International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 7 July 2022, pp: 1615-1618 www.ijaem.net ISSN: 2395-5252

Survey on migrating Angular JS applications to Angular 2+ versions

Sai Praneeth A, Dr. B.M Sagar
Department of ISE, RV College of Engineering, Banglore, India
Department of ISE, RV College of Engineering, Banglore, India

Submitted: 15-07-2022 Revised: 27-07-2022 Accepted: 29-07-2022

ABSTRACT—One of the most well-liked frameworks for creating desktop and mobile applications is Angular. It has become more popular among front end and back end developers since its debut in 2016. It has sophisticated features that improve the structure and effectiveness of code. Business executives should understand how to migrate from AngularJS to Angular and be prepared to do so. Each Angular version provides valuable advantages, but there are several advantages to using the most recent version. Angular is noticeably quicker than AngularJS, has a mobile-first approach, performs better with components, and facilitates easier version upgrades.

Rewriting the application essentially gives developers the flexibility to do whatever they want. Use whatever framework you like. The structure of the application may be redesigned and reengineered, and even the application's interface and usefulness can be enhanced. Although the advantages are evident, there are certain more factors to take into account before choosing to completely redesign the application.

Keywords—component,directive, MVC

I. INTRODUCTION

As previously noted, Google has formally declared that AngularJS development and maintenance will end in the summer of 2021. Since the most recent version, AngularJS 1.7, was published on July 1st, 2018, Long Term Support (LTS) of stable AngularJS has been gradually ending. The introduction of the following generation Angular 2+ in late 2016 also served as a warning sign around a year earlier.

Despite the fact that Google supported AngularJS and provided resources to the community for maintenance and support, the framework is open source. Therefore, even without Google's assistance, the framework could conceivably still receive support from a vibrant developer community. However, in practice, AngularJS, even the most recent version, is now out of date; Angular is generally acknowledged to be far superior; and there is no motivation to do so.

Software development frameworks in particular, as well as technologies more broadly, have a life cycle. That circle of life eventually results in either evolution or extinction. The growth of the AngularJS framework into the more feature-rich and adaptable Angular 2+ series, which as of this writing has reached the Angular 13 iteration, may be seen in the example of AngularJS. The original AngularJS framework left behind a rich history and made significant contributions to the advancement of JavaScript programming. The most recent revisions of the Angular framework are at the core of that legacy.

However, organizations that have active apps based on AngularJS will be affected by the formal end of maintenance for the framework. It denotes:

- The infrastructure, linked parts, and framework itself will cease to receive formal support. Functionality will be frozen, and there won't be any additional bug fixes or security patches as a result.
- Third-party libraries will cease to receive maintenance and contributions from their maintainers and contributors, who, if they haven't done so already, will switch to the active Angular 2+ frameworks
- As many of the developers who had been using AngularJS switch to newer, more advanced technologies, the activity of the worldwide community around it will decline. Included in this will be activity on significant public resources like StackOverflow
- There won't be many, if any, new blog articles, white papers, or tutorials on AngularJS, which most likely implies there won't be any new discoveries or best practises
- As more developers get interested in working on the modern Angular framework, it will

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 7 July 2022, pp: 1615-1618 www.ijaem.net ISSN: 2395-5252

become more difficult to find skilled and motivated individuals to support AngularJS projects.

In comparison, the Angular 2+ framework is actively being developed and already offers amazing new capabilities and efficiency, with the promise of many more to come. Additionally, the community for Angular 2+ is quite active and regularly creating new tutorials, blogs, and other tools to debate problems and ideas.

II. METHODOLOGY

Before the migration itself begins, there are a few procedures to be taken. Although not required, the data in this area may aid in the migration procedure. Your life will be simpler when migration begins if you start preparing for it sooner rather than later.

A. Configure TypeScript.

Technically, TypeScript is not required for Angular to function. Even Dart and JavaScript are compatible with it. However, setting up and using TypeScript in advance is desirable .This makes one less thing to worry about when making preparations for the actual migration because TypeScript is prepared.

This can be put up with the aid of @types/angular. Ensure that TS and JS code are compatible with one another. Fortunately, the TS compiler makes doing it very simple. — The allowJs compiler option enables TS compiler to compile JS files.

Also consider a TS module integration with the project's chosen module system. With the —module="commonjs" and —esModuleInterop=true options, our framework utilizes CommonJS.

