OULU UNIVERSITY OF
APPLIED SCIENCES

The Power of Reusable Components and The Future of The Web

The Power of Reusable Components and The Future of The Web

Yassine Laadraoui
Bachelor’s Thesis
autumn 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Information Technology

Author: Yassine Laadraoui
Title of the bachelor’s thesis: The Power of Reusable Components and The Future of The Web.
Supervisor: Kari Jyrkkä
Term and year of completion: Autumn 2020 Number of pages: 68

The subject of this thesis is the study of the web component technology and a breakdown of Angular, View.js and React. The three framework that dominated the sphere of web components and web development in general during the last few years.

the breakdown of each of the frameworks focuses on the structure of each framework’s code, and how they differ when initializing a project. Followed by a study of their most prominent features then their advantages and challenges.

Finally, the thesis provides a comparison of the frameworks by comparing the most relevant information of the framework such as speed and size and other criteria.

Keywords: Web components, Javascript Framework, front-end development

PREFACE

This thesis has no client, and it is the author's, Yassine Laadraoui's, personal take on the future of web development as a mandatory part of his studies. The objectives of the thesis were to research the web component technology, compare current top frameworks using them to provide a deeper understanding of which framework implements this technology best and what framework would be best to learn and work with. The author applied some background knowledge learned in school about the development. The research happened on the Internet with Google and online library searches. A wide variety of books, articles and other sources were used as materials. The main results are a description of how the web components work, a comparison of three major frameworks. And the comparison focused on features that determine the success of a framework or its failure.

Oulu, 5.9.2020
Yassine Laadraoui

CONTENTS

TERMS AND ABBREVIATIONS

AJAX: Asynchronous JavaScript And XML

ASP : Application Service Provider

CERN: The European Organization for Nuclear Research.

CGI: Computer Generated Imagery.

CSS: Cascading style sheets.

DNS: Domain name service.

DOM: Document Object Model

HTML: Hypertext markup language.

HTTP: Hypertext Transfer Protocol.

MVC: Model View Controller.

JS: JavaScript.

JSON: JavaScript Object Notation.

JSP: Java Server Page

JSX: JavaScript Extension

PERL: Practical Extraction and Report Language

PHP: Hypertext Preprocessor.

REST: Representational state transfer

SEO: Search Engine Optimization

SSR: Server-Side Rendering

URL: Uniform Resource Locator.

UI: User Interface

TCP/IP: transmission control protocol and internet protocol.

XML: Extensible Markup Language.

1 AIM OF THIS THESIS

The aim of this thesis is to give a detailed description of the web development technology and its three popular frameworks -React, Angular and Vue.JS- and then move to compare the three frameworks.

The first chapter of the thesis will focus on the history of the web and different stages of its development starting with the first live website from CERN and going through web 2.0 to web 3.0 Stating both their advantages and disadvantages.

Then the thesis moves to the current state of the web and what lead to the creation of web components. Next, there is an overview of the web components and the three technologies that makes it possible and its benefits and challenges. Then, the JavaScript frameworks, which are based on web components, are handled.

In chapters 4, 5 and 6 the focus will be on Vue.JS, Angular and React with a detailed description of each one, focusing on their respective strengths.

The seventh chapter focuses on a deep analysis of the different frameworks, depending on criteria chosen in the beginning of said chapter. These comparison points are the biggest points that sway users one way or another.

the main aim of this thesis is to give the reader an understanding of web components and the 3 different frameworks that lead the use of this technology. With this understanding the reader can have enough information to decide which framework to use in a project.

2 INTRODUCTION TO WEB HISTORY

2.1 History of the web

Prior to 1989, no information sharing system has existed commercially nor academically. In the year, a British scientist named Tim Berners-Lee created a system of information system for departments in CERN named the World Wide Web.

2.1.1 Tim Berners-Lee’s Hypertext proposal:

in the year 1989 as Berners-Lee was working for CERN, he noticed a pattern in the discussions between his colleagues, whenever they were talking about a big project, giving that at that time it was the era of huge discoveries thanks to LHC. Their talks ultimately settled when faced with one problem in particular which is the huge amount of information involved in such a project and how they would keep track and manage this information across multiple departments. The structure at CERN, to put in Tim’s words, is a multiply connected “web” [6], which means that as time goes by these webs will become more and more complicated. Lacking a reliable information sharing system, information about what projects are being worked on or what facilities or equipment exists is usually shared through the word of mouth.
Other types of information such as technical knowledge that has to do with a certain project, is usually recorded but giving the huge amount of data gathered and recorded every day, the process of finding a certain piece of information became this time a demanding task that lead to a decrease in productivity and sometimes information is lost all together in the piles of data. So often the only place to find the information is in the minds of people who actually worked on the project. During that period of time, the typical length of stay for a CERN employee was about two years. So some technical details are lost forever and when a new employee arrives, they are giving hints on who are the right people to talk to get them up to speed, which takes time and resources that could be better used otherwise.

"CERN is a model in miniature of the rest of world in a few years' time. CERN meets now some problems which the rest of the world will have to face soon. In 10 years, there may be many commercial solutions to the problems above, while today we need something to allow us to continue." [6]

2.1.2 Requirements for such system

For this system to be practical in the CERN environment, it needs to satisfy a few requirements listed below:

- First, the system needs to allow **remote access across networks** as CERN is a distributed across many computer and access to remote machines is essential.
- Second, the system needs to be **Heterogenetic,** meaning that the data could be accessed from different types of systems.
- Third, the system needs to be **Non-Centralized**, meaning that it should allow new systems to be merged without any type of center for control.
- Fourth, the system needs to allow **Access to existing data**. To allow the system to get off the ground quicker, it needs to allow access to existing data in CERN in the form of hypertext.
- Fifth, the system should allow one to add **Private links** from and to public information.
- Sixth, the system should be able to **store and display ASCII text** with the possibility of adding graphics.
- Seventh, given a large hypertext database, new uses can emerge, such as **Data Analysis**. This gives the possibility of new features e.g. search.
- Finally, the system should have **Live links**, so that the data can be retrieved whenever the linked is pressed. Because data of a document is always changing, this is necessary. And this allows a new possibility of having an application fire up if a link is pressed.[6]

A non-requirement that could be added is security because a common problem with hypertext is the copyright enforcement and data security. Sophisticated systems for authorization can be designed to protect sensitive data in the future.

2.1.3 Solution: The World Wide Web

The World Wide Web is a collection of documents and resources from across the world linked through the use of hyperlinks. These documents, which can include text, images, and other multimedia, are delivered using internet protocols.

The World Wide Web came in as a combination of four basic ideas, hypertext, resources identifiers, a client-server model of computing and a markup language.

2.1.3.1 How the Web Works

Previously it was mentioned a client-server model of computing as being one of the four basic ideas that made the creation of the web possible. The client part is responsible for the retrieval of information resources, such as web pages or computer readable files, by sending a request to a server located somewhere in the world by the use of its URL. Usually this client is a web browser which makes it possible to display these resources on the user's machine (e.g. computer, tablet, phone). The client is not a passive agent in this dynamic as it can also send data to the server by the use of web forms and the server can either save this data or process it in some way. Web pages are of the information resources submitted by a server and they are usually grouped together in the form of collections of related material referred to as Websites. These websites can contain multiple hyperlinks where a user can click on one and jump to another website which is referred to as surfing or browsing the web.

When you type a URL into the browser, the modern browsers usually keep a local DNS cache to see if the domain has already been resolved recently. If so, it will get the IP address from there. And if not, it will use a DNS resolver, such as getHostByName POSIX system call to retrieve information either from a local host or from a public DNS server, such as Google's 8.8.8.8 server. The browser then receives an IP address that corresponds to the requested server, to which the browser sends an HTTP request for the server to send a copy of the website to the client. This request and all other data is sent using the TCP/IP protocol. If the server approves the request, it will send an "200 OK" message, and then it starts sending the website's files to the browsers as data packets.

The HTTP Server sends the "index.htm" web page. This is the default page when no other filename is specified in the URL. If you enter a URL, including a filename (for example http://yassine.me/prject.htm), the HTTP Server sends this page. The default page index.htm is a static page. This means that the content of the page stored into the server is sent unmodified to the web client on request. Usually, this page contains links to other static or dynamic pages on the server. When a browser requests the default page, the HTTP Server tries to open index.htm as a default web page. If this page does not exist, the web server tries to open index.cgi instead. If this page is also not present, the web server responds with an Error 404 - Not Found.[7]

2.1.4 Different versions of the web

2.1.4.1 Web 1.0

the web invented by Tim Berners-Lee is called web 1.0 and it is only limited to read actions, where a small amount of people creates web pages and give access to a large number of clients via the Internet. In the early days the interactions of the clients with the web page were limited to only reading the information provided and they could not submit their own information, such as comments or add articles.

Web technologies are also used such as XML, XHTML and CSS. In web1.0 both server-side and client-side scripting are used such as ASP, PHP, JSP, CGI, PERL as server-side scripting and JavaScript, VBScript, flash as client side.[12]

Short comings of web 1.0

The early version of the web was slow and chunky in nature and every time a new piece of information was entered to the web pages, a reload was necessary. Web 1.0's biggest problem was that it did not support two-way communications. The client-pull model made communication capabilities limited as it could only be initiated by the client. This model made the networking possibilities limited as it ignores the human sense for building networks and the limited number of writers made the users of the early web starved for resources. The assumption that the web was a publishing tool instead of the public space limited its capabilities.

2.1.4.2 Web 2.0

Web 2.0 is the version of the web that allowed a write and read operation. It basically provided a new operation by using existing web technologies to allow a user to add, update and upload content into the web. It evolved the network into a platform by the creation of places for sharing knowledge such as wiki or web blogs.
The web 2.0 was based on few ideas that allowed the jump from a read only web to this form:

❖ User generated content: this means that every single user can be a publisher on the web and post their own content through wikis or blogs and other public spaces.

- Architecture of participation: the design of an architecture for online technologies that allows participation and collaborative.
- Openness: open access and providing open source software and re-use of free data.

The technologies used to make the web2.0 were present way before the creation of it. They were all technologies that were behind the Internet, such as XHTML, CSS, AJAX, Flash.

Infrastructure of web 2.0

- Content Syndication: is a technology that allows content from one website to many websites. The two most known examples of content syndication is RSS and Atom.
- Ajax-based Internet Technology: Ajax means Asynchronous JavaScript and XML and it is used to allow the client side of the connection to retrieve data from the server side asynchronously meaning that the data will be sent from the server to the client as it appears in the server. An example of this would be the modern-day chat or texting features available on many social media sites.
- Document Object Model is a tree representation of an HTML document, with many DOM nodes in every document.
- REST: Representational State Transfer (REST) is an architecture style that represents a set of constraints used to create web services. It allows the requesting systems to access and manipulate textual representation of web resources.
- XML: is a markup language that defines the encoding of documents in both human and machine-readable format.
- CSS: Cascading Style Sheet, is a style sheet language used to describe the representation of a document defined in a markup language.

Flaws of web 2.0

With the openness and accessibility that web 2.0 offers, an interesting set of applications were developed. Although they were as optimal as the community has ever seen, they were not without flaws. Because of the new found ability to upload your own content to a website, a malicious user could, for example, upload a content designed to run code or carry a malware to perform an authorized task or insert malicious code in a piece of free software to collect information, such as Trojans and other types of hacking tools. Web 2.0 has common vulnerabilities that are well known by any computer programmer. Some of them are:

- ❖ Cross Site Scripting (XSS): Is a widespread security flaw that exist in any application where a user is allowed to input data. It is a type of injection attack where a malicious user uploads a script into a trusted website where other users can view that data uploaded and use the same-origin policy to bypass security checks. Since the victims browsers assume that all data from a website, which has permission, is trusted it does not verify it and executes it and the script ends up gaining access to information, such as cookies or session tokens or other sensitive information. Up until 2007, XSS accounted for 84% of all security vulnerabilities documented by the firm Symantec and in 2017 it was still considered a major threat vector.
- ❖ Cross Site Request Forgery (CSRF): if the XSS attack uses the website's trust to attack a user, a CSRF attack uses a website's trust in a user to perform unauthorized actions. There are many ways this type of attack can be performed either by tricking a user into submitting an unwanted request to a website such as changing a password or sending information. Such requests can be hidden in image tags or hidden forms. Another way it could be done is by taking their cookies and injecting it into a browser to be identified as the victim and gain login into their account.
- ❖ SQL Injection: an injection attack that relies on input fields in a web application to run malicious SQL code. A malicious user would type in an SQL command designed to run a certain task in the database side. A well-done SQL injection attack can allow the malicious user to spoof identity,

tamper with existing data, void transactions, change balance of an account, disclose all data in the database, destroy data or gain administrator access to the database. These types of attacks are common in PHP and ASP applications due to their interface structure and they can be avoided by verifying a data before submitting it into the query in the database.

