HOCHSCHULE
DER MEDIEN

Bachelorarbeit

im Studiengang Mobile Medien

Comparison of Three Popular JavaScript Frameworks: Angular, React and Vue.js

vorgelegt von

Marlene C. Hasslinger

an der Hochschule der Medien Stuttgart

am 12. August 2019

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

Erstprüfer:	Prof. Dr. Joachim Charzinski
Zweitprüfer:	Prof. Walter Kriha

Ehrenwörtliche Erklärung

Hiermit versichere ich, Marlene Hasslinger, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit mit dem Titel: "Comparison of Three Popular JavaScript Frameworks: Angular, React and Vue.js" selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden. Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 24 Abs. 2 Bachelor-SPO (7 Semester), § 23 Abs. 2 Master-SPO (3 Semester) bzw. § 19 Abs. 2 Master-SPO (4 Semester und berufsbegleitend) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Datum: _____ Unterschrift: _____

Kurzfassung

JavaScript hat sich in den letzten Jahren zur meistgenutzten Programmiersprache entwickelt und die Anzahl der Software-Frameworks rund um JavaScript ist gewachsen. Das Ziel dieser Arbeit ist es, eine Hilfestellung für Entscheidungsfragen zwischen JavaScript-Frameworks in der Frontend-Entwicklung zu leisten. Dazu werden drei bekannte JavaScript-Frameworks, Angular, React und Vue.js, miteinander verglichen. Hierfür werden zunächst allgemeine Informationen über JavaScript, Software-Frameworks und Webanwendungen vermittelt, um ein grundlegendes Verständnis für das Thema zu schaffen. Anschließend werden sechs Kriterien definiert, um die Frameworks individuell zu beurteilen und später untereinander zu vergleichen. Die Informationen und Daten, die zur Bewertung der Frameworks herangezogen werden, entstammen einer Literaturrecherche und Performance-Messungen. Das Ergebnis der Untersuchung zeigt, dass keines der Frameworks in allen Kriterien als Bestes abschneidet und die Entscheidung für ein Framework ausschlaggebend von den individuellen Anforderungen eines Projekts abhängt. Lesenswert ist die Arbeit für alle, die sich für Frontend-Entwicklung interessieren und bereits Erfahrung mit HyperText Markup Language (HTML), Cascading Style Sheets (CSS) und JavaScript haben. Insbesondere Leser, die vor der Entscheidung stehen welches der oben angeführten JavaScript-Frameworks erlernt oder für ein Projekt benutzt werden soll, können von dieser Arbeit profitieren.

Abstract

In the past years, JavaScript has become the most used programming language and the number of software frameworks around it has grown. The goal of this paper is to support decisions between JavaScript frameworks that are used in front-end development. Therefore, Angular, React and Vue.js, three popular JavaScript frameworks are compared. First, background information about JavaScript, software frameworks and web applications is provided to contribute to a fundamental understanding of the topic. Then, six criteria are defined to evaluate each framework and to compare them with each other later on. The information and data used to evaluate the frameworks are collected from a literature review and performance measurements. The result of the investigation shows that none of the frameworks scores best in all criteria and that the selection of a framework depends crucially on the individual requirements of a project. This paper is interesting for anyone who is interested in front-end development and has some experience with HyperText Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript. Especially readers who are deciding which of the above-mentioned JavaScript frameworks to learn or to use for a project can benefit from this work.

Contents

List of Figures

List of Tables

List of Code Examples

List of Abbreviations

AJAX	Asynchronous JavaScript and XML
API	Application Program Interface
CLI	Command-Line Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
ES5	ECMAScript 5
ES6	ECMAScript 6
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JSX	JavaScript XML
MPA	Multi-Page Application
NgModule	Angular Module
NgRx	A State Management Framework for Angular Applications
NPM	Node Package Manager
Redux	A State Management Framework
RxJS	Reactive Extensions for JavaScript
SPA	Single-Page Application
UI	User Interface
URL	Uniform Resource Locator

UX User Experience

Vuex A State Management Framework for Vue Applications

XML Extensible Markup Language

1 Introduction

This chapter introduces the background of the thesis and summarizes its objective and methods. Also, an overview of the structure is given.

1.1 Background

Over the years, web pages have evolved from displaying static HTML pages to offering a complex and interactive User Experience (UX) [1]. JavaScript is a scripting language that handles interactivity and dynamic changes within the front-end of web pages. A front-end is the visible part of an application that displays data to a user in a browser. A survey by W3Techs states that 95.2% of the ten million most popular web pages use JavaScript [2]. Also, according to the Stack Overflow[1] annual developer survey in 2019, JavaScript has been the most popular technology among all programming languages for 7 years in a row. The number of technologies, including tools, frameworks and libraries, around JavaScript is large and new ones are released frequently. John Hannah reports a number of 57 JavaScript frameworks and libraries as of the beginning of 2018 [4]. Consequently, front-end developers have many options to choose from regarding the set of technologies for a project.

Angular, React and Vue.js are among the most popular JavaScript front-end frameworks [5, 4]. All three frameworks offer solutions for common challenges in front-end development. Some of the most important challenges are the synchronization of the data and the User Interface (UI) of a web application [6], event handling and Document Object Model (DOM) manipulation [7, 8].

1 Stack Overflow is a question-and-answer platform for developers [3].

1.2 Objective and Methodology

This paper aims to help make decisions between JavaScript frameworks for the development of web front-ends. Therefore, three of the most popular frameworks, Angular, React and Vue.js, are investigated and compared with each other. Six criteria are selected and defined for the evaluation of the frameworks. The criteria are not prioritized as the individual needs for each use case differ. Consequently, this paper does not present an overall metric with a single framework as the best option, but shows up how the frameworks perform in different aspects to support educated decisions. The data and information used for the comparison results from a literature review and performance measurements.

1.3 Structure

This thesis starts with an overview of relevant articles and papers that have been published about the comparison of JavaScript frameworks. Then, background information is provided to contribute to a fundamental understanding of JavaScript, frameworks and web applications. Chapter 4 defines in detail which methods are used for the following investigation and comparison. Also, the comparison criteria are defined and divided into sub criteria. In the last two chapters, the frameworks are compared with each other and the results are presented and interpreted. Finally, an outlook is given and possible future work arising from this study is discussed.

2 Related Work

First, related work about methods to compare JavaScript frameworks is addressed. Then, papers that deal with the comparison of the particular frameworks Angular, React and Vue.js are addressed. It is worth noting that the number of papers that deal with these topics is small, despite the popularity of JavaScript frameworks and libraries.

Gizas et al. developed a method to compare the JavaScript frameworks ExtJS[1], Dojo[2], jQuery[3], MooTools[4], Prototype[5] and YUI[6] by conducting software quality tests and performance tests [9]. After interviewing four front-end developers about their views on how to choose a JavaScript framework, Graziotin et al. extended this method by adding the comparison aspects adequacy of documentation, community participation and code quantity [10].

The publications that compare Angular, React and Vue.js are limited to a bachelor's dissertation and a number of blog posts. Eric Wohlgethan compared Angular, React and Vue.js in a bachelor's dissertation based on features, technical aspects, learning curves and popularity. His conclusion is that the advantages and disadvantages of each framework depend heavily on the individual needs and circumstances of a project [11]. The company FusionCharts published a blog post that compares Angular, React, Ember.js and Vue.js based on features, popularity, integration support, learning curves and framework documentation. Similar to Wohlgethan, FusionCharts does not suggest a single framework as the best option and also recommends choosing a framework based on the individual needs of a use case [12].

1 See https://www.sencha.com/products/extjs/.
2 See https://dojotoolkit.org.
3 See https://jquery.com.
4 See https://mootools.net.
5 See http://prototypejs.org.
6 See https://yuilibrary.com.

Another blog post, published by Shaumik Daityari, compares Angular, React and Vue.js based on license agreements, popularity, job market, popularity, performance and learning curves. His conclusion matches Wohlgethan's and FusionCharts' conclusion. No single framework is suggested as a general best choice. It is suggested to make a choice based on the use case and the individual requirements of a project [13].

Technologies change frequently, therefore, this paper evaluates recent versions of the three frameworks and compares them based on a combination of the above-mentioned criteria, plus two self-defined criteria. The results can be used to help make an educated decision about which framework is best-suited for the individual needs of a project.

3 Background

This chapter provides background information that contributes to a fundamental understanding of JavaScript, frameworks, libraries and web applications.

3.1 JavaScript

JavaScript is a scripting language that is read and executed by the JavaScript engine of a web browser. Its implementation is based on ECMA-262[1] [14]. Therefore, web browsers need to implement the features specified in this standard [15, Chapter 1, What is JavaScript?]. ECMAScript 5 (ES5), a version of the standard that was released in 2009, is the latest version of the standard that is supported by the six most used browsers, Chrome, Firefox, Edge, Internet Explorer, Safari and Opera [16]. Chrome, Firefox, Edge, Safari and Opera support ECMAScript 6 (ES6), a newer version of the standard. Internet Explorer does not yet fully support ES6 [17]. Two major concepts that were introduced in ES6 are classes and modules[2] [14]. JavaScript compiler, such as Babel[3], can compile newer implementations of the ECMAScript to ES5 for a wider browser compatibility.

JavaScript started as a browser-based language, but through, e.g. Node.js[4], an open source server environment, it can also be used for back-end development [15, Chapter 1, What is JavaScript?]. For the purpose of comparing front-end frameworks, this paper only investigates JavaScript as a client-side language. TypeScript and JSX are two more client-side languages that are important for the understanding of this paper. Both are compiled to JavaScript before they can be executed in a browser.

1 ECMA-262 is a programming language specification for the ECMAScript general-purpose programming language [14]
2 Modules allow a user to export defined code pieces and import them in other files [18].
3 See https://babeljs.io.
4 See https://nodejs.org/en/.

3.1.1 TypeScript

TypeScript is a scripting language developed by Microsoft that extends the ECMAScript specification (see Figure 3.1). Therefore, it is a strict superset of JavaScript and any valid JavaScript code is valid TypeScript. Browsers cannot understand TypeScript. It first has to be compiled to JavaScript before it can be run by a browser [19]. A key feature of TypeScript is static typing. It enables developers to define data types for variables. The compiler can then verify that a variable is used in a valid manner and prevent run-time errors. Angular applications use TypeScript as a primary language [20, Chapter 4, Overview].

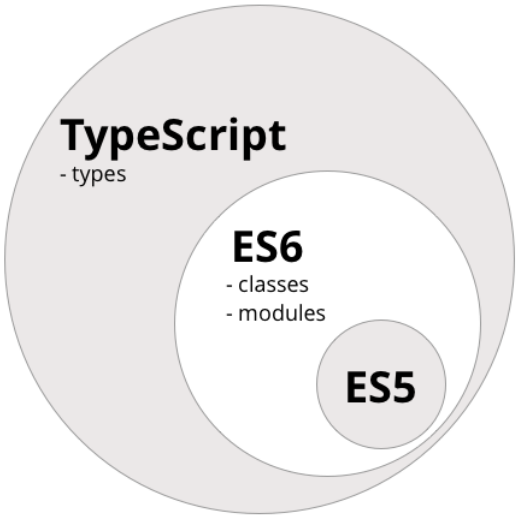

Figure 3.1: TypeScript as a Superset of ECMAScript (Adapted from [21])

3.1.2 JSX

JavaScript XML (JSX) is a syntax extension of JavaScript created by Facebook to define the UI of React applications. It provides a set of XML tags that looks similar to HTML. As TypeScript, JSX needs to be compiled to JavaScript before a browser can understand it [22, Chapter 2, Digging Deeper in JSX]. Code Example 3.1 shows how a JSX syntax is compiled into a function call to the React library. In place of HTML attribute names, JSX uses a camel case[5] property naming convention. An example is the attribute class in HTML that is equivalent to the attribute className in JSX [24].

