Bachelor’s thesis

Information Technology

2019

Yan Li

FRONT-END TESTING – AN IMPORTANT PART OF QUALITY ASSURANCE IN FRONT-END DEVELOPMENT

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology

2019 | 47

Yan Li

FRONT-END TESTING — AN IMPORTANT PART OF QUALITY ASSURANCE IN FRONT-END DEVELOPMENT

Thanks to the significantly improved web technology, nowadays, we can watch videos, play games, check maps, book tickets and fill application forms on websites. The websites have become more and more dynamic and interactive than before. Front-end development has made a great contribution to this since several features are being implemented into different websites constantly. However, more features means more work behind the scenes and ensuring everything works as expected on different devices and different browsers is crucial for the front-end developers.

The purpose of this thesis was to show how significant front-end testing is and how to implement the tests in order to make websites run properly under different circumstances. In this thesis, some basic knowledge on front-end development was introduced and front-end testing was widely discussed, finally, a web application was built along with chosen testing tools including Jasmine, TestCafe, BrowserStack, BackstopJS and Axe to demonstrate how to perform unit test, functional test, cross-browser test, visual regression test and accessibility test respectively. Most of the tests passed for the first time and some tests failed with specified error information. Additionally, Continuous Integration workflow was also conducted to show how it worked which turned out that it was a very efficient and convenient workflow for software development and software testing.

KEYWORDS:

Front-end testing, JavaScript, CSS, testing tools, Continuous Integration, Software Quality Assurance

CONTENTS

FIGURES

LIST OF ABBREVIATIONS (OR) SYMBOLS

CI	Continuous Integration
CMMI	Capability Maturity Model Integrated
CSS	Cascading Style Sheets
DOM	Document Object Model
E2E	End-to-End
ES	ECMAScript
HTML	Hypertext Markup Language
IE	Internet Explorer
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
JS	JavaScript
POS	Point of Sale
SDLC	Software Development Life Cycle
SPA	Single Page Application
SQA	Software Quality Assurance
TMM	Test Maturity Model
UI	User Interface
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WCAG	Web Content Accessibility Guidelines
WHO	World Health Organization

1 INTRODUCTION

Web development has evolved dramatically in recent years. Most of the websites we visit everyday are far more different than before; they are more dynamic, interactive and attractive, it is no longer just text on the screen with some links and a few buttons or with some pictures inserted sometimes. Nowadays, there are many more activities we can do on a website, such as watching videos, playing games, checking maps, booking tickets, filling application forms and communicating instantly and so on. All of these features can be easily implemented into a website thanks to the fast growth of front-end development.

However, as more and more features are being implemented into websites, how can we make sure that everything will still work as expected when they are being displayed on different devices with different screen sizes and browsers? When a website becomes larger and more complex, any changes may prevent it from working properly, for example, some minor changes on CSS could lead to the lose of responsiveness; A variable in JavaScript declared in a modern way may not supported in some old fashioned browsers like IE; Also, a new added function with a bad way of dealing with mathematics will slow down the loading speed of that page. There are many other issues like this which affect the normal operation of a website in front-end development.

In reality, the best practice for the issues mentioned above is constantly tesing during front-end development. Many developers are still confused about front-end testing due to the uncertainty of what to test. Front-end testing includes unit testing, UI testing, performance testing, accessibility testing, etc. Nonethless, with front-end development becoming more and more complex, it is the developers’ responsibility to assure the stability and consistency (Sedlock, 2017).

This thesis aims to emphasize the importance of front-end testing and demonstrate how to implement the tests. In the thesis, a set of chosen testing tools will be used for testing a created web application to make sure everything works normally.

Chapter 1 introduces basic knowledge on the front-end development. Chapter 2 discusses front-end testing which includes why to test, what to test and some testing tools that we need for testing. Chapter 3 discusses testing as part of development process. Chapter 4 demonstrates the implementation of front-end testing and Chapter 5 gives the conclusion on the whole thesis.

