General overview

Introduction to Angular 6

Single Page Application (SPA)

- Web application that fits on a single page
- all the code (JS, HTML, CSS) is retrieved with a single page load.

SPA - No Full Page Refresh

- Navigation is achieved without refreshing the whole page
  - Just load the part of the page which needs to be changed
- After the initial page load
  - no more HTML gets sent over the network
  - Data gets requested from the server (or sent to the server)
- takes a lot less time and bandwidth than constantly sending HTML
- Payload type typically JSON
- New HTML generated to create new portions of the page using the payload data
  - Performed by the client (i.e. in the browser), rather than traditional server side rendering.

SPA - Better User Experience

- SPA feels like a native application
  - fast and responsive
- Constantly reloading everything from the backend server slows things down
  - due to network latency in fetching a lot of redundant HTML

SPA – Deployment and versioning

- Simpler to deploy
  - compared to traditional server-side rendered applications
- 3 static files:
  - one index.html file, with a CSS bundle and a Javascript bundle.
  - can be uploaded to any static content server like Apache, Nginx, Amazon S3 or Firebase Hosting
- Versioning and rollback is easier
  - Version the build output (index + CSS + JS bundles)
- Configure the server that is serving the SPA with a parameter
  - specifies which version of the frontend application to build

SPA problems

- More Complex to Build
- SEO
  - To index SPA app, search engine crawlers need be able to execute JavaScript.
  - May need to create static HTML snapshots especially for search engines (Angular Universal)
- Initial Load is Slow
  - SPA needs to download more resources when you open it

Angular overview

- Angular is a platform and framework for building client-side applications
  - typically Single-Page Applications (SPAs) using HTML and TypeScript
- Angular is written in TypeScript
  - implements core and optional functionality as a set of TypeScript libraries that you import into your apps

Angular features

- Custom components
  - Angular allows you to build your own components that combine customized functionality with UI rendering logic into reusable units
  - They also work well with web components.
- Data binding
  - Supports seamlessly moving data between the JavaScript application and the view, and react to view events without having to explicitly write the glue code
- Dependency injection
  - Allows the creation of modular services, and have them injected wherever they are needed
  - Greatly improves their testability and reusability

Angular features

- Testing-centric
  - Angular has been built from the ground up with testability in mind
  - test specs are automatically generated
- Comprehensive
  - provides out-of-the-box solutions for server HTTP communication, routing , etc

AngularJS and Angular

- One of the first and most popular web application frameworks in Javascript
  - Used to bring structure and consistency to SPA development
  - Also support scalable and maintainable web applications
- AngularJS - Any release from 1.0 through 2.0
- Angular - For versions 2.0 and greater
- Version 2.0
  - complete rewrite and significantly different architecturally from Angular 1.0, not backward compatible with AngularJS
- All versions after it are planned as incremental changes upon it.
  - major releases on a six-month schedule, with a focus on easy upgrades
  - Upgrading between versions of Angular (2 -> 4 -> 5 -> 6 -> 7) should be an almost trivial upgrade.

Angular JS vs Angular

- AngularJS was focused solely on building web applications in the browser
  - Framework with a large ecosystem of third-party modules used to easily add features to your application
- Angular is a completely new version of the framework
  - leveraged a lot of the newer web technologies (modules and web component)
  - also improving existing features of AngularJS (dependency injection and templating)
  - Leaner core library and makes additional features available as separate packages that can be used as needed