5 Camel case is a naming convention that capitalizes every first letter of a word in a compound word [23].

```
1 // JSX syntax
2 <h1>Hello World</h1>
3
4 // After compilation to JavaScript
5 React.createElement("h1", null, "Hello World");
```

Code Example 3.1: JSX Compiled to JavaScript [22, Chapter 2, Digging Deeper in JSX]

3.2 Front-end JavaScript Frameworks

The term framework has no traditional definition. It is often used interchangeably with the term library [15, Chapter 4, JavaScript Libraries and Frameworks]. Libraries are collections of re-usable code that focus on solving common programming challenges. Examples are utility libraries for mathematical functions and libraries for accessing databases or file systems. Most definitions acknowledge that both concepts are based on the principle that pre-written code helps a developer to solve a common problem [25]. Different opinions exist about how libraries and frameworks differ from each other. This paper assumes that the major difference is found in the "inversion of control". When working with a library, a developer calls code provided by the library. Frameworks provide developers with a skeleton that can be filled with his or her own code. The code written by a developer is then called by the framework. It is possible for frameworks to contain libraries. Figure 3.2 visualizes the relation between libraries, frameworks and self-written code [26].

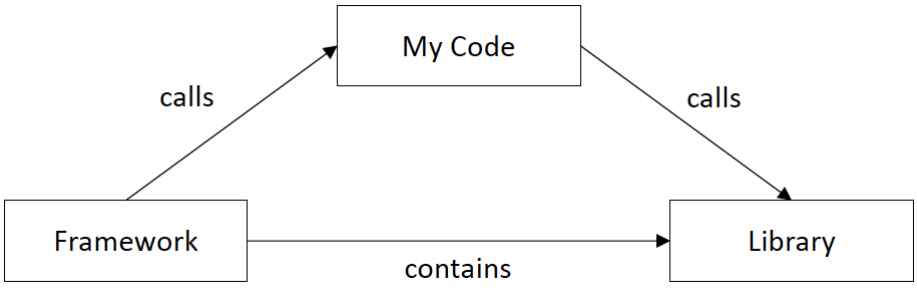

Figure 3.2: Relation Between Libraries, Frameworks and Self-Written Code (Adapted from [26])

The JavaScript frameworks covered in this paper are used in front-end development. A front-end is the visible part of an application. It receives data from a back-end and displays it. HTML, CSS and JavaScript are the three essential parts of a front-end. HTML defines the structure of a UI and CSS adds styling to the structure. JavaScript is responsible for dynamic changes, interactivity and communication with the back-end. A back-end is the part of an application that runs on a server[6] [28, Chapter 6, Where Does Frontend Start?]. It accesses data that is stored in databases and answers requests from clients (e.g. web browsers).

In front-end development, JavaScript frameworks most commonly help to synchronize the data and the UI of an application [6] and to select UI elements in order to manipulate them or to retrieve user input. Other important features they provide are event handling and DOM manipulations [7]. A DOM is a representation of a web page in the form of nodes and objects. It enables JavaScript to change the structure of a web page [29].

3.3 Single-Page and Multi-Page Applications

Two major types of modern web applications are Single-Page Applications (SPAs) and Multi-Page Applications (MPAs). Some JavaScript frameworks, such as Angular, are specifically made for the creation of SPAs.

In MPAs, most of the application is executed on a server. The client is only used to display HTML pages. When a client calls a server, an HyperText Transfer Protocol (HTTP) request is sent to a server and an HTML page is returned as a response. Every time a user performs an action, such as submitting a form with a POST[7] request, the HTML page is replaced by a new page through a web page refresh (see Figure 3.3) [20, Chapter 1, Introducing the Client and Server]. Data is saved on the server-side and every time it is requested or updated, an HTTP request is also sent and causes a web page refresh. This leads to waiting times when a user interacts with an MPA [30, Chapter 1, Comparing Application Types].

6 A server is a device or program that manages and shares network resources for other devices or programs [27].
7 POST is an HTTP method used to send data to a server.

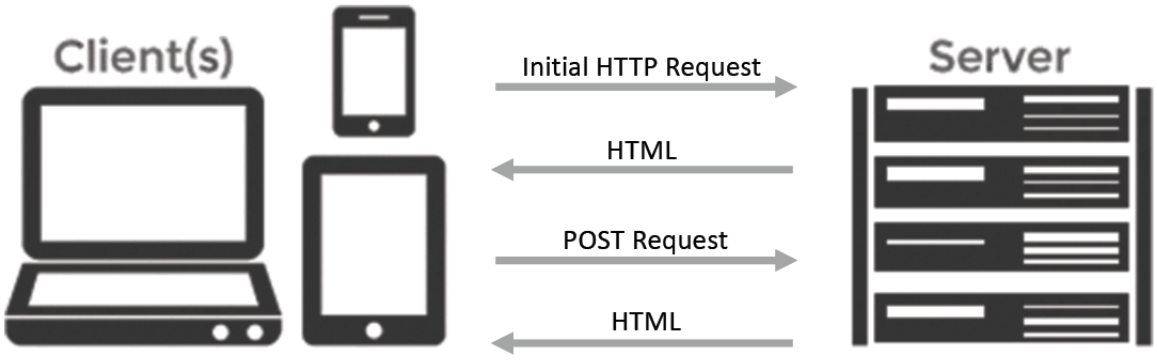

Figure 3.3: Communication Between Client and Server in an MPA (Adapted from [31])

In contrast, SPAs are web applications that request a single HTML page and dynamically update parts of the page on the client-side as the user interacts with it. After an initial HTTP request, interactions with the server happen through Asynchronous JavaScript and XML (AJAX) calls. AJAX enables web pages to be updated asynchronously by exchanging data with a web server in the background without a page refresh [32]. Instead of HTML pages, only data in JSON[8] format is received from the server. This data is used to update the UI without reloading the page. Therefore, SPAs are more fluid and responsive than MPAs when a user is interacting with it [34].

8 JSON is a syntax that is used for storing and exchanging data [33].

4 Methodology

In this chapter, it is explained how the three frameworks Angular, React and Vue.js are evaluated and compared to each other. First, it is outlined how the criteria for the comparison are selected. Then, it is explained how the information that builds the basis for the evaluation is obtained. Lastly, the selected criteria are defined in detail.

4.1 Approach

The three frameworks Angular, React and Vue.js are compared based on a list of criteria that is partly inspired by the publications mentioned in Chapter 2 and partly self-defined. The criteria popularity and ease of development are derived from Graziotin et al. [10]. The criterion performance is derived from Gizas et al. [9]. The criterion learning curve is inspired by a number of blog posts [12, 13, 35]. Scalability and stability are self-defined criteria. All criteria are evaluated separately. A prioritization of the criteria does not take place because the system requirements vary for each use case. Therefore, no overall metric can be created and no single one framework is presented as the best option for all use cases. Instead, it is shown up how the frameworks perform in different aspects to help make a choice about which framework is most suitable for the specific needs of a project.

All information used for comparing the defined criteria results from a literature review and performance measurements. The literature review operates in three steps, as shown in Figure 4.1. In the first step, primary literature and official framework documentation are analyzed. In the next step, secondary and tertiary literature, such as journal articles and blog posts, are analyzed. In the last step, the results are verified by reiterating the first two steps.

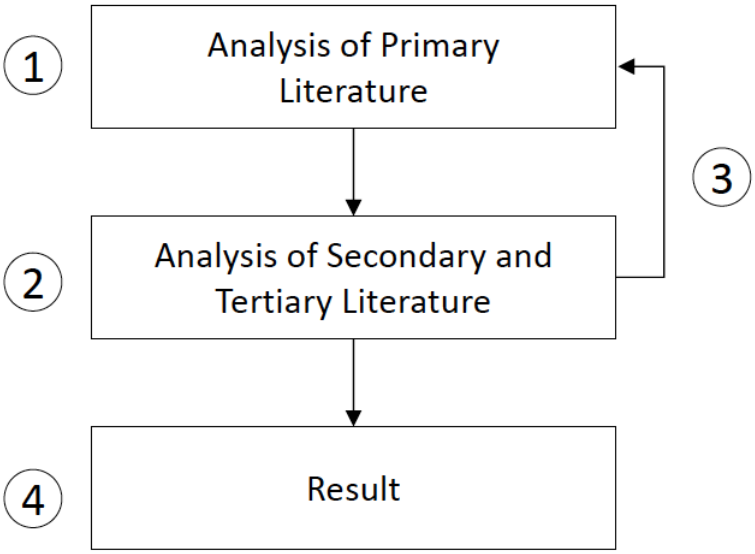

Figure 4.1: Literature Review as a Reiterating Process

For the criteria performance and scalability, time measurements and file size measurements are conducted. Some measurements are conducted with the help of js-framework-benchmark[1]. Js-framework-benchmark is a project that determines a variety of performance metrics for JavaScript frameworks. It provides identical web applications that are built in different JavaScript frameworks, so they can be compared with each other. The applications comprise a table with three columns and six buttons for table operations, such as creating, updating, swapping and deleting rows. Figure A.1 shows the UI of a reference application built with Angular. Each of the frameworks is investigated under consideration of the practices that are suggested by the organization that is supporting and maintaining the framework. To ensure an objective comparison, all information regarding download numbers, questions asked online and numbers of jobs listed for the frameworks are collected on the same day.

1 See https://github.com/krausest/js-framework-benchmark.

4.2 Definition of Criteria

This section defines the criteria used to evaluate and compare the frameworks. Most criteria are split into sub criteria.

4.2.1 Popularity

The popularity of a framework matters for several reasons. The bigger the community around a framework, the higher the chance to find solutions for common problems in online forums or blogs that may not be found in the official documentation. A big community can also help to assure quality of a framework through active bug reports [36].

- **Percentage of Monthly Asked Questions on Stack Overflow**: Stack Overflow[2] is a question-and-answer forum for programmers. It enables programmers to ask questions and share their knowledge [3]. Stack Overflow provides a trend tool[3] that is used to keep track of the users' interests in programming languages and technologies based on the monthly asked questions [37].
- **Percentage of Developers Using the Framework**: Stack Overflow's Annual Developer Survey is the largest and most comprehensive survey of developers worldwide. Since 2019, the survey includes a section about which web frameworks developers use. Participants can select multiple frameworks [5].
- **Node Package Manager (NPM) Downloads**: NPM[4] is the largest software registry worldwide. It is used by open source developers to share code packages so that others can download them [38]. NPM-Stat[5] is a tool that uses NPM's download counts to track how often a package is installed over a time period. It is considered how many times the frameworks were downloaded in 2019.
- **Jobs on Indeed**: Indeed[6] is one of the most popular job websites in the world [39]. It is investigated how many jobs are listed for the frameworks.

2 See https://stackoverflow.com.
3 See https://insights.stackoverflow.com/trends.
4 See https://www.npmjs.com.
5 See https://npm-stat.com.
6 See https://www.indeed.com.

4.2.2 Learning Curve

Depending on how difficult it is to learn a technology, its learning curve is considered small or steep. A small learning curve indicates that a technology is easy to learn and a steep learning curve indicates that it is difficult to learn. Blog posts and articles that address the learning curves of all three frameworks are considered to evaluate the learning curves.

4.2.3 Ease of Development

The criterion Ease of Development covers a set of sub criteria:

- **Availability of Instructions**: The availability of instructions for a framework is an important factor for a user's learning process and his or her ability to solve problems with a technology. Therefore, it is considered if the frameworks have detailed documentation.
- **Percentage of Questions on Stack Overflow That Are Unanswered**: For each framework, it is considered which percentage of the total number of questions on Stack Overflow is unanswered.
- **Debugging Options**: Debugging is the process of locating and removing program errors. Debugging tools help developers to understand and solve problems [40]. It is considered if the frameworks provide advanced debugging options.

