



**KTH Computer Science
and Communication**

Comparison of Single-Page Application Frameworks

A method of how to compare Single-Page Application frameworks written in
JavaScript

ERIC MOLIN

KTH CSC Supervisor: Dilian Gurov
Principal Supervisor: Sven Norman
Examiner: Olov Engwall
May 29, 2016

Abstract

This degree project is carried out on behalf of Decerno, an IT consultancy company. The focus is to formulate a method of how to compare Single-Page Application (SPA) frameworks written in JavaScript.

This method is based upon an abstraction of SPA frameworks. The abstraction is done with a criteria-based approach. These criteria are collected from literature and interviews conducted at Decerno. Every criterion is defined and has a set of questions, which corresponds to the criterion. Concepts are extracted from other comparative methods, such as performance test and code comparison. This concluded in a theoretical part with the criteria and questions and a practical part with performance tests and code comparison.

Finally, the method is tested on three different Frameworks, AngularJS, Angular 2 and React. By using a prototype implemented in the three frameworks, a code comparison and performance tests is conducted. According to the method, AngularJS is suggested to be the best choice. However, some results are similar and for future work, this method could be evaluated to other comparative methods or be extended with more criteria and questions.

Referat

Jämförelse av Ramverk För Single-Page Applikationer

Detta examensarbete har utförts på uppdrag av Decerno, ett IT-konsultföretag. Fokus är att formulera en metod för hur man kan jämföra Single-Page Application (SPA) ramverk skrivna i JavaScript.

Denna metod är baserad på en abstraktion av SPA ramverk. Abstraktionen har ett kriteriebaserat tillvägagångssätt. Dessa kriterier är hämtade från litteraturen och intervjuer på Decerno. Varje kriterie definieras och har ett antal frågor som motsvarar kriteriet. Koncept extraieras från andra komparativa metoder, såsom prestandatest och kodjämförelse. Detta delades in i en teoretisk del med kriterierna och frågorna samt en praktisk del med prestandatester samt kodjämförelse.

Slutligen testas metoden på tre olika ramverk, AngularJS, Angular 2 och React. Genom att använda en prototype som är implementerad i de tre ramverken, görs en kodjämförelse samt prestandatester. I enlighet med metoden, föreslås AngularJS att vara det bästa valet. Men vissa resultat är likartade och i framtiden skulle denna metod kunna utökas med flera kriterier eller utvärderas med andra jämförbara metoder.

Contents

1	Introduction	1
1.1	Background	1
1.2	The Problem	2
1.2.1	Problem Description	2
1.2.2	Research Questions	2
1.2.3	Delimitation	2
1.3	Approach	2
1.4	Contribution	3
1.5	Thesis structure	3
2	Background	5
2.1	JavaScript	5
2.1.1	ECMAScript	5
2.1.2	Additional JavaScript Languages	6
2.2	Single-Page Application	6
2.2.1	Definition of a Single-Page Application	7
2.2.2	Communication	7
2.2.3	Execution	8
2.2.4	Data bindings	8
2.2.5	States	9
2.3	Software Architecture	10
2.3.1	Definition of Software Architecture	10
2.3.2	Identify Software Architecture	11
2.4	Framework	11
2.4.1	Definition of a Framework	11
2.4.2	Purpose of a Framework	12
2.5	Related Work	12
2.5.1	Software Architecture Comparison	12
2.5.2	Framework Comparison	13
3	Methodology	14
3.1	Overview	14
3.2	Interviews	15

3.3	Performance Measuring Methodology	15
3.3.1	Data Bindings	16
3.3.2	Loading Time	16
3.3.3	Resource Allocation	16
3.4	Tools	16
3.5	Prototype	16
4	Formulating a Comparison Method	18
4.1	Results of the Interviews	18
4.2	Results of the Literature Research	18
4.3	Criteria Collection	20
4.4	Criteria Definition	21
4.4.1	Security	21
4.4.2	Modularity	22
4.4.3	Popularity	22
4.4.4	Maturity and Stability	23
4.4.5	Simplicity and Usability	23
4.4.6	Portability and Compatibility	24
4.4.7	Cache Performance and Persistence	24
4.4.8	Testability	25
4.4.9	Maintainability	25
5	Comparison of SPA Frameworks	27
5.1	Frameworks	27
5.1.1	AngularJS	27
5.1.2	React	27
5.1.3	Angular 2	28
5.2	Practical Comparison	28
5.2.1	Performance Results	28
5.2.2	Code comparison	31
5.3	Theoretical Comparison	34
5.4	Comparison Conclusions	36
6	Discussion and Conclusion	37
6.1	Summary	37
6.2	Discussion	37
6.3	Socio-ethical discussion	37
6.4	Future work	38
	Bibliography	39

Chapter 1

Introduction

The first chapter introduces the background to the thesis, a description of the problem and its delimitation. The problem is broken down into three research questions.

1.1 Background

Web applications have suffered from poor user interactivity for many years despite of their popularity [42]. During the evolution of the Web in the early 2000's, it evolves into what is referred to as Web 2.0. Among new techniques and concepts of Web 2.0 is AJAX (Asynchronous JavaScript and XML). This technique made it possible for web applications to asynchronously fetch new data and to update the web page, without refreshing it [44]. Later, this evolved into a new type of web application, known as SPA (Single-Page Application).

Developers often use one or many frameworks and libraries when developing large and complex web applications. They can reuse code and therefore spend more time on designing the current application by choosing an appropriate framework. On the contrary, a less suitable framework can affect the development time and reduce the quality of the application. Still, web framework comparison is not an establish discipline and the closest field is software architecture comparison. Software architecture comparison is also a young discipline and one of the most popular methods are SAAM (Software Architecture Analysis Method), originated from 1996 [21].

This project is performed at Decerno. It is a small IT consultancy company who has built custom systems since the 80's. Their focus is web development and because they make custom systems, they choose tools and framework as required for the particular project. Currently, all their customers demand systems built as an SPA. However, new framework is developed and older ones are majorly re-worked. Without any scientific methodology for comparing SPA frameworks written in JavaScript, it can be troublesome to choose which one to use.

1.2 The Problem



The purpose of this thesis project is to propose a method of how to compare SPA frameworks. This section gives a further definition of the problem.

1.2.1 Problem Description



A large number of frameworks exist and are constantly evolving. Besides, there is no definition of a framework or what to include. The complexity of SPA frameworks makes a comparison to other frameworks difficult. Thus, there are a lot to take into consideration when deciding what framework to use when developing SPAs. For example, there are differences in its data bindings, cache performance and loading time.

First, to be able to make a comparison it is necessary to establish a definition of what a framework is and what to include. Second, by making an abstraction of an SPA framework, it lowers the complexity of the comparison. This abstraction can provide a base to design a comparison method and also a base to investigate different criteria that may be included in a method to compare SPA frameworks.

1.2.2 Research Questions

To solve the problem of how to compare SPA frameworks, three research questions are proposed:

RQ1: What is a proper abstraction of an SPA framework?



RQ2: How can a method be formulated to compare SPA frameworks?

RQ3: Can this method be used to give a recommendation about what SPA framework to use?

1.2.3 Delimitation

The study is based on SPA frameworks written in JavaScript. Most of the SPAs have a back-end service as a database or similar. However, the focus is solely on the client side of the SPA, where the part of the SPA using the JavaScript framework is. The proposed method focus on SPAs of such size and complexity that it benefits from using a properly chosen framework.

1.3 Approach

To be able to answer RQ1 and RQ2, a literature study of software architecture comparison and framework comparison is conducted. In addition, ten interviews are made. To make an abstraction of an SPA, concepts and criteria are chosen by a quantitative and qualitative approach. To answer RQ3, three frameworks are chosen and the method suggested by RQ2 is used to compare these.

1.4. CONTRIBUTION

1.4 Contribution

This thesis project contributes to an approach for making an abstraction of SPA frameworks. Furthermore, how this abstraction can be used to formulate a scientific method of how to compare different SPA frameworks.

1.5 Thesis structure

Chapter 2: ~~This chapter~~ contains a brief history of JavaScript, the architecture of an SPA, what a framework is and the purpose of a framework. Then, a method of describing software architecture is presented.

Chapter 3: ~~This chapter~~ contains the methodology of this thesis.



Chapter 4: ~~This chapter~~ contains the results of the interview and the literature study. Next, a method is formulated with the abstraction of an SPA framework.

Chapter 5: ~~This chapter~~ contains a comparison between three SPA frameworks by using the method proposed in chapter 3 and 4.

Chapter 6: ~~This chapter~~ contains a discussion about the thesis and a summary. Also, this chapter includes a socio-ethical discussion about the research.

Web Related Glossary

View: The visual representation of the web application.

Model: The data models or objects that are used in the web application.

Template: A template of how the view should be represented with the data from the models.

DOM (Document Object Model): Representation of objects and structure within a HTML or XML document.

Business logic: The logic of the web application, regardless of what client is used.

Component: A DOM-node that represents a visual component in a web application.

JSON (JavaScript Object Notation): JSON is a text-based format to show data. An example of a JSON object can be found in listing 1.1. It contains a person object with the key value pairs of `firstName`, `lastName` and `age`. This is supposed to resemble a person named John Smith who is 25 years old.

```
1 {"person":  
2   {  
3     "firstName": "John",  
4     "lastName": "Smith",  
5     "age": 25  
6   }  
7 }
```

Listing 1.1. Example of a JSON object


Chapter 2

Background

Chapter 2 includes a brief summary of how JavaScript has developed. Furthermore, there is an introduction to SPAs, to software architecture and to frameworks. These are explained and how they are related to web development. In addition, related work to software architecture comparison and framework comparisons is presented.

2.1 JavaScript

2.1.1 ECMAScript

 While the Web 1.0 era, the development of a programming language that comes to be known as JavaScript begin. In 1994, a company called Netscape, creates Netscape Navigator, a browser widely used during the 90's. In short, the communication with the web where static during this time and therefore, the company decides to develop a more dynamic browser [48].

