Software Product Line Libraries & Frameworks for Web-based Applications

Devatar, Sachin (A20443522)- sdevatar@hawk.iit.edu

Abstract— **Web applications have evolved from publishing static information to serving dynamic and complex web content involving business functions. Besides, increasing number of browsers and platforms, devices are driving web application software to accommodate a wide range of client -side specifications without increasing development cost and time. To implement such system capable of handling such diversity as well as complex functionality, Software Product Line Engineering (SPLE) is a promising software development strategy. In recent days, product line has become an important topic in software development domain. The changes in market needs demand software requirements to be flexible in product lines. Finding various suitable components to construct an efficient and comprehensive product line with low costs is one of the critical problems in the domain which aspires to be satisfied. This paper gives an overview of four different software product line frameworks for web-based applications. A detailed comparative analysis of these frameworks based on issues and challenges is also performed.**

Keywords- ***Software Product Line, Frameworks, Frontend Frameworks***

1. Introduction

Web applications are complex systems that is built using a variety of software components and frameworks for serving dynamic content to users. Web Technologies are the tools and techniques that are utilized for the process of communication between different types of devices over the internet. In terms of web applications, software reuse can be thought of deploying commonalities among multiple members of a product family. A web framework is a code library that makes web development much faster and easier by providing common patterns for building reliable, scalable, and maintainable web applications. Software Product Lines are a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a product family or market segment.

2. Overview of Software Product Lines

In today's software industry, cost pressures keep increasing and timelines become more challenging. Improving efficiency and reducing time to market becomes a matter of survival for any organization. Applying Software Product Line principles in a software development organization will help the organization achieve increasingly challenging business goals efficiently.

A Software Product Line can be defined in various ways, depending on the relative perspective. Software Product Line Engineering provides strategies to implement, maintain and evolve SPLs. Software Product Lines can be termed as a collection of software-intensive systems which share a common set of features that satisfy the specific needs of a market segment or mission. SPLE consists of a systematic way to capture the common parts between products of a family of systems, while accommodating the differences. Software Product Line Engineering helps in creating a set of reusable software assets between products in the family. The major significance of SPL is to provide the guidance to an organization on how to shift towards product line approach for any software. The main benefit is a maintainable and comprehensible software asset base that can be reused. SPL approaches are gradually being adopted and increasing quality in software engineering. By incorporating SPLE in web-based development, advantages of commonalities and variabilities can be used to build custom web products. As such, Web engineering based on a standardized platform with reusable components is a logical next step in the evolution of Web application development.

3. SPL Frameworks in Web Based Applications

Web frameworks provide a standard way to build and deploy web applications. Web frameworks aim to reduce the overhead associated with common activities performed in web development and promote code reuse across products. SPL Frameworks when designed well will make software development more efficient by providing a set of rules and various approaches for domain analysis and implementation.

A web application framework is a software framework that is designed to support the

development of web applications including web services, web resources, and web APIs.
As such in web application domain, frameworks help to deal with cross cutting behaviors, support variability management at diverse abstract levels that could accommodate particularities in web-based languages, reduce complexity of service customization process, allow an average business user to generate ready to run web applications and many others.

A well-designed SPL framework could reduce Software Building Effort by providing an approach or a set of rules for domain analysis, requirements analysis, or a set of programs for do-main implementation, and software generation. They provide a standard way to build and deploy web applications on the World Wide Web (WWW). Few well known examples of web frameworks include AngularJS, ReactJS, VueJS, svelte.js.

4. Frontend Frameworks for Web Applications

4.1. Angular

4.1.1. Overview

Usage of frameworks is a crucial technology to be successful in developing business, Angular
has exploded in usage because of its unique features. Angular is a JavaScript library and its purpose is to efficiently create front-end by using Model View Controller design pattern. It is a structural framework for building dynamic single page web applications providing declarative templates with databinding. Angular extends HTML as template language to express the application's components and bind data.
Angular uses ES2015 and ES2016 in addition to 'Typescript' as a compiler to support classes and module loaders. When first launched, Angular was perceived as too revolutionary for its time, where ES6 was close to standardization and two-way data binding was needed for data over forms applications.