2 KNOWLEDGE ON FRONT-END DEVELOPMENT

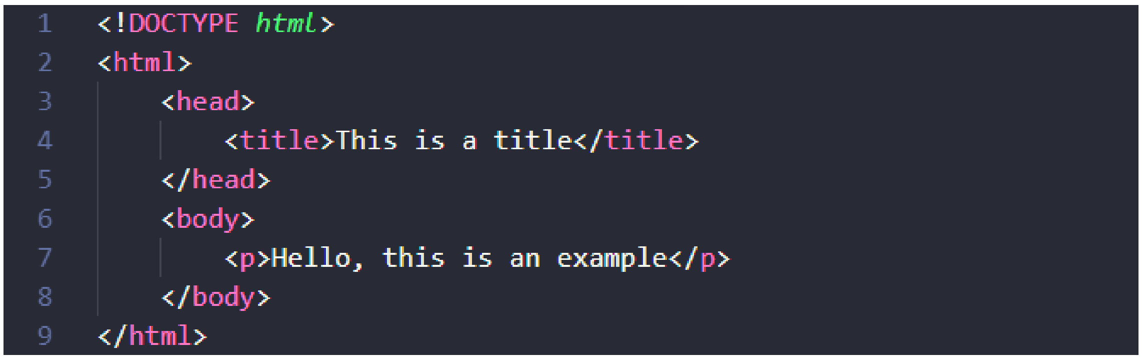
2.1 Basic concepts

Front-end web development, also known as client-side development is the practice of producing HTML, CSS and JavaScript for a website or Web Application so that a user can see and interact with them directly (Lindley, 2018).

2.1.1 HTML

HTML is short for Hypertext Markup Language. It was created by Tim Berner-Lee in the early 1990s and the first public release was in 1991. In 1993, the IETF published the first HTML proposal as a standard. The first full official specification-HTML 2.0-was released in 1995 and the W3C took over the HTML standard and published HTML 3.2 in 1997. By the end of 1997, HTML 4.0 was released with three variations: strict, transitional and frameset. It was the thriving time of web and it had been the HTML standard for more than a decade until late 2014, HTML 5.0 appeared.

HTML is the foundation of a website and tags are the building blocks of it. All the contents including links, images and videos we see on a website are made by HTML tags. HTML tag works in pairs-opening tags and closing tags-with some exceptions such as <input> tag and <img> tag. HTML has many tags to display the content on a website, including<a></a>, <p></p>, <div></div>, <form></form>, etc. Figure 1 below is a simple example of HTML:



```html
<!DOCTYPE html>
<html>
    <head>
        <title>This is a title</title>
    </head>
    <body>
        <p>Hello, this is an example</p>
    </body>
</html>
```

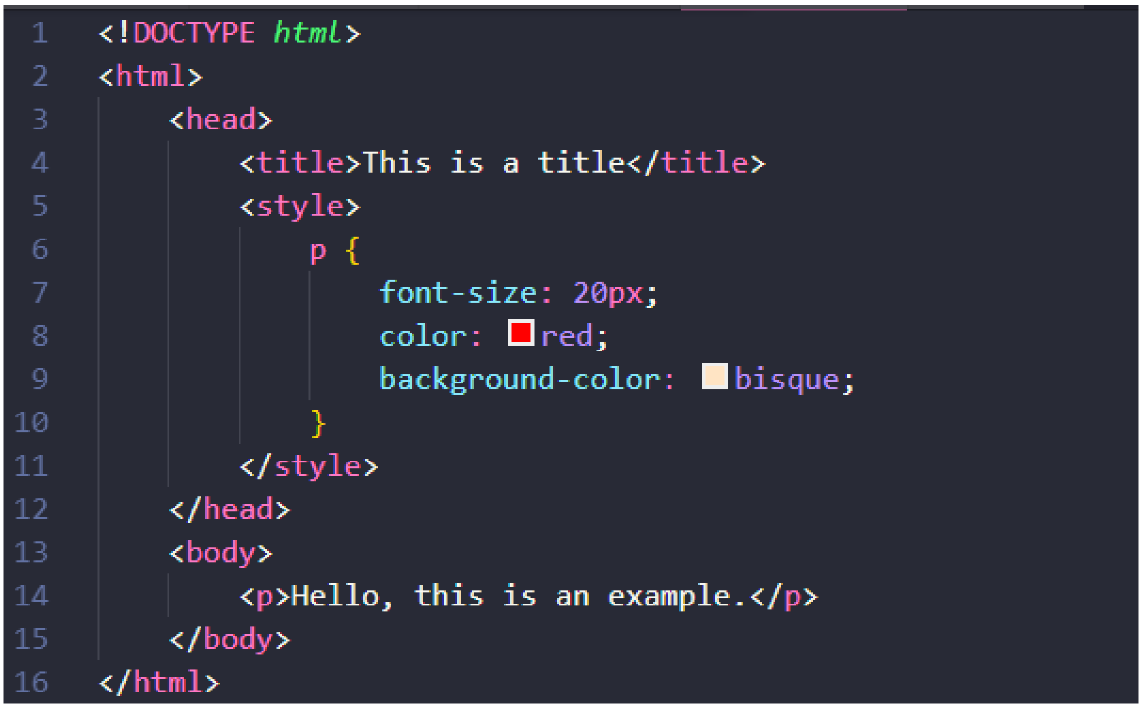
Figure 1. HTML code example.