- XSS Worm: is an attack where a malicious user uploads XSS code into a website. The worm spreads to the users when they visit that website and it can be harmful when giving the coder access to personal data or just executing a certain task on the website from the victim's profile.
- Authentication flaws: with the two way system that the web2.0 relies on users are usually asked to create an account in web applications and this can come with its own set of flaws, such as weak passwords that can be guessed or figured out through brute force attacks. Also, the use of weak password recovery systems can allow the theft of someone's account. With authentication comes the concept of session management and this session ID. Sometimes these session IDs can be insufficient in length or have no expiry time so if they are stolen, they can lead to huge problems. Likely these are flaws related to the design of web application and can be fixed with the use of multi-factor authentication. Session timeouts and a server-side session ID generation system with long IDs do not include a session ID in the URL, password strength checks, minimum password length.
- Information Leakage: is a security flaw were a website or application reveals data that is meant to be secret. Sometimes this data can be sensitive. The most common types of data leakage in the web is through the use of HTML comments where sensitive or developer specific data is written or through improper server configuration.

Safeguards in web 2.0

As with any system developed by humans, the creation of web 2.0 lead to the creation of systems and architectures to safeguards flaws that web present in this new version of the web.

- SSL or Secure Socket Layer Certificates: Is a small file that contains a cryptographic key that is bound to an organization. By installing this certificate in the organization’s web server, it activates the HTTPS protocol and allows a secure connection to the web server from a browser. When the certificate is installed, all web traffic between the web server and the web browser will be secure and a padlock or a green bar will show up in the browser to mark that this website is secure, it is usually present when entering credit card details or browsing social media. this technology works by the use of two keys one of them is public and presents on the public domain which can be used to encrypt data and that data can only be decrypted by the use of the user’s private key. SSL certificate keeps data secure, increases the website’s ranking on search engines, enhances trust of users and improves your conversion rates. Such certificates can be purchased from organizations GlobalSign.
- Risk Reporting Tools: Are a set of tools used to monitor suspicious activities and report them. It is present on many websites such as Facebook or Instagram, and when you receive a notification that some suspicious activity is happening in your account that is the result of the work done by these tools. A common web application based on this idea is Google Alerts.
- Web authorization: Enterprises should add new security policy and authorization to refine authentication and access.

2.1.4.3 Web 3.0

Web 3.0 is the third generation of internet-based services. It includes websites and applications that focus on the use of machine-based analysis of data to provide a data-driven and semantic web to create intelligent, connected and open websites.
As of today, there is no currently defined understanding of Web 3.0. Major internet and computer pioneers have been at odds on a specific and mutual understanding on what web 3.0 is.
Quotes from web pioneers are listed below:

- Google CEO, Eric Schmidt says that web3.0 will be "applications which are pieced together – relatively small, the data are in the cloud and it can be run on any device(pc or mobile),very fast, very customizable and distributed virally(social network, email)"[35]

- Yahoo founder, Jerry Yang stated "...you don't have to be a computer scientist to create a program. We are seeing that manifest in Web2.0 and 3.0 will be a great extension of that, a true communal medium...the distinction between professional, semi-professional and consumers will get blurred, creating a network effect of business and applications"[35]
- As Gian Gonzaga, Ph.D., senior director of research and development at the dating site eHarmony, says, "The Web can give something back that was not previously known. Web 3.0 learns and understands who you are and gives you something back." [35]
- Berners-Lee originally expressed his vision of the Semantic Web as follows: "I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A "Semantic Web", which makes this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize." [6]

From these quotes, some ideas on which technologies that make web 3.0 a possibility can be deduced. Some of them are currently available, others are being developed or not widely used. Here are some of these technologies that will make up the next wave of the web:

- Semantic Web: this concept basically means that the web will be able to understand the different meanings of data available online. It will be able to tell the meaning and the difference between two sentences.
- Artificial Intelligence: is the use of machine intelligence to understand data and filter websites and present users with data set to their preferences. In the current version of the web, the filtration of preferences is done by peer review, but this method is not perfect as it is affected by user biases. AI can solve these problems since it does not have any biases and with the increasing computational power, it can predict what the users want to see.
- 3D Web: in the last few years, a lot of virtual reality product and technologies have become popular. Virtual reality headsets, such as Facebook's Oculus Rift or Microsoft's HoloLens, are used to enhance your virtual or augmented reality experience in the gaming industry but with development and this technology becoming cheaper it gave a new horizon for tech companies to operate in. The creation of a website that can be navigated by the use of these headsets, projects for social-media websites that use this technologies have been in development. If you follow tech news you might have seen Facebook's take on this trend by their project called Facebook Horizon, where a user can interact with other users by using an avatar and can shop and play games using it, offering a more connected experience.

This thesis will focus on Eric Schmidt's definition of web 3.0 that means that the web 3.0 will be consisting of apps that are made up of a lot of components linked together.

3 REUSABLE WEB COMPONENTS

3.1 Introduction

An average internet user likes to think of a website as a homogenous structure that defines the website from top to bottom and that definition up to recent times have been somewhat correct, as all elements of a website are defined in the same file. But with the advancement of web technology and software engineering methods, developers and scientists have realized how flawed this approach is because it forces the developer to rewrite the same code in a different part of the website. For example, if a website has two buttons or image files, a developer has to write the code to define the button or the picture twice. This approach wastes time that could have been used to write different parts of the website and it also makes the source code for the website bigger rendering it harder to understand and reducing the clarity and the beauty of the code itself.

This is where reusable web components come into play. Considering the example of building a set of Lego toys, you start by placing one block on top of another. These blocks can have different shapes and different colors so you can have the same block with different properties and this is basically the underlying idea behind web components. A standalone block of code that defines a website element, be it an image holder or a paragraph or any possible element you can find in a website. These blocks can be as large as you want them to be and have different functionality, but to keep your code clean and readable, it is recommended to keep your components as small as possible and they should have a single function. And like Lego blocks, you can simply place it in your code, and you will have the html element it defines on your website.

In the figure 1 below, you can see the Google home page. Each red square is a component or a Lego brick, the only difference is that web components can have a parent child relationship as one component can contain multiple components

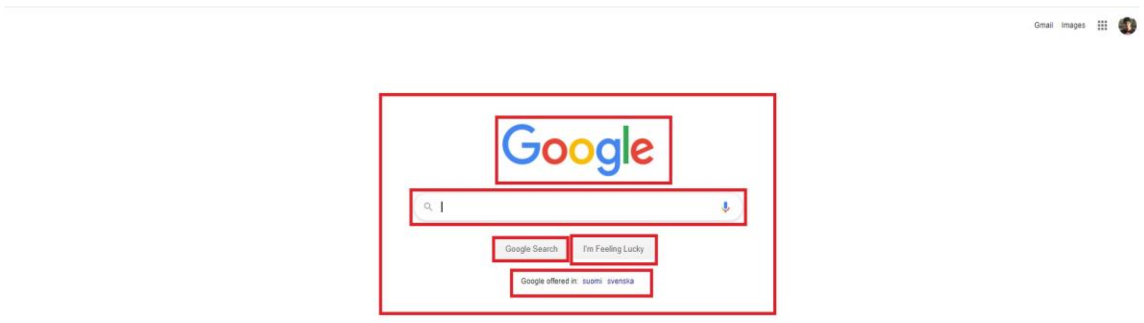
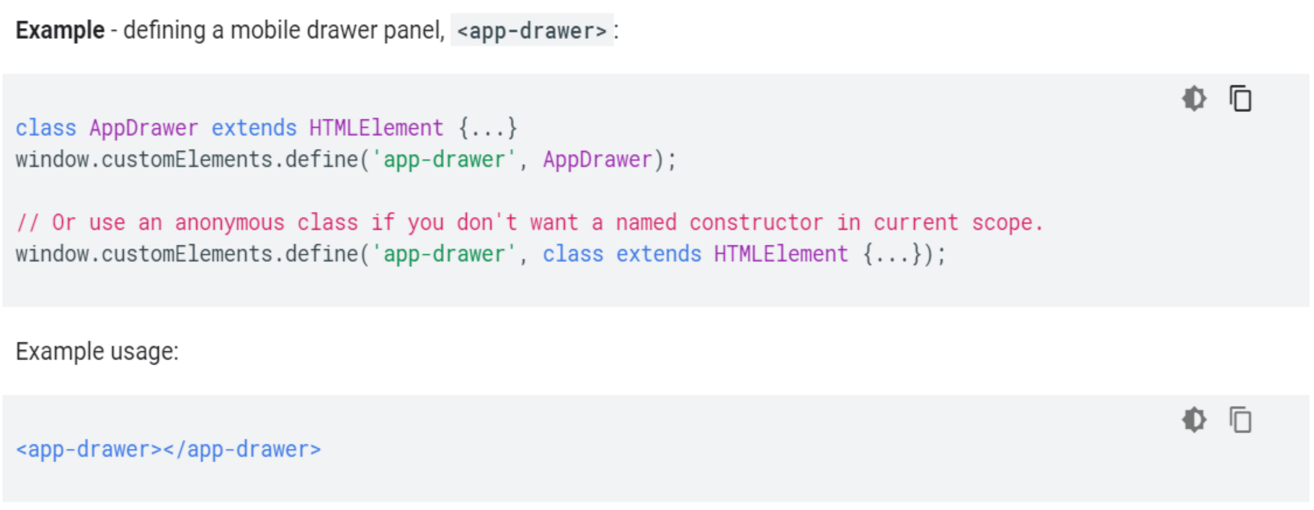


Figure 1 Google home page [38]

In order to provide this versatility and reusability, web components rely on three main technologies:

3.1.1 Custom Elements

The custom Element API is the foundation of web components, it allows you to create your own fully featured HTML tags or to extend components written by other developers, By defining a custom element, programmers can inform the browser on how to properly construct components and how elements of those components should react to changes. As of this time, there are still many limitations to custom elements that prevent them from explaining the behaviors of existing HTML elements.


Example - defining a mobile drawer panel, `<app-drawer>`:

```
class AppDrawer extends HTMLElement {...}
window.customElements.define('app-drawer', AppDrawer);

// Or use an anonymous class if you don't want a named constructor in current scope.
window.customElements.define('app-drawer', class extends HTMLElement {...});
```

Example usage:

```
<app-drawer></app-drawer>
```

Figure 2 Example of a custom element [25]

In figure 2 the class keyword is used to identify that what comes after, is a stand-alone class extending the HTMLElement. The content of the class is included in the brackets “{}”. After it, comes a window’s function that allows the use of the class as an html tag.

After you define your custom element, you can now call upon simply by using a tag mark “<>” and the name you gave to the component. In the case of our example, it is called app-drawer.

3.1.2 Shadow DOM

A DOM or Document Object Model is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure where each node is an object representing a part of the document.[37]

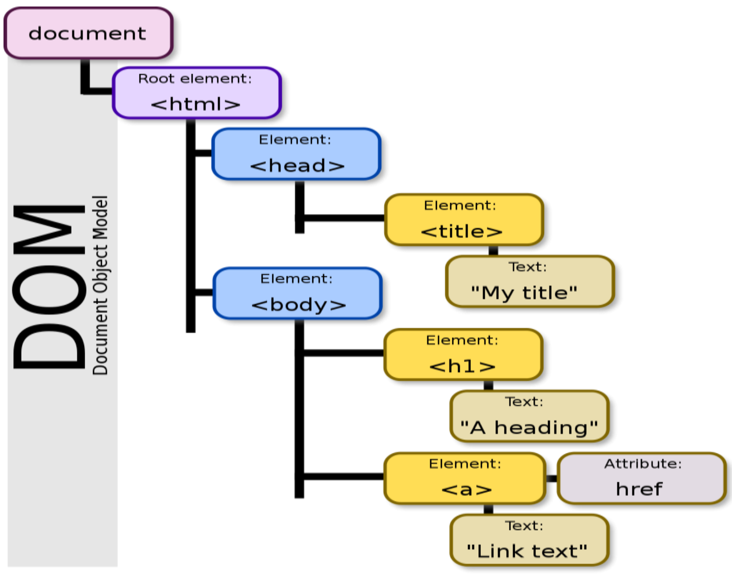


Figure 3 Example of DOM hierarchy in an HTML document [26]

Figure 3 shows the concept of DOM found in all HTML based websites and apps. It shows the tree framework that is the DOM.