Angular

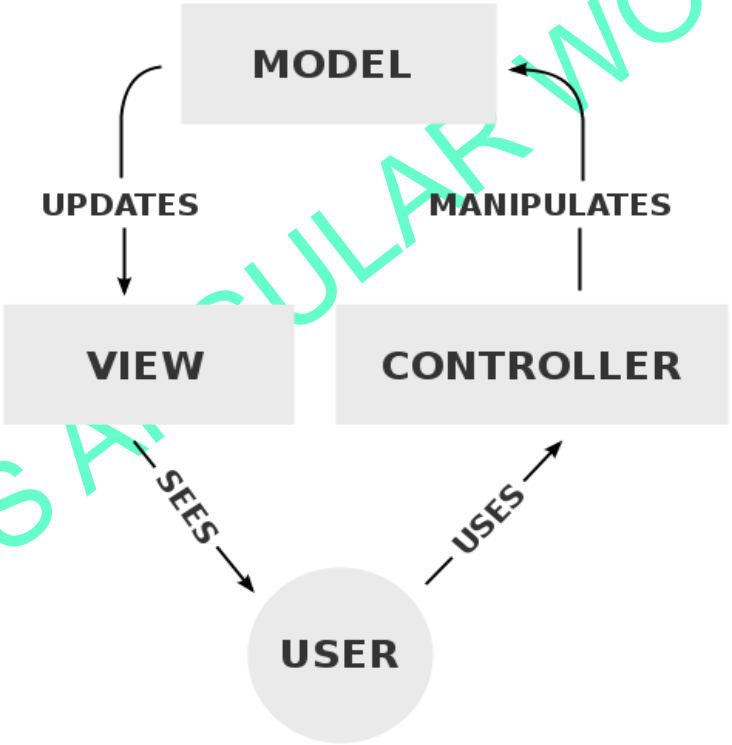
- It also has many tools that make it a full-rounded platform beyond a framework:
  - Dedicated CLI for application development, testing, and deployment
  - Offline rendering capabilities on many back-end server platforms
  - Desktop-, mobile-, and browser-based application execution environments
  - Comprehensive UI component libraries, such as Material Design

MVC architecture

- Traditional architecture for developing UI
- Model
  - the model is an abstract representation of your data.
- View
  - the view represents the presentation layer and the actual UI.
- Controller
  - the controller is an interface for handling user interactions and connects both the model and the view.
- Specifically designed so that the view and the model don’t need to know anything about each other
  - Allows developers to work simultaneously on different components of a web application without impacting one another

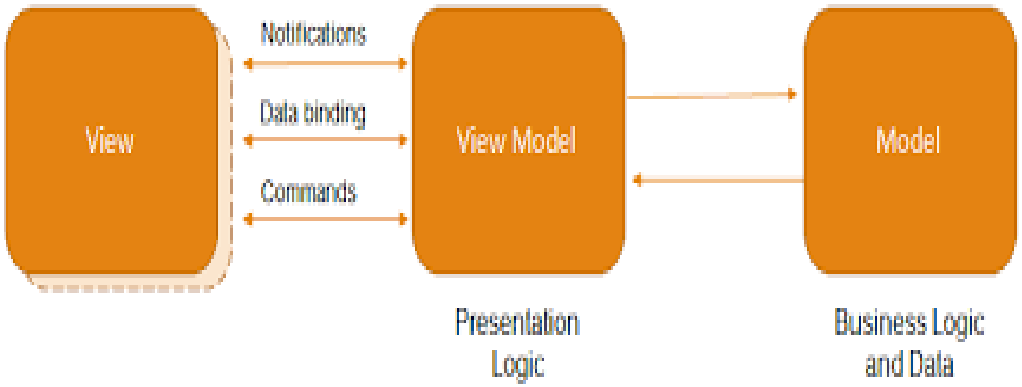
MVC architecture

- Drawbacks
  - Use of controllers to manipulate data models creates clutter in the backend
  - It also does not work well for SPAs.



MVVM architecture

- AngularJS follows MVVM pattern
  - replaces the Controller with a View-Model
- ViewModel acts as a binder that binds data between the view and model
  - Allows the view and model to communicate directly with each other
- The View-Model synchronizes the data between a view and a model
  - Changes made to a UI element automatically propagate to the model and vice versa



Component architecture

- Angular has a component-based architecture
  - Every Angular application has at least one component known as the root component.
- Each component has an associated class
  - This is responsible for handling the business logic
  - There is also a corresponding template that represents the view layer.
- Multiple, closely related components can be stacked together to create a module
- Each module forms a functional unit on its own