2.1.2 CSS

CSS stands for Cascading Style Sheets. It is a style sheet language used for describing the presentation of a document written in a markup language like HTML.

CSS can decorate a website and make it the way people desire. such as, colours of a page, shapes and position of items and some animation effects, all of which can be taken care of by CSS. CSS rules consist of properties and selectors. Properties with different values determine how the content of a web page looks, for example, making the background of the web page yellow and the size of the font 20 pixels. Selectors, on the other hand, allow us to choose which part of the page we want to apply these properties to, for instance, changing the font of the first paragraph.

Based on the previous example, some CSS rules can be applied like this.



```
<!DOCTYPE html>
<html>
    <head>
        <title>This is a title</title>
        <style>
            p {
                font-size: 20px;
                color: 🟥red;
                background-color: ⬜bisque;
            }
        </style>
    </head>
    <body>
        <p>Hello, this is an example.</p>
    </body>
</html>
```

Figure 2. CSS code example.

The CSS rules are usually placed in the <head> section when there are no so many rules for the page in a small project. However, when a project becomes larger, the best practice is to separate the CSS rules into a single file and link it in the <head> section in the HTML file.

CSS has become more and more powerful, there are many CSS frameworks that have been created based on it, such as, Bootstrap, Foundation, Semantic UI, Bulma and UIkit and so on, among which Bootstrap and Foundation are the most popular CSS framework due to their powfulness and ease of use. The CSS framework contains most of CSS properties and provides more or less different modules such grid, icons and web typography. Frameworks make it much easier and time-saving when developing on a larger project since the most of the style has been preset, front-end developers only need to make some modifications to make it the way they want.

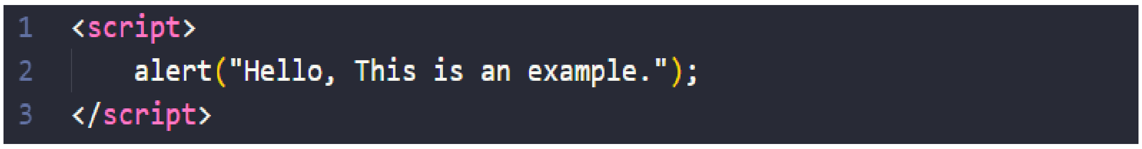
2.1.3 JavaScript

JavaScript is a cross-platform and powerful client-side scripting language. It was created by Brendan Eich in 1995. JavaScript was called LiveScript in the beginning and renamed later. Also, ECMAScript cannot be ignored when JavaScript is being discussed as it is a simple standard for JavaScript and it adds new features to it. ECMAScript is often short for ES, it is a subset of JavaScript and JavaScript is basically ECMASript at its core but built upon it. The latest version of JavaScript at the moment is ECMAScript 2018, which was finalized in June 2018, known as ES9 (madasamy, 2018).

JavaScript is one of the most popular programming languages in the world although it was not as powerful and popular as it is today during the early days. As the fast development of the web world, Javascript has become more and more important in the web development, it constantly evolves with powerful features and makes websites more dynamic and interactive (flaviocopes.com/javascript-introduction/, 2018).

Since JavaScript is a scripting language, it cannot run on its own and it runs on browsers. Any modern web browser supports javascript but some new features of JavaScript are not supported in some old browsers, such as Internet Explorer. When an HTML web page with some JavaScript in it is requested by a user, the script is sent to the browser and the browser then executes it (guru99.com/introduction-to-javascript.html, 2019).

A simple JS example shown as below:



```
<script>
    alert("Hello, This is an example.");
</script>
```

Figure 3. JavaScript code example.

After running the code above in a browser, the message in the parenthesis will be displayed in a pop-up window in the browser.

Similar to CSS rules, the best practice for JavaScript file is to separate it from the HTML file for the sake of efficiency and convenience when a project gets larger. However, if a small JavaScript needs to be added into the HTML file, the best place to hold the JavaScript is the bottom of the body section as the page will be loaded faster.