B. Module Loader Usage

The built-in module systems of TypeScript or ES2015 may be used by using a module loader like SystemJS, Webpack, or Browserify. You may utilise the import and export features to specifically state what code can and will be shared among other application components. Use the CommonJS style require and module.exports features for ES5 apps. In any scenario, the module loader will take care of loading every piece of necessary code for the application in the proper sequence.

Module loaders also facilitate the packaging of all applications into production bundles with batteries when moving applications into production.

C. Component Directives Usage

The primary primitive in Angular from which user interfaces are constructed is a component.

These directives establish input/output bindings, controllers, and templates in the same way as Angular components do. Comparatively to apps developed using lower-level capabilities like ng-controller, ng-include, and scope inheritance, applications built with component directives are considerably simpler to move to Angular.

D. Usage of ngUpgrade library

Angular offers an official method to make step-by-step migration easier even if it is not backward compatible with AngularJS. Within a hybrid application, the ngUpgrade library offers capabilities to mix and match AngularJS and Angular code. The two frameworks are launched, the code is executed in the appropriate framework for which it was written (for example, AngularJS code is executed in the AngularJS framework, and Angular code is executed in the Angular framework), and tools provided by ngUpgrade connect the components in both frameworks so that they can work together with little developer setup. With its documentation outlining the key steps in the migrating process, ngUpgrade is a tremendous assistance.

III. RESULTS AND DISCUSSION

A paradigm leap from AngularJS, Angular 2 changes not just the language but also the fundamental architecture and method of data binding. Nevertheless, programmers and web developers continue to utilise AngularJS and Angular in accordance with their needs.

A. Architecture

The MVC, or Model View Controller, architecture is supported by AngularJS. You place the required result in the controller and the business logic in the model, and Angular handles all the processing to produce that output. AngularJS automatically creates the model pipelines.

In contrast, Angular's fundamental building blocks are components and directives. The only thing that components are are directives with a predetermined template. They provide the applications a contemporary framework, which makes it simpler to develop and manage bigger applications.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 7 July 2022, pp: 1615-1618 www.ijaem.net ISSN: 2395-5252

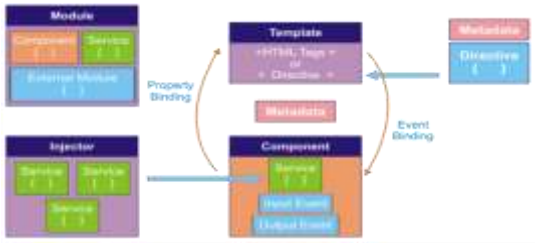

Fig. 1. Angular 2+ Architecture

B. Dependency Injection (DI) And Angular CLI

Dependency injection is used by both AngularJS and Angular, although they go about it quite differently. Different link functions, controller functions, and directive definitions all have DI injected into them in AngularJS. On the other side, Angular uses declarations, function Object() { [native code] } functions, and providers to build a hierarchical dependency injection framework.

With Angular 2+, a command-line interface is included. It is used to swiftly and effectively generate components, provide services, and even finish projects. With the help of dynamic type checking, linting, etc., you may quickly create multiple versions of the same code for various platforms. There is no separate CLI for AngularJS.

C. Performance And Mobile Support

AngularJS is significantly slower than Angular. Developers assert that Angular apps may be up to five times quicker than AngularJS applications if properly constructed.

As more and more complicated apps are created using the original AngularJS, two-way binding, which gave it popularity among web developers, has turned out to be its downfall. AngularJS uses a digest cycle to continuously compare each scoped variable with its previous value in order to assure and implement two-way binding. Due to the random nature of this digest cycle's operation, checking may continue indefinitely as the program's size increases, which would negatively affect application performance.

Angular's flux design uses unidirectional data flow for change detection, which speeds up applications.

In contrast to Angular, AngularJS does not allow mobile development. Because of this, AngularJS appears somewhat antiquated in today's mobile-first computing era.

D. Comparative analysis

TABLE I
Comparison of parameters between AngularJs and Angular

Parameters	AngularJS	Angular
Speed	Shortened development period because of two-way binding	With each updated iteration, performance and speed have improved.
Data Binding	Two way binding	Property,event ,template binding and interpolation
Tooling Support	Third Party Tools	CLI tools
Dependency Injection	Does not use dependency injection	Uses hierarchical dependency injection
Reactive Programming	No	Yes
Mobile Support	No	Yes
Material Design	No	Yes

Angular is way faster than AngularJS because Angular uses a component-based design and a far better data binding technique.

An Angular application's components are very autonomous and self-sufficient, making them reusable and test-friendly.

It is simpler to scale up, replace, and maintain separate components.

Applications built using Angular may be displayed on both browsers and mobile devices.

