How does the virtual DOM compare to other DOM update mechanisms in JavaScript frameworks?

Daisy Muyldermans

A research paper submitted to the University of Dublin,
In partial fulfilment of the requirements for the degree of Master of Science Interactive Digital Media

2019

Declaration

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: http://www.tcd.ie/calendar

I have also completed the Online Tutorial on avoiding plagiarism 'Ready, Steady, Write', located at http://tcdie.libguides.com/plagiarism/ready-steady-write

I declare that the work described in this research Paper is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Daisy Muyldermans

10 May 2019

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this research paper upon request.

signed: _____

Daisy Muyldermans
10 May 2019

Acknowledgements

I would like to take the opportunity to thank my lecturer and supervisor, Kathryn Cassidy, for her support throughout this process. Her lectures have taught me a lot and will definitely help me in the future. I would like to thank James for his support this year, I would not have made it this far without him. I would also like to thank the web developers for the time they took to fill out my survey. And I would like to thank my friends who listened and supported me when needed.

This research project would not have been possible without your help, thank you!

Abstract

Name: Daisy Muyldermans

Supervisor: Kathryn Cassidy

Year: 2019

Title of degree: Master of Science Interactive Digital Media

How Does the virtual DOM compare to other DOM updating mechanisms in JavaScript frameworks?

Facebook created its own JavaScript framework called React. This framework uses a concept called the virtual DOM. The discussion on whether the virtual DOM is more effective than other frameworks has been ongoing in the JavaScript community. Facebook has created a framework called React that uses a DOM updating mechanism called the virtual DOM.

This research project aims to compare the virtual DOM to other DOM updating mechanisms in other JavaScript frameworks. The virtual DOM method will be compared to other alternatives such as Key-value-observation and the digest cycle.

Online surveys were sent out to web developers to ask about their experience with the virtual DOM and with a JavaScript framework called React, which uses the virtual DOM technique. The findings of the survey results will be compared to the theoretical research in this project.

Results show that the virtual DOM scores high in terms of better debugging and short learning curve. Developers have the opinion that Facebook played a huge role in making React a popular framework.The virtual DOM scored better in terms of learning curve and debugging. It is also cheaper than other techniques.

Table of contents

List of figures

1. Introduction

JavaScript is a programming language that has become popular in the world of front-end development. The reason why JavaScript is widely used is because it is the only language that is supported by the web natively. JavaScript is used by 94.5% of all websites (Medium, 2017) and the reason why is because JavaScript allows to create dynamic web applications. It is the programming language that allows web developers to build responsive designs that work on different devices, browsers and operating systems (Medium, 2017). This is also why JavaScript is a language that is constantly changing. New libraries, new frameworks and new languages including the Javascript language are being created and updated on a regular basis. The main reason for creating new Javascript frameworks is to make the lives of front-end developers easier, but having to choose from so many frameworks can become mind-boggling for most developers (Greif, Benitte and Rambeau, 2018). Applications are getting larger and more complex, and the responsibility of a web developer is therefore increasing.

Facebook has created a framework called React, a framework that uses a concept called the virtual DOM to manage dynamic changes. It has been argued that using the virtual DOM can result in more efficient application development (Grov, 2015). This research project will therefore analyse how efficient the virtual DOM is in front-end development and why it could be one of the most efficient techniques compared to techniques used in other frameworks.

As mentioned above, React is a framework created by Facebook, the company uses React for the chat system and to create Instagram's front-end. Aside from Facebook, the largest bank in Russia called Sberbank[1], Firefox[2], Netflix[3], and Khan Academy[4], a non-profit educational organisation offering exercises and videos to teach mathematics and other fields, are working with React. Paypal[5] uses React to build the company's applications. And lastly, New York Times[6] uses the virtual DOM for various parts of their web page (Grov, 2015).

Social media applications are popular and used by a large number of the population on a daily basis. The large amount of users also means that there is a large amount of underlying data (Grov, 2015). The data in social media applications need constant updating so that the changes

---

[1] www.sberbank.ru/en/about/about_sberbank

[2] www.mozilla.org/en-US/firefox/new/

[3] www.netflix.com

[4] www.khanacademy.org

[5] www.paypal.com

[6] www.nytimes.com

are being visualised in the app. Interacting with the application creates a demand for updating as well (Grov,2015). That is why there is a high demand for Javascript Frameworks because it makes the process of updating easier.

The rise of JavaScript frameworks and large-scale applications makes it interesting to ask how the virtual DOM compares to other DOM update mechanisms in JavaScript frameworks.

2. Background

The discussion about the virtual DOM and the question about how advantageous it is for web developers has been an ongoing discussion. Comparing the usage of the virtual DOM with techniques used by other frameworks and offering the advantages and disadvantages of using it could be beneficial for web developers and make it easier for developers to find a JavaScript framework most suitable for their projects.

What is also important to note is that Facebook open-sourced React. It is common to open-source a JavaScript Framework. One of the benefits of this method is that the framework can be made better by other web developers or engineers. The framework will keep evolving because of the higher chance of someone being able to add new features and to fix bugs (Costa,2016). Open sourcing frameworks also means that smaller companies have the opportunity to use a particular framework too, and it is the reason why, apart from Facebook, other companies, either big or small can start using React too.(Grov, 2015).

2.1 History of JavaScript and emergence of frameworks

In order to understand why Javascript frameworks were created, it is important to discuss the history of Javascript first. In 1995, Brendan Eich created Javascript in just ten days (Miller, 2018). The language has played an important role in web development ever since. It is also important to notice that Javascript was created only four years after the World Wide Web became available to the public. Around that time, at least thirty seconds were needed to open a new web page and access to internet was made possible via telephone-line modems. That is why filling out forms was a difficult process, forgetting to fill out a field on a form resulted in having to wait for the computer to sent a request to the server, after sending the request the server would then send a response which would show the error. This illustrates how difficult it was to get something done in a reasonable amount of time because a computer would have to correspond over slow internet for it to be able to complete a task.

Javascript was created to solve this problem, it became possible to check whether required fields were filled in or not before sending a request to the server. Users were informed that a

field was blank first and this made sure that the communication between their computer and server began after everything was filled out. Soon after that, Javascript become more widely used for many other tasks. The programming language has played an important role in making the internet a more modern prospect. Websites became more interactive because of the invention of Javascript. All websites on the internet are using basic HTML. A website can be built by only using HTML, which would be called a static website. It is not possible to interact with a static website. Introducing Javascript as a client-side programming language made it possible for websites to change without having to send a request to the server (Miller,2018).

In 2006, jQuery was released, a JavaScript library. The main goal of jQuery was to help developers solve issues with differences in web browser implementations (Barker, 2018). This library was described as functional, clean and easy to learn. After the creation of jQuery, developers performed tests on jQuery's maintainability.

Jeremy Ashkenas found that large jQuery projects became chaotic (Barker, 2018). In 2010, Backbone was created by Ashkenas, its main goal was to create single page applications. jQuery was not required for Backbone as it was an independent framework, but it was possible to create certain functionality together with jQuery. During that same year AngularJS was created by Adam Abrons and Misko Hevery (Barker, 2018). The reason why AngularJS is now used by Google is because Hevery started working for the company. AngularJS became the first application to be used for front-end development. What was special about AngularJS at that time was its main feature of bi-directional data binding which enabled to bind a model's data to HTML markup, this made real-time updates and changes possible (Barker, 2018).

Knockout (Barker, 2018) had a similar approach to AngularJS when it comes to bi-directional data binding, although it can be said that AngularJS has a broader functionality. This Javascript library uses a Model-View-View Model (MVVM) (sorensen and Mihailescu, 2010) architecture. The MVVM architecture makes the process for web developers to create dynamic and interactive user interfaces with a logical underlying data model easier. Besides Knockout and AngularJS there is another framework called Meteor (Barker, 2018). It is an open source platform that can be used for creating web, mobile and desktop applications. Meteor uses Node.js

Facebook made its own framework called React public in 2013. It has gained popularity ever since. The main reason for the creation of React was that it needed to fix certain bugs for Facebook. React has been gaining popularity ever since, resulting in creating an enormous community in the JavaScript world.

In 2014, Evan You created Vue (Copes, 2018). when looking at the following numbers, it can be said that Vue is a framework preferred by many developers. In 2017, Vue had 36,700 stars on Github, the number increased significantly compared to 2016, where it received 7,600 stars on Github. Vue is called a progressive framework, that is, the framework adapts to the needs of the developer. Vue can be easily introduced into an app by simply including a script tag, more can be added according to one's needs.

Yehuda Katz and Tom Dale created Ember, this framework was initially released in December 2011. It was inspired by Sprout core, which is an open-source JavaScript framework. The main aim of Ember is to create projects with the least amount of people, the main idea was to make sure that projects that need more than twenty people to be able to create it, could be done with less than three people with Ember. Katz and Dale also wanted to create something that was easy for beginning web developers to learn. It is believed that Ember takes front-end to the next level by creating a framework allows better coding structure and allows making a more powerful and lively application by ensuring a better UX design, because this ensures bigger growth in the community (Honeypot, 2019).

Ember was not expected to become such a popular framework, and eventually became the underdog of the front-end frameworks (Honeypot, 2019). There are several reasons why ember.js became this successful. This framework designed templates that update automatically when the underlying data changes, this allows for developers to write less code. Ember is based on the Model-View-View-Model. This framework eliminates writing repetitive code. Experienced developers have more options to write mobile and desktop applications with this framework.