2.2 JavaScript and its achievement

The previous section discussed what front-end development is and what it includes. The basic knowledge of HTML, CSS and JavaScript has been introduced. However, since JavaScript is one of the most used programming languages and it has evoled so much during the past two decades. Besides, when testing the front-end development, in fact, we are testing JavaScript most of the time as it provides most of the functionalities and features, therefore, providing more details about JavaScript helps establish good understandings of front-end development.

JavaScript has become the absolute dominator in web development based on its popularity. According to a survey(insights.stackoverflow.com/survey/2018, 2018) among IT professionals and enthusiasts on Stack Overflow in 2018, JavaScript is still the most used language among the most popular technologies.

The result of the survey is shown below in Figure 4:

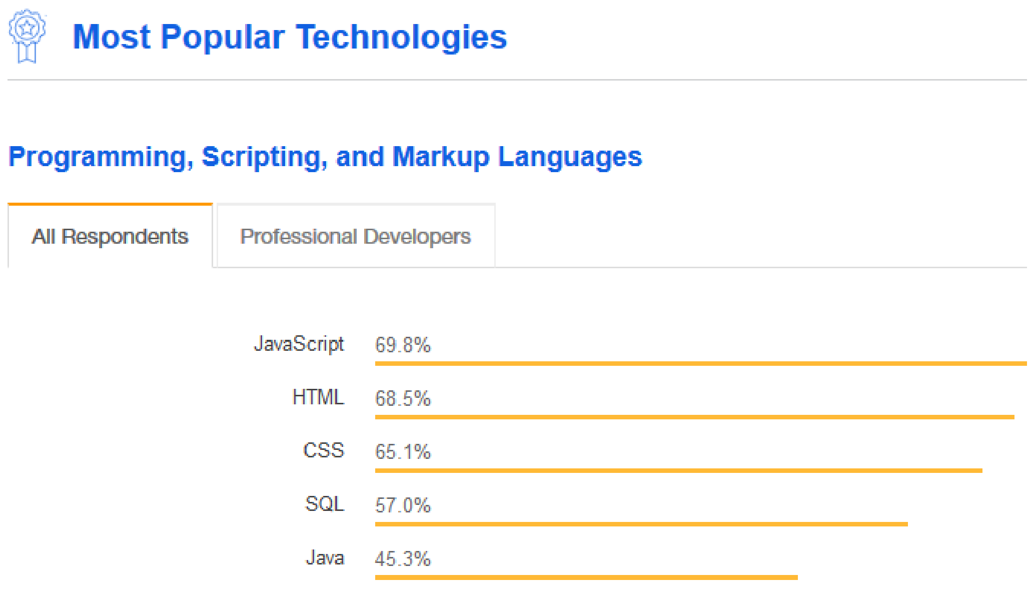


Figure 4. The popularity of JavaScript (Stack Overflow, 2018).

As JavaScript keeps developing and evolving, there have been more and more tools expanded from it, such as Angular (angular.io/docs, 2019), React (reactjs.org, 2019), Vue (vuejs.org, 2019), etc. These are JavaScript frameworks and libraries which make front-end development easier and faster. Sometimes, using plain javaScript (also called pure JavaScript or Vanilla Javascript) may not be very convenient or efficient enough for a project, this is when frameworks and libraries come in handy.

Frameworks and libraries are both written by someone else to provide help with some common problems. This two terms are occasionally used interchangeably, however, there is a difference between them which is called inversion of control. When using a library, the developer is in charge of the flow of the application and is allowed to choose when and where to call the library. On the other hand, when using a framework, the flow of the application is under the control of the framework, it provides some places where developers can insert their own code, but it calls the inserted code when needed (Wozniewicz, 2019).

Figure 5 below shows the difference between a framework and a library.

