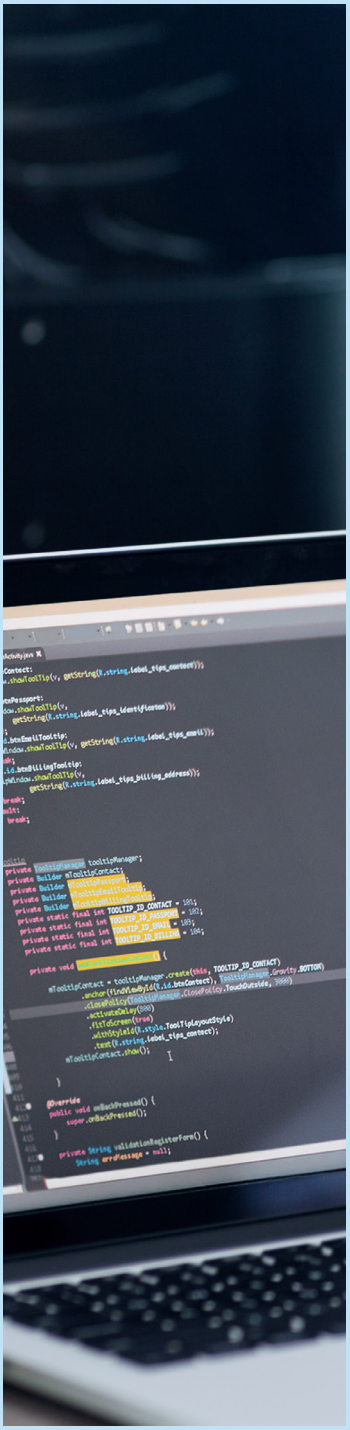
VIEW POINT



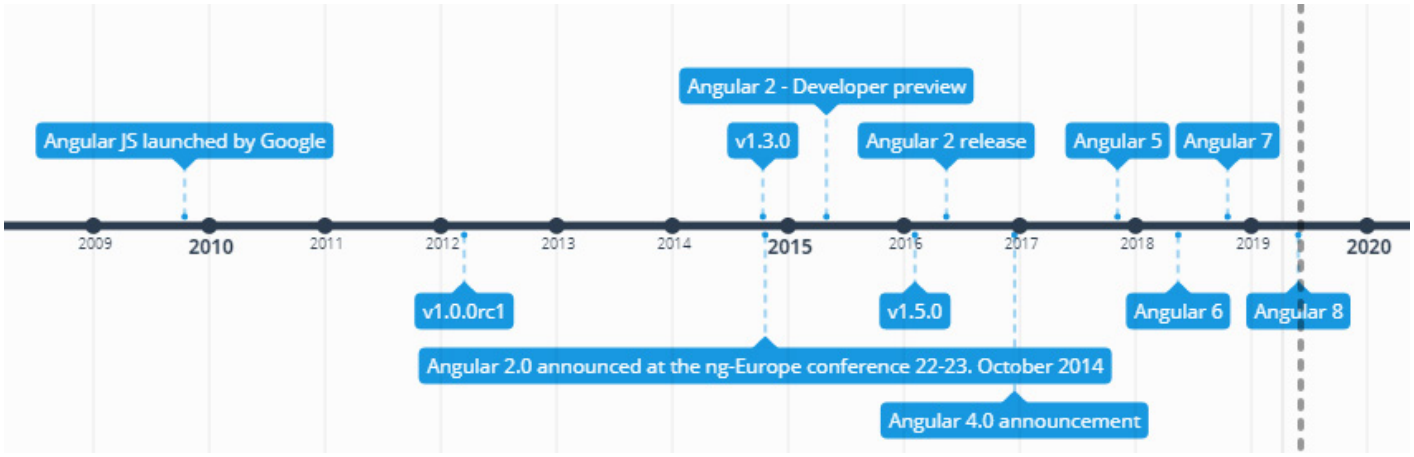
ANGULAR

Abstract

The advent of modern browsers and network technologies led to a digital disruption, where user experience, ease of access and speed have been key towards delivering consumer experience. Enterprise solutions delivering digital content from servers in response to browser requests, have transformed to full blown browser based apps leveraging advancements in HTML5, offline access and AJAX.