Microsoft, LinkedIn and Intercom are three companies using Ember, but there are others who are using Ember for front-end projects. Ember's message to the world is that they wanted to create something that when something goes wrong, the blame does not go directly to the developer, which is excellent for beginners.

2.2 History of the virtual DOM

In order to understand why the concept of the virtual DOM became popular in the JavaScript community and the front-end world, it is necessary to discuss the original DOM first. DOM stands for Document Object Model and consists of two parts. The first part is the object-based representation of the HTML document, the second part is the API, which stands for application programming interface, the API is used to manipulate the object, specifically from JavaScript.

The exact definition and explanation of the DOM was released by The World Wide Web consortium (W3C)[7], an international association promoting and developing standards for the World Wide Web. Their aim is to guarantee a continuous growth of the Web. It is also the W3C who invented the name Document Object Model for this specific term. W3C found Document Object Model a suitable name because the DOM is perceived as an object model in the concept of traditional object-orientated programming, that is, a pattern of programming where the solution is modelled as a collection of collaborating objects (Eng, 2017). The DOM is a model for a HTML document, which consists of a hierarchical set of objects. Each object corresponds to a HTML element in the document, and has attributes that correspond to HTML element attributes. The DOM illustrates the structure of the document and the behaviour of its elements. Documents are modelled and use objects, the model illustrates the structure of the document and depicts the behaviour of its objects.

DOM documents are mainly depicted as a tree structure. Standard HTML can be inserted into a tree structure, as shown in figure 1. This figure illustrates the relationship between different tags within a HTML document, the tree represented below is a similar concept to that of a family tree. The first tag on top of the page is the <HTML> tag and is called *the root* of the tree. Underneath the root are the children of the root tag. In this particular image the children are the <BODY> and <HEAD> tag. It is possible for child tags to have their own children and this is shown in the example below. The <h1> and <P> tag are children of the <BODY> tag, and the <TITLE> tag is a child of the <HEAD> tag. When two child-tags have the same parent, then the term siblings is given to those multiple tags.

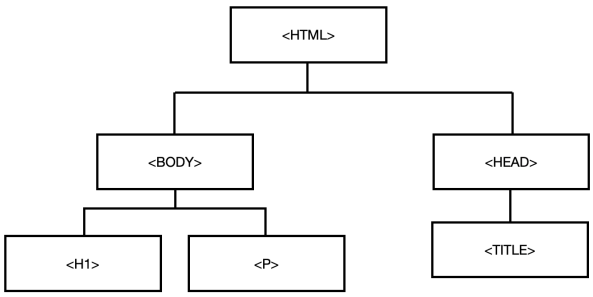

Figure 1: DOM tree

The first specification of the DOM was created in 1998 when building and managing web pages was done differently compared to now, this was mainly because there was no immense dependence on the DOM's API's to create and update page content as frequently as is

[7] https://www.w3schools.com/js/js_htmldom.asp

necessary today (Aderinokun, 2018). In the past, JavaScript provided a number of ways to interact with the DOM, such as *document.geElementsByClassName()* and *document.getElementById()*. However it is only beneficial to use simple methods such as *document.getElementsByClassName()* on a small scale. The main problem occurring nowadays is that the web applications used today need constant updating, and updating the DOM regularly can become expensive. Furthermore, it is easier to update larger parts of the document rather than to update specific elements, this is due to the structure of the API's (Aderinokun, 2018). Figure 2 shows an example of a simple HTML document, within the document is an unordered list with a list item included.

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    </head>
<body>
  <ul class="unordered_list">
    <li class="list.item"> This is a list item</li>
  </ul>
 </body>
</html>
```

Figure 2: example of HTML document

For example, when the content in the <li> tag is required to be changed, then the first thing that needs to be done is for the DOM's API's to search for the elements that need updating. Afterwards new elements will have to be created, attributes and content will need to be added too. After all these steps the DOM elements can finally be updated. This technique is far from efficient on a larger scale because the elements need updating multiple times in a minute and will therefore make this technique too expensive to use.

There are two main reasons that contribute towards making the DOM more slow. The first main reason is when a script triggers an enormous amount of reflows or repaints. The second reason is when it takes too long for a script to locate a specific node in the DOM tree (Wilton-Jones, 2006). Repaint is a term that can also be called redraw and occurs whenever something becomes visible that was not visible before, and oppositely (Wilton-Jones, 2006). The following example will illustrate what repaint means. When the user changes the background colour for example, it will not make a huge difference to the whole web page, but when the width of a certain element gets changed, the rest of the web page will need to be adjusted. It is then the role of the engine to find the certain element to check where everything needs to be displayed on the layout of the page. If the parent elements are changed then the child elements also need adjusting. The elements that come after the changed element in the DOM need to be reflowed

in order to be displayed in the correct way. Ancestor elements will also need to be checked when their child elements are changed. The last step is for everything to be repainted (Wilton-Jones, 2006).

One of the main reasons why the DOM scripts are slow is because reflows can become expensive because of the performance rate. It is an especially slow process when it runs on devices with low processing power, a phone is an example of one of those devices, the reason why it is particularly slow on phones is because in many circumstances the process is similar to laying out the whole page again (Wilton-Jones, 2006).

A repaint happens when changes are made, for example the change of an outline, visibility or background colour (Sullivan, 2009). Repaint is expensive because all nodes in the DOM tree must be verified by the browser on visibility. Reflow is equally bad for performance because it needs to make changes that affect the layout of a page. When a reflow of an element happens it causes the next reflow of all child and ancestor elements plus the elements following those elements in the DOM (Sullivan, 2009). Figure 3 shows a simple HTML document. A reflow on the <p> tag would mean that it would trigger a reflow on the tag, in some browsers a reflow might also occur on the ancestor tags, in this case on the and <body> tags. The <h5> tag and the <ol> will be reflowed just because they follow the <p> tag in the DOM. Reflows often cause re-rendering (Sullivan, 2009). Reflows are mostly caused when writing CSS. Resizing the window, changing the font, adding or removing a stylesheet, a user typing text in an input box are some examples of other situations that can cause a reflow (Sullivan, 2009).

