Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Bachelor Thesis

Evaluation of the efficiency using ReactJS JavaScript library in development

Khasanov Ilkhomjon

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Ilkhomjon Khasanov

Informatics

Thesis title

Evaluation of the efficiency using ReactJS JavaScript library in development

Objectives of thesis

The main objective of the thesis is to evaluate the efficiency of ReactJS JavaScript library in development.
The partial objectives of the thesis are such as following:
- To make a current literature review of current state of the art of JavaScript based technologies.
- To develop a comparison test of the efficiency of ReactJS and Angular.
- To evaluate the efficiency of ReactJS and formulate the conclusion.

Methodology

A methodology of the thesis is based on study and analysis of information resources. The comprehensive overview of ReactJs will be described and evaluated alongside with Angular. The comparison will be made by using a relevant empirical method from multiple-criteria decision analysis. Depending on the research findings and results of the evaluation part, final conclusion and recommendation will be stated.

The proposed extent of the thesis

30 – 40 pages

Keywords

React JS, Angular, Components, JSX, Regular DOM, Virtual DOM, JavaScript, TypeScript, Flux, Redux, React Native, HTML, Library, Front – end Framework

Recommended information sources

AMBLER, Tim and Nicholas CLOUD. JavaScript frameworks for modern web dev . New York: Apress, 2015. Expert's voice in Web development. ISBN 978-1-4842-0663-8.
BANKS, Alex a Eve PORCELLO. Learning React: functional web development with React and Redux. Sebastopol, CA: O'Reilly Media, 2017. ISBN 1491954590.
CROCKFORD, Douglas. JavaScript: the good parts. Sebastopol: O´Reilly, 2008. ISBN 978-0-596-51774-8.
DUCKETT, Jon. Javascript & jquery: interactive front-end web development . Indianapolis, IN: John Wiley, 2014. ISBN 1118871650.
FAIN, Yakov and Anton MOISEEV. Angular 2 development with TypeScript . Shelter Island, NY: Manning Publications Co., 2017. ISBN 1617293121.
FEDOSEJEV, Artemij and BUSH, Alex. React.js essentials: a fast-paced guide to designing and building scalable and maintainable web apps with React.js. Birmingham : Packt Publishing, 2015. ISBN 978-1-78355-162-0.
FLANAGAN, David. JavaScript: the definitive guide. Fifth edition. Sepastopol, CA: O'Reilly, 2006. ISBN 978-0-596-10199-2.
GREEN, Brad and Shyam SESHADRI. AngularJS . Sebastopol, CA: O'Reilly Media, 2013. ISBN 9781449344856.
M., Vipul A. and SONPATKI, Prathamesh. ReactJS by example: building modern web applications with React. Birmingham, UK : Packt Publishing, 2016. ISBN 978-1-78528-964-4.
SENGUPTA, Doel, SINGHAL, Manu and CORVALAN, Danillo. Getting Started with React. Livery Place, 35 Livery Street, B3 2PB, UK : Packt Publishing, 2016. ISBN 978-1-78355-057-9.

Expected date of thesis defence

2019/20 SS – FEM

The Bachelor Thesis Supervisor

Ing. Miloš Ulman, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 11. 9. 2018

Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 19. 10. 2018

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 12. 11. 2019

Declaration

I declare that I have worked on my bachelor thesis titled "Evaluation of the efficiency using ReactJS JavaScript library in development" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on 29/10/2019 ______________________________

Acknowledgement

I would like to thank my supervisor Mr. Miloš Ulman, Ph.D.

Evaluation of the efficiency using ReactJS JavaScript library in development

Summary: The main aim this bachelor thesis is to evaluate how React performs alongside with Angular 2 in regard to find out the superior framework when developing the user interface in application.

Keywords: React JS, Angular, Components, JSX, Regular DOM, Virtual DOM, JavaScript, TypeScript, Flux, Redux, React Native, HTML, Library, Front-end Framework.

Table of content

1 Introduction 10

2 Objectives and Methodology 11

2.1 Objectives 11
2.2 Methodology 11

3 Literature Review 12

3.1 Web Technologies Architecture 12
3.1.1 Background 12
3.1.2 Front-end web technologies 13
3.1.3 Model-View-Control pattern 15
3.1.3 Single-page application 17
3.2 JavaScript 18
3.2.1 Client-Side JavaScript 19
3.3 JavaScript UI Frameworks and Libraries 20
3.3.1 Web statistics and Survey 22
3.4 Angular 27
3.4.1 Typescript on Angular 2 28
3.4.2 Directive and components 29
3.4.3 Modules 30
3.4.4 Services 31
3.4.5 Templates and Forms 32
3.4.6 Dependency Injection 32
3.4.7 Data Binding and Meta Data 33
3.5 React 34
3.5.1 JSX 36
3.5.2 Components of React, Props and State 36
3.5.3 Virtual DOM 38
3.5.4 Templates and Forms 39
3.5.5 Flux Architecture 40
3.5.6 Data binding in React 41

4 Practical Part 42

4.1 Real-World comparison of front-end frameworks with benchmarks 42
4.1.1 Performance 43
4.1.2 JS bundle size 44
4.1.3 First Contentful Paint 44
4.1.4 Line of codes 45

4.2 Stefan Krause’s benchmark tool 46
4.2.1 Head-to-head performance comparison 46
4.2.2 Performing the benchmark, Start-up time and Memory allocation 46

5 Results and Discussion 48
5.1 Which of the frameworks proposed better user interface? 48
5.1.1 Performance 48
5.1.2 Templating 49
5.1.3 Structuring data model 49
5.1.4 Code reshuffling 49
5.1.5 Package management 49
5.1.6 Learnability 50

6 Conclusion 51

7 References 52

8 Bibliography 52

List of abbreviations

SPA	Single Page Application
UI	User Interface
API	Application Programming Interface
DOM	Document Object Model
JS	JavaScript
JSX	JavaScript and XML
MVC	Model View Controller
MVP	Model-View-Presenter
MVVM	Model View-View Model
ES5/6/7	ECMA Script 5/6/7
HTML	Hypertext Mark-up Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language
FCP	First Contentful Paint

1 Introduction

Web development has seen a huge advent of Single Page Application (SPA) in the past couple years. Early progress was simple related a complete page to perform a change in the display or perform a user action. The problem with this was a big round-trip time for the complete request to reach the web server and back to the client. (M, et al., 2016)

Then came AJAX, which sent a request to the server, and could update parts of the page without reloading the current page. Moving in the same direction, it defined that the emergence of the SPAs. (M, et al., 2016)

Wrapping up the heavy front-end contend and delivering it to the client browser just once, while maintaining a small channel for communication with the server based on any event, this is usually complemented by thin API (Application Programming Interface) on the web server. (M, et al., 2016)

The growth in such apps has been efficiently complemented by ReactJS JavaScript library, created by Facebook in 2013, to build user interface (UIs) that can respond to user's input events along with creating and maintaining states. States are used to maintain changes to components, so the page loads faster by comparing only the changed and the updated part of the web page. React provides a one-way data flow that reduces complexity compared with a traditional data-binding system, which facilitates creating reusable and encapsulated components. (Sengupta, et al., 2016)

Therefore, ReactJS is not just another JavaScript library though many developers consider it to be the V of the MVC (Model-View-Controller) application. Nowadays, performance and portability are vital to build UI, mainly due to the large use of Internet-accessible devices and the fast-paced developmental phases of the projects. Hence this can result in complex front-end code. The need for using a library that helps user's code grow in both performance and quality is important. Otherwise, it will end up writing big HTML (Hypertext Markup Language) files with UI logic everywhere that takes ages to modify and can compromise code quality. (Sengupta, et al., 2016)

2 Objectives and Methodology

2.1 Objectives

The main objective of the thesis is to evaluate the efficiency of ReactJS JavaScript library in development.

The partial objectives of the thesis are such following:

Firstly, to make current literature review of current state of the area of JavaScript based technologies and gather a structured information about advantages and disadvantages of the frameworks.

Secondly, to develop a comparison test of the efficiency of ReactJS and Angular and to offer developers an objective data to simplify and facilitate their choice between described frameworks for their future projects.

At last, to evaluate the efficiency of ReactJS and formulate the conclusion using various approaches.

2.2 Methodology

A methodology of the thesis is based on study and analysis of information resources. The comprehensive overview of ReactJS will be described and evaluated alongside with Angular. The comparison will be made by using a relevant empirical method from multiple-criteria decision analysis. Depending on the research findings and results of the evaluation part, final conclusion and recommendation will be stated.

3 Literature Review

3.1 Web Technologies Architecture

3.1.1 Background

A traditional web application engineering comprises of a client and a web server. The client can be, for example, a browser in a work area PC or a cell phone. The client sends HTTP requests to the web server, which at that point plays out some usefulness dependent on the request and potentially restores some outcome. In the image beneath *(Figure 1)* is the example of a traditional web application information move model.

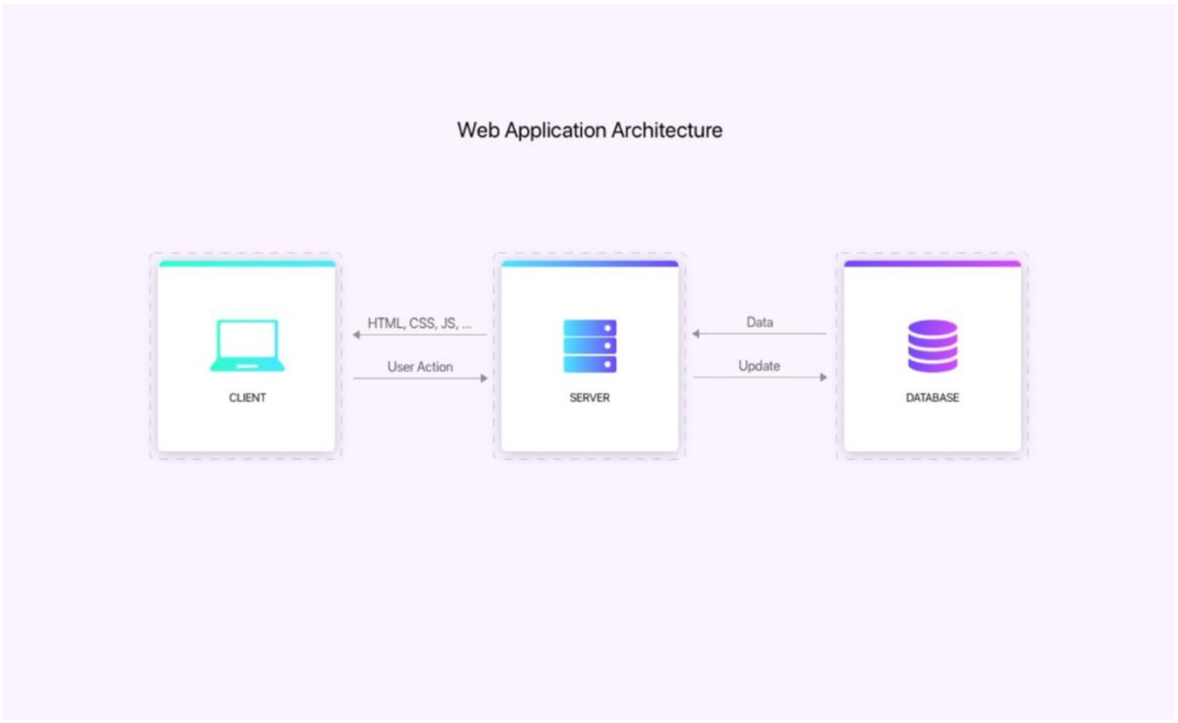


Figure 1. web application data transfer model (source: sombra)

In *Figure 1*, browser client requests a website page from the web server, and the web server thusly asks a few information from the database, lastly restores a HTML report, alongside CSS styles to the client's program.

A multi-page application for the most part comprises of various pages. At the point when the client clicks a link on the page, it diverts the program to an alternate HTML document existing on the server. The required template documents are recovered too, on the off chance that they are not yet stored in browser's cache. The server generally has different undertakings also, notwithstanding returning HTML records. The server can validate the client, make database changes, or for example, send messages.

In web application engineering, routing is a procedure where each not same URL compares to a particular page or content. In a multi-page application model, routing is more of a plainly obvious piece of an application as the pages are isolated additionally in the record framework. Utilizing URLs and routes, programs can bookmark, share a connection, and explore on websites utilizing back and direct buttons

Some tasks can happen additionally at the client's end. Handling in the browser mostly uses JavaScript and contains assignments like adjusting the DOM, making intuitive representations, or showing alerts on the screen. HTML DOM (Document Object Model) is a standard tree sort of structure, or model, which the program utilizes for getting to various components. There are numerous sorts of arrangements, however, the most foundation of the assignments separated is typically the equivalent in traditional multi-page web applications. (Sambra, 2019)

3.1.2 Front-end web technologies

With current web developing, there are 2 ordinary layers have in the web programming architecture, to be specific, the introduction layer, which is recognized as front-end and the data access layer prevalently recognized as back-end. Thus, front-end component includes in each and everything worried about the client or the customers in a web application. This incorporates every one of the users with the UIs. The front-end of web improvement is essentially handled with the utilization of standard advances, for example, JavaScript, HTML and CSS.

HTML is one of the huge tools and known as the foundation of the web innovations which works as a translator of ordinary plain content, pictures, and other sound, visual segments into the internet browsers. HTML records are masterminded and composed with blends of various labels, in the form of various Angular brackets. The fifth significant form of HTML was presented in 2013 as HTML5 and it carried a progressive adjustment to web improvement. HTML gives the major base to all innovations. (Dev, 2019)

Correspondingly, CSS is another significant web innovation, which is in charge of styling a HTML report. Whole special visualizations, format, colors, text styles and so forth are managed by CSS. Between whole front-end advances, CSS has an essential job in UI structure. HTML and CSS are approved with W3C an institutionalized Markup validator administration. In addition, HTML and CSS are remaining as the center advances required for present day web improvement. *Figure 2* delineates the range of innovations in front-end web development.

