Comparison: Angular vs. React vs. Vue. Which framework is the best choice?

Jelica Cincović, Marija Punt
University of Belgrade, School of Electrical Engineering, Belgrade, Serbia
jelica.cincovic@etf.rs, marija.punt@etf.rs

***Abstract* - JavaScript is now one of the most popular programming languages in the software development industry, and as it is growing, a lot of new JavaScript frameworks have entered the market. JavaScript frameworks enable easier and faster front-end development, but with a wide variety of frameworks available, developers can often find themselves in a dilemma choosing which framework is the best for them. This paper focuses on the three most prominent frameworks (Angular, React and Vue) and performs a detailed comparison based on specific applications that were made and a survey that was conducted. The parameters by which the comparison was made are: popularity, performance, community support, learning curve, migrations and flexibility. As a result of this comparison, a manual was created to help both beginners as well as experienced developers, to choose the right JavaScript framework, depending on their needs and abilities.**

I. INTRODUCTION

The JavaScript language has always been at the core of front-end development, and it has been constantly evolving. From simple dynamic operations on Hyper Text Markup Language (HTML) pages called Dynamic Hyper Text Markup Language (DHTML), through a number of libraries such as jQuery, Polymer and Velocity and now with the possibility to execute JavaScript on the server side, and to make full-stack JavaScript web applications, this language is a leading trend. With JavaScript now in the center of interest of young developers as well as experienced companies, many JavaScript frameworks have emerged. JavaScript frameworks were developed in order to make front-end coding easier, more efficient and organized using components, third party libraries, routing, template systems and other modules. It is now simple to connect the client-side of the application made in one of the many JavaScript frameworks, with the server-side made in Node.js that can also communicate with databases. With an endless number of JavaScript frameworks such as: Angular, React, Vue, Meteor, Backbone, Mithril, and many others, it is hard to decide which one is the best choice.

To be able to choose from the variety of JavaScript frameworks, it is very important to know what benefits and advantages JavaScript frameworks offer. Angular and React have both earned their place as leading frameworks, but in the last few years an increase in interest for Vue is seen. There are a number of tutorials, online courses and articles that show how to work with each of these frameworks, how to quickly build applications or how to get the best out of what the framework has to offer. This paper sums up the differences between those frameworks and shows their strengths and weaknesses. It is important for every developer to wisely choose the appropriate technologies, to know its pros and cons, and to be able to select the best framework, based on the needs of the application he is developing. Without a thorough comparison, it is not possible to make the best choice with certainty. This paper is intended to guide developers who are unsure which JavaScript framework suits them best by comparing the three most popular frameworks (Angular, React and Vue).

II. METHODOLOGY

When comparing technologies, it is crucial to take a complete look at the various aspects, in order to provide an as realistic picture as possible.

The methodology of this research included the following phases:

- Detailed analysis of the ensuing frameworks: Angular, React and Vue;
- Comparison of the frameworks based on concrete applications implemented in these frameworks. Aspects that were compared are: popularity, performance, community support, learning curve, migrations and flexibility [1];
- Analysis of the results of the survey conducted on young developers that have never worked with JavaScript frameworks, and the ones that had previous experience, giving them the same problem to solve, choosing between Angular, React or Vue;
- Creation of a user manual based on conclusions that were drawn from the results of the comparison and survey.

III. FRAMEWORKS OVERVIEW

JavaScript frameworks were created with the main purpose of simplifying front-end coding and making it more structured and organized.

Most JavaScript frameworks are based on components. Each component contains both business logic and front-end part (HTML and Cascading Style Sheets (CSS)). The components are interchanged with the help of a router.

Also, developers often use terms “framework” and “library” as synonyms. The main difference between those two is in the control of the flow of the application. With libraries, developers are in charge of the flow of the application. They are calling the library from the code whenever is needed. On the other side, frameworks are in charge of the flow of the application, and not the developer itself. Frameworks provide developers, places to insert their code, but the framework calls the code when needed.