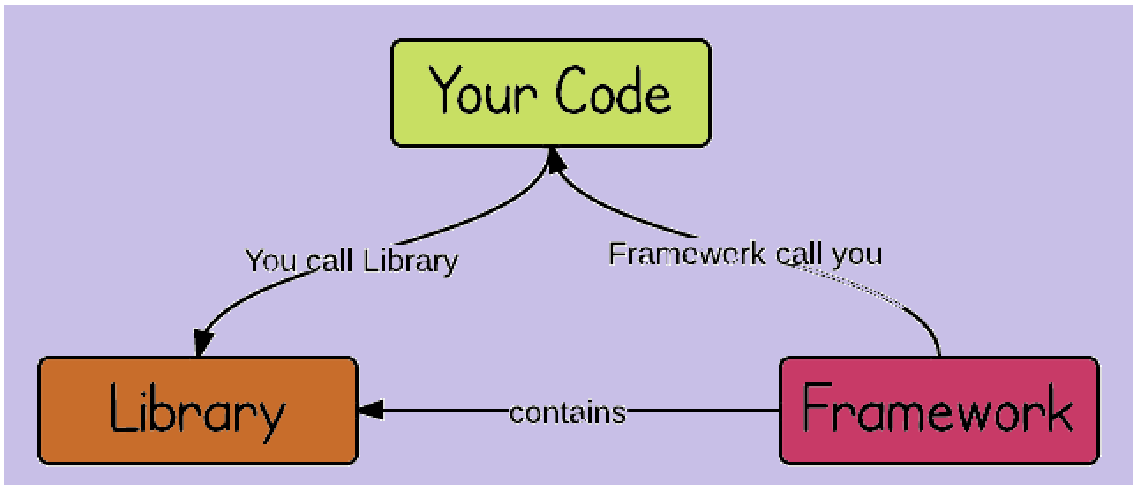


Figure 5. The difference between Framework and Library (Brennan, 2017).

Among the front-end frameworks and libraries, Angular and React are the most popular ones at the moment.

2.2.1 Angular (JS)

Angular JS is a JavaScript framework which was developed by Google in 2010. Although, in 2016, Angular JS was rewritten with TypeScript which is a superset of JavaScript, also, Angular JS was renamed as Angular. therefore, Angular is a TypeScript based JavaScript framework. Angular is a very popular front-end frame work and it is widely used by developers for creating single page application or short for SPA. Angular can be used used on many platforms, such as Web applications, desktop applications and mobile applications. One cadebase for many applications (bumblebee.co.za, 2019).

2.2.2 React

React is a JavaScript library which was developed by Facebook and released in 2013. It is used for building user interface. React devides the UI into a collection of components which makes it easier to adjust the interface and state constantly without reload the entire page. Similar to Angular, React is also used to build single-page applications (Copes, 2018).

There are many other frameworks and libraries which help develop web applications faster and easier. Each one of them has its own advantages and disadvantages, only choosing the one which suits a project the most.

3 HOW TO ASSURE THE QUALITY OF FRONT-END DEVELOPMENT

3.1 Introduction to Software Quality Assurance

Software Quality Assurance (SQA) is a process that assures all software engineering processes, methods and activities are monitored, it determines whether a software product meets specified requirements (softwaretestinghelp.com, 2019). Software Quality Assurance is very important in software development, it helps developers create high-quality software products that meet clients' requirements and expectations which leads to building trust and loyalty with clients (Rouse, 2019).

There are several defined standards which Software Quality Assurance needs to follow in order to establish and maintain a set of requirements for developing reliable software products. The standards include ISO 9000, CMMI (Capability Maturity Model Integrated) and TMM (Test Maturity Model). ISO 9000 stands for International Organization for Standardization, it was established in 1987 and it helps the organizations ensure quality to their customers and other stakeholders. CMMI is a process improvement approach developed specially for software process improvement. It is based on the process maturity framework and used as a general aid in business processes in the Software Industry. TMM is similar to CMMI, it assesses the maturity of processes in a testing environment (guru99.com, 2019).

According to Wikipedia, SQA encompasses the entire software development process, including requirements definition, software design, coding, code reviews, source code control, software configuration management, testing, release management and product integration.

3.1.1 Requirements definition

Requirements definition is a very crucial part of the software developments, it is the beginning of a project and it shows how the final product looks like or functions. Any inaccurate or incorrect definition of the requirements will lead to consequences of incompleted functionalities, production delays, wasted resources or dissatisfaction from customers (hyperthot.com, 2019).

3.1.2 Software design

As long as the requirements definition is clear, software design can be started by the software designers based on the requirements, it is a process that transforms the requirements into a form. During the designing phase, designers create a basic frame for the software product using different tools and make a list of functions and features that fit the needs.

3.1.3 Coding, Code reviews and source code control

After the completion of the software design, coding takes place with selected programming languages to build the designed software program. Additionally, code review is needed in order to keep the code quality. This can be done by one or more people from the same team except for the author of the code. Also, for the convenience of source code management, a scorce code control system is often used, it allows developers to track changes of the source code and retrieve any version of the original source code. There are many source code control systems available, such as GitHub, GitLab, Mercurial and Beanstalk, etc.