For this reason, Netscape hired Brendan Eich in 1995 to implement a Lisp dialect called Schema in Netscape Navigator. Before Eich begins, Netscape collaborates with Sun, a software and hardware company, later bought by Oracle. Sun wanted to include their language Java in Netscape Navigator, thus making their browser need two separate languages. At that time Netscape decides that their programming language needs a similar syntax to Java, and then ruled out Scheme and other languages as Perl, TCL and Python. Eich completed a new programming language called Mocha in ten days, in 1995. Then Netscape changed the name to LiveScript and implemented the language in Netscape Navigator the same year. The use of Java increases and is widely spread during this time of period. According to this, Netscape changes name of the language from LiveScript to JavaScript [48].

In 1996, shortly after JavaScript was launched Microsoft implemented their version of JavaScript in their browser Internet Explorer 3.0. To be able to reduce competition from Microsoft, Netscape decides to standardize JavaScript and turns to the standard organization ECMA. They start to work with the specification ECMA-262, however, they are not allowed to use the trademark JavaScript so the name ECMAScript is chosen. During the years 1997 to 1999 ECMAScript 1-3 are



released. The Technical Committee 39, includes Microsoft, Google and Mozilla discard their work with ECMAScript 4 because the companies could not agree on a set of features for ECMAScript 4. In July 2008 the Technical Committee 39 meet again and agree to drop some features and that all future changes to ECMAScript should be less radical. Instead of releasing ECMAScript 4, it was renamed ECMAScript 5 and releases in December 2009. The most current version ECMAScript 6, is launched in June 2015 [49].

2.1.2 Additional JavaScript Languages

More syntaxes and languages exist that can be compiled to JavaScript, for example, CoffeeScript, TypeScript, JSX and Dart. These extend JavaScript with new functionality or another syntax that is easier to read or easier to use. In listing 2.1, is an example of a class called HelloWorld (with the function `getHelloWorld`) written in TypeScript presented.

```
1 class HelloWorld {
2     getHelloWorld() {
3         return "Hello, world";
4     }
5 }
```

Listing 2.1. HelloWorld class written in TypeScript

However, a web browser cannot interpret TypeScript. On the contrary, by compiling TypeScript into JavaScript, the browser is able to execute the code. Listing 2.2 shows the compilation of the HelloWorld class from listing 2.1.

```
1 var HelloWorld = (function () {
2     function HelloWorld() {
3     }
4     HelloWorld.prototype.getHelloWorld = function () {
5         return "Hello, world";
6     };
7     return HelloWorld;
8 }());
```

Listing 2.2. HelloWorld class compiled into JavaScript

In this example, TypeScript gives the HelloWorld class a C++ or Java-like syntax and removes the need for nested functions.

2.2 Single-Page Application

In this section, the fundamentals of SPAs are described by introducing how they communicate with a server, how SPAs are executed in the browser and how data bindings work.

2.2. SINGLE-PAGE APPLICATION

2.2.1 Definition of a Single-Page Application

It is generally agreed that there is no exact definition of the concept SPA. A. Mesbah and A. van Deursen define it as: “the single-page web interface is composed of individual components which can be updated/replaced independently, so that the entire page does not need to be reloaded on each user action” [42]. To achieve a better understanding, these key attributes are defined further:

Web interface: The interaction between a user and a web server.

Individual components: The SPA is split into smaller and individual components.

Updated/replaced: A component can be updated or replaced by a new component or page.

Reloaded: A typical web page needs to be reloaded, in contrast to an SPA.

User action: A user can make input to an SPA from any I/O device, which causes some action to occur.

2.2.2 Communication

Figure 2.1 illustrates a typical communication of an SPA between a user and a web server.

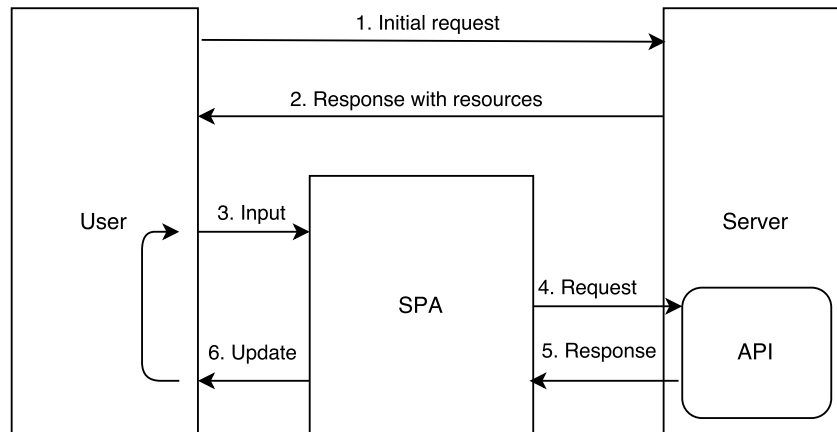


Figure 2.1. Figure of the communication between a user and a web server.

1. Initial request: The initial request is often made from a browser on a desktop computer or mobile device. This is done by a HTTP request from the user to a specific URL.

- 2. Response with resources:** The HTTP request is handled by the web server. The web server responds by sending the JavaScript dependencies and additional libraries. When the user receives the response from the server, the SPA is executed and loaded into the web browser.
- 3. Input:** The user is now able to make input to the SPA that causes changes in the state of the SPA. These changes are handled either by the application or by fetching new data via AJAX from the server API.
- 4. Request:** The request from the SPA to the API is made asynchronously. The communication between the SPA and the server is typically made with JSON.
- 5. Response:** The results are sent back to the SPA as soon as the API can handle the request. Mostly, this is done by sending JSON back to the SPA.
- 6. Update:** With the new data received from the server, the SPA updates the components. This update is done by a re-rendering of the DOM, performed with JavaScript. When the re-render is done, the SPA is ready for new input from the user. This results in a loop from the third step to the sixth step until the user exits the SPA.

2.2.3 Execution

JavaScript is executed in the browser and the way it is interpreted may vary between different browsers. The reason behind this is that the JavaScript engine and its implementation differs. Mozilla FireFox has an engine called SpiderMonkey [43]. Google Chrome for desktop computers got V8 [30]. Apple uses JavaScriptCore in their WebKit [22] browser engine for its desktop and mobile browsers. In other words, both the performance and support for newer functionality in JavaScript can differ between browsers. Consequently, it is important to test the SPA framework in as many browsers as possible.

2.2.4 Data bindings

There are two different types of data bindings. The one-way data binding is used in many traditional server-side web applications. A template and one or many models are merged on the server and sent to the user's view. Any changes to the models or the view are not reflected after the merge. To update the model, it is necessary for the user to send back the view to the server. These changes must be processed by the server and be sent back as a new merge of the template and the model [7].

The two-way data binding is common in many SPA frameworks. The view can be seen as a "single-source-of-truth" of the model. This makes all changes in the view instantly reflected on the model and all the changes to the model are propagated into the view [7]. However, since the two-way data binding is two ways, the reasoning about how the application can behave is difficult. The developer team at Facebook Inc. state: "We found that two-way data bindings led to cascading

2.2. SINGLE-PAGE APPLICATION

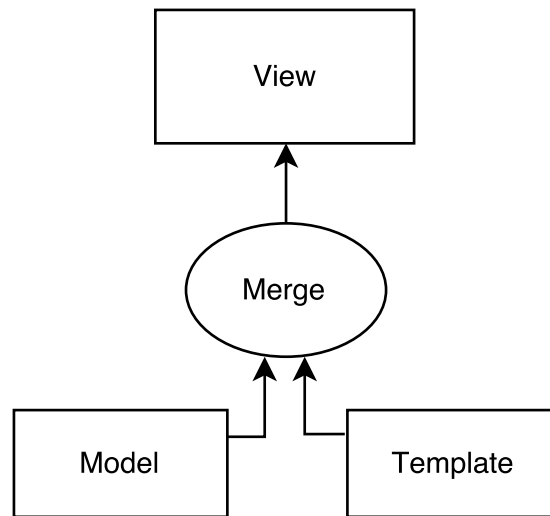


Figure 2.2. Figure of a one-way data binding

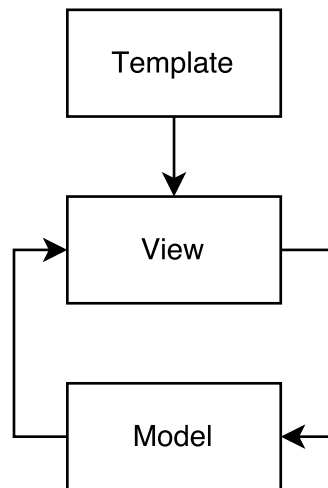


Figure 2.3. Figure of a two-way data binding

updates, where changing one object led to another object changing, which could also trigger more updates.” [23] This led to trouble with their Messenger application in the Facebook ecosystem, and to solve the problem with the two-way data binding they developed a new type of data binding [12]. It is called Flux and is a one-way data binding with four components instead of three [23].

2.2.5 States

States have always existed in web applications, but to increase the user interactivity, more states are required. In server-side rendering, it is difficult to implement smaller

changes in the components. These small changes also need a trip to retrieve HTML from the server. Typically, SPAs have more complex states than traditional server-side applications [58]:

DOM events that cause state changes: I/O into forms, the fields are validated and gives feedback to the user.

Application state changes: Interaction with buttons cause a new page to appear.

Global state changes: Going offline in a real time application.

Delayed data from the API: AJAX call is delayed between the SPA and the server.

Data model changes: A data model is changed and an update is sent to the client.

2.3 Software Architecture

This section defines what software architecture is. In addition, a method of how to identify a software architecture is proposed.

2.3.1 Definition of Software Architecture

Software architecture is a process of finding a structured solution that meets all operational and technical requirements of a software project. This involves optimizing common quality attributes, security, performance and a wide range of factors. Each of these decisions have impact on the quality, performance, maintainability and the overall success of the software [59].