```
<body>
<div class="error">
        <h4>My Module</h4>
        <p><strong>Error:</strong>Description of
the error…</p>
        <h5>Corrective action required:</h5>
        <ol>
                <li>Step one</li>
                <li>Step two</li>
        </ol>
</div>
</body>
```

Figure 3: simple HTML document (Sullivan, 2009)

Changing the virtual DOM is cheaper because only memory is being changed, this makes the process faster. When a developer uses a JavaScript framework such as React, then there is no

need to worry about what needs to be updated, the framework handles that for the developer. The main reason why JavaScript frameworks are highly desirable for productivity nowadays is because it keeps the UI in sync with the state (Gimeno, 2018). Writing UI's as complex as they are nowadays is too complicated to write in Vanilla JavaScript.

It is not efficient to frequently update the original DOM and that is why the virtual DOM was created (Aderinokun, 2018). It is developed to improve the performance of regularly updating the DOM. the virtual DOM is a JavaScript object representation of the original DOM (Aderinokun, 2018). It can be seen as a new method of interacting with the original DOM rather than defining it as an official specification. The virtual DOM can be understood as a copy of the original DOM, this copy can be updated and manipulated multiple times without the usage of the DOM's API's. Working with the virtual DOM is also more straightforward, the actual DOM has a lot of redundant properties. The original DOM will only be updated after making all the updates on the virtual DOM.

Frameworks such as React or Vue adopted the technique of the virtual DOM to improve the performance of updating the original DOM (Aderinokun, 2018). The virtual DOM and the diffing process, which will be explained in chapter three of this research project, make these frameworks desirable to use.

2.3 Why React was created as a language and implementation of the virtual DOM

React's story begins at Facebook's ads organisation. The biggest issue at the ads organisation was trying to keep the user interface synchronised to the business logic and the state of the application (Smith et al.,2016). Facebook tried to solve this problem by using a centralised event bus, where it was either putting events into a queue, or by having listeners (Smith et al., 2016), that is, a procedure or function in a computer program that waits for an event to occur, for the event without doing anything else. It soon became clear that this was not an efficient way of working. Facebook then decided to start using a DOM-monitoring system called Bolt, which had its own flaws. The problem with Bolt was that each time a value was changed, it remained unsure what kind of update it was going to make, a single update, cascading updates, or no updates at all (Smith et al., 2016). In addition, it was difficult to determine when those updates might occur. There was a need for something that would connect the change handlers, that is, code that should be run after a detection of change, and that would be easy for engineers to understand. Bolt was not solving those problems and on top of that, it was creating a lot of bugs that were almost impossible for engineers to fix. All of the problems mentioned above contributed towards the start of the idea behind React (Smith et al., 2016).

Facebook was in need of a framework that could serve large scale user interfaces, with data that changes over time. This describes why React is useful to Facebook engineers, it was designed to solve problems that were bound to occur while building complicated UIs (Smith et al., 2016).

Implementing the virtual DOM in React was a smart approach of Facebook because it offers a wide range of benefits. The first benefit of using the virtual DOM is that it makes application development easier. The virtual DOM eliminates having to use dirty checking, a technique used by another Javascript framework called Angular (angular.io, 2010). Dirty checking is also called the digest cycle and is a change detection technique, the process can be treated as a loop which checks if the data values in the code have changed. This approach is different to the virtual DOM technique because the dirty checking technique only updates the parts that have changed (Seshadri and Green, 2014). The virtual DOM looks at what needs to be updated, rather than checking what data has changed or comparing old values to new values.

The following reasons explain other benefits for using React and therefore using the virtual DOM. There are less details that need to be taken care of by a web developer, there is no need to look at what needs to be updated, the framework takes care of that automatically. The last benefit that will be discussed here that the virtual DOM makes it possible for the developer to focus on one main task: explaining how the UI should look, the rest is being dealt with automatically.

2.4 Summary

This chapter described the history of JavaScript and looked at how JavaScript frameworks emerged. Chapter two discussed the development of the concept of a virtual DOM and looked in particular at the development of the React framework.

3. The virtual DOM

3.1 Previous work about the virtual DOM

The virtual DOM has become popular due to the creation of React by Facebook. Research on the virtual DOM was ongoing even before the creation of React (Psaila, 2008), but it can be stated that more research sources started to appear after Facebook's React. The next section will discuss what has been previously researched, most sources have investigated the performance (Grov, 2015), impact, learning curve of the virtual DOM (Naim 2015) and experiences with JavaScript Frameworks (Greif, Benitte and Rambeau, 2018). Overall there seems to be a general agreement that the virtual DOM is a more efficient change detection mechanism when building large scale applications.

Guiseppe Psaila (Psaila, 2008) has published a paper in the 2008 19th International Workshop on Database and Expert System Application about the the virtual DOM. Psaila believes that the standardised main memory representation methods, that is, the DOM, are not efficient enough to represent large XML documents in terms of amount of memory space and the amount of time it takes to load large XML documents in the main memory. Psaila proposes the following solution to this problem, using a virtual DOM, as this is believed to provide an efficient representation technique for large XML documents. The reason why this appears to be efficient according to Psaila is because it adopts a specifically designed virtual memory technique. A comparative experiment is carried out and mentioned in the paper, this experiment compares the virtual DOM to a standard DOM package. The results of the experiments show that the approach followed to design a virtual DOM is effective for large and very large documents because it avoids the trashing phenomenon, trashing refers to the problems that occur when a computer's virtual memory resources are overused. This trashing phenomenon is usually caused by the virtual memory mechanism provided by the operating system which is not optimised for that specific problem.

Naimul Islam Naim wrote a thesis in 2017 (Naim, 2017) on front-end frameworks. The thesis is stated to be an in-depth research of the ReactJS library based on Javascript. Overall the thesis gives clear instructions on how to start with React, what features and functionalities React has and when to not use this framework and choose another suitable framework. It is stated in the thesis that ReactJS became popular in a short period of time. The framework is also known for being able to handle a large amount of users and data. Naim concludes that React brought a new dimension in web application development. The thesis specifically mentions compared to other frameworks React has a fast rendering library and this is beneficial for application efficiency.

Marianne Grov has written a Master's thesis in 2015 (Grov,2015) and found the following benefits of using the virtual DOM. when looking at performance it can be stated that the virtual DOM is a better alternative than dirty checking. According to Grov's research, performance issues arose when using dirty checking for large applications and complex situations. The virtual DOM demonstrated the best results in simplicity and scalability and has been a contribution towards UI development at a higher abstraction level. Grov's research has concluded that getting the developer to make expansions non automatically does not necessarily achieve good performance. This study has also shown that JavaScript's usage areas have been expanded in the front-end development world. Lastly, Grov also concludes that the fact that Facebook is

using the virtual DOM has made this type of change detection mechanism more popular, this company was also the one that introduced the benefits of the virtual DOM into UI development.

Sacha Greif, Raphael Benitte and Michael Rambeau have created a website called State of JS (Greif, Benitte, Rambeau, 2018). They have sent surveys to around 20,000 JavaScript developers to ask about their experiences with JavaScript frameworks. The main aim of the survey was to find out which frameworks the participants were using the most, which ones they are the happiest with and which frameworks they are most eager to learn. The results are shown on the website and are supposed to help in having an overview of the numbers and insights of all JavaScript frameworks. The most recent statistics are from the year 2018. The results from 2017 and 2016 are also available on the page, it is beneficial to be able to compare between the different years. The website does not only show the survey, there also useful links to help web developers with the learning process of new frameworks.

3.2 virtual DOM vs other mechanisms

There are multiple Javascript frameworks that use a different change detection mechanism than the virtual DOM. One of those is Angular, a framework made popular because of Google that uses a method called dirty checking which is also called the digest cycle. The aim of the digest cycle is to keep the UI up to date in an application made with AngularJS. Snapshots of data are taken over time and are being compared and checked for changes. The reason why it is also called dirty checking is because it scans the scope for changes as all watched variables are in a single loop, this loop is called the digest cycle, any value change of any variable forces to re-assign values of other watched variables in the DOM. Angular's detection mechanism will be broken down in a couple of steps to gain a better understanding of the process.

From the moment an application or an HTML part is loaded within AngularJS, the framework starts running its compilation steps and keeps track of all the watchers and listeners that are needed for the HTML. If the watchers and listeners are linked with the scope, that is, the binding part between the HTML and the Javascript, the current values will then start to be displayed in the UI. It is important to notice that AngularJS pays attention to each of the elements linked to the HTML, this is done for each element for each scope.

The digest cycle will start to get triggered once an activity is performed, for example when a user clicks a button. This activity also changes the model value. AngularJS begins in the digest cycle with the *$rootScope,* each watcher will check in the scope to figure out whether the current value is different from the value that is being displayed in the UI. In other words, Angular starts comparing the old value to the new value. When there is no change between the old

value and new value, the digest cycle will repeat the process for all parent scopes until it has verified all the scopes.

When the values are different, this means, when a watcher is found that reports a change in state (Seshadri and Green, 2014), AngularJS gets interrupted and the digest cycle will start to run again. This process happens to make sure that the watcher that was already checked before did not get implicated by the change found in that particular watcher afterwards. The digest cycle reruns to ensure no change in data is missed (Seshadri and Green, 2014).
The digest cycle will re-run every time a change is encountered. It will stop running once the digest cycle is stabilised. This process can create one particular problem, that is that the digest cycle will end up in an infinite loop, this problem is avoided by setting the digest cycle to rerun ten times by default. This process takes averagely two to three cycles for a normal AngularJS application. When the digest cycle stabilises, the framework gathers all the UI updates, those will then get triggered all together at the same time (Seshadri and Green, 2014). Figure 4 illustrates the process of the digest cycle.

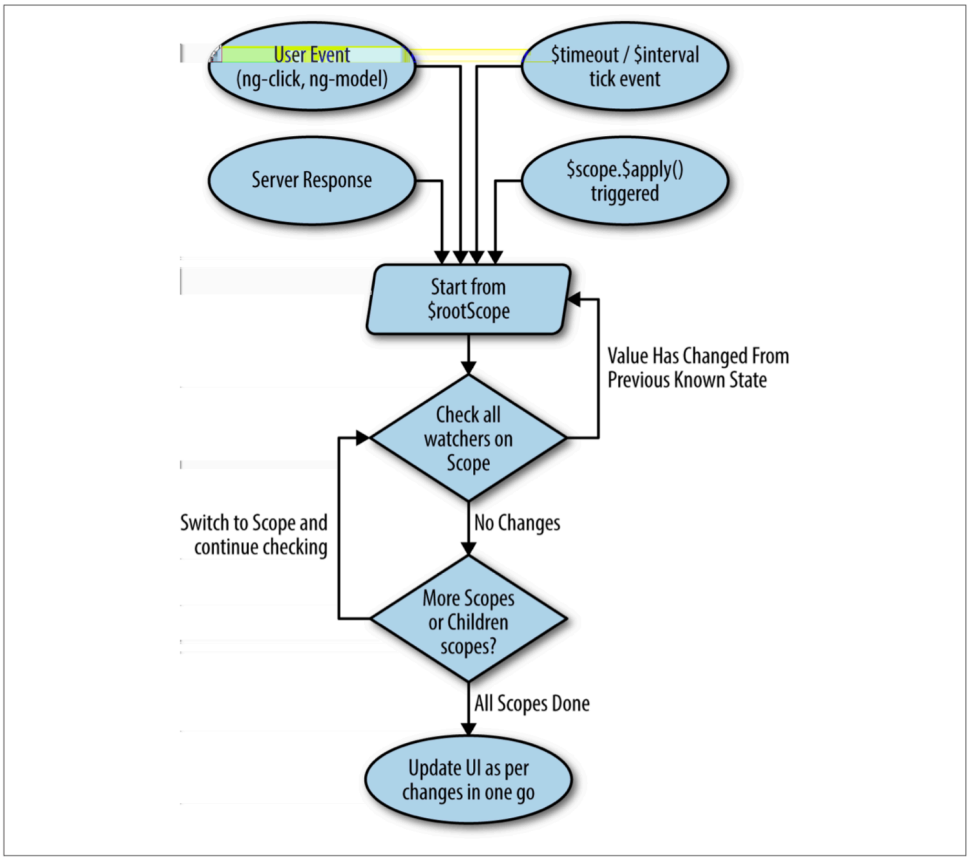

Figure 4: process of the digest cycle (Seshadri and Green, 2014).

the digest cycle starts with the help of calling two functions, the *$watch()* and *$digest()* functions. If the values are the same, nothing happens. In case the values are different, the digest cycle goes through all the scope objects like Angular expressions or directives and checks which objects got affected due to the activity performed by the user. The change in the objects will then be given to the watchers, that will update the view as per model value change, the digest cycle will then run again to check if all values are synced up (Seshadri and Green, 2014).

Angular uses two-way-data-binding, this means that when properties in the model get updated, the UI gets updated as well, when the UI gets updated, the changes will get back to the model (Grov, 2015). In order to better understand the process of the digest cycle, the functions $watch(), $digest() and $apply will be explained more carefully. These functions are also called the Angular $scope functions.

firstly, $watch() will be explained. This function watches the changes in scope objects. Scope objects are the binding parts between the HTML and the Javascript. The watch expressions are automatically attached to the objects under the scope. This function accepts three parameters: expression, listener and equality object. Listener and equality object are optional parameters (Grov, 2015).

The $digest() function takes care of the performance of the dirty checking process. It used to re-evaluate all the watchers in the scope object and its child scope objects and it may cause performance in the large scope hierarchy. This function is not called on operations such as a like button or on an AJAX call. It is only called when Angular thinks it's necessary (Grov, 2015).

The $apply function is mainly used when a change in any model outside the Angular context happens, informing needs to happen by calling this function manually. When this function is finished the framework will call the digest function internally, and as a result all data bindings are updated (Grov, 2015).

Ember.js uses a technique called data binding. Ember sends out events from the data model when changes occur. It is possible to bind the UI to the data model, this allows for a listener to know when to update events linked to the UI. When changes occur only the part of the app that need to do something will be activated (Cravens and Brady, 2014).

There are downsides when using one of the mentioned above change detection mechanisms. It has been argued that Angular's dirty checking approach results in data binding limits (Turnbull, 2014). Another argument is the use of the $scope() functions. It can be difficult for a web-developer with little experience with AngularJS to debug those scope functions (Rajput, 2016).

There is another approach that is often used and is called Key-value observing (KVO) (developer.apple.com2019), this technique notifies the application when a change is made by the user. The changing of a text field, variable value are some examples of those possible changes. KVO is efficient in the following situation: it is particularly used in the communication between model and controller layers in an application. The task of the controller object is to observe properties of model objects, the view object inspects properties of model objects through a controller. The benefits of using KVO in a web application will be illustrated in the following example. It can be similar to a person and their bank account. The person is the object that interacts with the other object. The account is the person's savings account at a bank. When the balance or interest rate of the account changes, the person object needs to be made aware of those particular aspects(developer.apple.com2019).

The KVO technique means that the object, person, will receive an interrupt when certain details of the account changes. To use the KVO technique, objects must be KVO compliant first, that means it must contain certain requirements (developer.apple.com, 2019).

Key-Value observing is used by iOS developers, it is used in a programming language called Swift, designed by Apple. Swift has a different approach to other frameworks because it requires reactive programming, that is, a programming language where, when a change is made, all the objects depending on that change need to be notified and need to respond according to the newly made value (Costa, 2016).

KVO is also not an efficient technique because it requires more work from the developer's side. The underlying pattern of KVO does not make the process of maintaining and debugging easier (Sandofsky, 2016). It was similar to the use of Object.observe in Vanilla JavaScript, which has failed to be useful and has become redundant (MDN Web Docs, 2019).

3.3 Case study: learning JavaScript programming using a virtual DOM

This section will give an example of a Javascript project made as part of an assignment for the MSc in Interactive Digital Media. The goal was to create a game called Sokoban, which is a Japanese puzzle video game. Figure 5 shows an example of the game. This game will give an

example of how the technique of the virtual DOM can be used in a JavaScript project for beginners. This project will be compared to the technique used in React.

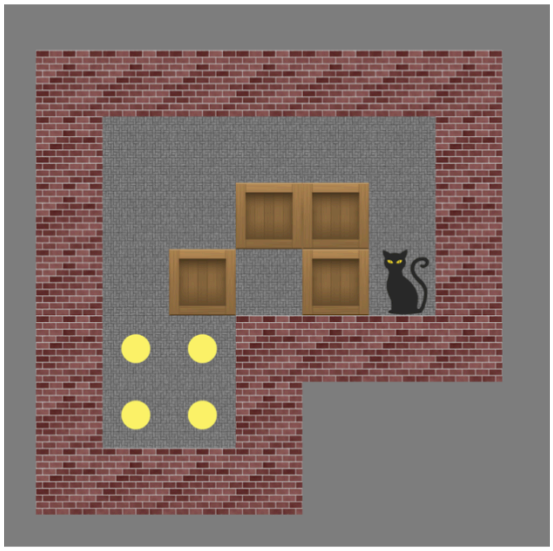

Figure 5: example of Sokoban game

There are two arrays in the javascript code. One array is the original board, the second array is the board that will get updated once the player starts playing the game. Figure 6 shows the code for the two arrays. The # symbol stands for the walls in the game, F stands for floor, the $ sings are the crates that the player needs to move, the dots are the goals the player will have to move the crates into and the @ symbol is the player. There is an *boardLOriginal* and a *boardL* for each level.