4.2.4 Scalability

A scalable software can grow and manage increased demand while retaining good performance [41]. In this paper, increased demand is investigated in terms of data objects that are shared over many components in an application. Another aspect of scalability is the maintainability of software. It should be possible to add new features at a later time without having to rewrite the rest of an application [42, 43]. Three aspects are investigated:

- **State Management**: Web applications have state data, often referred to as just state, that contains information that is used in a UI. These data objects typically do not just affect one but several parts of a UI. It is analyzed how the frameworks manage this data and ensures that all parts of an application are synchronized.

- **Project Structure**: It is examined how the frameworks organize files in a project.
- **Modularity**: The idea of Modularity is to isolate different parts of an application as much as possible and to have well-defined interfaces for the parts that work together so they can easily be maintained or replaced if needed [44]. It is reviewed how the frameworks realize this concept.

4.2.5 Performance

Website speed is a ranking factor for Google [45] and has an important impact on the UX of a web application [46]. Therefore, three aspects are considered to determine the performance of a framework:

- **Total Byte Weight**: The size of a website or web application has an impact on the time it takes to download and initially load it. Google Chrome's network activity inspection tool[7] is used to investigate the total byte weight of all transferred resources for reference applications provided by js-framework-benchmark. Each framework uses the same CSS and font resource files. Also, a first-time visit is emulated by emptying the browser cache before measurements are conducted.
- **Script Boot Up Time**: The time taken for loading, parsing and rendering is measured for all frameworks.
- **Operation Benchmarks**: Several table operations are performed on reference applications provided by js-framework-benchmark. For each operation, the time taken for script execution, rendering and painting is measured. Script execution refers to the execution of JavaScript as a response to an event. Rendering describes the process of parsing HTML and CSS to create a render tree that contains information about all visible elements on a page. Also, the exact position and sizing of elements is calculated [47]. Painting is a term to describe the process of filling in pixels on a screen [48].

  The benchmarks are automated with Selenium WebDriver[8], a tool for automating web browsers. Each benchmark is performed twelve times on a MacBook Pro (13-inch 2,9 GHz Intel Core i5,

7 See https://developers.google.com/web/tools/chrome-devtools/network/.
8 See https://www.seleniumhq.org/projects/webdriver/.

8 GB RAM, OSX 10.13.6) in Google Chrome (Version 74.0.3729.169 64-Bit). As a result, the average time (mean) for each operation is presented with a 95% confidence interval.

The operations that are measured are:

- Creating 1,000 rows in a table after the page has loaded
- Creating 10,000 rows
- Updating every 10th row of a table with 10,000 rows
- Replacing 1,000 rows of the table
- Removing a row on a table with 1,000 rows
- Clearing a table with 10,000 rows

4.2.6 Stability

Stability is understood as the probability of unexpected effects that are caused by changes in software [49]. The stability of a framework has a large impact on how maintainable a web page or web application is. When working with an unstable framework, many adjustments may need to be made to a software as a consequence of breaking Application Program Interface (API) changes.

All three frameworks use semantic versioning. Semantic versioning suggests three levels of change that can be introduced with a release. An example for a version number that follows the principles of semantic versioning is 7.2.11. The first number indicates major releases that contain API updates. Major releases are not backward compatible[9]. Therefore, when updating to a new major release, software developer assistance is expected. The second number indicates minor releases that add functionality. Minor releases are backward compatible and do not require software developer assistance unless a developer wants to use a newly released feature. The last number stands for patch releases that contain backward compatible bug fixes [51].

- **Software Releases**: It is reviewed how frequently major versions are released that include breaking API changes and require developer assistance.

9 A system is considered backward compatible when it is able to function with interfaces and data from earlier versions [50].

5 Investigation of Frameworks

In this chapter, the three frameworks are individually presented. For each framework, some background information is outlined and then the characteristics and core features are explained. Finally, the frameworks are reviewed in terms of the criteria defined in Section 4.2.

5.1 Angular

Angular is a framework to create single-page applications in HTML, TypeScript and CSS [52]. It is maintained by Google. The first version of Angular, also known as AngularJS, was released in 2009 [20, Chapter 2, Overview]. The name derives from the fact that it runs on JavaScript and is therefore very different from all versions released after Angular 2. All versions past Angular 2 use TypeScript and are referred to as Angular [20, Introduction, Angular and Naming]. Angular 8 was released in May 2019 and is the latest version of the framework. In this paper, only Angular is considered. AngularJS is not regarded.

5.1.1 Characteristics

An Angular application comprises at least one, usually several Angular Modules (NgModules). NgModules are classes that define relationships between the building blocks of an application, such as components and services. Components are UI building blocks that contain a class that holds application logic and an HTML template that defines a view[1]. The HTML template is combined with Angular markup to add program logic. Event and property bindings are used connect HTML elements with application data. To share data and logic across components, services are used. Services are injected into components as dependencies. Both services and components are classes with meta information that defines their type [52].

1 A view is what is displayed in a browser after an HTML template and application data have been processed together [53].

Angular supports two-way data binding. This means that the application data can change the DOM, and conversely, changes in the DOM (through user interaction) can change the application data. A router module allows to map components to Uniform Resource Locators (URLs) so a user can navigate the interface of a SPA. Figure 5.1 shows a visualization of Angular's architecture [52].

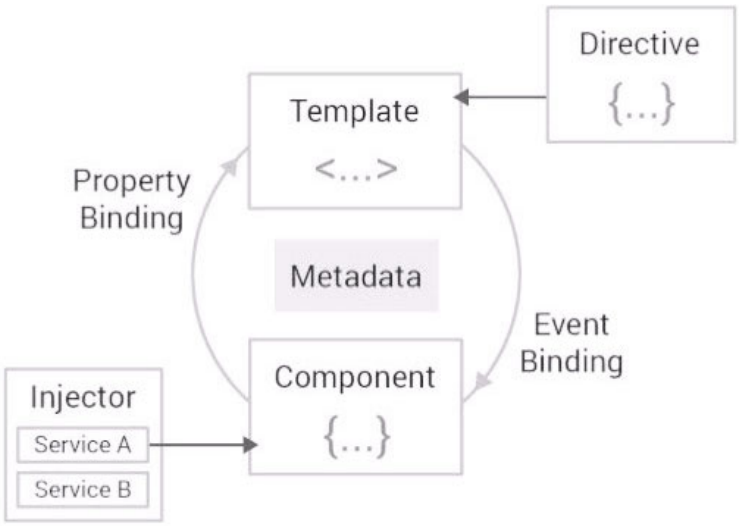

Figure 5.1: Visualization of Angular's Architecture (Adapted from [54])

5.1.2 Popularity

Table 5.1 shows the popularity of Angular in the form of Stack Overflow trends, NPM downloads and jobs listed on Indeed. Regarding the number of developers using the framework, it needs to be noted that the survey the data is obtained from did not differentiate between Angular and Angular.js. The difference between Angular and Angular.js is explained in 5.1.

Percentage of Monthly Asked Questions on Stack Overflow	2.8% [55]
Percentage of Developers Using the Framework	30.7% [5]
NPM Downloads in 2019	40,103,722 [56]
Jobs on Indeed	14,816 [57]

Table 5.1: Popularity Characteristics of Angular

5.1.3 Learning Curve

Angular is a framework that offers a complete solution for a web application front-end. It requires developers to learn several concepts such as Angular's component system, NgModules, services, routing and dependency injection. Aside from HTML and CSS, developers need to know TypeScript (see Section 3.1.1). Also, developers need to understand Reactive Extensions for JavaScript (RxJS) [58], a library Angular uses for reactive programming with asynchronous streams and callbacks [59, 60]. Overall, due to the large number of concepts that a developer needs to know, blog entries that address Angular's learning curve consider Angular difficult to learn [13, 61, 62, 63, 35, 58].

5.1.4 Ease of Development

Availability of Instructions Angular has detailed documentation[2] that provides information about the setup of a development environment, core features, development workflow, project structure, configuration files, release practices, testing, Command-Line Interface (CLI) commands and about its API. Additionally, a cheat sheet[3] is provided for quick reference.

Percentage of Questions on Stack Overflow That Are Unanswered The total number of questions asked on Stack Overflow regarding Angular is 170,313. 34,811 (20.44%) of these questions are unanswered [64].

Debugging Options Angular has a built-in debugging tool that provides two helpful commands. To see what the state of a component is, the command ng.probe($0).componentInstance can be executed in the browser console after selecting a component with the Chrome inspection functionality. The command ng.profiler.timeChangeDetection() can detect the time an application needs for computing changes that lead to UI updates [65].

Also, Angular implements a JSON pipe that converts a data object into a JSON format and prints it on the page for debugging purposes (see Code Example 5.1) [66].

2 See https://angular.io/docs.
3 See https://angular.io/guide/cheatsheet.

```
@Component({
  selector: 'json-pipe',
  template: `<div>
    <p>Without JSON pipe:</p>
    <pre>{{object}}</pre>
    <p>With JSON pipe:</p>
    <pre>{{object | json}}</pre>
  </div>`
})

export class JsonPipeComponent {
  object: Object = {foo: 'bar', baz: 'qux', nested: {xyz: 3, numbers: [1, 2, 3, 4, 5]}};
}
```

Code Example 5.1: Angular's JSON Pipe [66]

Augury[4], a browser extension developed by Google, can be used within Google Chrome's and Firefox's DevTools panel for more advanced debugging. It gives a visual representation of the structure of an Angular application and of the relationships between its building blocks. Also, dependencies, data binding, events and object properties can be investigated [67].

Aside from debugging options specific to Angular, native JavaScript debugging features are supported within Angular applications. JavaScript debugger statements make it possibles to stop code from executing at a defined moment and JavaScript's console.log() allows a developer to write into the browser console [68].

5.1.5 Scalability

State Management There are three types of communication between components. Parent components can share data with their child components, child components can share data with their parent component and unrelated or sibling components can share data between each other. Parent components are components that have at least one or more components (child components) nested inside of them. Components are considered siblings if they share the same parent component [69].

4 See https://augury.rangle.io.

To communicate with a child component, a parent component can pass data through a variable in its HTML template. Line 2 in Code Example 5.2 shows how a variable parentMessage, that would be defined in the parent component class, is passed to a child component. In the child component class, the data can then be accessed by using an @Input() decorator (see line 5 in Code Example 5.2) [70].

```
1 // In the parent component
2 <app-child [childMessage]="parentMessage"></app-child>
3
4 // In the child component
5 @Input() childMessage: string;
```

Code Example 5.2: Communication from a Parent to a Child Component in Angular (Adapted from [69])

Child components can communicate with their parent in two ways. As shown in Code Example 5.3, a parent component can declare a variable with a @ViewChild() decorator and then access its child's attributes and functions. The child's view needs to be initialized for the parent to access it. Therefore, the child's data needs to be accessed within the ngAfterViewInit() function that is triggered when a component's views and its child views are initialized [69].

```
1  export class ParentComponent implements AfterViewInit {
2
3    @ViewChild(ChildComponent) child;
4
5    constructor() { }
6
7    message:string;
8
9    ngAfterViewInit() {
10     this.message = this.child.message
11   }
12 }
```

Code Example 5.3: Communication from a Child to a Parent Component in Angular (Adapted from [69])

Another way for child components to communicate with their parent is by using an @Output() decorator and an EventEmitter object. This approach is ideal when data changes are shared that occur through user events. In the child component, a variable is declared with an @Output() decorator that holds an EventEmitter. The parent component subscribes to a child's EventEmitter through a property in its template. When the function emit() is called on the EventEmitter in the child, the parent can access the data that is sent as a parameter in the emit() function [70].