A large variety of definitions of software architecture are found in literature. For example, in *Patterns of Enterprise Application Architecture*, Martin Fowler proposes: “The highest-level breakdown of a system into its parts; the decisions that are hard to change; there are multiple architectures in a system; what is architecturally significant can change over a system’s lifetime; and, in the end, architecture boils down to whatever the important stuff is.” [19].

In *Software Architecture in Practice (2nd edition)*, Bass, Clements, and Kazman define software architecture as: “The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. Architecture is concerned with the public side of interfaces; private details of elements – details having to do solely with internal implementation – are not architectural.” [8].

2.4. FRAMEWORK

2.3.2 Identify Software Architecture

There are great many methods of describing software architecture, such as 4+1, Agile modeling, IEEE 1471 and UML (Unified Model Language). The aim of all these methods is to give an idea of describing the software architecture in different type of views. These views can be seen as core parts of a system and all of these methods have their own idea of which views are the most important and how they should be presented [60].

The model that is used in this paper is proposed by Microsoft Research, *Application Architecture Guide, 2nd Edition*. This is an iterative method in five steps. This would help to put together the key decisions of a software architecture. An iterative process can make the developers able to refine the design over time and even through the life cycle of the software [60].

The five steps consist of:

- 1. Identify Architecture Objectives:** Give a precise specification of the objectives of the architecture. This helps with focusing on the right problems in the design.
- 2. Key Scenarios:** By identifying key scenarios, the focus is brought to what is most important for the architecture. These scenarios can be used to evaluate a candidate architecture.
- 3. Application Overview:** The application should reflect reality and its context must be identified. For example, application type, deployment architecture, architectural styles and technologies.
- 4. Key Issues:** Based on quality attributes the key issues must be identified. In this field, the most mistakes are made when designing an application.
- 5. Candidate Solutions:** Create an architecture prototype and evaluate it by the key scenarios, issues and deployment constraints. When this is done, one more iteration can be done for further refinement of the candidate architecture.

2.4 Framework

This section defines what a framework is and its purpose.

2.4.1 Definition of a Framework

In literature, there are many definitions of what a framework is. These frameworks are often referred to as software or application framework. Ralph E. Johnson states that a framework is a reusable design that is represented by a set of abstract classes. In addition, he calls a framework for a skeleton application that can be customized by a developer [36]. Besides of being called a skeleton application, a framework can be seen as a ‘semi-complete’ application [53].

2.4.2 Purpose of a Framework

Dirk Riehle discusses in his dissertation, *Framework Design*, that a developer who uses a framework, reuses its design. Furthermore, the developer is able to solve the problems that fall within the domain of the framework [50]. By reusing code the developer enhances the effectiveness and efficiency of innovation which results in higher quality product [35, p. 4].

2.5 Related Work

This section introduces different methods for software architecture comparison and framework comparison.

2.5.1 Software Architecture Comparison

SAAM is used to evaluate one candidate architecture at a time. This evaluation is performed via scenarios, quality attribute and quality metrics. The quality attributes and metrics are used to define the business value, while the scenario is used to understand the architecture. Different architectural approaches are identified and are evaluated using these scenarios, attributes and metrics [38].

ATAM is similar to SAAM, but this method focuses on finding tradeoffs. For every tradeoff, each architectural approach is evaluated of how this tradeoff can affect it [13].

FAAM (Family Architecture Analysis Method) can only evaluate information-system family architectures. System quality requirements are taken and evaluated against the proposed architectural approaches [16].

CBAM (Cost Benefit Analysis Method) is an enhancement of the ATAM. The authors propose ATAM with the addition of finding the benefit and cost for each of the proposed architectural approaches [33].

ALMA (Architecture Level Modifiability Analysis) is used to evaluate one architecture candidate. The method includes a set of scenarios and are evaluated of how well the architecture is supporting these scenarios [9].

SACAM (The Software Architecture Comparison Analysis Method) can compare different candidate architectures. This method extracts quality criteria from the business goals and describes them as scenarios. These scenarios are then evaluated given metrics and predefined architectural views [57].

DoSAM (Domain-specific Software Architecture Comparison Model) is used to compare different architectures given a specific domain. By collecting quality attributes, a domain specific architecture blueprint can be created. This is used as a schema for how the architecture is put together. Quality metrics are collected and the blueprints are evaluated given these metrics [10].

ARID (Active Reviews for Intermediate Designs) is a combination of design reviews and ATAM [39].

2.5. RELATED WORK

2.5.2 Framework Comparison

This section describes current research in the field of framework comparison.

Tim Malmström proposes a method of how to compare SPA frameworks written in JavaScript. He uses seven requirements taken from interviews. After defining the requirements, he discusses how the frameworks individually fulfill the requirements. Malmström also performs a performance test on two of the frameworks [41].

Anton Gerdessen in [20] proposes a method of comparing two Java back-end frameworks. He collects criteria from literature studies and removes the criteria that could not be applied to a web related framework. The criteria are sorted into two domains and Gerdessen creates a theoretical framework is used to compare the two frameworks. Then, he iterates through all the criteria and reviewed the two frameworks side by side. A more in depth analysis is performed with two of the chosen criteria on the two frameworks [20].

Joe Lennon proposes a method of comparing JavaScript frameworks. He uses a set of features and compares how these features is implemented and used within the frameworks. Furthermore, he develops a prototype using these frameworks and performs a code comparison between the prototypes [40].

Maria del Pilar Salas-Zarate et al. present a list of best practices for web development and use these for comparison of web frameworks. Furthermore, the authors implement a prototype using the Lift framework [15].

In [21], Ignacio Fernández-Villamor et al. propose a method of how to compare agile web frameworks. They define a “blueprint architecture” containing eight criteria important for a web framework. These criteria are defined using a set of questions that summarize the general features of an agile web framework. This results in a table where the authors use a percentage of how the framework fulfills the question. Furthermore, the authors choose to use weights on each question. For each framework and the strength and weaknesses are revealed by a final discussion [21].

In [55], T. C. Shan and W. W. Hua propose a list of design principles that all web frameworks should follow and a taxonomy is proposed of how to group different web frameworks written in Java [55].

Chapter 3

Methodology

Chapter 3 contains an overview of the methodology being used for the interviews, the literature study and the performance measuring. In addition, an explanation of the prototype and tools that is used to measure the performance of the SPA frameworks.

3.1 Overview

In this paper, a combination of quantitative and qualitative methods are used to develop a way to compare SPA frameworks. The following sections describe how the research has been carried out.



The first step, before proposing a method of how to compare SPA frameworks, is to conduct a literature study. This study is performed in order to find out whether similar studies have been carried out and find comparative methods used previously. Finally, the analysis of the methods used in the literature resulted in a first draft of a comparison method. This method consists of a theoretical and a practical comparison, which are extended upon further in the study.



Secondly, a more comprehensive literature study is conducted to provide a basis for an abstraction of SPA frameworks. This study formed the main source of data that were collected and sorted into different criteria. As suggested by [14, p. 88-90], key concepts and terms are extracted. However, the word *criterion* is used instead of *terms*. As a complement to the literature study, ten interviews are held with employees at Decerno, who encounter SPAs on a daily basis.



For each criterion, a set of questions are proposed. These questions are formulated after an analysis of general features existing in some of the current SPA frameworks. In addition, questions from other framework comparisons and interviews are selected and placed into the corresponding criteria.

Finally, a testing methodology is chosen and performance tests are performed on the prototypes.

3.2. INTERVIEWS

3.2 Interviews

Interviews are often used as a qualitative method where participants are able to elaborate on what they think or feel [17]. The aim of the interviews is to gather more information concerning which criteria are important when choosing an SPA framework, hence these interviews are held with a more quantitative approach. All participants hold different technical roles at Decerno, such as front-end developer, system developer or project manager. The reason for choosing employees with different roles is because all of them are engaged in work with SPAs on a daily basis, but not all of them are working full time with writing code. They are encouraged to speak freely and keywords from their statements are written down. However, notes are not taken on why  how. The participants could speak freely without any interruption until no more  criteria could be found [54].

An interview situation may contain personal information that might be exposed in the research. Some of these issues are explained by Michael Q. Patton in [46], such as consent and confidentiality. To achieve an ethical interview, the participants are introduced to the purpose of this thesis and told how their answers  used. All the participants in the interviews take part voluntarily and their answers are going to remain anonymous. After the interview, the notes are presented and the permission to use the data in the thesis is acquired. 

3.3 Performance Measuring Methodology

A general method of performance testing proposed by [34] is used. This method is used to obtain a structure of how to perform a performance test of a web application. The method consists of seven steps:

1. Identify test environment
2. Identify performance acceptance criteria
3. Plan and design tests
4. Configure test environment
5. Implement the test design
6. Execute the test
7. Analyze results, report and retest

Typically, it is the back-end service being tested in a web application. Instead, this paper covers the client side of an SPA. The crucial part of an SPA's performance is the data bindings, the loading time and equally important is resource allocation.

3.3.1 Data Bindings

Data bindings are a crucial part for web applications. Typically, a two-way data binding is used in SPA frameworks (section 2.2.4). These bindings can be tested by loading data and see how fast it is reflected in the view. Besides, the tests consist of updating the data bindings with new data. This test shows how quickly data bindings react to changes in the data model.

3.3.2 Loading Time

For the initial load, the server needs to send the JavaScript file to the user and the SPA must initialize. For a typical user, the loading time of an application should not exceed 0.1 - 1 second. However, up to 10 seconds are the upper limit for most users, meaning users initial thoughts might be interrupted and they decide to do something else [45].

3.3.3 Resource Allocation

A smartphone might not have the same amount of memory available as a desktop computer. Therefore, it is necessary to limit the memory load when an SPA is running. There are various ways to control this, such as limiting the amount of data bindings during the first initial load.

