exposing an Angular 2 service into an Angular 1 injector will also be supported, but will require that you to provide a simple mapping configuration.

The result is that services can be easily moved one at a time from Angular 1 to Angular 2 over independent commits and communicate in a mixed-environment.

Component Nesting and Transclusion

In both versions of Angular, we define a component as a directive which has its own template. For incremental migration, you'll need to be able to migrate these components one at a time. This means that ng-upgrade needs to enable components from each framework to nest within each other.

To solve this, ng-upgrade will allow you to wrap Angular 1 components in a facade so that they can be used in an Angular 2 component. Conversely, you can wrap Angular 2 components to be used in Angular 1. This will fully work with transclusion in Angular 1 and its analog of content-projection in Angular 2.

In this nested-component world, each template is fully owned by either Angular 1 or Angular 2 and will fully follow its syntax and semantics. This is not an emulation mode which mostly looks like the other, but an actual execution in each framework, depending on the type of component. This means that components which are upgraded to Angular 2 will get all of the benefits of Angular 2, and not just better syntax.

This also means that an Angular 1 component will always use Angular 1 Dependency Injection, even when used in an Angular 2 template, and an Angular 2 component will always use Angular 2 Dependency Injection, even when used in an Angular 1 template.

Change Detection

Mixing Angular 1 and Angular 2 components implies that Angular 1 scopes and Angular 2 components are interleaved. For this reason, ng-upgrade will make sure that the change detection (Scope digest in Angular 1 and Change Detectors in Angular 2) are interleaved in the same way to maintain a predictable evaluation order of expressions.

ng-upgrade takes this into account and bridges the scope digest from Angular 1 and change detection in Angular 2 in a way that creates a single cohesive digest cycle spanning both frameworks.

Typical application upgrade process

Here is an example of what an Angular 1 project upgrade to Angular 2 may look like.

1. Include the Angular 2 and ng-upgrade libraries with your existing application
2. Pick a component which you would like to migrate
 - a. Edit an Angular 1 directive's template to conform to Angular 2 syntax
 - b. Convert the directive's controller/linking function into Angular 2 syntax/semantics
 - c. Use ng-upgrade to export the directive (now a Component) as an Angular 1 component (this is needed if you wish to call the new Angular 2 component from an Angular 1 template)

3. Pick a service which you would like to migrate
 - a. Most services should require minimal to no change.
 - b. Configure the service in Angular 2
 - c. (optionally) re-export the service into Angular 1 using ng-upgrade if it's still used by other parts of your Angular 1 code.
4. Repeat doing step #2 and #3 in an order convenient for your application development
5. Once no more services/components need to be converted drop the top level Angular 1 bootstrap and replace with Angular 2 bootstrap.

Note that each individual change can be checked in separately and the application continues working letting you continue to release updates as you wish.

We are not planning on adding support for allowing non-component directives to be usable on both sides. We think most of the non-component directives are not needed in Angular 2 as they are supported directly by the new template syntax (i.e. ng-click vs (click))

Q&A

I heard Angular 2 doesn't support 2-way bindings. How will I replace them?

Actually, Angular 2 supports two way data binding and ng-model, though with slightly different syntax.

When we set out to build Angular 2 we wanted to fix issues with the Angular digest cycle. To solve this we chose to create a unidirectional data-flow for change detection. At first it was not clear to us how the two way forms data-binding of ng-model in Angular 1 fits in, but we always knew that we had to make forms in Angular 2 as simple as forms in Angular 1.

After a few iterations we managed to fix what was broken with multiple digests and still retain the power and simplicity of ng-model in Angular 1.

Two way data-binding has a new syntax: [(property-name)]= "expression" to make it explicit that the expression is bound in both directions. Because for most scenarios this is just a small syntactic change we expect easy migration.

As an example, if in Angular 1 you have:

```
<input type="text" ng-model="model.name" />
```

You would convert to this in Angular 2:

```
<input type="text" [(ng-model)]="model.name" />
```

What languages can I use with Angular 2?

Angular 2 APIs fully support coding in today's JavaScript (ES5), the next version of JavaScript (ES6 or ES2015), TypeScript, and Dart.

While it's a perfectly fine choice to continue with today's JavaScript, we'd like to suggest that you explore ES6 and TypeScript (which is a superset of ES6) as they provide dramatic improvements to your productivity.

ES6 provides much improved syntax and standards for common libraries like promises and modules. TypeScript gives you dramatically better code navigation, automated refactoring in IDEs, documentation, finding errors, and more.

Both ES6 and TypeScript are easy to adopt as they are supersets of today's ES5. This means that all your existing code is valid and you can add their features a little at a time.

What should I do with \$watch in our codebase?

In order to gain speed and predictability, in Angular 2 you specify watch expressions declaratively in the HTML template or in annotations on your component directives. One additional benefit from this is that Angular 2 applications can be safely minified/obfuscated for smaller payload.

For any scenarios that don't fit with these mechanisms, you can take advantage of observables -- ES7 Observables as in Rx.js for JavaScript or Streams in Dart.

What can I do today to prepare myself for the migration?

Follow the best practices and build your application using components and services in Angular 1 as described in the [AngularJS Style Guide \[https://github.com/johnpapa/angular-styleguide\]](https://github.com/johnpapa/angular-styleguide) .

Wasn't the original upgrade plan to use the new Component Router?

The upgrade plan that we announced at ng-conf 2015 was based on upgrading a whole view at a time and having the Component Router handle communication between the two versions of Angular.

The feedback we received was that while yes, this was incremental, it wasn't incremental enough. We went back and redesigned for the plan as described above.

Are there more details you can share?

Yes! In the [Angular 1 to Angular 2 Upgrade Strategy](https://docs.google.com/document/d/1xvBZoFuNq9hsgRhPPZOJC-Z48AHEbIBPIOCBTSD8m0Y) [https://docs.google.com/document/d/1xvBZoFuNq9hsgRhPPZOJC-Z48AHEbIBPIOCBTSD8m0Y] design doc.

We're working on a series of upcoming posts on related topics including:

- Mapping your Angular 1 knowledge to Angular 2.
- A set of FAQs on details around Angular 2.
- Detailed migration guide with working code samples.

See you back here soon!

Posted 4 weeks ago by [Brad Green](#)



Add a comment

28th May

Angular 1.4.0 - jaracimrman-existence

Angular 1.4.0 has arrived! This is a truly community driven release.

This release brings many feature enhancements and performance improvements, while at the same time introducing as few breaking changes as possible. For apps following best practices, we expect the migration from Angular 1.3 to 1.4 to be smooth and the list of breaking changes is documented in the [migration doc](https://docs.angularjs.org/guide/migration#migrating-from-1-3-to-1-4) [https://docs.angularjs.org/guide/migration#migrating-from-1-3-to-1-4] .

We started work on 1.4 in November last year. The features and issues we concentrated on were guided by an analysis of feedback from developers using the framework in real projects. The planning was done in public, see the [planning spreadsheet](#) [https://drive.google.com/open?]

id=1F4JmM25GeaVWLv7oaJrmfFZoE8ioepqNZQKd6xx_Jc4&authuser=0] and [video of the planning meeting](#) [https://www.youtube.com/watch?v=Uae9_8aFo-o] . The weekly meeting minutes [are available online](#) [https://docs.google.com/document/d/1xKEbydyUEOQ_gTbcbxy_k2myctG8EiVbeMgLgXTxlc0/edit] .

New Features

In over 400 commits, we have continually improved the docs, fixed more than 100 bugs, and added over 30 features. Here is a rundown of the new things you can benefit from in this release.

Animation

Matias completely refactored animations, giving it more powerful features and squashing tons of edge-case bugs at the same time. This code overhaul is backwards compatible, except for a small number of [documented api changes](#) [https://docs.angularjs.org/guide/migration#migrating-from-1-3-to-1-4] . This refactoring makes it possible to imperatively control CSS-based transitions/keyframes via a new service called **\$animateCss**. We can now also animate elements across pages via animation anchoring.

