LAB University of Applied Sciences

Implementation of React-Redux in Web Application

LAB UNIVERSITY OF APPLIED SCIENCES LTD
Degree Bachelor of Business Administration
Degree programme in Business Information Technology
Autumn 2020
Le Tran Anh Vu
Tran Nguyen Thuy An

Abstract

Author(s) Le, Vu Tran, An	Type of publication Bachelor’s thesis	Published Autumn 2020
	Number of pages 46	
Title of publication **Implementation of React-redux in web application**		
Name of Degree Degree Business Information Technology		

Abstract

Recently, JavaScript has become a trend programming language. When Node.js was released, a massive amount of libraries and packages were developed. JavaScript has been developed for many environments, such as frontend, backend, and even mobile applications.

This thesis introduces React.js, a user interface library, and discusses its benefits. Moreover, the thesis also describes React.js’s existing challenge which is data management in global. After this the thesis examines one solution for the problem, and this is to apply the use of Redux. Redux is one of the most popular utilities for React.js to manage application data.

By using React.js and Redux in an example application, the authors were able to point out benefits of saving time of development and minimize the effort to maintain code.

Keywords

web development, frontend application, React.js, Node.js, food delivery

CONTENTS

LIST OF FIGURES

LIST OF IMAGES

LIST OF TABLES

LIST OF ABBREVIATIONS

SEO	Search engine optimization
HTTP	Hypertext Transfer Protocol
JS	JavaScript
XML	Extensible Markup Language
UI	User interface
DOM	The Document Object Model
SPA	Single Page Application
CSR	Client-side rendering
CMS	Content management system
NPM	Node Package Manager

1 INTRODUCTION

JavaScript is one of the main programming languages in computer science (Elliott 2014, 2). There is a reason why various software developers have chosen JavaScript for web application. According to Elliott (2014, 2), JavaScript is an event-driven and non-blocking language. Therefore, it has high performance. Moreover, JavaScript's syntax is easy to read and understand. It is quite familiar to developers who have experience with C or C++. In 2009, Node.js known as a JavaScript environment, was released. Node.js is built on Google's V8 JavaScript Engine. It is an important milestone for JavaScript, because many libraries and frameworks were built based on Node.js.

After Node.js was born, many big technology companies started to research and develop utilities, packages, libraries, and JavaScript frameworks. JavaScript can be developed anywhere in development sides. It can be written either in the backend side or in the frontend side or even in mobile devices. Based on that, Google released its frontend framework, named Angular.js, in 2010. After that, Facebook also published its open-sourced named React.js in 2013. After being launched, React.js started a new trend among frontend developers. It helps React.js manage the website can do more functionalities.

React.js is one of the top frontend libraries (Wattenberger 2019a). Many big companies apply React.js such as Facebook, Instagram, Firefox, New York Times, PayPal, Netflix, and Khan Academy (Grov 2015, 5). However, React.js still contains some problems in data management. To solve those problems, many React.js's utilities and packages have been released. One of those is Redux. It helps React.js manages application data more robustly and predictably.

This thesis studies React.js and Redux. The thesis explores why developers choose React.js and how it can save time in project development and maintenance. In addition, the thesis aims to find out the benefits of using Redux.

1.1 Thesis objective

This thesis aims to explore the advantages of using a modern frontend library, Redux-React.js that can be used to create a user-friendly web applications that offer a good customer experience. Moreover, the thesis examines how React.js can help reduce the effort needed to develop and maintain applications.

1.2 Research questions

Based on the objective, the thesis has three research questions. The questions help to examine the benefits of using React.js and Redux in developing web applications.

- Why choose React?
- Why does React.js save time in application development and maintenance?
- What are the benefits of using Redux?

To answer the research questions, the authors will collect the available master's theses, books, and official technical documentations. Lastly, the thesis is based on the practical findings of an application to conclude.

1.3 Limitations

This thesis does not cover search engine optimization (SEO) for React.js applications. Testing is another step of developing web applications. But this topic is not discussed in this thesis.

1.4 Thesis structure

The thesis includes four chapters. Chapter 1 is the introduction, which is an overview of the thesis. This chapter is included the thesis objective, research questions, limitations, and thesis structure. Chapter 2 presents the theoretical framework of the thesis. This chapter introduces the technical background and their terminologies used in the thesis. Chapter 3 introduces the example application, which is used to answer the research's questions. In this chapter, the application's features are introduced and analyzed to represent the using technical frameworks or libraries in chapter 2. The last chapter describes the findings and the conclusion. The Firgure 1 shows the thesis's structure.

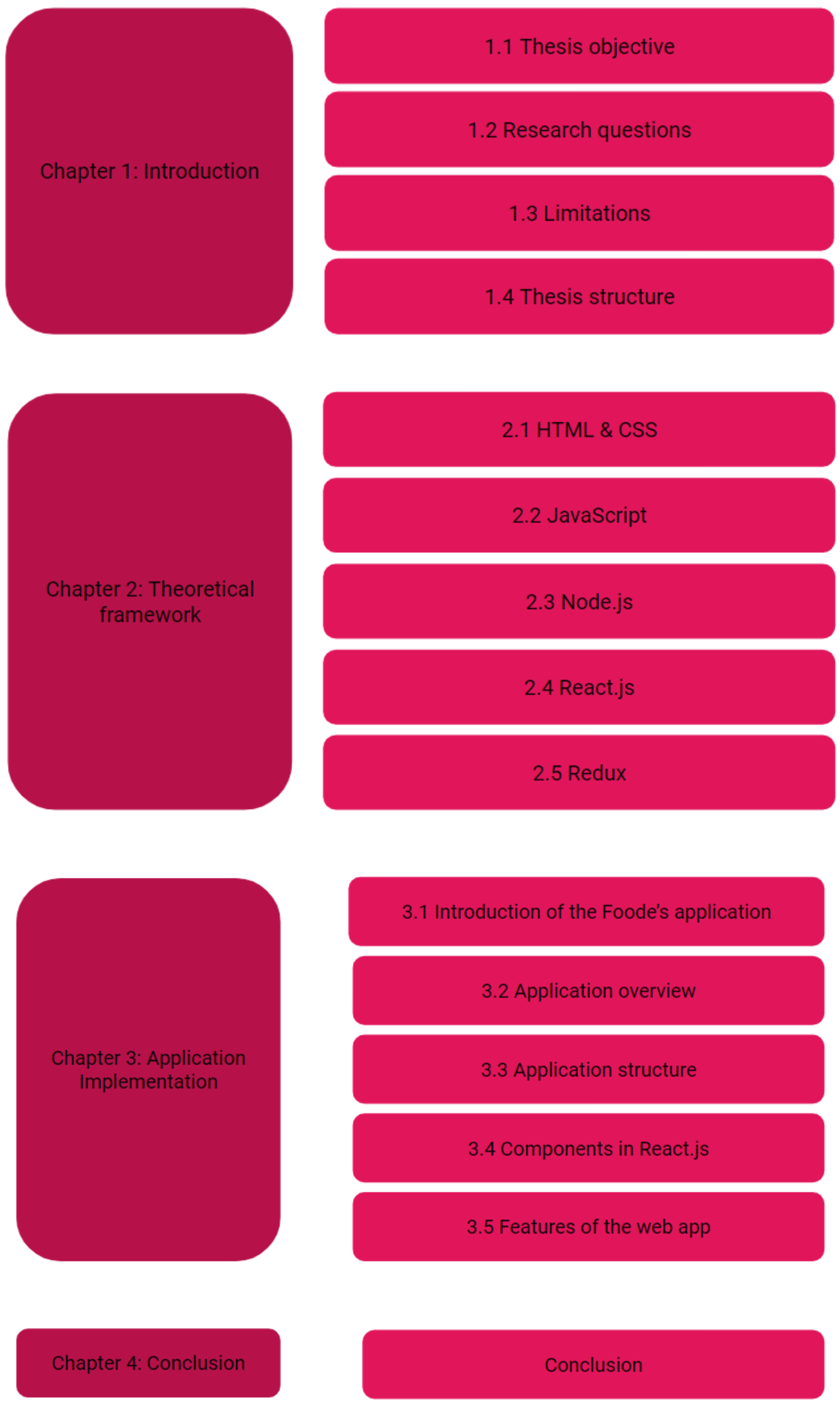


Figure 1. Thesis structure

2 THEORETICAL FRAMEWORK

In this chapter, the authors describe the theoretical concepts of technical terms. Those terms are requisites for building the web application. The authors answer the question: “Why choose React?” by defining React.js.

2.1 HTML & CSS

According to Mozilla and individual contributors (2020a), HTML (the Hypertext Mark-up Language) and CSS (Cascading Style Sheets) are two leading technologies for creating Web pages. HTML plays the role of describing the structuring of the page using mark-up. HTML elements create the building blocks of HTML pages. For example, image and other objects such as interactive forms. HTML provides structured documents such as headings, paragraphs, lists, links, quotes, and other items. CSS is a stylesheet language used to describe the presentation of HTML or XML documents. It explains how HTML elements should be displayed on the screen, such as colours, fonts, speech, media. CSS supports different devices that adapt the presentation, such as computer screens, laptop screens, tablet screens, or mobile screens. Put simply, HTML constructs a webpage's skeleton, whereas CSS is used for decorating the webpage.

2.2 JavaScript

JavaScript, called Live Script, was created by Brendan Eich in 1995 during his time at Netscape. The first version of the language was available in Netscape Navigator 2. Live Script was renamed as JavaScript three months later. JavaScript was first submitted to ECMA International in November 1996. The standards process continued for a few years, with ECMAScript 2 in June 1998, and ECMAScript 3 in December 1999. As of 2012, all modern browsers have fully supported ECMAScript 5.1, and on July 17, 2015, ECMA International published the sixth major version of ECMAScript, known as ECMAScript 2015. The latest version is ECMAScript 2020 or ES9. (Mozilla and individual contributors 2020b.).

JavaScript is a lightweight, interpreted, and compiled programming language with first-class functions. It is a well-known scripting language for websites and applied in non-browser environments such as Node.js or Apache Acrobat. JavaScript is a prototype-based, multi-paradigm scripting language dynamic that supports object-oriented, imperative, and functional programming styles (Mozilla and individual contributors 2020b).

JavaScript usually runs on the client’s browser like Chrome, Firefox, or Edge. JavaScript language is easy to learn and widely used for controlling web page events. The number of professional developers using this language is shown in Figure 2 provided by Developer Survey 2020, with 47,184 responses.

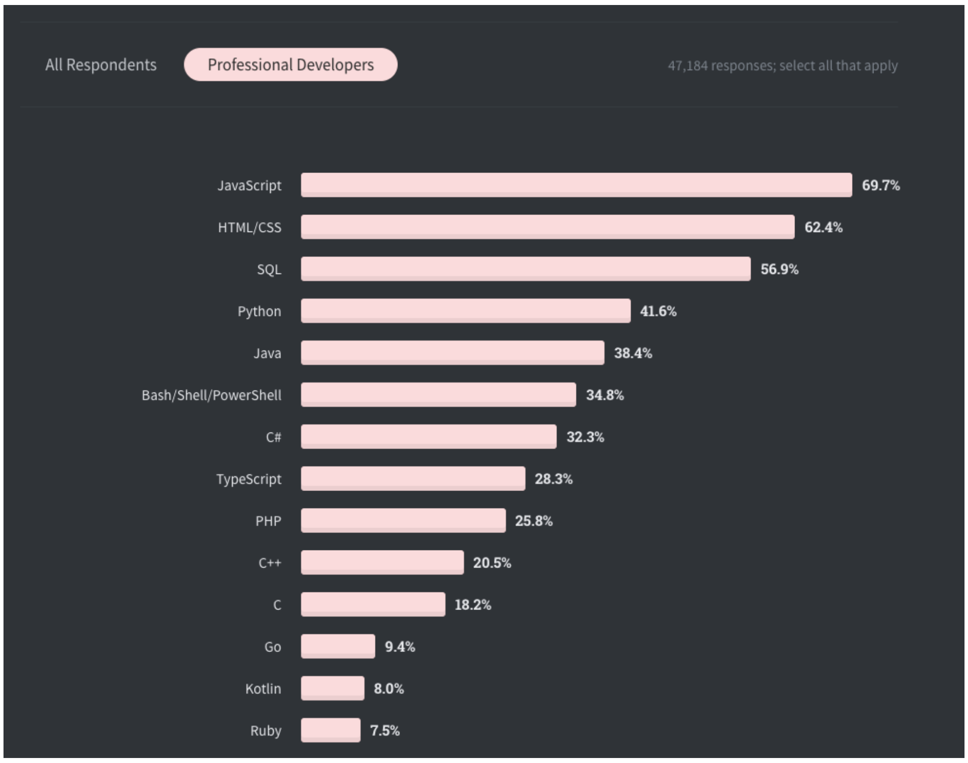


Figure 2. JavaScript’s ranking among professional developers (Developer Survey 2020)

Because of the large number of professional developers who use JavaScript, tools, libraries, and frameworks have been released (Developer Survey 2020).

2.3 Node.js

According to OpenJS foundation (2020), "Node.js is a cross-platform JavaScript runtime built on Chrome's V8 JavaScript engine". Node.js allows JavaScript to jump out of the browser's environment and run anywhere (Satheesh, D' Mello & Krol 2015, 2). Node.js becomes one of the most choosing technologies for designing small modules, also known as packages (Casciaro 2020, 69). A Node.js app runs in a single process, without creating a new thread for every request (OpenJS Foundation 2020). Node.js is the model doing the set of asynchronous I/O operations preventing the code blocking of JavaScript. When Node.js is operating in I/O, it will read from the network instead of blocking the thread and repeat the cycles waiting. Node.js is accessing the database of the file. Then, Node.js continued replies and calls back the operations without starting the new thread for other operations. As a result, it can process thousands of events by merely running one server. In addition, Node.js is picked by millions of frontend developers who use JavaScript for web applications. (OpenJS Foundation 2020). Many Node.js frameworks are used on website. There are, for example, Express.js, Socket.io, Strapi, Next.js, model toolkits like Angular, React.js.

2.4 Node Package Manager (NPM)

According to Teixeira (2012, 8), the Node Package Manager (NPM) is a third-party package repository. NPM provision the public registry where a developer can publish their package freely for everyone to use. Thereon, NPM also provides a command-line tool for developers to download third-party open-source packages and apply them instantly. Therefore, it helps Node.js and JavaScript libraries widely known and used by many developers worldwide. Node or JavaScript developer could share their package everywhere. It could save time to solve the same problem by downloading the existing box. Developers install, uninstall, up-load, and update the package in NPM by command-lines interface (CLI) (Teixeira 2012, 9). NPM's CLI must be installed in the host machine before using it.