A Shadow DOM is a tree of DOM nodes. It provides a separate DOM tree for your component so that the component would have a total encapsulation and the functionality of the components would not spill to other elements in the HTML. A shadow DOM forms its own scope. For example, a shadow DOM subtree can contain IDs and styles that overlap with IDs and styles in the document (the original HTML document), but they do not clash with the IDs in the original tree. It provides encapsulation for a subtree (children of the parent component) from the parent page. This subtree contains its own markup, CSS, JavaScript, or any asset that can be included in a web page without any of the assets impacting the parent page, or vice versa.

The shadow DOM consists of two major elements that define it:

Shadow Root: is a root node for the shadow DOM branch. Creating a shadow root on a node point on the parent page makes the said node a shadow host.

Shadow Host: is a regular node that contains within it a shadow subtree. Any child node that resides under the shadow host is still selectable except the shadow root.

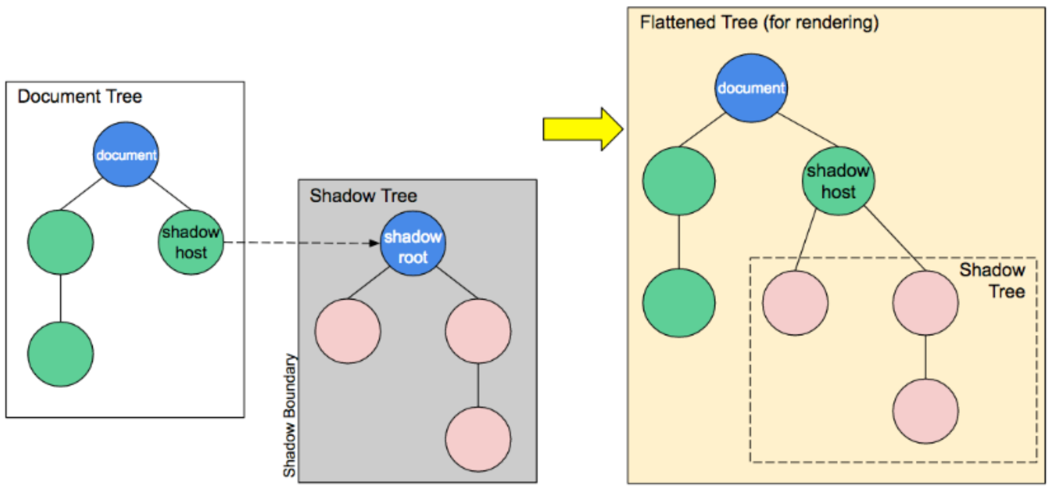
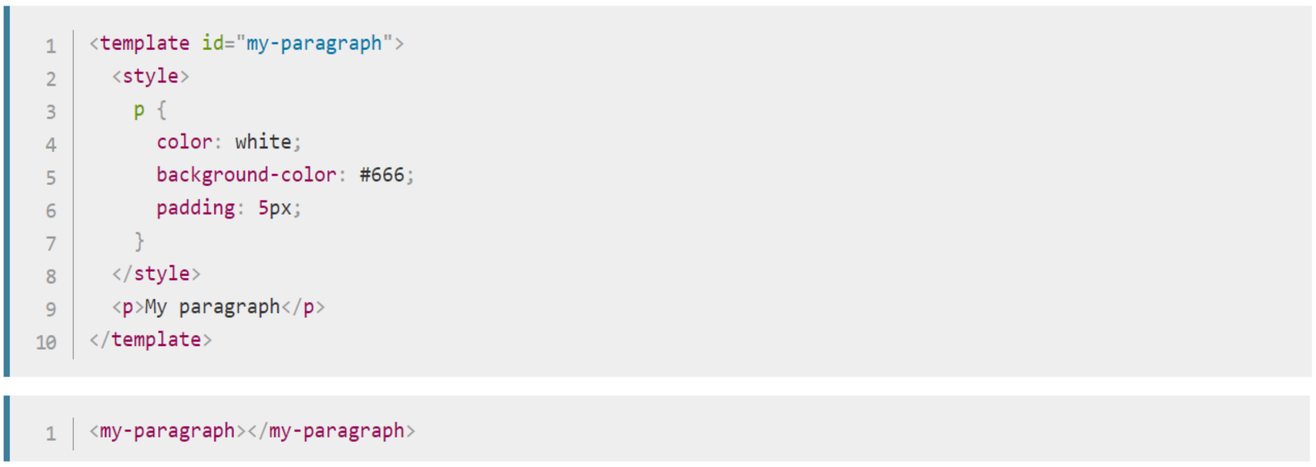


Figure 4 Interaction between DOM and Shadow DOM [28]

Figure 4 shows two types of DOM, one within the other, the second node in the document tree (named shadow host) is an access point to a shadow DOM that includes its own html elements. To the right of the figure, a flattened tree is shown which describes how a shadow DOM interacts with the DOM and the browser.

3.1.3 Template

The ability to create a reusable DOM structure using a new <template> element which describes a standard DOM-based approach for client-side templating. It allows you to declare fragments of HTML that will be inactive on the page load and instantiate later.



```html
<template id="my-paragraph">
  <style>
    p {
      color: white;
      background-color: #666;
      padding: 5px;
    }
  </style>
  <p>My paragraph</p>
</template>
```

```html
<my-paragraph></my-paragraph>
```

Figure 5 Example of Template [28]

Figure 5 describes how a template is written. It includes normal HTML and CSS code, to create a new HTML element called “my-paragraph”.

Using the <template> tag gives us a few new properties to use:

The content being inactive, which means your markup is hidden in the DOM and does not render. So, any content within the template would not have any side effects, scripts would not run, images would not load, and audio would not play until your template is used. Moreover, the content is not considered to be part of the document, so using statements, such as document.getElementById() on the main page, would not return any child of the template. Templates can be placed anywhere inside of the <head>, <body>, or <frameset> and can contain any type of content.

3.2 challenges of web components

Implementation: The W3C web component specification is very new to the browser technology and not completely implemented by the browsers.

Shared resource: A web component has its own scoped resources. There may be cases where some of the resources between the components are common.

Performance: Increase in the number of web components takes more time to get used inside the DOM.

Scalability: By default, attributes of Web Components can only take primitive values, such as string, number and Boolean. Changes of the values of these attributes can be watched and reacted to through the attributeChangedCallback method. Attributes cannot take complex values like objects and arrays, so in practice, data binding can only be done through attributeChangedCallback with primitive values.

SEO: is the process of optimizing a website or an app to be easily mapped and ranked by search engines. In a web component, the HTML code is inside the component template. This different strategy of expressing your HTML creates problems in the search engine because its crawlers are used to the HTML being the main file on a website.

The Lack of Statelessness: Web Components are extended from the HTML Element, which is the base class for all HTML tags. By nature, Web Components are class-based. That is why they are stateless, and because of that it is quite hard to build an app to scale for large use. Web components frameworks, like React, provide works around for this problem. In the case of React they stated with function components and then they were replaced by React Hooks.

Theming Web Components: web components can use a Shadow DOM which provides a scoped CSS. So components can be styled with CSS from the inside and the style only applies to the components. This approach solves a lot of problems with naming scopes but at the same time it limits the possibility of a consumer of the web components to style it the way they want. Of course, there is a way for consumers to style the component but the properties of said component should be explicitly exposed. Some browsers, like Chrome and Firefox, support CSS Shadow Parts which makes theming easier.

No server-side rendering: Web components rely on the Shadow DOM so there is no way to have server-side rendering as it is done by the browser.

3.3 Different Web Component Libraries

Web components can be hard to write from scratch as it requires writing a lot of boilerplate code and web component specification is not completely implemented by the browsers. However, there are many libraries with polyfill support that make creating custom web components a straightforward task and saves a lot of time and effort. You do not need to use a library to create and share custom web components but if you are writing a lot of custom elements, using a library can make your code simpler and cleaner. However, when choosing a library, you have to make sure to find one with a large feature set to cover your use cases and should be able to create components as stand-alone elements and without implementation details leak and it should have a high value-to-payload ratio which means that they provide a lot of value in proportion to their download size.

In recent years, web components specifications have been used to create several different libraries, with their own specific features and challenges. Some of these libraries became really popular in the web development community because of how versatile and efficient they are.

Here are the most popular libraries with their definitions and their features:

- Vue.JS
- Angular
- React

In the following chapter, you will have the chance to understand these different frameworks in depth.

4 VUE.JS

4.1 What is VUE.JS

Vue.JS is a JavaScript framework for building user interfaces powering sophisticated Single-Page Applications when used with modern tools and support libraries.

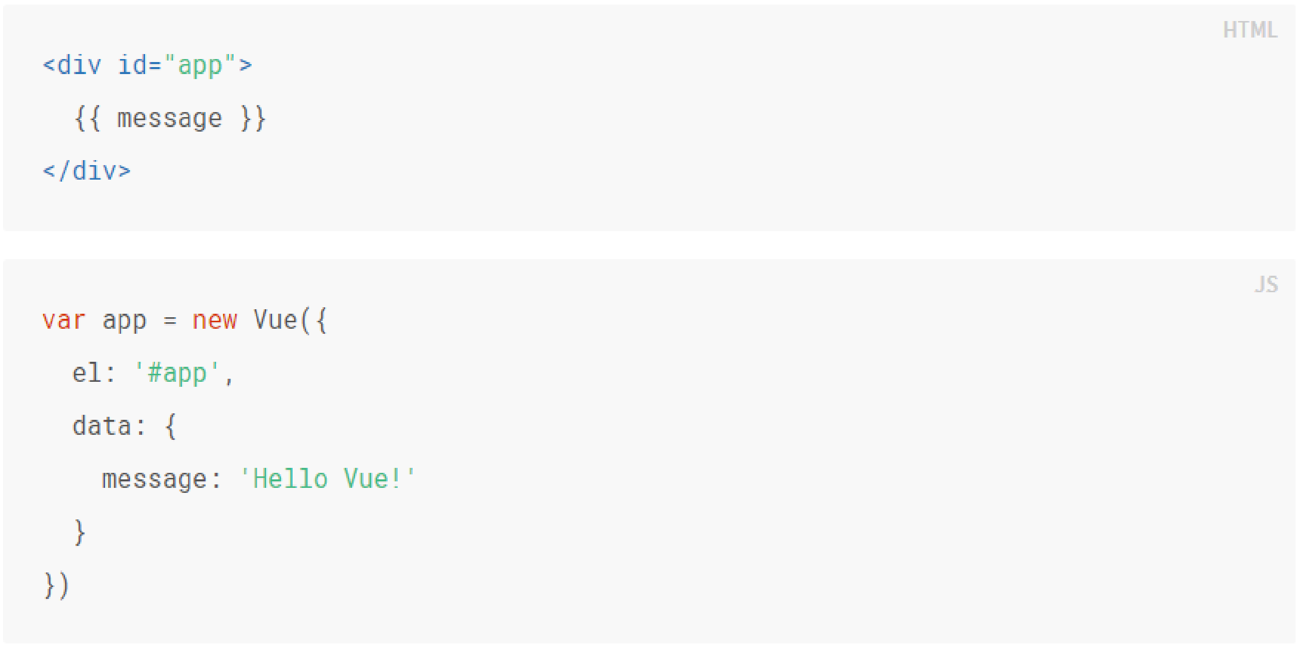
A single-page application (SPA) is a web application or website that interacts with the web browser by dynamically rewriting the current web page with new data from the web server, instead of the default method of the browser loading entire new pages.[30]

Vue.JS was created by Evan You, and it is maintained by him and the rest of the active core team members. The original thought behind developing Vue was that after Evan worked for Google using Angular, he decided to build something similar to Angular but only include the parts of it he liked in order to keep the framework light. His thoughts are summed up in this quote “I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight."[36]

One specialty about Vue is that it is entirely developed by the open source community and not a large enterprise. It started out as a hobby project by You until he decided to quit his job and work full-time on it. The financing was completely done through Patreon. Patreon is a community where people can become a subscriber for a specific project that they want to support with a monthly payment. The supporters of the Vue project already contributed more than 4,000$ in total per month. As of April 2020, this figure has increased to more than 16,000$[20]. This rise in popularity can be understood by looking at other related community pages: On GitHub, success and popularity is displayed by the number of stars a repository or project has. Vue has gained more than 40,000 stars in the course of 2017 making it the highest rising project overall of that year. Part of this derives from the strong bonding and extensive support from the PHP community, especially Laravel, where Vue is used as the default view engine.