```
var boardL1Original = [
    [ "#", "#", "#","#","#","#","#" ],
    [ "#", "F", "F","F","F","F","#" ],
    [ "#", "F", "F","F","F","F","#" ],
    [ "#", "F", "F","F","F","F","#" ],
    [ "#", ".", ".",".","#","#","#","#" ],
    [ "#", ".", ".","#"," "," "," " ],
    [ "#", "#", "#","#"," "," "," " ],

];

var boardL1 = [
    [ "#", "#", "#","#","#","#","#" ],
    [ "#", "F", "F","F","F","F","#" ],
    [ "#", "F", "F","$","$","F","#" ],
    [ "#", "F", "$","F","$","@","#" ],
    [ "#", ".", ".",".","#","#","#","#" ],
    [ "#", ".", ".","#"," "," "," " ],
    [ "#", "#", "#","#"," "," "," " ],

];
```

Figure 6: code for the two Sokoban arrays

The second array used in this particular project is similar to the virtual DOM technique, the second array is used to update the DOM, that particular second board that is created is similar to the virtual DOM and the render() function that is used is the reconciliation function. Figure 7 shows the code for the render function. This function is then called after the array of the board needs to be updated. There are several situations where the board needs to be updated. One example is when the player succeeds in putting a crate into a goal, the crate will change colour to clarify that it is in the right place.

```
// function to render the board
//This function also sets the walls, player, crates and floors in the table
function render() {
    $("#container").empty();
    var table = document.createElement("table");
    $(table).attr("border", 1);
    $(table).attr("cellpadding", 20);
    for (var x=0; x<board.length; x++) {
        var row = document.createElement("tr");
        for (var y=0; y<board[x].length; y++) {
            var cell = document.createElement("td");
            var img = document.createElement("img");
            if (board[x][y]==="#"){
                $(img).attr("src", "images/wall.jpg");
            } else if (board[x][y]==="$"){
                $(img).attr("src", "images/crate.png");
            }
            else if (board[x][y]==="F"){
                $(img).attr("src", "images/floor.png");
            }
            else if(board[x][y]==="@"){
                $(img).attr("src", "images/player.jpg");
            }
            else if(board[x][y]==="."){
                $(img).attr("src", "images/goal.jpg");
            }else if(board[x][y]==="*"){
                $(img).attr("src", "images/darkcrate.png");
            }


            $(img).appendTo($(cell));
            $(cell).attr("id", `${x}:${y}`);
            $(cell).appendTo($(row));
            if(board[x][y] === " ") {
                $(cell).empty();
            }

        }
        $(row).appendTo($(table));
    }
    $("#container").html($(table));

