International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Comparison of JavaScript Frontend Frameworks and Web API Services

Samarth Maganahalli[1], Prof. Rashmi R[2]

[1]*Information Science and Engineering Dept., R. V. College of Engineering, Bengaluru, India*
[2]*Assistant Professor, Information Science and Engineering Dept., R. V. College of Engineering, Bengaluru, India*

Abstract - *In this era of constant innovation and technology developments, some tools have stayed for long while many others have faded or have been dominated by newer technologies. However, we would all be able to concur that the JavaScript programming language will be staying for a very long term. This astounding bit of technology is cherished by millions for its adaptability — it very well may be deployed on both the server side and customer side, utilized for mobile applications as well as desktop. Hence, in this period, picking a tech stack has become a tedious task. Each factor - project objectives, resources, time, app size, end-users and budget need to be considered. Regardless of whether the work is a freelance project or just about forming ideas, both the upsides and downsides of each framework need to be considered in detail. To simplify the search, this paper gives insights about the top customer side JavaScript frameworks – Angular, React, Vue, and analyze their favorable circumstances and drawbacks. Also, interest in Web API services has increased rapidly. To exchange data among the applications in a standard manner is the fundamental objective of web services. In this paper, a comparison between two popular choices - SOAP and RESTful web services is drawn to make it easier for project managers to decide on the tech stack.*

Key Words: React, React.JS, Angular, Angular.JS, Vue.JS, REST, RESTful APIs, SOAP

1. INTRODUCTION

JavaScript was brought to the front row with HTML5 when Google released Chrome V8 engine in 2008. Before this release, JavaScript's main goal was working with Cascading Style Sheers (CSS) for better user interface and do perform some minor script actions like validations of forms. After the release of V8 engine in which the speed increased more than 58 times than any version of Internet Explorer, JavaScript's performance became comparable to that of Java or C++. Hence, web projects can meet speeds that are in par with traditional desktop software. Node.js extends usability of JavaScript such that it can now be used for server-side application development too. Although it has been eleven years since Node.js was released, there are many new JavaScript frameworks that are in the market and affect development of web applications. In the following sections, the paper will give insights on major front-end frameworks and libraries.

Traditionally, technologies like DCOM, CORBA and RMI were used to create client and server applications. These are used in systems that are highly coupled, meaning that both server and client are dependent on each other. But in reality, clients may not always have prior knowledge of web services before they actually use it, therefore Web Services are platform independent and loosely coupled. There are two major types of web services being used – SOAP principles based, and REST principles based. In the following sections, a comparison is drawn between the two and why RESTful web services have better performance.

2. LITERATURE SURVEY

Detailed study of the background, the internal structure and thorough analysis was carried out of each of the 3 front-end frameworks[1]. The types of web applications - Single Page application and Multi-Page application, differences between the two including pros and cons of each one of them and the analysis of these frontend frameworks for both the types of applications was studied[2]. The pros and cons of each framework and library under separate commercial criteria and possible future of front-end development in e-Business was gathered[3]. Detailed understanding of the web services SOAP and REST carried out, the differences in the architecture and performance comparison between the two was gathered[4].

3. FRONT-END FRAMEWORKS

3.1 Angular

Angular is an open source, frontend web application framework developed and maintained by Google. It has a Model-View-Controller (MVC) architecture and makes development, maintenance, and testing easier for developers. It is great for building highly active and interactive web applications, but it is most popular for Single page applications. It is sort of like an all-in-one framework since it contains tools for form validations, state-management solution, and routing. It is built on TypeScript which is a typed superset of JavaScript.

3.2 React.js

React.js is a front-end framework that is developed and maintained by Facebook which focuses on building and rendering components for a web page. Although everything is built in JavaScript, it is recommended to be used with JSX, a JavaScript syntax extension that describes what the UI should look like. It is different from Angular in a sense that it is not a complete suite of utilities but it is a collection of tools that helps build components that can be utilized in a page. This results in React.js being lightweight but it also required for the developers to use additional libraries to account for the features that React.js natively doesn't offer. For example,

developers need to take help of third-party tools to add routing in their application.

3.2 Vue.js

Vue.js is an open-source front-end framework that is not combines features from both React.js and Angular and is not backed by large tech companies like the two do. It was created by Evan You, former Google employee and is maintained by a core team led by himself. Like Angular, Vue.js is a complete suite of tools but with fewer features and packages. Programmers of all experience levels, even including those with a beginner-level experience in JavaScript would feel this framework as a straightforward option, thanks to its simplicity of syntax and extensive documentation.

4. COMPARITIVE ANALYSIS

4.1 Syntax

Projects that are built using Angular use TypeScript, which is a types superset of JavaScript. TypeScript cannot run in browsers natively, so tools are included in Angular that compile TypeScript code to JavaScript code that is browser compatible. Meanwhile, React.js uses a special JavaScript flavor called JSX, which is not really part of JavaScript language. Projects that use React.js are set up such that this "HTML in JavaScript" syntax is supported by compiling it to regular, browser compatible JavaScript code behind the scenes. Projects built using Vue, use regular JavaScript (TypeScript is supported too) and uses a feature called "Single File Components". It is a framework that is all about components (like React.js) but it splits the UI template files (i.e HTML) and core JavaScript logic apart (like Angular).