For communication between unrelated components, a shared service can be used. The service defines a data object that is to be shared and provides a function to update it. Any component can then inject the service and subscribe to the data object [70]. To change data, a component calls the update function provided by the service [69].

State management can fully be handled by Angular without the use of an external library, but the community around Angular developed a framework called NgRx[5] to help manage state in complex applications. NgRx uses a single immutable data structure that saves state in a store. The store is a controlled container that has functions that can be called to access the state and to update it by creating a new data structure that overwrites the old one [71].

Project Structure An Angular project can be initialized with the Angular CLI. An application skeleton is then set up in a src/ folder. It contains the following files that are relevant for structuring and maintaining an application [72]:

- **app/**: contains modules, components and services
  - **app.component.ts**: the root component
  - **app.component.html**: the HTML template for the root component
  - **app.component.css**: the CSS for the root component
  - **app.component.spec.ts**: contains unit tests for the root component
  - **app.module.ts**: declares all components used in the root module and defines how the application is assembled

5 See https://ngrx.io/.

- **assets/**: contains images and other assets
- **environments/**: contains build configuration options for different target environments
- **index.html**: the main HTML page of a SPA that is delivered to a browser
- **main.ts**: the main entry point that is responsible for bootstrapping an application
- **styles.sass**: lists all CSS files
- **test.ts**: contains unit tests

Within the app/ folder, NgModules are used to organize code and bundle related components [73].

Modularity Angular applications are modular on two levels. NgModules bundle building blocks that are related to each other (see Figure 5.2). Each building block used within an NgModule is declared in the metadata of a module file. Through an export property, a building block can be declared as public so it can be used in other NgModules [74]. Another instance of Angular's modularity is the component concept Angular uses. Components are the basic UI building blocks of Angular applications. A component comprises a view that is defined by an HTML template and a class that defines the application logic for the view [75].

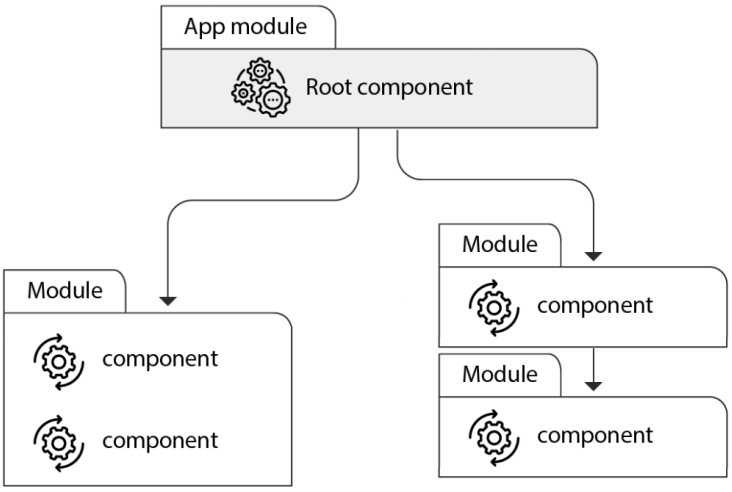

Figure 5.2: Relation between NgModules and Components [76]

5.1.6 Performance

For this criterion, a reference application built with Angular 7.1.4 is investigated. The results of the performance benchmarks defined in Section 4.2.5 are displayed in Table 5.2 below.

It stands out that the operations to create 1,000 rows, replace 1,000 rows and update every tenth row in a table with 10,000 rows take a very similar amount of time.

	Measurement ± 95% Confidence Interval
Total Byte Weight	359 KB ± 0
Script Boot Up	220.93 ms ± 13.53 ms
Create 1,000 Rows	181.74 ms ± 18.68 ms
Create 10,000 Rows	1725.39 ms ± 65.88 ms
Replace 1,000 Rows	174.84 ms ± 11.87 ms
Update Every 10th Row	185.09 ms ± 11 ms
Remove a Row	55.27 ms ± 5.26 ms
Clear 10,000 Rows	382.29 ms ± 28.38 ms

Table 5.2: Results of the Benchmarks Performed on an Angular Reference Application

5.1.7 Stability

Software Releases Angular's version releases are based on a semantic versioning convention (see Section 4.2.6). Google releases major versions that include API changes and significant new features every six months. For each major release, there are about one to three minor releases that are backward compatible and contain smaller features. Patch releases come out almost weekly [77].

5.2 React

React is a JavaScript framework that is used to build interactive user interfaces for web pages and web applications. Facebook created it to develop complex user interfaces that work with changing application state. React's initial release was in 2013 [78, Chapter 1, Defining React]. Version 16.8.6 is the latest version of the framework. It was published in March 2019. Its official documentation refers to React as a library [79]. However, in this paper, given the definition of libraries and frameworks in Section 3.2, React is understood as a framework because it calls code that is written by a developer. An example is shown below in Code Example 5.4, where React's lifecycle method render() calls self-written code.

5.2.1 Characteristics

React applications are built in ES6 JavaScript, JSX (see Section 3.1.2) and CSS. Through a component-based approach, React breaks complex user interfaces into independent and reusable pieces. A component defines a view in JSX and is rendered when it is called by React's virtual DOM, as shown in line 10 of Code Example 5.4. React's virtual DOM is a virtual representation of the UI of an application that is synced with the "real" DOM [80]. Components can have their own state data or use data of parent components via "props". Props are properties that are passed to a component from outside. To give an example, in line 10 of Code Example 5.4, the name "Taylor" is passed to a component as props. Within the component, the name can then be accessed with this.props.name [79].

```
1  class HelloMessage extends React.Component {
2    render() {
3      return (
4        <div>
5          Hello {this.props.name}
6        </div>
7        // ...
8
9  ReactDOM.render(
10   <HelloMessage name="Taylor" />,
11   document.getElementById('hello-example')
12 );
```

Code Example 5.4: Rendering a React Component (Adapted from [79])

React implements a one-way data flow. This means that user input cannot directly change application state through DOM manipulations. Instead, the input data needs to be passed down in the component hierarchy [81] so the state data model can update the view from within the affected component. Figure 5.3 is a visualization of this concept [58].

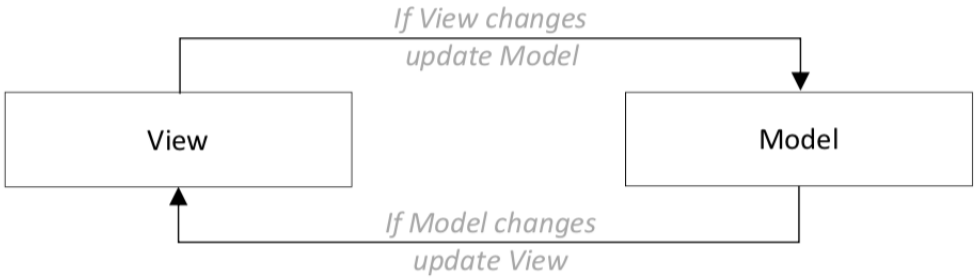

Figure 5.3: Visualization of React's One-Way Data Flow (Adapted from [82])

5.2.2 Popularity

Table 5.3 shows the popularity of React in the form of Stack Overflow Trends, NPM downloads and jobs listed on Indeed.

Percentage of Monthly Asked Questions on Stack Overflow	3.18% [55]
Percentage of Developers Using the Framework	31.3% [5]
NPM Downloads in 2019	113,337,137 [56]
Jobs on Indeed	62,303 [83]

Table 5.3: Popularity Characteristics of React

5.2.3 Learning Curve

React requires a developer to know ES6 JavaScript, JSX and CSS. On top of these programming languages, a developer needs to understand React's concepts regarding component rendering, event handling and data management with state and props [58]. React's learning curve is considered moderate to steep by blog entries that address how easy or difficult it is to learn React [58, 12, 35, 63, 13, 61, 62].

5.2.4 Ease of Development

Availability of Instructions React's documentation[6] provides information about setting up a React project, JSX, component rendering, state and props, event handling, testing, React's virtual DOM and React's API. Additionally, the documentation includes an FAQ section that covers React's main concepts.

Percentage of Questions on Stack Overflow That Are Unanswered The total number of questions asked on Stack Overflow regarding React is 144,649. 27,842 (19.25%) of these questions are unanswered [84].

Debugging Options Facebook has developed React Developer Tools[7], a browser extension for Google Chrome and Firefox. It adds React-specific debugging options to built-in developer tools in browsers. It enables developers to inspect the virtual DOM, the component hierarchy and state of a React application. React Developer Tools also exists as a standalone app that is compatible with other browsers [85]. On top of React-specific debugging options, native JavaScript debugging features, such as debugger statements and console.log() calls, are supported.

5.2.5 Scalability

State Management React has stateful and stateless components. Stateful components have their own state data. Stateless components receive data from the outside as props. A stateful component can change its own state by calling this.setState() (see line 9 of Code Example 5.5). State is only accessible to the component that owns it or to a child when state is passed to it as props [86].

Props are read-only and cannot be modified [87]. To have a child component communicate with a parent, a data object is declared as state in the parent component and an update function is passed to the child (see line 18 of Code Example 5.5). In this way, when the child component calls the update method, the state is updated internally in the parent component [88].

6 See https://reactjs.org/docs/getting-started.html.
7 See https://github.com/facebook/react-devtools.

```
class ParentComponent extends React.Component {
  constructor(props) {
    super(props);
    this.handleValueChange = this.handleValueChange.bind(this);
    this.state = {value: 10};
  }

  handleValueChange (value) {
    this.setState({value});
  }

  render() {
    const value = this.state.value;
    return (
      <div>
        <childComponent
          value={value}
          handleValueChange={this.handleValueChange} />
        // ...
```

Code Example 5.5: Communication from a Child to a Parent Component in React (Adapted from [88])

If components communicate that are not related to each other in a child-parent relationship, the state to be shared is declared in the closest common ancestor. Both components can then access the shared state via props and manipulate it by triggering the update function in the common ancestor component [88].

Aside from React's built-in state management options, there are several state management libraries available that can be used with React. A popular state management library for React is Redux [89]. Redux is similar to NgRx (see Section 5.1.5). It saves application state in an object tree within a single store object. State is read-only and can only be changed with an action, an object that describes what happened that causes a change. Actions are used together with the old state object to return a new state object [90]. Other state management libraries that can be used with react are MobX[8] and Apollo[9] [91].

8 See https://mobx.js.org.
9 See https://www.apollographql.com/docs/react/essentials/local-state/.

Project Structure React has an official tool[10] that can set up an initial project for a React application. The setup contains two folders, public/ and src/ and several files. Below, relevant files are explained [92].

- **public/**
  - **index.html**: the page template that is delivered to a browser
- **src/**
  - **app.css**: a CSS file for the root component
  - **app.js**: the root component
  - **app.test.js**: contains unit tests
  - **index.css**: a global CSS file for all components
  - **index.js**: JavaScript entry point

React does not specify a way of organizing components in the src/ folder. The documentation only shows up some common structuring approaches, such as grouping files based on features, routes or file types [93].

Modularity React applications are modular in terms of their component-based organization. Components are independent and reusable code pieces that can be nested [87].

5.2.6 Performance

For this criterion, a reference application built with React 16.8.3 is investigated. The results of the performance benchmarks defined in Section 4.2.5 are displayed in Table 5.4 below.

It is interesting the times it takes to create and to replace 1,000 rows are similar, but the time to update 1,000 rows is significantly longer.