4.2 Structure

This section will discuss the code structure of Vue.JS app:



```html
<div id="app">
  {{ message }}
</div>
```

```js
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

Figure 6 Simple Vue.JS code [22]

The figure above shows the variable “app” which contains an object of the Vue class that holds two properties. The first one is called “el” which refers to an element and it connects the vue.JS class variable to the html element with the ID “app”. The second property is an array that holds a property called message. by using the “{{}}”. You can have your variable render directly in your element.

The app variable created using the Vue class is called a view-model.

By attaching your html to the view-model, it gains the total control of the html element and can perform different actions, such as input handling and conditional rendering all within the Vue instance.

4.3 Features:

4.3.1 Components:

Vue components are an extension of basic HTML elements used to encapsulate code for better usability. At a high level, components are custom elements to which the Vue’s compiler (the compiler serves the same role as the Java virtual machine serves, by translating the Vue syntax into The HTML code the browser can understand) attaches behavior. In Vue, a component is called a Vue instance.

4.3.2 Templates:

Vue uses an HTML-based template by binding it to a Vue instance (the part of the component that holds the logic functions). All Vue templates are valid HTML code that can be parsed by any browser and/or an HTML parser. Vue compiles the templates into a virtual DOM. The virtual DOM allows Vue to render components in its memory before updating the browser. Combined with the reactivity system, the template-based approach allows Vue to calculate the minimal number of components to re-render and apply the minimal amount of DOM manipulations when the app state changes.

4.3.3 Reactivity:

Simply put, reactivity refers to the process of tracking internal state changes that accrue within a Vue component, so the system knows precisely when to re-render, and which components to re-render if changes are made. Vue.JS achieves this by mapping the properties of every components and attaching a setter/getter functions, these functions are invisible to the user, but they allow Vue.JS to perform dependency tracking and re-rendering when properties are modified.[29]

4.3.4 Transitions:

Vue supports transition effects in multiple ways: Automatically apply classes for CSS transitions and animations.

- Integrating third-party CSS animation libraries, such as Animate.css.
- Using JavaScript to directly manipulate the DOM during transition hooks.
- Integrating third-party JavaScript animation libraries, such as Velocity.js.

4.3.5 Routing:

Vue does not come with a prebuilt routing system but it does not mean that Vue is not capable of routing. The open source "vue-router" package provides an API to update the application's URL and supports navigation history. It supports nested routes for components within a component. With vue-router, components must merely be mapped to the appropriate routes, and the library will provide a complete routing system for your components.

4.3.6 Server-Side Rendering

Vue.JS makes server side rendering possible by rendering components into HTML strings on the server. Then it sends them directly to the browser, and finally it integrates it into your static HTML, This process is known as SSR or Server-Side Rendering

Some of the benefits of SSR are

- A better SEO, as the search engine crawlers will directly see the fully rendered page.
- Faster rendering time. Server-rendered markup does not need to wait until all JavaScript has been downloaded and executed to be displayed and it is independent of the client’s internet speed, so the client will see a fully rendered page sooner.

4.4 Problems of Vue.JS

Smaller market share and community – Compared to other frameworks, e.g. React or Angular, Vue.JS has a much smaller community of developers surrounding it due to it being new on the market. This results in lack of resources and documents which makes dealing with bugs and errors harder compared to its close competitors.

Problematic integration into larger projects – When trying to integrate Vue to existing projects, developers often find out that Vue.js causes issues with compatibility with other libraries.

English documentation – some stages in the development of Vue.js projects might be a little more complicated because of the lack of full English documentation. However, as more and more material in the Vue.js documentation is being published by the team behind Vue or the community, that problem is bound to disappear soon.

5 ANGULAR

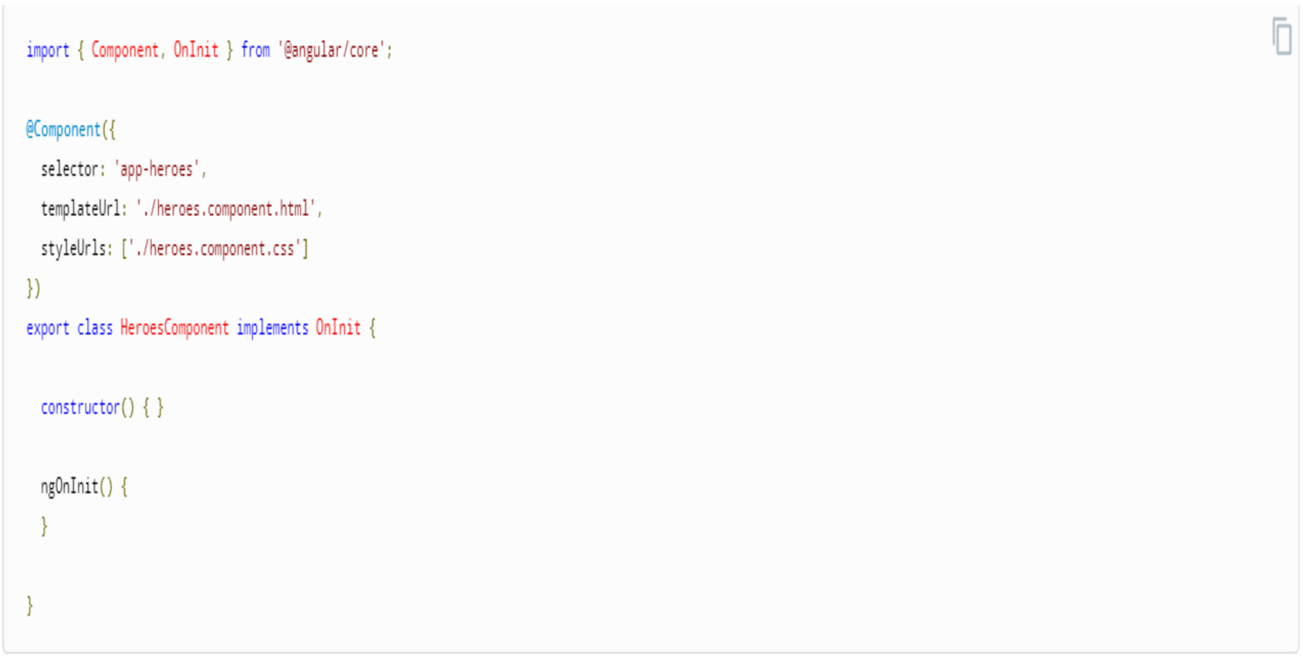
AngularJS was the first framework to fully integrate the web component technology on a grand scale and it became the most used JavaScript frameworks, as it was initially introduced by Google corporation in 2012. Angular is built on Model-View-Controller concept in mind. It was created by a Google engineer in efforts to prove that the approach his fellow engineers were taking is slow and unreliable. He re-wrote a Google project that took his team months in just few weeks by using his newly created library. Angular introduced an approach to separating the logic side and the front-end side of the project. It did this by introducing Controllers which are object functions that handle the logic part of the application, and a View, which refers to the HTML part, and finally a Model, where functions that communicate with the other world reside, functions such as REST API calls and Database manipulation functions.

Nowadays, AngularJS is outdated and not in use. As it was introduced early on AngularJS showed the potential a framework built on this architecture could provide but it also showed the limitation to the initial release of angular.

To solve this Angular 2 was released with an overhauled design that got rid of controllers, scope and view with TypeScript's component approach.

TypeScript is an open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript and adds optional static typing to the language. TypeScript is designed for development of large applications and trans-compiles to JavaScript [31].

5.1 Structure

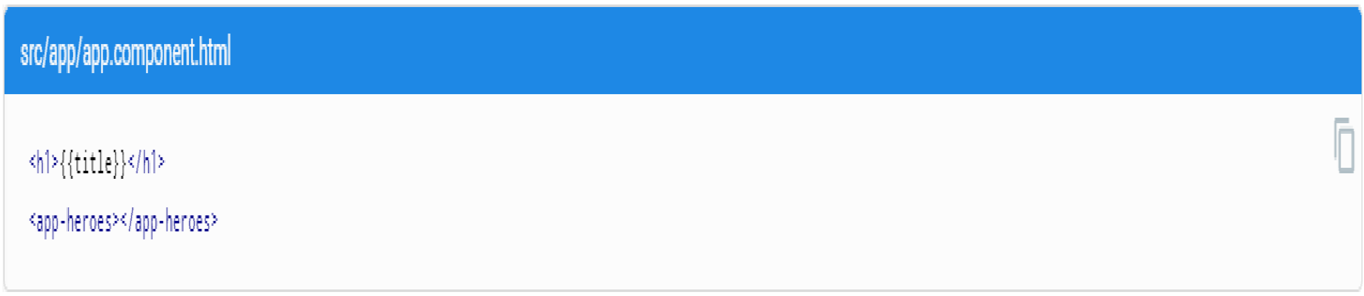


```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Figure 7 Basic source code of an angular component [32]

Figure 7 is a simple example of an Angular component. It first begins by importing the “Component” and “OnInit” from the core Angular library. These are the main functions that an Angular component will rely on. Then “@component” is used to mark a decorator function that specifies the Angular Metadata to be included, inside it you can find a “Selector” which matches the name of the HTML element that identifies this component. “templateUrl” refers to the file where the HTML data is hosted and finally “StyleUrls” refers to the location of the CSS files used to style this component.

The “ngOnInit()” is a lifecycle hook. Angular calls ngOnInit() shortly after creating a component. It's a good place to put initialization logic.



src/app/app.component.html

```
<h1>{{title}}</h1>
<app-heroes></app-heroes>
```

Figure 8 Use of Angular Component[32]

Like Vue and React, to use an Angular component, it merely requires the use of the selector within a “<>” tag, as demonstrated by the figure 8 above.

5.2 Features

5.2.1 Components

Every Angular application has at least one component, the root component which connects a component hierarchy with the page document object model (DOM). It is usually referred to as the App component or root-app component, in pure HTML it serves the same function as index.html serves. Each component has a defined class that contains application data and logic, and the HTML for the component is defined in a view file ‘.HTML’, to be displayed in a target environment.

After the @Component() decorator, the class immediately below it is the component class, and it contains the template and the metadata for the component.

5.2.2 Templates, directives, and data binding

A template holds the HTML to your component. A directive provides program logic, and binding functions allow you to move data between the template and directive allowing for the user input and output or function that executes when certain conditions are met. There are two types of data binding:

- Event binding allows you to execute functions when a user has made some sort of an event in the app. An example is the “ngOnInit” function in the figure 3 above
- Property binding lets you render values that are computed from your application data into the HTML.

Before an HTML element is displayed, Angular evaluates the directives and resolves the binding syntax (in figure 9 “{{title}}” will be exchanged for the value of the variable named title) in the template to show the HTML elements with the new data according to the program’s data and logic. Angular supports two-way data binding, meaning that changes in the DOM can affect your app’s flow, for example form an input from a user.

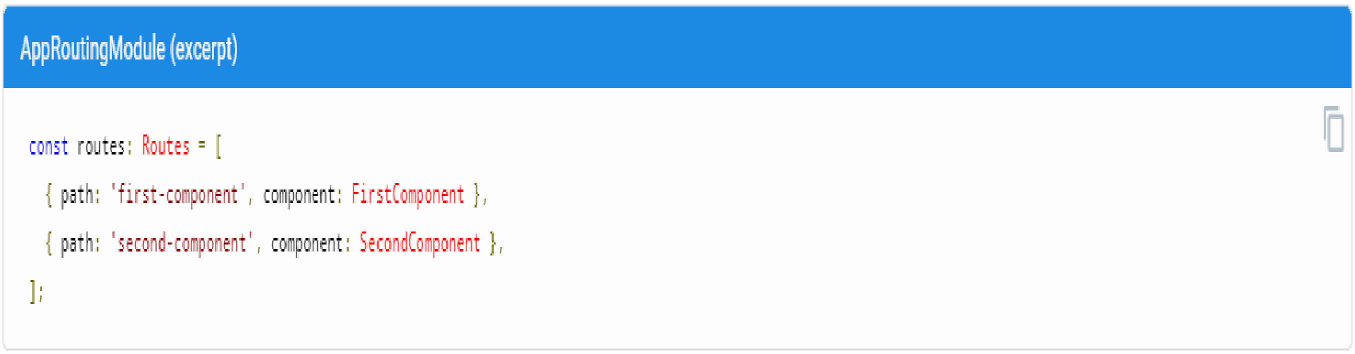
Templates also have “pipes” which are used by including a simple “|” next to the binded data. It helps improve the user experience by transforming values for display, for example, using pipes to display dates and currency values that are appropriate for a user's location. Angular provides predefined pipes for common transformations, and the user can also create new pipes.