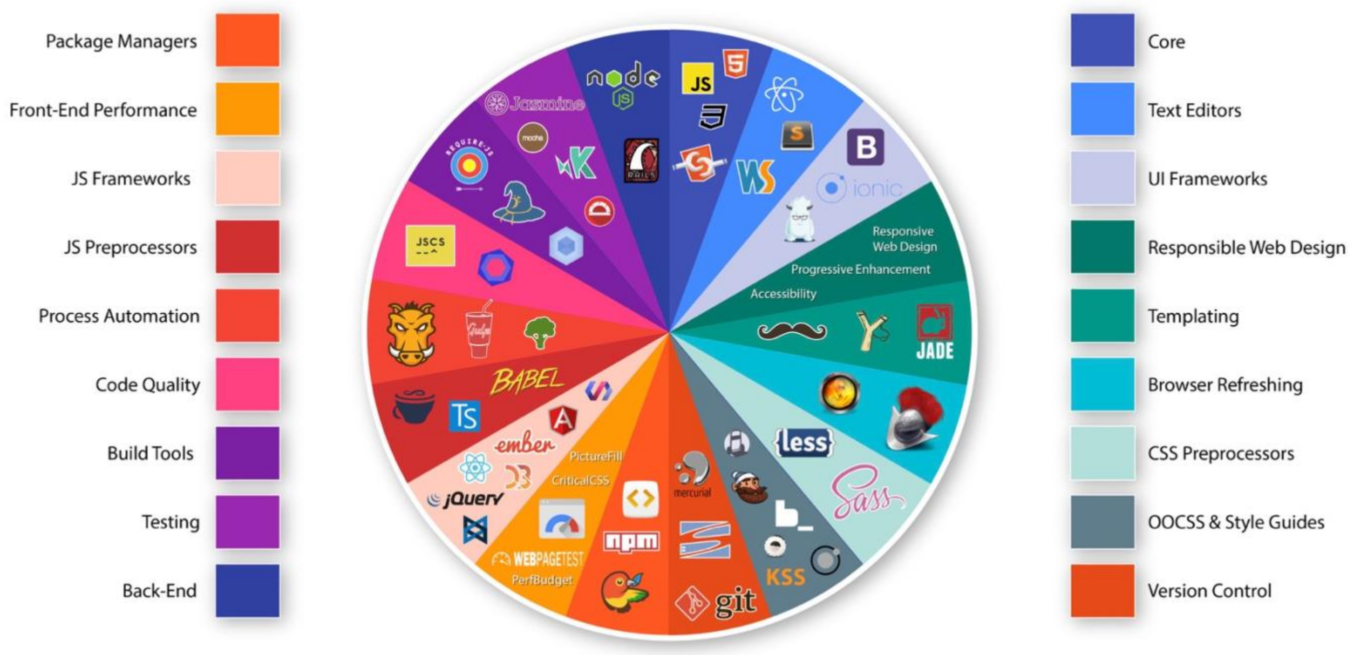


Figure 2 - Front-end Spectrum (source: medium)

Furthermore, another useful and ostensibly the best significant web innovation for completely practical and dynamic web advancement is JavaScript. JavaScript is a significant level, coordinated programming language, that is institutionalized with the ECMAScript language detail. JavaScript is regularly considered as a practical programming language, which floats in the whole present-day program. It has carried a transformation to World Wide Web, due to its similarity and versatility in various parts of the web. Along the presentation of various open-source libraries and frameworks over the most recent couple of decades, and JavaScript has changed the manner in which how developers are engaged with structure of creating the web programming. In any case, JavaScript is going forward and advancing so quick that the developers need to adapt new adjustments in the libraries and systems. Which has come about as an overwhelming assignment for each programmer to choose which frameworks or libraries to pursue and push ahead.

3.1.3 Model-View-Control pattern

The structure of the best UI application begins with a great application design. UI application plans are constantly connected with explicit arrangements of structure issues and patterns. While discussing client-side web advancement, there are distinctive structural examples, presented for a decent UI. Configuration examples as MVC (Model-View-Controller), MVVM (Model-View-View-Model), also MVP (Model-View-Presenter) gives great backing system to present day web UI advancement.

Model-View-Controller (MVC) is a structural pattern where the data model, the obvious view, and the utilitarian controller are isolated. The data model displays all the information substance of the application, the view part demonstrates noticeable portrayal of data, and the controller manages the preparing of the information and movements it between a model and a view. Figure 3 represents the MVC design.

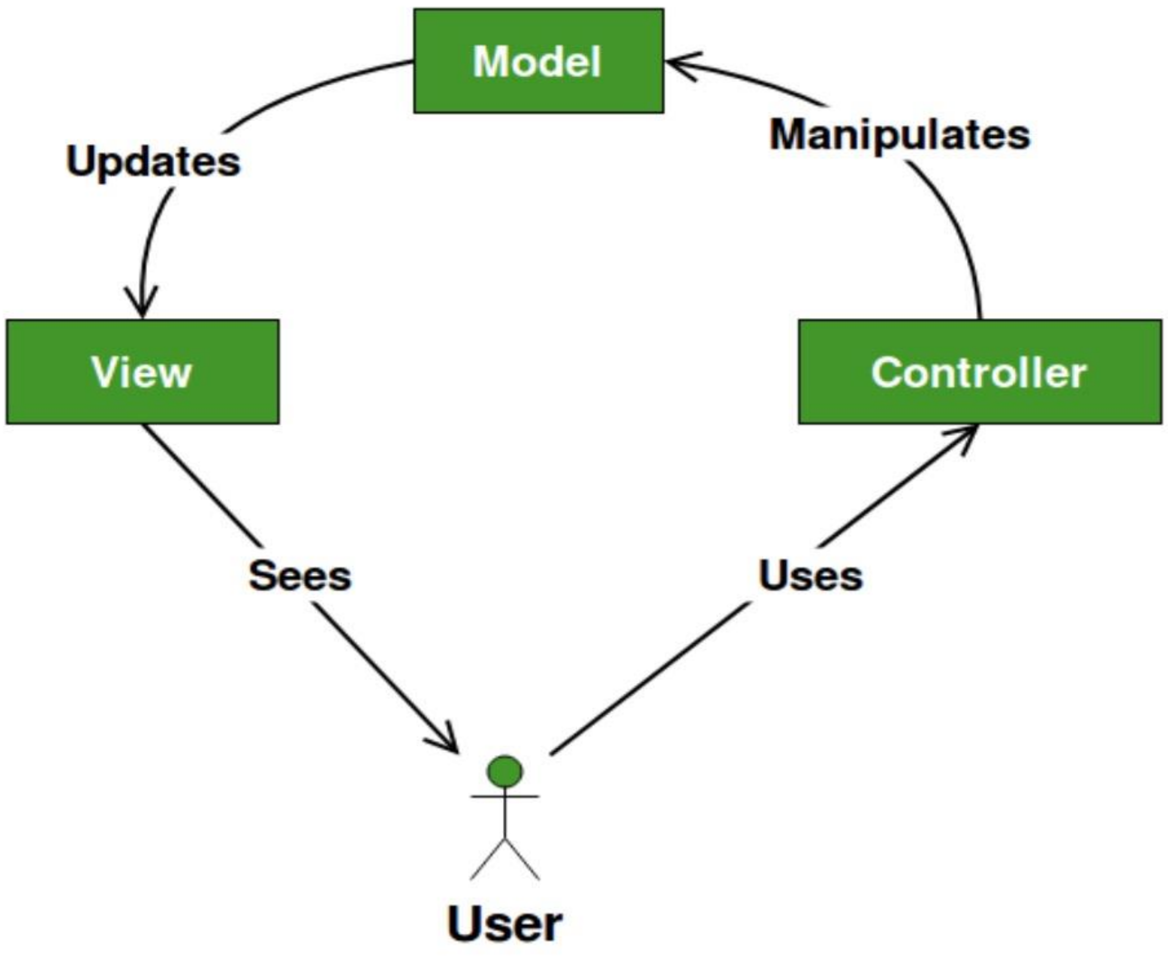


Figure 3. Model-View-Controller model (Source: geeksforgeeks)

In *Figure 3*, because of client's activities, the controller is manipulating the model, as well as, hence the view is refreshed also. The MVC model characterizes a standard method to separate application in parts, thus understanding the usefulness of the application ends up simpler indifferent of the technology utilized. The MVC model is utilized broadly in multi-page applications. (GeeksforGeeks, 2019)

Building up a UI application is an unpredictable task, when it is related with manipulating, processing and rendering huge business information. Sometimes developers get lost whenever into the visual parts of the applications containing countless client interfaces related with the business rationale. The control of enormous business information gets confusion to the code association the development process. The fundamental prerequisite for acceptable code association target around reusable, clear and viable codes.

The plan of a decent UI application begins with great application design. UI application plans are constantly connected with explicit arrangements of structure issues and examples. When discussing client-side web improvement, there are distinctive architectural examples offered for a decent UI. Configuration examples like MVC (Model-View-Controller), MVVM (Model-View-View-Model), and MVP (Model-View-moderator) gives great helping architecture for present day web UI advancement. A significant angle to note about the UI architecture tool is that it changes very quick. The innovations that are on promotion today will never again be in exist a year after. In this way, whatever the disarray the new innovation brings, it will consistently be a smart thought to start work with great application architecture.

1.1.3 **Single-page application**

A single page application (SPA) works distinctively as far as information move or routing, contrasted with a multi-page application. Single-page applications theoretically comprise just one single page. At the point when a user requests a web page, it gets a full websites code instantly. Arrived content frequently contains just a halfway complete HTML report. It is supplemented powerfully at the user's end, generally with JavaScript. Generally, the visible view is made by a progression of JavaScript functions, which supplements the HTML DOM on run-time. No further demands are required. However, all the obvious content is made and adjusted by JavaScript. Therefore, for example, the page changing in a SPA isn't generally page changing, yet rather replacing the content of the current page with another content. This content is created by JavaScript functions.

3.2 JavaScript

JavaScript is an interpreted programming language with object-oriented (OO) functions. Linguistically, the central JavaScript language looks like C, C ++ and Java, with programming constructs such as the if statement, the while loop, and the operator &&. The similarity ends with this syntactic agreement. Nevertheless, JavaScript is a bit of a written language, which means that variables do not have to have the specified type. Objects in JavaScript assign property names to random property values. So that, they are closer to hash tables or associative arrays (in Perl) than structures (in C) or objects (in C ++ or Java). The OO inheritance mechanism of JavaScript is based on prototypes such as little-known native language. This is very different from inheritance in C ++ and Java. Like Perl, JavaScript is an interpreted language and it inspired by Perl in different areas, for example its common expression and array handling functions. (Flanagan, 2006)

The main JavaScript language supports strings, numbers, and Boolean values as primitive data types. As well as, it includes integrated support for data, array and regular expression objects.

JavaScript is the most commonly used in web browsers, and in this context the general-purpose core is extended with objects that allow scripts to communicate with the user, control the web browser, and change the contents of the document that appears within the web browser window. The built-in version of c executes scripts that are embedded in HTML web pages. It is usually called JavaScript on the client-side to emphasize that scripts are executed by the client computer instead of the web server. (Crockford, 2019)

The basis JavaScript language and the integrated data types are subject to international standards and the compatibility between the implementations is very fine. The parts of JavaScript client-side are formally standardized, other parts are the actual standards and the rest parts are specific extensions of the browser. Compatibility with cross-browser is often a vital concern for JavaScript programmers on the client-side. (Crockford, 2019)

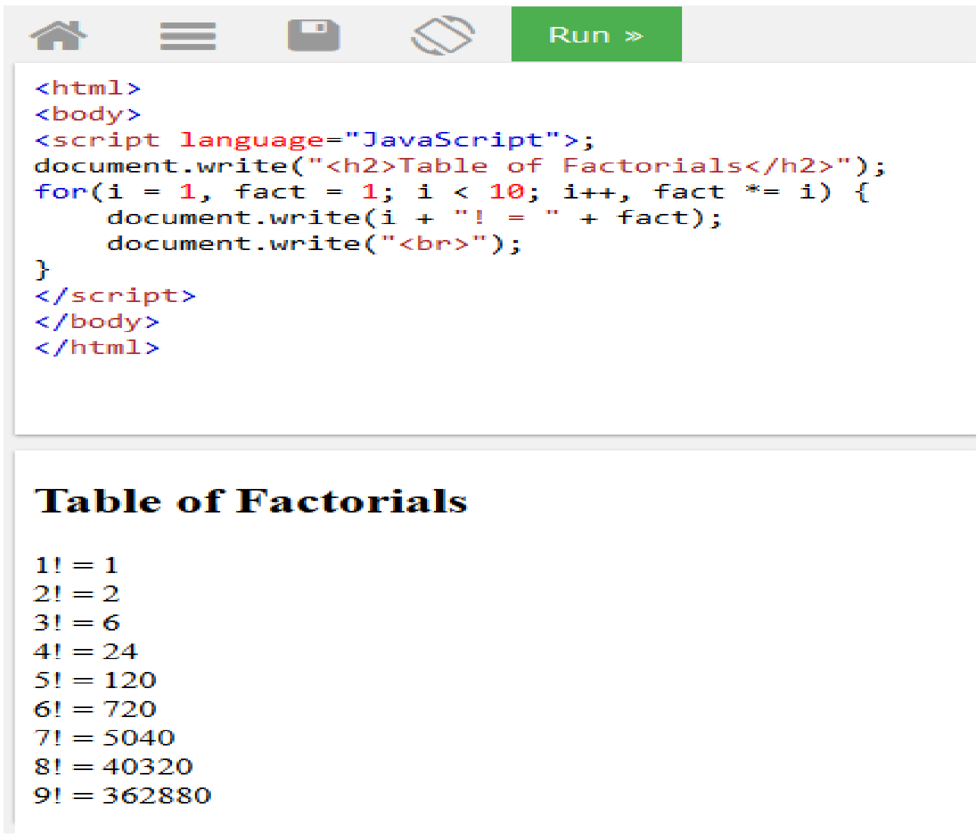
However, one of the most common misconceptions about JavaScript is that it is a simplified version of Java. In addition to the incomplete syntactical correspondence and the fact that Java and JavaScript can provide executable content in the web browser, for that reason the two languages have no relation to each other. (Crockford, 2019)

3.2.1 Client-Side JavaScript

When a JavaScript interpreter is embedded in a web browser, the result is client-side JavaScript. This is by far the most common variant of JavaScript, when most people refer to JavaScript, they usually mean client-side JavaScript. (Flanagan, 2006)

Client-Side JavaScript combines the scripting ability of a JavaScript interpreter with the Document Object Model (DOM) defined by a web browser. Documents may contain JavaScript scripts, and those scripts can use the DOM to modify the document or control the web browser that display the document. In other way, it can be defined that the client-side JavaScript adds behaviour to otherwise static web content. Thus, Client-side JavaScript is at the heart of web development techniques such as DHTML (Dynamic HyperText Markup Language). (Flanagan, 2006)

JavaScript program, or "script," embedded in a Web page. When loaded into a JavaScript-enabled browser, it produces the output shown in Notice that the <script> and </script> tags are used to embed client-side JavaScript code within an HTML file. (WebMaster, 2019) *(Figure 4)*



```
<html>
<body>
<script language="JavaScript">;
document.write("<h2>Table of Factorials</h2>");
for(i = 1, fact = 1; i < 10; i++, fact *= i) {
    document.write(i + "! = " + fact);
    document.write("<br>");
}
</script>
</body>
</html>
```

```
Table of Factorials

1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
```

Figure 4 - Client-Side JavaScript Simplified Examples (source: modified according to w3schools)

3.3 JavaScript UI Frameworks and Libraries

JavaScript frameworks are the application framework made and written in centre JavaScript to give a generic usefulness to a totally different application with the goal that the codes and functionalities of the product could be reused. JavaScript UI systems give framework highlights to various libraries, tools and API. When working in cutting edge projects, JavaScript systems support in architecting codes that make them increasingly coherent also secure.