10 See https://github.com/facebook/create-react-app.

	Measurement ± 95% Confidence Interval
Total Byte Weight	261 KB ± 0
Script Boot Up	83.49 ms ± 5.12 ms
Create 1,000 Rows	185.12 ms ± 9.21 ms
Create 10,000 Rows	2026.81 ms ± 114.61 ms
Replace 1,000 Rows	167.71 ms ± 6.04 ms
Update every 10th Row	231.58 ms ± 15.14 ms
Remove a Row	62.77 ms ± 3.49 ms
Clear 10,000 Rows	185.15 ms ± 8.48 ms

Table 5.4: Results of the Benchmarks Performed on a React Reference Application

5.2.7 Stability

Software Releases React follows semantic versioning principles (see Section 4.2.6). Facebook tries to minimize the number of major releases that include API changes and require developer assistance. To indicate when and where developer assistance is required, React provides development warnings before major releases when a syntax or function that changes is used in an application. A fixed schedule for how often new versions are released does not exist. The last two major versions were released in April 2016 and September 2017. The next major version is supposed to come out sometime in 2019. New features are typically released in backward compatible minor versions [94].

5.3 Vue.js

Vue.js is a JavaScript framework for building user interfaces created by the former Google employee Evan You. Its documentation refers to Vue.js as a progressive framework because it is incrementally adoptable [95]. This means that Vue.js can be used on individual parts of a user interface, but also power complex SPAs when combined with supporting third-party libraries. The first version of Vue.js was released in 2014 [96]. The most recent version is v2.6.10 [97].

5.3.1 Characteristics

A Vue.js application comprises a nested component tree and has at least one Vue instance [98]. A Vue instance is an object that holds application data. Components are reusable Vue instances that have their own template, data, methods and styling [99]. Vue.js supports the use of ES5 and ES6 JavaScript.

Code Example 5.6 shows how a root Vue instance is bound to an HTML element by setting a property el equal to an element's ID selector. The property data within the Vue instance holds data for the selected element. The value of the property message is accessed in the template by String Interpolation in line 3. String Interpolation is a type of data binding that uses double curly brackets to insert a string value into a template. If the data in the Vue instance changes, the element is updated [100, 95].

Vue.js uses special attributes called directives to add functionality to the template of a component. Directives start with a "v-" prefix [101]. An example is v-model that allows two-way data binding on form input [102] and v-if (see Code Example 5.6 line 3) that conditionally renders an element depending on a Boolean value [101].

```
1 // HTML template
2 <div id="app">
3   <p v-if="seen">{{ message }}</p>
4 </div>
5
6 // Vue instance
7 var app = new Vue({
8   el: '#app',
```

```
  data: {
    message: 'Hello Vue!',
    seen: true
  }
})
```

Code Example 5.6: Example of a Root Vue Instance (Adapted from [95])

5.3.2 Popularity

Table 5.5 shows the popularity of Vue.js in the form of Stack Overflow Trends, NPM downloads and jobs listed on Indeed.

Percentage of Monthly Asked Questions on Stack Overflow	0.94% [55]
Percentage of Developers Using the Framework	15.2% [5].
NPM Downloads in 2019	19,374,555 [56]
Jobs on Indeed	3,447 [103]

Table 5.5: Popularity Characteristics of Vue.js

5.3.3 Learning Curve

To use Vue.js, a developer needs to know HTML, CSS and JavaScript [95]. Vue.js is a progressive framework, that means it is up to the developer if Vue.js is used for only a single part of the UI or if it is used in a more complex manner. Therefore, if Vue.js is just used for a small and isolated part of the UI, a developer may not need to learn all of Vue.js' concepts. Blog entries regarding how easy or difficult it is to learn Vue.js describe it as intuitive to use and consider its learning curve easy to moderate [104, 12, 35, 63, 13, 61, 62].

5.3.4 Ease of Development

Availability of Instructions The documentation[11] of Vue.js provides detailed information about installing Vue.js, data management, data binding, rendering, components, testing and Vue.js' API. Additionally, Vue.js provides a framework comparison page[12] to identify which of Vue.js' concepts are similar to other frameworks and which are different. This can be useful for developers that are switching to Vue.js from other frameworks.

Percentage of Questions on Stack Overflow That Are Unanswered The total number of questions asked on Stack Overflow regarding Vue.js is 36,238. 7,347 (20.27%) of these questions are unanswered [105].

Debugging Options For debugging, Vue.js' documentation suggests the use of Vue Devtools[13]. Vue Devtools is available as a standalone application that works in any environment and as a browser extension for Google Chrome and Firefox [106]. It allows developers to view an application's component tree, to inspect the props and data of selected components and to retrace event chains. With a live-edit feature, developers can change data properties and see instantly how the application is affected. Additionally, the tool gives the option to do "time travel" debugging for Vuex[14], a state management library for Vue.js applications [107]. Time travel debugging allows to cycle through different versions of state objects [108].

Aside from Vue-specific debugging options, native JavaScript debugging features, such as debugger statements and console.log() calls, are supported [107].

5.3.5 Scalability

State Management In a Vue.js application, a parent component can pass data to child components with props. Props are custom attributes that are registered on child components, as shown in line 10 of Code Example 5.7. After registering them, they are treated as regular data properties once a value is passed by the parent component (see Code Example 5.7 lines 3-5) [99].

11 See https://vuejs.org/v2/guide/.
12 See https://vuejs.org/v2/guide/comparison.html.
13 See https://github.com/vuejs/vue-devtools.
14 See https://vuex.vuejs.org.

```
// Parent component
<div id="app">
   <blog-post title="My journey with Vue"></blog-post>
   <blog-post title="Blogging with Vue"></blog-post>
   <blog-post title="Why Vue is so fun"></blog-post>
</div>

// Child component
Vue.component('blog-post', {
  props: ['title'],
  template: '<h3>{{ title }}</h3>'
})
```

Code Example 5.7: Passing Data from a Parent to Child Components in Vue.js (Adapted from [99])

For communication from a child component to a parent component, the parent component implements an event listener v-on in its template. Then it can receive an event when a child component calls $emit. The emit method takes at least an event name as an argument and, optionally, values can be passed [99].

When components communicate that are not connected to each other through a child-parent relationship, it is recommended to use a centralized event hub [109]. A centralized event hub can be an empty Vue instance that is shared between the communicating components. The functions $emit, $on and $off are then called on the event hub to emit events, listen for events and unregister event listeners [109].

For more complex applications, it is recommended to use a state management library, such as Vuex [110, 109]. Vuex is a popular state management library created by the Vue.js team that offers a centralized store for state data. A store container holds all application state and when state changes, affected components are updated. To update state, a mutation (a method that changes the state) needs to be defined and committed [111].

Project Structure Vue.js' CLI[15] can be used to create an initial project structure for a SPA. It contains the following files that are relevant for structuring and maintaining an application [112]:

- **src/**
  - **assets/**: contains images and other assets
  - **App.vue**: the root component
  - **components/**: contains UI components
  - **main.js**: the main entry point that imports Vue.js and renders Vue.js' main component to the DOM
- **index.html**: the main HTML page of an SPA
- **test/**: contains files for testing

Except for suggesting to store components in a component/ folder within separate files for each component [113], Vue.js does not enforce or suggest how to structure files within the src/ folder.

Modularity Vue.js applications are modular by being split up into components that comprise separate UI building blocks [99].

5.3.6 Performance

For this criterion, a reference application built with Vue.js 2.6.2 is investigated. The results of the performance benchmarks defined in Section 4.2.5 are displayed in Table 5.6 below.

It is surprising that updating 1,000 rows takes more than twice as long as it takes to create or to replace 1,000 rows.

5.3.7 Stability

Software Releases Vue's release policy follows semantic versioning principles (see Section 4.2.6). An official schedule for the releases of major versions does not exist [114]. However, the Vue.js team states that they announce major releases at least six months before the release date. The past two major versions of Vue.js were released in October 2015 and September 2016 [115]. Since

15 https://cli.vuejs.org

	Measurement ± 95% confidence interval
Total Byte Weight	211 KB ± 0
Script Boot Up	65.82 ms ± 13.35 ms
Create 1,000 Rows	185.34 ms ± 7.03 ms
Create 10,000 Rows	1722.41 ms ± 107.05 ms
Replace 1,000 Rows	166.34 ms ± 8.7 ms
Update Every 10th Row	379.83 ms ± 51.42 ms
Remove a Row	73.51 ms ± 4.61 ms
Clear 10,000 Rows	214.94 ms ± 18.65 ms

Table 5.6: Results of the Benchmarks Performed on a Vue.js Reference Application

Vue.js version 2.6, minor versions come out every three months. Patch releases that include bug fixes come out as often as necessary [114].

6 Comparison

In this chapter, the three frameworks are compared to each other regarding the criteria defined in Section 4.2. Similarities and differences are shown up.

6.1 Popularity

According to all four aspects that were considered to evaluate the popularity of the frameworks, React is the most popular, Angular the second most popular and Vue.js the least popular framework. In regard to the monthly asked questions on Stack Overflow and the percentage of developers using the framework, Angular and React are similar. Concerning the number of NPM downloads and jobs on Indeed, the differences are more significant. This discrepancy could be a consequence of the lack of distinction between Angular and AngularJS. The results for Vue.js are significantly lower than for Angular and React. Table 6.2 shows the exact results for all criteria.

	Angular	React	Vue.js
Percentage of Monthly Asked Questions on Stack Overflow	2.8%	3.18%	0.94%
Percentage of Developers Using the Framework (Participants could select multiple frameworks)	30.7%	31.3%	15.2%
NPM Downloads in 2019	40,103,722	113,337,137	19,374,555
Jobs on Indeed	14,816	62,303	3,447

Table 6.1: Overview of the Numbers of Questions on Stack Overflow for Angular, React and Vue.js

6.2 Learning Curve

Seven developer blogs that evaluate the learning curves of Angular, React and Vue.js were investigated for this criterion. Six of the seven blogs state that Angular has the steepest, React the second steepest and Vue.js the smallest learning curve [104, 58, 12, 35, 63, 13, 61]. One blog expresses that React and Vue.js are both easier to learn than Angular, but does not specify which of the two frameworks, React or Vue.js, is easier to learn [62].

The main reason for Angular's steep learning curve is the large number of concepts Angular includes [58]. Angular and React both require learning another language, TypeScript and JSX, on top of HTML, CSS and JavaScript [35]. Vue.js only requires knowledge of HTML, CSS and JavaScript [62]. Also, Vue.js can be added incrementally to a project. Therefore, a developer may not need to learn the whole spectrum of concepts that Vue.js provides if it is used only for a small part of an application.

6.3 Ease of Development

Availability of Instructions Each of the frameworks has detailed documentation that provides information about its concepts and features. Additionally, Angular provides a cheat sheet for quick references about its most important features. React's documentation includes an FAQ section that answers popular questions about React's main concepts. Vue.js' documentation contains a page that compares Vue.js with other popular frameworks to help developers that are transitioning to Vue.js from other frameworks.

Percentage of Questions on Stack Overflow That Are Unanswered The percentage of unanswered questions on Stack Overflow is similar for all three frameworks. With 19.25% React has the lowest percentage of unanswered questions. Vue.js and Angular have about 1% more unanswered questions than React. The exact numbers are shown in Table 6.2.

	Angular	React	Vue.js
Total Number of Questions Asked on Stack Overflow	170,313	144,649	36,238
Number of Unanswered Questions	34,811	27,842	7,347
Percentage of Unanswered Questions	20.44%	19.25%	20.27%.

Table 6.2: Overview of the Popularity Characteristics of Angular, React and Vue.js

Debugging Options Aside from native JavaScript debugging features that all three frameworks support, Angular has more built-in debugging options than React and Vue.js. For all three frameworks, advanced debugging tools exist in the form of browser extensions for Google Chrome and Firefox. Additionally, React and Vue.js provide stand-alone applications of their debugging tools that are compatible with any environment. If browsers other than the above-mentioned ones are used, React and Vue.js offer better debugging options than Angular.

