The History of Angular

Dave Gavigan Follow
Apr 3, 2018 · 5 min read

The past, present, and future of Angular

As of this writing, Angular is currently at 5.2 (with release candidates for 6.0).

Angular took a lot of the community by storm when it was first released in 2010. Back then libraries like dojo, backbone, and knockout were at-large when it came to developing large web applications.

Fast forward to 2016, and the core team at Google delivered the successor to AngularJS… Angular2. Or, “Just Angular”. It came with a lot of excitement and criticism.

As a developer who cut their teeth by writing Angular apps since 2012, I've seen this framework and the community around it evolve to what it is today.

2010: The birth of AngularJS

Prior to its release, a Google employee by the name of Miško Hevery, was developing a side project. This side project was to help make building web applications easier for a couple internal projects he was working on. This side project later became known as AngularJS (Angular because of the < > in HTML).

Misko and a few others started to create a few more internal applications with AngularJS, eventually releasing it as an open source project in 2010.

The community got their hands on it and started building amazing applications. The Ionic Framework built on top of Apache Cordova from Ionic (formerly DriftyCo) really put gas on the fire allowing developers to use AngularJS to power their mobile applications as well.

Some of the biggest brands started to incorporate it into their toolset for both their web and mobile app development.

2014 — 2015: The Great Rewrite

Several years after its initial release, the landscape of web development started to change and AngularJS hit a wall.

New advancements and standards in JavaScript emerged, and Angular started to get left behind the bleeding edge. More importantly, the core team had reached a limit to what they could do to improve on the framework for the growing demands.

The team at google and community took what was once a small internal project and pushed it to new frontiers like mobile and large enterprise applications. However, it was never intended or designed for these situations when Miško first created it. With every pull request we modified AngularJS the best we could to fit our needs.

When the core team at Google sought to deliver 2.0, they didn't want to be shackled to their previous design of AngularJS. They wanted to build a framework from the ground up to solve difficult problems when building large and cross-platform applications. That meant a total rewrite.

The framework so many had grown to learn and love was going down the pipes. All the great third party projects created like Angular Material (v1.x) and Bootstrap would be eventually deprecated.

Fun Fact: Angular vs AngularJS

AngularJS 1.x always kept the 1 in its versioning. Whether it was a breaking change or not, the major version was always 1.x.x

With Angular 2.0 the team has since decided to follow semantic versioning practice of (Major).(Minor).(Patch). If any breaking change is introduced into a new release, it bumps up the major version. Since the initial release there have been around 3 breaking changes (FYI: we skipped version 3 going from 2 → 4 to get several angular packages in alignment)

2016: The Panic Period

Developers and managers alike started to freak out. The complete rewrite meant a doomsday scenario for their current AngularJS projects running in production. “How can we support this application in 3, 5, or 6 years from now?!”

Even worse, their was no clear migration strategy to port an AngularJS 1.X application to the new Angular 2.0.

Teams jumped the gun and started to build new Angular2.0 apps while it was still in beta. The thought was “I better get going if this is going to be a total rewrite”.

What they found was a lot of frustrating new concepts and breaking changes with every release. It was at this point teams quickly started to turn their back on AngularJS for libraries like React. Developers wrote break-up letters to their once beloved JavaScript project.

Growing Pains

I don’t blame a lot of teams that made a switch. Its understandable they needed a dependable framework today and not tomorrow. However, I do think there are some salty opinions floating out there after only spending limited time with beta releases.

Beta software is prone to breaking changes. Its software that isn’t fully fleshed out. In early stages it was rather cumbersome to even create a new

Angular project. Build sizes were rather large, syntax was very unfamiliar, and an overwhelming amount of new concepts like TypeScript were thrown at developers.

Teams are googling Angular and seeing these breakup letters and getting scared to head in another direction. Its unfortunate they don't take time to evaluate the project for themselves and see all the progress the team has made of recent.

2017 — 2018: Emerging from the rubble

Fast forward to 2018 and the Angular framework has reached several major releases in efforts to stabilize their framework. Each release bringing better build sizes, stable APIs, and overall better performance.

It was a heavy price to pay, but the team ultimately redesigned Angular from the ground up to give us more than we ever had in 1.x. Unlike libraries such as React, Angular really provides solutions at every corner of building a large application. Libraries like React still require outside projects to fill in the missing pieces to our project. There are pros and cons to both, but thats another topic for another time.

The Cliffs of Insanity: Dramatic Shifts in Technologies on Stack Overflow

I find most negative feedback of Angular comes from the early 2.0 versions. Some just might not like the new flavor of Angular. Thats completely fine.

Different strokes for different folks.

Since the initial release, some great tools like the Angular CLI have come out to make Angular development a much more enjoyable experience. Other projects for the new flavor of Angular, like Angular Material, Angular Bootstrap, and Ionic, have also matured to use in production environments for the new re-write.

Going Forward

As for our old friend AngularJS, it's still supported. The core team recently announced it will release a final 1.7 and move into long-term support.

Does Angular have some catching up to do? Sure. But the gap is small enough to not be of concern. The benefits of operating in a unified framework can outweigh a fragmented landscape for some teams.

With stable APIs and plenty of open source projects to power our apps, the project is more than capable of producing great applications for large teams.

Angular finally looks to be regaining some of its lost momentum.

Need help on your Angular or React projects? Contact us at The Startup Lab for training or consulting services

Angularjs Angular JavaScript Startup

209 1

WRITTEN BY

Dave Gavigan

Web & Mobile Developer, Founder of https://www.thestartuplab.io . I help people build their products and launch their business

Follow