With the focus shifting to building highly complex browser based applications delivering rich experience, developers were challenged with the primary concerns of maintenance and scalability. Libraries like jQuery did not bring a clear separation of concerns, and often resulted in cluttered codebase and excessive DOM manipulation. This resulted in the evolution of frontend JavaScript frameworks like Angular, providing an architecture for building interactive web apps, with support for data binding, enabling faster development of CRUD based applications. This PoV focuses on Angular and how it helps in building web and mobile apps with rich experience.

Infosys®
Navigate your next



Angular

Angular is a front-end open-sourced JavaScript based web framework mainly maintained by Google and open source community to address the challenges faced by developers building single-page applications. It was first officially released in October 2010 as AngularJS (or Angular 1.x) and later completely revamped on September 2016 as Angular (Angular 2+).

Angular had the following advantages over the traditional development of web applications:

- Provides structural framework for building rich interactive web apps, with support for data binding, view hierarchy and route navigations enabling faster development of CRUD based applications.
- Intends to decouple DOM binding and manipulations from app code in jQuery, Backbone era, introducing clear separation of concerns.
- Better abstraction and articulation of presentational elements as loosely coupled components.
- Encapsulation of business functionality by means of reusable services and dependency injection.
- Focus on independent unit testing.

Inevitable Framework Revision

While AngularJS came with huge boost for developers building rich web applications, there were few limitations that resulted in redesign of the framework from ground up. Key challenges were:

- The concept of Scope was confusing to developers and misused in many contexts.
- Though envisioned with component design, development of custom components / directives required a detailed understanding of digest cycle.
- In larger applications, the performance considerations forced developers to be knowledgeable of the internals of AngularJS such as, the digest cycles, watches, bootstrapping etc.
- Lack of a standardized tool chain, to help productivity and to troubleshoot issues.

Angular Redefined

Angular 2.0, being a complete rewrite from AngularJS, came with significant improvements in the core framework both in terms of performance and developer experience. The major advantages were:

- Typescript bringing Object Oriented techniques to JavaScript and makes the code more structured and organized. Angular 2.0 leveraged typescript to ensure type safety of the applications built.
- Declarative metadata by means of Typescript annotations.
- Adherence to web standards, enforcing developers to think more in terms of components, enabling better modularity and wider reuse.
- Improved bootstrapping, component tree and change detection strategies.
- Ahead of Time compilation (AOT) for better security and performance.
- Provides easy upgrade options from AngularJS (1.x) and do incremental development in latest version.
- More choices of languages - ES5, ES6, Type Script or Dart.
- CLI gives a head start with creation of boiler plate code, enabling rapid application development.

Where does Angular fit?

While deciding the technology to be used for building a web application, fitment analysis for the suitability of Angular is important. The scenarios where Angular would be a good fit are:

- Single page applications which need capabilities of a full-fledged web application development framework like application modularity, dependency injection and a component driven User Interface.
- Mobile first web apps and progressive web applications with offline capability.
- CRUD based applications demanding rich user experience and high interactivity.
- Cross platform mobile applications built over web based technologies (such as *NativeScript* and *Ionic*).

Hybrid Development Platform

Angular is not only used to create dynamic web applications. It can be used to build intuitive mobile and desktop applications, supporting multiple form factors.

Frameworks like Ionic and Cordova help packaging an angular web app to iOS and Android platforms with access to native device capabilities. As for building native desktop applications, there are frameworks like Electron and Photon Kit targeting the Windows, Linux and MacOS platforms. NativeScript relies on angular to build mobile apps rendering native components, and utilizing JavaScript runtime core.