Typescript is a new feature embedded and integrated with Angular. Typescript provides the best IntelliSense developer experience available in the JavaScript world today.

4.1.2. Key Features of Angular

4.1.2.1. Two-way data binding/flow

Data binding is a very useful and powerful feature that is used in software development technologies. It acts as a bridge between the view and business logic of the application. Data binding lets you treat the model as the single source- of-truth in your application.
Two-way data binding in Angular enables exchange of data to and for i.e., between the view and the model. When data in the model changes, the view reflects the change, and when data in the view changes, the model is updated as well.

4.1.2.2. Templates

Angular templates are dynamic. An Angular HTML template renders an interface or view, in the browser with a lot more functionality just like regular HTML. When Angular renders them, it transforms the DOM according to the instructions given by directives.

4.1.2.3. Component Routing

In a single-page web app, we change what the user sees by showing or hiding portions of the view that correspond to components, rather than going out to the server to get a new page. As users perform actions, they need to move between the different views that we have defined. That is where a router comes in. The component routing in angular redirects a URL to another URL, resolve data before a page is displayed and runs scripts when a page is activated or deactivated.

4.1.2.4. In-built HTTP requests library

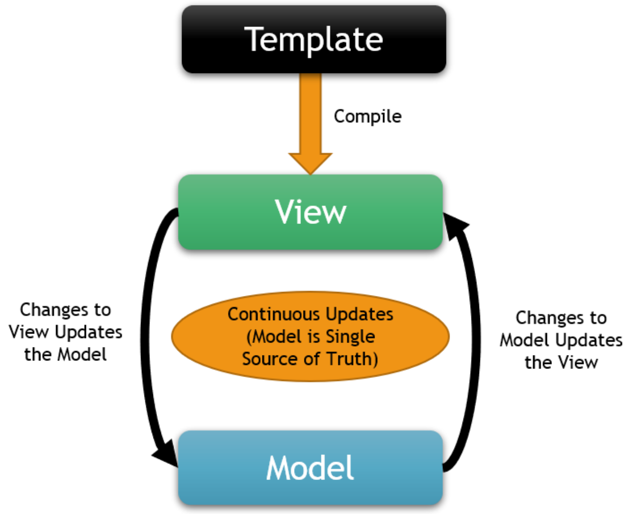
Most of the front-end applications needs to communicate with a server over the HTTP protocol to download or upload data and access other back-end services. Angular provides a simplified client HTTP API for web applications.

4.1.2.5. Cross Platform

Angular was designed with flexibility in mind. That is why Angular is a cross-platform framework, which is not limited by the browser.

The only thing which is required by Angular to be executed is a JavaScript engine.

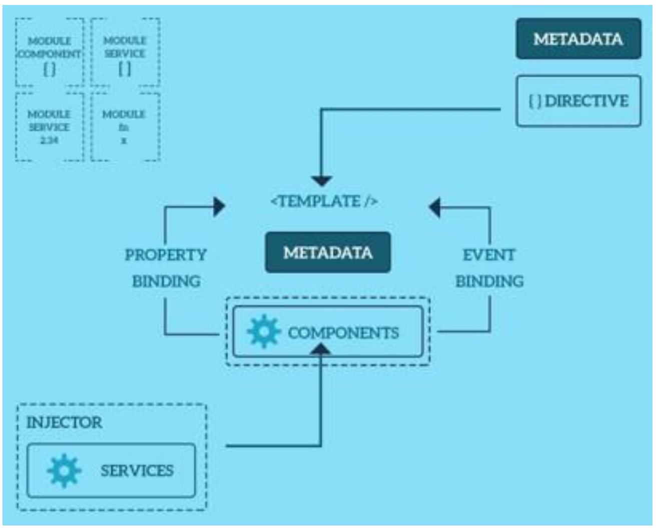
4.1.3. Angular: Two Way Binding


Template
Compile
View
Changes to View Updates the Model
Continuous Updates (Model is Single Source of Truth)
Changes to Model Updates the View
Model