To install the packages, type this command to the terminal (Teixeira 2012, 10)

```
npm install <package's name>
```

To uninstall the packages, the command is (Teixeira 2012, 11)

```
npm uninstall <package's name>
```

And to update the packages, developers can type (Teixeira 2012, 11)

```
npm update <package's name>
```

2.5 React.js

React.js was initially released in 2013. Jordan Walke, Facebook's developer, created it. By 2020, the React.js has become one of the most popular JavaScript libraries, and it continued to developed and maintained by Facebook.

React.js is a JavaScript user interface library. React.js was created to solve the problems of working on complex user interface projects such as web application, E-commerce, CMS, and Mobile application. Moreover, React.js provides an option client-side rendering (CSR), which helps the application render the layout just on the browser. As a result, the browser can share the workload of the server. Moreover, CSR can increase user experience. Users do not see the white-blank page when they navigate to another page because the rendering content is now on the browser.

2.5.1 Terminologies

Lightweight DOM (Document Object Model) For Better Performance

When using JavaScript the traditional way, developers need to look at what data changed, and imperative events make a change to the DOM, then keep it up to date manually. After that, re-render happens on every page triggering an event. React.js provides a more optimized and lightweight document object model. It interacts with the virtual DOM, stored in the memory, instead of connecting directly with the DOM. The image below explains clearer how React.js works. It calculates the difference between the old and the new virtual DOM after changing happened. The components with event changed are updated to the actual DOM. As a result, the site re-render is fast. (Aggarwal 2018, 2.)

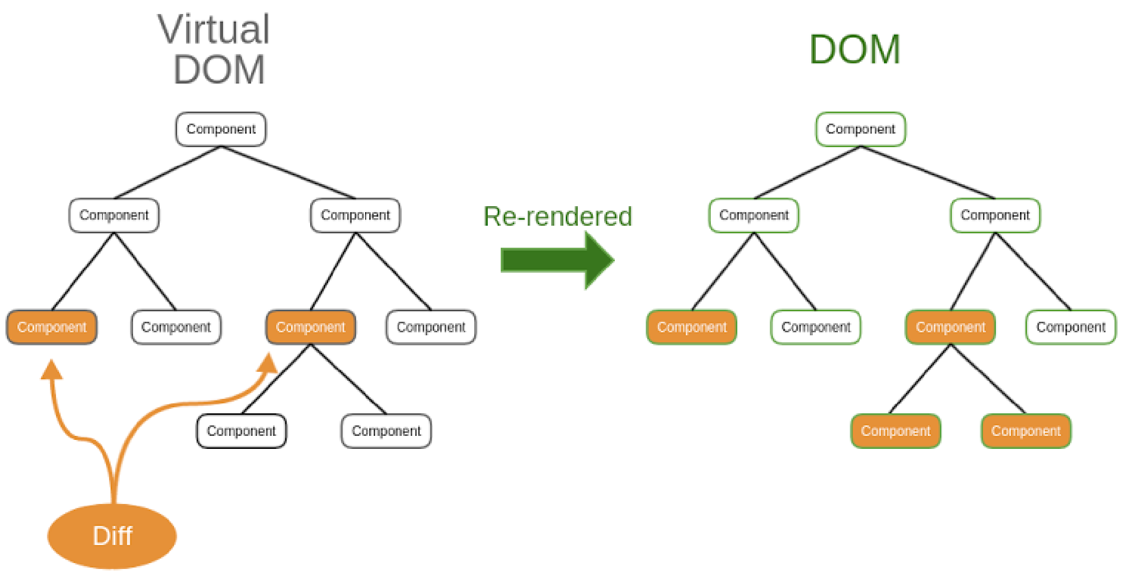


Figure 3. Virtual DOM and real DOM (Naukri Engineering 2017)

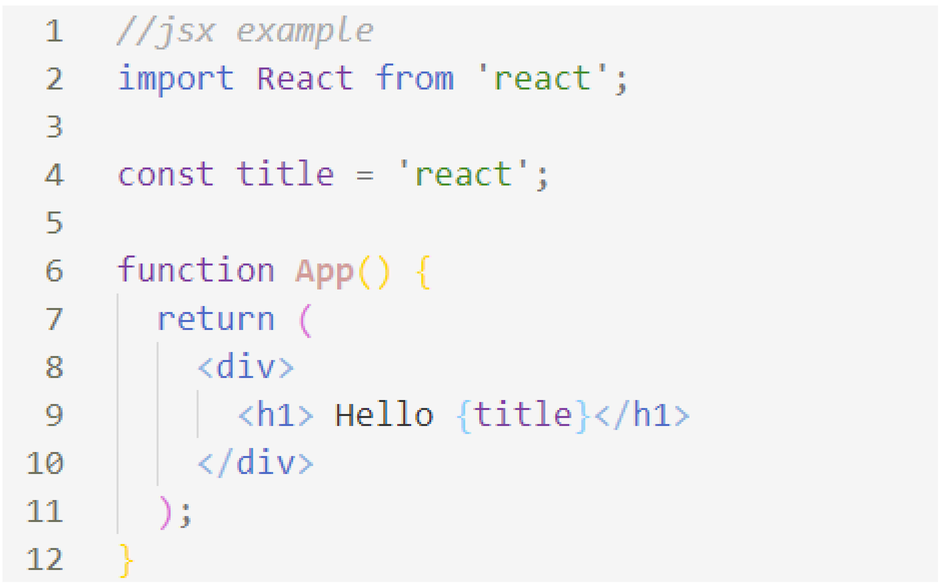
2.5.2 Virtual DOM

React.js is fast and optimized because it has a virtual DOM. Virtual DOM is used to check the DOM blocks' change before allowing the browser to repaint the user interface. It is very suitable for applications, which frequently require changing user interface's logic (Facebook Open Source 2020a).

React.js is a user interface library to address the problems of large-scale applications and data sets that change over time (Gackenheimer 2015, 3). When users interact with a website, the state of user interface changed. The virtual DOM compares the updated version with the previous one to find the different nodes. Then, the virtual DOM sends those nodes for the DOM. The DOM would change based on what is updated. Instead of loading the whole page, the application only renders specified parts (Aggarwal 2018, 2).

2.5.3 Introducing JSX

JSX is a syntax extension of JavaScript for function calls and object construction. The JSX syntax looks like XML/HTML but it is not. JSX helps developers in a great way to write React.js components. JSX combines syntax between HTML and JavaScript, allowing developers to write the HTML code and embed JavaScript expression inside (Facebook Open Source 2020b).



```
 1  //jsx example
 2  import React from 'react';
 3
 4  const title = 'react';
 5
 6  function App() {
 7    return (
 8      <div>
 9        <h1> Hello {title}</h1>
10      </div>
11    );
12  }
```

Image 1. JSX syntax

In Image 1 above, from lines 27 to 29, JSX provisions the HTML-JavaScript mixture. So that the title is a variable and is embedded to "<h1>" tag. And the result shown on the screen is "hello react."

There are many benefits of using JSX (Mardan 2017, 42):

JSX increases developer experience because of the XML-like syntax that helps to represent nested declarative structures.

Because it reminds an HTML template, JSX improves the productivity. Using it saves time.

React.js does not require JSX syntax. Developers can choose plain JavaScript for creating UI's components (which is introduced later). However, React.js encourages developers to use JSX to experience all the best features and convenience of React.js's developers.

2.5.4 States and properties

According to Mardan (2017, 80), the main difference between states and properties is that states are mutable, but properties are immutable. Moreover, properties have been passed from their parent component, and it could been become different values by the parents. In the contract, states need to be defined that have been changed in the components itself. Therefore, properties are used to determine the interface upon creation and remain the static, in the contrast, states are set and update by the object. Components access states and properties as component attribute, but both are used for different purposes (see Figure 4).

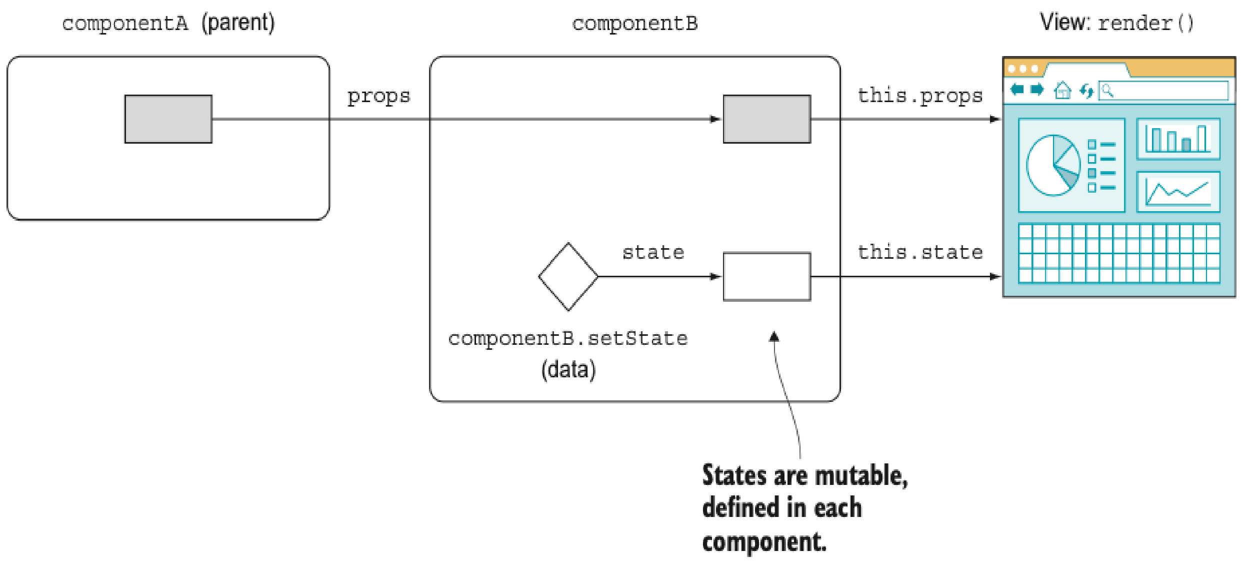


Figure 4. Communication between components use properties and states (Mardan 2017, 81)

2.5.5 React.js component-based architecture

One of the advantages of React.js is component-based architecture. A component is vital for scaling an app in the future and reduce the workload of maintenance.

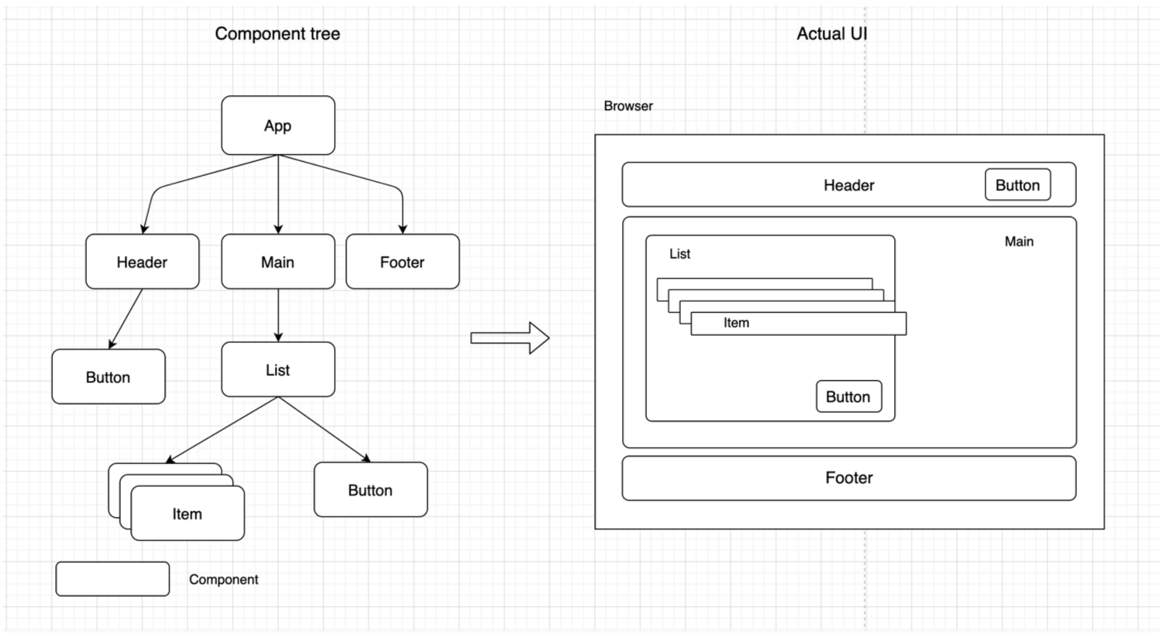
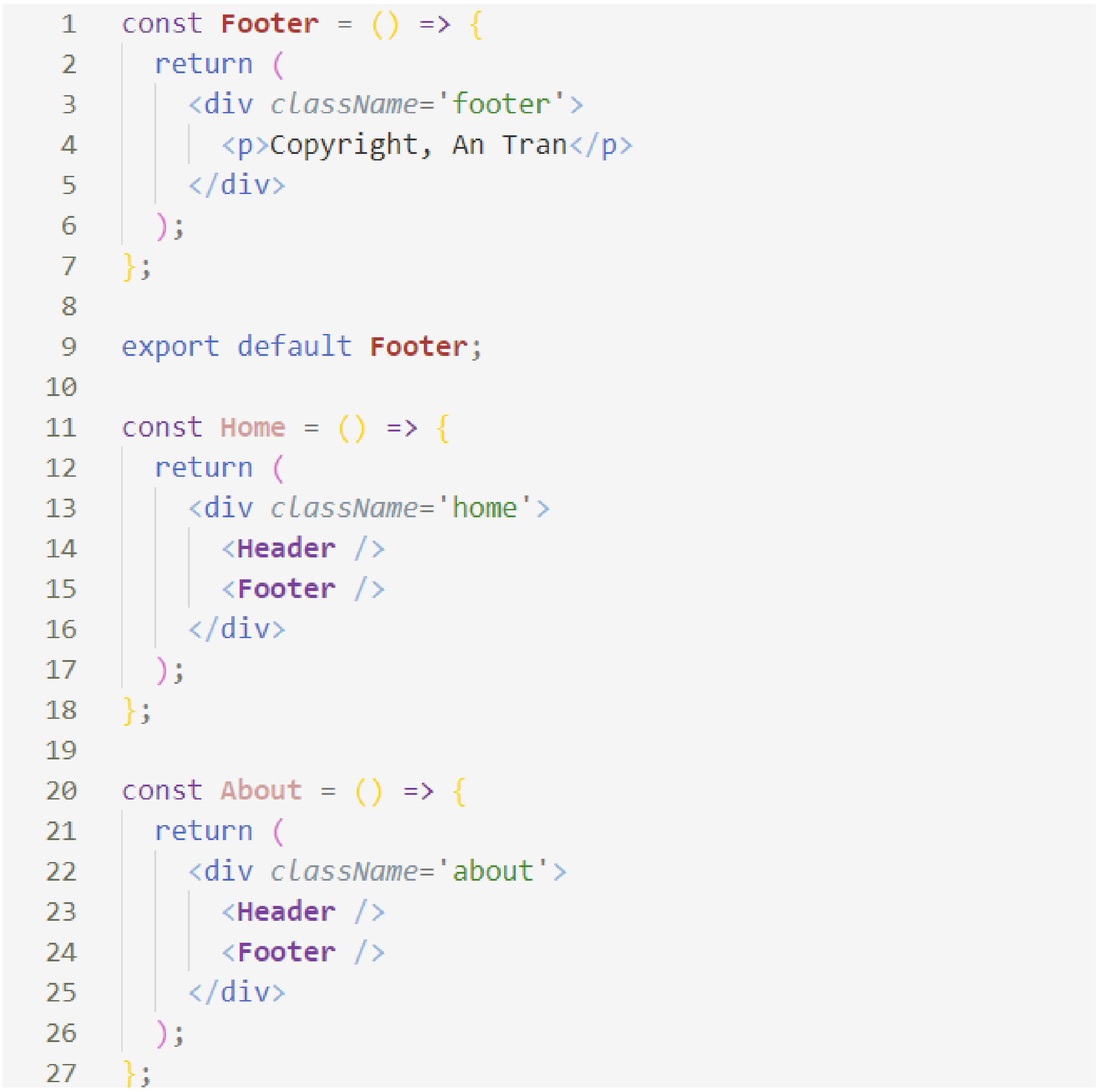


Figure 5. React.js components and pages

To quickly see benefits of React.js, the authors demonstrate a simple static website. It contains Home, About, and Contact pages. Each page always has the following parts: Header, Main, Footer. The common parts of a website are Header and Footer (Image 2). Without React.js, developers need to add the Header and Footer snippets for every page. This is cumbersome and redundant. With React, developers could reduce the boilerplate by creating separate components for the Header and Footer. These components can be used in many places by simply calling the components. Additionally, developers just need to change the user interfaces of Header and Footer in their components, and they will apply to all pages. That is the advantage of React.js components. (Mardan 2017, 10.).

