4ME305: Evaluation of JavaScript Web-Frameworks

Rema Salman
School of Computer Science, Physics and
Mathematics, Linnaeus University, Sweden
rs223bc@student.lnu.se

Abstract. In this paper, I describe and evaluate two chosen JavaScript frameworks, Node.js and React.js, based on the criteria list provided in the assignment description. I also describe my implementation of the assignment requirements, as covered in the practical part. Additionally, I discuss some aspects of the criteria based on my practical experiences.

Keywords: Evaluation, Criteria, Web-Framework, JavaScript, React.js, Next.js, Node.js, Nest.js.

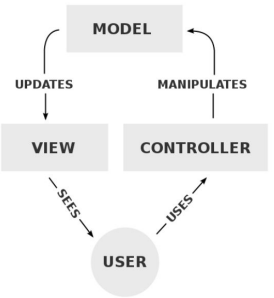
INTRODUCTION

Framework technologies were created based on established software concepts, aiming to facilitate design and development efforts, reduce software resources (cost and time), and enhance software quality. Software frameworks have many advantages, e.g. "modularity, reusability, extensibility, and inversion of control" [1]. For web-applications, their frameworks illustrate blueprints that provide standardized architectural patterns, templates, data persistence, security, etc. [2]. Web-frameworks is divided into server-side and client-side. The former runs on server environments and the latter runs in web-browsers, making it widely adaptable to JavaScript-based implementation [2]. JavaScript is a popular interpreted programming language that is used for implementing different applications and dynamic web-pages because it supports functional, imperative, and object-oriented programming[1].

The goal of this assignment is to gain a deeper understanding of web-frameworks evaluation depending on different factors and criteria, as well as the aspects that should be considered when designing, developing, and testing a web-application. For completing the evaluation, a simple web-application was developed using two of the proposed Javascript frameworks (Node.js and React.js) to demonstrate the development process and build some practical experience. The web-application implements a REST (Representational State Transfer) API (Application Programming Interface) for handling the logical and data transfers, while the GUI (graphical User Interface) allows the users' interactions with the application via a web browser. Due to time and size constraints of this assignment, I evaluated two frameworks and used one programming language (JavaScript) for the implementation.

FRAMEWORK EVALUATION

Modern architectural patterns support (e.g., MVC)



There are several types of frameworks concerning structural features, which is usually defined based on the application parameters, e.g. complexity and scalability [2-4]. Node.js ecosystem consists of several frameworks that can be installed via the Node Package Manager (NPM), which provides pre-made and reusable Node modules. Node frameworks have different classifications based on structural features. Node's most famous framework, Express, is classified as an HTTP server library. Express does not have to be implemented following the MVC pattern, but if it is needed, the package (express-generator[2]) helps fastly creating an MVC skeleton of the application. Meanwhile, Express is the foundation for other frameworks, such as Nest that is classified as a MVC frameworks. It is a TypeScript-based framework that combines object-oriented and functional structures. Both frameworks (Express and Nest) allow building simple and scalable web-applications [3]. Moreover, Node ecosystem has Next.js that is classified as full-stack frameworks on API framework and it is used in single-page applications (SPA) [4]. Finally, client-side frameworks rely on JavaScript for building scalable and interactive user interfaces. Therefore, React considers the View in the architectural pattern.

[1] About JavaScript - JavaScript
[2] Express application generator

Rendering technology

Different template rendering engines can be used in Express but the most popular Node template language is Pug (formerly known as Jade). The templates handle layout and data placeholders differently[3]. However, "most templating engines take a very HTML-centric approach", where some templating engines can run on both client and server [5]. Other famous template engines beside Jade are Mustache, Underscore Templates, Embedded JS Templates, HandlebarsJS, React, etc.[4] On the other hand, React renders on components directly, using a ReactDOM (virtual DOM). This virtual version enhances scripting performance in DOM operations.

Responsive Web support

Responsive support is the design of dynamically adjustable layouts and content of the web-site regardless of the device type or size. This method has become a standard expectation from modern web-apps, which provides users with easily accessed and consistent content [5]. It is typically manipulated with CSS and JavaScript rather than HTML since the same HTML is sent to different devices when loading websites. The manipulations include resizing, hiding, moving, or replacing elements [2]. One of the most popular AJAX-based frameworks for responsive design is bootstrap[5]. It can be included in both server and client-side, where it allows manipulating the DOM (Document Object Model) elements.

Form-Validation

A common practice is handling form validation in both client and server sides. Although Express uses middlewares to process parameters from the POST and GET requests (from forms), it does not provide any support for form handling operations. Meanwhile, React does not also support any form of validation. Therefore, developers tend to either manually check forms or via using external libraries to do the validation. E.g. Node module (express-validator) is used in the server-side for form validation and sanitization, while Formik library is used in React[6].

Internationalization

Node.js and React.js do not provide build-in functionality for internationalization/localization, however, developers integrate the module/library (i18n[7]), which uses dynamic JSON storage/files to replace the calls with the predefined ones, e.g. translations and transformation of units. Simultaneously, localization is based on business requirements, thus developers accordingly have to fulfil web-app specifications.