Source: [1]

Databinding in Angular apps is the automatic synchronization of data between the model and view components. Two-way data binding can be achieved using a ng-bind or ng-model directive in Angular. Any data-related changes affecting the model are immediately propagated to the matching view(s), and that any changes made in the view(s) by the user are immediately reflected in the underlying model. When app data changes, so does the UI, and conversely. This data binding approach makes it possible to create a stand-alone library accompanied with the functions and controls.

4.1.4. Angular: Architecture



Source: [3]

Modules: Angular apps are modular and Angular has its own modularity system called Angular modules or ngModules. Every Angular app has at least one Angular module class, the root module, named as AppModule.

Decorator: A directive/decorator on a component class adds the metadata, including a pointer to the associated template. Decorators are functions that modify JavaScript classes. Angular has many decorators that attach metadata to classes so that it knows what those classes mean and how they should work.

Directives: Directives and binding markup in a component's template modifies the views based on program data and logic. There are two kinds of directives structural and attribute directives.

Structural directives alter layout by adding, removing, and replacing elements in DOM.

Attribute directives alter the appearance or behavior of an existing element. In templates they look like regular HTML attributes, hence the name.

Angular libraries: Angular ships as a collection of JavaScript modules.

Injector: The injector provides services to a component, such as the router service that lets us define navigation among views.

Component: A component controls a patch of screen called a view. A component and template define an Angular view. Angular creates, updates, and destroys components as the user moves through the application.

Metadata: Metadata tells Angular how to process a class. The metadata in a Component tells Angular where to get the major building blocks you specify for the component.

Service: Service is a broad category encompassing any value, function, or feature that your application needs. A service is typically a class with a narrow, well-defined purpose.

Dependency Injection: Dependency injection is a way to supply a new instance of a class with the fully formed dependencies it requires. Angular uses dependency injection to provide new components with the services they need.

4.1.5. Real World Applications using Angular

a. Google uses Angular in many of their products like Cloud Console, Voice, Messages, Firebase.

b. Microsoft: Office Live WebApp and Add-in management, Xbox Web Store.

c. Travel: Delta Airlines Website, AirAsia

d. Food: GrubHub, PizzaHut, HelloFresh.

Aspire systems, a data analytics company uses Angular to provide an enriching user interface to their users. [3]
Microsoft Office Live web application uses angular to develop Add-ins. [4]

4.2. ReactJS

4.2.1. Overview

ReactJS is a declarative, component-based library which is deployed for the development of interactive user interfaces. It incorporates the view (V) layer in M-V-C (Model View Controller) pattern and provides better user experiences with blazing fast and robust web apps development. It efficiently updates through rendering the exact components to the view of each state and makes the data changes in the application. When data changes, react will calculate the minimum required changes using in-memory Virtual DOM and then updates the browser's displayed DOM efficiently.
A React application is a collection of discrete components, each representing a single view. When an application is built with React, the code is generally predictable since React wraps the DOM mutative, imperative API with a declarative one such that it raises the level of abstraction and simplifies the programming model. The combination of React and the rapid iteration cycle of the web, has enabled to make some excellent products including many Facebook components.