- Complete refactor of internal animation code ([c8700f04](#) [https://github.com/angular/angular.js/commit/c8700f04fb6fb5dc21ac24de8665c0476d6db5ef])
- CSS-driven Javascript animations ([see docs](#) [https://code.angularjs.org/commit/docs/api/ngAnimate#css-js-animations-together])
- Better support for animation callback events ([see docs](#) [https://docs.angularjs.org/api/ngAnimate#callbacks-and-promises])
- Move elements between views with animation anchoring ([see docs](#) [https://code.angularjs.org/commit/docs/api/ngAnimate#animation-anchoring-via-ng-animate-ref])
- Provide support for animations on elements outside of **\$rootElement** via **\$animate.pin()** ([e41faaa2](#) [https://github.com/angular/angular.js/commit/e41faaa2a155a42bcc66952497a6f33866878508])

\$http

Pawel did some great work fixing outstanding issues in the \$http service and also implemented a mechanism for providing custom URL parameter serialization, so now it is easy to connect to end-points that expect parameters to follow the jQuery-style parameter serialization.

- support custom params serializers (including a new jQuery-style serializer) ([6c8464ad](#) [https://github.com/angular/angular.js/commit/6c8464ad14dd308349f632245c1a064c9aae242a] , [#3740](#)

[\[https://github.com/angular/angular.js/issues/3740\]](https://github.com/angular/angular.js/issues/3740) , [#7429](https://github.com/angular/angular.js/issues/7429) [\[https://github.com/angular/angular.js/issues/7429\]](https://github.com/angular/angular.js/issues/7429) , [#9224](https://github.com/angular/angular.js/issues/9224) [\[https://github.com/angular/angular.js/issues/9224\]](https://github.com/angular/angular.js/issues/9224) , [#11461](https://github.com/angular/angular.js/issues/11461) [\[https://github.com/angular/angular.js/issues/11461\]](https://github.com/angular/angular.js/issues/11461))

- provide a config object as an argument to header functions ([d435464c](https://github.com/angular/angular.js/commit/d435464c51d3912f56cfc830d86bfc64a1578327) [\[https://github.com/angular/angular.js/commit/d435464c51d3912f56cfc830d86bfc64a1578327\]](https://github.com/angular/angular.js/commit/d435464c51d3912f56cfc830d86bfc64a1578327) , [#7235](https://github.com/angular/angular.js/issues/7235) [\[https://github.com/angular/angular.js/issues/7235\]](https://github.com/angular/angular.js/issues/7235) , [#10622](https://github.com/angular/angular.js/issues/10622) [\[https://github.com/angular/angular.js/issues/10622\]](https://github.com/angular/angular.js/issues/10622))

Internationalization

Chirayu worked hard on improving i18n support for Angular apps. The first piece of this work was to provide a new **ngMessageFormat** module that supports the ICU MessageFormat interpolation syntax.

- extend interpolation with MessageFormat like syntax ([1e58488a](https://github.com/angular/angular.js/commit/1e58488ad65abf7031bab5813523bb9d86dbd28c) [\[https://github.com/angular/angular.js/commit/1e58488ad65abf7031bab5813523bb9d86dbd28c\]](https://github.com/angular/angular.js/commit/1e58488ad65abf7031bab5813523bb9d86dbd28c) , [#11152](https://github.com/angular/angular.js/issues/11152) [\[https://github.com/angular/angular.js/issues/11152\]](https://github.com/angular/angular.js/issues/11152))
- Add **FIRSTDAYOFWEEK** and **WEEKENDRANGE** properties in the ngLocale data ([3d149c7f](https://github.com/angular/angular.js/commit/3d149c7f20ffabab5a635af9ddcfc7105112ab4a) [\[https://github.com/angular/angular.js/commit/3d149c7f20ffabab5a635af9ddcfc7105112ab4a\]](https://github.com/angular/angular.js/commit/3d149c7f20ffabab5a635af9ddcfc7105112ab4a))

Compiler related

To complement our push towards more use of **controller as** syntax in our component directives, Caitlin added support for binding a directive's element attributes straight onto the directive's controller.

- allow using **bindToController** as object, support both new/isolate scopes ([35498d70](https://github.com/angular/angular.js/commit/35498d7045ba9138016464a344e2c145ce5264c1) [\[https://github.com/angular/angular.js/commit/35498d7045ba9138016464a344e2c145ce5264c1\]](https://github.com/angular/angular.js/commit/35498d7045ba9138016464a344e2c145ce5264c1) , [#10420](https://github.com/angular/angular.js/issues/10420) [\[https://github.com/angular/angular.js/issues/10420\]](https://github.com/angular/angular.js/issues/10420) , [#10467](https://github.com/angular/angular.js/issues/10467) [\[https://github.com/angular/angular.js/issues/10467\]](https://github.com/angular/angular.js/issues/10467))

Cookies

Shahar had stepped up to implement the much needed overhaul of the **ngCookies** module. The **\$cookieStore** has been deprecated and its functionality moved over to the **\$cookies** service, which now has a much cleaner interface. He also implemented the frequently requested feature to set cookie options for individual cookies as well as configuring the defaults via **\$cookiesProvider**.

- **\$cookies:**
 - allow passing cookie options ([92c366d2](https://github.com/angular/angular.js/commit/92c366d205da36ec26502aded23db71a6473dad7) [\[https://github.com/angular/angular.js/commit/92c366d205da36ec26502aded23db71a6473dad7\]](https://github.com/angular/angular.js/commit/92c366d205da36ec26502aded23db71a6473dad7) , [#8324](https://github.com/angular/angular.js/issues/8324) [\[https://github.com/angular/angular.js/issues/8324\]](https://github.com/angular/angular.js/issues/8324) , [#3988](https://github.com/angular/angular.js/issues/3988) [\[https://github.com/angular/angular.js/issues/3988\]](https://github.com/angular/angular.js/issues/3988) , [#1786](https://github.com/angular/angular.js/issues/1786) [\[https://github.com/angular/angular.js/issues/1786\]](https://github.com/angular/angular.js/issues/1786) , [#950](https://github.com/angular/angular.js/issues/950) [\[https://github.com/angular/angular.js/issues/950\]](https://github.com/angular/angular.js/issues/950))
 - move logic into \$cookies and deprecate \$cookieStore ([38fbe3ee](https://github.com/angular/angular.js/commit/38fbe3ee8370fc449b82d80df07b5c2ed2cd5fbe) [\[https://github.com/angular/angular.js/commit/38fbe3ee8370fc449b82d80df07b5c2ed2cd5fbe\]](https://github.com/angular/angular.js/commit/38fbe3ee8370fc449b82d80df07b5c2ed2cd5fbe) , [#6411](https://github.com/angular/angular.js/issues/6411) [\[https://github.com/angular/angular.js/issues/6411\]](https://github.com/angular/angular.js/issues/6411) , [#7631](https://github.com/angular/angular.js/issues/7631) [\[https://github.com/angular/angular.js/issues/7631\]](https://github.com/angular/angular.js/issues/7631))
- **\$cookiesProvider:** provide path, domain, expires and secure options ([53c66369](https://github.com/angular/angular.js/commit/53c663699126815eabc2a3bc1e3bafc8b3874268) [\[https://github.com/angular/angular.js/commit/53c663699126815eabc2a3bc1e3bafc8b3874268\]](https://github.com/angular/angular.js/commit/53c663699126815eabc2a3bc1e3bafc8b3874268))

Form Related

Matias was at it again, this time adding dynamic message support to the ngMessages directive, which makes it much easier to display messages requested at runtime from the server.