3.4 Tools

To test the data bindings, the benchmarking framework BenchmarkJS is used [2]. The standard approach of doing performance tests is to run a certain test and present the results with a certain margin of error. This tests do not work properly on JavaScript applications since the time can vary from one browser to another browser. The approach of BenchmarkJS is to repeat the tests until 1% margin of error is achieved. It includes different type of clocks to get a better reliability of the time measurements [11]. The resource allocation and loading time tests are performed using the analytics tool provided by Google Chrome.

3.5 Prototype

The prototype is created by TodoMVC [6]. The code is taken from their GitHub repository [61]. TodoMVC is chosen because it is a minimal example of a web application with basic functionality. These applications are created by experienced developers who utilizes the way the framework is supposed to be used. The main functionality of the TodoMVC application is to be able to add tasks to a list, which can be marked as completed. The application has three main components:

3.5. PROTOTYPE

Input field: This text field is used for adding a task. At the bottom of the task list, the task is created as an uncompleted task. This component also consists of a button for marking all tasks as completed or uncompleted.

Task list: This component is a list of tasks. The tasks individually can be marked as completed or uncompleted, can be renamed or removed from the list.

Footer: The footer component is the row in the bottom of the list. This contains four buttons and one counter. The counter represents how many uncompleted tasks are left in the task list. The buttons “all”, “active” and “completed” are sorting options for the task list. The “Clear completed” button removes all completed tasks from the task list. The footer is hidden when there are no tasks in the list.

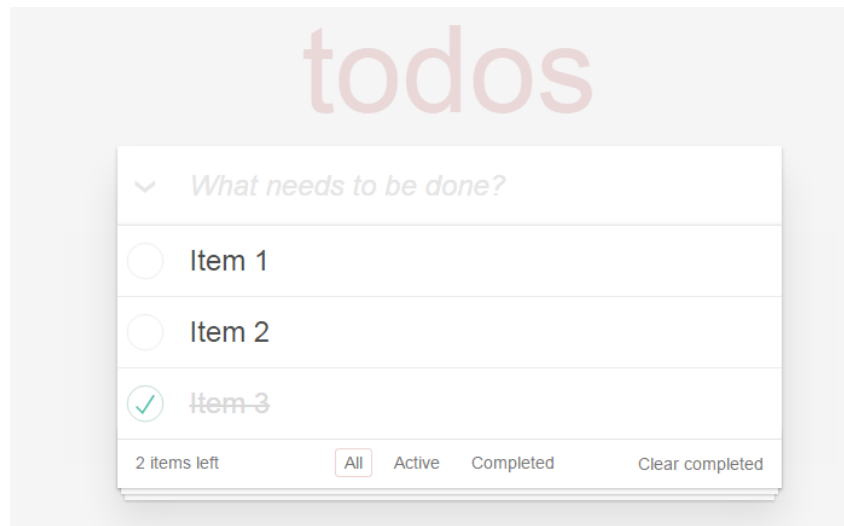


Figure 3.1. Figure of the TodoMVC application

Chapter 4

Formulating a Comparison Method

In chapter 4, the results of the interviews and the literature research are presented. Furthermore, the theoretical comparison is presented as a method with criteria and questions.

4.1 Results of the Interviews

The interviews are completed as described in section 3.2. The attendances are ten employees at Decerno with varying professional roles. The criteria that they think are important when choosing an SPA framework are summarized below:

- Cache performance
- Compatibility
- Developer guidelines
- Documentation
- Does in-house competence exists?
- Is it simple to use?
- Maturity
- Performance
- Popularity
- Portability
- Scalability
- Security
- Size
- Testability
- Who is the developer team behind it?

4.2 Results of the Literature Research

There is no doubt, when comparing something as complex as a framework, an abstraction is needed. As proposed by [20], an abstraction can be made by creating a “conceptual framework”, “blueprint architecture” or “blueprint framework”. The abstraction can be built with either a scenario- or criteria-based approach.

All methods mentioned in section 2.5 are using some of these concepts, which are summarized below:

4.2. RESULTS OF THE LITERATURE RESEARCH

Concept	Reference
Criteria-based comparison	[10, 15, 21, 40, 41, 57]
Scenario-based comparison	[13, 16, 33, 38, 39]
Collect business goals	[10, 13, 57]
Scoring with yes/no	[15, 21, 40]
Scoring with percentage	[10, 21, 57]
Scoring with discussion	[9, 13, 20, 33, 38, 39, 41, 57]
Score is weighted	[10, 21, 38, 57]
Evaluation of one candidate	[13, 16, 33, 38, 39]
Comparing different candidates	[10, 15, 21, 40, 41, 57]
Contain “social aspects”	[13, 16, 39]
Prototyping	[13, 15, 38, 40, 41]
Performance testing	[41]
One/many metric/metrics for every criteria	[15, 20, 21, 41]

Table 4.1. Concepts found in literature

Conclusions

As seen in table 4.1, all the scenario-based methods can only evaluate one candidate architecture at a time and the criteria-based methods are used to compare different candidates. In [10, 13, 57], the authors propose the use of business goals as a base for collecting scenarios. These scenarios are specific and may vary from project to project.

Another way of comparing software architecture is to utilize many different groups of employee and external experts. Some of these aspects proposed by [13, 16, 39]:

- Workshops with different development teams to get a broader perspective.
- Involving the whole team in the beginning of the development phase.
- Involving external experts in the design process.
- Presenting the findings for the other teams involved in the design process.

These social aspects are highly recommended when working in a project. Scenarios can fit well in an initial process of development. However, this paper proposes a method that can be used to compare SPA frameworks, therefore scenarios is not be an appropriate choice. The reason behind this is that the method proposed in this thesis is not have any specific parts that require other participants. It is also recommended that the same person or persons perform the method, since the answers can vary from person to person and making the actual comparison biased.

The authors behind SACAM, C. Stoermer, F. Bachmann and C. Verhoef state: “Comparing software architectures implies a set of criteria. A comparison without any criteria produces no sound reasoning about a selection.” [57]. As stated by

[57], the use of criteria is giving a sound reasoning of how well the architecture or framework performs when compared to another.

Naturally, it is important to identify a number of criteria which are general and essential. This thesis **has focus** on identifying a set of criteria as a fundamental part of the proposed method. The comparison is done by defining each criteria and formulating questions as proposed by [15,21,40,41]. These questions can be seen as metrics and how well the questions fill the criteria, as proposed by [15,20,21,41]. Weights can be used, however this thesis does not promote any specific way of using it. Ignacio Fernández-Villamor et al. in [21], are using weights on each question, and gives a score for each criteria based on the questions and their weight.

4.3 Criteria Collection

The table below consists of a summary of criteria which are found in the literature research (section 4.2) and the interviews (section 4.1):

Criteria	Reference
Cache performance	[41], (Section 4.1)
Documentation	[41, 55], (Section 4.1)
Does “in-house” experience exists	(Section 4.1)
Integration	[55]
Maintainability	[41], (Section 4.1)
Maturity	[41], (Section 4.1)
Modifiability	[20]
Modularity	[20, 21, 41], (Section 4.1)
Performance	[20, 41, 55], (Section 4.1)
Persistence	[20, 21]
Popularity	[21, 41]
Portability	[20, 40, 41], (Section 4.1)
Presentation	[21]
Reliability	[15]
Scalability	[20, 55], (Section 4.1)
Security	[15, 20, 21], (Section 4.1)
Simplicity	[55], (Section 4.1)
Size of the framework	(Section 4.1)
Testability	[20, 21, 41], (Section 4.1)
Transparency	[20]
Usability	[15, 21]

Table 4.2. Criteria found by literature research and interviews

4.4. CRITERIA DEFINITION

Conclusions

In order to limit the number of criteria, only the criteria with two or more references are selected. Stability and maturity are related because software maturity impacts on the stability, therefore they are put together to one criterion. To be able to use an SPA framework in as many browsers as possible, compatibility and portability are essential and thus these two criteria are grouped together. In computer science, persistence often relates to keeping the current state of the application in memory. To keep the state of an SPA in memory, good cache performance is required. This results in grouping persistence and cache performance together. Usability can be defined by the ISO 9241-11 standard, which covers the field where a user should be satisfied with the usage of the software [18]. This pairs well with simplicity, because if something is simple to use it should have a high usability, on the contrary, something that is not simple to use should have a lower grade of usability. Therefore, simplicity and usability is grouped.

Documentation, size of the framework and in-house experience are chosen as questions to use within their corresponding criteria.

4.4 Criteria Definition

In this section, the criteria found in section 4.3 are given wider definitions and questions that fulfill the criteria.

4.4.1 Security

Today, many web applications store user credentials or credit card numbers in a database that is connected to the Internet. If it would be a security breach in the web application, all the user's data can be stolen and the trust of users could be lost. OWASP (Open Web Application Security Project) [4] is a non-profit organization and an online community which main goal is to improve the security of software. One of their most well-known contribution is their top 10 list of the most common security issues with web applications.

SQ1: When a security related bug is found, does the framework have a security policy [3]? A security policy is how a security issue is handled by the developers behind the framework. This policy can be about responding to the issue by e-mail, who is notified and how long the framework update takes.

SQ2: Does the framework have a promotion for finding bugs [29]? These kind of promotions tend to give money to people who find security bugs in a web application. This might cause more people trying to find security related bugs.

SQ3: Does the framework prevent Cross Site Request Forgery [1]? Cross Site Request Forgery is a security issue where a form in a web application is vulnerable to manipulation, that the user is not aware of.

SQ4: Does the framework prevent Cross Site Scripting [5]? Cross Site Scripting is another security flaw in web applications. By utilizing this flaw, malicious JavaScript can be executed in the browser.

4.4.2 Modularity

The purpose for modularity is to give a structure for how the development with the particular SPA should be. An example of modularity could be to separate all components. If a component is moved from the application, it should work as an isolated unit and be fully functional. Separation could also mean having a separated view and data layer in the application. This approach makes it easier to change an isolated functionality in a specific component and overall make the SPA easier to maintain.