5.2.3 Routing

The Angular Router module provides a routing service that defines a linking path between the different application states and UI hierarchies. It is the same as traditional browser navigation except that the Angular app does not have to re-render to navigate from one link to another.

The router model maps URL-like paths to their respective views/UI instead of pages. When a user interacts with the UI by clicking a link, that action normally loads a new page in the browser. The Angular router blocks the browser's default action and shows or hides view/UI hierarchies.

By setting you navigation rules (an example in figure 9 below), the router can figure out which link refers to what view. You can navigate to new views when the user clicks a button or a link. The router model has a browser like history feature, so the back and forward buttons work as well as in a normal browser.


AppRoutingModule (excerpt)

```
const routes: Routes = [
  { path: 'first-component', component: FirstComponent },
  { path: 'second-component', component: SecondComponent },
];
```

Figure 9 Example of Routing rules [33]

Figure 9 shows how to use the route module. To set the rules for routing, you need to create a constant that extends the Routes class. Within it, you can include paths to your components. The “Path” variable refers to the URL extension and “Component” refers to which component should be shown when the URL is invoked.

5.2.4 Modules

Angular Modules differ from JavaScript (ES) modules. An NgModule components can work in harmony with other user-built components, such as services, to form functional units which can be used with any component the user builds. Every Angular app has at least one module called the root module, named AppModule, which provides the functions necessary to launch the application. An app typically contains many functional modules. Like JavaScript modules, NgModules can import functions from other modules and allow their own functions to be exported to other modules. For example, the router module is used by importing the Router NgModule. [33]

5.3 Advantages of Angular:

- Angular presents you not only with the tools but also design patterns to build your project in a maintainable way. When an Angular application is made properly, the code is not a tangle of classes and methods that are hard to modify and test. The code is structured conveniently, and you would not need to spend much time in order to understand what is going on.

- Angular is built with TypeScript, which in turn relies on JS ES6. TypeScript is a superset of JS so the skills and knowledge transfers over, with the addition of new features like static typing, interfaces, classes, namespaces, decorators etc.
- With Angular, you already have lots of tools to start crafting the application right away. Directives gives HTML elements dynamic behavior, using FormControl gives more control over the HTML static forms and introduce various validation rules to accept input. asynchronous HTTP requests of various types allows for better data handling. routing is done with little hassle by using NgRouter.
- Components are decoupled. Angular removes tight coupling of various components in the application. Injections happen in NodeJS-style, so it allows for the replacement of various components with ease. [39]
- All DOM manipulation happens where it should happen. With Angular, you do not tightly couple presentation and the application’s logic because of the MVC architecture making your markup much cleaner and simpler.
- Testing is built into the framework. Angular apps are built to be thoroughly tested and it supports both unit and end-to-end testing using libraries like Jasmine and Karma.
- Angular is mobile and desktop-ready, meaning you have one framework for multiple platforms.
- Angular is actively maintained and has a large community and ecosystem. You can find plenty of materials on this framework as well as many community-built tools.[32]

5.4 Challenges of Angular:

- TypeScript is a superset of JavaScript, so you will need to be comfortable with it as well.

- It is a good idea to get the grasp of the Angular CLI to speed up the development process even further.
- Node’s package manager is used extensively to install Angular itself and other components, so you will need to be comfortable with that as well.
- Learning how to set up a task runner like Gulp or Grunt can come in really handy, as there can be lots of things to be done before the application is actually deployed to production.
- While developing the app, it is vital to be able to debug the code, so you should know how to work with debugging tools like Augury.[40]

6 REACT

React is the most popular web component framework at the moment With over 5.5 million weekly downloads.

I

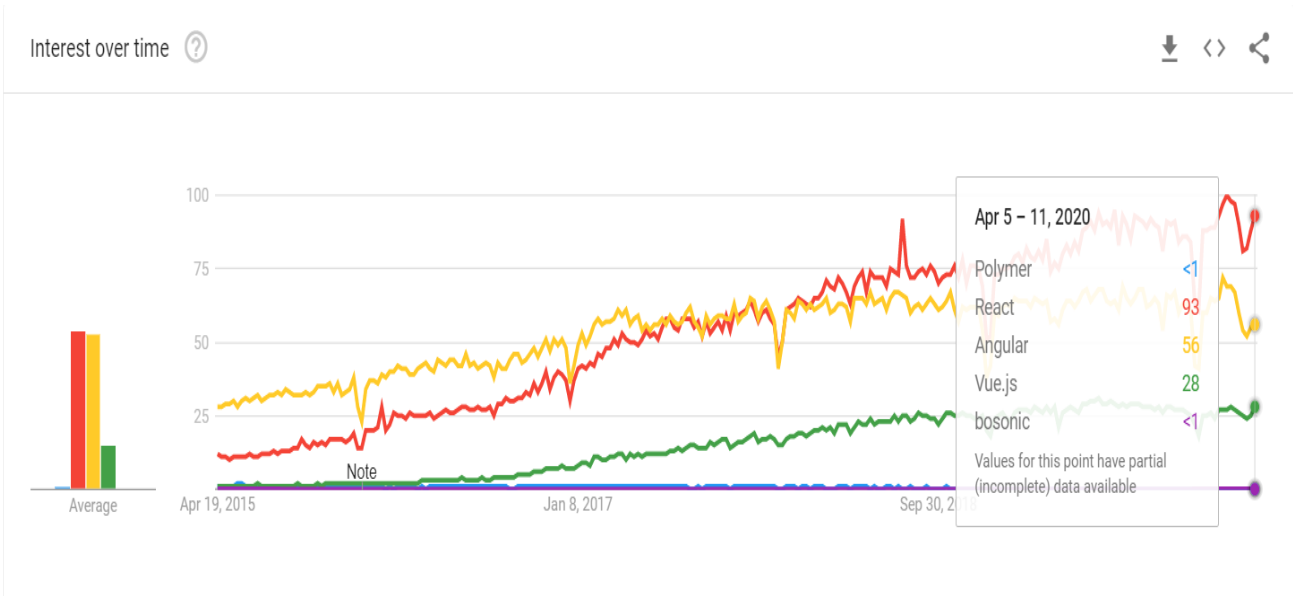


Figure 10. Comparison of different web frameworks - Google Trends

Figure 10 shows a comparison between Angular, React and Vue.JS. React is the most popular framework/library followed by Angular and then Vue.JS.

6.1 What Is React?

React is a JavaScript library mainly used to create user interfaces. It was built at Facebook to address some of the challenges of building a large-scale, data-driven website. React was built to preserve the speed of JavaScript and use a new way of rendering webpages, making them dynamic and responsive to user input. The React project was started by Jordan Walke, a Facebook software engineer, in 2011. His goal at the time was to simplify and speed up the development process while maintaining a comfortable user experience, he decided to create a library that would allow the building of a web interface with JavaScript. Facebook was confronted with a major user experience challenge which is building a dynamic UI with high performance [4]. For example, Facebook wanted to make news feed updates happen simultaneously with people using chat. The idea seemed impossible due to the lack of technology to support, but in 2011 Facebook released the ReactJS library on the basis of JavaScript. While testing the framework they realized that ReactJS was faster than any other implementation of its kind (mainly Angular). When React was released in 2013, the project was initially viewed with some skepticism because the conventions of React are quite unique. So the web development community was both interested and seemingly baffled by the changes React was doing to the idea of always separating your application’s logic from its UI. React is managed by Facebook and the open source community [4].

React challenged conventions that up to that point were the de-facto standards for JavaScript framework best practices. React does this by introducing a new architecture and shifting the status quo of what creating a scalable and maintainable JavaScript applications and user interfaces has to be. React provided a shift in front-end development and it also came with a rich set of features that made developing a single-page application or user interfaces easier for developers from many skill levels from those who have just been introduced to JavaScript, to experienced developers [4].

A common misconception about React is that it is a full-scale JavaScript framework on the level of other frameworks such as Backbone, Knockout.js, AngularJS, Ember, Dojo, or any of the numerous MVC frameworks that exist. However, React makes up only one particular piece of what these frameworks do. React, in its simplest form, takes on the view part from MVC, MVVM, or MV* frameworks. As stated in the previous paragraph, React is a way to describe the user interface of an application and a mechanism to change that over time as data changes [4].

React can be used as a base in the development of single-page applications, websites or mobile applications. However, React is only concerned with rendering data to the DOM, it allows the user to effectively re-construct the DOM in JavaScript and push only those changes to the DOM which have actually occurred. this means that a React application would require additional libraries for state management and routing. Redux and React Router are the libraries commonly used with React [4].

React works in declarative code meaning that the code describes what we want instead of saying how to do it, as you would with imperative code. At its core, declarative code is like going to a restaurant and ordering a meal. You tell the waiter what your choice is, but you do not tell the chef how to cook it. Declarative code describes the end result, but it does not act as a step-by-step guide of how to do it. In practice, that means that the declarative code is small, easier to understand and modify, and creates less bugs. An example of declarative code would be something like:

<header>

<SlideShow>

<Paragraph>

Each one of these tags is an example of a declarative code as it does not tell us how the header, for example, works but mainly declares that there should be a component, or a piece of code called header, in this place holder like a normal HTML tag.

6.2 What Problems Does React Solve?

React does not intend to solve every problem in user interface design and front-end development. React solves a specific set of problems, and in general, it is focused on a single problem. According to Facebook, React builds large-scale user interfaces with data that changes over time [34]. Building large-scale user interfaces with data that changes over time could probably be something that many web developers can relate to in their own work or hobby coding experiences as most web applications are data driven and receive new pieces of data every second. Problems arise when your project code is no longer maintainable. You must add extra pieces of code to get the data to bind properly whenever you add a new feature or a new HTML code. Sometimes you must restructure an application because a secondary business requirement has inadvertently broken the way the interface renders a few interactions after the user starts a task. All of this leads to user interfaces that are filled with bugs and highly interconnected so changes in one part of your code can break your interface. These are all problems that React attempts to solve.

Previous sections have discussed Angular and how the first versions of Angular, AngularJS, relied on a Model-View-Controller architecture with a two-way data binding which means that an application must contain views that listen to models, and then the views would update their html based on either user interaction or the data coming from the model. In a small application this is not a great performance or productivity obstacle, but the scale of the application will inevitably grow as new models and views are added to the application. This quickly becomes more and more complicated. Items that are deep in the rendering queue or in a different model are now affecting the output of other items. In many cases an update that happens may not even be fully known by the developer because maintaining a tracking mechanism of different processes becomes increasingly difficult. This makes developing and testing your code harder and ultimately leads to more bugs, which means that it becomes harder to develop a method or new feature and release it. The code is now less predictable, less understandable and development time has skyrocketed. This is exactly the problem that React sets out to solve [4].