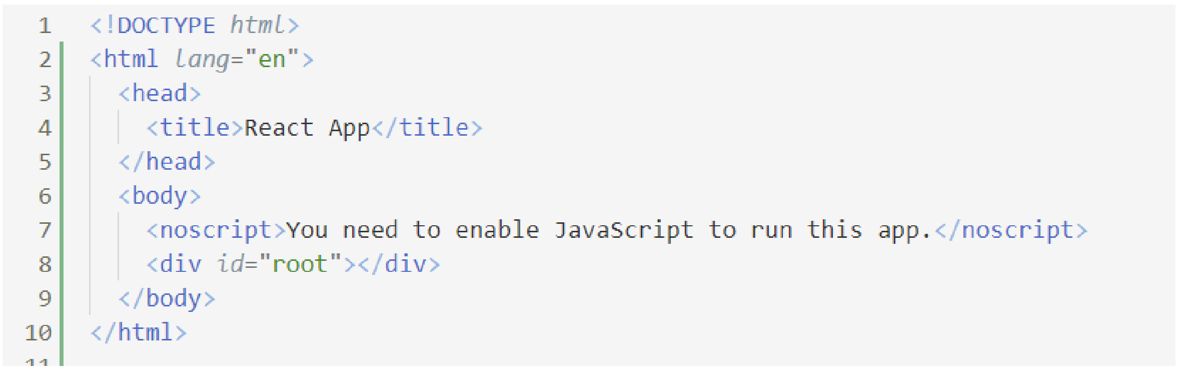


```
const Footer = () => {
  return (
    <div className='footer'>
      <p>Copyright, An Tran</p>
    </div>
  );
};

export default Footer;

const Home = () => {
  return (
    <div className='home'>
      <Header />
      <Footer />
    </div>
  );
};

const About = () => {
  return (
    <div className='about'>
      <Header />
      <Footer />
    </div>
  );
};
```

Image 2. How to apply React.js components to web pages

2.5.6 How React.js works

Initially, in rendering React.js combines the application components and transfers them to virtual DOM. Then, it renders the virtual DOM to real DOM and lets the browsers do their job to render and paint the element to user interfaces. Technically, React.js transfers the JSX syntax from components into a plain JavaScript file. Then the JavaScript file is attached to an HTML file (see the Image 3).



```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Image 3. Initial HTML file used in React

When the application has an updating, React.js looks up the changed components in virtual DOM. It decides the most optimized way to update those components to the real DOM.

2.5.7 Comparison with other frontend frameworks

This thesis compares three well-known frameworks: React, Vue, and Angular. According to Wattenberger (2019b), 21,717 respondents covered five aspects of the experiences or knew the JavaScript frameworks (see the Figure 6). All respondents who are developers know all those frameworks. 71.7% of developers stated that React.js is the most favoured choice, and they are willing to continue working with React.js in the future. Meanwhile, Vue and Angular got 40.5% and 21.9%, respectively. Angular is voted that developers would not work with it again, by 35.8%. Around 34.2 % of developers desire to study Vue.js, 12% want to learn React.js, and 9.7% want to learn Angular.

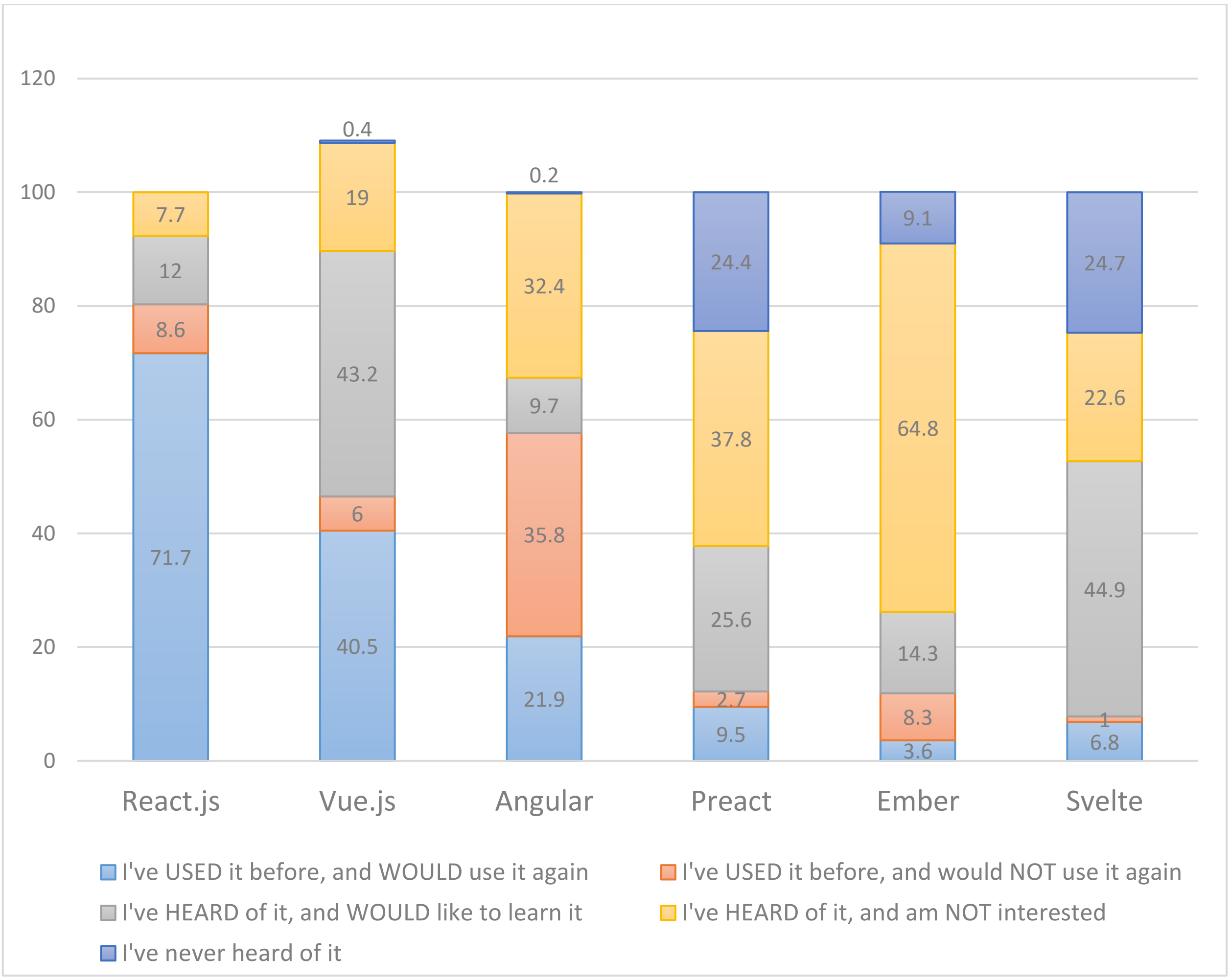


Figure 6. Comparison between React, Vue.js and Angular (Wattenberger 2019b)

Statistics:

The authors have collected statistics in Table 1 above from React, Vue.js, and Angular's GitHub and the node packager managers from npmjs.com on the 6th of September 2020. React.js is downloaded approximately four times more often than Angular and Vue. Besides, React.js's has over 1500 contributors and 453 issues. All those figures in the table show that React.js is a maturity frontend library supported by a broad community.

Table 1. Compare between React, Angular and Vue.js (GitHub 2020a; GitHub 2020b; GitHub 2020c; NPM 2020a; NPM 2020b; NPM 2020c)

Attributes	**React**	**Angular**	**Vue**
Initial release	May 2013	Sep 2016	Feb 2014
npm download (weekly)	7,996,769	1,838,683	1,899,960
Star	Over 155,000	Over 65,600	Over 171,000
Issues	453	Over 2700	329
Contributors	1,500	1,212	371

Learning curve:

Vue is the brightest one in learning curve because Vue combines both the robust strength of Angular and React.js into it. Vue syntax is easy to use.

React.js is the second library that is easy to learn. The official document of React.js is useful and easy to use. React.js only focuses on V in MVC, so there are fewer terminologies than in Angular.

The developer must understand the MVC model and other terminologies such as dependency injection, module, services, and controllers. Angular is strict its file structure so that developers have a lack of option for structuring their projects.

Summary:

React.js is the most popular and mature frontend library currently. Vue is getting to develop rapidly, Angular, on the other hand, is a framework. It includes everything for developers, while React.js and Vue require developers to download more packages for implementation. Angular is like the full options of the frontend framework and possesses a well-designed architecture and file structure, developers do not need to download packages.

Answering which is the best choice for frontend applications is hard. React.js and Vue are the best fit for opinionated applications where developers can freely pick to pick any utilities and tools to develop. In contrast, Angular is the best for a well-defined application developed by an experienced team.

2.6 Redux

The more complex the application becomes, the more components are created. Data management is becoming a problem that developers need to solve. There are some scenarios where data can be shared between many components in the application. This issue makes the data harder to manipulate and update the application.

Redux was created to solve this problem. It is one of the hottest libraries in frontend development currently (Wattenberger, 2019a). The open-source was created by Dan Abramov and Andrew Clark in June 2015. According to Abramov and the Redux documentation authors (2020a), Redux helps JavaScript applications manage the data-flow architecture more efficiently. In other words, it maintains the whole application's state. It does this with just a single immutable state object. Redux is usually used with libraries such as React.js, Angular. It is becoming a popular library, thanks to its simplicity and it being lightweight. There are many benefits to use Redux. Firstly, the Store is the place to save the current state of the application. The Store makes it easier to predict the data outcome. Secondly, with a predictable data outcome and a strict structure, it reduces the developer's effort in maintaining the code. Thirdly, combining the developer tools allows developers to track actions and state changes in real-time. Finally, having tremendous community support is always a big plus for every developer.

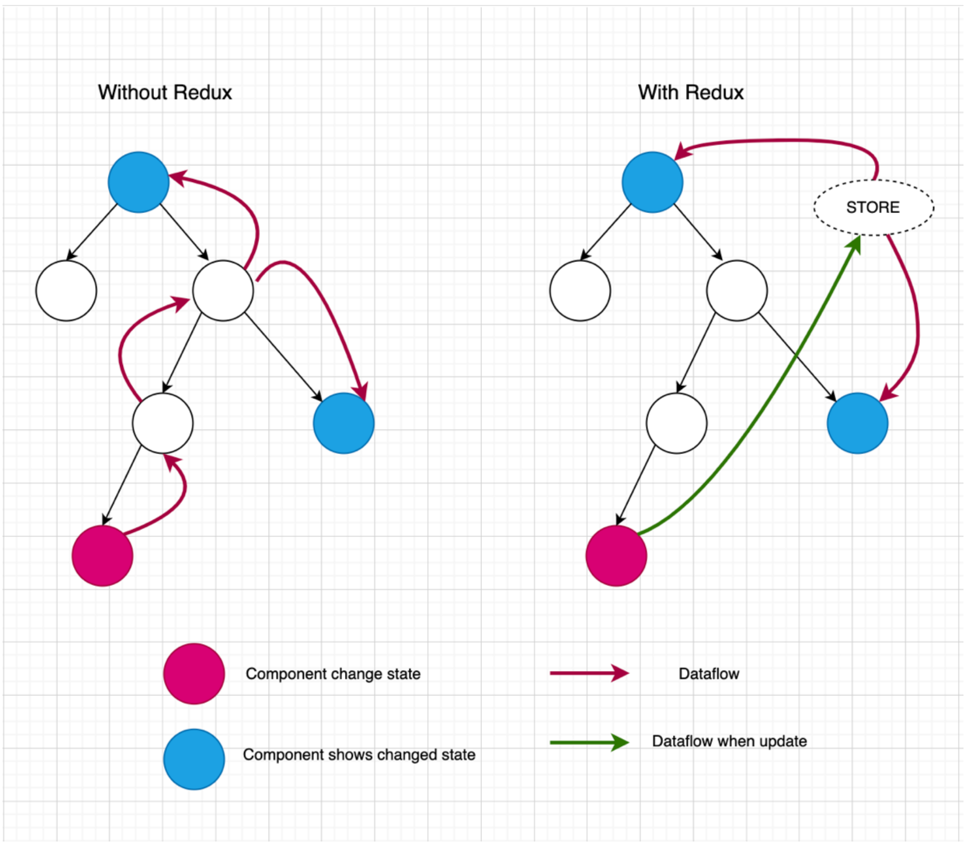


Figure 7. State management with and without Redux

In the figure above, without Redux, data must go a long way to present in other components. With Redux, if there is a mutation from data, this data is sent out to the store. Therefore, the subscribing components are notified to show changes in data on the screen. This Redux mechanism is a single source of truth.

2.6.1 Redux libraries

According to Abramov and the Redux documentation authors (2020a; 2020b), Redux is a standalone JavaScript library. It is used for many purposes or frameworks. Redux's best well-known library is React-Redux, which allows developers to manage state in React.js by using the Redux mechanism. However, Redux does not support React.js but for other frameworks like jQuery, Ember, Angular, Vue, or even Vanilla JavaScript (pure JavaScript).

There are two other popular libraries of Redux, which are the Redux-toolkit and the Redux Devtool Chrome extension. In this thesis, the authors only focus on the most famous libraries, that is React-Redux. The authors may call React-Redux as Redux for more convenience.

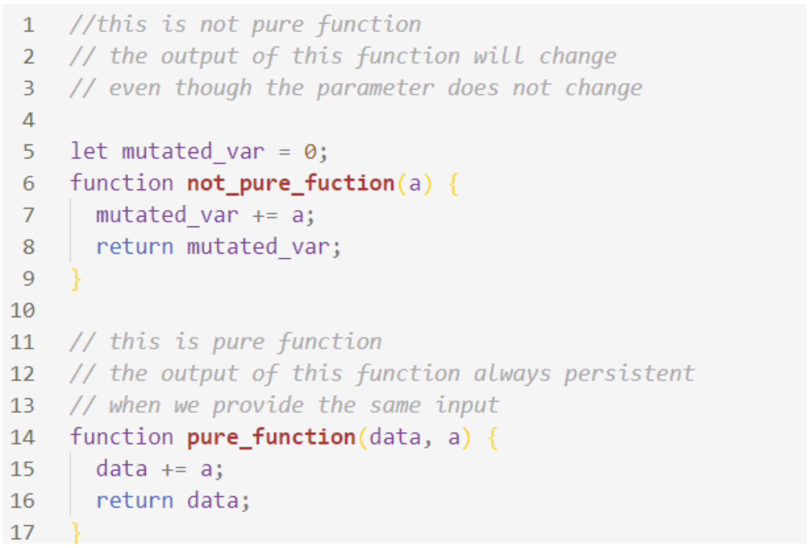
2.6.2 Principles

Redux introduces three principles in their product (Abramov and individual contributors. 2020c).

The first principle is “single source of truth”. Redux attempts to store all data in one place and just one place. Any manipulating actions only access to one place.

The second principle is “the state is read-only”. Because all data is stored in a unique place, it would be easy to mutate data by mistake. In order to prevent that, the Redux set all data or state are read-only so that there is no method or strict about mutating data directly.

And the last principle is “changes are made with pure functions”. There is only one way to update data, which uses pure functions. The pure function is the function that can predict the return value. Pure functions always return the same output every time it receives the same parameters. Pure function prevents side effects, which means it only affects the variables, which are located within pure functions. The image 4 demonstrates how pure function work. The reason why Redux requires pure functions to mutate data because it prevents the data is copied shallowly. Consequently, the application can have unexpected bugs in the future.