4.2 Learning Curve

Vue.js is found to be the framework that is the easiest to learn among the three. There are two core reasons for it:

1. No need to special setup: Getting started with a Vue.js project is as easy as importing Vue library into an HTML file and adding JavaScript code blocks to it. No additional project setup or "behind the scenes" compilation is required.
2. HTML and JavaScript only: Vue.js makes use of HTML and JavaScript only and no other special syntax need to be learnt. Meanwhile, both Angular and React.js need a bit more complex project setup i.e developer tools like Webpack to get started. Also it is needed to learn TypeScript (for Angular) or JSX (for React).

4.3 Features

Table - 1: Comparison of features provided

Feature	Angular	React	Vue
UI/DOM Manipulation	Yes	Yes	Yes
State Management	Yes	Yes	Yes
Routing	Yes	No	Yes
Form Validation and Handling	Yes	No	No
HTTP Client	Yes	No	No

4.4 Performance

4.4.1 DOM Manipulation

When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects. With the object model, JavaScript gets all the power it needs to create dynamic HTML. Basically, it is an API that can be used to interface the document with, and is available in many languages as a library. Manipulating/Changing the DOM means using this API to change the document

As shown in the Table 2, it can be concluded that all three frameworks have very similar performance in terms of DOM manipulation. All three frameworks are delivering great runtime performance and Angular, React and Vue are all getting used in production on big websites with a lot of traffic and complex user interfaces.

Table - 2: Comparison of DOM manipulation times (in ms)

Name	vue-v2.6.2-keyed	react-v16.8.6-keyed	angular-v8.2.14-keyed
create rows Duration for creating 1000 rows after the page loaded.	160.3 ± 5.6 (1.02)	167.0 ± 11.4 (1.06)	157.4 ± 12.7 (1.00)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	128.3 ± 2.2 (1.00)	129.5 ± 2.5 (1.01)	132.1 ± 1.7 (1.03)
clear rows Duration to clear the table filled with 10.000 rows.	155.4 ± 4.1 (1.12)	139.0 ± 4.0 (1.00)	243.1± 4.0 (1.75)

slowdown geometric mean	1.26	1.32	1.32

4.4.2 Startup Metrics

The major factor here is the size of the code generated and uploaded to the deployment server. Bundle sizes are often compared by building a basic "Hello World" app using the three frameworks. In this case, Angular results in a bigger size than the counterparts. For large sized applications, all three frameworks produce roughly the same size of code bundles. Other metrics that are related to startup can be gathered form the table below.

Table - 3: Comparison of startup metrics (in ms)

Name	vue-v2.6.2-keyed	react-v16.8.6-keyed	angular-v8.2.14-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,313.0 ± 39.1 (1.00)	2,526.8 ± 14.6 (1.09)	2,837.3 ± ± 55.9 (1.23)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	54.8 ± 44.9 (1.00)	78.5 ± 42.4 (1.43)	122.0 ± 24.6 (2.23)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	210.9 ± 0.0 (1.00)	260.6 ± 0.0 (1.24)	302.1 ± 0.0 (1.43)

5. CONCLUSION

Does any of the reviewed three frameworks stand out such that majority of the use cases can be recommended with it? The clear answer is no. The decisions to choose a certain framework depends heavily on the use case and other circumstances – size of project, developer knowledge, previous experiences, and other metrics. But from the comparisons performed, some key highlights can be obtained.

Why choose Angular?

- For apps that contain highly dynamic content
- For developing large, enterprise-grade applications
- MVVM (Model-View-ViewModel) that allows developers to work separately on the same app section using the same set of data.

Why choose React?

- For applications that depend on speed
- For applications that need versatility
- For applications that are suitable for international audience
- For cases where in skills learned in React can also be applied to React Native development

Why choose Vue?

- For developing very lightweight applications
- For integrating a framework into an already existing application
- For the sake of detailed documentation

6. WEB API SERVICES

6.1 SOAP

Simple Object Access Protocol (SOAP) is a standard web API service protocol that has been around for a very long time. It was developed by Microsoft to replace older technologies like Common Request Broker Architecture (CORBA) and Distributed Component Object Model (DCOM) that didn't work well on internet. These older technologies fail because they rely heavily on binary messaging while SOAP relies exclusively on XML which works better over the internet. Requests and responses that are made using XML in SOAP become very complex and in some cases these requests need to be built manually which also poses a problem since SOAP is intolerant of errors.

6.2 REST

Representational State Transfer (REST) is another standard web API service architecture that was developed as a response to the shortcomings of REST. The main aim was to provide a simpler process to access web services and fix the problems associated with SOAP. It is in fact a lighter-weight alternative to the hard to use and cumbersome SOAP. REST relies on URL for requests and can use four types of HTTP 1.1 verbs (GET, POST, PUT and DELETE) to carry out the tasks instead of using XML. Also, unlike SOAP, REST doesn't rely on XML for the response. Data can be output in many types of formats including – Comma Separated Values (CSV), Really Simple Syndication (RSS) or JavaScript Object Notation (JSON). The output can be received in a form that is easy to parse in the language that is being used for the application.