Due to React being only responsible for the view part of an application, the development process requires a stack of various technologies to be effective:

- Compiler (for JSX5)
- Modules (and an appropriate loader) for the application structure

- Build process
- Routing
- State management

6.3 Structure

6.3.1 React Components

React components are the main building blocks when structuring a React application. React components are created by extending from the base “React.Component” class using ES6. Or, more traditionally, using the “React.createClass” method

```
class myComponent extends React.Component {
                render() {
                                return (
                                                Hello World
    ); } }
```

Components as Functions, These are components created using a pure JS functions that return exactly one React element. The name of the component is the name of the function. This approach, however, has its limitations: Neither can the state be altered, nor can you access lifecycle methods (e.g., componentDidMount) [4]

Components as ES6 Classes, In this approach, a JS class is used to create the component. The class always has to extend the super class “React.Component”. Furthermore, the HTML should be contained in a Render() method which can only return one root element meaning that all your HTML has to be contained in one HTML tag. The aforementioned limitations of the first approach do not apply here: Classes can utilize states, lifecycle hooks and more. [4]

React’ best practices specify that a developer should always use a function component whenever possible. That should support efficiency and re-usability of the simple but often needed components. Class components are usually located at a high position in the hierarchical tree of the application. They work as parent components and handle the various states that are then passed down to the child function components. [4]

The React.Component class also has lifecycle events that manage the creation and destruction of component instances. Here are some of the most used events in the React development cycle (figure 11):

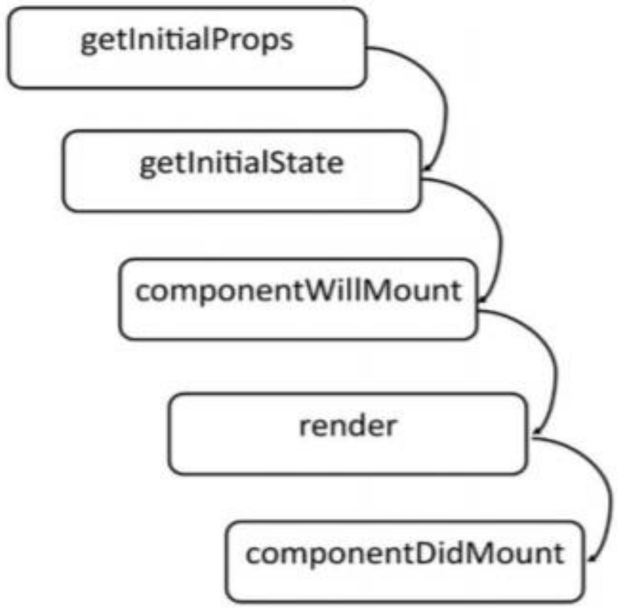


Figure 11 Function invocation order during the initial render of a React component [4]

ComponentWillMount:

componentWillMount is a lifecycle event that executes before React renders the component class to your DOM. “componentWillMount” is executed before the initial render of your component.

componentDidMount:

componentDidMount is a client-side function of React, after the component has been rendered to the DOM, this function executes.
componentDidMount is a good place to write your component’s logic or initialize any third-party JavaScript that requires a DOM. For example, a drag-and-drop library or a library that handles touch events.

componentWillReceiveProps:

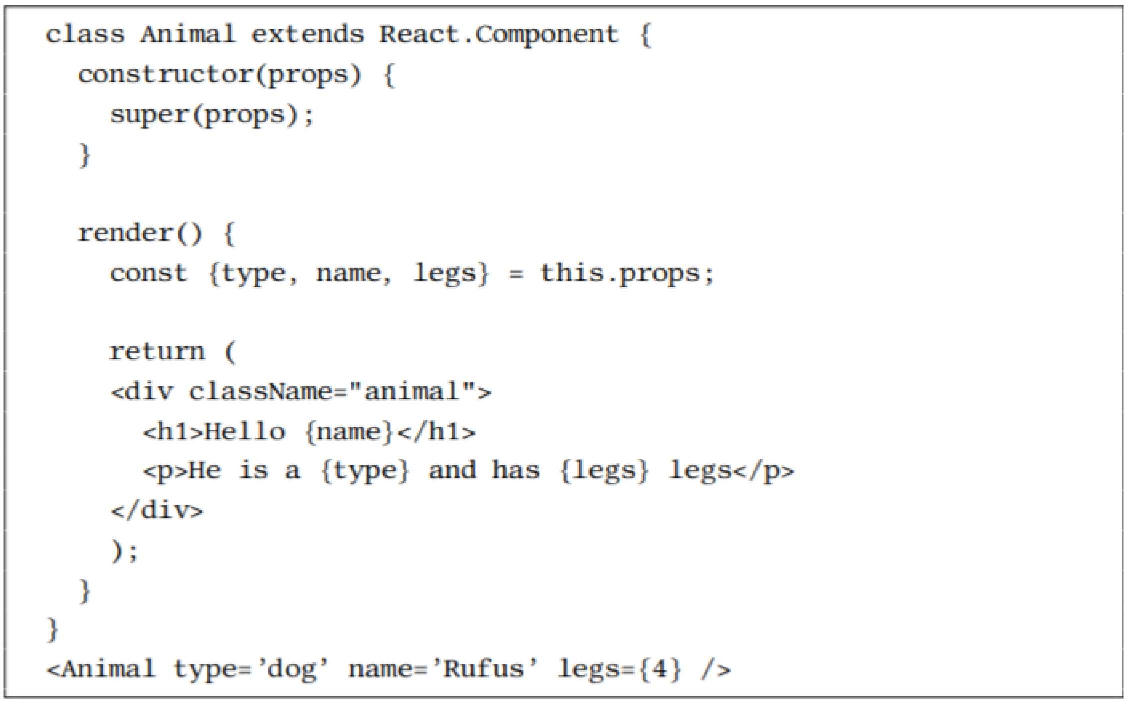
componentWillReceiveProps is executed when the component will be receiving props. This function is called every time there is a prop change, but never on the first render.

ComponentWillUnmount

Mounting refers to when a component is rendered to the DOM. componentWillUnmount, would be invoked immediately before the component is no longer needed. This method is used to unmount/destroy the component. When a parent component is unmounted, its children are unmounted first.

Previously the concept of Props and States have been discussed but never explained, the following paragraphs explain these concepts in more details

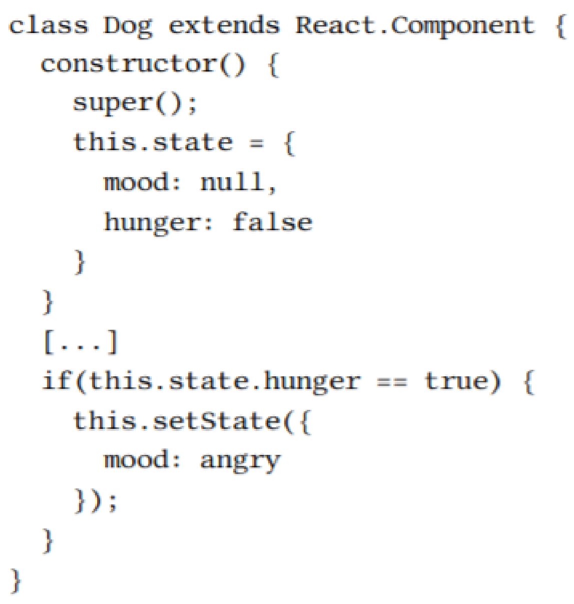
Props stands for properties and is being used for passing data from one component to another. Data with props can only be passed in a uni-directional flow meaning that it flows one way from a parent to a child Furthermore, data coming from the parent cannot be changed by child components (figure 12) [4].



```
class Animal extends React.Component {
  constructor(props) {
    super(props);
  }

  render() {
    const {type, name, legs} = this.props;

    return (
    <div className="animal">
      <h1>Hello {name}</h1>
      <p>He is a {type} and has {legs} legs</p>
    </div>
    );
  }
}
<Animal type='dog' name='Rufus' legs={4} />
```

Figure 12 A case use of Props in a React Component – Book: introduction to React

State allows components to create data and managed locally. So unlike props, components cannot pass data with state, but they can create and manage it within a said component. State should not be modified by direct access, but instead a special method called setState() is invoked. One of the best features of state that its counterpart (props) does not have is that when state changes, React gets informed and immediately re-renders the DOM, not the whole DOM, but only the component to which the state belongs. State should be used very carefully because it re-renders the components and could lead to infinite rendering if there are memory leaks in the component. This is why most developers prefer to use functional components and keep class-based components to an absolute minimum (figure 13) [4].



```
class Dog extends React.Component {
  constructor() {
    super();
    this.state = {
      mood: null,
      hunger: false
    }
  }
  [...]
  if(this.state.hunger == true) {
    this.setState({
      mood: angry
    });
  }
}
```

Figure 13 A Use Case Of State in a React Component [4]

6.4 Features

JSX

or JavaScript Extension is an XML/HTML-like syntax developed by Facebook to be used by React. It extends ECMAScript so that XML/HTML code can be encapsulated by JavaScript/React code. JSX is intended to be used by compilers like Babel to compile HTML found in JavaScript files into standard JavaScript objects which a JavaScript engine will parse.

By using JSX, a developer can write HTML/XML-like structures in the same file where they write JavaScript code. Then the Babel compiler will transform these expressions into actual JavaScript code. Normally, developers used to put JavaScript inside HTML but instead JSX allows the user to put HTML into JavaScript.[20]

JSX is not a requirement to write good React code. It is however, the agreed upon best practice when it comes to React development. [4]

Flux

Flux is a design pattern whose purpose is to keep date flowing in one direction, from a parent to a child, as it is the preferred method of data handling in React. Before Flux was introduced by Facebook, web development architecture was dominated by variations of the MVC design pattern. Flux is an alternative design pattern to MVC that complements the functional approach of React. [4]

Routing

React does not natively support routing but routing can be handled with the React Router library. The React Router is based on the same principles as the Ember-Router, a routing framework for JavaScript. [4]

State Management

As your application grows in size and complexity, the need to keep track of your states grows as well. React applications often use Redux. Redux is a state management library. In React version 16, Hooks were introduced to provide an easy way for state management without using an outside library,

Hooks are functions that attach to a State of a certain component and allows the user to track it or interact with it by the use of multiple feathers. [4]

6.5 Advantages of React

6.5.1 Higher code stability

React provides greater code stability as data flows in one direction only. Whenever the developers use data binding techniques, then the changes in the child structure would not affect parent structures as opposed to other frameworks.

A two-way data flow has a significant disadvantage when it comes to maintaining or updating code over a long length of time in this architecture. Child elements may affect the parent if changed. Facebook removed these issues in React JS, making it just the view system with an unidirectional dataflow. [4]

6.5.2 Easy to understand

A JavaScript developer can quickly and easily work with React. A basic knowledge of HTML, CSS and JS will efficiently serve the purpose. For mobile app development, the React Native version offers multiple advantages over its competitors and it is similar to React making accessing the mobile app business easy for a novice developer.

6.5.3 Strong Open Source Community

The React library is available to more than 2,000 contributors on GitHub. There is an active community of Reactjs developers on significant platforms, e.g. StackOverflow, Slack, Freenode IRC and different types of forums, that provide a satisfactory answer to the questions. The strong React community will make the transition more accessible for the new developers. React was released as an open-source project by Facebook which means that ReactJS uses all advantages of free access – a lot of useful applications and additional tools from off-company developers and at least two features (batching and pruning) are community developed [41]

6.5.4 Reusability of Components

Facebook added the functionality to reuse the code components that in turn have made the life of React developers easy. It saves the time of the developers and frees them from the hassle of writing the same code repetitively. In case of design technology, it reuses the same assets. otherwise, designers would have to draw corporate logos, over and over again. It is pretty obvious that reusing is a design efficiency. In programming, this process is more difficult. System upgrades often turn into a complicated task as every change can affect the work of other components in an entangled system. React does not suffer from these problems because all React components are isolated and a change in one does not affect others. This allows the reusing of components without affecting the other components and their functionalities [41]

6.5.5 Quick rendering

The virtual DOM in ReactJS helps in removing the bottlenecks from code rendering that helps in making the process smoother. With minimal re rendering, the Virtual DOM will apply changes virtually and perform DOM changes. It minimizes the time required to make the changes to the DOM and offers a fast performance. Users can notice it when writing in Facebook chat and seeing a simultaneously updating news feed. Moreover, in React React elements are already connected to the DOM. This approach allows developers to work with UI-objects faster and use hot reloading (applying changes in a real-time mode). This made programming faster. [41]

6.5.6 Helpful Developer Toolkit

React enjoys one of the best debugging and design tools. “React Developer Tools”, a browser extension can be downloaded for Chrome and Firefox. It makes development easier due to its extensive toolkit for developers. Being a great extension, it allows to observe reactive components and inspect the current state and props of different components. [41]

6.6 Challenges of React

6.6.1 High pace of development

React is fairly new in the grand scheme of things and this means that it is constantly evolving to accommodate a wide range of features as demands for them grow. And sometimes big features lead to a major change in the framework where developers have to learn the framework from the ground up over and over with every major version. [41]

6.6.2 Poor documentation

With the fast pace of the development React is showing, constant releases and tools are being presented into the ecosystem and this accelerating pace leaves no room for developers to write proper documentations that could be looked at by the community and checked for efficiency. So React has a number of poorly written documentations [41]

6.6.3 JSX as a barrier

JSX is a major technology in React development, which makes it another barrier of entry for new developer. Updating and the growing complexity of this library poses problems even for experienced developers. [41]

6.6.4 SEO hassle

There have been reports about problems in SEO concerning dynamic web pages and client-side rendered content, but Google confirmed that their web crawlers are capable of reading dynamic content. However, you still have to do some testing to ensure that your app is indexed by Google as there were problems reported by some users. While this is not a big problem, SEO adds up to your development effort. [41]

7 ANALYSIS OF THE 3 DIFFERENT FRAMEWORKS

The following sections will cover various aspects of decision making in terms of adapting a new framework. While the preceding chapters contained many, more technical related parts of the technologies, this comparison chapter will pick the most relevant ones and furthermore include comparisons between the three different frameworks based on the following criteria:

- stability: has the framework frequently have a stable developer ready version with a manageable number of bugs?
- Technical aspects and architecture: What feature each framework provides and how does their underlying architecture differ to solve different problems?
- learning curve: How long does it take to get started with a framework?
- size: How big is the framework and how big are the compiled final projects produced by using these frameworks respective CLI?
- Runtime Performance: How well does the framework render its elements, how fast does it execute the code?