6.4 Scalability

State Management All three frameworks provide ways for child and parent components to share data between each other. In Angular applications, a shared service can be used for communication between unrelated or sibling components. In Vue.js applications, a centralized event hub can be used in simple scenarios. For more complex scenarios, it is recommended to use a state management library. React does not offer a way for unrelated or sibling components to share data directly. Instead, the shared data is uplifted to the closest common ancestor and is accessed via props. This can become confusing when a lot of data is shared over many components.

Both frameworks, Vue.js and React, suggest the use of a state management library for complex and large-scale scenarios. Angular offers a complete state management solution and does not require an external state management library. Therefore, Angular is considered the most scalable framework in the category state management.

Project Structure Angular, React and Vue.js provide CLI tools that can be used to set up an initial project structure. In an Angular application, most code is organized within a folder named app/. In this folder, related components, services and other building blocks are bundled in NgModules. Vue.js suggests storing components within a component/ folder but does not specify any best practices for organizing files within the folder. React also leaves it up to the developer to decide how to organize files within a project.

React and Vue.js are both flexible regarding the structure of a project. Angular offers a concrete approach for organizing files within a project through NgModules.

Modularity Angular realizes the concept of modularity through components and NgModules. Components define the UI building blocks of an application. NgModules define the relationships between components and other related building blocks. React and Vue.js realize modularity only through a component-based approach that splits an application into reusable and independent UI building blocks. How components are grouped is not determined by the two frameworks. Therefore, Angular has a more modular structure than React and Vue.js.

6.5 Performance

The reference applications used for the performance benchmarks are built with Angular 7.1.4, React 16.8.3 and Vue.js 2.6.2. Table 6.3 shows the total byte weight of the three reference applications. Figure 6.1 shows the results for all time-based performance benchmarks with a 95% confidence interval. For better readability, the figure does not display the times it took to create 10,000 rows with the frameworks.

The Vue.js application has the smallest total byte weight and the shortest script boot up time. Therefore, Vue.js is the fastest framework to initially load an application. The total byte weight of the React application is a little larger than Vue.js' and it takes a little longer for the script boot up than Vue.js. The Angular application has the largest total byte weight and a significantly longer script boot up time than both other frameworks.

Creating and replacing 1,000 rows takes almost the same time with all three frameworks. Creating 10,000 rows takes 1,720 ms with Angular and 1,722 ms with Vue.js. React takes around 300 ms

	Angular	React	Vue.js
Total Byte Weight	359 KB	261 KB	221 KB

Table 6.3: Sizes of the Reference Applications Built With Angular, React and Vue.js

longer than them. Updating rows takes the shortest time with Angular, followed by React. Vue.js takes significantly more time to update rows. The time differences between the frameworks to remove a row are small. Clearing a table with 10,000 rows is the fastest with React, followed by Vue.js. Angular takes significantly longer for this task.

Overall, Vue.js has the best initial page load performance, but it takes longer for updating rows than Angular and React. Angular takes the shortest time for three out of five table operations, but it takes significantly longer than Vue.js and React to initially boot an application and to clear a table with 10,000 rows. On average, React is in the middle between Angular and Vue.js for most benchmarks.

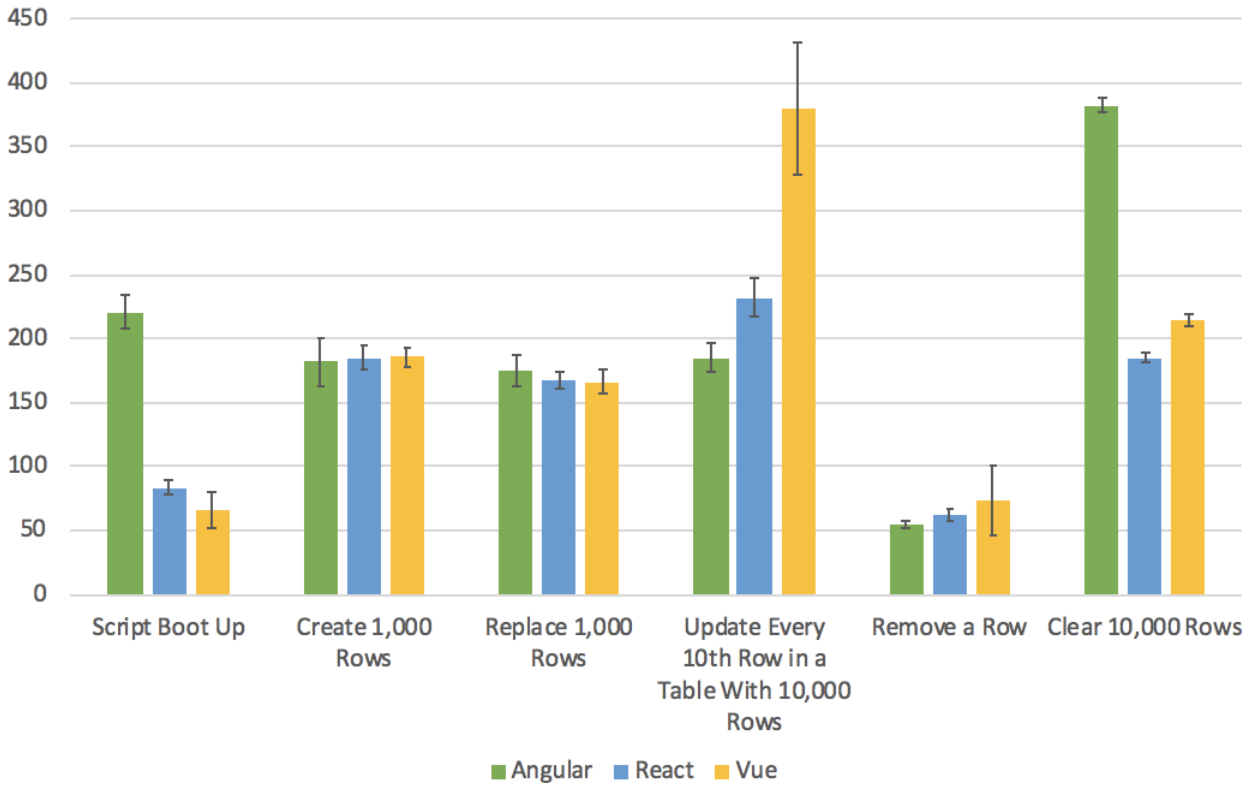

Figure 6.1: Overview of Six Benchmarks Performed on Angular, React and Vue.js in ms with a 95% Confidence Interval

6.6 Stability

Software Releases Angular, React and Vue.js follow semantic versioning principles (see Section 4.2.6). Hence, all three frameworks indicate which releases include breaking API changes and require developer assistance.

Major versions for Angular are released every six months. For each major release, there are about one to three minor releases. Patch releases come out almost weekly. React does not have a fixed release schedule. The last two major versions were released in April 2016 and September 2017. The next major version is supposed to come out sometime in 2019. Vue.js also does not provide a release schedule for major versions, but states to announce breaking API changes at least six months before the release date. The past two major versions of Vue.js have been released in October 2015 and September 2016. Minor releases come out every three months.

In conclusion, Angular is the only framework with a fixed release schedule. It has the most frequent breaking API changes, judging by past releases of the three frameworks. React's and Vue.js' past releases have been at odd times, so it is difficult to compare them with each other regarding the frequency of API changes.

7 Conclusion and Outlook

This chapter summarizes the results of the comparison and gives suggestions for a few use cases. Also, an outlook is given and future work that arises from this study is discussed.

7.1 Conclusion

Three popular technologies, Angular, React and Vue.js, were compared in this paper based on six criteria to help make educated decisions between JavaScript frameworks. The criteria were not prioritized because system requirements differ for each use case. Consequently, this paper does provide a total metric. However, if the individual needs of a use case are outlined, a prioritization can be defined for the criteria and it is possible to create an overall metric.

Similar to the publications mentioned in Chapter 2, this paper comes to the conclusion that no framework is the clear winner in all aspects. The frameworks are similar in the criterion ease of development. All of them have detailed documentation and a similar ratio of unanswered to answered questions on Stack Overflow. Also, advanced debugging tools are available for all three frameworks, but React and Vue.js' debugging tools are more cross-browser compatible than Angular's. In the remaining criteria, more significant differences could be determined.

At the moment, React is the most popular, Angular the second most popular and Vue.js the least popular framework. Vue.js has the smallest learning curve, followed by React and Angular has the steepest learning curve. React and Vue.js depend on the use of separate state management libraries for complex large-scale applications. Angular offers a complete state management solution that does not require the use of a separate library. Angular enforces a clear project structure with NgModules, whereas React and Vue.js are more flexible and leave design choices regarding the file organization of a project to the developer. The concept of modularity is implemented by React

and Vue.js through a component-based approach. Angular groups related components further in NgModules and, therefore, has an additional instance of modularity.

Vue.js has the shortest initial page loading time and Angular has performed best regarding most table operations that were measured. Angular's initial page loading time is significantly longer than React's and Vue.js'. React has been in the middle between both other frameworks for most performance measurements.

Angular is the only framework that has a fixed release schedule. It has the most frequent breaking API changes of the three frameworks, judging by past events. React and Vue.js have no fixed release schedules. In the past, the times between the releases of major versions for the two frameworks have fluctuated between eleven months and three years.

From these results, several recommendations can be derived:

- For simple applications that do not share data across many components, Vue.js and React can be recommended.
- For large and complex applications with many components that share data, Angular can be recommended because it enforces a maintainable project structure and comes with a complete built-in state management solution. React and Vue.js can also be used for complex applications but extra libraries are needed for state management.
- For the creation of SPAs, Angular is recommended if the use of many third-party libraries is not wanted.
- If only a small part of an application is supposed to be controlled by a framework, Vue.js is a good choice because it can be used only on an isolated part of a user interface.
- If flexibility is preferred over a given project structure, React and Vue.js are suitable choices.
- If low maintenance costs are an important factor for a project, Vue.js and React can be good choices based on their infrequent API updates (provided that the frequency of major releases for both frameworks does not change significantly).
- For developers that do not want to learn any languages on top of HTML, CSS and JavaScript, Vue.js is a good option.

- If a developer has experience with TypeScript, Angular can be a good choice.
- If browsers other than Google Chrome or Firebox are used in the development process, React and Vue.js are better options than Angular regarding debugging capabilities.

It needs to be noted that some criteria considered in this paper are not clearly distinct from each other. For example, the quality of a framework's documentation has an effect on the number of questions that are asked and answered on Stack Overflow. Also, the size of a framework has an impact on its script boot up time. Concerning the performance tests, it needs to be considered that they were conducted on a simple application and that results may be different if the benchmarks were performed on a more complex application.

7.2 Outlook and Future Work

JavaScript frameworks are constantly adapting and improving. Therefore, with the release of upcoming versions of the investigated frameworks, this paper may be outdated. Especially the popularity of frameworks changes frequently. Therefore, the investigation and comparison of JavaScript frameworks is a continuous process that does not come to a final, never changing result.

Regarding the criterion ease of development, a comparison based on a literature review and Stack Overflow statistics could not determine significant differences between the frameworks. For future work, a controlled experiment with three developer groups that have a similar level of knowledge and experience could determine differences more precisely. Each group could build the same application with a different framework. It could then be measured how many times the developers search for information outside of official the documentation and how much time is spent for solving errors with debugging tools.

It could be interesting to investigate the three frameworks based on more complex reference applications than this paper does. Also, the investigation of the frameworks, particularly with regard to the development of SPA, could bring up useful findings.