Numerous frameworks have been presented throughout the years. Such as, ReactJS, Vue.js Ember.js, AngularJs, Angular 2 are the striking names that remain as essential JavaScript frameworks. These frameworks are made with various determinations, yet they all have shared objectives to facilitate the improvement procedure. Therefore, there are gigantic advantages of utilizing JavaScript frameworks for current client side-web advancement.

As, frameworks are the eventual fate of web development, utilization of frameworks in an application decreases time and cost that prompts the stock of value items for users. Frameworks give impeccable code architecture that makes codes utilized in the application increasingly readable and comprehensible.

Codes composed by utilizing frameworks can be reusable for example same codes can be reused the application on numerous occasions. This aides in the improvement of complex application quicker. Also, it lessens the game between user-side and server-side web improvement by making a decent link, good validation backing and correspondence with servers. The highlights utilized, for example, two-way data binding, controllers, segments and modules give unrivalled administration support in client-side web development. (Educba, 2019)

Essentially, with the requests of JavaScript, there has been dynamic improvement of JavaScript libraries by the enormous companies, for example, Microsoft, Instagram, Netflix, New York Times, WhatsApp, Khan academy, Dropbox, Reddit, Sony, Google, PayPal, Nike, General Motors and Facebook. JavaScript libraries are the piles of pre-composed codes in JavaScript which intends to make web-driven application advancement simpler. The fundamental purpose for the advancement of JavaScript libraries is to render the enormous information into the UI segment of the web. ReactJS, JQuery, vue.js are the eminent names of JavaScript libraries. (TechMagic, 2019).

JavaScript libraries can be sent into any of the progressed JavaScript frameworks, for example, AngularJs, Angular 2 and Backbone.js that pursues the MVC configuration design. JavaScript libraries give the V for instance View, in MVC configuration design.

AJAX backing is the most central thing for a smooth UI and JavaScript libraries fit into this.

JavaScript libraries have extraordinary open-source networks which are continually refreshing the changes.

Contrasting with the libraries of other useful programming languages, JavaScript libraries are relatively big and can deal with a gigantic quantity of data the user interface segments.

JavaScript libraries are basic and simple to understand for the beginner developers.

In no way JavaScript frameworks give a similar degree of performance as JavaScript Libraries.

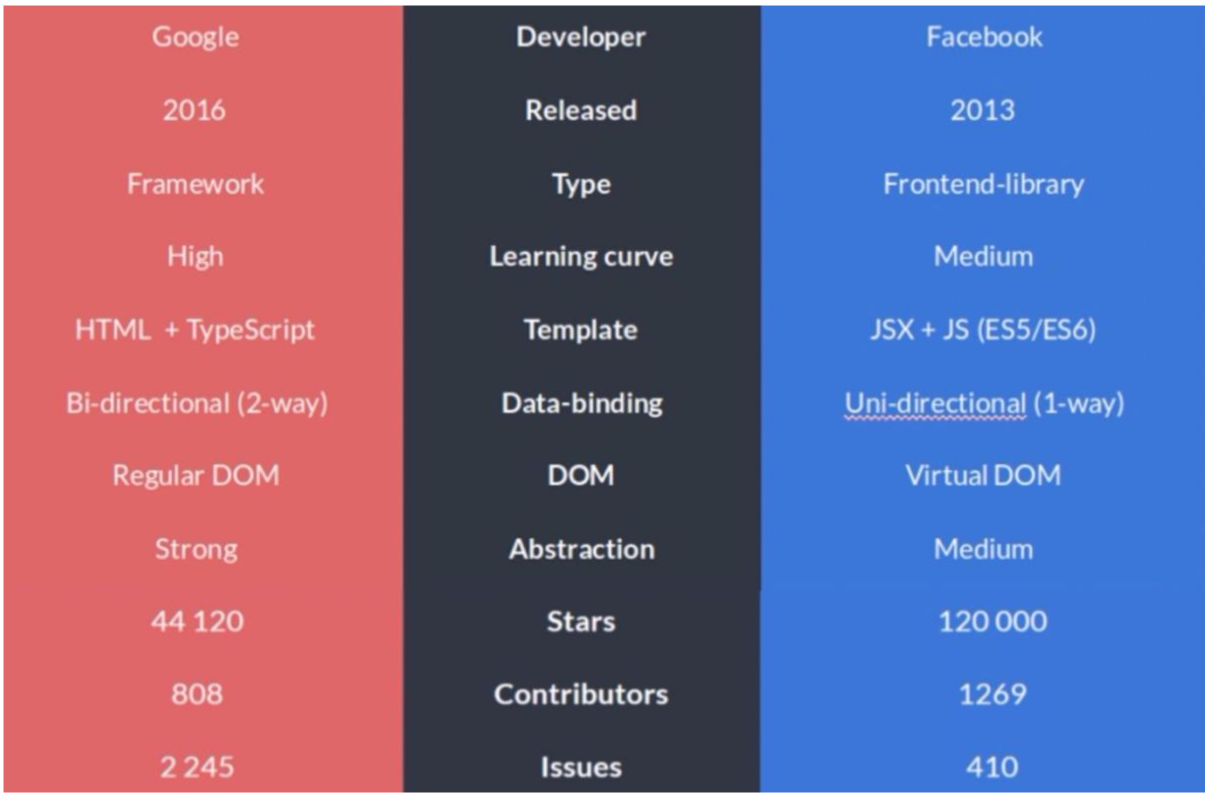
Both the frameworks and libraries of JavaScript give a broad scope of highlights for the plan of a strong UI. In any case, it has been truly a troublesome assignment to choose the correct innovation. An ongoing pattern demonstrates that Angular 2 and React are the most supported frameworks and libraries for web UI improvement. The utilization of both these innovations includes prevalent performance, code lucidity and code reusable highlights in the application. (Guide, 2019)

This theory consolidates an investigation and usage of features of React and the ongoing adaptation of AngularJs: Angular 2. it was presented in 2016 with the underlying arrival of beta form and numerous developers really relocated from its past version AngularJs. Despite the fact that React being was at its best in 2016. Developers migrated from React to Angular 2.

3.3.1 Web statistics and Survey

Angular and React are the frameworks and libraries from Google and Facebook individually, however the side-effects of both of these innovations are the aftereffect of open-source communities. The excellence of open-source networks is that it gives each developer with an opportunity to contribute and participate in any open-source venture. Open-source people group of Angular and React are extremely enormous and there are gigantic heaps of commitments from the developers in online networks like Github, Stackoverflow. (freeCodeCamp, 2019)

Github repositories of Angular and React demonstrate the fame of these two advances throughout the year. Straight on examination among Angular and React is demonstrated as follows:



Google	Developer	Facebook
2016	Released	2013
Framework	Type	Frontend-library
High	Learning curve	Medium
HTML + TypeScript	Template	JSX + JS (ES5/ES6)
Bi-directional (2-way)	Data-binding	Uni-directional (1-way)
Regular DOM	DOM	Virtual DOM
Strong	Abstraction	Medium
44 120	Stars	120 000
808	Contributors	1269
2 245	Issues	410

Figure 5 - side-to-side comparison between Angular (left) and React (right) as of 2019 (source: freecodecamp

Figure 5 demonstrates that React has greater prominence and has solid open-source communities over Angular starting at 2019. In any case, the steady arrival of Angular 2 was later than that of React. Angular 2 has not been embraced hardly. As, it has presented the entirely different style of improvement which isn't perfect with the past versions. Developers need to gain proficiency with the Angular 2 syntax structure and everything without any preparation and this may be definitive after some time. Be that as it may, taking a look at the information from Figure 5, Angular 2 won't meet a similar level as that of React inside a similar span of time.

Correspondingly, the Stackoverflow overview of 2019 which was come about 74,000 developers demonstrates that React is the most prominent than some other web innovation. Angular made the 9th place in the rundown.

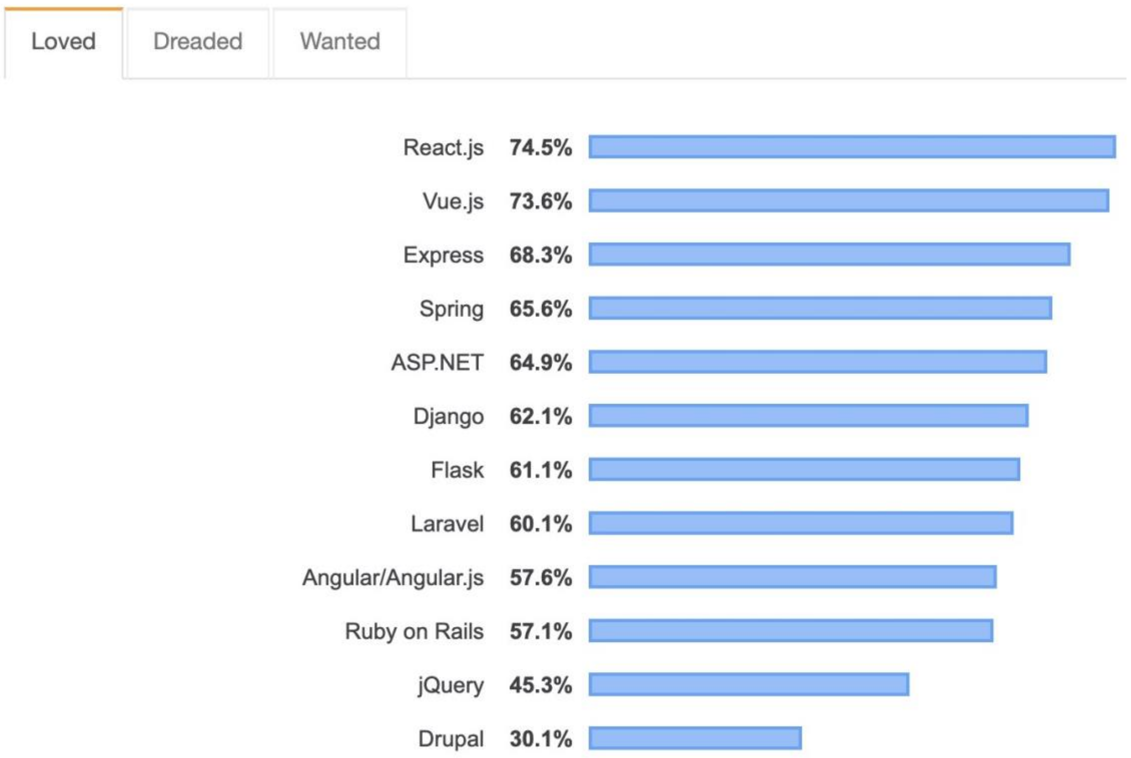


Figure 6 - Most Loved, Dreaded, and Wanted Web Frameworks in 2019 (source: stackoverflow)

As *Figure 6* demonstrates that React is remaining in the principal position in the survey aftereffects of the most loved web advances in 2019 with 74.5% out of 74,000 developers when Angular stands with 56.6% out of aggregate.

Google search trends could be precarious for demonstrating enthusiasm for specific subjects. As, you need to ensure that the terms are on equivalent balance. For this situation, we saw catchphrases "React javascript" and "Angular javascript" to ensure we weren't looking at inconsistent varieties of watchwords.

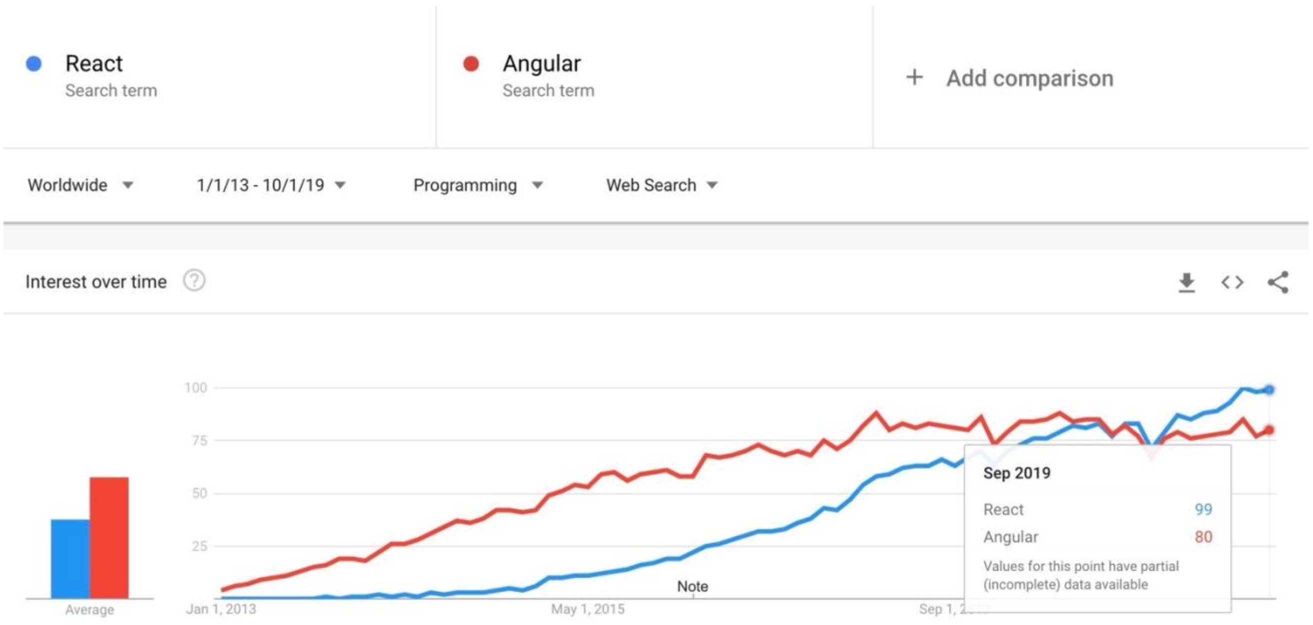


Figure 7 - Google search trends survey in 2019 (source: googletrends)

Figure 7 trend demonstrates that React is on top subsequent to outperforming Angular in 2019. While these numbers aren't indisputable, they give another point of view on React's predictable prominence. (Tecla, 2019)

Designers' fulfilment with React is affirmed by the 2018 State of JS Report also. For this situation, the diagram mirrors the developer fulfilment of the system, in light of whether they would utilize it once more, take a look at below results:

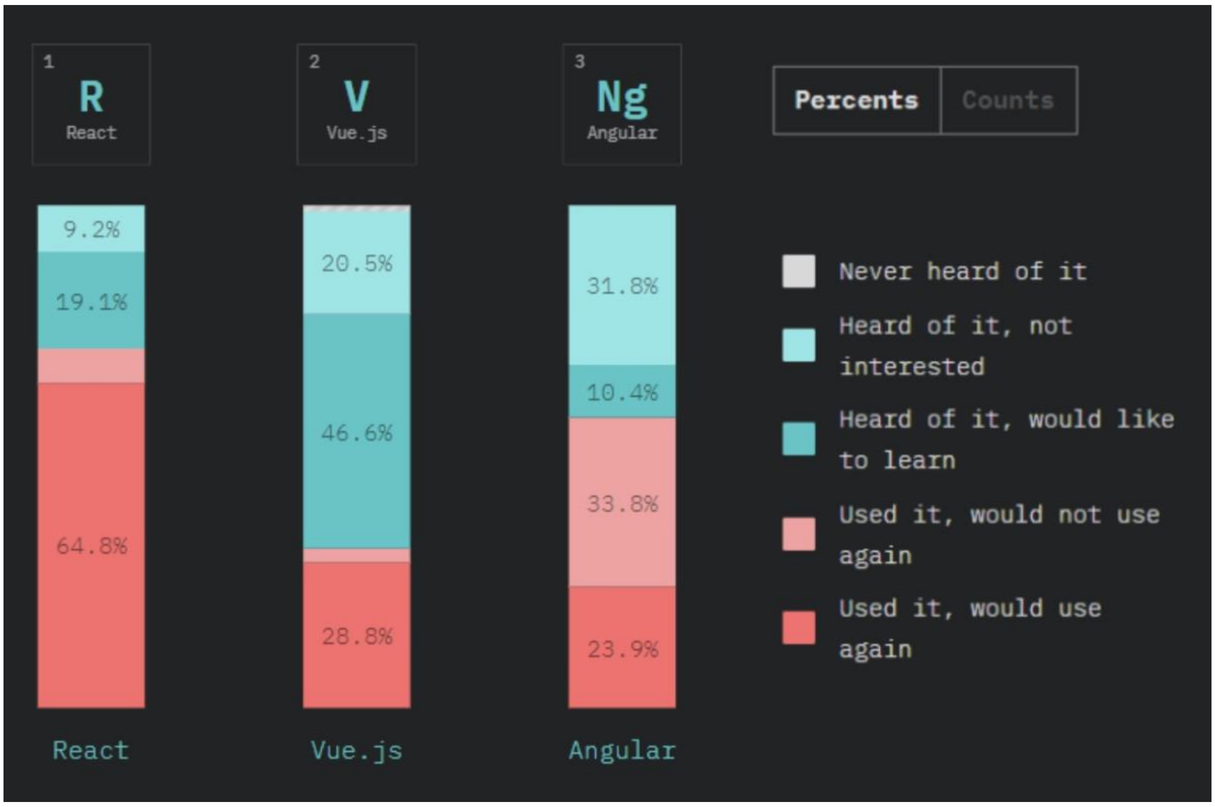
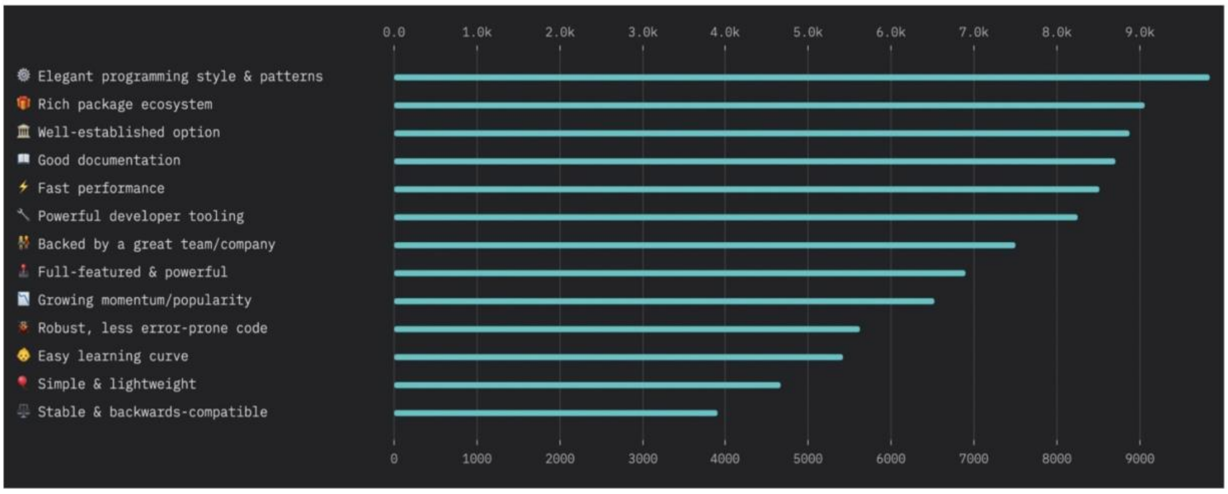


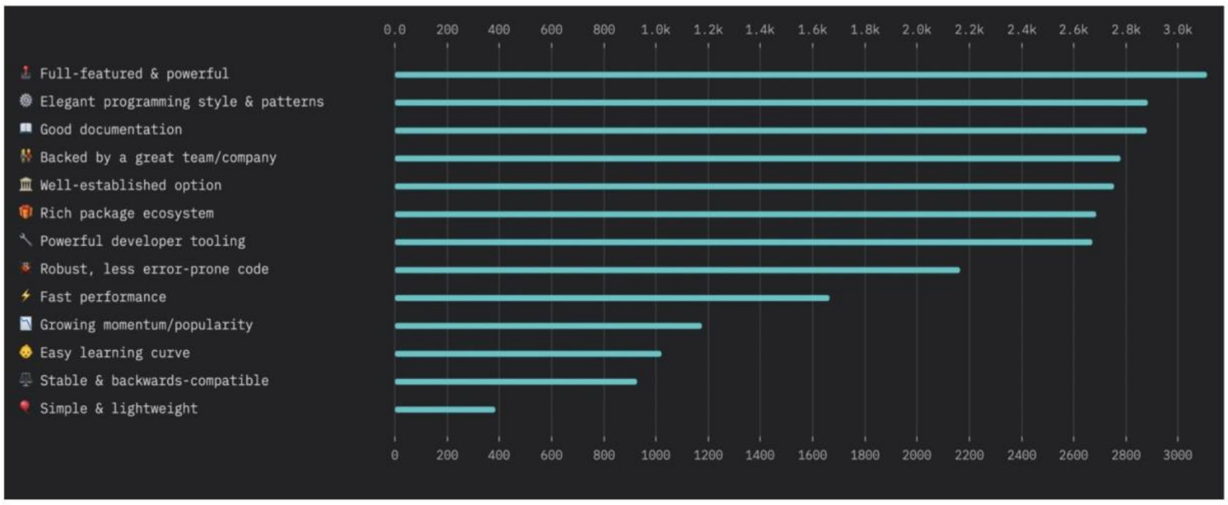
Figure 8 - Developer Framework Satisfaction in 2018 (source: Tecla)

As per *Figure 8*, React beats the two classes, with 64.8% saying they would utilize it once more, contrasted with 28.8% for Vue and 23.9% for Angular. Nonetheless, there's a major hop in fulfilment among Vue and Angular. 46.6% of developers might want to learn Vue, while 33% would not utilize Angular once more. That is a remarkable difference. (Tecla, 2019)

At long last, how about we study why those developers like these frameworks. As you can see the most enjoyed part of each in the State of JS Report down below:



React



Angular

Figure 9 – Most liked Aspects of Each Framework survey in 2018 (source: stateofjs)

As *Figure 9* shows the most liked viewpoint for every one of them was special. So, that gives us the point that no one framework is superior than the other. They all have remarkable highlights and perspectives that make them stand apart among the rest. However, React has more overall scores comparing Angular. (Tecla, 2019)

The fundamental explanation for most of the respondents inclining toward React is a direct result of its JavaScript driven features. Respond enables the developer to compose everything in one record and there is no requirement for discrete HTML, CSS and JavaScript. React consolidates HTML and JavaScript into the new features called JSX.

Numerous developers and organizations have adopted these two advances. The decision of the developer and organizations varies by their experience and the backend advances they pursue. The UI of well-known web applications like Netflix, Imgur, Airbnb, Yahoo Mail and so on. are composed and kept up in React. Likely, the page modules and single page applications like YouTube player, Instagram filters, and YouTube patterns are composed in Angular 2.

After far reaching research among Angular 2 and React, it appears that both are attempting to keep up their matchless quality in the realm of front-end web improvement. Being a crisp innovation, Angular 2 is on an extended publicity right now. React has figured out how to remain as a leader beginning from its release and it appears that React will be the developer's selection for such a significant number of years to originate from now.

3.4 Angular

AngularJs has turned into a principal framework of decision for each front-end developer for couple of years from 2009. The idea of AngularJs was presented in 2009 by Misko Hevery and Adam Abrons, however, now it is authoritatively been kept up by the huge enterprise Google. The underlying form of AngularJs version 1.0 was came out in 2012, and on August 2018, AngularJs released its most up to date form 1.7.8 and this is as yet considered as the present stable version. The name Angular is presented from the idea of the HTML labels for example HTML tags are Angular and Angular naming came up from the Angular brackets of HTML. AngularJs was created to conquer the helplessness and difficulties in a web application, for example, bug fixing, top notch UI and security. (o7planning, 2019)

After research for roughly two years, a similar group from AngularJs announced the last form of Angular 2 in late 2016. Angular 2 is centred around giving the best server-side rendering functionalities for immaculate UI. Angular 2 is essentially known as the names, Angular or Angular 2. There are numerous problems overwhelmed by Angular 2 over AngularJs. A couple of them are featured underneath. (stuff, 2019)

Firstly, Angular 2 gives a lot better execution over AngularJs. Also, Angular 2 applications put small weight on the virtual machines as a result of less memory utilization. Unrivalled format reusing functionalities, cache review highlights are among the prominent things in Angular 2 that give better execution.

Secondly, Angular 2 has progressed and amazing templating highlights. The advancement condition and IDES are present day in Angular 2 and are superior to the runtime of AngularJs. This causes the format authors to discover the mistakes quick and can spare time in the layout composing process.

Thirdly, Angular 2 is progressively worried for the future works in examination with AngularJs, since web advances are increasingly committed on live information catching with the web works, for example, web crawling. Angular 2 is engaged to satisfy the requirements for smooth activity of server-side rendering to the UI.

Lastly, Angular 2 has feature support for web stage portable applications, such as mobile applications. The local mobile user interface framework of the Angular 2 makes the completely practical local portable user interface in iOS and android platforms. (colorlib, 2019)

3.4.1 Typescript on Angular 2

Typescript is an open-source programming language created by Microsoft. It's scripting language written in core JavaScript, and typescript much like present day item arranged programming languages, for example, C#, C++ or Java. Typescript is utilized to build up the web-based JavaScript application on client-side or server side. (C#Corner, 2019)

Despite, Angular 2 applications could be consisting with ES5, Es6 and dart, Typescript gives a wide scope of highlights in making an enormous Angular 2 application. The primary purpose for utilizing Typescript on Angular 2 applications is that it gives extraordinary tooling features. Exploring, code refactoring, auto finishing, static composing and affluent syntax generator are among the eminent highlights of Typescript in creating Angular 2 applications. Figure 10 demonstrates the connection between Typescript with the ES5, ES6 and ES7.

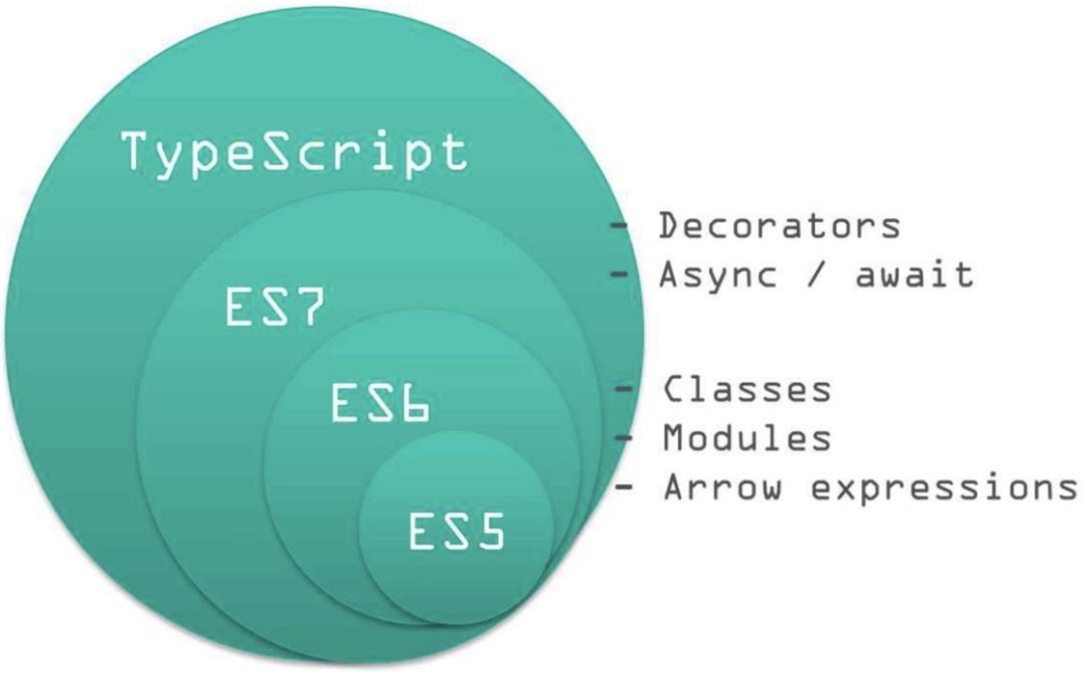


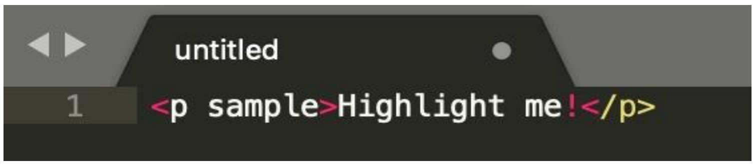
Figure 10 - Relationship of Typescript with ES5, ES6, and ES7 (source: infoQ)

For each situation *Figure 10* TypeScript generates code that carries on the equivalent in ES5, ES6 and ES7. There isn't distinction in expressivity of TypeScript and ES6. The thing that matters is that TypeScript compiler encourages you with static analysis of your code. Apart from that, whatever you can program in ES6, you will be able to program in TypeScript and the other way around. (stackoverflow, 2019)

3.4.2 Directive and components

Essentially, there are three sorts of order in Angular 2, attributes, structural and component directives. Component orders manages the templating part to make the UI, while structural directives change the record article models format by including and disengaging the DOM object. So also, attributes assume a job in changing the visual appearances of segments and components.