}
```

Figure 7: render function

The render function in this project generates the HTML for the board, and re-renders the whole HTML every time. The main issue with this function is that it does not check whether a value has changed or not, it keeps re-rendering even if two values are the same. For example, the F symbol in the function represents the floor of the board in the HTML, when the F symbol is found in the array, an image of a tile will be rendered to the HTML. When the board needs to be re-rendered, the function will not check whether the image of the tile was changed or not, it will re-render automatically and will not check the difference. It is possible to create a diffing algorithm, but that would require a lot of work, and that is why JavaScript frameworks are beneficial.

This approach is similar to what is used In react, the render() function allows you to pass in a template and the element to render it into, and that function will handle the rest. The Sokoban project shows that something similar can be created with vanilla javaScript as well. This needs to be checked inside the code, afterwards the render function is called. For example when the player manages to move the crate inside one of the goals, the crate changes colour. The new crate will have to be showed in the HTML.

One major difference and problem with the render function in the Sokoban example is that using this particular function on its own can be rather slow. Everything is redrawn, instead of only redrawing the changed parts. React implements a solution to make this process faster.

React uses reconciliation to solve this problem, it is also called the diffing process (Reactjs.org, 2019). A previous rendered tree will need to be maintained and compared to the newly created tree, this is the process of keeping one's own version of the DOM, which is called the virtual DOM. the diff algorithm is used to search for the smallest changes between two virtual DOM representations (Grov, 2015). What is important to note is that this algorithm needs to be incorporated into the code efficiently in order for it to give well-performing results (Grov,2015). React is the framework with the biggest virtual DOM implementation today, and it forms a good example of how the diffing process is done because the aim of this algorithm is to execute a minimal number of steps to go from the UI render that already exists to the next UI render (Grov, 2015).

The render() function can be seen as a tree of React elements (Reactjs.org, 2019). This function will return a different tree of React elements after each state or props update. It is then up to React to match the UI with the most recent tree. React uses a technique called the diffing algorithm, it compares one root element with the other first. When two root elements have different types, React will change the old tree completely and make a new tree. When this happens, all old DOM nodes are destroyed and the new tree will insert new nodes into the DOM. When two React DOM elements have the same type, React will look at both attributes and will keep the same underlying DOM node. The part that gets updated are the changed attributes.

One reason why this concept is a good idea is because web developers do not need to have a detailed overview of which changes are made every update. Reconciliation makes the process of writing applications easier.

3.4 Summary

This chapter described how the virtual DOM works and looked at prior research in this area. It also compared the virtual DOM to other methods of change detection. Finally, this chapter looked at a case study of the Sokoban project to illustrate how a novice programmer might find the concept of a virtual DOM useful.

4. Design and methodology

4.1 Methods used

This research project will use an online survey and will analyse the responses to explore the question of whether the virtual DOM is a better technique compared to other updating mechanisms in other frameworks. The results of the online survey will then be compared to the research mentioned in previous chapters.

The topic of the virtual DOM has become popular over the years. articles , blog posts and presentations (Kussain, 2018) have been widely published by web developers who work within the field. Research has been carried out to gain further insight on this topic and to gain understanding as of why the virtual DOM has received this much attention over the years.

The aim of this research project is to analyse the use of the virtual DOM. The method will be compared to other alternatives. Research has been carried out to determine whether the virtual DOM is a good way of simplifying the process for web developers. This research paper will also examine the impact of the virtual DOM in front-end development. This topic has become popular within the JavaScript community, and articles, blog posts and presentations are regularly published by developers within the field. Comparing the virtual DOM with other DOM change detection mechanisms and offering potential benefits and disadvantages of working with the virtual DOM could be beneficial for web developers and will possibly make their choice on which JS framework to choose easier.

Qualitative methods are used to gain insight and understanding of the experiences and perspectives of the candidates. Collecting qualitative data is beneficial for this research project because the main aim is to research whether the virtual DOM is a suitable tool for web developers, therefore a subjective opinion is needed to gain more information about the experiences of web developers with regard to the virtual DOM. The online survey will reveal trends in thought and opinions among the participants. Qualitative research allows for deeper exploration of the research question, the open-answer questions will allow developers to explain their needs and articulate any frustration or fascination with the virtual DOM. The open-answer questions will also help to discover why React and the virtual DOM is popular.

4.2 Explanation of surveys

To gain more insight on this topic, an online survey has been carried out to identify the learning curve and the experiences of web developers with the virtual DOM. This online survey has been carried out to strengthen the conclusion on whether the virtual DOM makes the process easier for web developers. Online surveys were sent out to web developers and software engineers who have experience in working with React or the virtual DOM. Gender does not matter for this type of research but considering the field, it will be a predominantly male public. Most participants work in Ireland and or have the Irish nationality. The online survey consists of ten questions with an open-answer format. The questions are in regard to the participant's experience and opinion about the virtual DOM and their experience and knowledge on Javascript Frameworks. Participants can choose how much they want to elaborate on each question. The interview questions were available on Google Forms, a link was send out to all participants willing to fill out the survey. Participants were found by sharing the online survey on LinkedIn, Facebook and Reddit.

The current study was subject to certain ethical issues. All participants were required to read a consent and information form and to confirm they have read and agreed to the consent form before proceeding to the questions. Participants were fully informed regarding the objectives of the study. The online surveys require collecting personal information of participants. The survey is hosted on Google forms and is fully compliant with GDPR. Personal data will be collected of participants. Age ,gender and email addresses are collected and is necessary for the following reasons. The email address is necessary in case of needing a follow up after the online survey has been filled in, it is possible that certain answers need more explanation or were not entirely clear. The age of the participant is necessary to gain more insight on their experience with JavaScript Frameworks. Participants will be asked what their gender is to be able to have an overview of the male to female ratio.

The online survey is divided into three main sections. The first section contains of background questions to ask the participant about their level of experience as a front-end developer. the second part asks questions in regard to pre-virtual DOM and how the virtual DOM has changed the way of coding compared to the pre-virtual DOM times. In the last part the participant is asked to answer questions that ask to compare the virtual DOM to other change detection mechanisms of other JS frameworks.

Questions about React are asked because this framework uses the virtual DOM, it is therefore easier to ask participants questions about the framework in regard to the virtual DOM. Although

there are other frameworks that use the virtual DOM technique, React appears to be the most popular framework.

4.3 Analysis on how answers are going to be used

Although there are a lot of resources to be found on the virtual DOM, it remains difficult to understand what the real benefits are without asking the experience of web developers who work with the virtual DOM on almost a daily basis. The answers of the online survey will therefore be used to strengthen the arguments made in this research project.

The survey results from the web developers will strengthen the understanding of the efficiency of the virtual DOM. The answers of the participants will provide insights on whether the virtual DOM scores better in performance, learning, flexibility and adaptability compared to what is used in other frameworks.

Survey results are shown in the following chapter of the research project and will be compared to the theoretical research from the previous chapters. The results will be implemented in this research project. The research and the survey results will be compared to each other and check whether the findings in the research project are matching the experience of the web developers. All respondents are web developers who have experience with working with the virtual DOM and with React. The survey questions can be found in appendix B.

4.4 Summary

Chapter four described the survey methodology and the type of analysis that will be carried out on the responses.

5. Evaluation and analysis

5.1 Results of survey

The results of the online survey will be presented in this chapter. The outcomes of the survey are presented for each question in the online survey. It is possible that direct quotations will be used in the discussion.