- **ngMessages:** provide support for dynamic message resolution ([c9a4421f](https://github.com/angular/angular.js/commit/c9a4421fc3c97448527eade1f42eb2f487ec2e0) [\[https://github.com/angular/angular.js/commit/c9a4421fc3c97448527eade1f42eb2f487ec2e0\]](https://github.com/angular/angular.js/commit/c9a4421fc3c97448527eade1f42eb2f487ec2e0) , [#10036](https://github.com/angular/angular.js/issues/10036) [\[https://github.com/angular/angular.js/issues/10036\]](https://github.com/angular/angular.js/issues/10036) , [#9338](https://github.com/angular/angular.js/issues/9338) [\[https://github.com/angular/angular.js/issues/9338\]](https://github.com/angular/angular.js/issues/9338))

ngOptions was completely refactored to allow a number of annoying bugs to be fixed, it also now supports disabling an option using the **`disabled when`** syntax, thanks to Stephen Barker.

- **ngOptions:** add support for disabling an option ([da9eac86](https://github.com/angular/angular.js/commit/da9eac8660343b1cd9fdcf9d2d1bda06067142d7) [\[https://github.com/angular/angular.js/commit/da9eac8660343b1cd9fdcf9d2d1bda06067142d7\]](https://github.com/angular/angular.js/commit/da9eac8660343b1cd9fdcf9d2d1bda06067142d7) , [#638](https://github.com/angular/angular.js/issues/638) [\[https://github.com/angular/angular.js/issues/638\]](https://github.com/angular/angular.js/issues/638) , [#11017](https://github.com/angular/angular.js/issues/11017) [\[https://github.com/angular/angular.js/issues/11017\]](https://github.com/angular/angular.js/issues/11017))

Shahar added improved support for specifying the timezone on date/time input elements that are using **ngModel**.

- **ngModel:** support conversion to timezone other than UTC ([0413bee8](https://github.com/angular/angular.js/commit/0413bee8cc563a6555f8d42d5f183f6fbefc7350) [\[https://github.com/angular/angular.js/commit/0413bee8cc563a6555f8d42d5f183f6fbefc7350\]](https://github.com/angular/angular.js/commit/0413bee8cc563a6555f8d42d5f183f6fbefc7350) , [#11005](https://github.com/angular/angular.js/issues/11005) [\[https://github.com/angular/angular.js/issues/11005\]](https://github.com/angular/angular.js/issues/11005))

jQuery related

Michel Boudreau worked with Michał to allow us to specify exactly which version of jQuery (if any) we want Angular to use. This option also allows developers to instruct Angular to always use [jqLite](https://docs.angularjs.org/api/ng/function/angular.element) [\[https://docs.angularjs.org/api/ng/function/angular.element\]](https://docs.angularjs.org/api/ng/function/angular.element) even if jQuery is present on the page, which can significantly improve performance for applications that are doing a lot of DOM manipulation via Angular's templating engine.

- **ng-jq:** adds the ability to force jqLite or a specific jQuery version ([09ee82d8](https://github.com/angular/angular.js/commit/09ee82d84dcbea4a6e8d85903af82dcd087a78a7) [\[https://github.com/angular/angular.js/commit/09ee82d84dcbea4a6e8d85903af82dcd087a78a7\]](https://github.com/angular/angular.js/commit/09ee82d84dcbea4a6e8d85903af82dcd087a78a7))

Accessibility

Marcy Sutton continued to improve the accessibility of apps built with Angular with new features in the ngAria module:

- automatically add button role to **ngClick** ([bb365070](https://github.com/angular/angular.js/commit/bb365070a3ed7c2d26056d378ab6a8ef493b23cc) [\[https://github.com/angular/angular.js/commit/bb365070a3ed7c2d26056d378ab6a8ef493b23cc\]](https://github.com/angular/angular.js/commit/bb365070a3ed7c2d26056d378ab6a8ef493b23cc) , [#9254](https://github.com/angular/angular.js/issues/9254) [\[https://github.com/angular/angular.js/issues/9254\]](https://github.com/angular/angular.js/issues/9254) , [#10318](https://github.com/angular/angular.js/issues/10318) [\[https://github.com/angular/angular.js/issues/10318\]](https://github.com/angular/angular.js/issues/10318))
- automatically add roles to custom inputs ([29cdaee2](https://github.com/angular/angular.js/commit/29cdaee2b6e853bc3f8882a00661698d146ecd18) [\[https://github.com/angular/angular.js/commit/29cdaee2b6e853bc3f8882a00661698d146ecd18\]](https://github.com/angular/angular.js/commit/29cdaee2b6e853bc3f8882a00661698d146ecd18) , [#10012](https://github.com/angular/angular.js/issues/10012) [\[https://github.com/angular/angular.js/issues/10012\]](https://github.com/angular/angular.js/issues/10012) , [#10318](https://github.com/angular/angular.js/issues/10318) [\[https://github.com/angular/angular.js/issues/10318\]](https://github.com/angular/angular.js/issues/10318))

Filters

We tweaked and improved a number of the filters. Shahar did some nice work to bring better timezone support to the date/time filters; Tamer Aydın helped to add a start index to the **limitTo** filter; Georgios completely rebuilt the **filter** filter to be able to compare with deep objects; and Quentin improved the support for infinity in the **number** filter.

- **number filter:** display Infinity symbol when number is **Infinity** ([51d67742](https://github.com/angular/angular.js/commit/51d6774286202b55ade402ca097e417e70fd546b) [\[https://github.com/angular/angular.js/commit/51d6774286202b55ade402ca097e417e70fd546b\]](https://github.com/angular/angular.js/commit/51d6774286202b55ade402ca097e417e70fd546b) , [#10421](https://github.com/angular/angular.js/issues/10421) [\[https://github.com/angular/angular.js/issues/10421\]](https://github.com/angular/angular.js/issues/10421))
- **date/time filters:** support conversion to timezone other than UTC ([c6d8512a](https://github.com/angular/angular.js/commit/c6d8512a1d7345516d1bd9a039d81821b9518bff) [\[https://github.com/angular/angular.js/commit/c6d8512a1d7345516d1bd9a039d81821b9518bff\]](https://github.com/angular/angular.js/commit/c6d8512a1d7345516d1bd9a039d81821b9518bff) , [#10858](https://github.com/angular/angular.js/issues/10858) [\[https://github.com/angular/angular.js/issues/10858\]](https://github.com/angular/angular.js/issues/10858))

- **limitTo filter:**

- extend the filter to take a beginning index argument ([aaae3cc4](https://github.com/angular/angular.js/commit/aaae3cc4160417e6dad802ed9d9f6d5471821a87) [<https://github.com/angular/angular.js/commit/aaae3cc4160417e6dad802ed9d9f6d5471821a87>] , [#5355](https://github.com/angular/angular.js/issues/5355) [<https://github.com/angular/angular.js/issues/5355>] , [#10899](https://github.com/angular/angular.js/issues/10899) [<https://github.com/angular/angular.js/issues/10899>])
- ignore limit when invalid ([a3c3bf33](https://github.com/angular/angular.js/commit/a3c3bf3332e5685dc319c46faef882cb6ac246e1) [<https://github.com/angular/angular.js/commit/a3c3bf3332e5685dc319c46faef882cb6ac246e1>] , [#10510](https://github.com/angular/angular.js/issues/10510) [<https://github.com/angular/angular.js/issues/10510>])

- **filter filter:**

- compare object with custom **toString()** to primitive ([f8c42161](https://github.com/angular/angular.js/commit/f8c421617096a8d613f4eb6d0f5b098ee149c029) [<https://github.com/angular/angular.js/commit/f8c421617096a8d613f4eb6d0f5b098ee149c029>] , [#10464](https://github.com/angular/angular.js/issues/10464) [<https://github.com/angular/angular.js/issues/10464>] , [#10548](https://github.com/angular/angular.js/issues/10548) [<https://github.com/angular/angular.js/issues/10548>])
- correctly handle deep expression objects ([f7cf846045](https://github.com/angular/angular.js/commit/f7cf846045b1e2fb39c62e304c61b44d5c805e31) [<https://github.com/angular/angular.js/commit/f7cf846045b1e2fb39c62e304c61b44d5c805e31>] , [#9597](https://github.com/angular/angular.js/pull/9757) [<https://github.com/angular/angular.js/pull/9757>])
- allow array like objects to be filtered ([1b0d0fd8](https://github.com/angular/angular.js/commit/1b0d0fd8d00b42dff798845fe0947d594372613) [<https://github.com/angular/angular.js/commit/1b0d0fd8d00b42dff798845fe0947d594372613>] , [#11782](https://github.com/angular/angular.js/issues/11782) [<https://github.com/angular/angular.js/issues/11782>] , [#11787](https://github.com/angular/angular.js/issues/11787) [<https://github.com/angular/angular.js/issues/11787>])