In the next chapters overview of Angular, React and Vue frameworks is presented.

A. Angular

Angular is a JavaScript framework developed by Google. It is used for efficiently creating advanced single-page web applications, and it is written in TypeScript [2]. TypeScript is a superset of JavaScript. It offers types, interfaces, async functions and decorators, and many others features, but at the end it compiles to plain JavaScript code which can run on any browser [3].

Angular applications follow modular programming concepts as a software design technique. Modular programming is based on separating program functionalities into logically independent and interchangeable modules. Angular modularity system is called `NgModules`. Every `NgModule` can contain components or service providers which belong to one logical unit. Scope of those components and service providers is defined by the containing `NgModule`. Every Angular application has at least one module, the root module called AppModule. A class decorated with `@NgModule()` represents a `NgModule`. Decorator `@NgModule()` contains several properties that describe the module:

- *`declarations`*: components and directives that belong to this module;
- *`exports`*: components and directives that should be visible in other modules;
- *`imports`*: other modules which are needed by this module;
- *`providers`*: services that this module exports to the global set of all services;
- *`bootstrap`*: root component.

Small applications usually have only root module, but more complex applications tend to split their functionalities into more feature modules.

Angular applications can very often import other modules from Angular libraries. For example, importing `FormsModule` from forms library can be achieved using the following code line:

```
import { FormsModule } from '@angular/forms';
```

A class decorated with `@Component()` represents a component class. Each component has its own business logic in TypeScript file, and its template in HTML and CSS files. Together, components with their template define a view. `@Component()` decorator has the following properties:

- *`selector`*: a CSS selector that tells Angular in which HTML tag to insert an instance of the component;
- *`templateUrl`*: relative path to the components template;
- *`providers`*: services that component requires;

For connecting parts of the template with the parts of the component Angular uses two-way data binding. There are four forms of data binding markup:

- Accessing component's property value from template using `{{component's_property_name}}` interpolation;
- Passing value from parent to child component using property binding `[child's_component_property]`="`parent's_component_property`";
- Calling component's method from template using event binding `(click)`="`component's_method`";
- Connecting input boxes from template forms with component's properties `[(ngModel)]`="`component's_property_name`".

Each Angular component has a lifecycle managed by Angular framework itself. Angular offers lifecycle hook methods to capture those moments in a component's life. Lifecycle sequence contains following methods:

- `ngOnChanges()` – called whenever data-bound properties change;
- `ngOnInit()` – called after initialization of component's data-bound properties;
- `ngDoCheck()` – called after ngOnChanges() and ngOnInit() to check for any additional change detection;
- `ngAfterContentInit()` – called after full component's content initialization;
- `ngAfterContentChecked()` – called after checking component's content;
- `ngAfterViewInit()` – called after component's view initialized;
- `ngAfterViewChecked()` – called after checking component's view for changes;
- `ngOnDestroy()` – called just before the component is destroyed.

Angular encourages developers to insert services into their applications. A service is a class with specific purpose. Components should just contain view-related functionalities and use services for any other more complex tasks. For the purpose of using services in components a Dependency Injection (DI) is inserted into Angular framework, which means a service can be injected into component and component can use functionalities of injected service. A service is a class with `@Injectable()` decorator. Inserting service into a component can be achieved using the following code line:

```
constructor(private service: Service) { }
```

which will tell the Angular framework that this component needs a service with the name Service.

B. React

React is a JavaScript framework developed by Facebook. Compared to Angular, whose components have logic and template in separate files, React considers these two things to be closely related and its components contain both logic and template in the same file. In order to implement this, it uses JavaScript XML (JSX). JSX looks like a template language but it has the full strength of JavaScript. It describes the looks of the user interface and produces React elements. JSX is optional, but there are a lot of advantages to it [4]. The following lines of code show the JSX example:

```
const name = 'My name';
const element = <h1>Hello, {name}</h1>;
```

Developers can also define functions and call them inside any HTML element. It is also possible to use JSX inside loops or if statements.