4.2.2. Key Features

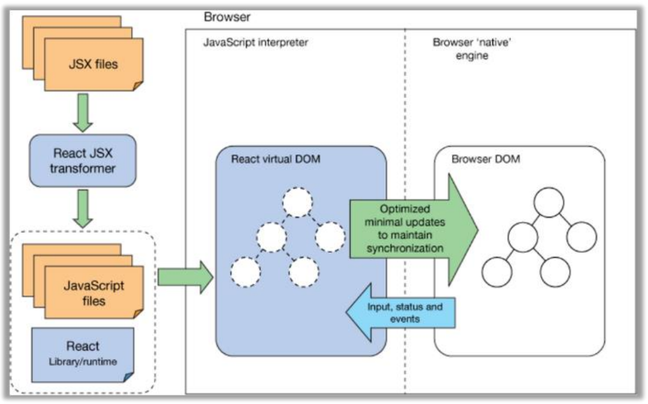
Component based library- React has a collection of components that is organized in a meaningful manner, and often provides a way to browse and preview those components along with associated assets. A Component encapsulates template, data, and logic (JSX). Components manages its own state and reacts to changes in state/data.

Virtual DOM: The virtual DOM (VDOM) is a programming concept where an ideal, or "virtual", representation of a UI is kept in memory and synced with the "real" DOM by a library such as React DOM.
One-way data binding/flow (Parent -> Child)- In React, data flows one way from parent to child. React one way binding is performed by listening to a "change" event, read from the data source and calling setState() on the components. ReactJS is designed for unidirectional data flow that is downstream is allowed and supported. This was done as the components need to be immutable and data within them must not change under any circumstances. Hence, react listens to data coming in one direction and not the other.

Promotes re-usable components- Reusable components are those React components that can be used multiple times in your application. As a result, they need to be generic enough so that it is free from complex business logic. A React component can be simply written as a JavaScript function. This function accepts props and returns a React element. Functional components can also be stateless. Rendering every component builds the user interface experiencing faster and efficient.

View and Controller at Single Place- View controller or controller view is a top-level component that holds all state and passes it to children as props. Views and controllers can have their own state.

4.2.3. ReactJS Control Flow



Source: https://developer.ibm.com/tutorials/wa-react-intro/

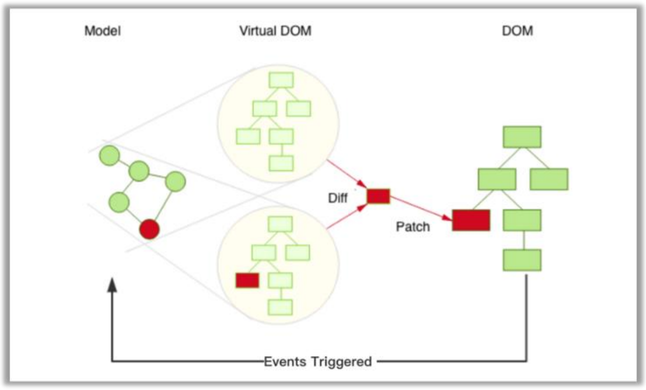
JSX: JSX stands for JavaScript XML. It is a syntax extension to JavaScript. JSX allows us to write

HTML and JavaScript combined in React Components thereby reducing the amount of code required to build our Components. JSX lets you specify the DOM elements before the components right inside of JavaScript files. This means the logic behind the components and the visuals are all in one place.

React JSX transformer: Transpiling is the process of traducing some source code from one programming language and producing the equivalent code in another programming language. Since, JSX is not understood by the browsers, a Transpiler will convert the JSX code into standard JavaScript code and create a bundle along with the react runtime library that is understood by all browsers. Babel with Webpack is a popular Transpiler and bundler for React.

The web UI components hierarchy is specified and feed it to Reacts virtual DOM. React takes care of the synchronization of your UI with the browser’s actual DOM at the appropriate time. Reacts virtual DOM implementation is self-contained and does not depend on the browser. It can even be used for server-side rendering. The virtual DOM performs an optimized diff of its internal state against the browsers’ DOMs and performs the minimal updates required to keep the UI consistent. The react runtime library will then watch for state change events and will keep the view in sync with the data efficiently using virtual dom.

4.2.4. ReactJS: How Virtual DOM Works?



Source: [3]

React provides a much efficient and light weight document object model. It does not interact with the DOM generated by the browser but reacts to the document object model stored in the memory.