Testing

- **ngMock:**

- **\$ExceptionHandler:** log errors when rethrowing ([deb3cb4d](https://github.com/angular/angular.js/commit/deb3cb4daef0054457bd9fb8995829fff0e8f1e4) [<https://github.com/angular/angular.js/commit/deb3cb4daef0054457bd9fb8995829fff0e8f1e4>] , [#10540](https://github.com/angular/angular.js/issues/10540) [<https://github.com/angular/angular.js/issues/10540>] , [#10564](https://github.com/angular/angular.js/issues/10564) [<https://github.com/angular/angular.js/issues/10564>])
- **annotations:** cleanup \$inject annotations after each test ([0baa17a3](https://github.com/angular/angular.js/commit/0baa17a3b7ad2b242df2b277b81cebdf75b04287) [<https://github.com/angular/angular.js/commit/0baa17a3b7ad2b242df2b277b81cebdf75b04287>] & [6ec59460](https://github.com/angular/angular.js/commit/6ec5946094ee92b820bbacc886fa2367715e60b4) [<https://github.com/angular/angular.js/commit/6ec5946094ee92b820bbacc886fa2367715e60b4>])
- **\$controller:** allow mock \$controller service to set up controller bindings ([d02d0585](https://github.com/angular/angular.js/commit/d02d0585a086ecd2e1de628218b5a6d85c8fc7bd) [<https://github.com/angular/angular.js/commit/d02d0585a086ecd2e1de628218b5a6d85c8fc7bd>] , [#9425](https://github.com/angular/angular.js/issues/9425) [<https://github.com/angular/angular.js/issues/9425>] , [#11239](https://github.com/angular/angular.js/issues/11239) [<https://github.com/angular/angular.js/issues/11239>])

- **Travis:**

- run unit tests on iOS 8 ([2cdb2016](https://github.com/angular/angular.js/commit/2cdb2016b9d89abfb5ab988b67d5f26f3bf21908) [<https://github.com/angular/angular.js/commit/2cdb2016b9d89abfb5ab988b67d5f26f3bf21908>] , [#11479](https://github.com/angular/angular.js/issues/11479) [<https://github.com/angular/angular.js/issues/11479>])

- **Benchmarks:**

- add **ngModel** benchmarks to **largetable-bp** ([b8dbdb0c5e2cd176c6d94d60f781cfc02e646592](https://github.com/angular/angular.js/commit/b8dbdb0c5e2cd176c6d94d60f781cfc02e646592)), [#11082](https://github.com/angular/angular.js/issues/11082) ([\[https://github.com/angular/angular.js/issues/11082\]](https://github.com/angular/angular.js/issues/11082))

Miscellaneous

Other features and improvements that are part of this release include:

- **angular.merge:**

- provide an alternative to **angular.extend** that merges 'deeply' ([c0498d45feb913c318224ea70b5adf7112df6bac](https://github.com/angular/angular.js/commit/c0498d45feb913c318224ea70b5adf7112df6bac)), [#10507](https://github.com/angular/angular.js/issues/10507) ([\[https://github.com/angular/angular.js/issues/10507\]](https://github.com/angular/angular.js/issues/10507) , [#10519](https://github.com/angular/angular.js/issues/10519) [<https://github.com/angular/angular.js/issues/10519>])

- **angular.Module:**

- add decorator method for creating service decorators ([e57138d7eff1210f99238c475fff57530bf0ab19](https://github.com/angular/angular.js/commit/e57138d7eff1210f99238c475fff57530bf0ab19)), [#11305](https://github.com/angular/angular.js/issues/11305) ([\[https://github.com/angular/angular.js/issues/11305\]](https://github.com/angular/angular.js/issues/11305) , [#11300](https://github.com/angular/angular.js/issues/11300) [<https://github.com/angular/angular.js/issues/11300>])

- **\$anchorScroll:**

- allow scrolling to a specified element ([731c8b5e2d01a44aa91f967f1a6acbadb8005a8b](https://github.com/angular/angular.js/commit/731c8b5e2d01a44aa91f967f1a6acbadb8005a8b)), [#4568](https://github.com/angular/angular.js/issues/4568) ([\[https://github.com/angular/angular.js/issues/4568\]](https://github.com/angular/angular.js/issues/4568) , [#9596](https://github.com/angular/angular.js/issues/9596) [<https://github.com/angular/angular.js/issues/9596>])

- **ngClass:**

- add support for conditional map within an array ([4588e627bb7238b2113241919b948d0e5166c76d](https://github.com/angular/angular.js/commit/4588e627bb7238b2113241919b948d0e5166c76d)), [#4807](https://github.com/angular/angular.js/issues/4807) ([\[https://github.com/angular/angular.js/issues/4807\]](https://github.com/angular/angular.js/issues/4807))

- **\$timeout:**

- allow **fn** to be an optional parameter ([5a60302389162c6ef45f311c1aaa65a00d538c66](https://github.com/angular/angular.js/commit/5a60302389162c6ef45f311c1aaa65a00d538c66)), [#9176](https://github.com/angular/angular.js/issues/9176) ([\[https://github.com/angular/angular.js/issues/9176\]](https://github.com/angular/angular.js/issues/9176))
- pass additional arguments to the callback ([3a4b6b83efdb8051e5c4803c0892c19ceb2cba50](https://github.com/angular/angular.js/commit/3a4b6b83efdb8051e5c4803c0892c19ceb2cba50)), [#10631](https://github.com/angular/angular.js/issues/10631) ([\[https://github.com/angular/angular.js/issues/10631\]](https://github.com/angular/angular.js/issues/10631))

- **\$interval:** pass additional arguments to the callback ([4f1f9cfdb721cf308ca1162b2227836dc1d28388](https://github.com/angular/angular.js/commit/4f1f9cfdb721cf308ca1162b2227836dc1d28388)), [#10632](https://github.com/angular/angular.js/issues/10632) ([\[https://github.com/angular/angular.js/issues/10632\]](https://github.com/angular/angular.js/issues/10632))

[\[https://github.com/angular/angular.js/issues/10632\]](https://github.com/angular/angular.js/issues/10632))

- **\$resource:**
 - include request context in error message ([266bc652](https://github.com/angular/angular.js/commit/266bc6520ba4d188dbc949643def102604f98905) [\[https://github.com/angular/angular.js/commit/266bc6520ba4d188dbc949643def102604f98905\]](https://github.com/angular/angular.js/commit/266bc6520ba4d188dbc949643def102604f98905) , [#11363](https://github.com/angular/angular.js/issues/11363) [\[https://github.com/angular/angular.js/issues/11363\]](https://github.com/angular/angular.js/issues/11363))
- **Scope:**
 - remove history event handler when app is torn down ([d996305b](https://github.com/angular/angular.js/commit/d996305b4470f80fbb1cbddf54b7d10ffbb6ab47) [\[https://github.com/angular/angular.js/commit/d996305b4470f80fbb1cbddf54b7d10ffbb6ab47\]](https://github.com/angular/angular.js/commit/d996305b4470f80fbb1cbddf54b7d10ffbb6ab47) , [#9897](https://github.com/angular/angular.js/issues/9897) [\[https://github.com/angular/angular.js/issues/9897\]](https://github.com/angular/angular.js/issues/9897) , [#9905](https://github.com/angular/angular.js/issues/9905) [\[https://github.com/angular/angular.js/issues/9905\]](https://github.com/angular/angular.js/issues/9905))
- **CommonJS:**
 - The new optional CommonJS support makes using Angular modules in environments like npm and browserify easier.