The basic building unit in React is React element. Elements are plain objects used for describing HTML elements. Developers can render their elements into a root Document Object Model (DOM) node, by applying the following code lines:

```
const element = <h1>Hello</h1>;
ReactDOM.render(element,
    document.getElementById('root'));
```

ReactDOM also only updates what is necessary, by comparing element's current and previous state.

Components represent independent logical units that based on the received input apply the business logic and return a React element which is viewed on the screen. The simplest way to create a component is to create a JavaScript function that accepts properties as an argument and produces a React element as a returning value. The following code lines, create Welcome component that takes property object as input, and returns React element that shows the name value passed through properties object:

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}
```

It is also important to emphasize that components must never change their own properties, and for that purpose the state concept was introduced.

State object can be used only in class components. It is defined in the constructor of the class, and the object can have as many properties as developer wants. Each of those properties can be updated using the method `setState()`.

From the previous, one can see that developers can easily add state object in class components, but in case of a function component, developers must use Hooks. Hooks are a new concept added in React version 16.8. They are completely optional, and they are mostly used to make coding in React easier and more approachable to the developers who are not familiar with the class concepts. For example, without creating state object we can add state property to a function component with the following code line:

```
const[count, setCount] = useState(0);
```

Count property is created, with the starting value 0, and it can be updated using the method `setCount()`. To accomplish this behavior, we used `useState` hook.

Similar to Angular, React components have their lifecycle. On creation and insertion of the component, following methods are called:

- `constructor()` – used for initializing local state before component is mounted;
- `getDervidedStateFromProps()` – called right before the render method;
- `render()` – renders the view;
- `componentDidMount()` – called after the component is rendered.

On re-rendering the component, following methods are called:

- `getDervidedStateFromProps();`
- `shouldComponentUpdate()` – called to inform React if a component should update;
- `render();`
- `getSnapshotBeforeUpdate()` – called right before the most recent update is committed to the DOM;
- `componentDidUpdate()` – called after the update occurs.

On removing the component from the DOM the following method is called:

- `componentWillUnmount()` – called before the component is unmounted and destroyed.

C. Vue

Vue is a progressive JavaScript framework for developing single-page web applications, created by Evan You. It is made to be very flexible and to integrate well with other libraries, and it does not require learning any new technologies [5].

The core of Vue is a reactive data binding system, that makes it very easy to keep your data and view in sync. Vue embraces the data-driven view concept, where we use special interpolation in our HTML code in order to bind data from the model. Very similar to Angular, the code for accessing property is: `{{property_name}}`.

There are also a lot of Vue directives prefixed with v- and they apply special behavior to the HTML elements. Some examples are: `v-if`, `v-for`, `v-model`, `v-on:click` and many others.

Every Vue application starts with creating a Vue instance with the following code line:

```
var vm = new Vue({options});
```

Developers can pass options object to a Vue instance, and all the properties found in that object are added to the Vue's reactivity system. When the values of these properties change, the view part of the application that represents these properties will automatically update.

Every instance has its own lifecycle, and during that lifecycle following lifecycle hooks (methods) are called:

- `beforeCreate()` – called after the instance has been initialized;
- `created()` – called after the instance is created;
- `beforeMount()` – called right before the render function is called for the first time;
- `mounted()` – called after the render function is called for the first time;
- `beforeUpdate()` – called when update occurs, before view is updated;
- `updated()` – called after the view is re-rendered;

- `activated()` – called when instance is activated;
- `deactivated()` – called when instance is deactivated;
- `beforeDestroyed()` – called right before the Vue instance is destroyed;
- `destroyed()` – called after the Vue instance has been destroyed;

As mentioned in previous overviews of the frameworks, components are reusable, self-contained logical abstractions that allows us to create sophisticated large-scale web applications. Components are Vue instances. Each component has its data and template and it can be enriched with methods. Component's data must be a function, so that each component instance can have its own copy of the data object. In the template part of the component, only one root element can exist. To register a component to the Vue component system, developer should examine the following code line:

```
Vue.component('my-component-name', {/*...*/});
```

This works very well for smaller applications, but as complex the applications get, the more difficult it is to take care of unique component's names or to write without HTML and CSS support. For that reason, the single-file components are made, with vue extension. Each file now represents the component, and it has three parts: template, script and style. This part makes Vue similar to React because of their view in terms of separating concepts. They believe that different concepts do not have to be separated by files, and belong to different architectural layers, but should be grouped into smaller logical units of components, where they will be easier to maintain in one file.

IV. Comparison

After the characteristics of each framework were presented individually, they were compared based on concrete applications implemented in these frameworks. Aspects that were compared are: popularity, performance, community support, learning curve, migrations and flexibility, as these aspects play a key role in choosing the right technology for any application.

A. *Popularity*

Popularity was measured based on two parameters. The first parameter was the number of web searches on Google for each framework for the past five years [6]. Results were provided by Google trends [7], and they show that React is currently the most searched framework and it has shown the highest growth in the previous period.

The second parameter was the number of Stack Overflow questions in past years. Results were provided by Stack Overflow trends [8], and they show that React has the most questions on this platform, Angular is on the second place and Vue on third.

B. *Performance*

Based on Js-framework-benchmark [9], and comparison of Angular, React and Vue on startup metrics and memory allocation was made and the results of comparison are shown in the Table I.

TABLE I.
JS-FRAMEWORK-BENCHMARK COMPARISON

Parameter name	Angular	React	Vue
Script bootup time (total ms required to compile script pages)	151.3ms	53.2ms	64.9ms
Ready memory (memory used after page loads)	2.7MBs	1.3MBs	1.2MBs

The results show that Angular framework has the largest overload and that it requires most memory allocation.

The concrete applications implemented in all three frameworks have the following functionalities:

- User login;
- User registration;
- Buying products in an online shop with implemented products cart.

Project sizes of all applications and number of code lines are presented in Table II.

TABLE II.
PROJECT SIZES AND NUMBER OF CODE LINES COMPARISON

Parameter name	Angular	React	Vue
Empty project size (without node modules)	510KB	615KB	453KB
Project size (without node modules)	660KB	774KB	614KB
Project size (with node modules)	233MB	145MB	93,5MB
Number of code lines	970	991	703

The results show that Vue frameworks is most lightweight, React is in the middle (number of code lines could be minimized with the usage of Hooks concept) and Angular has the largest overload with its node modules.

Applications were also compared with the help of Chrome DevTools options. The results presented in Table III only show what was previously established.

TABLE III.
CHROME DEVTOOLS COMPARISON

Parameter name	Angular	React	Vue
Loading needed scripts	4ms	3ms	3ms
Rendering pages	313ms	280ms	268ms

C. *Community support*