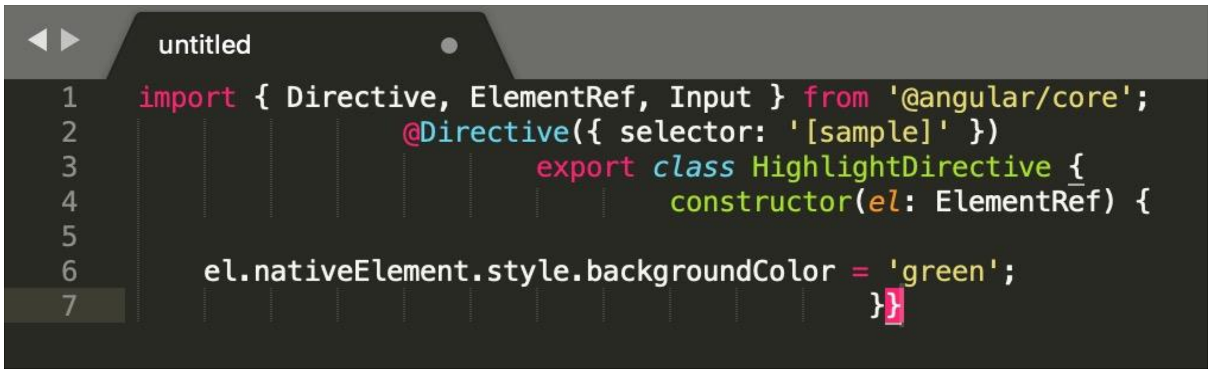
A case of a code property and its application into the Angular 2 directive is demonstrated as follows shown in *Figure 11.*



```
<p sample>Highlight me!</p>
```

Figure 11 - Listing 1, Angular 2 code attribute. (source: own)

Here, sample attribute is written in the directive code demonstrated as follows *Figure 12.*



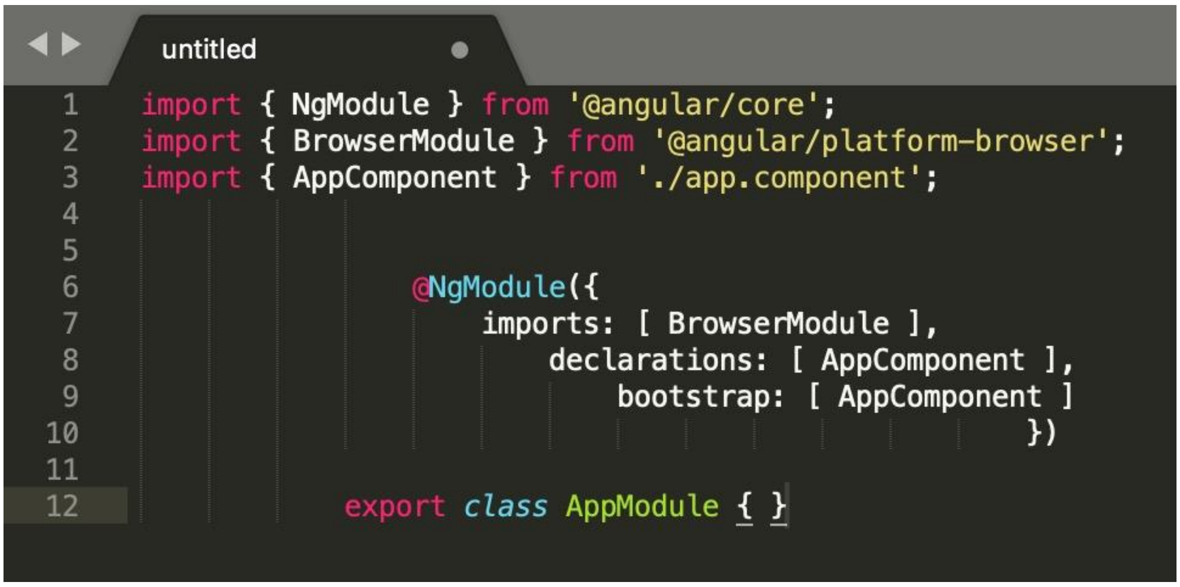
```
import { Directive, ElementRef, Input } from '@angular/core';
                @Directive({ selector: '[sample]' })
                        export class HighlightDirective {
                                constructor(el: ElementRef) {

    el.nativeElement.style.backgroundColor = 'green';
                                                }}
```

Figure 12 - Listing 2, Angular 2 Directive application. (source: own)

The model directives appeared in listings 1-2 is enhanced with the import explanation which underscores the Angular 2 core library that has three explicit symbols: directive, ElementRef (feature that allows the security assault in DOM item) and Input. The directive gives functionalities to the @Directive decorator. Thus, ElementRef goes about as a constructor for code access in DOM, and input is in charge of information binding. (Angular, 2019)

3.4.3 Modules

The route module in Angular 2 is a class that depicts how various pieces of the application are overseen all the while in Angular 2. Each Angular 2 application should be equipped with at least one Angular 2 module that is known as the route module. Naming of the Angular 2 course module could be everything besides official Angular 2 documentation has recommended the name "AppModule". Fundamentally, route modules are utilized to make a straightforward application however numerous applications utilize the element modules. Each Angular 2 module, regardless of whether it is route or feature, is a class with a decorator called @NgModule, and it’s demonstrated as follows *Figure 13.*



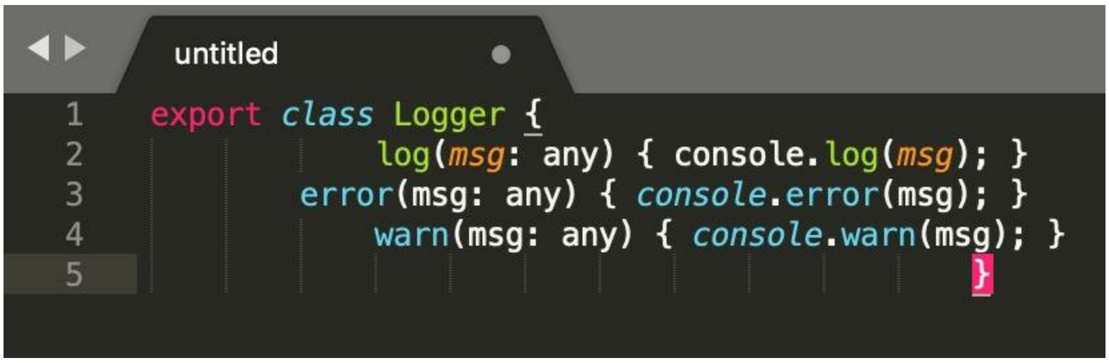
```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';


            @NgModule({
                imports: [ BrowserModule ],
                    declarations: [ AppComponent ],
                        bootstrap: [ AppComponent ]
                                                })

        export class AppModule { }
```

Figure 13 - Listing 3, Angular 2 route module (source: own)

Overhead code in listing 3 diagrams the root module with its properties. Imports, providers, declarations, exports and bootstrap are the significant properties of a root module. (Angular, 2019)

3.4.4 Services

A Service in an Angular 2 application is additionally a class that doesn't own anything to do with itself. However, Angular 2 part includes a few things from services. Services in Angular 2 are injection with dependency injection that alludes to the highlights such logging, information incorporation, application setup and so on. Listing 2 represents the model code for Angular 2 services, see *Figure 14*. (v2, 2019)



```
export class Logger {
            log(msg: any) { console.log(msg); }
        error(msg: any) { console.error(msg); }
            warn(msg: any) { console.warn(msg); }
                                                }
```

Figure 14 - Listing 4, login services to the web browser in Angular 2. (source: own)

3.4.5 Templates and Forms

Angular 2 templates are much like the HTML documents, however, they really a type of HTML that aides Angular 2 when rendering the segment to its user interface segment. Angular 2 formats utilizes all the HTML based syntaxes uses and components, for example, <h1> <p> <li> and so forth, yet in the event that offers some exceptional highlights, for example, interpolation and expression. (Angular, 2019)

3.4.6 Dependency Injection

Dependency in basic terms is really an administrations and reliance injection that is a conventional method for giving a support of the Angular 2 segment, so as to make another occasion of class. A dependency injection is ended up with Angular 2 and they are utilized generally.

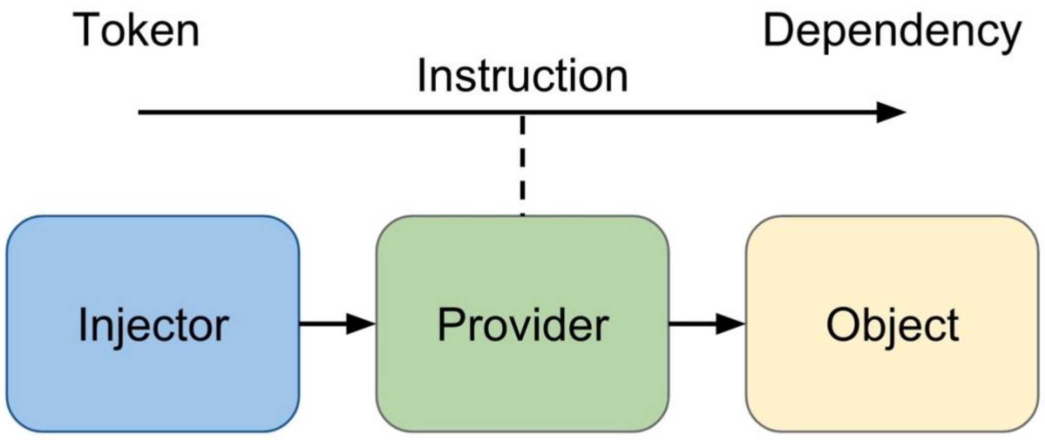


Figure 15 - Components in Dependency Injection (source: github)

Figure 15 represents the dependency Injection segments in Angular 2. A dependency injection in Angular 2 essentially gathers three specific things: an injector, provider and dependency.

Those are: An injector: Injector uncovers the APIs so as to make dependency cases, a supplier: Provider gets aid from the injector to make reliance examples, a dependency: An item is made with the conditions. (v2, 2019)

3.4.7 Data Binding and Meta Data

Data binding is the procedure of synchronization of information between various parts. In Angular 2 application, data binding does the coordination among the part and the formats. Angular 2 applications utilize 2-way data binding that updates the information in the view and controller of the application at the same time.

Favourable circumstances of data binding in Angular 2 application are featured beneath:

- Data binding has made the client sources of info quick. Prior, client's responses were pushed into HTML.
- Data binding gives the productivity in an Angular 2 application for client input as it fills in as an imparting bridge among layouts and segments.

Nevertheless, 2-way data binding systems makes couple vulnerability in web applications, those are stated as follows:

- The adjustment in value in data binding alters the occasion.
- At the point when a few directives are utilized many times, there is the opportunity of break in the applied data binding.

Likewise, another best significant basic design tool utilized in the Angular 2 application is meta-data. Meta-data is in charge of the preparing class into Angular 2 applications. A class will stay a class in Angular 2 application and it never acts as part. Meta-data is expected to process the class into parts in Angular 2 application.

Points of interest of utilizing Meta data in Angular 2 application is featured underneath:

- Meta-data gives the reasonable dependency injection style in Angular 2 application.
- Make-data causes developers to feel the applications as a type of segment tree and Angular 2 style.
- Meta data in Angular 2 is future arranged as it is by all accounts concentrated on upgrading of frameworks. (v2, 2019)

Similarly, Angular 2 Meta-data likewise offers couple disadvantages:

- A manual connecting of the parts can lead the clients denied of getting out majority of the user experience.

3.5 React

The story of React (ReactJS, React.js) was created by Jordan Walke, who was a software engineer at Facebook. At that point in 2013 React was open-sourced and the last release was made in 2019 version 16.10. React centres around making the best interactive UIs in contrast with its challengers AngularJs, and Angular 2. The greatest changes came with React over other existing frameworks and libraries are the utilization of the virtual DOM. Truly, React isn't simply viewed as an ordinary front end library, yet it has turned into a worldview in the realm of front-end web development. (o7planning, 2019)

React isn’t similar like the huge frameworks, for example Angular 2. React is established upon the principles of UNIX, such as, consolidating the basic things to get the most out of it. React enables developers to write the definitive JavaScript segments, which are very progressively better than the traditional way wrapping up the HTML code layouts into JavaScript.

Essentially, MVC configuration examples are viewed as the most basic structure design for UI configurations in front-end web advancement. MVC patterns based on three parts: the model, view and controller as you can see in *Figure 3*. Most of the front-end frameworks centre around these three sections. In any case, React just spotlights on the DOM, since it is the acceptable portrayal of the View. Truly, React gives the view part "V" in the MVC configuration design.

In addition, React is completely an alternate idea in the web application improvement. React has adopted one of kind strategies in taking care of the issues looked by huge scale UIs. This is in actuality made to comprehend the enormous scale UI issues of Facebook and Instagram. Alongside taking care of the issues in UI, React accompanied making the development procedure simpler for developers with changed aptitude levels. (Sengupta, et al., 2016)

Principal points of utilizing React are featured beneath:

- Parsing HTML into JavaScript is a standard, although React is JavaScript driven for example, it enables the developer to write JavaScript codes to produce HTML. This component is very exceptional and loved by larger part of the web developers. As, JavaScript innovations are running everywhere throughout the web applications from front-end to back-end. In this way, modifying with the React will be a smart thought.
- Though, React applications are created with the settlement of the components. React application could move up the style, view and state in one part, which is clearly superior to the moving up with segments and directives.
- Bugs fixing in React is quick in contrast with the enormous frameworks like Angular 2.
- Despite the fact that it requires some time to build up the React application, however later it will spare much additional time with regards to upkeep.
- React is created with the way of thinking, adapt once, write anyplace. In terms of React, it doesn’t matter about the technology stacks. Because, React segments can be rendered utilizing node.js into the server. Additionally, it enables the developer to include new highlights in it without the utilization of any prior existing codes.

We will discuss more about React features starting from section 3.5.1.

3.5.1 JSX

JSX is a HTML-like language structure expansion features presented by React. JSX is utilized in the React to portray the UI segment of the application. JSX may look something like the mark-up language like HTML or XML yet it is completely JavaScript driven and completely useful highlights used to make React application. Be that as it may, it isn't obligatory to utilize JXS in any React application, however the application outfitted with JSX gives superior level of execution, quick blunder dealing with and simple deployment.