MQ1: Does the framework support a design pattern which leads to easier development of separated layers or components [41]? As mentioned above, separation is important and a framework can aid this by proposing a design pattern that is suitable for a component-based design or layer structure.

MQ2: Does the framework support a component-based approach of development? By having components that work as isolated units leads to better modularity where components can be moved or changed without interfering the rest of the application.

4.4.3 Popularity

Using a popular framework could be an important factor for both maturity and security. Much experience is gathered on StackOverflow and similar forums where developers discuss how they solved common problems. A large developer base can produce more bug reports and more people are likely to be involved in fixing these issues [52].

PQ1: How are the frameworks related to each other on Google Trends [41]?

A Google Trends graph can give an indication of how popular a framework is and how the number of searches has increased. If more than one framework is allowed to be displayed, it is possible to compare their popularity progression. This graph gives an overview of which framework is most popular and how this has changed over time.

PQ2: How many GitHub watchers does the framework repository have [41]?

If a framework has many followers on GitHub, a wider audience are able to try a version that still is in development.

PQ3: How many StackOverflow questions does the framework have [41]?

Mostly, problems are never unique and by having many questions already answered about the framework could save time.

4.4. CRITERIA DEFINITION

4.4.4 Maturity and Stability

There is no exact method of how to measure the maturity and stability of software. A CMM (Capability Maturity Model) can be used to measure the maturity of a process of software development within an organization [47]. However, it is difficult to know the processes since these are not publicly available. Instead, maturity and stability can be defined as how widely the framework is used by large corporations.

MSQ1: Is there any system of a larger corporation that uses the SPA framework in a production environment? If a large company uses this framework in one of their production system, this can be an indication of the maturity and stability of the framework and their belief in its future.

4.4.5 Simplicity and Usability

It is almost impossible to add new functionality to a software without doing any changes to the code base. By keeping the changes small, some defects can be avoided. In contrast, if a major change is required, the developer needs not to introduce more complexity than necessary. Given a context, the simplicity can vary between the users of the framework. For the original developer of a software it might be simpler to change the code than for a new developer. This issue can be solved by some kind of competence sharing, such as documentation or guidelines [37, p. 49-50].

SUQ1: Does in-house experience exists (Section 4.1)? The possibility for success is higher if there already exists knowledge of the framework. Furthermore, experienced staff can teach less experienced colleagues.

SUQ2: Does documentation exist [41], (Section 4.1)? When working with a new framework or expanding an existing application with new functionality, documentation is important to be able to solve problems.

SUQ3: Does any third-party tool provide code generation [21]? Instead of writing repetitive code, third-party tools can generate code and save time for the developer.

SUQ4: Does any IDE support this framework? An IDE (Integrated Development Environment) serves the developer as a support for auto completion of code or API documentation while writing.

SUQ5: Which JavaScript language does the framework promote? There are many different JavaScript versions and there exist a variety of JavaScript-based languages where the aim is to simplify development.

SUQ6: What is the cyclomatic complexity of the prototype? The cyclomatic complexity measurement can be used as an indicator of how complex the code is in an application. Having a low cyclomatic complexity can keep the software more reliable and maintainable [62].

4.4.6 Portability and Compatibility

JavaScript is executed in the browser and it is important that the framework is supported in as many browsers as possible. Equally important is the support for mobile devices, since many people are using their mobile devices for browsing the Internet.

PCQ1: Which browsers are every release tested with [41]? For portability reasons, it is important that all releases are tested in as many browsers as possible.

PCQ2: Which is the current latest browser versions supported [40,41]?

Most of the users tend to update their browsers more regularly except the Internet Explorer users. Most of the Internet Explorer users are spread out through versions 8, 9, 10, 11 and also Microsoft's new web browser Edge 12 and Edge 13 [56]. Therefore, it is important to support the latest version available.



PCQ3: Which is the earliest browser version supported? Legacy support is another important factor because many users are still not using the latest versions of their current web browser. As seen in PCQ2, the user base of Internet Explorer is more spread out through many versions of the web browser.

PCQ4: Which version of JavaScript (ECMAScript) is the framework built with?

Currently, ECMAScript 6 is not fully supported in all browsers. For this reason, all JavaScript that use the new ECMAScript 6 functionality need to be compiled to ECMAScript 5. If the framework is built with ECMAScript 6 or any future version of ECMAScript that is not currently fully supported in all browsers, needs to compile into the latest version that is currently supported.

PCQ5: Does the framework support mobile devices? Today, many users

browse the Web with their mobile device. Therefore, it is important for framework to support mobile devices.



PCQ6: Is the framework compatible with other libraries that is required by the application?

When developing a web application, it is common to use more than one library or framework. Therefore, it is important that the framework should compatible with the libraries or frameworks that is used.

4.4.7 Cache Performance and Persistence

Cache performance and persistence is important in an SPA to enhance the user experience. If the initial loading time is too high, a user might exit the SPA before it is properly loaded. Another reason to have some utilities for boosting cache performance is to reduce the data load from the server to the SPA. Hence, this can save resources on the server and decrease the time the user has to wait.

4.4. CRITERIA DEFINITION

CPQ1: What is the size of the JavaScript file (Section 4.1)? For a user to be able to use an SPA, the JavaScript files need to be downloaded. A larger file size increases the loading time and **place** more strain on the server.

CPQ2: Does any functionality exists to support less data transfer between the SPA and the server [41]? The framework can save data transfers by caching resources.

4.4.8 Testability

Whether the developers use a test-driven development or not, it is important to have a framework that can be properly tested. To ensure that changes do not interfere with old tests or that new functionality can be tested before a new release.

TQ1: Does the framework support unit testing [21,41]? Unit testing allows testing of a module to ensure its functionality.

TQ2: Does the framework support integration testing [21]? Integration testing or end-to-end testing allows testing of the full application.

TQ3: Does the framework support functional testing [21]? Functional testing allows testing of certain logic in the application.

TQ4: Does the framework support performance testing [21,41]? Performance testing allows testing of parts or the whole application through benchmarking or profiling.

TQ5: Does the framework support mocking of objects [21,41]? Mocking of objects makes some tests easier to perform when there is not sufficient data or if it is hard to define the data.

4.4.9 Maintainability

In software development, it is common for one team to start the development and another one taking over. One example of having a high level of maintainability is that a new developer can start to fix issues and bugs quickly [51].

MaQ1: Does the framework promotes any developer guidelines? When working in a large team it is important that every involved developer knows how to work with the framework.

MaQ2: Does experience of this framework exist among programmers [21]? If there exist many developers with knowledge of this particular framework, it reduces the risk of not being able to recruit experienced developers.

MaQ3: Who is the developer behind this (Section 4.1)? Is the developer a single person, a developer team or a company? A company that develops a

framework has professional people that work with it on a daily basis, a single person might not be able to work daily with the framework and might not update the framework as frequently as necessary.

MaQ4: Does the team behind the framework use it in their own production environment? If the developer team behind the framework use their own framework in a production environment, this can be an indication that they believe in their product and that it is suitable for a production system.

MaQ5: How many lines of code is the prototype written in [52]? An application with fewer lines are suggested to be more maintainable [52].

Chapter 5

Comparison of SPA Frameworks

Chapter 5 contains a comparison between AngularJS, React and Angular 2, by using the method proposed in chapter 3 and 4.

5.1 Frameworks

In this section, the frameworks AngularJS, React and Angular 2 are introduced by a short summary.

5.1.1 AngularJS

AngularJS is developed by Google Inc. and the initial release was in 2010. The aim of AngularJS is to extend the HTML vocabulary with data bindings. This is implemented with `ng-tags`, which bind the view to one or many models. [26]. The data bindings in AngularJS are implemented with dirty checking. This means that if a value is bound to the view through a model, it is not immediately updated, instead it is updated when AngularJS does the dirty checking on the value [32]. In addition, AngularJS was created with testability in mind and therefore has built-in dependency injection. This makes it easier to test individual components [27].

Current version

Version 1.5.5. March 18, 2016.

5.1.2 React

React is a framework developed by Facebook Inc. and the initial release was in 2013. It uses a Virtual DOM, where new DOM trees can be created with JavaScript, which creates a simpler programming model. The data bindings in React are implemented with something Facebook Inc. calls *diff algorithm*. This algorithm causes a full re-render of the application every time something changes. To detect the changes in the Virtual DOM, React compares the DOM trees in every level and re-renders when

a change is found [25]. In addition, React has its own JavaScript language called JSX. This provides an extension to the JavaScript language by adding XML-like syntax, which gives the programmer an ability to use a similar language to HTML within the code [24].

Current version

Version 0.14.8. March 29, 2016

5.1.3 Angular 2

Angular 2 is a major rewrite of AngularJS and is currently in beta. As AngularJS, Angular 2 also utilizes ng-tags, but with a different syntax. On the other hand, Angular 2 does not use dirty checking as AngularJS does. Instead Angular 2 uses component trees, where the parents are on a higher level in the tree than their children. Every component has a change detector class, where the parent can update its children if a change is detected [28]. Angular 2 offers better performance than AngularJS and one of the reasons is that it can load code dynamically when needed during the execution [31].

Current version

Version 2.0.0, beta 17. April 30, 2016.

5.2 Practical Comparison

In this section the practical comparison is made. This is done by presenting the results from the performance tests and a code comparison between the frameworks.