A Appendix

A.1 User Interface of a JS-Framework-Benchmark Reference Application

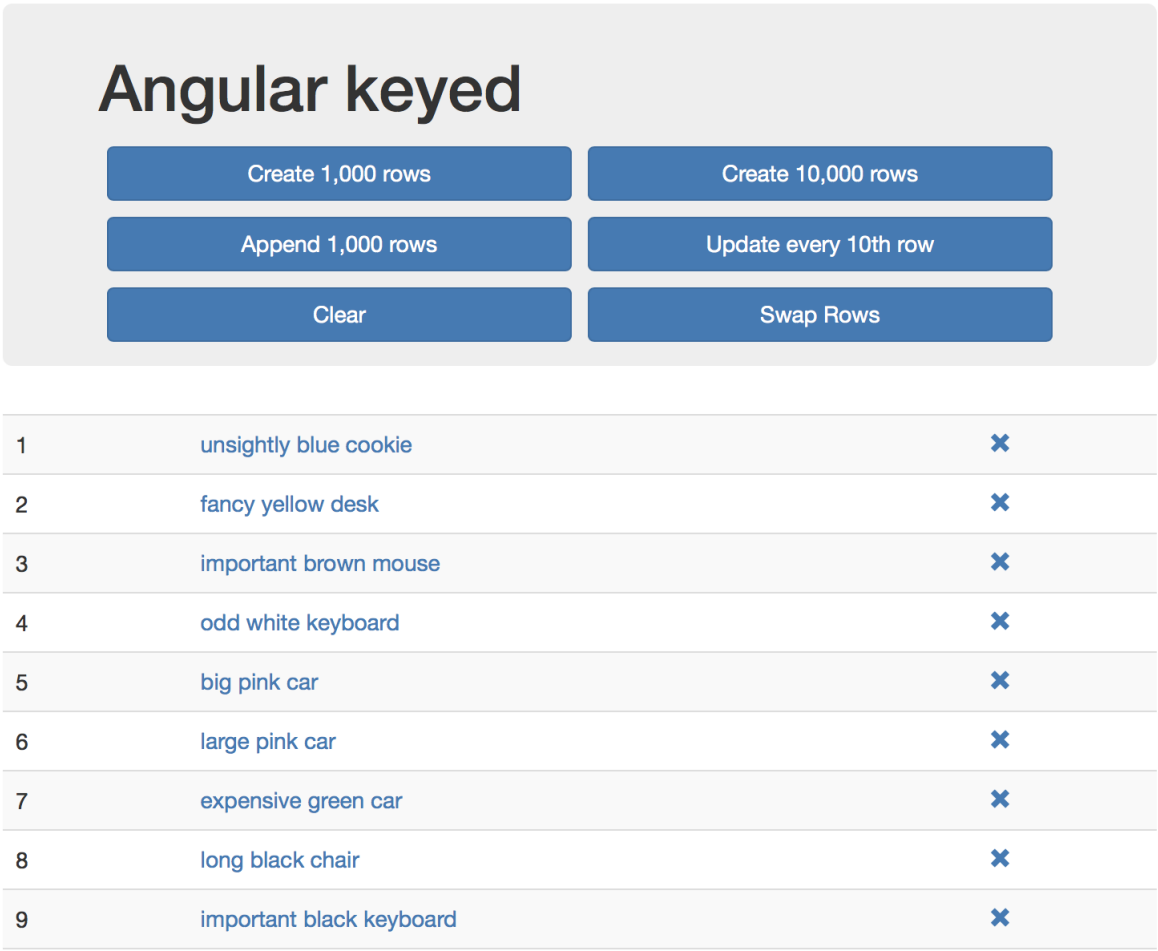

Figure A.1: User Interface of a JS-Framework-Benchmark Reference Application

Bibliography

[1] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar. "Characterizing Web Page Complexity and Its Impact". In: *IEEE/ACM Transactions on Networking* 22.3 (June 2014). ISSN: 1063-6692. DOI: 10.1109/TNET.2013.2269999. URL: https://ieeexplore.ieee.org/iel7/90/4359146/06557094.pdf.

[2] W3Techs. *Usage statistics of JavaScript as client-side programming language on websites*. 2019. URL: https://w3techs.com/technologies/details/cp-javascript/all/all (visited on 07/26/2019).

[3] Stack Overflow. *Welcome to Stack Overflow*. 2019. URL: https://stackoverflow.com/tour (visited on 05/28/2019).

[4] John Hannah. *The Ultimate Guide to JavaScript Frameworks*. 2018. URL: https://jsreport.io/the-ultimate-guide-to-javascript-frameworks/ (visited on 05/24/2019).

[5] Stack Overflow. *Stack Overflow Developer Survey 2019*. 2019. URL: https://insights.stackoverflow.com/survey/2019 (visited on 05/28/2019).

[6] Luke Joliat. *Do we still need JavaScript frameworks?* 2019. URL: https://www.freecodecamp.org/news/do-we-still-need-javascript-frameworks-42576735949b/ (visited on 07/26/2019).

[7] Joe Lennon. *Compare JavaScript frameworks*. 2010. URL: https://www.ibm.com/developerworks/library/wa-jsframeworks/wa-jsframeworks-pdf.pdf (visited on 05/25/2019).

[8] Amantia Pano, Daniel Graziotin, and Pekka Abrahamsson. "Factors and actors leading to the adoption of a JavaScript framework". In: *Empirical Software Engineering (2018)* 23 (2018), pp. 3503–3534. DOI: https://doi.org/10.1007/s10664-018-9613-x. URL: https://link.springer.com/article/10.1007/s10664-018-9613-x#citeas.

[9] Andreas Gizas, Sotiris Christodoulou, and Theodore Papatheodorou. "Comparative evaluation of JavaScript frameworks". In: *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion* (2012). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.309.7737&rep=rep1&type=pdf (visited on 07/21/2019).

[10] Daniel Graziotin and Pekka Abrahamsson. "Making Sense out of a Jungle of JavaScript Frameworks: towards a Practitioner-friendly Comparative Analysis". In: Lecture Notes in Computer Science 7983 (2013). 14th International Conference on Product-Focused Software Process Improvement (PROFES). DOI: https://doi.org/10.1007/978-3-642-39259-7_28. URL: https://link.springer.com/chapter/10.1007/978-3-642-39259-7_28.

[11] Eric Wohlgethan. "Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js". Bachelor's Thesis. Hochschule für Angewandte Wissenschaften Hamburg, 2018. URL: http://edoc.sub.uni-hamburg.de/haw/volltexte/2018/4350/ (visited on 07/21/2019).

[12] FusionCharts. *Top JavaScript Frontend Frameworks Comparison in 2018*. 2018. URL: https://www.fusioncharts.com/resources/developers/js-frontend-frameworks-comparison (visited on 05/26/2019).

[13] Shaumik Daityari. *Angular vs React vs Vue: Which Framework to Choose in 2019*. 2019. URL: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/ (visited on 05/26/2019).

[14] Brian Terlson et al. *Standard ECMA-262 - ECMAScript Language Specification*. Ecma International. 2019. URL: https://www.ecma-international.org/publications/standards/Ecma-262.htm (visited on 05/20/2019).

[15] Jonathan Reid and Thomas Valentine. *JavaScript Programmer's Reference*. Apress, 2013. DOI: https://doi.org/10.1007/978-1-4302-4630-5.

[16] W3schools. *Browser Statistics*. 2019. URL: https://www.w3schools.com/browsers/ (visited on 05/24/2019).

[17] W3Schools. *JavaScript Versions*. 2019. URL: https://www.w3schools.com/js/js_versions.asp (visited on 05/20/2019).

[18] Nodejs. *Node.js v12.5.0 Documentation - ECMAScript Modules*. 2019. URL: https://nodejs.org/api/esm.html#esm_ecmascript_modules (visited on 06/29/2019).

[19] Google. *TypeScript Configuration*. 2019. URL: https://angular.io/guide/typescript-configuration (visited on 05/21/2019).

[20] Mark Clow. *Angular 5 Projects: Learn to Build Single Page Web Applications Using 70+ Projects*. Apress, 2018.

[21] Nate Murray et al. *Angular 7: TypeScript*. n.d. URL: https://www.ng-book.com/2/p/TypeScript/ (visited on 05/21/2019).

[22] Cássio de Sousa Antonio. *Pro React*. Apress, 2015.

[23] Sharpened Productions. *CamelCase Definition*. n.d. URL: https://techterms.com/definition/camelcase (visited on 05/24/2019).

[24] Facebook. *Introducing JSX – React*. 2019. URL: https://reactjs.org/docs/introducing-jsx.html (visited on 05/21/2019).

[25] Brandon Wozniewicz. *The Difference Between a Framework and a Library*. 2019. URL: https://medium.freecodecamp.org/the-difference-between-a-framework-and-a-library-bd133054023f (visited on 05/23/2019).

[26] Program Creek. *Library vs. Framework?* 2011. URL: https://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/ (visited on 05/23/2019).

[27] Techopedia. *Server*. n.d. URL: https://www.techopedia.com/definition/2282/server (visited on 07/04/2019).

[28] Olga Filipova and Rui Vilão. *Software Development From A to Z: A Deep Dive into all the Roles Involved in the Creation of Software*. Apress, 2018. DOI: https://doi.org/10.1007/978-1-4842-3945-2.

[29] Eleasah Halsmer et al. *Introduction to the DOM*. 2019. URL: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (visited on 05/23/2019).

[30] Gil Fink and Ido Flatow. *Pro Single Page Application Development: Using Backbone.js and ASP.NET*. Apress, 2014. DOI: https://doi.org/10.1007/978-1-4302-6674-7.

[31] Joe Hanson. *What is HTTP Long Polling?* 2014. URL: https://www.pubnub.com/blog/2014-12-01-http-long-polling/ (visited on 05/23/2019).

[32] W3schools. *What is AJAX?* n.d. URL: https://www.w3schools.com/whatis/whatis_ajax.asp (visited on 05/24/2019).

[33] W3schools. *JSON - Introduction.* n.d. URL: https://www.w3schools.com/js/js_json_intro.asp (visited on 06/17/2019).

[34] Mike Wasson. *ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET.* 2013. URL: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx (visited on 05/24/2019).

[35] Maximilian Schwarzmüller. *React.js vs Angular vs Vue.* 2018. URL: https://www.academind.com/learn/angular/angular-vs-react-vs-vue-my-thoughts/ (visited on 07/05/2019).

[36] The Open Preservation Foundation. *Software Maturity.* n.d. URL: https://openpreservation.org/technology/principles/software-maturity/ (visited on 05/31/2019).

[37] David Robinson. *Introducing Stack Overflow Trends.* 2017. URL: https://stackoverflow.blog/2017/05/09/introducing-stack-overflow-trends/ (visited on 05/09/2019).

[38] W3schools. *What is npm?* n.d. URL: https://www.w3schools.com/whatis/whatis_npm.asp (visited on 05/28/2019).

[39] Indeed. *About Indeed.* 2019. URL: https://www.indeed.com/about (visited on 05/28/2019).

[40] Techopedia. *Debugging.* n.d. URL: https://www.techopedia.com/definition/16373/debugging (visited on 07/04/2019).

[41] Techopedia. *Scalability.* n.d. URL: https://www.techopedia.com/definition/9269/scalability (visited on 05/31/2019).

[42] Kate Swanberg. *Why Scalability Matters for Your App.* 2018. URL: https://www.koombea.com/blog/why-scalability-matters-for-your-app/ (visited on 05/31/2019).

[43] Talysson de Oliveira. *Scalable Frontend Architecture Fundamentals.* 2018. URL: https://blog.codeminer42.com/scalable-frontend-1-architecture-9b80a16b8ec7 (visited on 05/31/2019).

[44] Sander Mak. *Modules vs. microservices.* 2017. URL: https://www.oreilly.com/ideas/modules-vs-microservices (visited on 05/31/2019).