3.1.4 Software configuration management

Software configuration management is a process to systematically manage, organize, and control the changes in the documents, codes, and other entities during the Software Development Life Cycle (SDLC). Its primary goal is to increase productivity with minimal mistakes (guru99.com, 2019).

3.1.5 Testing

Software testing is a process of checking and detecting errors and defects in the software product. Testing can be conducted manually or by using automated testing tools. Testing is a significant part of software development as it identifies issues, detects programming errors in the software and helps developers fix them quickly.

3.1.6 Release management

Based on the definition in Wikipedia, Release management is the process of managing, planning, scheduling and controlling a software build through different stages and environments; including testing and deploying software releases.

3.1.7 Product integration

Product integration is the last step of the software development, the purpose is to assemble the product from the product components, ensure that the product, as integrated, behaves as expected (i.e., possesses the required functionality and quality attributes), and deliver the product (wibas.com, 2019).

In conclusion, all aspects of software development listed above are very important and they are also the parts of SQA, therefore, necessary standards need to be followed in order to produce a high-quality product.

However, this thesis mainly focuses on one of these aspects – testing, because testing is a very important part of Software Quality Assurance, it is the key step that identifies issues during the software development and help developers fix them in the early stage. In addition, software programs with bugs could be very dangerous, it can cause monetary and human loss, there are plenty of such examples in the history, for instance, Starbucks used to be forced to close nearly 60 percent of stores in the U.S and Canada because of the failure in the POS (Point of Sale) system that the transaction could not be processed and they had to serve coffee for free. In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3 billion - pound debt sale (guru99.com, 2019). In October 2018, Lion Air Flight 610 with Boeing 737 Max aircraft crashed due to a series of system failures which killed 189 people. Another similar tragedy happened five months later, Ethiopian Airlines Flight 302 crashed six minutes after takeoff and killed 157 people, it was the same Boeing 737 Max aircraft.

The examples listed above suggested how severe the software bugs could be and the consequences they caused. In order to identify these bugs and fix them to avoid any tragedies, testing is the step that developers need to take seriously.

Front-end development product is different from the kind software programs that used in the examples and it may not cause life and death directly even in the worst case, however, the point can be seen. Therefore, testing needs to be implemented to make sure the front-end product functions as expected based on the requirements definition, in the meanwhile, the quality of the product can be assured effectively.

3.2 Front-end testing

Front-end testing can be seen as a process of testing different parts of a website by using different testing tools to make sure that the entire website behaves as expected. Front-end testing includes unit testing, UI testing, functional testing, cross-browser testing, performance testing, accessibility testing, etc.

Many developers might get confused about the concept of front-end testing as it usually happens in the backend and embedded software development. Nonetheless, front-end testing not only exits but also is a very important part of the development nowadays. Front-end development has become more and more complex, most of the websites we visit everyday have much more features and functions than before, also, there are some very nice effects on the UI of the websites, especially some of those animated effects, they are very visually attractive and enjoyable to see. All of these great features come from the constructed code, therefore, any error in the code or some conflicts between the code can easily break the website. However, this could be avoided if the testings have been implemented during front-end development before the release of the website, this is why the front-end testing is needed in order to assure the quality of the code and the final products.

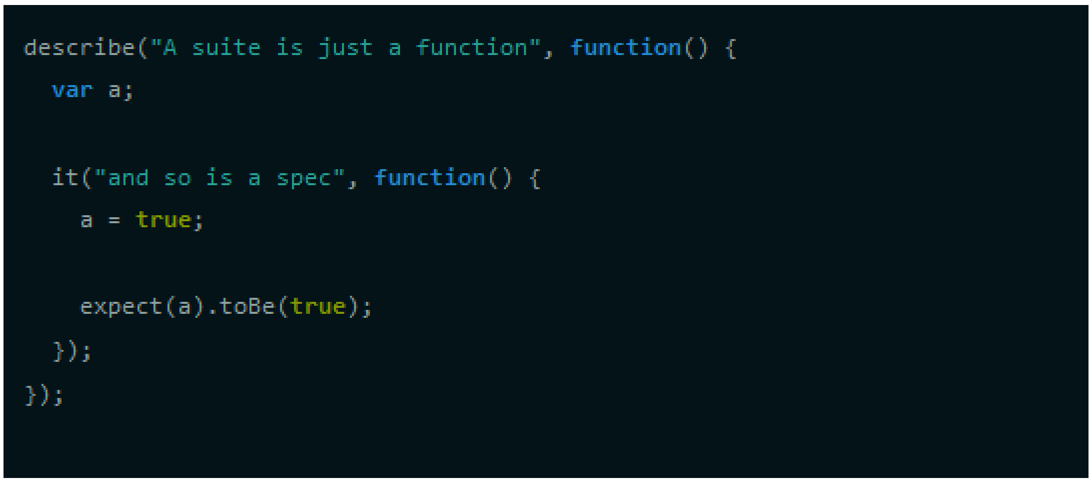
3.3 What to test and tools for the tests