Angular Architecture

Angular provides constructs to build a modular, decoupled, layered and testable web application. Key architectural elements of Angular include:

- **Modules:** Provide isolated compilation context to allow decomposing applications into manageable sub units.
- **Components:** An angular app consists of a composite component tree. Components take up the role of controller and view model in case of angular apps. Components are bound to templates which represents the views.
- **Directives:** Help altering appearance and behavior of UI elements or to introduce structural layout of components and UI elements. Directives help bind angular code directly to host elements in the DOM.
- **Pipes:** help transformations of display values in templates (such as currency formatting, date formatting etc.).
- **Router:** Help tracking the changes to browser URL and declaratively include components corresponding to the route state. Router also supports imposing routing restrictions based on application states.
- **Services:** To declaratively include application based functionality. Services are injected to components and other services by means of an Injector.

Angular CLI and Console

Angular CLI is an incredible tool that allows developers to setup an Angular application with boiler plate code, and workspace setup in a fraction of minutes. It reduces the complexity of creating the application, module, components from the scratch.

Angular Console is an external plugin developed on top of Angular CLI to serve as an UI for Angular CLI. With Angular Console, developers no longer need to memorize the commands, the Angular Console UI will enable user to create application, components visually.

CDK and Schematics

The Component Dev Kit (CDK) is a set of tools that implement common reusable patterns/features. CDK acts like an interface upon which view components can be developed.

Schematics are a "recipe" to generate and modify files in a project. In a way, Angular Schematics helps to create, manage and update the file system.

Libraries and Angular Elements

Angular libraries are reusable packages and provide projects that cannot run on its own but are imported in other Angular projects.

Angular Elements helps us publish Angular components as custom HTML elements, which could be used in non-angular context.

SSR and Angular Universal

Server Side Rendering(SSR) is a process by which pages and routes of a JavaScript based web app are rendered from the server.

Angular Universal helps in executing the Angular application in server and at a later point of time gets bootstrapped on the client. This gives a boost to angular applications demanding the need of SEO and faster bootstrap time on the browsers.

Angular 8 Features

On 29th May 2019, Angular team announced the official release of Angular 8. It comes with a whole bunch of cool new features that the community were eagerly expecting.

- IVY rendering engine: With this new rendering engine released in preview mode, the code generated by Angular compiler is easy to understand. Application rebuilding time is significantly faster with decreased payload size.
- Differential loading made default: Angular compiler will produce both legacy (ES5) and modern (ES2015+) javscript bundles which will differentially be loaded to the browser effectively improving the loading speed and time to be interactive for modern browsers.
- Enhancements to service workers: Service worker registration has a new option that allows to specify when the registration should take place. It is now possible to bypass the Service Worker for a specific request by adding the ngsw-bypass header. Further, hosting multiple apps on same domain has also been made possible with the help of the new updates.
- Web Workers in CLI: Provides new schematic to add Web Worker to any one of the components.
- Typescript upgraded to 3.4: Latest version has faster builds, omits helper type and improved excess property check in union types.
- Builder APIs: The new API helps in changing the behavior of the CLI command, including addition of new ones to create custom logic.
- Workspace APIs in CLI: The latest updates brings in an easy way to modify the angular.json file through CLI itself.
- Forms module enhancements: The AbstractControl class now offers a new method named markAllAsTouched. The FormArray class now offers a method, to quickly remove all the controls it contains.
- Dynamic Imports: All the lazy-loaded routes will use standard dynamic import syntax instead of a pre-defined string.
- Bazel build support: Bazel is a powerful tool which can keep track of the dependencies between different packages and build targets. It has a smart algorithm for determining the build dependencies. Bazel is independent of the tech stack.
- Router enhancements: Lazy loading of modules through import() format instead of string. This gives more control over when a module has been loaded.
- Angular Firebase integration: Possible to deploy the angular application directly to firebase using CLI.