Performance Improvements

Lucas proposed and implemented a complete rewrite of the Angular expression parser. The new parser is easier to maintain and up to 25% faster. On top of that change we have also speeded up scope watching, the compiler and **ngOptions**.

- **\$parse:**
 - new and more performant parser ([0d42426](https://github.com/angular/angular.js/commit/0d424263ead16635afb582affab2b147f8e71626) [\[https://github.com/angular/angular.js/commit/0d424263ead16635afb582affab2b147f8e71626\]](https://github.com/angular/angular.js/commit/0d424263ead16635afb582affab2b147f8e71626))
- **\$compile:**
 - replace **forEach(controller)** with plain loops ([5b522867](https://github.com/angular/angular.js/commit/5b5228675f67c8f5e04c7183c3ef5e71cb2bf08b) [\[https://github.com/angular/angular.js/commit/5b5228675f67c8f5e04c7183c3ef5e71cb2bf08b\]](https://github.com/angular/angular.js/commit/5b5228675f67c8f5e04c7183c3ef5e71cb2bf08b) , [#11084](https://github.com/angular/angular.js/issues/11084) [\[https://github.com/angular/angular.js/issues/11084\]](https://github.com/angular/angular.js/issues/11084))
 - avoid **.data** when fetching required controllers ([fa0aa839](https://github.com/angular/angular.js/commit/fa0aa83937378cf8fc720c38bcc5c78fc923624e) [\[https://github.com/angular/angular.js/commit/fa0aa83937378cf8fc720c38bcc5c78fc923624e\]](https://github.com/angular/angular.js/commit/fa0aa83937378cf8fc720c38bcc5c78fc923624e))
- **ngOptions:**
 - only perform deep equality check on **ngModel** if using **track by** ([171b9f7f](https://github.com/angular/angular.js/commit/171b9f7f2339ef9047b8526b2c3f36bb58d14feb) [\[https://github.com/angular/angular.js/commit/171b9f7f2339ef9047b8526b2c3f36bb58d14feb\]](https://github.com/angular/angular.js/commit/171b9f7f2339ef9047b8526b2c3f36bb58d14feb) , [#11448](https://github.com/angular/angular.js/issues/11448) [\[https://github.com/angular/angular.js/issues/11448\]](https://github.com/angular/angular.js/issues/11448) , [#11447](https://github.com/angular/angular.js/issues/11447) [\[https://github.com/angular/angular.js/issues/11447\]](https://github.com/angular/angular.js/issues/11447))
 - only watch labels if a **display** expression is specified ([51faaffd](https://github.com/angular/angular.js/commit/51faaffdbcc734c55d52ff6c42b386d5c90207ea) [\[https://github.com/angular/angular.js/commit/51faaffdbcc734c55d52ff6c42b386d5c90207ea\]](https://github.com/angular/angular.js/commit/51faaffdbcc734c55d52ff6c42b386d5c90207ea))

What's next?

There were two features that were pulled out of the 1.4 release. The component helper ([#10007](https://github.com/angular/angular.js/issues/10007) [\[https://github.com/angular/angular.js/issues/10007\]](https://github.com/angular/angular.js/issues/10007)) and the component oriented hierarchical router. The main reason for this decision was that both of these deliverables were not ready for the important task of simplifying the migration path from Angular 1 to Angular 2. Rather than delay the 1.4 release further, we decided to move these two deliverables into the 1.5 release.

The 1.5 planning is already underway, with the Component Router and component helper among the initial seeds. As before, the planning meeting will be published online. In this meeting the areas of focus and prioritization will be picked. At ng-conf we announced that the future focus of the 1.x releases will be migration path to Angular 2, so you can expect many of the deliverables to be in this area.

Thanks

This Angular version is the first to be run by a much broader community oriented team, including many people from outside of the Google Angular team such as [Lucas](https://github.com/lgalfaso) [\[https://github.com/lgalfaso\]](https://github.com/lgalfaso) , [Martin](https://github.com/Narretz) [\[https://github.com/Narretz\]](https://github.com/Narretz) , [Shahar](https://github.com/shahata) [\[https://github.com/shahata\]](https://github.com/shahata) , [Pawel](https://github.com/pkozlowski-opensource) [\[https://github.com/pkozlowski-opensource\]](https://github.com/pkozlowski-opensource) , [Michał](https://github.com/mzgol) [\[https://github.com/mzgol\]](https://github.com/mzgol) , [Georgios](https://github.com/gkalpak) [\[https://github.com/gkalpak\]](https://github.com/gkalpak) and [Jason](https://github.com/jbedard) [\[https://github.com/jbedard\]](https://github.com/jbedard) . These are people who are using Angular on a day-to-day basis in real, often large applications. They have worked tirelessly week after week to help get this release out. On top of these core team members, a number of other people from the community have contributed significant code to this release, or helped to review issues and pull requests. We'd like to thank them all.

Posted 28th May by [Pete Bacon Darwin](#)

Labels: [1.4.0](#), [announcements](#), [release](#)



Add a comment

21st March

Forms in Angular 2

Input handling is an important part of application development. The ng-model directive provided in Angular 1 is a great way to manage input, but we think we can do better. The new Angular Forms module is easier to use and reason about than ng-model, it provides the same conveniences as ng-model, but does not have its drawbacks. In this article we will talk about the design goals for the module and show how to use for common use cases.

Optimizing for real world apps

Let's look at what a simple HelloWorld example would look like in Angular 2:

```
@Component({
  selector: 'form-example'
})
@Template({
  // we are binding the input element to the control object
  // defined in the component's class
  inline: `<input [control]="username">Hello {{username.value}}!`,
  directives: [forms]
})
class FormExample {
  constructor() {
    this.username = new Control('World');
  }
}
```

This example is written using TypeScript 1.5 that supports type and metadata annotations, but it can be easily written in ES6 or ES5. You can find more information on annotations [here](https://docs.google.com/document/d/1uhs-a41dp2z0NLs-QiXYY-rqLGhgjmTf4iwBad2myzY/edit) [https://docs.google.com/document/d/1uhs-a41dp2z0NLs-QiXYY-rqLGhgjmTf4iwBad2myzY/edit]. The example also uses the new template syntax that you can learn about by watching [the keynote](https://www.youtube.com/watch?v=-dMBcqwwYA0) [https://www.youtube.com/watch?v=-dMBcqwwYA0] Misko and Rado gave at NgConf.

And, for comparison sake, here's what we'd write in Angular 1:

```
var module = angular.module("example", []);
```



```
module.controller("FormExample", function() {  
    this.username = "World";  
});  
  
<div ng-controller="FormExample as ctrl">  
    <input ng-model="ctrl.username"> Hello {{ctrl.username}}!  
</div>
```

At the surface Angular 1 example looks simpler than what we have just done in Angular 2. Let's talk about the challenges of Angular 1 approach with real world applications.

- You can not unit test the form behavior without compiling the associated template. This is because the template contains part of the application behavior.
- Though you can do dynamically generated data-driven forms in Angular 1, it is not easy and we want to enable this as a core use case for Angular 2.
- The ng-model directive was built using a generic two-way data-binding which makes it difficult to reason about your template statically. Will go into depth on this topic in a following blog post and describe how it lets us achieve significant performance improvements.
- There was no concept of an atomic form which could easily be validated, or reverted to original state.

Although Angular 2 uses an extra level of indirection, it grants major benefits. The control object decouples form behavior from the template, so that you can test it in isolation. Tests are simpler to write and faster to execute.

Now let's look at a more complex example so that you can see some of these properties.