7.1 Stability

Stability generally refers to how reliable the framework is, how many bugs can an average developer run to, how fast those bugs are patched, and do updates regularly force you to redesign your code.

Angular has Google as a parent company. That name comes with a certain reputation for reliable code especially In the area of web, and Angular has a longer history than the other two to prove its stability. Angular had only one major redesign that made developers migrate their code. When AngularJS became Angular 2, after that updates were seamless and retrofitting with the back versions. As for bugs, Angular has a number of bugs but they are patched at a fast pace and due to the large community bugs are also flagged at a higher rate.[3]

React also has the backing of Facebook and a large open source community making it the fastest changing framework between the three. That comes with a certain inconvenience for developers who have to update their projects regularly because certain functions are struck down in the newer updates, and also a large number of bugs, but the support is reliable as well and most bugs are easily fixable with a work around or an update.

Vue.JS is the new neighbor and it is still an unknown quantity, but most of the bugs that have been flagged have been fixed at a good rate that a huge number of developers are switching over to it for its simplicity which makes bugs rarer. [24]

7.2 Technical Aspects and Architecture

7.2.1 the main development language

While Vue is liberal in this case offering developers more than one way to write their code (e.g., ECMAScript5/6 or TypeScript since 2.5+) and also emphasizes doing that, Angular is more restrictive: Although it is possible to develop an application with Vanilla JS, all official resources as well as most of the available tutorials and code snippets require the usage of TypeScript. While this also means an advantage in terms of being familiar for developers with any object-orientated background, it may make simpler projects more complex. This goes on with Angular being dependent on the Dependency Injection, a concept which is not widespread among the JavaScript ecosystem. React, for example, does not rely on it. However, React also comes with a quasi-restriction by enforcing the use of JSX for developing. It definitely is the most diverse approach of all three frameworks as JSX implies that HTML is strongly intermixed with JS. Also, the concept of states can be a barrier to adaption as state handling in React is done by Redux and can be another burden to clear.[24]

7.2.2 Components

Vue, React and Angular share a relatively similar approach to structuring their components: Both split up a template (HTML), a style (CSS) and logic (JS). While all these frameworks offer the option to handle these parts either in one file or in three separate ones, Angular definitely prefers the separation while Vue and React emphasize the single file approach even offering a special file extension for this purpose: .vue, .jsx. [24]

Single file components have the advantage as they force the developer to write components as slim as possible to ensure simplicity and re-usability.

7.2.3 Data Handling

Data handling is the approach to binding. React uses one-way data binding only. Angular and Vue offer a data flow in two ways by recommending the use of unidirectional data flow. However, two-way binding may seem convenient at first but can cause difficulty the more the application grows in terms of size and complexity. With two-way binding it can sometimes be hard to track which data gets updated and where. Also, bugs may occur more often. Therefore, it is recommended to use one-way binding as extensively as possible even though this results in a generally higher coding effort for the developers. [24]

7.3 Learning Curve

Developers who attempt to learn React need to have a profound knowledge of JavaScript because it is the main programming language for this framework. One could say that React entails more JavaScript. This might become relevant when a company has designers that work close to the code. In the real world, it could eventually be hard to find designers that know how to modify JSX code in order to change structural or styling related parts. For this task working with HTML templates would be much easier. React breaks with long-term best practice of developers separating UI templates and in-lining JS logic but the usage of JSX causes them to be intermixed again. This can be seen as both positive and negative depending on the judge’s philosophy in the subject. Due to the use of JSX, developers have to become familiar with this new form of syntax to get started and due to the lack of state management, developers also have to use Redux. [24]

Vue.JS is the easiest to learn compared to the other two frameworks, as it mainly uses ES6 for developing and it does not require any additional compilers or transpilers (compilers that are specific to web component frameworks) as it only utilizes the basic web technologies. [24]

Angular has a rather steep learning curve. Setting up a project with the CLI only takes a few minutes. From there on, developers can take the tutorial on the official Angular homepage. It must be mentioned that the documentation is extensive as well as comprehensive. As Angular is a complete framework and it provides a full set of homogeneous APIs, features and tool. It requires time and experience to gain an understanding of all the features that are offered. The framework takes away many decisions for the developer on how to handle certain situations. This may be viewed as an imposed limitation. However, one could argue that especially this argument poses an advantage for companies that hire new developers. In the Angular world, a new developer, who has experience in Angular, will be familiar with the code base of the company relatively faster as almost all Angular project are structured similarly. Furthermore, developers with a background in object-orientated programming will find TypeScript to be close to, e.g., Java or C# and easy to access. [24]

7.4 Size

In terms of size Vue and React are the lightest with a huge difference compared to Angular. Because Angular is a full fledge framework with many features that do not exist in the other two then while being considered to be frameworks by the community the other two are more accurately described as libraries.

An approximate size of the Angular framework is 500KB while Vue and React only come at 80Kb and 100kb respectively.

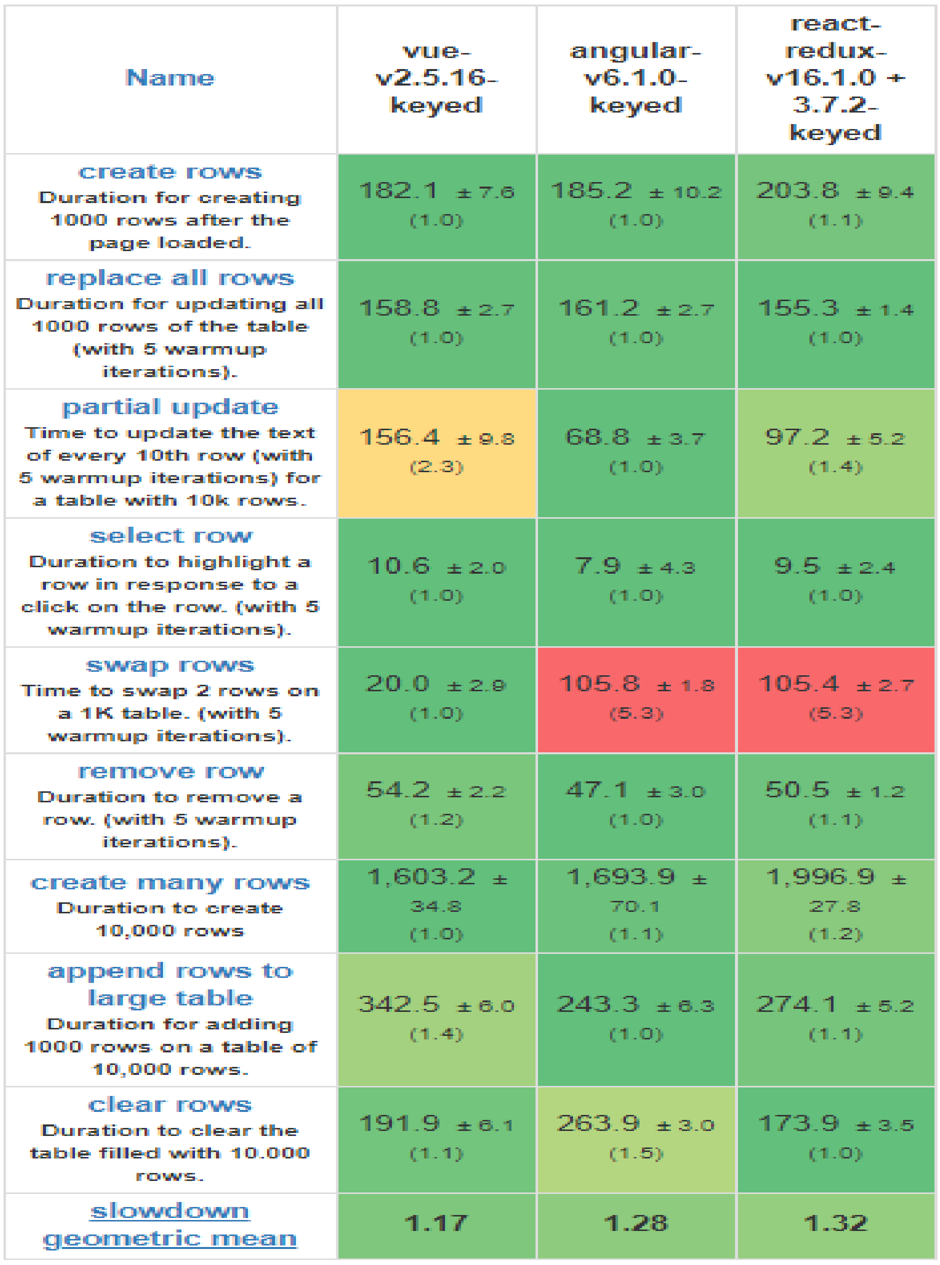
The latest versions of Angular, have been able to reduce the size of their projects. However, a Vue 2 project with Vue Router included weighs about 30KB

Gzipped. That is still significantly lighter than an AOT-compiled Angular application generated by Angular CLI that is approximately 65KB Gzipped.[23]

7.5 Runtime Performance:

React, Angular and Vue are exceptionally and similarly fast, so speed is unlikely to be a deciding factor when deciding from a first look, but due to different approaches to the DOM Manipulation, performance differences arise. Angular uses the real DOM and this affects its performance and ability to make dynamic applications and it results in the low performance of this framework. As for React, it uses a Virtual DOM and it is not dependent on the browser. The use of the virtual DOM allows it to manipulate the DOM with a low strain on the performance and to keep the bugs to a minimum. Vue has taken all the good attributes of frameworks launched before it. With the same concept, Vue is using the Virtual DOM popularized by React. This ensures a faster and bug-free performance.