React creates an in-memory node-tree cache of the HTML Document Object Model (DOM) wherein each node is an object representing a part of the UI element and their attributes, content, and properties. In react JavaScript objects will be bound to the components. These objects are called "State Variables". When the value of these objects changes either by a change in data source or user actions, a state change occurs for the component. Whenever there is a state change in the components, react creates a new virtual DOM representation. The difference between the previous Virtual DOM representation and the new one will be calculated to find optimized minimal updates for the browser's html DOM. This uses less memory than directly updating the browser's html dom. Hence it results in a blazing fast and robust performance of the application. The reason for highly efficient performance of the framework is essentially due to the virtual DOM feature of the library.

4.2.5. Real World Applications using ReactJS

a. Social Media: Facebook, Instagram, Twitter Entertainment: Netflix, Twitch
b. Payments: PayPal, Venmo
c. Communication: WhatsApp, Discord
d. Travel & Transportation: Uber, Airbnb
e. Facebook uses React in all their web applications: Facebook, Instagram, and WhatsApp.
f. PayPal uses react for building Isomorphic React Apps with React-Engine [19]
g. Netflix has created a robust video streaming application using React for the user interface and is using React for the Netflix TV software [20].

4.3. Vue.js

4.3.1. Overview

Vue.js is a lightweight front-end framework based on MVVM mode in Web applications. Model-

View-ViewModel framework focuses on declarative rendering and component composition to build UI and Single Page Web Applications. In MVVM mode architecture View Model, is a middleware and is responsible for communication between functions and data. Vue.js uses HTML-based template syntax and extends HTML attributes into Vue Directives. Vue Directive allows binding the rendered DOM to the underlying Vue instance's data. It Compiles the templates into virtual DOM render functions before updating the browser for better performance. Advanced features required for complex applications such as routing, state management and build tooling are offered by supporting libraries and packages. The main objective here is to provide the benefits of reactive data binding and composable view components with an API that is as simple as possible.

4.3.2. Key Features

One-way or Two-way data binding/flow: In One-way databinding the view (UI) part of application does not updates automatically when data Model is change. We need to write some custom code to make it updated every time a data model is changed. In Vue.js v-bind is used for one-way data flow or binding.
In Two-way data binding the UI fields are bound to model data dynamically such that when a UI field changes, the model data changes with it and vice-versa. In Vue.js v-model directive is used for two-way data binding.

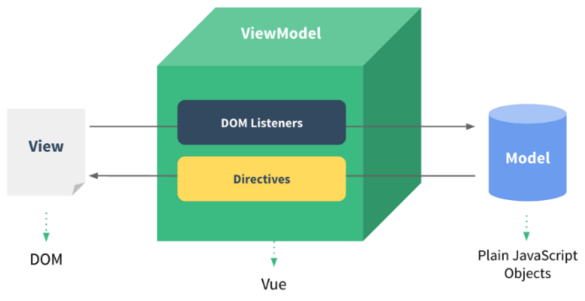
Reactivity System: Vue.js makes use of plain JavaScript objects binding and optimized re-rendering. When the JavaScript object changes, Vue will ensure that the bound HTML attribute is updated.

HTML based templates: VueJS uses html, JavaScript and css separately. It is very easy for a beginner to understand and adopt the VueJS style. The template-based approach for VueJS is very easy. Vue users can use template syntax or choose to directly write render functions using JSX. All Vue templates are valid HTML that can be parsed by specification-compliant browsers and HTML parsers.

Virtual DOM: Vue compiles the templates into virtual DOM render functions. A virtual Document Object Model (or “DOM”) allows Vue to render components in its memory before updating the browser. With virtual DOM, a JavaScript object is created which is the same as the real DOM. Any time a change needs to be made to the DOM, a new JavaScript object is created, and the changes are made. Later, both the JavaScript objects are compared, and the final changes are updated in the real DOM.

Lightweight framework: Vue can calculate the minimal number of components to re-render and apply the minimal amount of DOM manipulations when the app state changes.