Forms Mental Model

In Angular 2 working with Forms is broken down to three layers. At the very bottom we can work with HTML elements. On top of that Angular 2 provides the Form Controls API which we have shown in the example above. Finally, Angular 2 will also provide a data-driven forms API which will make building large-scale forms a snap.

[http://1.bp.blogspot.com/-IOIkh95P8wM/VQyN0NuqaVI/AAAAAAAAAGI/BsMWLte3bO8/s1600/forms_diagram.png]

Let's look at how this extra level of indirection benefits us for more realistic examples.

Data-Driven Forms

Let's say we would like to build a form such as this:

[\[http://2.bp.blogspot.com/-9rGZ04kPY3E/VQyNgsZ8s3I/AAAAAAAAAGA/5kD9i_QeP2c/s1600/form_screenshot.png\]](http://2.bp.blogspot.com/-9rGZ04kPY3E/VQyNgsZ8s3I/AAAAAAAAAGA/5kD9i_QeP2c/s1600/form_screenshot.png)

We will have an object model of an Address which needs to be presented to the user. The requirements also include providing correct layout, labels, and error handling. We would write it like this in Angular 2.

(This is proposed API, we would love to take your input on this.)

```
import {forms, required, materialDesign} from 'angular2/forms';
```

```
// Our model
class Address {
  street: string;
  city: string;
  state: string;
  zip: string;
```

```
    residential: boolean;
  }

@Component({
  selector: 'form-example'
})
@Template({
  // Form layout is automatic from the structure
  inline: `<form [form-structure]="form"></form>`
  directives: [forms]
})
class FormExample {
  constructor(fb: FormBuilder) {
    this.address = new Address();

    // defining a form structure and initializing it using
    // the passed in model
    this.form = fb.fromModel(address, [
      // describe the model field, labels and error handling
      {field: 'street', label: 'Street', validator: required},
      {field: 'city', label: 'City', validator: required},
      {field: 'state', label: 'State', size: 2,
        validator: required},
      {field: 'zip', label: 'Zip', size: 5,
        validator: zipCodeValidator},
      {field: 'isResidential', type: 'checkbox',
        label: 'Is residential'}
    ], {
      // update the model every time an input changes
      saveOnUpdate: true,
      // Allow setting different layout strategies
      layoutStrategy: materialDesign
    });
  }
}
```

```
    }  
  }  
  
  function zipCodeValidator(control) {  
    if (! control.value.match(/^\d\d\d\d\d(-\d\d\d\d)?$/)){  
      return {invalidZipCode: true};  
    }  
  }  
}
```

The above example shows how an existing model Address can be turned into a form. The process include describing the fields, labels, and validators in a declarative way in your component. The form HTML structure will be generated automatically based on the layout strategy provided to the builder. This helps keeping consistent look & feel through the application. Finally, it is also possible to control the write through behavior, which allows atomic writes to the model.

We have taken great care to ensure that the forms API is pluggable, allowing you to define custom validators, reuse web-component controls, and define layout and theme.

Form Controls

Although having data driven forms is convenient, sometimes you would like to have full control of how a form is laid out on page. Let's rebuild the same example using the lower level API, which allows you to control the HTML structure in full detail. (This works today in Angular 2 Alpha, but we are also happy to receive your input on improvements we could make.)

```
import {forms, required} from 'angular2/forms';  
  
// An example of typical model  
class Address {  
  street: string;  
  city: string;  
  state: string;  
  zip: string;  
  residential: boolean;  
}
```

```
function zipCodeValidator(control) {
  if (! control.value.match(/\\d\\d\\d\\d\\d(-\\d\\d\\d\\d)?/)){
    return {invalidZipCode: true};
  }
}

@Component({
  selector: 'form-example'
})
@Template({
  inline: `
    // explicitly defining the template of the form
    <form [form]="form">
      Street <input control="street">
        <div *if="form.hasError('street', 'required')">Required</div>

      City <input control="city">
        <div *if="form.hasError('city', 'required')">Required</div>

      State <input control="state" size="2">
        <div *if="form.hasError('state', 'required')">Required</div>

      Zip <input control="zip" size="5">
        <div *if="form.hasError('zip', 'invalidZipCoed')">
          Zip code is invalid
        </div>

      Residential <input control="isResidential" type="checkbox">
    </form>
  `,
  directives: [forms]
})
```

```
class FormExample {
  constructor(fb: FormBuilder) {
    this.address = new Address();

    // defining a form model
    this.form = fb.group({
      street: [this.address.street, required],
      city: [this.address.city, required],
      state: [this.address.city, required],
      zip: [this.address.zip, zipCodeValidator],
      isResidential: [this.address.isResidential]
    });

    this.form.changes.forEach(() => this.form.writeTo(this.address));
  }
}
```

When using the control form API we still define the behavior of the form in the component, but the way the form is laid out is defined in the template. This allows you to implement even the most unusual forms.

Summary

The new Angular 2 Forms module makes building forms easier than ever. It also adds benefits such as consistent look and feel for all forms in your application, easier unit testing, and more predictable behavior.

The API is still work in progress, we welcome any feedback. Please submit your input by filing issues or creating PRs at <http://github.com/angular/angular> [<http://github.com/angular/angular>].

Posted 21st March by [Victor Savkin](#)



Add a comment

18th March

New Angular Releases - 1.3.15 and 1.4.0-beta.6

After a short hiatus due to the excitement of ng-conf, we are excited to announce a new pair of releases.

The 1.3 branch has a new point release [1.3.15](https://github.com/angular/angular.js/blob/master/CHANGELOG.md#1315-locality-filtration-2015-03-17) [locality-filtration](https://github.com/angular/angular.js/blob/master/CHANGELOG.md#1315-locality-filtration-2015-03-17) [https://github.com/angular/angular.js/blob/master/CHANGELOG.md#1315-locality-filtration-2015-03-17] with a bunch of bug fixes.

The 1.4 branch is now closing in on a release candidate with what we hope will be the last beta [1.4.0-beta.6](https://github.com/angular/angular.js/blob/master/CHANGELOG.md#140-beta6-cookie-liberation-2015-03-17) [cookie-liberation](https://github.com/angular/angular.js/blob/master/CHANGELOG.md#140-beta6-cookie-liberation-2015-03-17) [https://github.com/angular/angular.js/blob/master/CHANGELOG.md#140-beta6-cookie-liberation-2015-03-17] .

Some of the highlights of the new beta release include:

- There has been a complete overhaul of `ngCookies` thanks to Shahar Talmi. This has a few breaking changes so be sure to check those out before you upgrade if you are using `$cookies`.
- Also thanks to Shahar you can now set the timezone for a `ngModel`.
- In keeping with their native counterparts, `$timeout` and `$interval` now accept additional parameters, which are passed through to the callback handler.
- `ngMock` now includes new `they` helpers. It also now patches `$controller` to make it easier to create controllers that are already bound to properties, which is useful when testing controllers that are used with `bindToController`.

Thanks as always to everyone who has contributed to these releases:

Amy, Anthony Zotti, Brian Ford, Caitlin Potter, Casey Howard, Ciro Nunes, Dav, Dave Jeffery, Devyn Stott, Diego, Edward Delaporte, Elliot Bentley, Izhaki, Jason Bedard, Josh Kramer, Julie Ralph, Marcin Wosinek, Marcy Sutton, Martin R. Hufsky, Martin Staffa, Matias Niemelä, Michel Boudreau, Pawel Kozlowski, Peter Bacon Darwin, Rouven Weißling, Santi Albo, Shahar Talmi, Steve Mao, b0ri5, bborowin, eeue56, gdi2290, jmarkevicius, robiferentz, rodyhaddad, svershin

Posted 18th March by [Pete Bacon Darwin](#)

Labels: [announcements](#), [release](#)**Add a comment****10th March**

Announcements from ng-conf

We're astounded by the work the ng-conf organizers did this year in creating an amazing experience for all attendees. Thanks for joining us! We all had a wonderful time with you, hearing about your experiences on Angular and getting your input on the future.