5.2.1 Performance Results

The tests shown in table 5.1 and 5.2 is performed with this test setup:

- Windows 8.1 Pro
- Intel Core i7-4790k CPU @ 4.00 GHz
- 16 GB RAM
- Google Chrome 50.0.2661.102m

5.2. PRACTICAL COMPARISON

Test case (ms)	AngularJS	React	Angular 2
Script loading time	757.6 ± 14.9%	844.2 ± 15.5%	880.5 ± 19.7%
Load 10 items	33	21	14
Load 100 items	182	67	46
Load 500 items	852	253	225
Load 1000 items	1350	553	384
Load 2000 items	2651	1050	827
Load 3000 items	3939	1425	1192
Load 4000 items	5210	1991	1591
Load 5000 items	6633	2426	1830
Edit 10 items	3	7	1
Edit 100 items	14	39	4
Edit 500 items	59	153	13
Edit 1000 items	108	291	23
Edit 2000 items	193	603	39
Edit 3000 items	285	962	55
Edit 4000 items	403	1210	73
Edit 5000 items	458	1510	93

Table 5.1. Script loading time and time of loading and editing 10-5000 items in AngularJS, React and Angular 2.

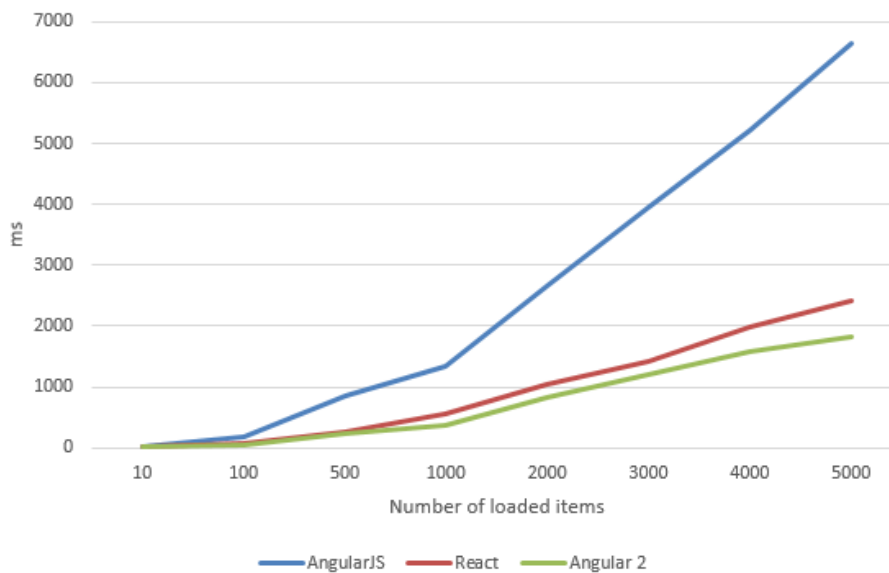


Figure 5.1. Figure of the results from table 5.1.

CHAPTER 5. COMPARISON OF SPA FRAMEWORKS

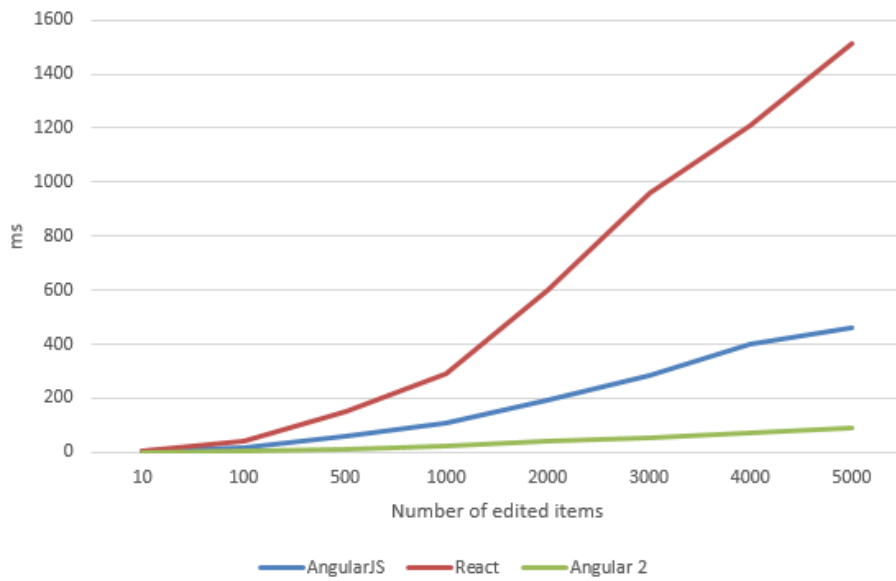


Figure 5.2. Figure of the results from table 5.1.



Test case (MB)	AngularJS	React	Angular 2
Initial load	10.5	10.8	11.2
Load 10 items	11	11.4	11.6
Load 100 items	13.5	12.3	13.3
Load 500 items	17.2	15.9	19.3

Table 5.2. Memory allocation when loading 0, 10, 100 and 500 items in AngularJS, React and Angular 2.

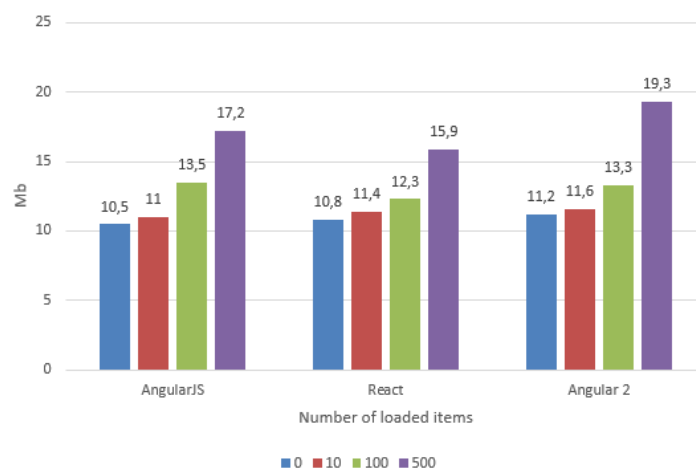


Figure 5.3. Figure of the results from table 5.2.



5.2. PRACTICAL COMPARISON

5.2.2 Code comparison

In this subsection a code comparison between AngularJS, React and Angular 2 is made. This comparison consists of comparing the prototype by some of its core features:

- How is the input saved?
- How is the footer hidden and shown?
- How are the task items created and shown in a list?

How this application is working can be read in section 3.5.

AngularJS

One of the main goals of AngularJS is to provide an extension to HTML. This is done with ng-tags, which are used within the HTML file to bind the view to the data models with the help of controllers. These controllers are written in JavaScript. In the following examples the controller is called TC and holds all the functionality for manipulating the model.

```
1 <form id="todo-form" ng-submit="TC.addTo()">
2 <input id="new-todo" placeholder="What needs to be done?" ng-model="
  TC.newTodo.title" autofocus>
3 </form>
```

Listing 5.1. Example of how to save the input in AngularJS

As seen in listing 5.1, it is a data binding between the input field and the model. The ng-model-tag binds the input field to TC.newTodo.title. By binding the title value to newTodo, the controller is able to create a new task object by accessing the input field. To submit this value, the ng-submit-tag is used. Then, the function addTo within the controller handles the creation of the task.

```
1 <footer id="footer" ng-show="TC.todos.length" ng-cloak>
2 ...
```

Listing 5.2. Example of hiding the footer in AngularJS

To show the footer, a ng-show-tag is used, as seen in listing 5.2. This tag makes the footer to be shown when the expression inside the tag is validated to true. TC.todos.length is evaluated to true if the task list's length is greater than zero. If the value is zero, the footer is hidden. To avoid flickering while loading the application, the ng-cloak-tag is used. This causes the footer to stay hidden until the expression inside the ng-show can be validated.

```
1 <li ng-repeat="todo in TC.todos | filter:TC.statusFilter track by
  $index">
2 ng-class="{completed: todo.completed, editing: todo === TC.editedTodo
  }">
3 <div class="view">
4 <input class="toggle" type="checkbox" ng-model="todo.completed">
```

```

5   ...
6 </div>
7   ...

```

Listing 5.3. Example of showing a list of tasks in AngularJS

As seen in listing 5.3, the `ng-repeat`-tag is used to iterate over a list of items. A filter is applied so the tasks can be sorted by their status. The `ng-class`-tag extends the HTML class-tag, which can alter the visual style of the HTML. The `ng-class` in this example alters how the item looks like if it is completed or if it is being edited, and adds the HTML-tag accordingly. However, this depends on the state of the task.

React

The React application is separated into four components, the input field, the task list, the task items and the footer. If one component uses another, both are rendered. In this case, the main application is the input field, which uses the list and the footer, the list uses the task items.

```

1 <h1>todos</h1>
2 <input
3   className="new-todo"
4   placeholder="What needs to be done?"
5   value={this.state.newTodo}
6   onKeyDown={this.handleNewTodoKeyDown}
7   onChange={this.handleChange}
8   autoFocus={true}
9 />

```

Listing 5.4. Input field component in React

```

1 handleNewTodoKeyDown: function (event) {
2   if (event.keyCode !== ENTER_KEY) {
3     return;
4   }
5   event.preventDefault();
6   var val = this.state.newTodo.trim();
7   if (val) {
8     this.props.model.addToDo(val);
9     this.setState({newTodo: ''});
10  }
11 }

```

Listing 5.5. Logic for the input field component

In listings 5.4 and 5.5, the input field can be seen, and it is a separate component with its own internal logic. In listing 5.4, the visual elements of the component are written. JavaScript do not support HTML, but with the help of JSX, it is possible to combine JavaScript with a HTML-like syntax. Regarding the logic in listing 5.5, the `onKeyDown` property is triggered when a user presses a key. In this case the `handleNewTodoKeyDown` function is called. `handleNewTodoKeyDown` is a function in

5.2. PRACTICAL COMPARISON

listing 5.5. This function checks whether the Enter key is pressed or not. If it is, the value from the input field is saved and a trim function is called on the input. The `addTodo` function handles the addition to the model and the `setState` function resets the state of the input field to an empty string.

```
1 if (activeTodoCount || completedCount) {
2   footer =
3   <TodoFooter
4     count={activeTodoCount}
5     completedCount={completedCount}
6     nowShowing={this.state.nowShowing}
7     onClearCompleted={this.clearCompleted}
8   />;
9 }
```

Listing 5.6. Example of hiding the footer in React

In listing 5.6, the `TodoFooter` component and its logic is presented. If `activeTodoCount` (the amount of active tasks) or `completedCount` (amount of completed tasks) is larger than zero, the footer is shown and the values from the model is bound to it via JSX. The footer component, `TodoFooter`, uses these values and cause a re-render of the application.

```
1 var todoItems = shownTodos.map(function (todo) {
2   return (
3     <TodoItem
4       key={todo.id}
5       todo={todo}
6     ...
7   />
8 );
9 }, this);
```

Listing 5.7. Example of a task item in React

```
1 <ul className="todo-list">
2   {todoItems}
3 </ul>
```

Listing 5.8. How to show all task items in React

In listing 5.7, React creates all tasks as separate components `<TodoItem ...>` via JSX. These are created and saved with their properties in a list, `todoItems`. This list is bound to the view with `{todoItems}`, as seen in listing 5.8.