Considerations on Angular Apps

While choosing Angular for building responsive web apps, there are few considerations to be kept in mind:

- Mobile first and responsive approach to ensure the app seamlessly work across multiple form factors.
- Choice of progressive web applications ensuring app like experience to websites with limited offline capability.
- Considerations towards the extent of SEO and discoverability of the app in public.
- Social Integration and Single Sign On (SSO).
- Dynamic contents and integration with Content Management Systems (CMS).
- Applications with high performance and better security.

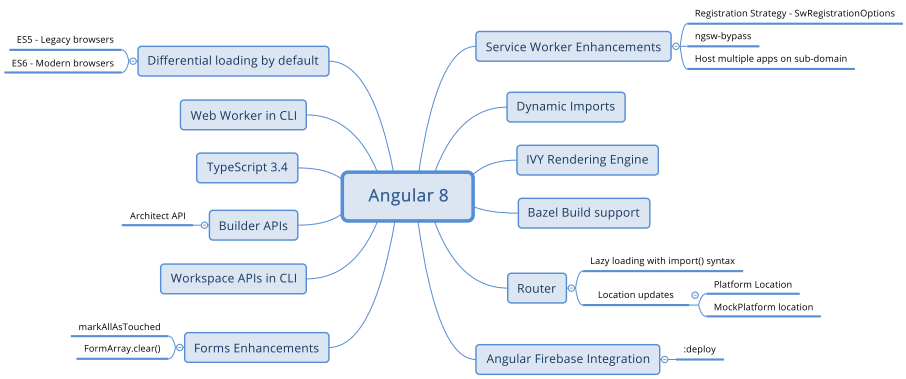


Figure 1: *Angular 8 - Features*

Development Practices

Following are the best practices to consider while development with Angular:

- Wireframe / UX break up to identify components, behavior and navigations.
- Design for reusability and consistency.
- Use SCSS / Stylus aligned to style guides and branding themes. Consider isolation of themes from components.
- Effectively use Angular CLI and Schematics for code generation.
- Minimize interactions, data exchange and event bubbling across component tree.
- Package generalized reusable components and services as independent libraries.
- Isolate shared components from feature components.
- Have common behavior abstracted as base components, services and shared reusable module.
- Have separate routing module, and on demand download of lazy modules.
- Effective use of mono repo pattern and angular workspaces.
- Use mediator services where ever required, to orchestrate between components.
- For complex and large scale enterprise apps, have a standard mechanism like ngrx to manage application state.
- Build components that are testable.
- Effectively use playground for unit testing and showcasing of common components.
- Use async pipes for managing reactive streams.
- Implement the component life cycle methods for proper initialization and clean up.
- Consider using AoT and tree shaking for build.

Developer Pitfalls

Following are few potential pitfalls the developers have to be cognizant of developing angular application:

- Improper use of view encapsulation of components.
- Excessing argument passing over component tree using input / output.
- Mutating the input data resulting in problems that are hard to troubleshoot.
- Incorrect management of application state through multiple channels and components.
- Improper use of change detection strategies, causing performance and functional issues.
- Improper content transformation and ineffective use of pipes.
- Using functions in template interpolations.
- Leaving out Zombie subscriptions, without unregistering them.
- Improper module exports, component declarations and use of providers.
- Direct DOM manipulations, bypassing change detectors.
- Keeping logic inside components.
- Ineffective use of content projection.
- Exporting the same component in multiple NgModule.
- Incorrect use of Providers.
- Component layer holding business logic.

Enterprise Angular Adoption

From its launch, Angular has been one of the most sought after frameworks by front-end developers. Over the years Angular has evolved from being a niche library to a full blown framework helping development of rich, highly intuitive mobile and web applications. Community has also helped to evolve a large platform ecosystem to build rich web applications with Angular. The ecosystem includes, but not limited to:

- IDEs, CLI, development tools, & testing tools.
- Data access and storage libraries.
- UI components toolkits
- Cross platform development tools.
- Books, workshops and trainings.
- Community groups and curators.