Community support was measured by comparing the number of Github repositories. The results are presented in Figure 1.

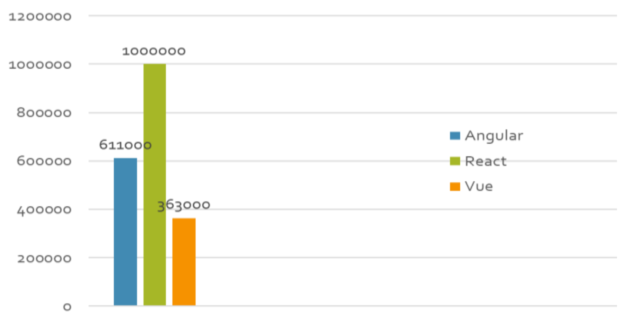

Figure 1. Number of Github repositories by frameworks

The results show that React is having the largest community support, Angular takes the second place and Vue the third. This is very expected given that Angular and React were developed by two large companies such as Google and Facebook, and Vue was the product of a smaller team led by Evan You.

D. *Learning curve*

As the overviews of the frameworks were discussed, it is shown that Angular uses TypeScript and React uses JSX. Those are all new technologies that can be challenging to the developers. Vue has the smallest learning curve as it does not require anything else but basic front-end technologies such as HTML, CSS and JavaScript.

E. *Migrations*

Migration represents a transition from older framework versions to the newer one. The goal of each JavaScript framework is to reduce the developer's job when migrating.

Angular often provides version updates, around every six months, which can be challenging to the developers, but they also provide help with migrations through their website. React strives to have a minimal number of major releases, and when they occur, they provide automated scripts which are run when the migration happens. Vue keeps their Application programming interface (API) about 90% the same throughout development, but when those small changes happen, they also offer help from the migration helper.

F. *Flexibility*

When it comes to flexibility in the application development Angular is very strict. Everything is provided in the Angular package and the defined code architecture must be respected. React and Vue are much more flexible in that field. They put no restrictions in the application structure and they also combine well with other libraries.

After analyzing the above parameters, a comparison table (Table IV) was made, that will help developers to select the right JavaScript framework based on their needs.

TABLE IV.
COMPARISON TABLE

Parameter name	Angular	React	Vue
Popularity	stagnation	rising	rising
Performance	largest overload	lightweight	very lightweight
Community support	medium	large	small
Learning curve	typescript	JSX, hooks	no need for new technologies
Migrations	often	rare	90% same
Flexibility	small	big	big

V. SURVEY RESULTS

In support of the results, a survey was conducted. The survey consisted of bringing together young developers that have never worked with JavaScript frameworks, and the ones that had previous experience, giving them the same problem to solve, choosing between Angular, React or Vue. The results of the survey are shown in the following figures.

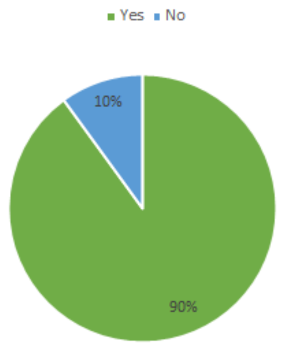

Figure 2. Pie chart view of the answers to the following question: "Have you used technologies like HTML, CSS and JavaScript before?"

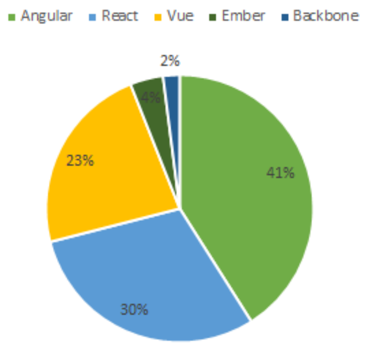

Figure 3. Pie chart view of the answers to the following question: "What JavaScript frameworks have you heard of?"

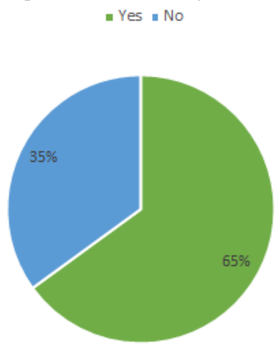

Figure 4. Pie chart view of the answers to the following question: "Have you used JavaScript frameworks before?"

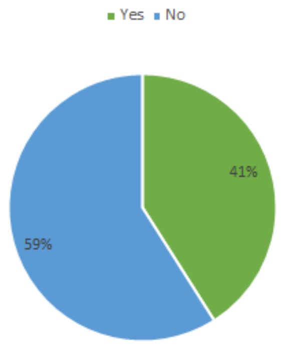

Figure 5. Pie chart view of the answers to the following question: "Was the first encounter with the JavaScript frameworks difficult?"