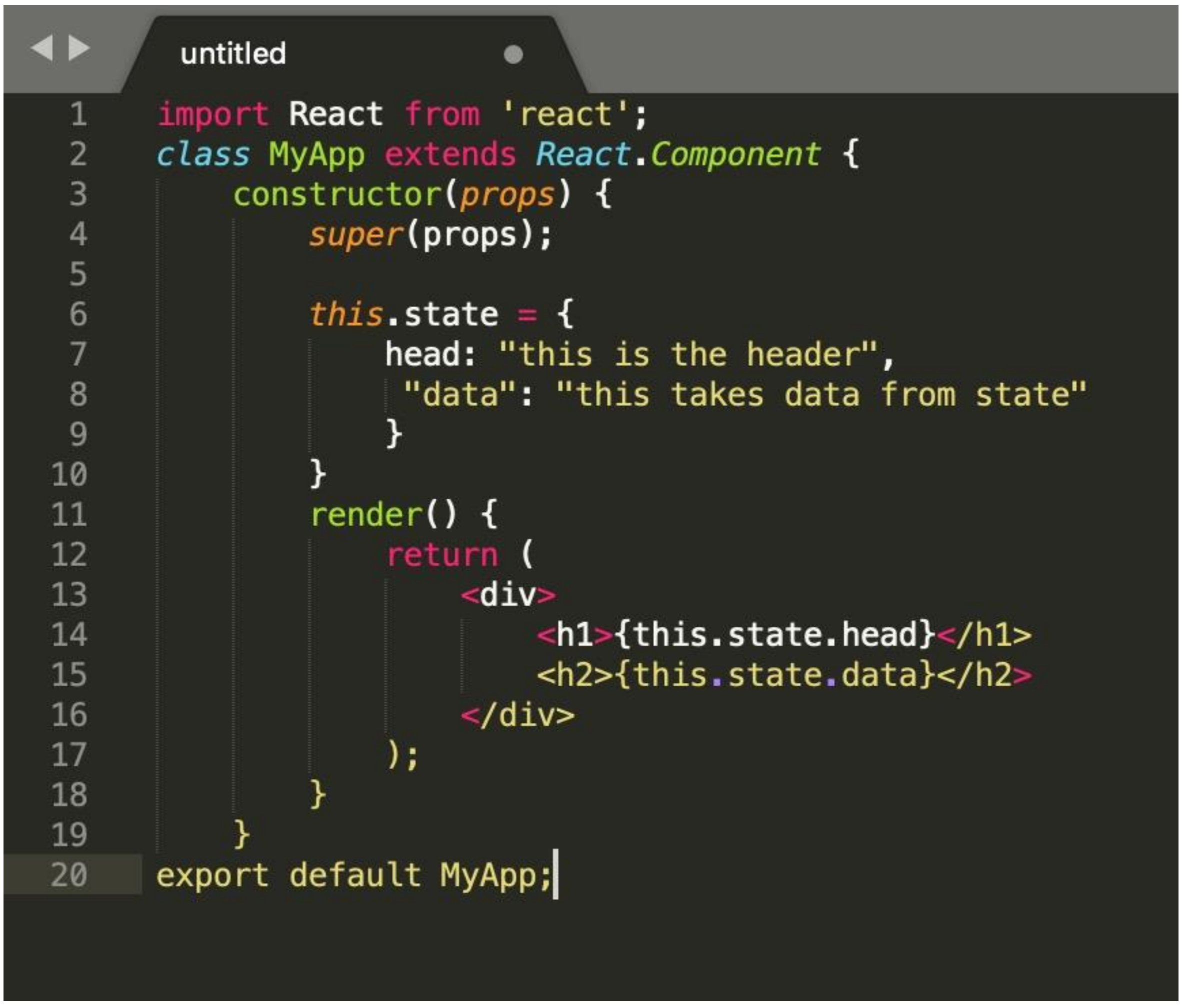
In another words, to benefit fully from React, code and HTML are written in a new language called Java Serialization to XML (JSX). The JSX is a JavaScript syntax extension that looks similar to XML. It is used to build UI components in ReactJS. It’s very likely to HTML with some subtle differences. JSX extends JavaScript in such a way that it can be easily build ReactJS components with the same understanding as building HTML pages. It’s commonly mixed with JavaScript code because ReactJS thinks about UI in a different way. In other meaning developer can embed HTML and CSS within the JavaScript code. This allows for rapid development of a single feature of a single component in a single file. The JSX code is “compiled” into normal JavaScript to enhance performance while running in the browser. (Craig Klementowski, 2017)

JXS contains precisely the comparable components as HTML, for example, <h1>, <p>, <li> and so forth, yet these elements should be settled inside one <div> component in JXS, which later should be put into a component. Actually, all the JXS components have to be included within the React segment pursued by rendering in record Object Model or exporting it into another part. (React.js.org, 2019)

3.5.2 Components of React, Props and State

React components are the structure obstruct in React application advancement, particularly when looking at refreshing the UI. When all is said in done, React part is viewed as like as the JavaScript functions. React grants to make the parts with a way called React.createClass ().

"Props" are the communication channel highlights utilized in React segment. It goes about as an extension between the parent and child parts see *Figure 16*.



```
import React from 'react';
class MyApp extends React.Component {
    constructor(props) {
        super(props);

        this.state = {
            head: "this is the header",
             "data": "this takes data from state"
            }
        }
        render() {
            return (
                <div>
                    <h1>{this.state.head}</h1>
                    <h2>{this.state.data}</h2>
                </div>
            );
        }
    }
export default MyApp;
```

Figure 16 - Listing 5, Representation of components, state and props in React. (source: own)

Also, the "State" is known as the core of the React part. The "State" is main element in React part that makes the segment progressively intuitive and dynamic. The "State" enables the part to restore an identical output for the provided information.

3.5.3 Virtual DOM

DOM manipulation in modern and dynamic web is viewed as a groundwork highlight. Be that as it may, it is quite slow in correlation with other JavaScript tasks. The purpose for the sluggishness of the DOM control is a direct result of its customary updates created by the cutting-edge JavaScript frameworks. JavaScript frameworks should update just specific item in DOM, yet things are not occurring in that manner. A framework really reconstructs and afterward updating the whole item in DOM regardless of whether the updates is required for just specific item. Along these lines of updating from JavaScript structures are letting down the execution of each web applications. Thinking about this specific issue in the DOM control, React presented the idea of Virtual DOM.

Virtual DOM is totally founded on the DOM object and designed as the lightweight duplicate of DOM. The DOM items are mindful to show the data into the UI part of the application, however virtual DOM Object has nothing to do with the UI, yet it provides unique highlights of modifying and updating the items in the back of user interface. (bitsofco.de, 2019)

While browser's DOM is known more slower than JavaScript, React utilizes its very own Virtual DOM, to play out all the interface handling activities and at exactly that point applies the progressions to the browser DOM, and it's demonstrated as follows *Figure 17.*

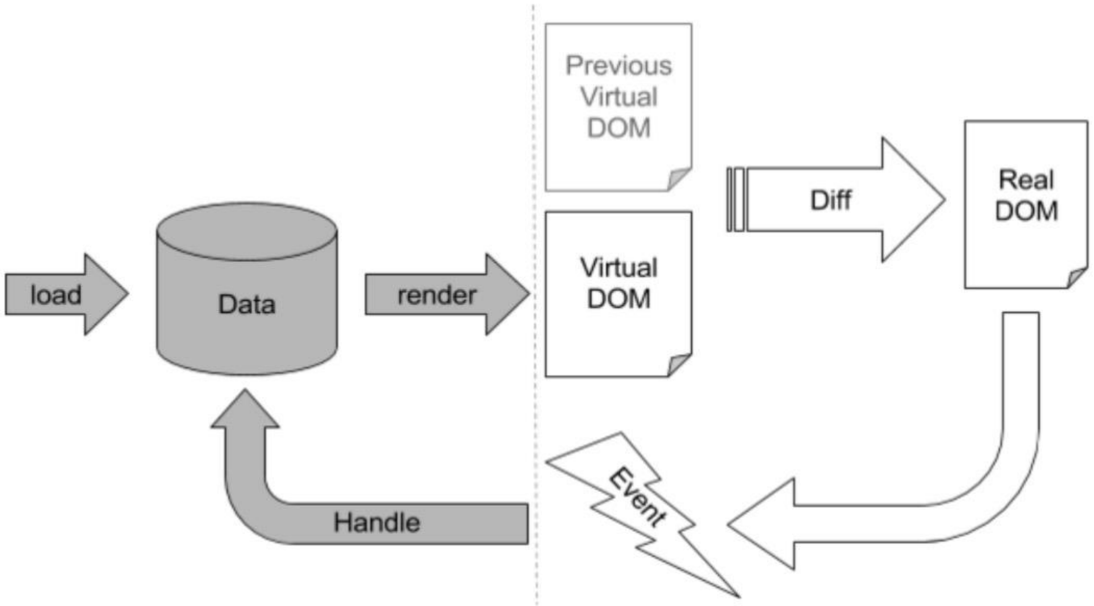


Figure 17 - Virtual DOM workflow visual representations (source: dev.afas.nl)

At the time rendering a JSX component, each virtual DOM receives updated. This doesn't look intriguing, yet with regards to the effectiveness and performance, virtual DOM is amazing comparing to the ordinary DOM. (bitsofco.de, 2019)

3.5.4 Templates and Forms

React provides a straightforward working layout that doesn't need a runtime situation. Syntaxes in React are understandable as HTML, these are supported by whole programs and IDEs (Integrated development environment). while examining the codes in a program, the assessment in HTML and the coding doesn't look very different as a result of the decisive coding. The formats created with React are steady in every one of the stages, for example, Typescript, ES6, and CommonsJs.

React templates perform contrastingly while contrasting and the Angular 2 layouts. In the first place, format containing the *.rt document should be aggregated into the JavaScript record. The record consequently incorporated returns a capacity and later provides a Virtual DOM dependent on the first DOM and client segments.

Essentially, React functions as an amazing UI library, since it centres around the view part of the application which is a UI segment. A view gets refreshed immediately, when there is an adjustment in the state. This method of updating the view in React settles on it a decent decision to conquer the structure approval blunder, because it effectively illustrates the errors in the structure. Be that as it may, it is quite difficult to state that structures made in React are completely secured. Though extra safety effort should be applied to the structure, in light of the fact that a programmer or a hacker could without much of a stretch bypass the React codes by basically permitting the regular variables directly into the server. (React.js.org, 2019)

3.5.5 Flux Architecture

Flux is an architecture utilized in React application. It isn't like the customary libraries or frameworks; however, it gives a well round of highlights to the unidirectional data flow in the React application. Besides, Facebook has supported a library named dispatcher for the Flux architecture. Prototypical Flux design contains the dispatcher library, so as to facilitate the event gaze by building the event framework alongside the assistance of Node 'Js eventEmmitter module. (Zone, 2019)

Fundamentally, Flux design is comprised of four diverse individual segments: Actions, Dispatcher, Store, and Component (Views).

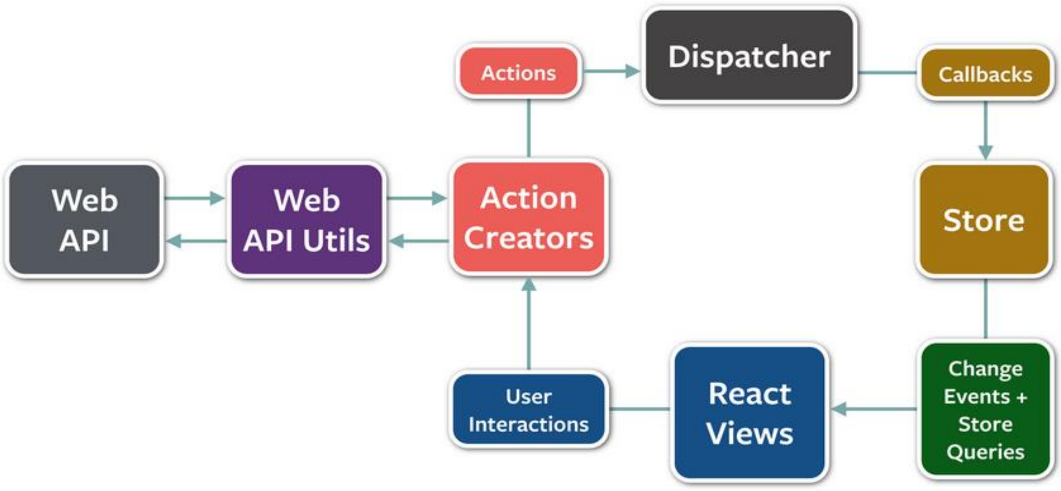


Figure 18 - Unidirectional Data flow in Flux Architecture (github)

Figure 18 outlines the data stream in the middle of the individual parts of the Flux system. The Action creators are helper methods, collected into a library, that creates an action from method parameters, assign it a type and provide it to the Dispatcher as it supports to pass data into the Dispatcher effectively. So, every action is sent to all stores via the callbacks the stores register with the dispatcher. After stores update themselves in response to an action, they emit a change event. Special views called controlled-views, listen for change events, retrieve the new data from the stores and provide the new data to the entire tree of their child views. (Zone, 2019)

Moreover, Dispatcher gets the actions and then it advances to the callback function. The information consequently will be sent, thus it stored in a holder for the application "Logic" and "State". At long last, the Data State from the

"Store" will at that point passed it to the user interface segment over the React "Props" strategy.

The data stream in the Flux design is unidirectional and expressed key to the pattern. There three unique parts, State, Dispatcher and Views, are absolute with well-characterized Output and Input. Be that as it may, Action part in Flux are expressed as an object with fresh data and recognizing property type.

The structure created in the Flux design provides an unmistakable thought regarding its motivation for a particular progression of data in the application. This provides a feeling that it depends on the reactive useful programming. In reactive functional programming, data streams just one way, for example application state is just sustained in the store. (Craig Klementowski, 2017)

3.5.6 Data binding in React

Reacts provides a one-way data binding, however it likewise bolsters the bidirectional data stream by the assistance of some other plugins. The one-way data stream gives the ground to the transmission of information from parent to child. The "Props" properties in React are in charge to transfer data from parent to child. The adjustment in "Props" modifies a React part which prompts the adjustment in Component tree and its properties. (React.js.org, 2019)

4 Practical Part

4.1 Real-World comparison of front-end frameworks with benchmarks

In the course of the most recent few years, we have seen a blast of front-end frameworks. Every last one of them is more than fit for making extraordinary web applications. There are many thoughts and master discussions on this topic, and everybody has their decision. Be that as it may, in this thesis, Author will adopt progressively functional strategy to look at these two frameworks by making a straightforward application utilizing every one of them.

Project portrayal. Application makes a programming interface call, to obtain the movie having (Crazy, Stupid) in their title, in 2018, then shows them in table format. The following are the connections to the application facilitated on GitHub.