```
//this is not pure function
// the output of this function will change
// even though the parameter does not change

let mutated_var = 0;
function not_pure_fuction(a) {
  mutated_var += a;
  return mutated_var;
}

// this is pure function
// the output of this function always persistent
// when we provide the same input
function pure_function(data, a) {
  data += a;
  return data;
}
```

Image 4. The example of pure and not pure function

2.6.3 Basic concept

Redux keeps the idea of one-way data binding of React, which means that every dataflow always goes in one direction (Abramov and individual contributors. 2020b).

One-way dataflow:

A simple example of "one-way dataflow" is presented below (Abramov and individual contributors. 2020b):

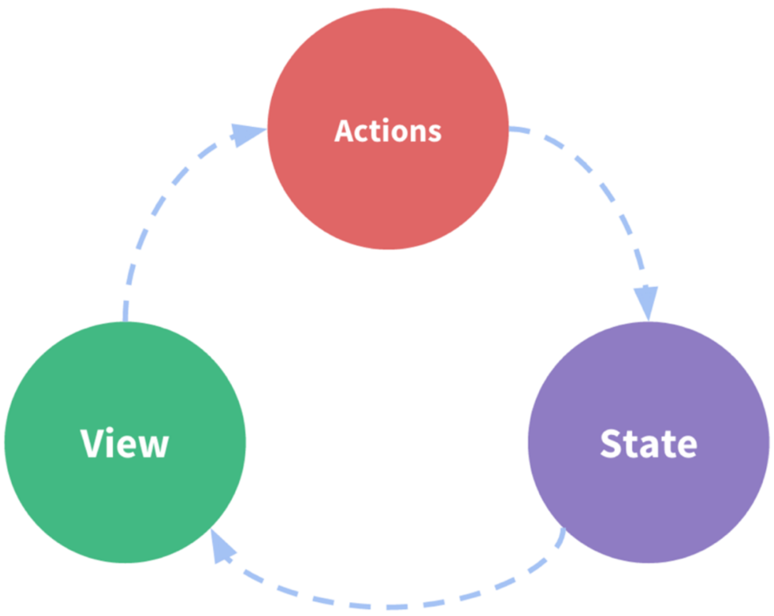


Figure 8. One-way dataflow (Abramov and individual contributors. 2020b)

In the Figure 8, the following concepts explain how One-way dataflow works (Abramov and individual contributors. 2020b). The state presents the data value or application context at one of the moments. It takes responsibility for controlling data, deciding how data is updated, and sending the newest amount for view. The view displays the data. It subscribes to the state for getting newly updated data. It also the terminal for the user can interact with the application and emit some change for data. The actions are noticed when some events are emitted from view. It plays the role of "data courier," dispatch the requests, occurrences, or changed data to the state. And this process keeps running repeatedly.

Terminology:

For Redux, the process is a bit complicate for manipulating the data in more context and scenarios.

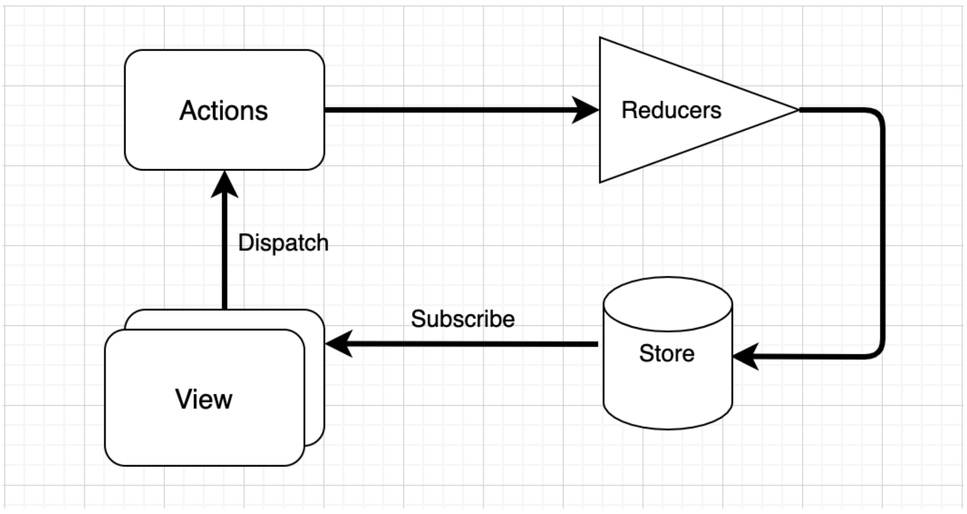


Figure 9. How Redux works

The view is the UI of the web application. It shows the layout and page content such as buttons, fields, texts. It is where users can interact with the application like request data and submit the form. In Redux, views Subscribe to the data from Store for always receiving the newest and reliable data. Besides, it listens to the behaviours of the user. Thus, whenever interact and take Action with the browser, the View is Dispatching to Reducers.

The action is literally the plain JavaScript object is used to describe the event that happens in the application. The action usually contains two-part: type and payload. Type is the key, which is used by Reducer for deciding which data should be updated. In contrast, the payload is the information of the action (it could be updated data or empty). In the real scenario, Redux uses another term named Action Creators. Action Creators, as it is named, is a function that returns an Action. Action creators usually use in an application. It helps the code more secure and neat due to calling a function that would be easier than writing the object everywhere. (Abramov and individual contributors. 2020b.)

The reducer is a pure function. It receives the current state, which is stored in the Store, and action dispatch. Reducer contains logic for deciding the way to update data. However, to recognize which data should be updated, the reducer must categorize the Type Of Action (that is why action contains two parts the payload and the type). (Abramov and individual contributors. 2020b.)

The store is the data storage in Redux. The Store is just the global plain JavaScript object. It is just the term in Redux, and data is saved in memory so that when the application is refreshed, the Store is empty. However, when using Redux in single-page applications, the reload action is not usually happening; therefore, Store can as storage. According to the image above, the store receives the return of new value from Reducers, saves, and sends newly updated data to Views, which subscribe to new data. (Abramov and individual contributors. 2020b.)

The dispatch function is used when Views would like to send new Action (event) to the Reducer (Abramov and individual contributors. 2020b.)

How Redux works:

As introduced above about ”one-way dataflow,” Redux’s dataflow could be known as the “next level” of such dataflow. Redux separates two main steps of its dataflow (Abramov and individual contributors. 2020b).

In the initial state, the store invokes the root reducer for the default or initial state (data). Therefore, all defined reducers are called once when the application runs for the first time. When the views are rendered, they access the store and receive the initial state from that.

In the update state, it divides into two main scenarios. First, when users create some events, the views dispatch the actions to content type and payload to the reducer. In there, the reducer is like the manufacture, receives the current state and new action, and returns the newly updated state for the store. Second, when the store receives a new state, it notices all the subscribed views in the application. Those such views get a new state and present new information to the application. Finally, those steps above keep running until the application close.

Why use React-Redux:

React-Redux is an official library from Redux, which could be used in React.js application, as mentioned above in the Redux libraries part. React.js is great for web applications. However, it still needs some other libraries to go along for developing a better application. React-Redux is one of the best useful libraries to React. It would help React.js for the three main purposes.

According to React.js Open Source (2020a), React.js was known as the unopinionated library. That means developers can do everything they like, such as coding style or file structures, and added other libraries to develop the application. It is excellent and fast for the initial state of a project or small project. However, If a team with more than two people take part in the project, it would be terrible as everyone has their style and libraries. The project becomes hard to maintain. According to Abramov and the Redux documentation authors (2020d), Redux encourages the application to follow its best practice coding style. Besides, Redux’s documentation is useful and well-known. Many people follow the same coding style and understand the Redux concept. Therefore, the application would be easier for maintaining.

As mentioned above in the concept part, Redux would save React.js from nightmare state management in the management state. From now, if React.js has any global state or data, it can delegate this state for Redux. Therefore, the state is well managed and predictable.

In debugging, Redux breaks the logic into small blocks such as action creators, dispatch, reducers, and stores. That would make the developer can keep track of the dataflow easily with Redux DevTools.

When should and should not Redux be used:

According to Abramov and the Redux documentation authors (2020b), developers should use Redux when the application requires one or more global state. And this state could be used in many application components such as users' login status and picked item list in e-commerce apps (Abramov and individual contributors. 2020b). Moreover, Redux could be applied for a medium, or big application requires more than one developer contributes to the application (Abramov and individual contributors. 2020b).

On the other hand, Redux is not the golden key to everything. React.js itself is excellent. Developers could build React.js without Redux. Redux was born for supporting React. In all the cases, developers can remove Redux from the tech list and use React. According to Abramov and the Redux documentation authors (2020e), developers do not need to use Redux unless they need it. Besides, in a small application, state or data is present only one of place or component. The component's structure is not too complicated. Thus, using Redux is like use the saw for cutting a chicken.

Trade-off:

Redux is great. However, It also contains some trade-off that developers have to concern to apply.

First, that is the learning curve. Developers must take quite a bit of time to understand the Redux workflow and its terminologies. In the beginning, Redux is not easy to reach.

Last, Redux setup and workflow are quite complex and are not built for quick coding (Abramov and individual contributors. 2020e). Developers must define reducer, store, actions every time they add a new state in Redux. When developers want to remove the state, they do not only remove a variable in a global object in JavaScript but also remove all reducers and actions appropriately.

3 APPLICATION IMPLEMENTATION

This chapter introduces the application which is used to examine the benefits of React.js and Redux. It also shows the usage of React.js component, how it is implemented and reduced a needed effort to develop and maintain applications. Some React.js packages are introduced, such as React Bootstrap, BlurHash, and Redux.

3.1 Introduction of the Foode's application

The Foode is a food ordering application developed with React.js and Redux. Users can register via their information, such as usernames and passwords. Users can also review the restaurant's list of meals, and they can also select which meal and its quantity on the restaurant detail page. All the user's orders are stored and presented in the application's header to check out quickly. Eventually, users must log in (if they did not) and fill in some necessary information to create their order. An order can have multiple meals, which various portions.

The application was built for introducing React.js features. Besides, the authors also use other packages for supporting React. The purpose of this application is to show the feasibility of React.js and Redux.

3.2 Application overview

This section introduces the application's pages and mentions its main features. It helps the readers to get an overall picture of this application. As a result, the readers could easily understand the later chapters.

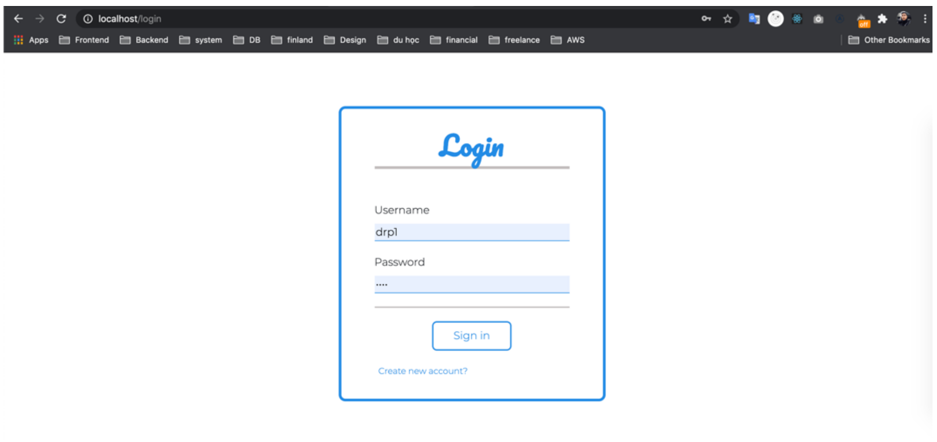


Image 5. Login page

Image 5 shows the login page of the application. Users could log in with their username and password. Besides, they could also register by clicking on the link “create a new account?”. The user interface of the register page is similar to the login page.

After logging in successfully, the home page appears and shows the list of meals in restaurants. The users can use the search bar to narrow the number of meals. This page does not require a login. The users can navigate straight to the home page as a guest.

The header of the page is consistent between pages. It shows the login status of the users and the information on their shopping cart.

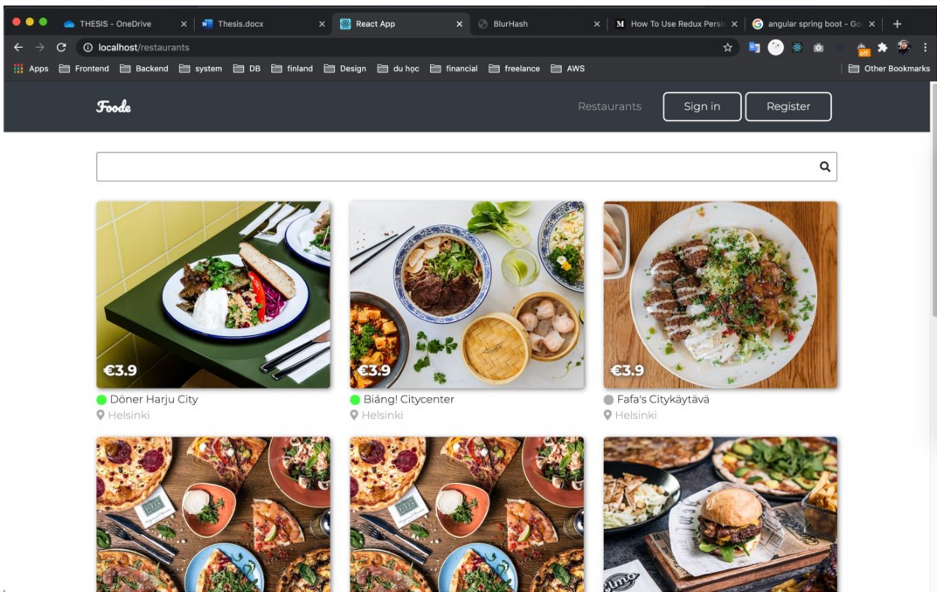
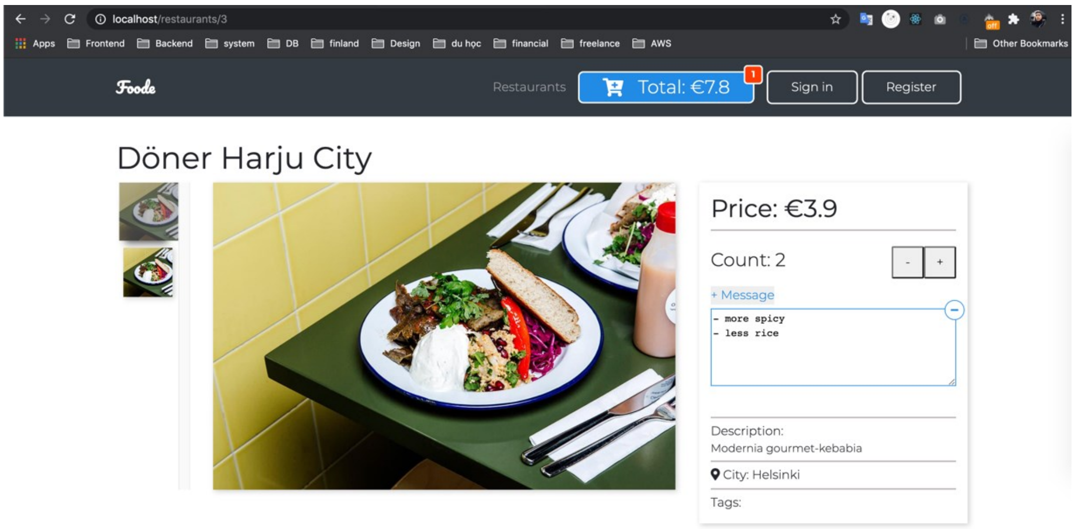


Image 6. Home page

In Image 7, the restaurant detail page is used to view, and select a particular meal. The users could increase or decrease the number of portions and leave a message in their order. On this page, Redux is applied to update the shopping cart data. The “blue button” in the header is used to indicate the total money. Whenever the users change the number of portions, the total money that users would pay, which changes properly.



Döner Harju City

Price: €3.9

Count: 2