4.3.3. Vue.js Framework Architecture



Source: [6]

Technically, Vue.js is focused on the View Model layer of the MVVM pattern. It connects the View and the Model via two-way data bindings. Vue.js uses DOM-based templating. Actual DOM manipulations and output formatting are abstracted away into Directives and Filters. Each Vue instance is associated with a corresponding DOM element. So, when a Vue instance is created, it recursively walks all child nodes of its root element while setting up the necessary data bindings.

Once the bindings are created, the DOM will then be kept in sync with the data. So that whenever data is modified, the DOM updates accordingly due to which most of our application logic is now directly manipulating data, rather than messing around with DOM updates. After the View is compiled, it becomes reactive to data changes. This makes code easier to write and to maintain.

4.3.4. Real World Applications using Vue.js

a. GitLab – Git based DevOps Lifecycle tool [7].
b. Grammarly – Online writing assistant.
c. 9GAG – User content sharing website.
d. Font Awesome – Font & Icon toolkit CDN.
e. Trustpilot - Online Review Community.
f. Laravel Website – PHP Web framework.

4.4. Svelte.js

4.4.1. Overview

Svelte is a component-based framework without any intermediate steps like virtual DOM. The components are defined in a HTML extended templating syntax. In Svelte state management is handled in pure JavaScript without complex third-party libraries. Svelte has its own compiler to convert the components into client-side vanilla JavaScript modules at build time and ensures that the browser does as little work as possible.

Svelte extends HTML by allowing JavaScript expressions in markup and provides directives to use conditions and loops, in a fashion like handlebars. It extends CSS by adding a scoping mechanism, allowing each component to define their own styles without the risk of clashing with other component's styles. It extends JavaScript by reinterpreting specific directives of the language to achieve true reactivity and ease component state management.

In Svelte, a component can have as many top-level elements as you like. Svelte runs at *build time*, converting your components into highly efficient *imperative* code that surgically updates the DOM. As a result, it allows to write ambitious applications with excellent performance characteristics.

4.4.2. Key Features

No Virtual DOM: Svelte complies the code to tiny, framework-less vanilla JS. Svelte writes code that surgically updates the DOM when the state of your app changes.

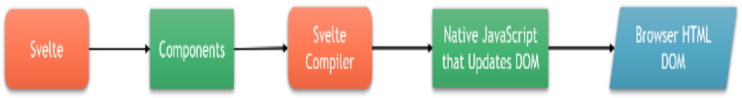
Truly Reactive: It can be used for building data-visualizations that need to display many DOM elements.

Code Splitting: With Svelte, code splitting is much more effective. This is because the framework is embedded in the component, and that component is quite small.

Fast & Efficient Performance: Svelte does not use DOM and uses specific directives and observables to efficiently update the browser's DOM.

Easy Learning Curve: Web developers with basic HTML, CSS, and JavaScript knowledge can easily grasp Svelte specifics in a short time and start building web applications.

4.4.3. DOM Update Handling in Svelte



Source:[**https://www.alphalogicinc.com/blog/best-javascript-framework-for-web-and-mobile-app/**](https://www.alphalogicinc.com/blog/best-javascript-framework-for-web-and-mobile-app/)

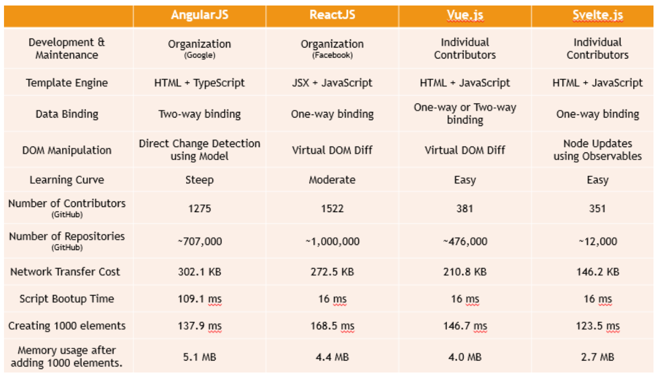
Svelte "compiles" the code at build time into vanilla JavaScript, so the final code can be executed without a runtime library. Svelte compiles the components into JavaScript instead of relying on concepts like Virtual DOM to update the browser DOM. Svelte offers true reactivity by surgically updating the DOM when the state of the app changes. This boosts performance and provides a developer experience that feels closer to plain JS than working with other frameworks.