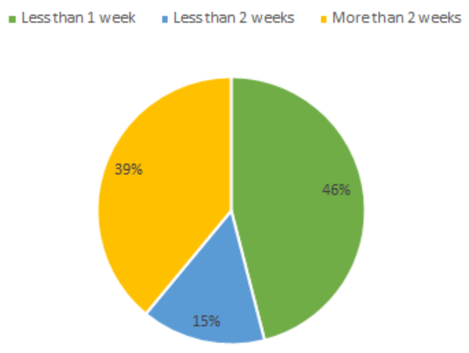

Figure 6. Pie chart view of the answers to the following question: "How long did it take you to get the project done?"

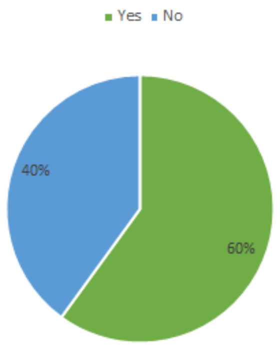

Figure 7. Pie chart view of the answers to the following question: "Would you like to continue training in this area?"

Developers who have used more than one JavaScript framework were asked to express their experiences and some of them are given below:

"Vue is easier to set up and learn, has built-in methods and helpers that help you to easily build something without doing the grunt work. - React offers high flexibility but has a higher learning curve and a lot of additional tooling needed for it to work".

"Vue seems the most approachable. React with redux has too many hidden things. Angular is somewhere in-between".

"Angular is a bit more complicated than React and Vue".

To sum up those results, it is seen that a lot of developers have heard of JavaScript frameworks, mostly about Angular, React and Vue. Most developers also have used those frameworks and they did not find them very difficult, and they are also willing to continue training in this area. Developers who have used more of these frameworks are agreed that Vue is the easiest one to learn.

VI. Conclusion

In this paper the comparison between the three most popular JavaScript frameworks (Angular, React and Vue) was presented. The frameworks were compared based on their popularity, performance, community support, learning curve, migrations and flexibility. Three applications providing the same functionalities to the users were developed in each framework. After development of the applications their code size, overhead generated by the framework and run-time performance were compared. Additionally, a survey among experienced and non-experienced developers was conducted.

It is concluded that Angular and React have a bigger community support relative to Vue, but also the deepest learning curve. React and Vue are more lightweight than Angular, more flexible in terms of application architecture and their popularity is rising. For developers who have no experience with JavaScript frameworks, Vue is the most approachable, and for those with more experience React would be a winner.

As possible improvements of this research, new comparison parameters can be added, such as: possible job positions in terms of popularity or new application's functionalities that will test frameworks' limits in terms of performance.

References

[1] Elar Saks, "JavaScript frameworks: Angular vs React vs Vue", Bachelor's Thesis, Haaha-Helia, University of Applied Sciences, 2019.

[2] TypeScript, https://www.typescriptlang.org/, [Accessed: 12 – Dec - 2019].

[3] Angular, https://angular.io/, [Accessed: 12 - Dec - 2019].

[4] React, https://reactjs.org/, [Accessed: 18 - Dec - 2019].

[5] Vue, https://vuejs.org/, [Accessed: 20 - Dec - 2019].

[6] Sanja Delčev, Dražen Drašković, "Modern JavaScript frameworks: A Survey Study", Zooming Innovation in Consumer Technologies Conference (ZINC), 2018.

[7] Google trends, https://trends.google.com/trends/?geo=US, [Accessed: 09 – Jan - 2020].

[8] Stack Overflow trends, https://insights.stackoverflow.com/trends?tags=angular%2Creactjs%2Cvue.js, [Accessed: 09 – Jan - 2020].

[9] JS-framework-benchmark, https://krausest.github.io/js-framework-benchmark/current.html, [Accessed: 10 – Jan - 2020].