```
- more spicy
- less rice
```

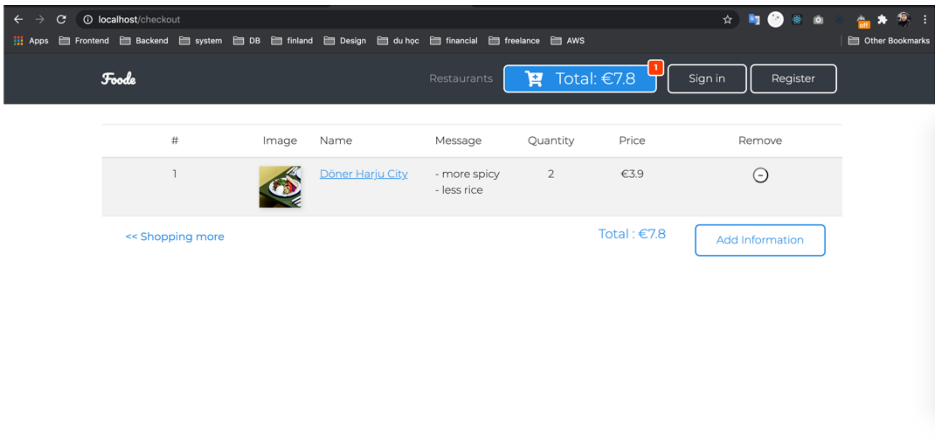
Description:
Modernia gourmet-kebabia

City: Helsinki

Tags:

Image 7. Restaurant detail page

After users finish selecting meals, users go to this page to check out and make an order. This page lists all the chosen meals and their quantity, the total amount of money. Besides, users also can remove meals if they change their minds. The checkout page looks like the example in Image 8.



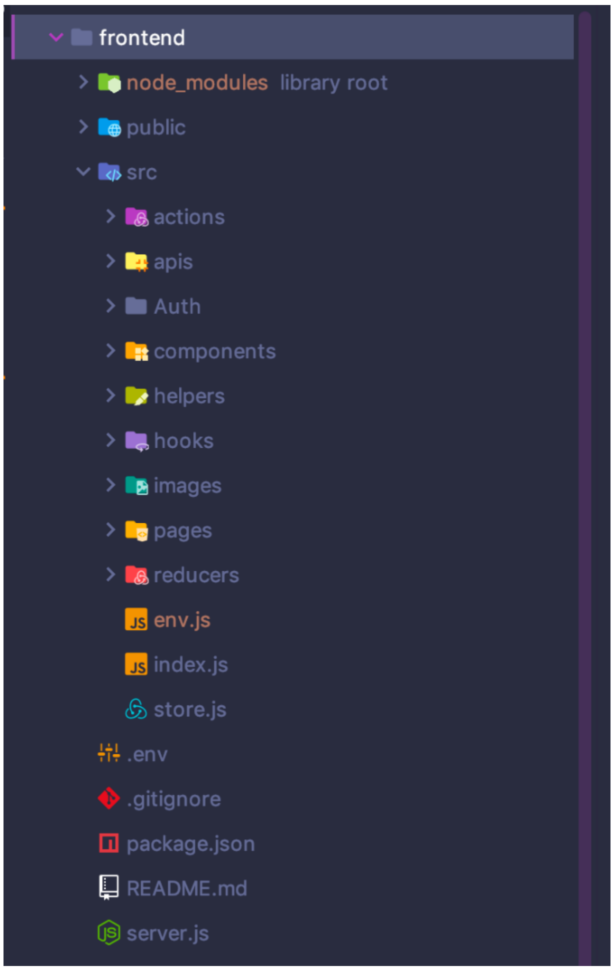
#	Image	Name	Message	Quantity	Price	Remove
1		Döner Harju City	- more spicy - less rice	2	€3.9	⊖

<< Shopping more Total : €7.8 Add Information

Image 8. Checkout page

3.3 Application structure

As the image below, React.js's folder is structured like this. There are many files and folders in the project. In the scope of this thesis, some of the main folders are introduced.



```
frontend
  node_modules  library root
  public
  src
    actions
    apis
    Auth
    components
    helpers
    hooks
    images
    pages
    reducers
    env.js
    index.js
    store.js
  .env
  .gitignore
  package.json
  README.md
  server.js
```

Image 9. Application structure

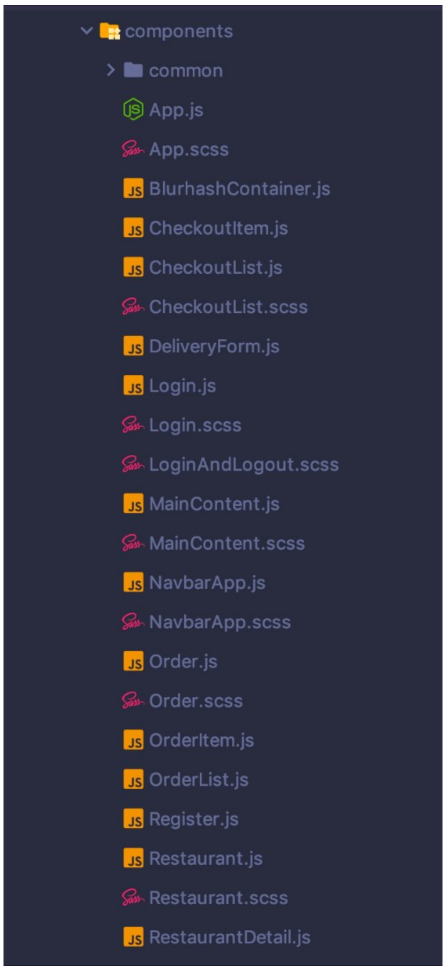
The node_modules folder contains all the dependencies (packages) which are used in this project. In any node's project, "node_modules" is always in the root directory. The src folder holds the source code of the project. Initially, React.js read the index.js file in src folder as the root development file in the application. All the logics and components are imported into this file. The following sections are explaining some of the core features in such folders.

Like the node_modules folder, the package.json is another Node.js's file. It describes the meta-information, such as the name of the project, versions, dependencies. When a dependency is installed in the project, its information is updated automatically in this file. This file is like the dependency bookkeeping of the node project. Due to the node_modules' huge size, all the Node.js's projects do not need to keep the node_modules folder when we push to Github, but keep the package.json. When developers want to install the packages, they need to run this command:

```
npm install
```

3.4 Components in React.js

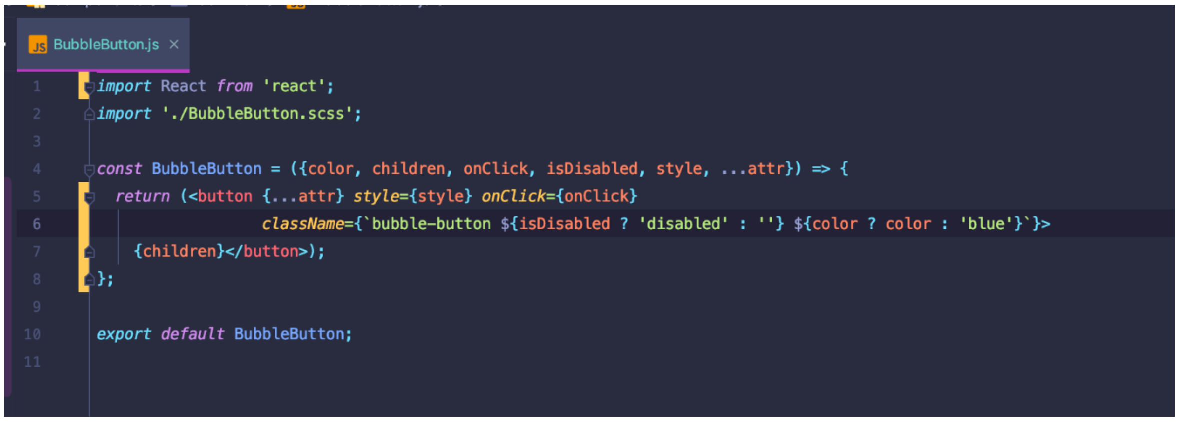
Each page of a web application has been built based on many components (see the Image 10). Depending on which components the page needs, developers could attach those components within this page. One of the most common components that web applications usually use is a button. Buttons could be used in many places in the app. In this part, the authors use the Bubble Button component to show how convenient React.js components in the application. And how the component connects and shares dynamic visual representation and logic.



```
components
  common
  App.js
  App.scss
  BlurhashContainer.js
  CheckoutItem.js
  CheckoutList.js
  CheckoutList.scss
  DeliveryForm.js
  Login.js
  Login.scss
  LoginAndLogout.scss
  MainContent.js
  MainContent.scss
  NavbarApp.js
  NavbarApp.scss
  Order.js
  Order.scss
  OrderItem.js
  OrderList.js
  Register.js
  Restaurant.js
  Restaurant.scss
  RestaurantDetail.js
```

Image 10. Component structure

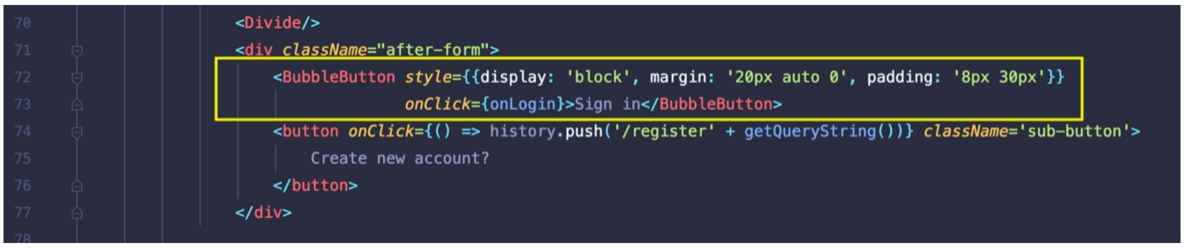
To reuse it, the authors created the BubbleButton component with various properties (see the Image 11). Every time the parent's components (page) want to use the BubbleButton component, they could add the BubbleButton inside their JSX and adjust the button style, colour, or any of values, based on their need via the properties. For example, on the Home page, developers modified the margin properties with the number: '20px auto 0', and on the About page, they could change the margin to: '50px auto 0' (see the Image 12).