Extensions for server-side application rendering are incorporated into Angular. This makes it possible for material on the client and server sides to be in sync, which is fantastic for SEO.

Because Angular allows lazy loading, applications load more quickly because just the components that are required are downloaded.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 4, Issue 7 July 2022, pp: 1615-1618 www.ijaem.net ISSN: 2395-5252

Angular's initial TypeScript approach results in clearer code, improved navigation, and high-quality output.

IV. CONCLUSION

Due to its built-in capabilities, including as two-way binding, the ability to create responsive online apps, and extremely responsive designs, AngularJS quickly became popular among web developers after its initial release. When higher performing front-end development frameworks like ReactJS were introduced, its shortcomings were brought to light. As previously noted, AngularJS was updated into Angular 2 to give web developers a better option.

The greatest alternative for creating huge and complicated applications right now is Angular. The Google development team is striving to optimize the speed and shrink the build size of Angular applications.The framework of the future is Angular. Google switched to TypeScript to enhance both the overall performance of apps and the maintainability of the code.

Angular currently appears to be focused on developing enterprise-level applications.

Engineers discover that creating and sustaining these apps utilizing the supposedly simple to learn, deploy, and manage frameworks like ReactJS starts to become challenging as an application's complexity and scale increase. Instead, Angular's reliability and technical support more than offset the high learning curve. Google has further positioned Angular for long-term, resource-intensive projects by committing to long-term support (LTS).

REFERENCES

[1]. Bin Wang, Ziyan Jiang, Yang Liu, Yuzhi Zhang, Ke Xu, Zhuang Yuan, "LoRa-based Fire Monitoring System", 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C), pp.961-967, 2021.

[2]. Mohamed Sultan, "Angular and the Trending Frameworks of Mobile and Web-based Platform Technologies: A Comparative Analysis", Proc. Future Technologies Conference, pp. 928-936, 2017.

[3]. Adam Andrzej, "CTO's guide to Svelte – what can the rising frontend framework do for you?", Oct. 2019,[online]Available:https://tsh.io/blog/svelte-framework/.

[4]. S. Guan, W. Hu and H. Zhou, "Front-end and Back-end Separation - React Based Framework for Networked Remote Control Laboratory", Chinese Control Conference CCC, vol. 2018-July, pp. 6314-6319, Oct. 2018

[5]. B. Verhaeghe, A. Etien, N. Anquetil, A. Seriai, L. Deru-Elle, S. Ducasse, et al., "GUI migration using MDE from GWT to Angular 6: An industrial case", 2019 IEEE 26th International Conference on Software Analysis Evolution and Reengineering (SANER), 2019.

[6]. M. P. Robillard and K. Kutschera, "Lessons learned while migrating from swing to javafx", IEEE Software, vol. 37, no. 3, pp. 78-85, 2019.

[7]. H. M. Sneed and C. Verhoef, "Cost-driven software migration: An experience report", Journal of Software: Evolution and Process, pp. e2236, 2020.

[8]. N. Anquetil, A. Etien, M. H. Houekpetodji, B. Verhaeghe, S. Ducasse, C. Toullec, et al., "Modular moose: A new generation of software reengineering platform", International Conference on Software and Systems Reuse ICSR2020, 2020.

[9]. A. A. Terekhov and C. Verhoef, "The realities of language conversions", IEEE Software, Nov. 2000.

[10]. K. Garcés, R. Casallas, C. Alvarez, E. Sandoval, A. Sala-Manca, F. Viera, et al., "White-box modernization of legacy applications: The oracle forms case study", Computer Standards & Interfaces, pp. 110-122, Oct. 2017.

[11]. S. Bragagnolo, N. Anquetil, S. Ducasse, S. Abderrahmane and M. Derras, "Analysing microsoft access projects: Building a model in a partially observable domain", Proc. Int. Conf. Softw. Syst. Reuse (ICSR'20), no. 12541, Dec. 2020.

[12]. L. Włodarski, B. Pereira, I. Povazan, J. Fabry and V. Zaytsev, "Qualify first! A large scale modernisation report", Proc. IEEE 26th Int. Conf. Softw. Anal. Evolution Reeng. (SANER), pp. 569-573, 2019.

[13]. F. Erdős, "Economical Aspects of UX Design and Development," 2019 10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), 2019, pp. 211-214, doi: 10.1109/CogInfoCom47531.2019.9089992.

[14]. B. Szabó, K. Hercegfi, "Research questions on integrating user experience approaches into software development processes" 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), pp. 243-246, 2017