Angular:	*Link at* (Abhay07, 2019)
React:	*Link at* (Abhay07, 2019)

Comparison metrices

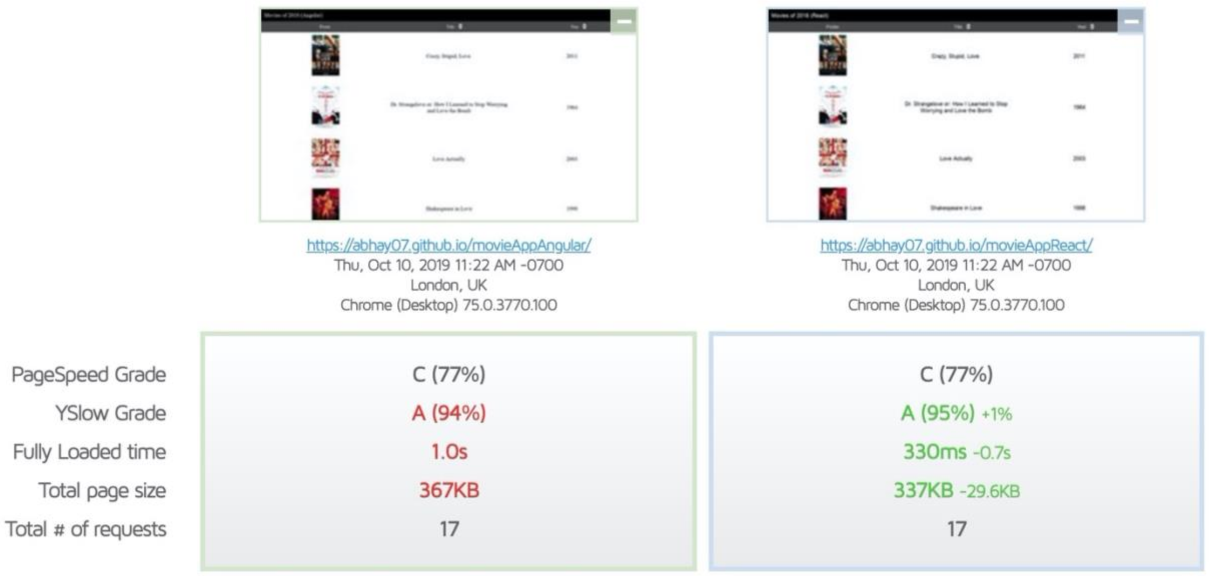
I will compare these applications on beneath criteria's using (*Lighthouse Audit* and *GTmetrix*)

- **Performance** - To what extent does this App take to display content and will be usable? Thus, performance of a site is dictated by numerous components, like Author is going to illustrate a couple of them on my "For what reason is my page slow?" page. PageSpeed and YSlow centre around the front-end execution of your site, along with components that are to a great extent in your control, similar to pictures and main website architecture. PageSpeed and YSlow scores reveal to you that how good your front-end is improved for loading time. (GTmetrix, 2019)
- **JS bundle size** - JavaScript pack size is the main asset that contrasts for the application in every framework. Pictures, CSS, and different parts are same.

- **First Contentful paint -** First Contentful Paint, also known as FCP, evaluates the time from route to the time, while the program renders the main piece of substance from the DOM, and for more information on FCP links at (GTmetrix, 2019) and (Developers, 2019)
- **Line of codes -** What number of lines of code did the creator or developer have to make the movie website, while both sites have identical contend and same components. The main folder, I will measure **src/** in each application. Here are those folders for Angular 2 (Abhay07, 2019) and for React (Abhay07, 2019)

4.1.1 Performance

Front-end is characterized as your page and every one of its parts, as rendered through browser. It covers those components: HTML, CSS, JavaScript, Media (picture, video and so on. Those are the components which browser utilizes to develop and render the page. GTmetrix investigates your webpage to check whether your site's front-end is following prescribed procedures, so you get PageSpeed and YSlow evaluations dependent on your page's equivalent to rule sets. The best grade implies that your page is advanced for a program to render as quick as could be allowed, see results in *Figure 19.*



	https://abhay07.github.io/movieAppAngular/ Thu, Oct 10, 2019 11:22 AM -0700 London, UK Chrome (Desktop) 75.0.3770.100	https://abhay07.github.io/movieAppReact/ Thu, Oct 10, 2019 11:22 AM -0700 London, UK Chrome (Desktop) 75.0.3770.100
PageSpeed Grade	C (77%)	C (77%)
YSlow Grade	A (94%)	A (95%) +1%
Fully Loaded time	1.0s	330ms -0.7s
Total page size	367KB	337KB -29.6KB
Total # of requests	17	17

Figure 19 - Overall performance compare report (source: Gmetrix)

4.1.2 JS bundle size

Thus, the name proposes, this shows transferred bundle size of a record or file. The complete consolidated record size of every one of your requests gather up the Total Page Size. Smaller the sum is, the easier your page is to download and along these it's quicker. As we can see in *Figure 20* Angular's js pack size is slightly higher comparing to React, **357KB** to **337KB**

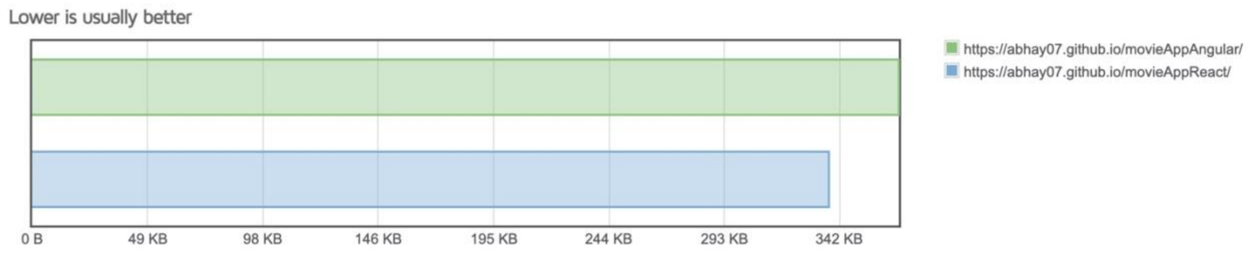


Figure 20 - JS bundle size (source: Gmetrix)

4.1.3 First Contentful Paint

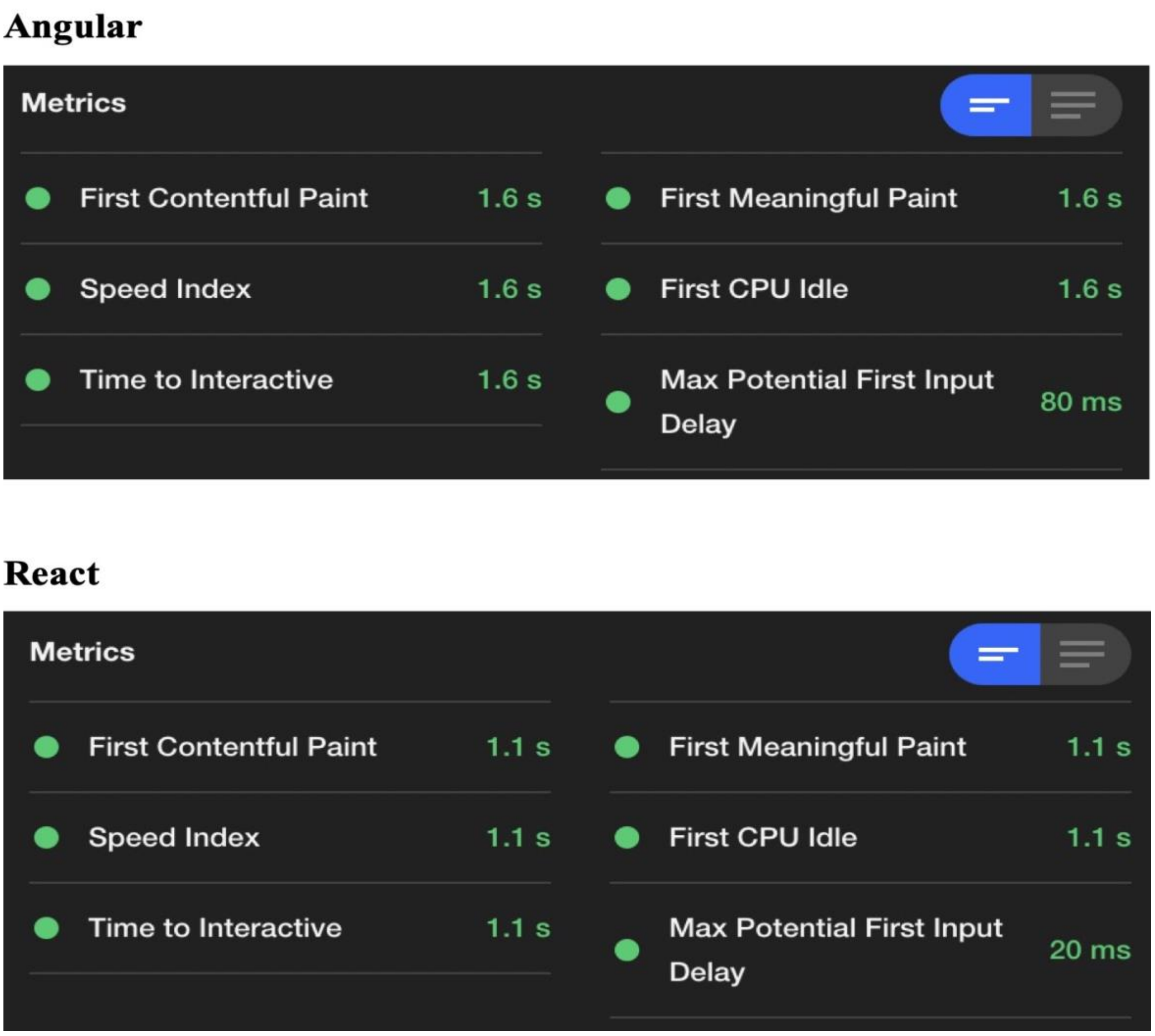
Author tried to compare both frameworks with Lighthouse audit which comes with google chrome using my personal laptop MacBook Pro 13-inch, 2018.

Audit settings:

- **Device** – Mobile
- **CPU and Network throttling** – Simulated Fast 3G, 4x CPU Slowdown
- **Clear storage** – checked

The performance grade from the below measurements:

- First Contentful Paint
- First Meaningful Paint
- Speed Index
- First CPU Idle
- Time to Interactive
- Max Potential First Input Delay or Estimated Input Latency



Angular

Metrics

●	First Contentful Paint	1.6 s	●	First Meaningful Paint	1.6 s
●	Speed Index	1.6 s	●	First CPU Idle	1.6 s
●	Time to Interactive	1.6 s	●	Max Potential First Input Delay	80 ms

React

Metrics

●	First Contentful Paint	1.1 s	●	First Meaningful Paint	1.1 s
●	Speed Index	1.1 s	●	First CPU Idle	1.1 s
●	Time to Interactive	1.1 s	●	Max Potential First Input Delay	20 ms

Figure 21 - Lighthouse audit (source: own)

As per *Figure 21* React performed superior with 1.1 s as First Contentful Paint than Angular’s 1.6 s

More details regarding Lighthouse scoring guide check at (Developers, 2019) and (Web, 2019)

4.1.4 Line of codes

That determines, how concise provided library/framework/language is. What number of lines of code does developer have to implement nearly the equivalent application? After adding up all the lines of code in **src/** folder, Angular came with **502**, while React had only **350** to create almost the same movie application.

4.2 Stefan Krause’s benchmark tool

Every one of the benchmarks shown beneath originated from Stefan Krause's practical examination. Author has chosen to utilize these with the goal for everybody to have an online reference for discussion. Thus, it will be available in GitHub. Results were last updated on 27.09.2019, and all the benchmarks were performed on a Razer Blade 15 Advanced, with the following specifications: (i7-8750H, 32 GB RAM, Ubuntu 19.04 (Linux 5.0.0-29, mitigations=off), Chrome 77.0.3865.90 (64-bit)) (GitHub, 2019)

4.2.1 Head-to-head performance comparison

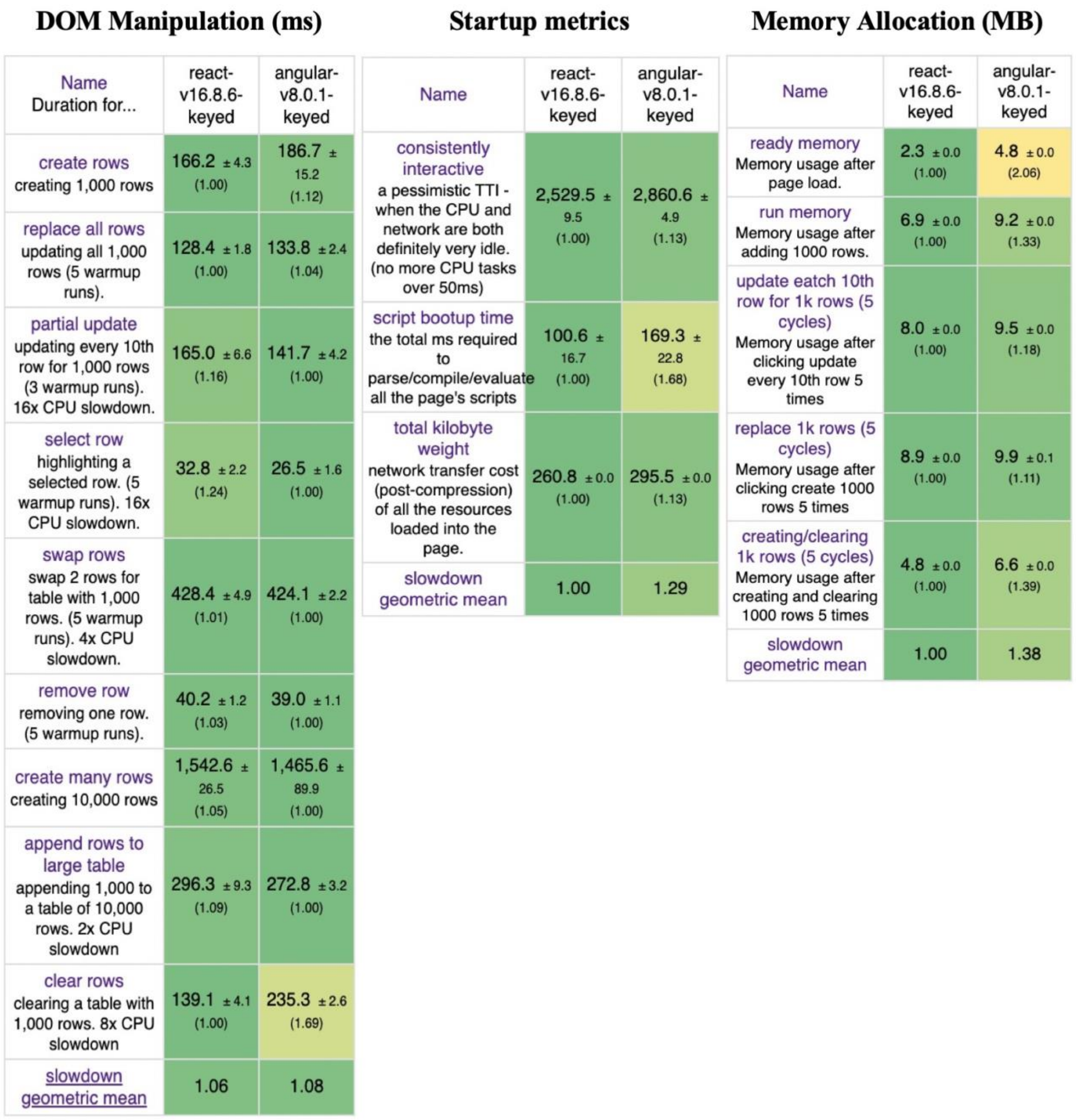
Since we already have a complete overview of both frameworks. They will be analysed by those three factors:

- **DOM manipulation:** it shows us to figure out which frameworks run better for profoundly dynamic apps, which demands a ton of collaboration with the DOM.
- **Start-up time:** it will provide us a thought of which frameworks are progressively appropriate when you need a quicker introductory load time, valuable for apps which demands higher speed.
- **Memory allocation:** it uncovers which frameworks superior, in terms of operation with the memory. For instance, performing mass activities, for example, writing/reading a large number of files from a database.