Angular 2

```
1 <input class="new-todo" placeholder="What needs to be done?"
2 autofocus="" [(ngModel)]="newTodoText" (keyup.enter)="addTodo()">
```

Listing 5.9. Example of to save the input in Angular 2

In listing 5.9, the view is able to connect to the model via the `ngModel`-tag. The `keyup.enter` is a function that runs every time the Enter key is pressed. The

function `addTodo` is within the controller and alters the model and saves task. In addition, the `addTodo` function resets the input field by setting the `newTodoText` to an empty string.

```
1 <footer class="footer" *ngIf="todoStore.todos.length > 0">
2 ...
```

Listing 5.10. Example of hiding the footer in Angular 2

In listing 5.10, the `ngIf`-tag is used in the same way as in AngularJS. If the expression is validated to true, the footer is shown. The `todoStore` is a data structure that makes it possible to save all task items. `todos` is a list that is saved in the `todoStore`. In AngularJS the `ng-cloak` property is necessary to avoid flickering, but this is included in Angular 2's `ngIf`-tag. Therefore, the footer is hidden until the validation can be done.

```
1 <li *ngFor="#todo of todoStore.todos" [class.completed]="todo.
  completed" [class.editing]="todo.editing">
2 <div class="view">
3 ...
4 </li>
5 ...
```

Listing 5.11. Example of showing task items in Angular 2

In listing 5.11, the `ngFor`-tag iterates over all tasks in the list `todos`. For every iteration, a `todo` task is created. The class brackets are used to alter the style of the tasks whether the task is completed, uncompleted or edited.

5.3 Theoretical Comparison

In this section the theoretical comparison is made. All questions are answered and are followed by a figure of Google Trends and conclusions.



Question	AngularJS	React	Angular 2
SQ1	Yes, an e-mail address	No	No
SQ2	No	No	No
SQ3	Yes	No	No
SQ4	Yes	Yes	No
MQ1	Yes	Yes	Yes
MQ2	No	Yes	Yes
PQ1	See figure 5.4	See figure 5.4	See figure 5.4
PQ2	4256	2993	1480
PQ3	166994	13692	961
MSQ1	Sony and Google	Instagram and Facebook	None

Table 5.3. Answers to the questions SQ1 to MSQ1

5.3. THEORETICAL COMPARISON

SUQ1	N/A	N/A	N/A
SUQ2	Yes	Yes	Yes
SUQ3	Yes	Yes	Yes
SUQ4	Yes	Yes	Yes
SUQ5	ECMAScript 5	JSX	TypeScript
SUQ6	6	31	6
PCQ1	Internet Explorer, Firefox and Chrome	Internet Explorer, Firefox and Chrome	Internet Explorer, Firefox and Chrome
PCQ2	Firefox 45, Safari 8, Internet Explorer 11, Chrome 50	Firefox 45, Safari 8, Internet Explorer 11, Chrome 50	Firefox 45, Safari 8, Internet Explorer 11, Chrome 50
PCQ3	Same as above, with Internet Explorer 9,10	Same as above, with Internet Explorer 9,10	Same as above, with Internet Explorer 9,10
PCQ4	ECMAScript 5	ECMAScript 5	ECMAScript 5
PCQ5	Yes	Yes	Yes
PCQ6	N/A	N/A	N/A

Table 5.4. Answers to the questions SUQ1 to PCQ6

CPQ1	154 kB	142 kB	621 kB
CPQ2	Yes, \$cachefactory	No	No
TQ1	Yes	Yes	Yes
TQ2	Yes	No	No
TQ3	Yes	Yes	Yes
TQ4	No	No	No
TQ5	Yes	Yes	No
MaQ1	Yes	Yes	No
MaQ2	Yes	Yes	No
MaQ3	Google Inc.	Facebook Inc.	Google Inc.
MaQ4	Yes	Yes	No
MaQ5	161	493	214

Table 5.5. Answers to the questions CPQ1 to MaQ5



Figure 5.4. Figure of Google Trends between AngularJS, React and Angular 2.

5.4 Comparison Conclusions

By applying the questions suggested in section 4.4, many similarities can be seen through the majority of the questions, and that make it hard to differentiate between the frameworks. However, the greatest differences can be found within the popularity criteria (PQ1-PQ3), in table 5.1. AngularJS has the most watchers on GitHub (PQ2), most questions on StackOverflow (PQ3) and the highest amount of searches on Google (figure 5.4). A possible reason behind this popularity may be that AngularJS is the oldest of the three frameworks and existed during the up going trend for SPAs in general.

Angular 2 is the largest of the three frameworks and therefore has the highest initial load time and allocates most resources during runtime (figure 5.2.1). One of the reasons behind this could be that Angular 2 still is in beta stage and code optimizations are yet to be developed. Furthermore, Angular 2 is the fastest framework to load 10-5000 items and to edit all these items (table 5.1). React is faster than AngularJS when loading items, but the slowest framework to edit items. AngularJS is overall the slowest framework when it comes to loading items. When loading and editing items, DOM is changed and therefore the implementation of the data bindings is important. As seen in table 5.1, React with its diff algorithm is slowest when editing and AngularJS with its dirty checking is slowest when new data bindings are created. However, as described in subsection 3.3.2, if it takes less than 1 second, the user's initial thoughts is not interrupted. With this in mind, AngularJS can load between 500-1000 items, React can load 2000 items and Angular 2 can load roughly 3000 items, as seen in table 5.1. A rough assumption could be that a typical SPA has 500-2000 data bindings, which makes all of the three frameworks a suitable choice if trying to be below the 1-second limit.

AngularJS and Angular2 have similar syntax. They both utilize ng-tags to extend the HTML with data bindings and data models through controllers. On the other hand, React has a component-based approach, where each component is responsible for its own state and logic. However, if any of these frameworks are unknown, Angular 2 has an almost nonexistent documentation and is a worse choice than React or AngularJS.

By performing this comparison, it is suggested that AngularJS is the most suitable, due to its popularity and stability.

Chapter 6

Discussion and Conclusion


This chapter consists of a discussion about the method, proposals for future work, a socio-ethical discussion and conclusions.

6.1 Summary

To be able to make an abstraction of an SPA, criteria and questions are collected. These are chosen based on current frameworks, methods proposed by research and interviews with developers. The abstraction is used as a base in the comparison method, in addition to performance test and code comparison. By using this method AngularJS is suggested to be the most suitable choice due to its popularity and stability. However, AngularJS has bad performance when creating more than 1000 data bindings.

6.2 Discussion



Only considering the yes and no questions, the conclusion would be that the  are different. However, taking the other questions in consideration, the answers were more similar for all of the frameworks. It is very favorable for frameworks to share complementary similar functionality. Thus, having a code comparison and performance tests was an appropriate part of the method. Nevertheless, this can also make a comparison between the frameworks more difficult, as there is a vague distinction between the frameworks. One possible reason for inexplicit difference between the answers could be the lack of criteria or questions. With more criteria and questions, there is a greater possibility for a larger variety of outcomes.

6.3 Socio-ethical discussion

One of the socio-ethical effects this paper, is that it could lead to a more diverse Internet. Today, there are many old web applications where the back-end is large and very computational heavy. By using an SPA framework, a smaller company

or startup with no funding, can place the logic on the client side and thus not needing a large back end service to serve the users. By having a properly chosen SPA framework it increase the software quality and reduce the development costs. This may create an opportunity where more web applications are being developed, that previously could not.

6.4 Future work

It would be appropriate to compare this method to other SPA comparison methods to see how it performs. However, this comparison is difficult since there is a lack of research within this field. But this field is likely to grow, since SPAs still are relatively new and are the new trend in web development.

In addition, to get a more diverse outcome of the comparison, more criteria and questions can be collected. This can be done by conducting more interviews with developers or other people with a technical role towards SPAs. Another way to improve this method can be by doing a more extensive literature study, however it might take a few years until more has been researched in this area. Alternatively, instead of creating more criteria or questions, the already chosen ones can be evaluated. As an example, the questions regarding browser compatibility, was this a correct approach, since many of the frameworks had similar compatibility. There might be a norm in web development that all applications should be compatible with these browsers and versions.

Moreover, this method could also be tested with more or other frameworks than the three chosen in this paper. If this were to be done, more variety in the results might be found. However, AngularJS is still one of the oldest SPA frameworks and is one of the most popular. It would be hard for another framework to try to compete with AngularJS.