[45] Edwin Toonen. *Does site speed influence SEO?* 2018. URL: https://yoast.com/does-site-speed-influence-seo/ (visited on 05/30/2019).

[46] Jakob Nielson. *Website Response Times*. 2010. URL: https://www.nngroup.com/articles/website-response-times/ (visited on 05/30/2019).

[47] Ohans Emmanuel. *How browser rendering works behind the scenes*. 2018. URL: https://blog.logrocket.com/how-browser-rendering-works-behind-the-scenes-6782b0e8fb10 (visited on 05/30/2019).

[48] Paul Lewis. *Rendering Performance*. 2019. URL: https://developers.google.com/web/fundamentals/performance/rendering/ (visited on 05/30/2019).

[49] International Organization for Standardization (ISO). *ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model*. International Organization for Standardization (ISO). 2001. URL: https://www.iso.org/standard/22749.html (visited on 07/22/2019).

[50] Margaret Rouse. *Backward Compatible (Backward Compatibility)*. 2005. URL: https://whatis.techtarget.com/definition/backward-compatible-backward-compatibility (visited on 06/09/2019).

[51] Tom Preston-Werner. *Semantic Versioning 2.0*. 2019. URL: https://semver.org (visited on 07/17/2019).

[52] Google. *Architecture Overview*. 2019. URL: https://angular.io/guide/architecture (visited on 06/01/2019).

[53] Techopedia. *View (MVC)*. n.d. URL: https://www.techopedia.com/definition/27453/view-mvc-aspnet (visited on 08/06/2019).

[54] W3REIGN. *Angular 2 – Architecture*. n.d. URL: https://w3reign.com/tutorial/angular/angular-2-architecture-2/ (visited on 06/02/2019).

[55] Stack Overflow. *Stack Overflow Trends*. 2019. URL: https://insights.stackoverflow.com/trends?tags=angular%2Cvue.js%2Creactjs (visited on 06/09/2019).

[56] Paul Vorbach. *npm-stat*. 2019. URL: https://npm-stat.com/charts.html?package=angular&package=react&package=vue&from=01%2F01%2F2019&to=06%2F09%2F2019 (visited on 06/09/2019).

[57] Indeed. *Angular Jobs*. 2019. URL: https://www.indeed.com/q-angular-jobs.html (visited on 06/10/2019).

[58] Mosh Hamedani. *React vs. Angular: The Complete Comparison*. 2018. URL: https://programmingwithmosh.com/react/react-vs-angular/ (visited on 06/14/2019).

[59] Google. *The RxJS library*. 2019. URL: https://angular.io/guide/rx-library (visited on 06/04/2019).

[60] AltexSoft. *The Good and the Bad of Angular Development*. 2018. URL: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/ (visited on 06/04/2019).

[61] Ankit Kumar. *React vs. Angular vs. Vue.js: A Complete Comparison Guide*. 2018. URL: https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu (visited on 07/05/2019).

[62] TechMagic. *React vs Angular vs Vue.js - What to choose in 2019? (updated)*. 2018. URL: https://medium.com/@TechMagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d (visited on 07/05/2019).

[63] Abhay Srivastav. *Benchmarking Angular, React and Vue for small web applications*. 2019. URL: https://blog.bitsrc.io/benchmarking-angular-react-and-vue-for-small-web-applications-e3cbd62d6565 (visited on 07/05/2019).

[64] Stack Overflow. *Questions tagged Angular*. 2019. URL: https://stackoverflow.com/questions/tagged/angular (visited on 06/12/2019).

[65] Jonas Bandi et al. *Developer Tools for Angular*. 2018. URL: https://github.com/angular/angular/blob/master/docs/TOOLS.md (visited on 06/04/2019).

[66] Google. *JsonPipe*. 2019. URL: https://angular.io/api/common/JsonPipe (visited on 06/04/2019).

[67] Google. *Getting Started*. 2019. URL: https://augury.rangle.io/pages/guides/index.html (visited on 06/05/2019).

[68] Vamsi Vempati. *A Guide To Debugging Angular Applications*. 2018. URL: https://medium.com/front-end-weekly/a-guide-to-debugging-angular-applications-5a36bd88b4cf (visited on 06/04/2019).

[69] Jeff Delaney. *Sharing Data Between Angular Components - Four Methods*. 2017. URL: https://angularfirebase.com/lessons/sharing-data-between-angular-components-four-methods/ (visited on 06/05/2019).

[70] Google. *Component Interaction*. 2019. URL: https://angular.io/guide/component-interaction (visited on 06/05/2019).

[71] John Crowson, Brandon Roberts, and Itay Oded. *What is NgRx?* 2019. URL: https://ngrx.io/docs (visited on 06/05/2019).

[72] Google. *Workspace and project file structure*. 2019. URL: https://angular.io/guide/file-structure (visited on 06/05/2019).

[73] Google. *NgModules*. 2019. URL: https://angular.io/guide/feature-modules (visited on 06/05/2019).

[74] Google. *Feature Modules*. 2019. URL: https://angular.io/guide/ngmodules (visited on 06/05/2019).

[75] Google. *Introduction to components*. 2019. URL: https://angular.io/guide/architecture-components (visited on 06/05/2019).

[76] Deepika Chhabra. *Angular 101 - A technical Guide to Basic UI Design with Angular*. 2018. URL: https://medium.com/formcept/angular-101-a-technical-guide-to-basic-ui-design-with-angular-605fb0090ae3 (visited on 06/24/2019).

[77] Google. *Angular Versioning and Releases*. 2019. URL: https://angular.io/guide/releases (visited on 06/05/2019).

[78] Cory Gackenheimer. *Introduction to React*. Apress, 2015. DOI: https://doi.org/10.1007/978-1-4842-1245-5.

[79] Facebook. *React - A JavaScript library for building user interfaces*. 2019. URL: https://reactjs.org (visited on 06/10/2019).

[80] Facebook. *Virtual DOM and Internals*. 2019. URL: https://reactjs.org/docs/faq-internals.html#what-is-the-virtual-dom (visited on 06/14/2019).

[81] Facebook. *Thinking in React - Step 4: Identify Where Your State Should Live*. 2019. URL: https://reactjs.org/docs/thinking-in-react.html (visited on 06/14/2019).

[82] Marta Kavrukova. *Is React's data-binding really one way? It seems like it is two way*. 2016. URL: https://stackoverflow.com/questions/32712605/is-reacts-data-binding-really-one-way-it-seems-like-it-is-two-way (visited on 06/14/2019).

[83] Indeed. *React Jobs*. 2019. URL: https://www.indeed.com/jobs?q=react&l= (visited on 06/10/2019).

[84] Stack Overflow. *Questions tagged React.js*. 2019. URL: https://stackoverflow.com/questions/tagged/reactjs (visited on 06/12/2019).

[85] Facebook. *React Developer Tools*. 2019. URL: https://github.com/facebook/react-devtools (visited on 06/14/2019).

[86] Facebook. *State and Lifecycle*. 2019. URL: https://reactjs.org/docs/state-and-lifecycle.html#the-data-flows-down (visited on 06/16/2019).

[87] Facebook. *Components and Props*. 2019. URL: https://reactjs.org/docs/components-and-props.html (visited on 06/16/2019).

[88] Facebook. *Lifting State Up*. 2019. URL: https://reactjs.org/docs/lifting-state-up.html (visited on 06/16/2019).

[89] Facebook. *Component State*. 2019. URL: https://reactjs.org/docs/faq-state.html (visited on 06/20/2019).

[90] Dan Abramov et al. *Three Principles*. 2018. URL: https://redux.js.org/introduction/three-principles (visited on 06/20/2019).

[91] Jonathan Saring. *State of React Management for 2019*. 2018. URL: https://blog.bitsrc.io/state-of-react-state-management-in-2019-779647206bbc (visited on 06/20/2019).

[92] Dan Abramov. *Folder Structure*. 2018. URL: https://facebook.github.io/create-react-app/docs/folder-structure (visited on 06/27/2019).

[93] Facebook. *File Structure*. 2019. URL: https://reactjs.org/docs/faq-structure.html (visited on 06/20/2019).

[94] Facebook. *Versioning Policy*. 2019. URL: https://reactjs.org/docs/faq-versioning.html (visited on 06/20/2019).

[95] Evan You et al. *Introduction*. 2019. URL: https://vuejs.org/v2/guide/ (visited on 06/24/2019).

[96] Evan You et al. *Vue.js 0.10.6, and what's next*. 2014. URL: https://vuejs.org/2014/07/29/vue-next/ (visited on 06/24/2019).

[97] Evan You et al. *Installation - Release Notes*. 2019. URL: https://vuejs.org/v2/guide/installation.html#Release-Notes (visited on 06/24/2019).

[98] Evan You et al. *The Vue Instance*. 2019. URL: https://vuejs.org/v2/guide/instance.html (visited on 06/24/2019).

[99] Evan You et al. *Components Basics*. 2019. URL: https://vuejs.org/v2/guide/components.html (visited on 06/24/2019).

[100] Evan You et al. *Template Syntax - Interpolations*. 2019. URL: https://vuejs.org/v2/guide/syntax.html#Interpolations (visited on 06/25/2019).

[101] Evan You et al. *Directives*. 2019. URL: https://vuejs.org/v2/guide/syntax.html#Directives (visited on 06/24/2019).

[102] Evan You et al. *Form Input Bindings*. 2019. URL: https://vuejs.org/v2/guide/forms.html#v-model-with-Components (visited on 06/24/2019).

[103] Indeed. *Vue Jobs*. 2019. URL: https://www.indeed.com/q-vue-jobs.html (visited on 06/10/2019).

[104] Mosh Hamedani. *React vs. Vue - A Wholesome Comparison*. 2018. URL: https://programmingwithmosh.com/javascript/react-vs-vue-a-wholesome-comparison/ (visited on 07/05/2019).

[105] Stack Overflow. *Questions tagged Vue.js*. 2019. URL: https://stackoverflow.com/questions/tagged/vue.js (visited on 06/12/2019).

[106] Evan You et al. *Vue-devtools*. 2019. URL: https://github.com/vuejs/vue-devtools (visited on 06/26/2019).

[107] Kenneth Auchenberg. *Debugging in VS Code*. 2019. URL: https://vuejs.org/v2/cookbook/debugging-in-vscode.html#ad (visited on 06/25/2019).

[108] Derick Sozo. *Speed Up Development with Vue DevTools*. 2018. URL: https://medium.com/vue-mastery/how-to-use-the-vue-devtools-af95191ff472 (visited on 06/25/2019).

[109] Evan You et al. *Migration from Vue 1.x - Events*. 2019. URL: https://vuejs.org/v2/guide/migration.html#dispatch-and-broadcast-replaced (visited on 06/26/2019).

[110] Evan You et al. *Style Guide - Priority D Rules*. 2019. URL: `https://vuejs.org/v2/style-guide/#Non-flux-state-management-use-with-caution` (visited on 06/26/2019).

[111] Evan You et al. *Getting Started*. 2019. URL: `https://vuex.vuejs.org/guide/` (visited on 06/26/2019).

[112] Evan You et al. *Creating a Project*. 2019. URL: `https://cli.vuejs.org/guide/creating-a-project.html#using-the-gui` (visited on 06/29/2019).

[113] Evan You et al. *Component Registration - Module Systems*. 2019. URL: `https://vuejs.org/v2/guide/components-registration.html#Module-Systems` (visited on 06/29/2019).

[114] Evan You et al. *Vue Project Roadmap - Release Management*. 2019. URL: `https://github.com/vuejs/roadmap` (visited on 06/29/2019).

[115] Evan You et al. *Releases*. 2019. URL: `https://github.com/vuejs/vue/releases` (visited on 07/11/2019).