Out of all the respondents, 13.3% of the participants are female, 86.7% are male. The participants were aged between 21 and 30 years old. Participants have at least one year of experience in front-end development. All participants have experience in working with the virtual DOM, the amount of experience ranges from six months to four years.

The survey asked to list the frameworks the participants have experience in. Apart from React, in which all developers have experience in, there are a number of other frameworks that the participants have written down. Angular appears to be the second most used framework among the developers. The reason why this question was asked was to ensure the participants were able to compare the virtual DOM in React to other update mechanisms. The chart below illustrates all the frameworks listed by the participants.

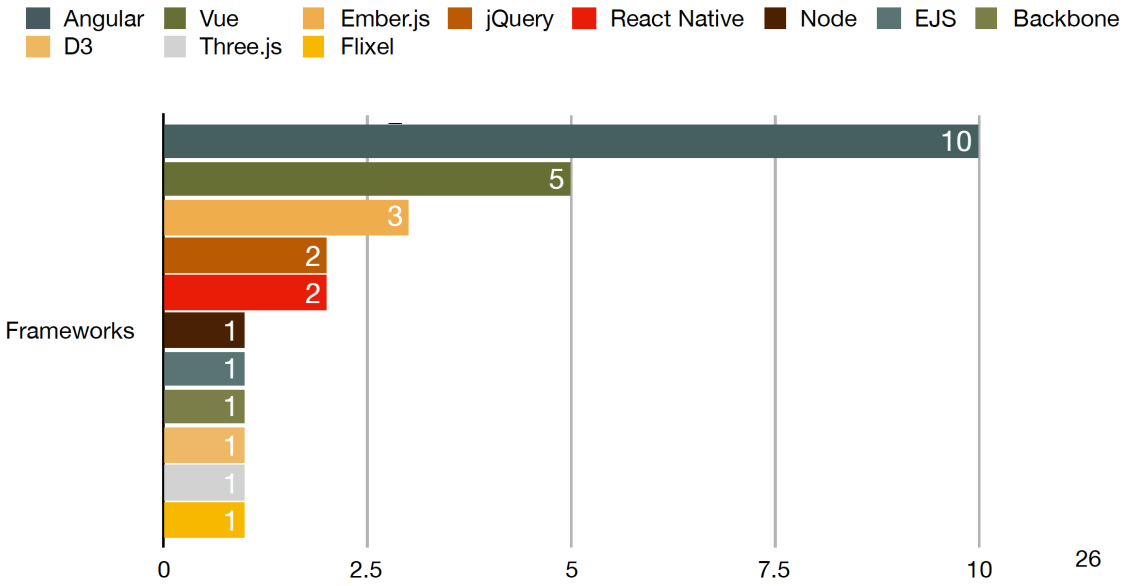

Figure 8: knowledge of JavaScript frameworks

Participants were asked to give advantages and limitations of the virtual DOM, specifically when used in React. The table below gives a summary of the replies of the participants. The main limitation seems to be the learning curve of the framework, compared to simpler libraries such as EJS or jQuery. Figure 8 shows a table with a summary of the responses of the survey.

Advantages	Limitations
Better data structure and component support	Higher learner curve compared to something like EJS
as it's just a view library, allows it to slot into a lot of different technical stacks and be quite flexible.	the lack of best practices to use it in can be an issue, especially for people learning it from scratch.
frequent updates resulting in new methodologies	constant learning (which is also a good thing, but its quite fast paced)

Advantages	Limitations
React's top-down data flow helps keep big apps consistent. Whenever data changes, it's fed to everywhere that wants to know about it. When you want to mutate data, it goes all the way back up to the source of the data. This helps you have a "single source of truth" for your data, keeping disconnected parts of your UI from falling our of sync with one another.	I haven't felt any limitations with React. The big downside is needing a compiler for JSX which complicates the getting-started process.
Easy to right nicely decouple components and no need to track what changed and mutate what you need	Was long and tidious to write
faster, declarative	one way state to render calculation
Quick render times without thinking about it	

Figure 9: advantages and limitations of the virtual DOM

The survey asked to specify how long it took to get used to working with the virtual DOM. The overall response of the participants was less than a year. Only a few responded it took more than one year.

Participants were asked to identify any problems that they experienced by using change detection mechanisms of other JavaScript frameworks and had to compare this to the virtual DOM. Overall, it seems that the virtual DOM makes debugging and the handling of bugs easier, performance increases, better error feedback and the updating of the real DOM happens only when necessary. The following table shows the answers of the participants.

Benefits of the virtual DOM compared to other frameworks
Debugging is easier, gives better error feedback
Updating the DOM when data changed and ensuring it's handled everywhere. Lots of boilerplate code that wasn't easily abstracted
performance and efficiency of work
When you have a large app where multiple independent parts of it rely on the same data, it's hard to keep everything in sync. The only way to have consistent results is to always re-render parts of the app whenever your data changes. The virtual DOM eliminates the need to do this--you only mutate the real DOM when the virtual one knows there will be a real change to what is rendered.
I haven't noticed any, but I imagine the performance is increased
Handling of bugs
Easier wrapper components which can have anything passed inside of them as children

Benefits of the virtual DOM compared to other frameworks
jQuery code was spaghetti and Angular 1 was scope hell
jquery madness
Having to write efficient dom updating logic

Figure 10: Benefits of the virtual DOM compared to other frameworks

The survey asked if React enforces clear code. Many of the participants responded that it does not enforce clear code. The reasons why were the following: the Framework is not very opinionated, and can be used to write bad code. It is easy to not follow React's methodology.

The second last question asked participants whether the syntax in React revolves around the virtual DOM. Many participants mentioned the rendering part and the virtual DOM. One respondent does not believe that the syntax in React revolves around the virtual DOM.

Does the syntax in React revolve around the virtual DOM
Yes, to my understanding React is used mostly for UI and it does that through the virtual DOM.
It revolves around the lifecycle of elements of the virtual DOM
It partially does. There might be a very close mapping between the elements you render and the virtual DOM.
Yes
No, it's just like any XML syntax

Figure 11: syntax of React

The final question in the survey asks the participant's opinion on why React is more widely used than other frameworks. Most of the respondents wrote that Facebook played an important role in making React a popular framework. There was also a general agreement among the respondents that the framework is easy to learn.

In your opinion, what makes this framework more widely used than other frameworks
it is well documented and supported, logical syntax, easy to learn, server side rendering
It came out at a good time. People were getting sick of MVC-pattern frameworks like Backbone and Ember because it was difficult to make complex UIs with them. React was very different and revolutionary, and a lot of people went with it. Myself included. It's also important to acknowledge the role Facebook played into it. They threw money at React by paying their devs to work on it, by flying them to conferences where they did talks on it, and so on. Grassroots projects didn't really stand a chance.

In your opinion, what makes this framework more widely used than other frameworks
In a nutshell, Facebook and corporate adoption, create-react-app, good learning tools and resources
Its strong focus on small re-usable components. Other frameworks like Angular require you to learn other uninteresting things like Angular Modules and Angular Services.
It's simplicity
Backed by Facebook, keep innovating and focused on doing one great thing and let libraries do the rest
Better api
Strong company backing
The virtual DOM is brilliant and really easy to learn

Figure 12: opinions on why React is more widely used

5.2 Comparing outcomes

The next section will look at the results of the online survey, and compare them to the outcomes of the arguments made in the previous chapters. The research of the previous chapters and the results of the survey can be divided in to the following three sections: knowledge of frameworks, debugging and code, learning curve and corporate support.

5.2.1 knowledge of frameworks

When comparing the results from the survey to the results from the website *state of JS* from Sacha Greif, Raphael Benitte and Michael Rumbeau, it appears that the results from this survey done for this research project are similar to the findings of Greif, Benitte and Rumbeau. In their overview of knowledge of front-end frameworks, they asked developers which frameworks they heard of, and which they would prefer to use again. React, Vue and Angular are the top three frameworks in that chart. Among those three frameworks, React scored the highest percentage in terms of willing to use this framework again. Vue came second and Angular third. The other frameworks mentioned in the chart are Preact, Ember and Polymer. For the survey carried out for this research project, one developer has experience in Ember as well.

The comparison of these two diagrams suggest that although the survey carried out for this research project has a small sample size, it is representative of JavaScript developers in terms of their knowledge of frameworks.

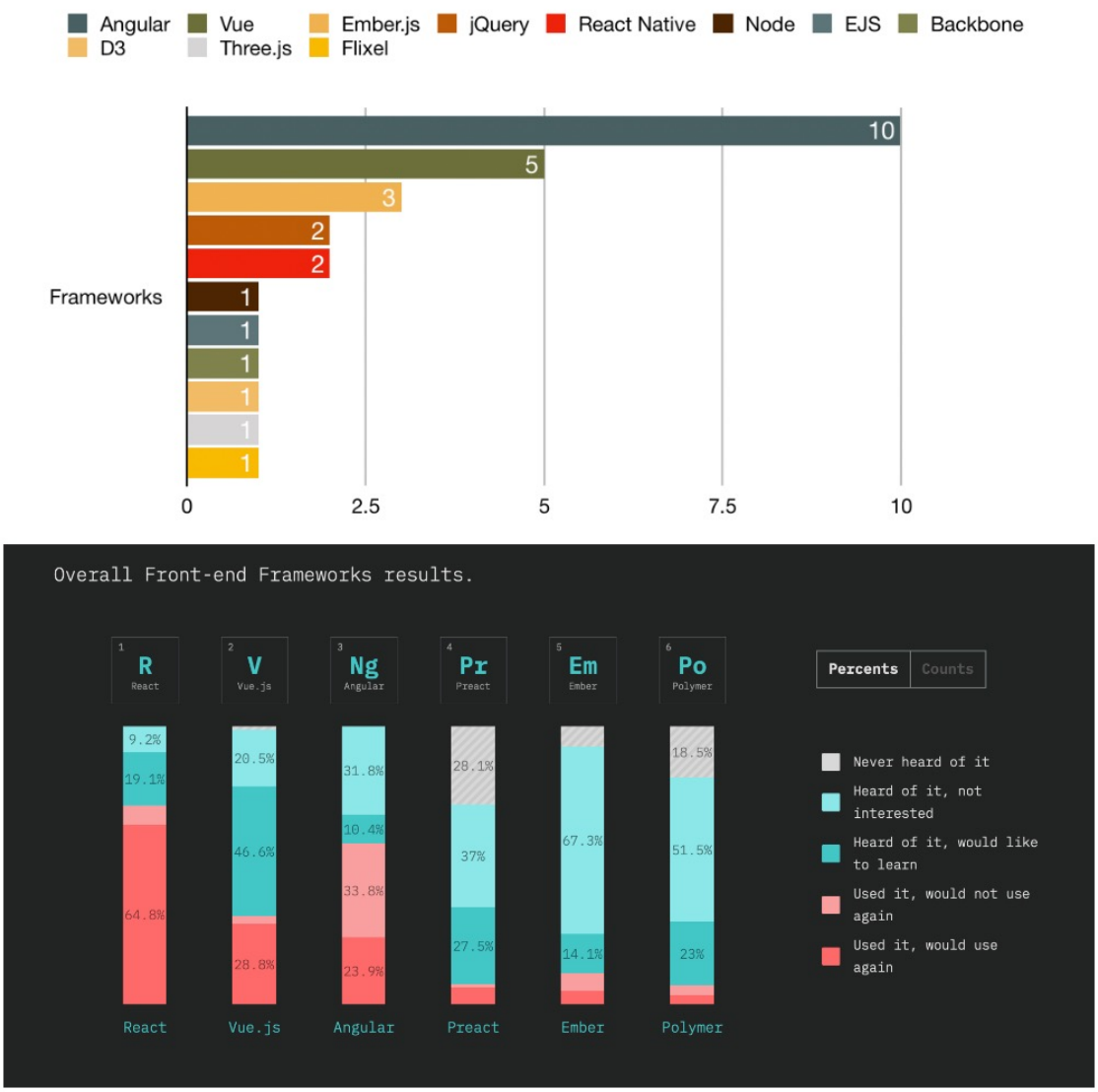

Figure 13: knowledge of frameworks, survey results compared to state of JS results (https://2018.stateofjs.com/front-end-frameworks/overview/)

5.2.2 Debugging and code

The results of the question of: "*What are the benefits of the virtual DOM compared to other frameworks?"* showed that, one important point made by the participants is that debugging is easier when using React. The following points were made by respondents.

1. Debugging is easier, there is a better error feedback.
2. Simplicity
3. The scope functions of Angular can be confusing, there are no scope functions in React.
4. jQuery code is unstructured and difficult to maintain compared to React.

The third point confirms the statement made about the scope functions in Angular mentioned earlier (Grov, 2015), which are not easy to debug for a beginning web developer. This research project mentioned in a previous chapter mentioned that jQuery was released to solve issues with differences in browser implementations. jQuery also seemed to be problematic for large scale projects and that is why Knockout was created by Jeremy Ashkenas. The fourth point

proves that frameworks are now better in terms of code and dealing with larger projects, participants confirmed that a framework, in this case React, have a high degree of efficiency.

5.2.3 Learning curve

When looking at the responses of the advantages and limitations of the virtual DOM question, many participants mentioned the learning curve. The following points were made.

1. A higher learning curve compared to something like EJS.
2. Lack of best practices to use it in can be an issue, especially when there is no prior knowledge.

Some of the participants stated that some limitations of working with the virtual DOM were things such as that it is long and tedious to write, and that it has a higher learning curve than EJS. However, when comparing a JavaScript framework to EJS or jQuery, it is difficult to point it out as a disadvantage of working with the virtual DOM because EJS and jQuery have a different approach.

Overall most participants stated that the virtual DOM is easy to work with. And that it is an easier technique compared to the techniques of other frameworks.

1. Good learning tools and resources
2. Other frameworks such as Angular require developers to learn other things like Angular Modules and Angular services