As mentioned above, there are many different tests in front-end development, however, not every single test has to be performed for a website, it all depends on the specific project and situation. Below are some front-end testing types that are very important and most performed during front-end development. In addition, some popular tools for these tests are also introduced and they are all automated tools, although, many other alternatives are available for the same type of test. The tools listed below are the most commonly used.

3.3.1 Unit test

Unit test is the lowest level of test among all the other front-end tests, it tests the code independently. For example, testing a function of calculation of two numbers to make sure the calculation is always correct. The purpose of unit test is to ensure the smallest bits of the code behave as expected. Usually, most of the tests that are being implemented for an application are unit tests. Additionally, there are also some benefits from it, for instance, Unit tests help developers identify software bugs in the early stage, also, unit tests foster simplicity by forcing developers to write flexible and configurable code (Sedlock, 2017).

Jasmine is a great tool for unit test, it is an open-source and a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks or require a DOM which is an interface to a web page and it allows the page’s content and structure to be manipulated by programs. Jasmine has a clean and obvious syntax (jasmine.github.io, 2019).



```
describe("A suite is just a function", function() {
  var a;

  it("and so is a spec", function() {
    a = true;

    expect(a).toBe(true);
  });
});
```

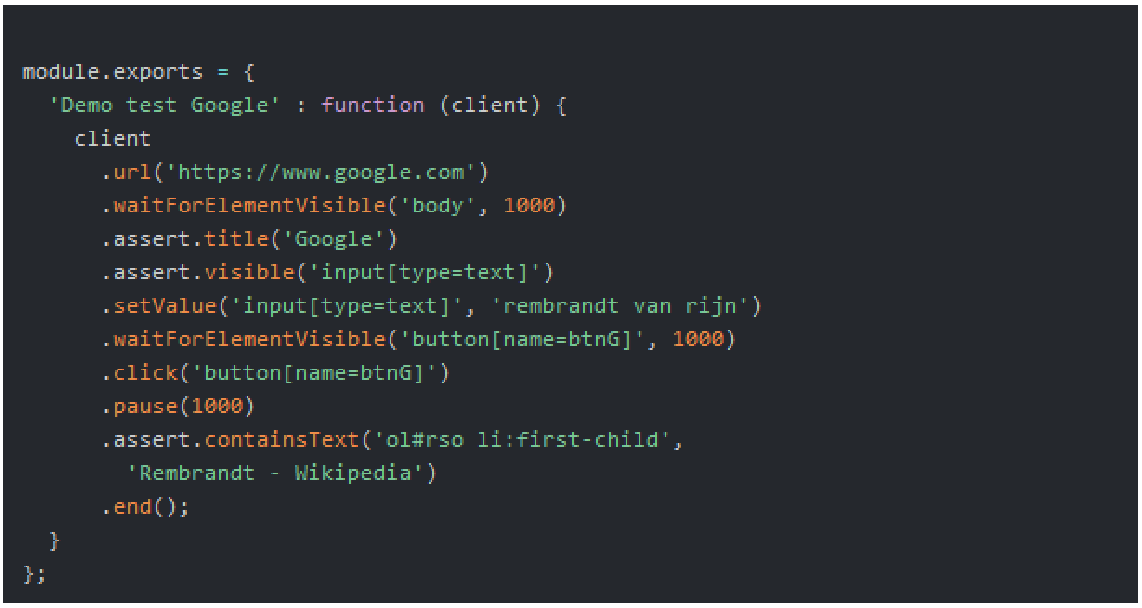
Figure 6. Jasmine sample code (jasmine.github.io, 2019).

There are some other unit-test tools such as Mocha, AVA, Tape, etc.

3.3.2 Functional test

Functional test is also referred to E2E(End-to-End) test or Browser test sometimes, it tests the complete functionality of an application. For example, in a web application, functional test is implemented by using some tools to automate a browser which is used to click around on the page like a real user does in the real world to test the application (codeutopia.net, 2019).