4.4.4. Real World Applications using Svelte.js

a. 1Password
b. The New York Times
c. GoDaddy
d. Phillips
e. Rakuten
f. Grainger

[18]

5. Comparative Analysis



	AngularJS	ReactJS	Vue.js	Svelte.js
Development & Maintenance	Organization (Google)	Organization (Facebook)	Individual Contributors	Individual Contributors
Template Engine	HTML + TypeScript	JSX + JavaScript	HTML + JavaScript	HTML + JavaScript
Data Binding	Two-way binding	One-way binding	One-way or Two-way binding	One-way binding
DOM Manipulation	Direct Change Detection using Model	Virtual DOM Diff	Virtual DOM Diff	Node Updates using Observables
Learning Curve	Steep	Moderate	Easy	Easy
Number of Contributors (GitHub)	1275	1522	381	351
Number of Repositories (GitHub)	~707,000	~1,000,000	~476,000	~12,000
Network Transfer Cost	302.1 KB	272.5 KB	210.8 KB	146.2 KB
Script Bootup Time	109.1 ms	16 ms	16 ms	16 ms
Creating 1000 elements	137.9 ms	168.5 ms	146.7 ms	123.5 ms
Memory usage after adding 1000 elements.	5.1 MB	4.4 MB	4.0 MB	2.7 MB

Benchmark website: https://krausest.github.io/js-framework-benchmark/current.html

The benchmark was run on a Razer Blade 15 Advanced (i7-8750H, 64 GB RAM, Fedora 33 beta (Linux 5.9.8-200, mitigations=off), Chrome 87.0.4280.66 (64-bit))

Network Transfer Cost: It is the kilobytes of data downloaded (post-compression) to load all the resources into the page.
Script Bootup Time: It is the total milli seconds required to parse/compile/evaluate the scripts by the browser.
Angular is a full-fledged MVC framework with all commonly used modules like routing and ajax call support included. Others are component frameworks focuses only on the view part; other features can be extended using 3rd party libraries.
Angular enforcers typescript and difficult to learn for beginners. React is moderately complex and can learnt relatively easily. Vue and Svelte are beginner friendly.
Vue and Angular share a relatively similar approach to structuring their components: Both split up template (HTML), style (CSS) and logic (JS). While both are of the option to handle these parts either in one file or in three separate, Angular prefers the separation while Vue emphasizes the single file approach even offering a special file extension for this purpose: .vue.

Another aspect in terms of data handling is the approach to binding. React and Svelte uses one-way data binding only. Angular and Vue uses both ways by recommending the implementation of ng-Model and v-model, respectively. However, two-way binding may seem convenient at first but can cause difficulty as the application grows in terms of size and complexity. With two-way binding it can sometimes be hard to track which data gets updated where.

While Angular is the only fully featured technology in this comparison which indeed lives up to being called a framework, React and Vue are just view libraries that are often named in the same contexts as full-blown frameworks because they offer best practices for a complete development setup.
Angular and React are more mature and widely used. Vue is moderately mature and used sparsely. Svelte is still not mature enough and is currently mainly used to build small scale applications.

6. Conclusion

The use of a framework is very important for real world projects. We are trying to compare various frameworks. And in this paper, we are finding which frameworks are best suited at various scenarios by taking into consideration various parameters such as data binding, DOM manipulation, script bootup time etc. Through analyzing each of these frameworks based on the different key features of application platforms we were able to compare them thoroughly.