Naimul Islam Naim has mentioned the short and easy learning curve of React. Naim stated that compared to other JavaScript libraries, it is easier to start building an application with React. The framework appears to be easy to read, even for developers who are unfamiliar with it. Naim confirms the statement made by one of the participants who stated that other frameworks, such as Angular require developers to learn specific concepts only for that particular framework.

These outcomes can also be linked to the example of the Sokoban project. The Sokoban project mentioned earlier in this research project showed that when using the render function, everything gets re-rendered, even the parts that do not need updating, and that it is possible to do the diffing process without using a framework, but that a JavaScript framework does the diffing process automatically. Because frameworks handle a lot of tasks automatically, performance increases, this is also something participants of the survey confirmed.

5.2.4 Corporate support

Almost all opinions on why React is more widely used than other frameworks mentioned the role of Facebook. The following points were raised in the survey

1. Funding to pay the web developers, to make sure they attended conferences to talk about React.
2. Timing of when the framework was released

It is interesting to see that a high number of participants mentioned Facebook as a reason for React's popularity. Angular is a framework made by engineers at Google, but it is somehow not the most widely used framework according to the opinion of many web developers, both the survey done for this research project and the results on the state of JS website confirm this. It can therefore be stated that even though may responses pointed out Facebook as a reason, there must be other reasons, most likely related to the syntax of the framework, on why React is this popular.

5.3 New findings

Throughout the survey, participants mentioned two important points on React and the virtual DOM. The first point to be made is that the surveys show that many web developers were not entirely happy with Model-View-Controller frameworks such as Backbone and Ember because of the difficulty to make complex UIs. React appears to be different and revolutionary. The second point that was raised was that using React does not necessarily mean that using the virtual DOM is compulsory. It is possible to use this framework and avoid the virtual DOM by updating everything to the actual DOM directly.

5.4 Summary

This chapter presented the results of the online survey. The full responses are included in appendix B. Repeated themes have been identified from the responses and looked at each in turn. That is knowledge, debugging and learning curve, and these responses have been analysed. A comparison of the survey responses to the findings of previous researchers has been made. This comparison supports for the findings that the virtual DOM is an efficient technique.

6. Conclusion

JavaScript is the only language that is supported by the web natively, and allows developers to create dynamic web applications and create responsive designs. javaScript is a language that is constantly changing, but as applications get bigger, especially due the rise of social media applications, new solutions had to be created.

Social media applications such as Facebook have a lot of users on a daily basis. Those web applications are large and need constant updating, and updating the regular DOM can become expensive and slow. It is therefore much more beneficial to use a JavaScript framework such as React to write large-scale applications.

Previous research outlined in section 3.1 of this research paper has found that the virtual DOM improves performance, has a lower learning curve, is cheaper and is widely used and heard of by developers. Giuseppe Psaila, has found that the virtual DOM proves to be faster for creating large applications (Psaila, 2008). The virtual DOM appears to have a short learning curve as well, it is easy for beginner developers to get used to the virtual DOM. The virtual DOM also improves performance when building large applications compared to other DOM update mechanisms such as dirty checking adapted by Angular. React is also the most widely used and heard of framework, according to survey results published on the website *State of JS* (Greif, Benitte, Rambeau, 2018). The virtual DOM is a memory, and can be manipulated many times before making changes to the real DOM, this is a reason why working with the virtual DOM is less expensive.

Findings in section 3.4 show that using the virtual DOM has more benefits than using KVO or dirty checking. Dirty checking results in data binding limits and KVO is not an efficient technique because it requires more work from developers. Debugging is also more complicated and is similar to *object.observe* which was used in Vanilla JavaScript but has now become redundant.

The Sokoban project shows that it is natural to work with the virtual DOM, especially for beginners. But that even though it is a natural way of coding, the render function used in this project does not check whether a value has changed or not, everything gets re-rendered, which makes this process rather slow. This project also shows that working with a JavaScript framework is therefore more beneficial because less needs to be taken care of by the developer. The framework takes care of the diffing process, which means less work for the developer. But when looking at the render function, it is also not beneficial to re-render everything, including values that do not need to be updated. When working with the virtual DOM only the values that need updating will be changed.
Finally, the results of the online survey agree with the statements made in this research paper. Section 5.2.3 shows that developers agree that the virtual DOM has a short learning curve. The participants of the survey also stated that debugging is easier when using React and the virtual DOM. Section 5.2.4 shows that there is an overall opinion that Facebook played an important role in making React and therefore also the virtual DOM a popular framework.

6.2 findings

There is a difference between what makes it easier for the developer to write and what makes it easier for the developer because the framework takes care of it automatically. Performance is something that is done automatically by the framework, which is beneficial for the developer but is not something that is easier for the developer to write. It is almost certain that the virtual DOM makes it easier for the developer to write large projects, but what is beneficial for using the framework is that it handles a lot of things automatically, and takes over certain tasks from the developer.

6.3 Possible future work

The topic of this research project is constantly changing. Frameworks tend to have a short life cycle, and new frameworks, are constantly being created. The JavaScript community also seems to add new features and update the existing frameworks on a regular basis. Possible future work could therefore be discussed further.

This research project has shown that many JavaScript developers consider corporate support of a framework to be an important reason for its popularity. As indicated in section 5.2.4 this does not appear to be the full story behind the popularity of React and the Virtual DOM. An interesting area of further research would be to investigate to what extent developers choose frameworks based on the syntax or features of the syntax versus the question of corporate support.

As outlined in the conclusion, there are two main points on why a framework is more beneficial for the web developer. There is a difference between what is easier for the developer to write and what is easier for the developer because the framework takes care of it automatically. It is therefore interesting to do further research on whether JavaScript frameworks will become or need to be more automated in the future.

The future of JavaScript and Javascript frameworks is always changing. It is therefore possible that there are already other alternatives or new techniques available. An interesting area of further research would be to investigate whether the virtual DOM will remain popular in the future or whether new DOM updating mechanisms are being created.

7. References

1. Aderinokun, I. (2018). *Understanding the Virtual DOM*. [online] bitsofcode. Available at: https://bitsofco.de/understanding-the-virtual-dom/ [Accessed 23 Apr. 2019].

2. Angular.io. (2010). *Angular*. [online] Available at: https://angular.io/docs [Accessed 8 May 2019].

3. Barker, A. (2018). *The Super-Brief History of JavaScript Frameworks For Those Somewhat Interested - DEV Community*. [online] Dev.to. Available at: https://dev.to/_adam_barker/the-super-brief-history-of-javascript-frameworks-for-those-somewhat-interested-3m82 [Accessed 29 Apr. 2019].

4. Copes, F. (2018). *The Vue Handbook: a thorough introduction to Vue.js*. [online] freeCodeCamp.org. Available at: https://medium.freecodecamp.org/the-vue-handbook-a-thorough-introduction-to-vue-js-1e86835d8446 [Accessed 1 May 2019].

5. Costa, C. (2016). *Reactive Programming with Swift*. 1st ed. Birmingham: Packt Publishing, pp.6,7.

6. Cravens, J. and Brady, T. (2014). *Building web apps with Ember.js*. 1st ed. Sebastopol, CA: O'Reilly Media, pp.1,15,16,17.

7. Eng, R. (2017). *Chapter 3: What is Object-Oriented Programming?*. [online] Medium. Available at: https://medium.com/learn-how-to-program/chapter-3-what-is-object-oriented-programming-d0a6ec0a7615 [Accessed 4 May 2019].

8. Gimeno, A. (2018). *The deepest reason why modern JavaScript frameworks exist*. [online] Medium. Available at: https://medium.com/dailyjs/the-deepest-reason-why-modern-javascript-frameworks-exist-933b86ebc445 [Accessed 1 May 2019].

9. Greif, S., Benitte, R. and Rambeau, M. (2018). *The State of JavaScript 2018*. [online] Stateofjs.com. Available at: https://stateofjs.com [Accessed 21 Apr. 2019].

10. Grov, M. (2015). Building User Interfaces Using Virtual DOM A comparison against dirty checking and KVO. Master. University of Oslo.

11. Honeypot (2019). *Ember.js: The Documentary*. [video] Available at: https://. www.youtube.com/watch?v=Cvz-9ccflKQ [Accessed 25 Apr. 2019].

12. Developer.apple.com. (2019). *Introduction to Key-Value Observing Programming Guide*. [online] Available at: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/KeyValueObserving/KeyValueObserving.html [Accessed 28 Apr. 2019].

