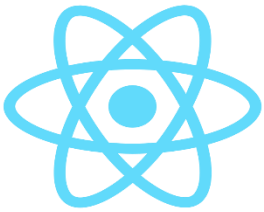
Frontend JavaScript Frameworks

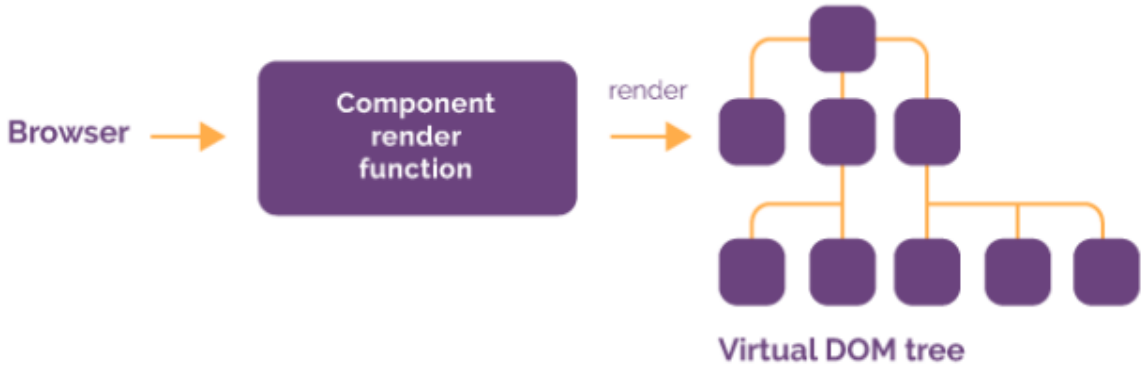
1. React



React.js is an open-source front end JavaScript library. React encourages you to use a reactive approach. Uses component-based architecture, JSX and unidirectional data flow. The unidirectional data flow is implemented by Flux. React apps are divided into multiple components which contain both business logic and HTML markup. Also introduced state and props. Using the state and props objects you can pass data from a component to the layout (state) or from a parent to child component (props). React is able to work with jQuery AJAX, fetch API, Super agent, and Axios.

<u>The React ecosystem includes :</u>

- The React library itself plus React-DOM for DOM manipulation.
- React-router for implementing routes.
- JSX, a special markup language that mixes HTML into JavaScript.
- React Create App, a command line interface that allows you to rapidly set up a React project.
- Axios and Redux-based libraries, JS libraries that let you organize communication with the backend.
- React Developer Tools for Chrome and Firefox browsers.
- React Native for development of native mobile applications for iOS and Android.



Advantages: Flexibility – there are many libraries to choose from

One Direction Data flow – difference between React and angular was that React was based on a downward One directional data flow architecture rather than a 2-way data binding adopted by Angular. Ensures that a child element cannot affect the parent.

Reusable components – React's component-based approach helps developers to import or reuse UI components.

Drawbacks: Deciding on libraries, may waste time. There's still no solid development workflow with React. When a React+Redux app grows bigger, you'll spend a lot of time applying small changes to multiple files instead of actually working on features.

Redux is considered the best and only library for enterprise-level React apps. At the same time, you need to think of the following: Redux can seriously slow down development performance. Redux make your work difficult when you implement a new feature and have to change too many things throughout the entire app.

Final thoughts: React is a good choice for this project as there will be little changes/development after the initial creation.

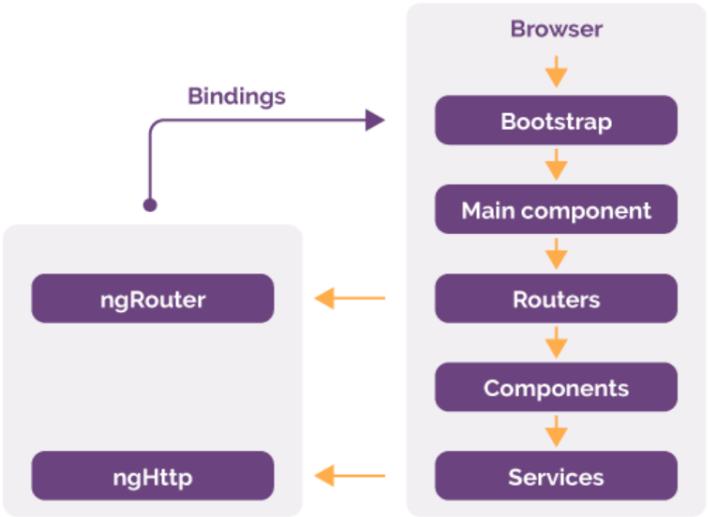
2. Angular

Angular is an open-source typescript-based framework used to build client-side single-page web applications. Angular has gone through some changes, the biggest of which was the shift from M-V-W architecture (Model-View-Whatever) to component-based architecture like React.

Today Angular is one of the most secure front-end JavaScript frameworks for building enterprise-scale applications from scratch.