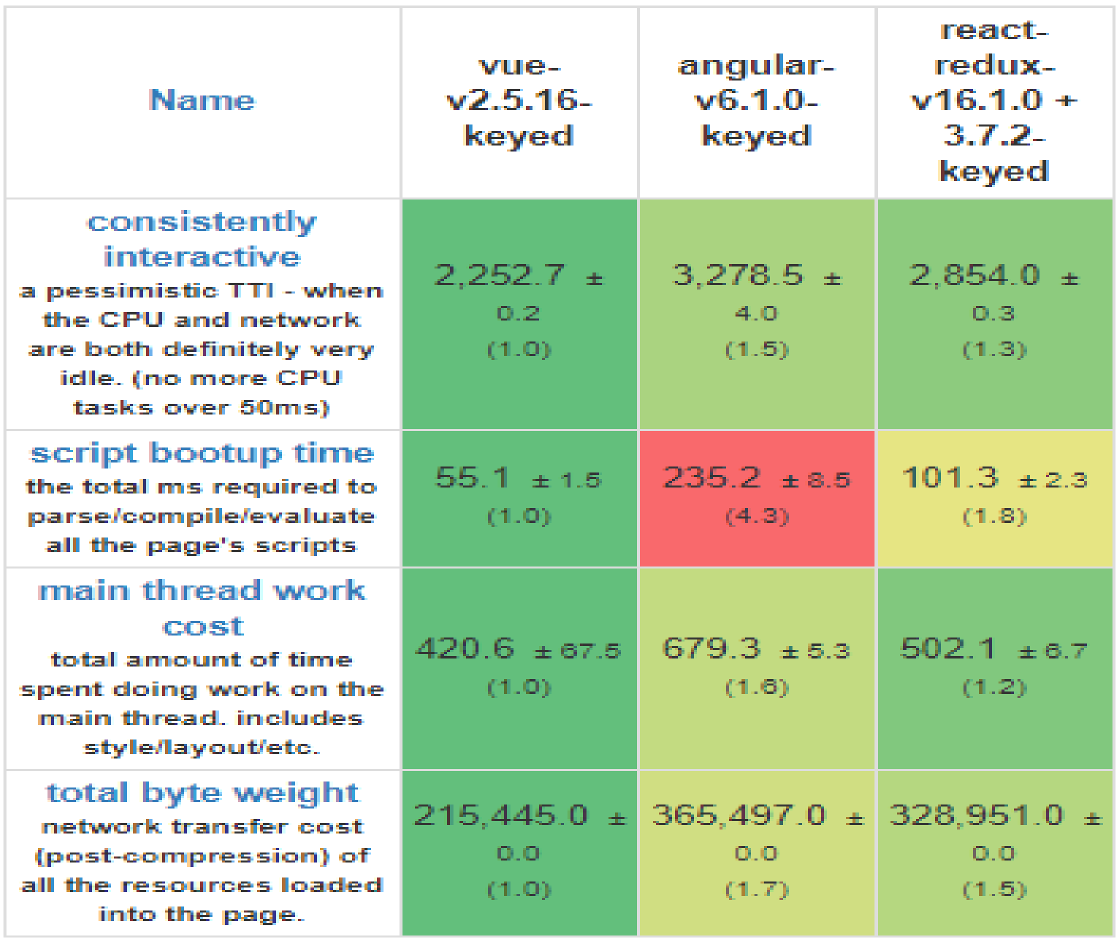
If the user knows what features they will be using in the project, they can choose a framework depending on the small speed differences shown in the graphs below:



Name	vue-v2.5.16-keyed	angular-v6.1.0-keyed	react-redux-v16.1.0 + 3.7.2-keyed
create rows Duration for creating 1000 rows after the page loaded.	182.1 ± 7.6 (1.0)	185.2 ± 10.2 (1.0)	203.8 ± 9.4 (1.1)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	158.8 ± 2.7 (1.0)	161.2 ± 2.7 (1.0)	155.3 ± 1.4 (1.0)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	156.4 ± 9.8 (2.3)	68.8 ± 3.7 (1.0)	97.2 ± 5.2 (1.4)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	10.6 ± 2.0 (1.0)	7.9 ± 4.3 (1.0)	9.5 ± 2.4 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	20.0 ± 2.9 (1.0)	105.8 ± 1.8 (5.3)	105.4 ± 2.7 (5.3)
remove row Duration to remove a row. (with 5 warmup iterations).	54.2 ± 2.2 (1.2)	47.1 ± 3.0 (1.0)	50.5 ± 1.2 (1.1)
create many rows Duration to create 10,000 rows	1,603.2 ± 34.8 (1.0)	1,693.9 ± 70.1 (1.1)	1,996.9 ± 27.8 (1.2)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	342.5 ± 6.0 (1.4)	243.3 ± 6.3 (1.0)	274.1 ± 5.2 (1.1)
clear rows Duration to clear the table filled with 10.000 rows.	191.9 ± 6.1 (1.1)	263.9 ± 3.0 (1.5)	173.9 ± 3.5 (1.0)
slowdown geometric mean	**1.17**	**1.28**	**1.32**

Figure 14 Measurements of DOM manipulation actions [22]

The figure 17 above shows the execution time of different tasks in different frameworks. Most of these frameworks are similar with few differences in certain actions.



Name	vue-v2.5.16-keyed	angular-v6.1.0-keyed	react-redux-v16.1.0 + 3.7.2-keyed
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,252.7 ± 0.2 (1.0)	3,278.5 ± 4.0 (1.5)	2,854.0 ± 0.3 (1.3)
script bootup time the total ms required to parse/compile/evaluate all the page's scripts	55.1 ± 1.5 (1.0)	235.2 ± 8.5 (4.3)	101.3 ± 2.3 (1.8)
main thread work cost total amount of time spent doing work on the main thread. includes style/layout/etc.	420.6 ± 67.5 (1.0)	679.3 ± 5.3 (1.6)	502.1 ± 6.7 (1.2)
total byte weight network transfer cost (post-compression) of all the resources loaded into the page.	215,445.0 ± 0.0 (1.0)	365,497.0 ± 0.0 (1.7)	328,951.0 ± 0.0 (1.5)

Figure 15 Startup Metrics [22]

The following metrics (figure 17) concern the startup of the app, the first loading of the frameworks and how much time it takes for the framework to perform the main tasks.

To summarise a few relevant use cases and their respective recommendations in terms of framework choice, the following collection can be made [24]:

- High knowledge of TypeScript ⇒ Angular
- structure of a project ⇒ Angular
- object-orientated programming (Java; C++; C#) background ⇒ Angular
- flexibility ⇒ React or Vue
- Large scale of applications ⇒ All three
- Easy learning curve ⇒ Vue

- the newest and the most popular technologies ⇒ Vue
- ecosystem/ Community ⇒ React
- Separation between HTML and logic ⇒ Vue
- Strong focus on using JavaScript ⇒ React

8 CONCLUSION

This thesis has tackled the subject of web components frameworks deeply and analysed every aspect of it to help the reader to choose a suitable framework for their next project. In the chapters above, the reader can find a detailed explanation of the 3 major frameworks and arguments for or against the use of each in a project. The matter of deciding which one is better is entirely subjective depending on the project specifications and the developer’s preferences. This means that the conclusion reached by the readers of this paper will be different from one to another.

As mentioned in the previous chapter, every recommendation is subject to the author’s own personal biases and is nothing but a recommendation and is not a rule. Deciding to adopt a new framework always has to be an elaborate process as it determines the success of future projects. However, in the JavaScript world, where new frameworks are published on a regular basis, it may not be smart to wait too long with the adoption of a new technology as it might be outdated by then. Although this, of course, depends largely on the size of a company: Whereas smaller development teams can test and adopt a new framework more quickly, larger companies need more time for the assessment as a wrong decision might result in financial struggles in hindsight. As an advice for the readers trying to choose a framework to start with, have some time testing all three, then you can make a more informed decision.

REFERENCES

1 Learning Web Component Development Discover the potential of web components using PolymerJS, Mozilla Brick, Bosonic, and ReactJS by Sandeep Kumar Patel.

https://www.oreilly.com/library/view/learning-web-component/9781784393649/

2 Developing Web Components By Jarrod Overson & Jason Strimpel.

https://www.adlibris.com/fi/e-kirja/developing-web-components-9781491905692

3 Angular: Up and Running Learning Angular, Step by Step by Shyam Seshadri.

https://www.oreilly.com/library/view/angular-up-and/9781491999820/

4 Introduction to React by Cory Gackenheimer

https://www.apress.com/gp/book/9781484212462

5 Learning React Functional Web Development with React and Redux by Alex Banks and Eve Porcello

https://www.oreilly.com/library/view/learning-react/9781491954614/

6 Information Management: A Proposal Tim Berners-Lee, CERN March 1989, May 1990 - date of retrieval: 00.12.2019

http://cds.cern.ch/record/369245/files/dd-89-001.pdf

7 Network Component Version 7.12.0 MDK Middleware for IPv4 and IPv6 Networking - date of retrieval: 07.12.2019

https://www.keil.com/pack/doc/mw/Network/html/group__ws__web__pages.html

8 CERN, From Wikipedia - date of retrieval: 16.12.2019

http://www.en.wikipedia.org/wiki/CERN

9 World Wide Web, Mcgill university - date of retrieval: 17.12.2019

https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/w/World_Wide_Web.htm

10 MDN web docs, How the Web works - date of retrieval: 22.12.2019

https://developer.mozilla.org/en-US/docs/Learn/Get-
ting_started_with_the_web/How_the_Web_works

11 How HTTP requests work - date of retrieval: 22.12.2019

https://flaviocopes.com/http-request/

12 Web 1.0 to Web 3.0 - Evolution of the Web and its various challenges by Keshab Nath, Sourish Dhar and Subhash Basishtha - date of retrieval: 25.12.2019

https://www.researchgate.net/publication/269310255_Web_10_to_Web_30_-
_Evolution_of_the_Web_and_its_various_challenges

13 Cross Site Scripting (XSS) - date of retrieval: 25.12.2019

https://owasp.org/www-community/attacks/xss/

14 Cross-Site Request Forgery - date of retrieval: 25.12.2019

https://en.wikipedia.org/wiki/Cross-site_request_forgery

15 SQL INJECTIONS - date of retrieval: 25.12.2019

https://owasp.org/www-community/attacks/SQL_Injection

16 What is an SSL Certificate? - date of retrieval: 25.12.2019

https://www.globalsign.com/en/ssl-information-center/what-is-an-ssl-certificate/

17 Getting Started with Web Components: Build modular and reusable components using HTML, CSS and JavaScript by Prateek Jadhwani – date of retrieval: June 2019

https://books.google.fi/books?id=x7aoDwAAQBAJ&lpg=PP1&dq=building%20reusable%20web%20components&pg=PA23#v=onepage&q&f=false

18 Custom Elements - date of retrieval: 25/01/2020

https://html.spec.whatwg.org/multipage/custom-elements.html#dom-attachinternals

19 Shadow Dom by W3C - date of retrieval: 25/01/2020

https://w3c.github.io/webcomponents/spec/shadow/

20 web components specifications - date of retrieval: 25/01/2020

webcomponents.org/specs

21 Reactenlightenment - What is JSX ? - date of retrieval: 26/03/2020

https://www.reactenlightenment.com/react-jsx/5.1.html

22 Introduction to Vue.JS - date of retrieval: 28/03/2020

https://vuejs.org/v2/guide/index.html#What-is-Vue-js

23 Benchmark of different frameworks - date of retrieval: 29/03/2020

https://stefankrause.net/js-frameworks-benchmark8/table.html

24 Comparison of Vue, React and Angular - date of retrieval: 01/04/2020

https://vuejs.org/v2/guide/comparison.html#Size

25 Custom Elements v1: Reusable Web Components - date of retrieval: 02/04/2020

https://developers.google.com/web/fundamentals/web-components/customele-ments

26 Document Object Model From Wikipedia, the free encyclopedia - date of re-trieval: 03/05/2020

https://en.wikipedia.org/wiki/Document_Object_Model

27 Using shadow DOM From MDN web docs – date of retrieval: 10/04/2019

https://developer.mozilla.org/en-US/docs/Web/Web_Components/Us-ing_shadow_DOM

28 Using template From MDN web docs – date of retrieval: 10/04/2019

https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_tem-plates_and_slots

29 Reactivity in Vue.js From VueJS – date of retrieval: 14/04/2019

https://v1.vuejs.org/guide/reactivity.html

30 Document Object Model From Wikipedia, the free encyclopedia - date of re-trieval: 04/03/2020

https://en.wikipedia.org/wiki/Single-page_application

31 Typescript From Wikipedia, the free encyclopedia - date of retrieval: 04/03/2020

https://en.wikipedia.org/wiki/TypeScript

32 Getting started with Angular - date of retrieval: 08/05/2020

https://angular.io/tutorial/toh-pt1

33 Angular Routing - date of retrieval: 07/05/2020

https://angular.io/guide/router

34 React official website- date of retrieval: 04/09/2020

https://reactjs.org/

35 Semantic Web and ontology by Dhana Nandini - date of retrieval: 04/09/2020

https://tcherg.com/assets/images/eBook/1509101694_tology.pdf

36 Between the Wires: An interview with Vue.js creator Evan You – date of retrieval: 05/09/2020

https://www.freecodecamp.org/news/between-the-wires-an-interview-with-vue-js-creator-evan-you-e383cbf57cc4

37 Document Object Model From Wikipedia, the free encyclopedia – date of retrieval: 05/09/2020

https://en.wikipedia.org/wiki/Document_Object_Model

38 Google Home Page encyclopedia – date of retrieval: 05/09/2020

www.google.com

39 Decoupling Angular components from Router From Medium – date of retrieval: 10/10/2020

https://medium.com/@klinkicz/decoupling-angular-components-from-router-f6510ae7eaed

40 A Guide To Debugging Angular Applications Medium – date of retrieval: 10/10/2020

https://medium.com/@vamsivempati/a-guide-to-debugging-angular-applications-5a36bd88b4cf

41 The Good and the Bad of ReactJS and React Native – date of retrieval: 11/10/2020

https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native/