4.2.2 Performing the benchmark, Start-up time and Memory allocation

Each color you will find in a picture *Figure 22* has a particular importance. Such as, A greener color implies that given framework runs superior to another.

When it gets yellowing implies shown framework runs worse. Every one of the tests are demonstrated between the keyed versions of the frameworks, and all units are shown in milliseconds divided by 95% confidence interval (Slowdown = Duration / Fastest).



DOM Manipulation (ms)

Name Duration for...	react-v16.8.6-keyed	angular-v8.0.1-keyed
create rows creating 1,000 rows	166.2 ± 4.3 (1.00)	186.7 ± 15.2 (1.12)
replace all rows updating all 1,000 rows (5 warmup runs).	128.4 ± 1.8 (1.00)	133.8 ± 2.4 (1.04)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	165.0 ± 6.6 (1.16)	141.7 ± 4.2 (1.00)
select row highlighting a selected row. (5 warmup runs). 16x CPU slowdown.	32.8 ± 2.2 (1.24)	26.5 ± 1.6 (1.00)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	428.4 ± 4.9 (1.01)	424.1 ± 2.2 (1.00)
remove row removing one row. (5 warmup runs).	40.2 ± 1.2 (1.03)	39.0 ± 1.1 (1.00)
create many rows creating 10,000 rows	1,542.6 ± 26.5 (1.05)	1,465.6 ± 89.9 (1.00)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown	296.3 ± 9.3 (1.09)	272.8 ± 3.2 (1.00)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown	139.1 ± 4.1 (1.00)	235.3 ± 2.6 (1.69)
slowdown geometric mean	1.06	1.08

Startup metrics

Name	react-v16.8.6-keyed	angular-v8.0.1-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,529.5 ± 9.5 (1.00)	2,860.6 ± 4.9 (1.13)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	100.6 ± 16.7 (1.00)	169.3 ± 22.8 (1.68)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	260.8 ± 0.0 (1.00)	295.5 ± 0.0 (1.13)
slowdown geometric mean	1.00	1.29

Memory Allocation (MB)

Name	react-v16.8.6-keyed	angular-v8.0.1-keyed
ready memory Memory usage after page load.	2.3 ± 0.0 (1.00)	4.8 ± 0.0 (2.06)
run memory Memory usage after adding 1000 rows.	6.9 ± 0.0 (1.00)	9.2 ± 0.0 (1.33)
update eatch 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	8.0 ± 0.0 (1.00)	9.5 ± 0.0 (1.18)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	8.9 ± 0.0 (1.00)	9.9 ± 0.1 (1.11)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	4.8 ± 0.0 (1.00)	6.6 ± 0.0 (1.39)
slowdown geometric mean	1.00	1.38

Figure 22 - Stefan Krause's benchmark (source: github)

5 Results and Discussion

A complete study of those 2 frameworks React and Angular 2 were made through the thesis thinking about various methodologies. Two significant steps were used to discover the objectives of the study. Right of the bat, author made complete literature review of both Angular 2 and React, including advantages and disadvantages of the frameworks. Furthermore, to develop a comparison test which evaluates the efficiency of React in development of UI.

The two stages offered strong highlights and viewpoint to the user interface application. Nonetheless, the objective of the investigation was to make various approaches. A specific arrangement of comparable rules was presented, and findings were talked about to do productive comparison among the frameworks.

5.1 Which of the frameworks proposed better user interface?

It's a significant question which can be asked each time concerning UI advancements. Thus, the responses to the above question is entirely broken down by the result of this proposition. Six significant central points are put into comparison between these two frameworks.

5.1.1 Performance

React shows better results in terms of performance in comparison with Angular 2. When refreshing and synchronizing the segments in the apps, React utilized virtual DOM in the application, though when the application design was complex and that brought about the better outcome. While with Angular 2 user interface the standard DOM was included for the refreshing all the DOM parts, not refreshing the essential one, which came about the slowness in the application.

5.1.2 Templating

While looking at the templating among 2 frameworks, Angular 2 proposed predominance, as the user interface development doesn't really require the JavaScript driven templating style. Templating is progressively concerned about user interface parts and even a straightforward mark-up provides modesty in templating than the complex JSX. Angular 2 utilizes the HTML mark-up in a basic way, that is obviously superior than composing the monotonous JSX codes. While building up the user interface, that was discovered that the templating structure in Angular 2 was more straightforward than ReactJS.

5.1.3 Structuring data model

At the point while it involves the information modelling comparison between Angular 2 and React, the two advancements provide similarity. Thus, performance of Angular 2 is delicate when it manages (Scopes) and it recommends utilizing huge models won't be allowed. This outcomes in an increasingly testable and straightforward codes on one hand, and then again powers to separate the typical techniques for revamping the back-ups. Be that as it may, React offers freedom of decision in organizing the information model without scaling down the performance. Nonetheless, it's up to totally on the capabilities of designers.

5.1.4 Code reshuffling

When looking at the code reshuffling between ReactJS and Angular 2, there are a lot of instant libraries accessible for Angular 2. Nonetheless, React gives the adaptability to reshuffle the code in its own specific manner.

5.1.5 Package management

React offers higher adaptability in package management comparing to Angular 2. React utilizes Require.js and Webpack to improve bundle management. In any case, Angular 2 doesn't enable the codes to run and deploy much flexibility. Thus, this outcomes in weak packaging of codes.

5.1.6 Learnability

In terms of learnability in React is increasingly better comparing Angular 2. However, learning Angular 2 is intriguing toward the start yet the later stage becomes disappointing as a result of its complex lifecycle. Then again, React is special due to the JavaScript driven feature and though a beginner engineer can ace it up to seven days. The straightforward eco-system and solid lifecycle are the extensive highlights, which put React ideal for the developers. Easier learning curve stands for there will constantly a superior possibility of concentrating on the user experience by beating the current issues.

6 Conclusion

The main objective of thesis was to evaluate the efficiency of Javascript frameworks, and summarizing every one of the methodologies applied, the study was at long last finished up as React being progressively better innovation for the advancement of a UI.

In this thesis Author only compared 2 identical simple applications in order to evaluate the efficiency of both frameworks, thus the main idea was to see the real difference how they perform.

The innovations for the improvement of UIs develop quickly like any other relatable part in the web development. The manner in which advances are developing in the front-end structures has been changing the logic and strategies utilized in the improvement procedure. The front-end and UI were intended for the creators. Nevertheless, the utilization of frameworks and libraries in the front-end has made dynamism in the web.

The final proposal of this examination is that it recommends designers to experience a pragmatic way to develop an indistinguishable UI application in the two innovations, and mainly results of this thesis for those developers who are trying to find out which frameworks more preferable in order to build and application mostly focused on user interface.

7 References

8 Bibliography

Facebook/Flux. [Online] [Cited: 10 April 2018.] https://facebook.github.io/flux/docs/in-depth-overview.html#content.

Abhay07. 2019. *Github.* [Online] 2019. https://github.com/Abhay07/movieAppReact/tree/master/src.

—. 2019. *Github.* [Online] 2019. https://github.com/Abhay07/movieAppAngular/tree/master/src.

—. 2019. *Movies of 2018 (Angular).* [Online] 2019. https://abhay07.github.io/movieAppAngular/.

—. 2019. *Movies of 2018 (React).* [Online] 2019. https://abhay07.github.io/movieAppReact/.

Angular. 2019. *Angular.* [Online] 2019. https://ngdocsdev.firebaseapp.com/docs/ts/latest/guide/ngmodule.html.

—. 2019. *Angular.* [Online] 2019. https://angular.io/api/core/Component.

bitsofco.de. 2019. bitsofco.de. *bitsofco.de.* [Online] 2019. https://bitsofco.de/understanding-the-virtual-dom/.

C#Corner. 2019. C#Corner. *C#Corner.* [Online] 2019. https://www.c-sharpcorner.com/article/what-is-typescript/.

colorlib. 2019. colorlib. *colorlib.* [Online] 2019. https://colorlib.com/wp/angular-components/.

Craig Klementowski, Tiffany Reid, Norbert Antunes, Jaiden Choong, Liqiao Huang, Ross Arnold. 2017. TACTICAL APPLICATIONS (TACAPPS) JAVASCRIPT FRAMEWORK INVESTIGATION. [Online] January 2017. [Cited: 10 April 2018.] http://www.bing.com/cr?IG=0F90B90087CE4E3B87BAF6820D0AC46D&CID=026BFF879F64678F27B8F4499ECB66E1&rd=1&h=NStbcaWkplozpb8nWbm1_7XzsnZGmYxoAyhfp9aKXm0&v=1&r=http%3a%2f%2fwww.dtic.mil%2fdtic%2ftr%2ffulltext%2fu2%2f1025789.pdf&p=DevEx,5066.1.

Crockford, Douglas. 2019. JavaScript: The Good Parts. [Online] 2019. https://amontalenti.com/2019/08/10/javascript-the-modern-parts. 9780596517748.

Dev. 2019. Dev. *Dev.* [Online] 2019. [Cited: 10 4, 2019.] https://dev.to/javinpaul/the-2019-web-development-frontend-backend-roadmap-4le2.

Developers, Tools for Web. 2019. First Contentful Paint. *Tools for Web Developers.* [Online] 2019. https://developers.google.com/web/tools/lighthouse/audits/first-contentful-paint.

—. 2019. Lighthouse Scoring Guide. *Tools for Web Developers.* [Online] 2019. https://developers.google.com/web/tools/lighthouse/v3/scoring.

Educba. 2019. Educba. *Educba.* [Online] 2019. https://www.educba.com/angular-vs-react/.

Flanagan, David. 2006. *JavaScript: the definitive guide.* Sepastopol, CA : O'Reilly, 2006. ISBN 978-0-596-10199-2.

freeCodeCamp. 2019. freeCodeCamp. [Online] 2019. https://www.freecodecamp.org/news/a-comparison-between-angular-and-react-and-their-core-languages-9de52f485a76/.

GeeksforGeeks. 2019. GeeksforGeeks. [Online] 2019. https://www.geeksforgeeks.org/mvc-design-pattern/.

GitHub. 2019. GitHub. *GitHub.* [Online] 10 16, 2019. [Cited: 10 16, 2019.] https://krausest.github.io/js-framework-benchmark/current.html.

github.com/facebook/flux. *https://github.com/facebook/flux.* [Online] [Cited: April 11, 2018.] https://github.com/facebook/flux.

GTmetrix. 2019. *GTmetrix.* [Online] 2019. https://gtmetrix.com/blog/waterfall-chart-updates-sorting-filtering-by-type-keyword-and-visual-updates/.

—. 2019. GTmetrix. *GTmetrix.* [Online] 10 10, 2019. [Cited: 10 10, 2019.] https://gtmetrix.com/why-is-my-page-slow.html.

Guide, Digital. 2019. Digital Guide. *Digital Guide.* [Online] 2019. https://www.ionos.com/digitalguide/websites/web-development/popular-javascript-frameworks-and-libraries/.

M, Vipul A. and Sonpatki, Prathamesh. 2016. *ReactJS by Example - Building Modern Web Applications with React.* Birmingham, UK : Packt Publishing, 2016. ISBN 978-1-78528-964-4.

o7planning. 2019. o7planning. *o7planning.* [Online] 2019. https://o7planning.org/en/12077/introducing-angularjs-and-angular.

Pieces, Bits and. Bits and Pieces. *Bits and Pieces.* [Online] [Cited: 10 10, 2019.] https://blog.bitsrc.io/benchmarking-angular-react-and-vue-for-small-web-applications-e3cbd62d6565.

React.js.org. 2019. *React.js.org.* [Online] 2019. https://reactjs.org/docs/getting-started.html.

report, State of JS. State of JS report. [Online] https://2018.stateofjs.com/front-end-frameworks/react/.

Sambra. 2019. Sambra. [Online] 2019. https://sombrainc.com/web-application-architecture.

Sengupta, Doel, Singhal, Manu and Corvalan, Danillo. 2016. *Getting Started with React.* Livery Place, 35 Livery Street, B3 2PB, UK : Packt Publishing, 2016. ISBN 978-1-78355-057-9.

SitePoint. The Basics of the Shadow DOM. [Online] [Cited: 11 April 2018.] https://www.sitepoint.com/the-basics-of-the-shadow-dom/.

stackoverflow. 2019. *stackoverflow.* [Online] 2019. https://stackoverflow.com/questions/35138080/relationship-between-typescript-and-es6.

stuff, net. 2019. net stuff. [Online] 2019. http://www.dotnet-stuff.com/tutorials/angular-js/introduction-of-angularjs.

TechMagic. 2019. TechMagic. *TechMagic.* [Online] 2019. https://blog.techmagic.co/angular-2-vs-react-what-to-chose-in-2017/.

Tecla. 2019. Tecla. [Online] 2019. https://www.tecla.io/blog/2019-stats-on-top-js-frameworks-react-angular-and-vue/.

v2, Angular. 2019. *Angular v2.* [Online] 2019. https://v2.angular.io/docs/ts/latest/cookbook/dependency-injection.html.

W3Schools. https://www.w3schools.com/. [Online] [Cited: 11 April 2018.] https://www.w3schools.com/code/tryit.asp?filename=FQ9T4MH309XV.

Web. 2019. Lighthouse. *Web.* [Online] 2019. https://developers.google.com/web/tools/lighthouse/#devtools.

WEBCOMPONENTS.ORG. https://www.webcomponents.org/. *WEBCOMPONENTS.ORG*. [Online] [Cited: 11 April 2018.] https://www.webcomponents.org/.

WebMaster. 2019. Client-Side JavaScript Examples. [Online] 2019. https://docstore.mik.ua/orelly/web/webnut/ch21_02.html.

Zone, Web Dev. 2019. DZone. *DZone.* [Online] 2019. https://dzone.com/articles/a-detailed-study-of-flux-the-reactjs-application-a.