Angular is mature and is suitable for progressive web apps and large-scale enterprise web apps and for experienced developers. ReactJS offers a performant and easy way to implement UI component framework. When it comes to dealing with enormous amounts of users and data, react is un-doubted adoptable. It is the most popular among the frameworks and is also suitable for large-scale and mid-scale web apps. Vue.js is moderately stable and provides a beginner friendly component framework to build reactive user interfaces. Vue.js can be used for small to mid-scale web applications. Svelte.js is relatively not mature enough and is currently only used to build small-scale web applications. But because of its high performance, independency from 3rd party state management libraries and constant developments, it is gaining popularity.

Finally, there is no absolute winner or ‘the best framework’. Each framework has its strengths and weaknesses, we must learn different frameworks to see which is more suitable to the situation we are in and what functionality we seek.

7. References

1) Bibhishan Sutar, Dr. Prathibha Adkar. “Angular JS and Its Important Components”. IRE Journals ISSN: 2456-8880 Vol. 2 Issue 1 (2019) (146-148).
2) Google, Angular.io. “Conceptual Reference Guide”. (n.d.). Website: https://angular.io/guide/architecture
3) “Enriching UI with AngularJS”. White Paper (2014). Website: https://blog.aspiresys.com/uncategorized/whitepaper-enriching-ui-with-angularjs/
4) “Develop Office Add-ins with Angular”. (2020). Website: https://docs.microsoft.com/en-us/office/dev/add-ins/develop/add-ins-with-angular2
5) Anh Tu Le, “Developing a web application for task management with ReactJS”. Thesis, Turku University of Applied Sciences (2020) (7-15).
6) Sanchit Aggarwal et al. “Modern Web-Development using ReactJS”. Journal ISSN: 2349-7688 Vol. 5, Issue 1 (2018)
7) Sing Li. “React: Create maintainable, high-performance UI components”. IBM Developer Blog (2015). Website: https://developer.ibm.com/tutorials/wa-react-intro/
8) Vue.js. “Vue.js Reactive Data Binding Overview”. Website: https://v1.vuejs.org/guide/overview.html
9) Evan You, Chris Fritz. “State of Vue.js”. Montaerail (2019). Website: https://www.monterail.com/state-of-vuejs-report
10) Rich harris. “Svelte 3: Rethinking reactivity”. Blog (2019). Website: https://svelte.dev/blog/svelte-3-rethinking-reactivity
11) Felt.coop. “Why Svelte is our choice for a large web project in 2020”. Blog (2020). Website: https://github.com/feltcoop/why-svelte
12) Eric Liu. “Introduction to Svelte and its core concepts”. IBM Developer Blog (2020). Website: https://developer.ibm.com/tutorials/svelte-introduction/
13) Snipcart. “Svelte 3 Tutorial: A JS App with That Magic Framework You Heard About”. Developer Blog (2019). Website: https://snipcart.com/blog/svelte-js-framework-tutorial
14) Rich harris. “Svelte 3: Rethinking reactivity”. Blog (2019). Website: https://svelte.dev/blog/svelte-3-rethinking-reactivity
15) Rich Harris. “Frameworks without the framework: why didn't we think of this sooner?”. Blog (2016). Website: https://svelte.dev/blog/frameworks-without-the-framework
16) Snipcart. “How Vue Components Work”. Developer Blog (2019). Website: https://snipcart.com/blog/vue-component-example-tutorial
17) Slava Vaniukov. “Why Svelte is the Most In-Demand Framework for Web Development”. Blog (2020). Website: https://www.softermii.com/blog/why-sveltejs-is-the-most-in-demand-framework-for-web-development
18) Svelte-Website: https://svelte.dev/
19) Isomorphic React Apps with React-Engine. Website: https://medium.com/paypal-engineering/isomorphic-react-apps-with-react-engine-17dae662379c
20) Jordanna Kwok. “Netflix Likes React”. The Netflix Tech Blog (2015). Website: https://netflixtechblog.com/netflix-likes-react-509675426db