In addition to the folks who came in person, more than 8000 people watched from ng-conf extended parties from 150 sites in 29 countries. Thanks for tuning in!

If you haven't had the time to watch through all of the videos from ng-conf last week, here's a summary of what we announced.

In Brief

Angular 2: It is screaming fast, simpler to learn, and you can mix and match it with your existing Angular 1 applications.

Angular 1: Version 1.4 will be out with a release candidate in a week or so with many exciting new features. The theme of 1.5 will be supporting integration with Angular 2.

AtScript is TypeScript: We've been collaborating with the TypeScript team on our proposed productivity extensions to JavaScript and have converged development efforts. Details from the TypeScript team are on [their blog](http://blogs.msdn.com/b/typescript/archive/2015/03/05/angular-2-0-built-on-typescript.aspx) [<http://blogs.msdn.com/b/typescript/archive/2015/03/05/angular-2-0-built-on-typescript.aspx>] . Many thanks to the TypeScript team!

Details

All videos from the talk are available on the [ng-conf 2015 playlist \[https://www.youtube.com/playlist?list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/playlist?list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7) on YouTube. Closed captions are in the works and will be added soon.

In the [Day 1 Keynote \[https://www.youtube.com/watch?v=QHulaj5Zxbl&index=1&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=QHulaj5Zxbl&index=1&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7) , we announced the following:

- We're nearly done with Angular 1.4 which will include better change detection performance, improvements to many APIs, support for the new [router \[https://www.youtube.com/watch?v=vecg70fPDFw&index=3&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=vecg70fPDFw&index=3&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7) , and new [i18n \[https://www.youtube.com/watch?v=iBBkCA1M-mc&index=26&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=iBBkCA1M-mc&index=26&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7) .
- [Angular Material \[https://www.youtube.com/watch?v=Qi31oO5u33U&index=30&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=Qi31oO5u33U&index=30&list=PLOETecp3DkCoNnlhE-7fovYvqwVPrRiY7) is nearing 1.0, contains 31 of the most useful material design UI elements with only a few to go.
- We'll continue building releases in the Angular 1 branch until the vast majority of folks have moved to Angular 2. We want the burden to be on the Angular team to make Angular 2 simple to learn, attractive in its features, and an easy target to migrate to from Angular 1.
- The Angular 1.5 release will start planning immediately after Angular 1.4 ships. We'll take community pull requests, issues, and comments to form its direction. A primary theme will be support to make for easy migration to Angular 2.
- We've had a great partnership with the TypeScript team to date. They've implemented the features we care most about from AtScript. Based on this convergence, we've decided to retire the AtScript name and call it all TypeScript.
- We'll work with them and many other partners to drive these features into standards via the TC39 process.
- We officially support Angular 2 development in today's JavaScript (ECMAScript 5), ES6, TypeScript, and Dart. Angular 2, like Angular 1, will additionally work with other compile-to-JavaScript languages like CoffeeScript and ClojureScript.
- Angular 2 is now in alpha release and you can see a preview of its new website at [angular.io \[http://angular.io/\]](http://angular.io) with its features, quickstart tutorial, and links to resources.
- Angular 2 dramatically simplifies the great developer productivity story we have in Angular 1.
- By improving templating idioms, we've generalized for 35 directives, now part of Angular 1, into a single concept in Angular 2.
- We've eliminated many surprises and corner cases like having to call `$apply` when using external libraries.
- Angular 2 builds on new web standards like ES6 and Web Components. This means that Angular 2 works seamlessly with the libraries also built on these standards, including Polymer, X-Tag, and others.
- We've dramatically improved performance in Angular 2. Change detection is up to 10x faster. We cache view rendering making overall performance in virtual scrolling and re-visiting views incredibly fast.
- Angular 2 is similarly better in terms of memory efficiency -- critical for mobile devices.
- Performance gets even better for developers who use immutable data structures or observable objects.

For details on how this all works, please also check out the [Day 2 Keynote \[https://www.youtube.com/watch?v=dMBcqvvYA0&index=21&list=PLOETecp3DkCoNlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=dMBcqvvYA0&index=21&list=PLOETecp3DkCoNlhE-7fovYvqwVPrRiY7) by Misko and Rado, the [Components talk \[https://www.youtube.com/watch?v=AbunztfV5vU&index=6&list=PLOETecp3DkCoNlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=AbunztfV5vU&index=6&list=PLOETecp3DkCoNlhE-7fovYvqwVPrRiY7) by Kara and Rachael from OpenTable, and [Change Detection Reinvented \[https://www.youtube.com/watch?v=jvKGQSFQf10&index=31&list=PLOETecp3DkCoNlhE-7fovYvqwVPrRiY7\]](https://www.youtube.com/watch?v=jvKGQSFQf10&index=31&list=PLOETecp3DkCoNlhE-7fovYvqwVPrRiY7) from Victor.

We're looking forward to your input on all of this through GitHub, Twitter, and G+. On to the rest of 2015!

Posted 10th March by [Brad Green](#)



Add a comment

25th February

New AngularJS Releases 1.4.0-beta.5 and 1.3.14

The ongoing improvement of AngularJS 1.x continues with another sparkling pair of releases.

- 1.4.0-beta.5 - karmic-stabilization
- 1.3.14 - instantaneous-browserification

See the [CHANGELOG \[https://github.com/angular/angular.js/blob/master/CHANGELOG.md#140-beta5-karmic-stabilization-2015-02-24\]](https://github.com/angular/angular.js/blob/master/CHANGELOG.md#140-beta5-karmic-stabilization-2015-02-24) for more detailed information.

CommonJS support

In both 1.4.0-beta.5 and 1.3.14 we have improved support for using Angular as CommonJS modules with tools like Browserify, by providing entry points in our npm packages that load the module and expose the

appropriate exports for consumption.

If you are using Browserify, you no longer need to provide shims for the core AngularJS modules. You can now write code that looks like:

```
var angular = require('angular');
angular.module('myMod', [
  require('angular-animate'),
  require('angular-mocks/ngMock'),
  require('angular-mocks/ngAnimateMock'),
  require('angular-i18n/en-us')
]);
```

Thanks to [@bclinkinbeard](https://github.com/bclinkinbeard) [<https://github.com/bclinkinbeard>] , [@bendrucker](https://github.com/bendrucker) [<https://github.com/bendrucker>] , [@kentcdodds](https://github.com/kentcdodds) [<https://github.com/kentcdodds>] , [@caitp](https://github.com/caitp) [<https://github.com/caitp>] and [@btford](https://github.com/btford) [<https://github.com/btford>] for their help in putting this together!

Bug Fixes

The 1.3.14 release contains fixes for ngModel, input directives and ngAria.

The 1.4.0-beta.5 release has these fixes along with others for \$http, select and templateRequest.

New Features

In 1.4.0-beta.5 we have added support for disabling options in a select element using ngOptions, extended the limitTo filter so that the limited collection doesn't have to start at the beginning of the collection and ngMessages now supports creating message lists dynamically, which means that you can do clever things such as loading messages on the fly from a server.

Still more to come!

We are still actively developing the 1.4 branch, with a few key new features still to be merged. Once 1.4.0 is released we are going to begin planning the 1.5 development schedule, ***so watch this space!***

Posted 25th February by [Pete Bacon Darwin](#)

Labels: [announcements](#), [release](#)

18th February

Preview of the New Angular Router

We've been working on a new router for both Angular 1 and Angular 2, and while APIs are still in flux, we want your feedback on the new router in 1.x.

You can [read the docs](http://angular.github.io/router/) , browse [the code on GitHub](http://github.com/angular/router) , and install it with `npm install angular-new-router`

There's still lots more work to be done, but we want to know what you think. You can help out by filing [issues and questions on GitHub](https://github.com/angular/router/issues) .

What about ngRoute?

ngRoute isn't going anywhere, but it's unlikely to receive new features.