```
BubbleButton.js ×
import React from 'react';
import './BubbleButton.scss';

const BubbleButton = ({color, children, onClick, isDisabled, style, ...attr}) => {
  return (<button {...attr} style={style} onClick={onClick}
                   className={`bubble-button ${isDisabled ? 'disabled' : ''} ${color ? color : 'blue'}`}>
    {children}</button>);
};

export default BubbleButton;
```

Image 11. Component BubbleButton definition

Thanks to components, developers do not need to rewrite the logic everywhere. Components also help them easily fix and maintain the code by going through one place (BubbleButton component).

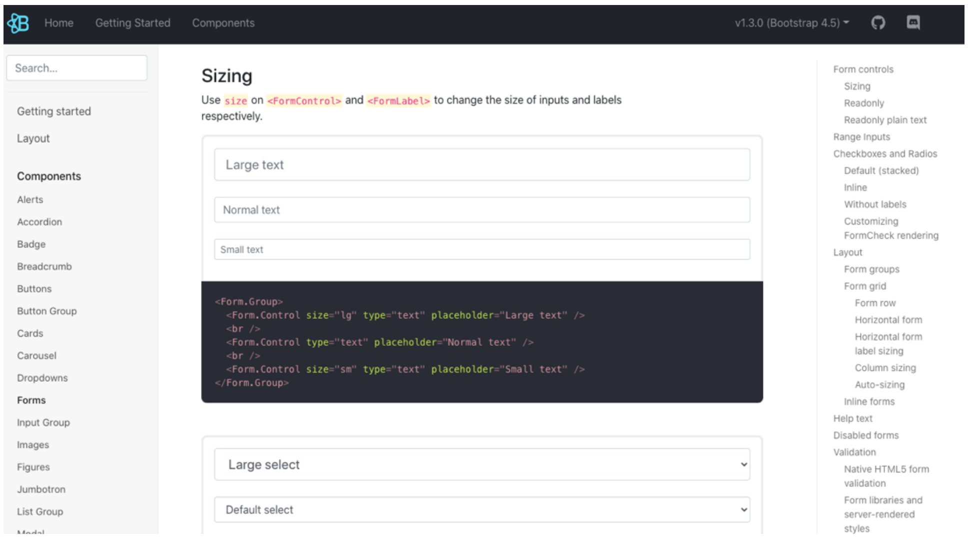


```
<Divide/>
<div className="after-form">
    <BubbleButton style={{display: 'block', margin: '20px auto 0', padding: '8px 30px'}}
                  onClick={onLogin}>Sign in</BubbleButton>
    <button onClick={() => history.push('/register' + getQueryString())} className='sub-button'>
        Create new account?
    </button>
</div>
```

Image 12. Using BubbleButton in another component

3.5 React.js's user interface library – React Bootstrap

React Bootstrap is the combination of React.js and Bootstrap. There are many Bootstrap themes from the Bootstrap style sheets that are available. It helps developers save time by using Bootstrap's components. Instead of creating the Form component themselves, they can install React-Bootstrap packages that use those components freely.



Sizing

Use `size` on `<FormControl>` and `<FormLabel>` to change the size of inputs and labels respectively.

```
<Form.Group>
  <Form.Control size="lg" type="text" placeholder="Large text" />
  <br />
  <Form.Control type="text" placeholder="Normal text" />
  <br />
  <Form.Control size="sm" type="text" placeholder="Small text" />
</Form.Group>
```

Image 13. React bootstrap documentation

3.6 Features of the web application

This chapter introduces the main features of the application. The authors indicate the benefits of the Image Loading React component. The authors show how Redux was applied to the app via the Shopping Cart component.

3.6.1 Image lazy loading

BlurHash is a compact representation of a placeholder for an image (Wolt, 2020). Without BlurHash, the images's placeholder is blank and grey during waiting for the page completed loading these images (Image 14). This effect makes the page look weird and ugly or might confuse the user. Therefore, BlurHash came to solve this problem. Blurhash creates by Engineer Dag Ågren, working at Wolt. When the app is applied BlurHash, the represent placeholders are given the short image URL and short “hash” string from the backend. This short “hash” string plays the role of blurring the pictures during loading; the strings help the page smoother and increasing experience for the user (Image 15).

The image 14 shows the website when it is not applied BlurHash.

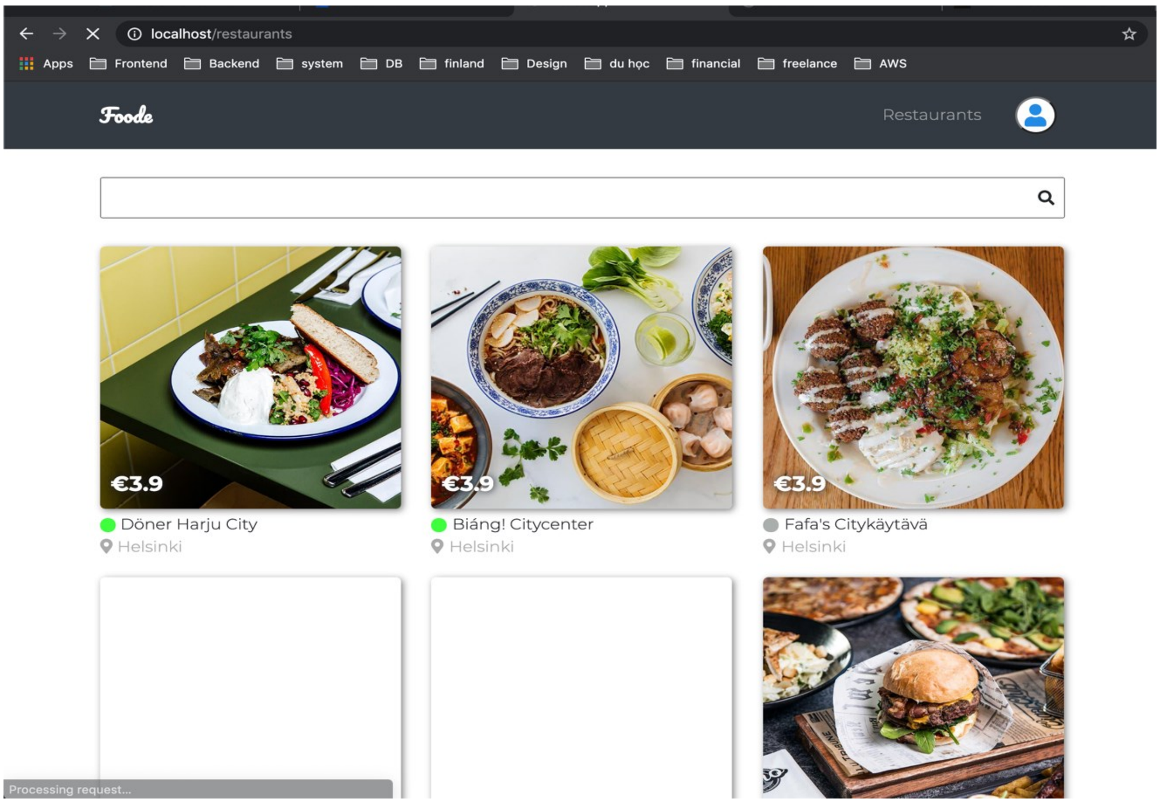


Image 14. Web app without BlurHash

The image 15 shows the website when it is applied BlurHash.

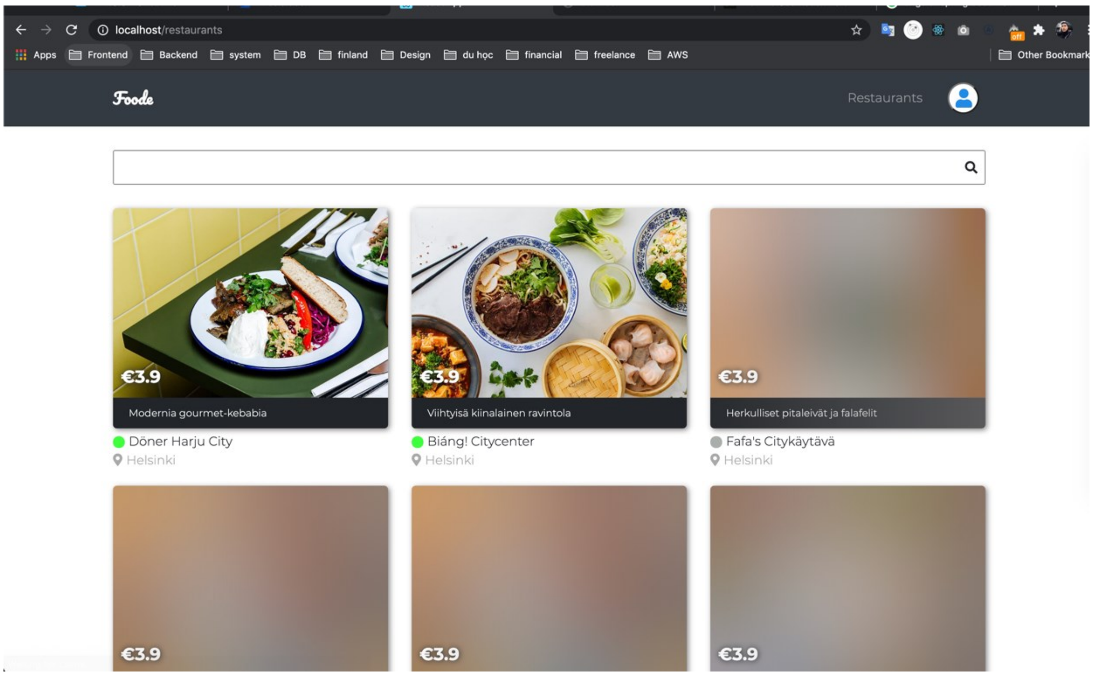
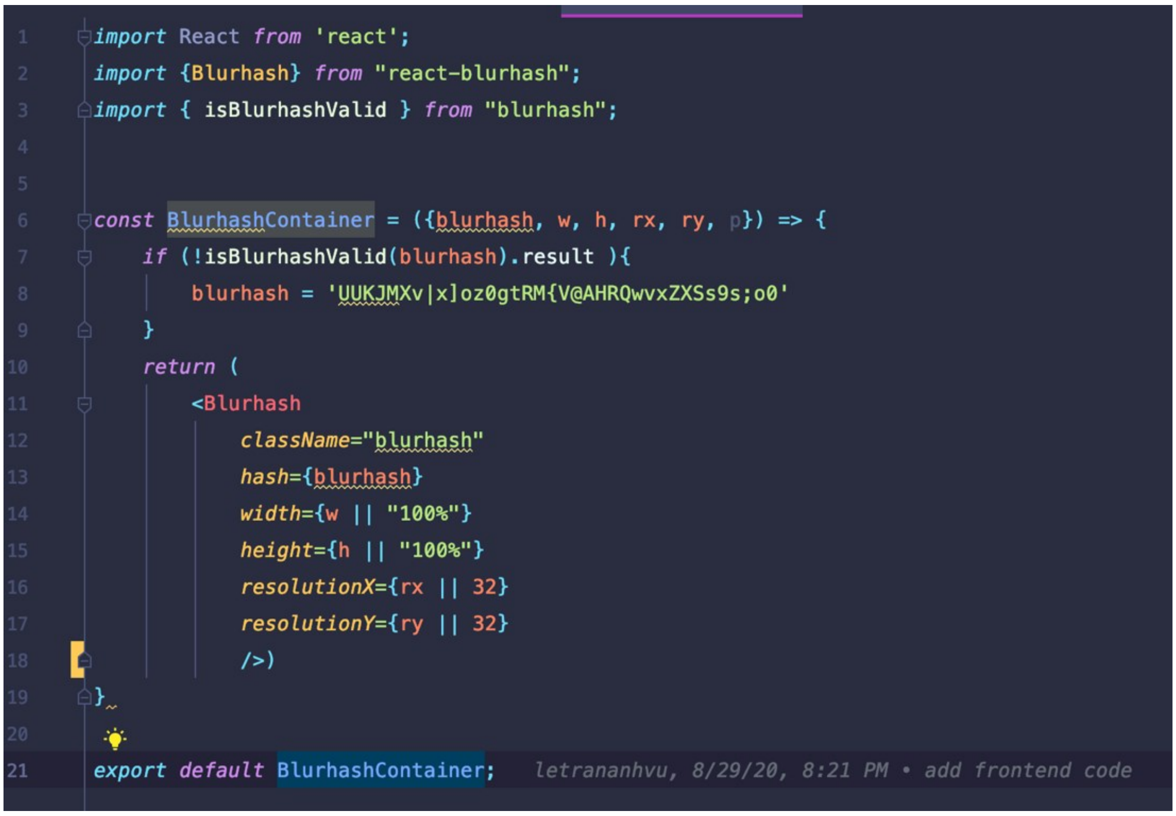


Image 15. Web app applied BlurHash

To use react-blurhash package, the author created “<BlurhashContrainer />” component with simple code at Image 16. Then this component could be used everywhere.

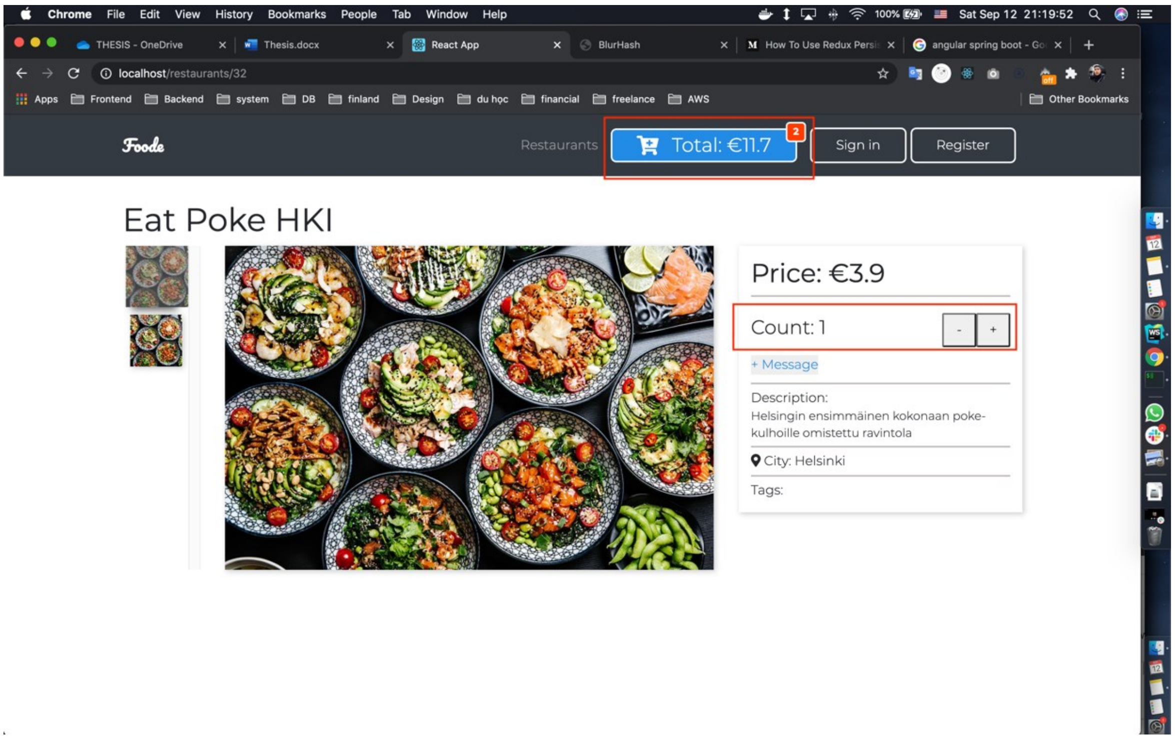