This has facilitated increased adoption of Angular for building enterprise web apps.

The Future

With the enhancements to Server Side Rendering methods and Service workers, Angular is leading the way in the general trend of web development leveraging both server and browser, to deliver the highest user experience. As for Progressive Web Apps and offline access, Angular gives more flexibility in terms of what and how content is being cached. By utilizing browser services like IndexedDB and Storage, Angular apps can be developed to work fully offline in a disconnected environment.

In modern digital era, it is imperative that systems are extremely scalable to meet unprecedented usage, especially in business to consumer scenarios. This implies that efficiency is paramount in serving requests with minimal computational and network costs, typically over cloud. With the SPA nature of angular apps, and the capability of serving responses within the browser, Angular turns out to be a good bet in serving user interfaces for massively scalable reactive web apps.

Since Angular distribution contains only static assets of HTML, JavaScript and CSS bundles, it is easy to deploy Angular apps to the edge and CDN, thereby improving the performance and experience.

Due to demand on high scalability and efficiency, the recent development trends has been favoring reactive, asynchronous and functional programming paradigms. The application architecture components, including user interfaces are expected to respond to streams of events and data in an efficient manner. Angular framework extensively relies on Reactive Extensions for Javascript (RxJS) and Observable patterns to meet the asynchronous behavior in modern web applications. This is further supplemented by support towards reactive state management, aligned to REDUX architecture by means of libraries such as NgRX.

Organizations choosing Angular as a preferred framework for building SPA, on massive and diverse scales often rely on standardizing the component libraries and frameworks across multiple web applications. Angular ecosystem enables building and maintaining an enterprise component repository with the support from tools like Angular Playground and Storybook.

Organizations are also driving towards reusability of presentational elements and business components across the Angular applications developed by multiple teams. The Angular Workspace APIs and mono repo patterns support exposing reusable artifacts within an organization as exportable Angular libraries.

With the increasing industry interest in MBaaS platforms and Serverless architectures, for building cloud native enterprise scale digital solutions, Angular will continue its traction as a preferred web development framework supporting offline capabilities, deployment on edge and easy integration to REST, GraphQL based end points.

Conclusion

Angular is extremely well-aligned for building next generation applications which need to be decoupled, fault-tolerant, reactive, performant and massively scalable

Regular framework upgrades, a constantly growing eco-system, ease of adoption and fitment with cloud native patterns ensure that all the capabilities needed for building next generation application are already in place.

Developers can focus on building what they need to build best, amazing user experiences!!

Google's backing, over 1M weekly downloads in npm and 49k stars in the official GitHub repo stand testimony to this adoption

About the authors

Amit Nigam, Principal Technology Architect DX - Infosys

Amit Nigam is a Principal Technology Architect with Infosys. He has over 19 years of experience in IT with recent focus on Digital Modernization and Transformation programs

His current technology focus areas are Modern Web Applications, JavaScript UI Frameworks, Mobile Applications and Cloud Native Application Architectures

He can be reached at amitnigam@infosys.com

Roy M J, Technology Lead, DX - Infosys

Roy is a Technology Lead with Infosys unit DX. He has over 9 years of experience in open source web technologies mainly focused on the front-end.

Over the years he has worked on front-end frameworks like Angular, React, Ionic, Cordova and back-end frameworks like Yii, CodeIgniter, OpenCart, WordPress etc.

He can be reached at roy.john01@infosys.com

References

https://angular.io

https://cli.angular.io

https://material.angular.io

https://blog.angular.io

https://blog.angular-university.io

https://blog.angularindepth.com

https://angular-checklist.io

For more information on this PoV, contact digital@infosys.com

For more information, contact askus@infosys.com

Infosys®
Navigate your next