13. Kussain, A. (2018). *Create your own virtual DOM to understand it (Part 1)*. [online] Medium. Available at: https://medium.com/@aibolkussain/create-your-own-virtual-dom-to-understand-it-part-1-47b9b6fc6dfb [Accessed 1 May 2019].

14. MDN Web Docs. (2019). *Object.observe()*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/observe [Accessed 8 May 2019].

15. Medium. (2017). *How important is JavaScript for Modern Web Developers?*. [online] Available at: https://medium.com/@mindfiresolutions.usa/how-important-is-javascript-for-modern-web-developers-2854309b9f52 [Accessed 2 May 2019].

16. Mihailescu, M. and Sorensen, E. (2010). Model-View-ViewModel (MVVM) Design Pattern using Windows Presentation Foundation (WPF) Technology. *Megabyte journal*.

17. Miller, D. (2018). *The power of JavaScript*. 1st ed. New York: Cavendish Square Publishing, pp.5,6,13,14,15.

18. Naim, N. (2017). ReactJS: An Open Source JavaScript Library for Front-end Development. Bachelor. Metropolia University of Applied Sciences.

19. Psaila, G. (2008). Virtual DOM: an Efficient Virtual Memory Representation for Large XML Documents. In: *DEXA*. Turin: IEEE Computer Society, pp.1,237.

20. Rajput, M. (2016). *The pros and cons of choosing AngularJS - JAXenter*. [online] JAXenter. Available at: https://jaxenter.com/the-pros-and-cons-of-choosing-angularjs-124850.html [Accessed 6 May 2019].

21. Reactjs.org. (2019). *Reconciliation – React*. [online] Available at: https://reactjs.org/docs/reconciliation.html [Accessed 27 Apr. 2019].

22. Sandofsky, B. (2016). *The Flaws of KVO*. [online] Medium. Available at: https://medium.com/@sandofsky/the-flaws-of-kvo-4a3034ade8e8 [Accessed 4 May 2019].

23. Seshadri, S. and Green, B. (2014). *AngularJS: Up And Running*. 1st ed. Sebastopol: O'Reilly Media, Inc., p.206.

24. Smith, D., Hunt, P., O'Shannessy, P. and Coatta, T. (2016). React: Facebook's Functional Turn on Writing Javascript. *Communications of the ACM*, 59(12), pp.56 - 62.

25. Sullivan, N. (2009). *Reflows & Repaints: CSS Performance making your JavaScript slow? | Stubbornella*. [online] Stubbornella.org. Available at: http://www.stubbornella.org/content/2009/03/27/reflows-repaints-css-performance-making-your-javascript-slow/ [Accessed 5 May 2019].

26. Turnbull, D. (2014). *Improving Angular Dirty Checking Performance - DZone Performance*. [online] dzone.com. Available at: https://dzone.com/articles/improving-angular-dirty [Accessed 8 May 2019].

27. Wilton-Jones, M. (2006). *Dev.Opera — Efficient JavaScript*. [online] Dev.opera.com. Available at: https://dev.opera.com/articles/efficient-javascript/?page=3#reflow [Accessed 5 May 2019].

8. Appendices

8.1 appendix A:survey questions

Background

1. How many years of experience do you have in front-end development?
2. Would you be able to give me a list of Javascript frameworks you have experience in?
3. How long have you been working with the virtual DOM?
4. How long did it take you to get used to working with the virtual DOM?

Pre-virtual DOM and Post virtual DOM

1. Does the virtual DOM solve problems that occurred in other Javascript frameworks?
2. Does working with the virtual DOM create other problems that did not exist pre-virtual DOM?

Difference of JS frameworks

1. Does the syntax in React revolve around the virtual DOM?
2. What are the advantages, limitations of (React), the virtual DOM?
3. Does this framework enforce clear code?
4. In your opinion, what makes this framework (React) more widely used?

8.2 Appendix B: full survey responses

What is your gender
Male
Male
Male
Male
Male
Male
Male
Male
Male
Male
Male
Male
Female

What is your age?
23
23
25
23
27
46
21
25
22
30
25
21
24
22

How many years of experience do you have in front-end development?
5
7
3
2
8
2 years
1
4
5
8
2
2
5
2

Would you be able to give me a list of Javascript frameworks you have experience in?
React, Node , Vue, EJS
Angular, Ember, Vue and I guess React if you'd call it a framework
react, angular
React, Angular, Vue
Backbone and React professionally. Ember and Angular experimentally.
React, Angular
Angular 5, Vue, React-Native, Redux
Reactjs
AngularJS, ReactJS, React Native
React, Angular, jQuery
React jquery angular
react, angular, jquery
Jquery d3 three.js react
Angular, Ember, React

How long did it take you to get used to working with the virtual DOM?
A few weeks
Not long
6 months
A few months, we never really covered anything like virtual DOM in college, so it was all learned on the job
The main challenge was getting into the swing of doing thing the React way and moving away from the old MVC n
I was educated to use it right from the start
I'm not sure
5 months
1
Very fast
1 year
1 month
Pretty quick
1 year

How long have you been working with the virtual DOM?
About 6 months
~2.5 years
2
2 years
4
2 years
1 year
1 year
1
3 years
1 year
2 years
1 year
1 year

Please identify any problems that you experienced using other JavaScript frameworks that are solved or lessened by using the Virtual DOM
Debugging is easier, gives better error feedback
Updating the DOM when data changed and ensuring it's handled everywhere. Lots of boilerplate code that wasn't easily abstracted
performance and efficiency of work
When you have a large app where multiple independent parts of it rely on the same data, it's hard to keep everything in sync. The only way to have consistent results is to always re-render parts of the app whenever your data changes. The virtual DOM eliminates the need to do this--you only mutate the real DOM when the virtual one knows there will be a real change to what is rendered.
n/a
I haven't noticed any, but I imagine the performance is increased
Handling of bugs
Easier wrapper components which can have anything passed inside of them as children
jQuery code was spagetti and Angular 1 was scope hell
jquery madness
Having to write efficient dom updating logic

Does working with the virtual DOM create other problems that did not exist pre-virtual DOM? If yes, please elaborate
Initial setup time is a bit longer
While it's often more convenient and has faster dev cycles, it's slower than well made manual DOM updates which occasionally can be an issue
not really
The only real problem is the performance overhead you get from having to maintain a virtual DOM. It costs memory to store it and CPU cycles to query it. That's not a problem you ever feel though.
Extreme drilling down of state and other state management issues
Not in my experience
None that i can think
no
Gotta learn another way of doing things
No

Does the syntax in React revolve around the virtual DOM?
Yes, to my understanding React is used mostly for UI and it does that through the virtual DOM.
I'd say it revolves around the lifecycle of elements of the V DOM
not sure about this q
I haven't thought of it in this way before, but I suppose it partially does. There might be a very close mapping between the elements you render and the virtual DOM.
Yes. Component render methods are the key aspect of how it renders the DOM
No
No
No its just like any XML syntax
yes
No

What are the advantages, limitations of (React), the virtual DOM?
Better data structures and component support but a higher learner curve compared to something like EJS
React being un-opinionated, as it's just a view library, allows it to slot into a lot of different technical stacks and be quite flexible. At the same time, the lack of best practices to use it in can
frequent updates resulting in new methodologies and constant learning (which is also a good thing, but its quite fast paced)
React's top-down data flow helps keep big apps consistent. Whenever data changes, it's fed to everywhere that wants to know about it. When you want to mutate data, it goes all the way back up to the source of the data. This helps you have a "single source of truth" for your data, keeping disconnected parts of your UI from falling our of sync with one another. I haven't felt any limitations with React. The big downside is needing a compiler for JSX which complicates the getting-started process. I'm not too sure what you meant with including the virtual DOM in this question. You can have a
Sometimes needs more overhead than you would like for simple things, but allows you to have dynamic content instantly re-rendered on state changes.
Not sure
Easy to right nicely decouple components and no need to track what changed and mutate what you need. Was long and tidious to write
faster, declarative, one way state to render calculation
Quick render times without thinking about it

Does this framework enforce clear code?
Relatively no I think, I prefer strongly typed solutions
React can be confusing nowadays. Context, Hooks and State aren't the most obvious when you read the code, but the lifecycle events generally do
no
React enforces patterns of how it expects things to be laid out, but in the end the developer can still make a mess of things by not following React's methodology or by getting sloppy and making massive unmaintainable components. I don't think you can get much better in terms of "enforcing" though.
I'm not sure what this refers to, but classes can be a bit obscure.
No
No, it can be used to create bad code, it's not very opinionated
A little bit by the way data flow but its mainly on the coder.
no
No
Yes

In your opinion, what makes this framework (React) more widely used than other frameworks?
Well tested, maintained, feature rich and well documented.
The fact that it's a view library, not an entire framework like Angular, etc. As a result, you can hook in things you want/need without it fighting you while it still provides the barebones you need to manage
it is well documented and supported, logical syntax, easy to learn, server side rendering
It came out at a good time. People were getting sick of MVC-pattern frameworks like Backbone and Ember because it was difficult to make complex UIs with them. React was very different and revolutionary, and a lot of people went with it. Myself included. It's also important to acknowledge the role Facebook played into it. They threw money at React by paying their devs to work on it, by flying them to conferences where they did talks on it, and so on. Grassroots projects didn't really stand a chance.
In a nutshell, Facebook and corporate adoption, create-react-app, good learning tools and resources
Its strong focus on small re-usable components. Other frameworks like Angular require you to learn other uninteresting things like Angular Modules and Angular Services.
It's simplicity
Backed by Facebook, keep inovating and focused on doing one great thing and let libraries do the rest
better api
Strong company backing
The virtual dom is brilliant and really easy to learn