```
import React from 'react';
import {Blurhash} from "react-blurhash";
import { isBlurhashValid } from "blurhash";


const BlurhashContainer = ({blurhash, w, h, rx, ry, p}) => {
    if (!isBlurhashValid(blurhash).result ){
        blurhash = 'UUKJMXv|x]oz0gtRM{V@AHRQwvxZXSs9s;o0'
    }
    return (
        <Blurhash
            className="blurhash"
            hash={blurhash}
            width={w || "100%"}
            height={h || "100%"}
            resolutionX={rx || 32}
            resolutionY={ry || 32}
            />)
}

export default BlurhashContainer;    letrananhvu, 8/29/20, 8:21 PM • add frontend code
```

Image 16. Applying Blurhash component in project

3.6.2 Shopping cart – Redux

In this section, the authors go through Redux's features. In the application, there are functions such as Add or Remove items in the Shopping Cart. How the Store, Reduce, and Action have been applied to the application.



Description:
Helsingin ensimmäinen kokonaan poke-kulhoille omistettu ravintola

City: Helsinki

Tags:

Image 17. Restaurant detail page

In the image above, whenever the “+” (plus) or “-” (minus) button is clicked, a new meal is added to the cart. Besides, the cart information is used in multiple places. First, the header's cart button uses such information to indicate the total amount of money that the user has shopped and the number of meals. Second, the right-side block of meal detail also needs cart information to check whether this meal is picked or not. The simple wireframe below presents the usage of cart information.

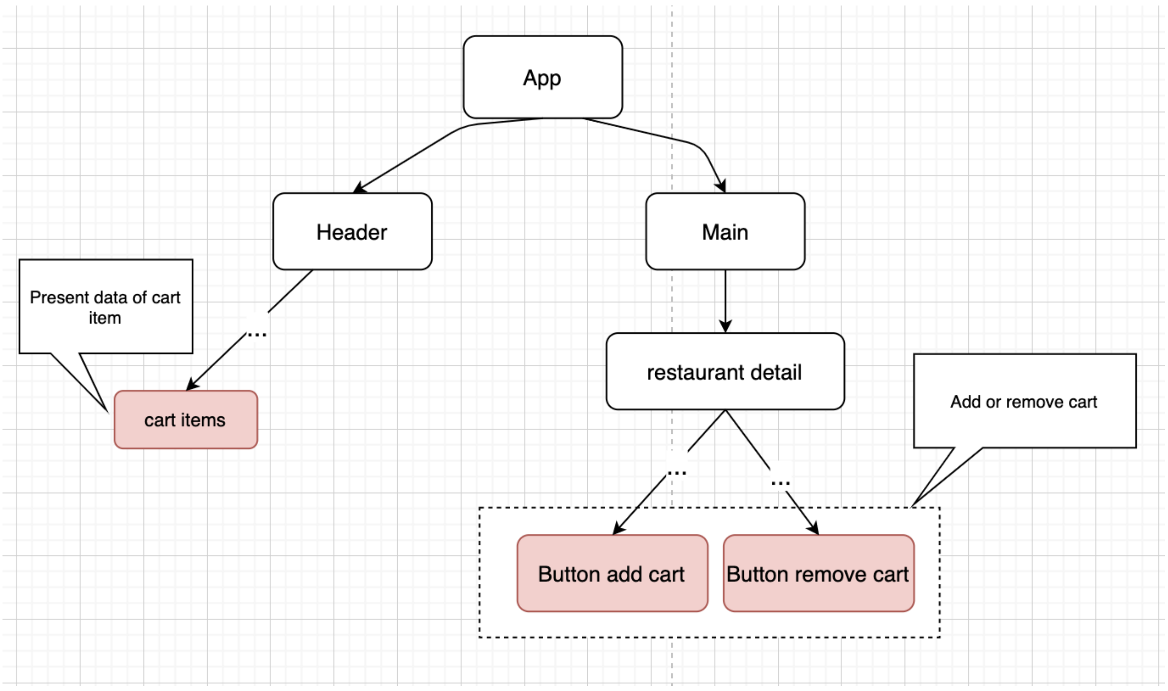


Figure 10. Restaurant detail wireframe

To implement this feature with React.js, the developers must apply heavy coupling between components. The red arrows denote the direction when cart information is updated on the left side of the image below. When the user clicks the buttons, these click events must go up to the App component. After that, the App component passes the updated cart information through many components until it reaches to Cart Button component. There are a lot of works to do for both developers and React.js. Moreover, it also tights the relationship between the components (each component which the arrow point to, must implement the way to pass Cart Information). Therefore, it reduces the dynamic and reusable of the component in React.js. The most beneficial component is like a Lego's block that could be plugged or unplugged. Thus, the solution on the left side of the image below is lengthy and hard to maintain.

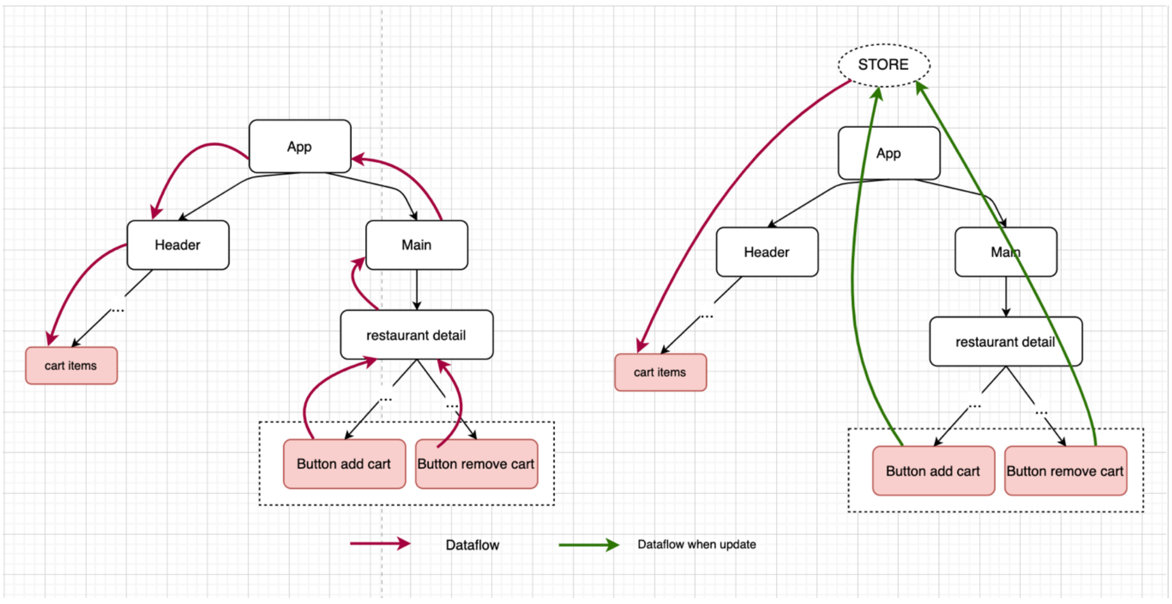
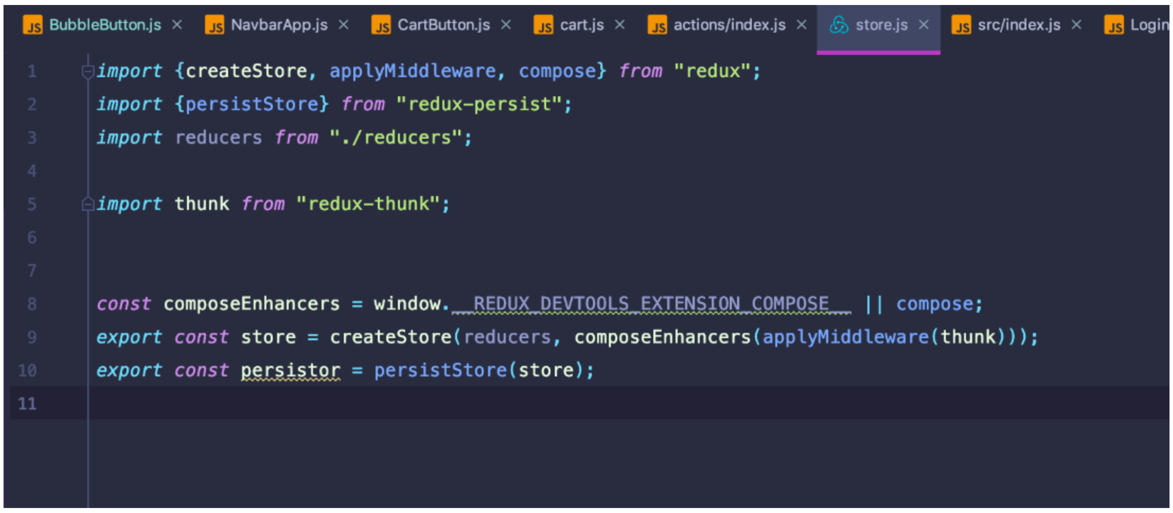


Figure 11. Cart information dataflow

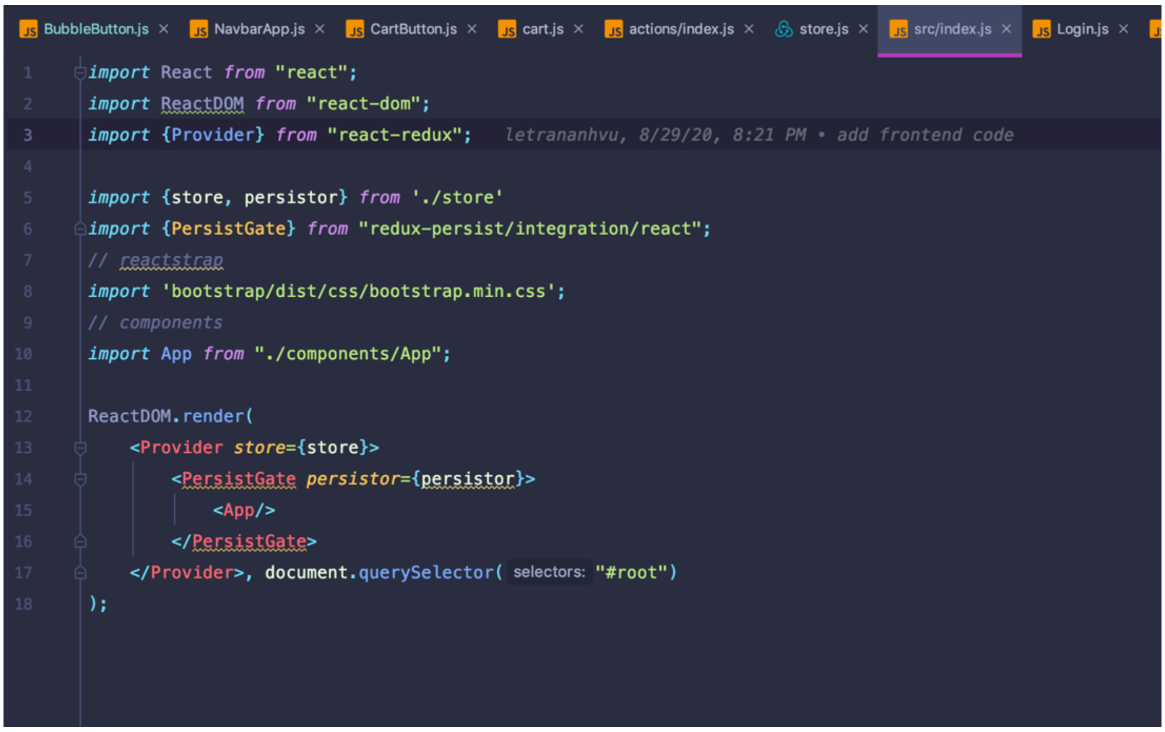
That is why React-Redux comes to the rescue. In the right side of the Figure 11, the cart information is stored in the “store” of Redux. Therefore, when the buttons or any events would like to change cart information, it dispatches the changing action to Redux and let Redux updates data. Then, the updated data would be sent to needed components such as the Cart button in the header. This approach loses the relationship between components, reduces the number of pass properties between components. Besides, it helps the application more reliable, highly scalable, and easy to be maintained.



```
1  import {createStore, applyMiddleware, compose} from "redux";
2  import {persistStore} from "redux-persist";
3  import reducers from "./reducers";
4
5  import thunk from "redux-thunk";
6
7
8  const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__ || compose;
9  export const store = createStore(reducers, composeEnhancers(applyMiddleware(thunk)));
10 export const persistor = persistStore(store);
11
```

Image 18. Declaration of Redux’s store

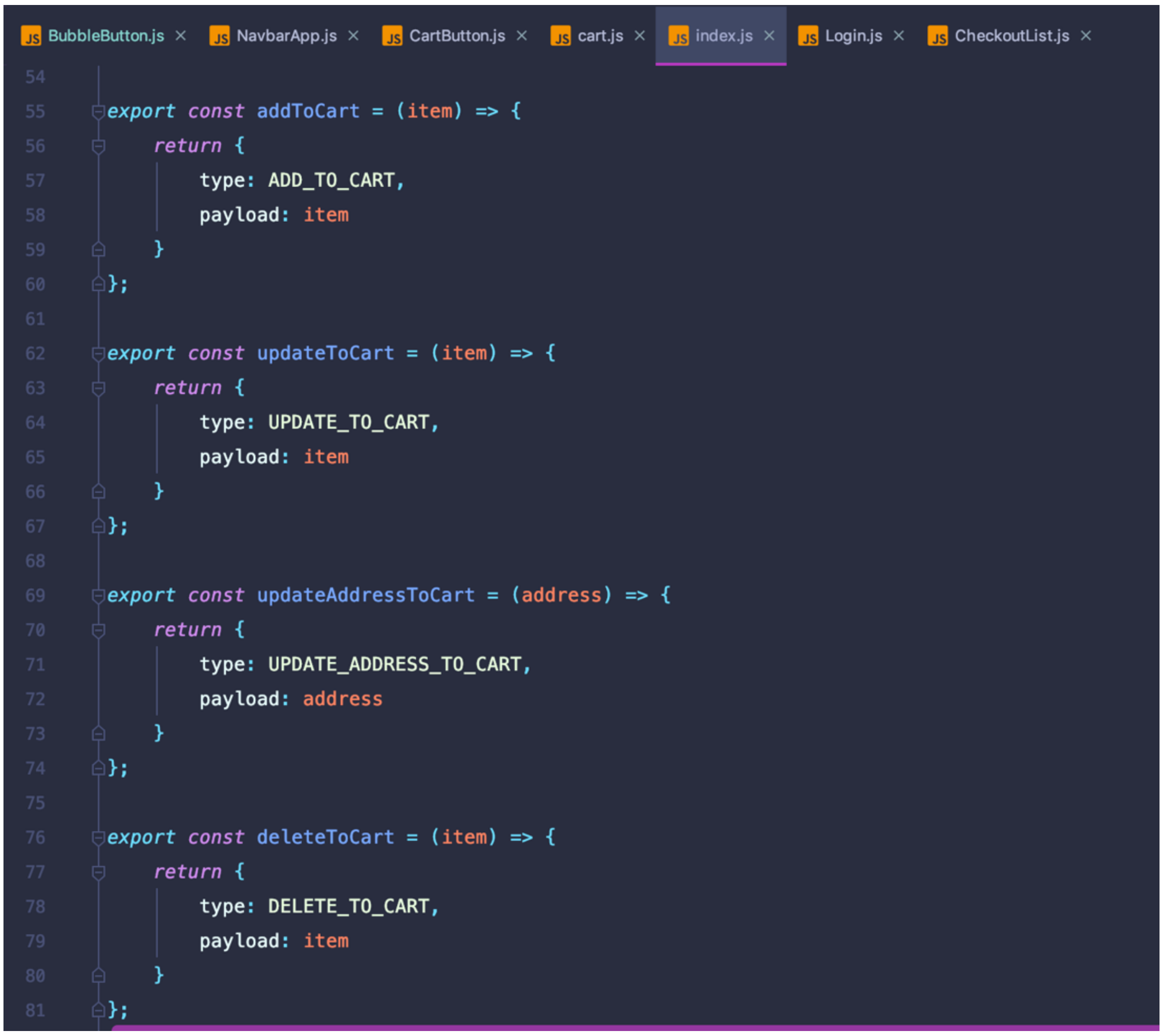
Suppose the application would like to use Redux. It must declare a “store”, “actions,” and “reducers”. In the Image 18, the store is created in line 9. After that, it is imported into the src/index.js file (Image 19, line 5) for attaching to the application root. From now, the store is life in the global of the application. Therefore, it can be used in any component of the project.