Angular's ecosystem includes :

- A series of modules that you can selectively install for Angular projects: @angular/common, @angular/compiler, @angular/core, @angular/forms, @angular/http, @angular/platform-browser, @angular/platform-browser-dynamic, @angular/router, and @angular/upgrade.
- TypeScript and CoffeeScript, supersets of JavaScript that can be used with Angular.
- Angular command line interface for quick project setup.
- Zone.js, a JS library used to implement zones, otherwise called execution context, in Angular apps.
- RxJS and the Observable pattern for asynchronous communication with server-side apps.
- Angular Augury, a special Chrome extension used for debugging Angular apps.
- Angular Universal, a project aimed at creating server-side apps with Angular.
- NativeScript, a mobile framework for developing native mobile applications for iOS and Android platforms.



Advantages: Angular wants you to design frontend apps in a prescribed way, unlike React.

Server performance – Angular supports out of the box caching and tons of features to ensure fast server performance.

Ideal for building Enterprise scale web applications.

Drawbacks: Steep learning curve.

Obliged to use TypeScript to ensure type safety in Angular apps. TypeScript makes the Angular 2+ framework not so pleasant to work with. If you've used Angular 1.x before, you'll have to learn Angular 2+ from the beginning, because it's kept little from previous versions. Have to get used to services, the dependency injection framework (React doesn't have dependency injection), and directives.

Angular is extremely bulky and large in size, roughly close to 550-600kb which might not be suited for most of the small-scale applications.

Final thoughts: Angular is a solid choice. Choose Angular if you don't like to constantly choose from additional libraries as with React.

3. Vue

Vue is an open-source lightweight front-end JavaScript framework used to build creative user interfaces and high-performance single page web applications with minimum effort. Most of Vue's features were adopted from Angular and React and it has made major improvements to these features.

The Vue ecosystem includes :

- Vue as a view library.
- Vue-loader for components.
- Vuex, a dedicated library for managing application state with Vue; Vuex is close in concept to Flux.
- Vue.js devtools for Chrome and Firefox.
- Axios and Vue-resource for communication between Vue and the backend.
- Nuxt.js, a project tailored to creating server-side applications with Vue; Nuxt.js is basically a competitor to Angular Universal.
- Weex, a JS library that supports Vue syntax and is used for mobile development.

Advantages: Vue is much simpler and less restrictive than Angular.

Vue offers high flexibility, not only can it function as an end-to-end full-fledged framework like Angular but also a view layer with state management like React.

Less steep learning curve. No need to learn a superset of JavaScript like TypeScript or JSX.

Drawbacks: Much less popular than React or Angular. You may have to think about creating your own solutions to solve various issues.

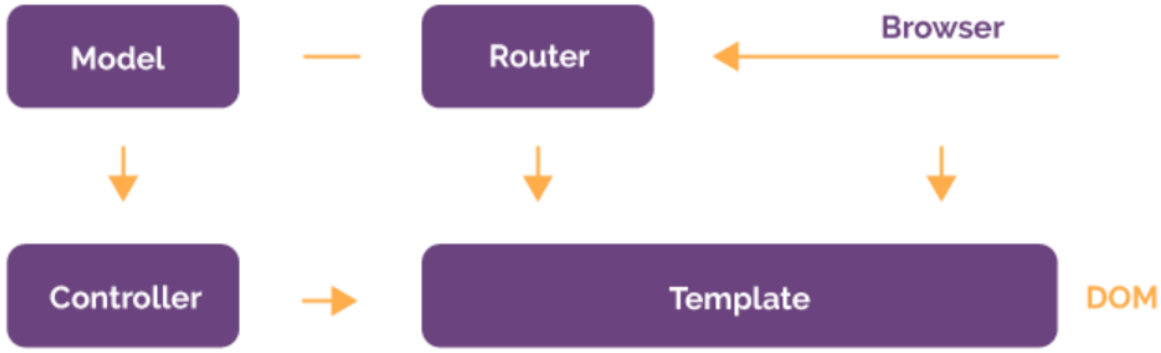
Final thoughts: Simpler but not as popular. Solid framework or solid choice of libraries outweighs simplicity. Not a bad choice but not the most appropriate choice for this project.

4. Ember

Ember lets you implement component-based applications just like Angular, React, and Vue do. Ember is said to be one of the most difficult JavaScript frameworks for front end web development due to its intricate architecture which nevertheless enables quick development of huge client-side applications.

<u>The Ember ecosystem includes:</u>

- The actual Ember.js library and Ember Data, a data management library.
- Ember server for development environments, built into the framework.
- Handlebars, a template engine designed specifically for Ember applications.
- QUnit, a testing JavaScript framework used with Ember.
- Ember CLI, a highly advanced command line interface tool for quick prototyping and managing Ember dependencies.
- Ember Inspector, a development tool for Chrome and Firefox.
- Ember Observer, public storage where various addons are stored. Ember addons are used for Ember apps to implement generic functionalities.



Final thoughts: If you need a cutting-edge framework to develop **complex** client-side apps, ember is the best/ only choice. // ours is not complex