Bibliography

- [1] AngularJS: Security. <https://docs.angularjs.org/guide/security>, 2016. Accessed: 2016-04-05.
- [2] BenchmarkJS. <https://benchmarkjs.com/>, 2016. Accessed: 2016-04-05.
- [3] EmberJS: Security. <http://emberjs.com/security/>, 2016. Accessed: 2016-04-05.
- [4] OWASP. https://www.owasp.org/index.php/Main_Page, 2016. Accessed: 2016-04-05.
- [5] React: JSX Gotchas. <https://facebook.github.io/react/docs/jsx-gotchas.html>, 2016. Accessed: 2016-04-05.
- [6] TodoMVC. <https://www.todomvc.com/>, 2016. Accessed: 2016-04-22.
- [7] AngularJS and community. Data Binding. <https://docs.angularjs.org/guide/databinding>, 2016. Accessed: 2016-04-05.
- [8] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2 edition, 2003.
- [9] PerOlof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet. Architecture-level modifiability analysis (alma). *Journal of Systems and Software*, 69(1–2):129 – 147, 2004.
- [10] Klaus Bergner, Andreas Rausch, Marc Sihling, and Thomas Ternité. Dosam – domain-specific software architecture comparison model. In *Proceedings of the First International Conference on Quality of Software Architectures and Software Quality, and Proceedings of the Second International Conference on Software Quality, QoSA’05*, pages 4–20, Berlin, Heidelberg, 2005. Springer-Verlag.
- [11] Mathias Bynens and John-David Dalton. Bulletproof JavaScript benchmarks. <http://calendar.perfplanet.com/2010/bulletproof-javascript-benchmarks/>, 2010. Accessed: 2016-04-05.

BIBLIOGRAPHY

- [12] Jing Chen. Hacker Way: Rethinking Web App Development at Facebook. <https://www.youtube.com/watch?v=nYkdrAPrdcw>, 2014.
- [13] Paul C. Clements. The architecture tradeoff analysis method. Technical Report CMU/SEI-2000-TN-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2000.
- [14] J. Collis and R. Hussey. *Business Research: A Practical Guide for Undergraduate and Postgraduate Students*. Palgrave Macmillan, 2009.
- [15] María del Pilar Salas-Zárate, Giner Alor-Hernández, Rafael Valencia-García, Lisbeth Rodríguez-Mazahua, Alejandro Rodríguez-González, and José Luis López Cuadrado. Analyzing best practices on web development frameworks: The lift approach. *Science of Computer Programming*, 102:1 – 19, 2015.
- [16] T. Dolan. *Architecture Assessment of Information-system Families: A Practical Perspective*. Technische Universiteit Eindhoven, 2001.
- [17] Rolf Ejvegård. *Vetenskaplig metod*. Studentlitteratur, 2009.
- [18] International Organization for Standardization. Iso 9241-11:1998, 1998.
- [19] M. Fowler. *Patterns of Enterprise Application Architecture*. A Martin Fowler signature book. Addison-Wesley, 2003.
- [20] Anton Gerdessen. Framework comparison method: Comparing two frameworks based on technical domains, focussing on customisability and modifiability. Master’s thesis, UvA, University of Amsterdam, 2007.
- [21] José Ignacio Fernández-Villamor, Laura Díaz-Casillas, and Carlos Á. Iglesias. A comparison model for agile web frameworks. In *Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*, EATIS ’08, pages 14:1–14:8, New York, NY, USA, 2008. ACM.
- [22] Apple Inc. JavaScriptCore. <http://trac.webkit.org/wiki/JavaScriptCore>, 2016. Accessed: 2016-04-22.
- [23] Facebook Inc. Overview. <https://facebook.github.io/flux/docs/overview.html#content>, 2016. Accessed: 2016-04-05.
- [24] Facebook Inc. React (Virtual) DOM Terminology. <https://facebook.github.io/react/docs/glossary.html>, 2016. Accessed: 2016-04-22.
- [25] Facebook Inc. Reconciliation. <https://facebook.github.io/react/docs/reconciliation.html>, 2016. Accessed: 2016-05-22.
- [26] Google Inc. AngularJS. <https://angularjs.org/>, 2016. Accessed: 2016-04-22.

- [27] Google Inc. AngularJS: Unit Testing. <https://docs.angularjs.org/guide/unit-testing>, 2016. Accessed: 2016-04-22.
- [28] Google Inc. ChangeDetectorRef. <https://angular.io/docs/ts/latest/api/core/ChangeDetectorRef-class.html>, 2016. Accessed: 2016-05-22.
- [29] Google Inc. Chrome Reward Program Rules. <https://www.google.com/about/appsecurity/chrome-rewards/>, 2016. Accessed: 2016-04-22.
- [30] Google Inc. Chrome V8. <https://developers.google.com/v8/>, 2016. Accessed: 2016-04-22.
- [31] Google Inc. Features & Benefits. <https://angular.io/features.html>, 2016. Accessed: 2016-04-22.
- [32] Google Inc. What are Scopes? <https://docs.angularjs.org/guide/scope>, 2016. Accessed: 2016-05-22.
- [33] Software Engineering Institute. Cost Benefit Analysis Method. <http://www.sei.cmu.edu/architecture/tools/evaluate/cbam.cfm>. Accessed: 2016-04-05.
- [34] Prashant Bansode Scott Barber J.D. Meier, Carlos Farre and Dennis Rea. *Performance Testing Guidance for Web Applications: Chapter 1 – Fundamentals of Web Application Performance Testing*. Microsoft Press, 2007.
- [35] Prashant Bansode Scott Barber J.D. Meier, Carlos Farre and Dennis Rea. *Reusing Open Source Code*. Microsoft Press, 2011.
- [36] Ralph E. Johnson. Components, frameworks, patterns. In *Proceedings of the 1997 Symposium on Software Reusability, SSR '97*, pages 10–17, New York, NY, USA, 1997. ACM.
- [37] Max Kanat-Alexander. *Code Simplicity*. O'Reilly Media, 2012.
- [38] Rick Kazman, Len Bass, Mike Webb, and Gregory Abowd. Saam: A method for analyzing the properties of software architectures. In *Proceedings of the 16th International Conference on Software Engineering, ICSE '94*, pages 81–90, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.
- [39] Rick Kazman, Mark Klein, Mario Barbacci, Thomas Longstaff, Howard Lipson, and S. Carriere. Active reviews for intermediate designs. Technical Report CMU/SEI-98-TR-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1998.
- [40] Joe Lennon. Compare JavaScript frameworks. <http://www.ibm.com/developerworks/library/wa-jsframeworks/>, 2010. Accessed: 2016-04-05.

BIBLIOGRAPHY

- [41] Tim Johan Malmström. Structuring modern web applications : A study of how to structure web clients to achieve modular, maintainable and longlived applications. Master's thesis, KTH, School of Computer Science and Communication (CSC), 2014.
- [42] A. Mesbah and A. van Deursen. Migrating multi-page web applications to single-page ajax interfaces. In *Software Maintenance and Reengineering, 2007. CSMR '07. 11th European Conference on*, pages 181–190, March 2007.
- [43] Mozilla. SpiderMonkey. <https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey>, 2016. Accessed: 2016-04-22.
- [44] S. Murugesan. Understanding web 2.0. *IT Professional*, 9(4):34–41, July 2007.
- [45] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [46] M.Q. Patton. *Qualitative Research & Evaluation Methods*. SAGE Publications, 2002.
- [47] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability maturity model, version 1.1. *IEEE Softw.*, 10(4):18–27, July 1993.
- [48] Dr. Axel Rauschmayer. Chapter 4. How JavaScript Was Created. <http://speakingjs.com/es5/ch04.html>, 2016. Accessed: 2016-04-05.
- [49] Dr. Axel Rauschmayer. Chapter 5. Standardization: ECMAScript. <http://speakingjs.com/es5/ch05.html>, 2016. Accessed: 2016-04-05.
- [50] Dirk Riehle. Framework design a role modeling approach. Diss. ETH No. 13509, 2000.
- [51] A. F. Rosene, J. E. Connolly, and K. M. Bracy. Software maintainability - what it means and how to achieve it. *IEEE Transactions on Reliability*, R-30(3):240–245, Aug 1981.
- [52] Ioannis Samoladas, Ioannis Stamelos, Lefteris Angelis, and Apostolos Oikonomou. Open source software development should strive for even greater code maintainability. *Commun. ACM*, 47(10):83–87, October 2004.
- [53] Douglas C. Schmidt and Frank Buschmann. Patterns, frameworks, and middleware: Their synergistic relationships. In *Proceedings of the 25th International Conference on Software Engineering, ICSE '03*, pages 694–704, Washington, DC, USA, 2003. IEEE Computer Society.
- [54] Meg Sewell. The use of qualitative interviews in evaluation. <http://ag.arizona.edu/sfcs/cyfernet/cyfar/Intervu5.htm>, 2016. Accessed: 2016-04-05.

- [55] T. C. Shan and W. W. Hua. Taxonomy of java web application frameworks. In *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on*, pages 378–385, Oct 2006.
- [56] StatCounter Global Stats. Top 12 Desktop Browser Versions Combining Chrome and Firefox (5+) from Apr 2015 to Apr 2016. http://gs.statcounter.com/#desktop-browser_version_partially__combined-ww-monthly-201504-201604, 2016. Accessed: 2016-05-02.
- [57] Christoph Stoermer, Felix Bachmann, and Chris Verhoef. Sacam: The software architecture comparison analysis method. Technical Report CMU/SEI-2003-TR-006, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.
- [58] Mikito Takada. Single page apps in depth. <http://singlepageappbook.com/goal.html>, 2016. Accessed: 2016-04-05.
- [59] Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide, 2nd Edition. Chapter 1: What is Software Architecture? <https://msdn.microsoft.com/en-us/library/ee658098.aspx>, 2009. Accessed: 2016-04-05.
- [60] Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide, 2nd Edition. Chapter 4: A Technique for Architecture and Design. <https://msdn.microsoft.com/en-us/library/ee658084.aspx>, 2009. Accessed: 2016-04-05.
- [61] TodoMVC. TodoMVC GitHub Repository. <https://github.com/tastejs/todomvc>, 2016. Accessed: 2016-05-22.
- [62] Arthur H. Watson and Thomas J. McCabe. Structured testing: A testing methodology using the cyclomatic complexity metric. Technical report, Computer Systems Laboratory, National Institute of Standards and Technology. <http://www.mccabe.com/pdf/mccabe-nist235r.pdf>.