Friendly URL

SEO-friendly URLs make websites easy to be displayed on top of search engines (search-friendly). Although Node.js has a modern endpoint routing it does not offer any built-in functions to convert any strings into URL-optimized slug. Therefore, developers have to manually add JavaScript slugify and encodeURI(), where the former slugify strings and the latter encrypts/decrypts special characters to convert strings into valid URLs. These functions are usually implemented with multiple language support, i.e. ember.js extension [5].

Performance and stability

Performance represents the time needed from the web-application to respond/view content based on the user's interactions. It depends on the size and complexity of the web-app, which is usually related to architectural and client-side concerns: HTTP requests and data handling, program's execution phases/ start-up, create and modify Dom elements, scripting and rendering, and platform support. E.g. fetching and rendering multi-media data takes longer than simple text. These matters are also affected by the user's environmental differences in the browser such as JavaScript engine, cache utilization, and polyfills needs [5]. The open-source of Node.js ecosystem has its impact on stability. This can be illustrated by the major contributions to its technology. Although the core modules are stable, since Joyent and other big contributors supervise these technologies, the other tools may lack quality and high coding standards that are defined by global organizations. The stability of web-frameworks can also depend on different factors: the release date, community support, and popularity. Generally, community support correlates with popularity. Popularity affects the community support and thus the efficiency of the

[3] Template primer - Learn web development | MDN
[4] Template Engines
[5] Bootstrap: https://getbootstrap.com/
[6] Formik form validation in React: https://reactjs.org/docs/forms.html#gatsby-focus-wrapper
[7] Node module: i18n - npm

development process because developers rely on official documentation and community support as the main sources of information about the frameworks. Node and React currently have the biggest community support compared with other JavaScript frameworks. Express has the highest scoring based on Github rating [3], while React has its discussion forums and chats. At the moment, companies use frameworks as a standard part of their development processes[8], see companies that use these frameworks in Table 1.

Table 1. Companies using the frameworks.

Node	React
Netflix, Trello, LinkedIn, Uber, PayPal, Medium, and eBay.	Facebook, Instagram, Khan Academy, Asana, Reddit, Airbnb, BBC.

Learning curve

The learning curve can differ based on the developer experience. React is considered to have an easier learning curve than other client-side frameworks (Angular) and server-side because it is built based on JavaScript and JSX (a combination of JavaScript and XML) that is very similar to HTML syntax. React also has a chrome extension which makes debugging easier. In contrast, Node may be easy to learn but it takes effort to develop web applications with it. The asynchronous programming (non-block code executions) is the cause of higher learning-curve.

IMPLEMENTATION DETAILS

I choose Express and React frameworks to develop the app for this assignment because it is relatively simple. I have manually integrated ***Firebase OAuth API*** to handle the OAuth third-party services like Google and Twitter as well as the traditional email login/register operations. It makes managing users easier and provides a unified SDK for all the functions needed and it also handles read and write authenticated users' data to the Firebase Realtime Database and Cloud Storage. Concerning the user experience and usability aspects, I used react-bootstrap that is based on Bootstrap (AJAX framework) for allowing a responsive design and manually developed forms validation. For the server, I implemented it to work as a proxy to redirect the REST request from the client-side to the external service Twitter. The requests body contains the authentication (key and secret tokens) from the user login, tweet, and encoded media/photo. Several Node modules were used throughout the implementation, check the source code on GitHub and the Prototype Demo of the web-app.

DISCUSSIONS

Based on my experience and the size and complexity of the website, I choose to implement the server-app using Nest framework because it provides a further structure to my applications rather than the abstraction of Express. I did not get the chance to use template engine technologies from the server-side, which I assume would be interesting to see its implementation process. I have previously implemented other systems, following the MVC pattern, while most software I developed before considered performance and scalability aspects. Moreover, learning both mentioned frameworks was kind of easy for me since I come from a Software Engineering background; however, I am still relatively new to the web development field and I need time to develop my skills. Additionally, I was impressed with the community support and detailed documentation offered for developers either for learning or general discussions purposes.

REFERENCES

[1] M. Fayad, and D. Schmidt. (1997) "Object-Oriented Application Frameworks". Communications of the ACM. 40. 10.1145/262793.262798.

[2] Casteleyn, S., Daniel, F., Dolog, P., Matera, M. (2009). Engineering Web Applications. Springer Dordrecht Heidelberg, London New York.

[3] Demashov, D., & Gosudarev, I. (2019). Efficiency Evaluation of Node.js Web-Server Frameworks.[online] Available: http://ceur-ws.org/Vol-2590/short14.pdf

[4] Young, A., Meck, B., & Cantelon, M. (2017). *Node. js in Action*. 2nd edn. Manning, Greenwich.

[5] Brown, E. (2019). *Web development with node and express: leveraging the JavaScript stack*. O'Reilly Media.

[8] Understanding client-side JavaScript frameworks - Learn web development | MDN