7. COMPARATIVE PERFORMANCE ANALYSIS OF WEB SERVICES

Table 4 and 5 gives an overview of the benchmark that was performed based on float and string data as parameter to these two web services[5]. The client service ran on a mobile emulator and the results are captured to reflect for the total response time and the size of the message.

Table - 4: Comparison of message sizes

Number of array elements	Message Size (byte)			
	SOAP/HTTP		*REST (HTTP)*	
	Concatenation of strings	Addition of float numbers	Concatenation of strings	Addition of float numbers
2	351	357	39	32
3	371	383	48	36
4	395	409	63	35
5	418	435	76	39

Table - 5: Comparison of response times

Number of array elements	Time (Milliseconds)			
	SOAP/HTTP		*REST (HTTP)*	
	Concatenation of strings	Addition of float numbers	Concatenation of strings	Addition of float numbers
2	781	781	359	359
3	828	781	344	407
4	828	922	359	375
5	969	1016	360	359

It is found that

1. The size of the messages communicated in case of RESTful web service is almost 10 times lesser than that in SOAP
2. The response time is also found to be almost 6 times lesser in RESTful than that in SOAP.

Now, multimedia conferencing (conferencing in audio-video, massive open online courses, online gaming, streaming) is taken into consideration. A performance evaluation is performed by considering different tasks like creating/ending conference, adding/removing participants in two different services based on SOAP and REST[6].

Table 6 and 7 depict the outcomes of the evaluation. It can be seen that

1. The delay in distributed environment is on average 4 times lesser in case of REST based services than SOAP based ones.
2. The delay on the same machine is on average 3 times lesser in case of REST based services than SOAP based ones.
3. The load on the network is nearly 3 times lesser in case of RESTful service than what is found in SOAP based web service.

Table - 5: Performance of SOAP service in a multimedia scenario

MULTIMEDIA CONFERENCING API	SOAP-based		
	Delay in a distributed environment (ms)	*Delay on the same machine (ms)*	*Network load (bytes)*
Create conference	848.4	381.7	767
Get conference information	818.6	335.3	546
Add participant	1325.3	334.2	578
Remove participant	1322.3	357	588
Get participants	787.1	342.7	615
Get participant information	766.2	346.7	619
End conference	1508.4	341.4	500

Table - 5: Performance of RESTful service in a multimedia scenario

MULTIMEDIA CONFERENCING API	REST-based		
	Delay in a distributed environment (ms)	*Delay on the same machine (ms)*	*Network load (bytes)*
Create conference	171.4	102.7	273
Get conference information	172.3	98.6	177
Add	368.8	103.3	200

participant			
Remove participant	382.9	107.2	195
Get participants	197.8	104.8	195
Get participant information	169.8	105	204
End conference	556.6	105.3	204

8. CONCLUSION

From the analysis performed, it can be safely said that in terms of performance, RESTful web service is a better choice compared to SOAP based web services. Web services based on SOAP principles have high response times, high network loads and lengthy messages. Also, most of the APIs that are built currently use REST and JSON, main reasons being that it consumes less bandwidth and is easier to understand by both the developers who build the APIs and the ones who consume them. The combination of REST and JSON has become some sort of a standard for majority of publicly accessible APIs. However, SOAP still remains to be a valuable protocol in cases where robust security, data privacy and integrity are a major issue.

REFERENCES

[1] Wohlgethan, Eric. "Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js." PhD diss., Hochschule für Angewandte Wissenschaften Hamburg, 2018.

[2] Kaluža, Marin, Krešimir Troskot, and Bernard Vukelić. "Comparison of front-end frameworks for web applications development." Zbornik Veleučilišta u Rijeci 6, no. 1 (2018): 261-282.

[3] Xing, YongKang, JiaPeng Huang, and YongYao Lai. "Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development." In Proceedings of the 2019 11th International Conference on Computer and Automation Engineering, pp. 68-72. 2019.

[4] Mumbaikar, Snehal, and Puja Padiya. "Web services based on soap and rest principles." International Journal of Scientific and Research Publications 3, no. 5 (2013): 1-4.

[5] Saad, Motaz K., Ramzi Abed, and Hatem M. Hamad. "Performance evaluation of restful web services for mobile devices." Performance Evaluation of RESTful Web Services for Mobile Devices (2010).

[6] Belqasmi, Fatna, Jagdeep Singh, Suhib Younis Bani Melhem, and Roch H. Glitho. "Soap-based vs. restful web services: A case study for multimedia conferencing." IEEE internet computing 16, no. 4 (2012): 54-63.

[7] Upadhyaya, B., Zou, Y., Xiao, H., Ng, J., & Lau, A. (2011, September). Migration of SOAP-based services to RESTful services. In 2011 13th IEEE International Symposium on Web Systems Evolution (WSE) (pp. 105-114). IEEE.