What's next?

We've been making good progress on Angular 2 – last week we [demoed a todo list app written with Angular 2](https://www.youtube.com/watch?v=uD6Okha_Yj0) at our Mountain View meetup. Next will be writing bindings for Angular 2 for this routing system.

We're also working on a proposal for [making lazy loading easier at the DI level in Angular 1](https://github.com/angular/angular.js/issues/11015) . Once this lands, the new router will be able to help you lazy load parts of your Angular 1 app.

Posted 18th February by [Brian Ford](#)

**Add a comment**

16th December 2014

Planning Angular 1.4

We had a planning meeting last week to decide what will go into AngularJS 1.4. Here is a summary of what happened. You can watch the video of the meeting on YouTube below.

Release Schedule

We decided that the first release (1.4.0) will be in Spring 2015, coinciding with [ng-conf](http://www.ng-conf.org/) on 5th March 2015. In the meantime, we'll continue with 1.3.x releases as usual.

Work Planning

Lucas prepared a [spreadsheet](https://docs.google.com/spreadsheets/d/1ihM8ORf8v38qAbnzJM4TiCqdKeCRz_1VO9JPjSpPWKU/edit?usp=sharing) [\[https://docs.google.com/spreadsheets/d/1ihM8ORf8v38qAbnzJM4TiCqdKeCRz_1VO9JPjSpPWKU/edit?usp=sharing\]](https://docs.google.com/spreadsheets/d/1ihM8ORf8v38qAbnzJM4TiCqdKeCRz_1VO9JPjSpPWKU/edit?usp=sharing) with a list of suggested items for 1.4 from the issues and PRs on GitHub with the most community interest. This focussed on large work items or things that would require breaking changes or considerable new API. The bulk of the meeting was going through these items, deciding whether they should appear in Angular 1.x and, if so, which release and who would own the item.

1.4 Targets

There is a [tracking spreadsheet](https://docs.google.com/spreadsheets/d/1F4JmM25GeaVWLv7oaJrmfFZoE8ioepqNZQKd6xx_Jc4/edit?usp=sharing) [\[https://docs.google.com/spreadsheets/d/1F4JmM25GeaVWLv7oaJrmfFZoE8ioepqNZQKd6xx_Jc4/edit?usp=sharing\]](https://docs.google.com/spreadsheets/d/1F4JmM25GeaVWLv7oaJrmfFZoE8ioepqNZQKd6xx_Jc4/edit?usp=sharing) for the items that are scheduled for 1.4.

The main themes for 1.4 are as follows:

- **Router** - *Brian* - a new router for Angular 1 and 2 - [Progress](https://docs.google.com/document/d/1-DBXTHaec6XH5qx2tKVgrjiLy76_ISrjgJv95RJ4/edit#) [\[https://docs.google.com/document/d/1-DBXTHaec6XH5qx2tKVgrjiLy76_ISrjgJv95RJ4/edit#\]](https://docs.google.com/document/d/1-DBXTHaec6XH5qx2tKVgrjiLy76_ISrjgJv95RJ4/edit#)
- **I18N** - *Chirayu* - provide a first class internationalization story for Angular - [Design Doc](https://docs.google.com/document/d/1mwyOFsAD-bPoXTk3Hthq0CAcGXCUw-BtTJMR4nGTy-0/edit?usp=sharing) [\[https://docs.google.com/document/d/1mwyOFsAD-bPoXTk3Hthq0CAcGXCUw-BtTJMR4nGTy-0/edit?usp=sharing\]](https://docs.google.com/document/d/1mwyOFsAD-bPoXTk3Hthq0CAcGXCUw-BtTJMR4nGTy-0/edit?usp=sharing)
- **Forms** - *Martin* - a fresh look at parsing/formatting/validation to simplify usage and maintenance (while fixing

numerous outstanding issues) - [Design Doc](#) [https://drive.google.com/open?id=1-MhomULgCFOXVsqaGrI7I-cXYMYmJM4BMzgy_anC93o&authuser=0]

- **HTTP** - *Pawel* - improvements to the \$http service, such as serialization, JSON parsing, testing mock DSL
- **Parser** - *Lucas* - performance improvements to \$parse service
- **Documentation** - *Caitlin* - redesign the look of the docs app to use Material Design

In addition, we're planning to include the following notable or breaking changes:

- **\$injector** - *Brian* - throw an error if you redefine a module, to help identify bugs faster ([#1779](#) [<https://github.com/angular/angular.js/issues/1779>])
- **\$compile** - *Igor* - provide extra new module.component helper for defining component type directives more easily ([#10007](#) [<https://github.com/angular/angular.js/issues/10007>]).
- **\$compile** - *Caitlin* - throw an error if non-optional isolated scope mappings are missing their attributes ([#9216](#) [<https://github.com/angular/angular.js/pull/9216>]).
- **Project layout/Modularity** - *Pete* - further partition angular.js into smaller optional modules/files to reduce the non-optional core file size (useful for mobile use cases).

Github Milestones and Labels

Finally we will start developing 1.4.x on the master branch soon. To support this here are the new labels/milestones for on-going development:

Milestones:

1.4.x - use this for issues and pull requests that are accepted and scheduled to be in 1.4

Labels:

branch: 1.2.x (replaces stable: yes)

branch: 1.3.x (replaces stable: no)

branch: 1.4.x (replaces 1.4 - for triaging 1.4.x issues and PRs)

Primary Focus: (new for items that we are focussing on for 1.4 - i.e. the stuff in the tracking spreadsheet)

Other Versions and Backporting

The master branch (i.e. 1.4.x) will now receive the majority of the focus.

The 1.3.x branch will receive version specific fixes backported from master.

The 1.2.x branch will now only receive fixes for security issues and major regressions.

Video

In our continued efforts to be transparent and open about our design and development activities, we published a recording of our planning meeting: https://www.youtube.com/watch?v=Uae9_8aFo-o [https://www.youtube.com/watch?v=Uae9_8aFo-o]

[\[https://www.youtube.com/watch?v=Uae9_8aFo-o\]](https://www.youtube.com/watch?v=Uae9_8aFo-o)

Just the Beginning

The planning we did for 1.4 is just the beginning. In addition to the outline above, we welcome additional suggestions for 1.4 via GitHub. Once we've released 1.4.0, we'll continue with 1.4.x releases that include non-breaking fixes that didn't make it into 1.4.0.

Posted 16th December 2014 by [Brian Ford](#)



Add a comment

20th November 2014

Angular CLA Infrastructure Changes

As of today, we are unifying our Contributor License Agreement (CLA) checking tool with the newly rolled out infrastructure for all Google open source projects. Just as before, a single CLA signature will enable you to contribute to any open source project from Google, but the new CLA bot is much faster, more reliable and can handle corporate CLAs better.

The new CLA signature repository requires a GitHub account to be linked with a Google account. In many but not all cases we were able to link the two accounts automatically. For those contributors where an automated match was not possible, we'll ask you to re-sign the CLA or add your GitHub username to your contact info the next time you send us a PR.

To check your CLA status you can visit <https://cla.developers.google.com/cla> [https://cla.developers.google.com/cla] . To see if your CLA is linked with a GitHub account please click on 'Edit Contact Information' next to a CLA record where you can check and edit your GitHub account name.

Once your GitHub account name is linked correctly, [Googlebot](https://github.com/googlebot) [https://github.com/googlebot] will comment on your future PRs and give it the label 'CLA: yes' (see [this PR](https://github.com/angular/angular.js/pull/10136) [https://github.com/angular/angular.js/pull/10136] for an "exemplary" conversation with Googlebot). If you already have an open PR that hasn't been verified, you will need to comment on it

for Googlebot to re-verify your CLA.

We apologize for the inconvenience, but this stuff is important for the Angular project and the community that depends on it. Thanks and stay awesome!

Posted 20th November 2014 by [Igor Minar](#)

Labels: [announcements](#)



Add a comment