```
1   import React from "react";
2   import ReactDOM from "react-dom";
3   import {Provider} from "react-redux";    letrananhvu, 8/29/20, 8:21 PM • add frontend code
4
5   import {store, persistor} from './store'
6   import {PersistGate} from "redux-persist/integration/react";
7   // reactstrap
8   import 'bootstrap/dist/css/bootstrap.min.css';
9   // components
10  import App from "./components/App";
11
12  ReactDOM.render(
13      <Provider store={store}>
14          <PersistGate persistor={persistor}>
15              <App/>
16          </PersistGate>
17      </Provider>, document.querySelector( selectors: "#root")
18  );
```

Image 19. Applying Redux’s store to the project

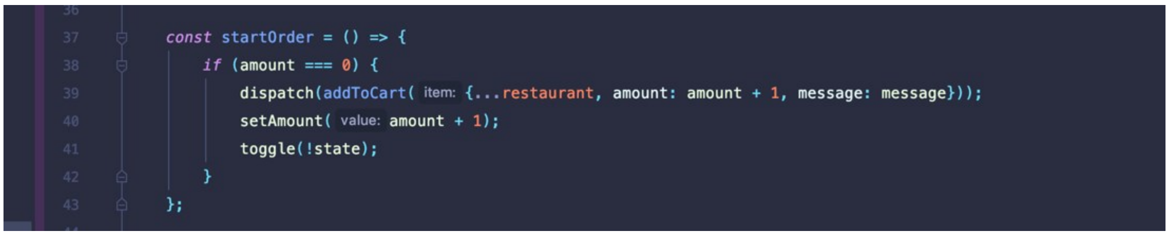
The action creators are declared in Image 20. they return the action, which contains types and payloads. The types are the name of the actions, and the payloads are the changed data. For instance, the "add an item to cart" action contains the "ADD_TO_CART" type and "item" payload. It is very straightforward to guess the usage of this action is to add new item to the cart. When developers want to use any action, they are dispatching this action in the component. To firing the dispatch function in the component, developers can trap the dispatch function in a particular event such as clicking or typing.



```
export const addToCart = (item) => {
    return {
        type: ADD_TO_CART,
        payload: item
    }
};

export const updateToCart = (item) => {
    return {
        type: UPDATE_TO_CART,
        payload: item
    }
};

export const updateAddressToCart = (address) => {
    return {
        type: UPDATE_ADDRESS_TO_CART,
        payload: address
    }
};

export const deleteToCart = (item) => {
    return {
        type: DELETE_TO_CART,
        payload: item
    }
};
```

Image 20. Declaration of Redux's actions

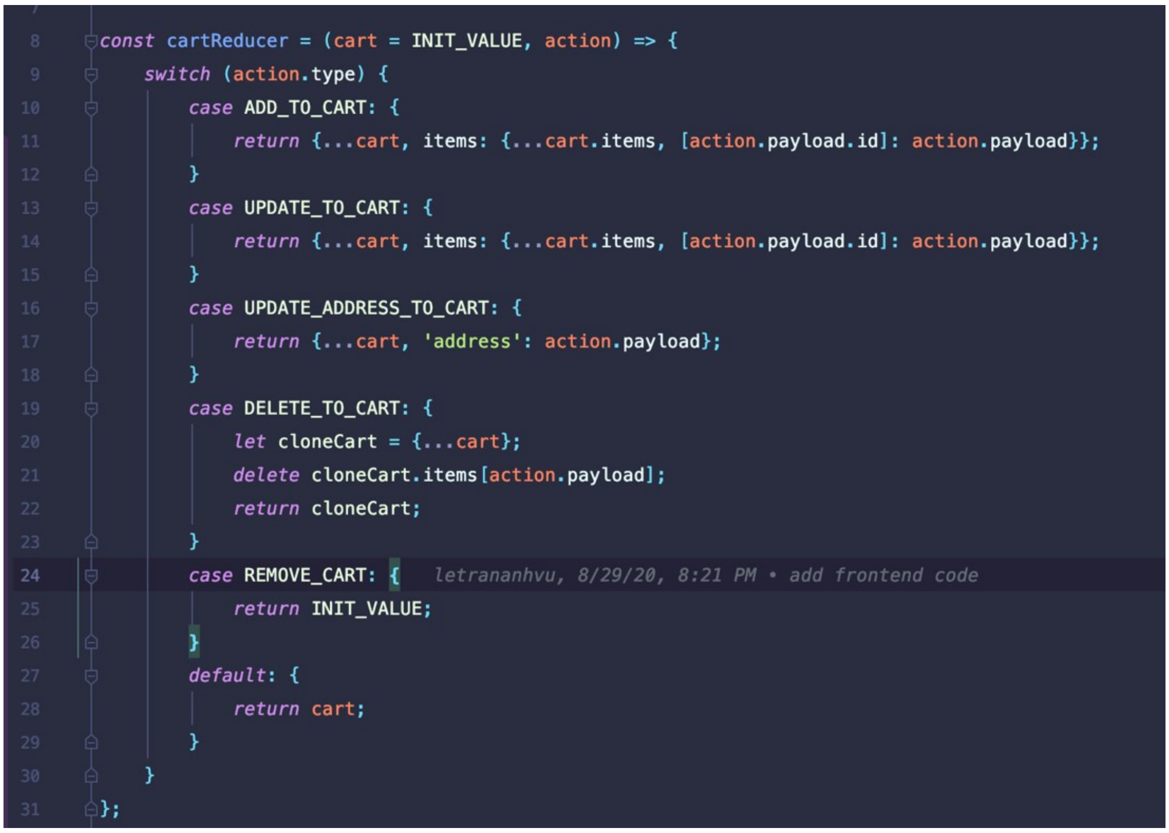
The image below is a call-back function located in a button component and trapped to the clicking event. So that, whenever users click the “add to cart” button on the screen, the “startOrder” function would be called and then dispatch the “addToCart” action.



```
const startOrder = () => {
    if (amount === 0) {
        dispatch(addToCart( item: {...restaurant, amount: amount + 1, message: message}));
        setAmount( value: amount + 1);
        toggle(!state);
    }
};
```

Image 21. A call-back function contains dispatching the action

After the action is dispatched, Redux looks up to all of the “reducers” to find the matched type. Based on the matched type, the reducer would return the new update data. For example, in the image below, when the ADD_TO_CART action is dispatched, it would return the update cart in line 11.



```
const cartReducer = (cart = INIT_VALUE, action) => {
    switch (action.type) {
        case ADD_TO_CART: {
            return {...cart, items: {...cart.items, [action.payload.id]: action.payload}};
        }
        case UPDATE_TO_CART: {
            return {...cart, items: {...cart.items, [action.payload.id]: action.payload}};
        }
        case UPDATE_ADDRESS_TO_CART: {
            return {...cart, 'address': action.payload};
        }
        case DELETE_TO_CART: {
            let cloneCart = {...cart};
            delete cloneCart.items[action.payload];
            return cloneCart;
        }
        case REMOVE_CART: {   letrananhvu, 8/29/20, 8:21 PM • add frontend code
            return INIT_VALUE;
        }
        default: {
            return cart;
        }
    }
};
```

Image 22. Declaration of Redux’s reducer

When the store's data is updated, it will let the subscribed components know about the changed data. Therefore, the subscriber component can re-render and show up to date data on the browser.

4 CONCLUSIONS

Web technology is getting more modern every day. Hundreds of technologies were created to solve existing technical problems and enhance experience for users. So, studying and choosing the technologies and adequate framework for web applications is necessary. This chapter concludes the thesis and answers the research questions:

- Why choose React?
- Why does React.js save time in application development and maintenance?
- What are the benefits of using Redux?

React.js library brings benefits for developing web applications. The advantage of React.js is that developers can share and reuse the components in the app. Besides, they can also customize the components as they wish via the properties and state features. Moreover, sharing the components in one source reduces the reduces workload and eases to maintain the app when needed. React.js is well-like among developers, and it is a library that they wish to work because the scalability of the project in the future is not difficult. Finally, React.js has a mature community that developers can easily find out the solution for their issue from the previous developers.

React-Redux is one of the most popular libraries in React.js application. The benefits of Redux have been proven in the implementation chapter. When data is shared between multiple components in the application, it is time to choose React-Redux. React-Redux helps to manage shared data more logical and reliable. To update data in React-Redux, developers must declare each Action. Then they decide how to update data in Reducer. React- Redux also relieves the coupling in components, so the components could be more reusable and dynamic. However, developers should concern before applying React-Redux in the project to avoid overkill at the beginning. According to Abramov and the Redux documentation authors (2020e), React-Redux is not built for a quick way to code, and developers should use it when they need it.

LIST OF REFERENCES

Aggarwal, S. 2018. Modern Web Development Using ReactJS. Retrieved on 3 September 2020. Available at: http://ijrra.net/Vol5issue1/IJRRA-05-01-27.pdf

Abramov, D & the Redux documentation authors. 2020a. Redux FAQ: React Redux. Retrieved on 3 September 2020. Available at: https://redux.js.org/faq/react-redux#why-should-i-use-react-redux

Abramov, D and the Redux documentation authors. 2020b. Redux Overview and Concepts. Retrieved on 3 September 2020. Available at: https://redux.js.org/tutorials/essentials/part-1-overview-concepts

Abramov, D and the Redux documentation authors. 2020c. Three principles. Retrieved on 3 September 2020. Available at: https://redux.js.org/introduction/three-principles

Abramov, D and the Redux documentation authors. 2020d. Redux Style Guide. Retrieved on 3 September 2020. Available at: https://redux.js.org/style-guide/style-guide

Abramov, D and the Redux documentation authors. 2020e. When should I use Redux. Retrieved on 3 September 2020. Available at: https://redux.js.org/faq/general#when-should-i-use-redux

Chaffey, D., Smith, P 2017. Digital Marketing Excellence: Planning, Optimizing and Integrating Online Marketing. Taylor & Francis Group.

Chander, A. 2013. The Electronic Silk Road: How the Web Binds the World Together in Commerce. Yale University Press.

Creswell, J.W. 2003 Research Design Qualitative, Quantitative, and Mixed Methods Approaches. 2nd Edition. Thousand Oaks: Sage Publications, Inc.

Casciaro, M. 2014. Node.js Design Patterns. Packt Publishing.

Dhandapani, S. 2020. Virtual DOM - the Difference Maker in React JS. Pluralsight LLC. Retrieved on 3 September 2020. Available at: https://www.pluralsight.com/guides/virtual-dom-difference-maker-react-js

Developer Survey. 2020. Most Popular Technologies. Stack Overflow. Retrieved on 3 September 2020. Available at: https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages-professional-developers

Elliott, E. 2014. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries. O'Reilly Media.

Facebook Open Source. 2020a. Virtual DOM and Internals. Facebook Inc. Retrieved on 3 September 2020. Available at: https://reactjs.org/docs/faq-internals.html

Facebook Open Source. 2020b. Introducing JSX. Facebook Inc. Retrieved on 3 September 2020. Available at: https://reactjs.org/docs/introducing-jsx.html

Gackenheimer, C. 2015. Introduction to React. New York.

GitHub. 2020a. Angular. Retrieved on 6 September 2020. Available at: https://github.com/angular/angular

GitHub. 2020b. React. Retrieved on 6 September 2020. Available at: https://github.com/facebook/react

GitHub. 2020c. Vue. Retrieved on 6 September 2020. Available at: https://github.com/vuejs/vue

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

Mardan, A. 2017.React Quickly: painless web apps with React, JSX, REDUX, and GraphQL. Shelter Island, NY, Manning Publications.

Mozilla & individual contributors. 2020a. HTML &CSS. Retrieved on 18 July 2020. Available at: https://developer.mozilla.org/en-US/docs/Web/HTML

Mozilla & individual contributors. 2020b. What is JavaScript?. MDN. Retrieved on 29 August 2020. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

Naukri Engineering. 2017. Virtual Dom. Retrieved on 4 October 2020. Available at: https://medium.com/naukri-engineering/naukriengineering-virtual-dom-fa8019c626b

NPM. 2020a. React. Retrieved on 6 September 2020. Available at: https://www.npmjs.com/package/react

NPM. 2020b. Vue. Retrieved on 6 September 2020. Available at: https://www.npmjs.com/package/vue

NPM. 2020c. Angular. Retrieved on 6 September 2020. Available at: https://www.npmjs.com/package/@angular/core

OpenJS Foundation. 2020. Retrieved on 9 September 2020. Available at: https://nodejs.org/en/

Satheesh, M. & D'mello, B. & Krol, J. 2015. Web Development with MongoDB and NodeJS. 2nd Edition. Packt Publishing

Teixeira, P. 2012. Professional Node.js: building Javascript based scalable software. Hoboken, N.J., Wiley.

Wattenberger, A. 2019a. Data layer. State of JavaScript. Retrieved on 9 September 2020. Available at: https://2019.stateofjs.com/data-layer/

Wattenberger, A. 2019b. Front End Frameworks. State of JavaScript. Retrieved on 3 September 2020. Available at: https://2019.stateofjs.com/front-end-frameworks/