However, unlike unit test which tests an individual function and there can be many unit tests in an application, it is a better practice to have less or only a small amount of functional tests, because functional tests are not as easy as unit tests to write and can be very difficult to maintain due to the high complexity. Additionally, since the functional tests simulate the real user interaction on a web page, therefore, they run very slow compare to the unit tests (codeutopia.net, 2019).



```
module.exports = {
  'Demo test Google' : function (client) {
    client
      .url('https://www.google.com')
      .waitForElementVisible('body', 1000)
      .assert.title('Google')
      .assert.visible('input[type=text]')
      .setValue('input[type=text]', 'rembrandt van rijn')
      .waitForElementVisible('button[name=btnG]', 1000)
      .click('button[name=btnG]')
      .pause(1000)
      .assert.containsText('ol#rso li:first-child',
        'Rembrandt - Wikipedia')
      .end();
  }
};
```

Figure 7. Functional test (nightwatchjs.org, 2019).

The sample code in Figure 7 is from Nightwatch official website. It is an automated testing and continuous integration framework based on Node.js and Selenium WebDriver. It is an End-to-End testing solution for browser based applications and websites (nightwatchjs.org, 2019).

The code above shows the simulation that navigates to google.com and searches for "rembrandt van rijn", then verifies if the term first result is the Wikipedia page of Rembrandt.

Some other functional testing tools such as Cypress and TestCafe also serve the same purpose with more or less different features.

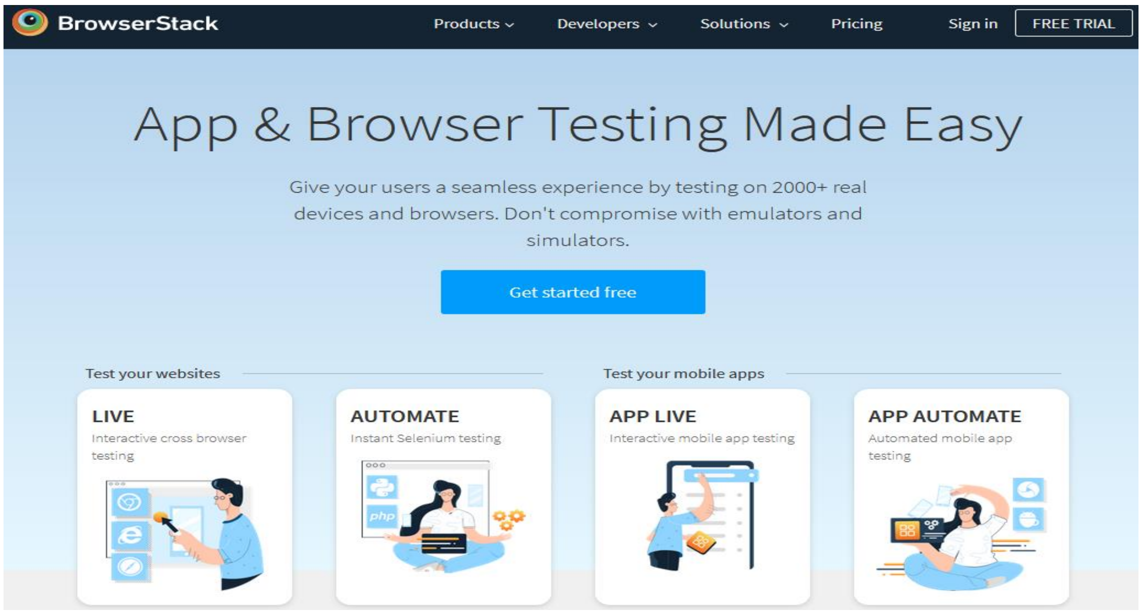
3.3.3 Cross Browser test

Many people have come across situations that when they are trying to open a website on a browser, instead of displaying the website, there is a note showing on the page “The browser you are using is not supported by this website, please switch to another browser”. Even though the website can be opend on this browser, some content just cannot be displayed properly, such as overlay text in some area, or losing responsiveness, in some cases, a specific function just does not work as intended.

All of these situations happen because of the lack of cross-browser test, so a website can be displayed normally only on the browser which is used during the development. Cross-browser testing tests the same website in different browsers to make sure it works as expected in all of them, it is a process to verify the compatibility between the website and different browsers.

Cross-browser test is a repeated task which runs the same set of test cases multipile times on different browsers, hence, an automated tool for this test will be more effective on time and cost (softwaretestinghelp.com, 2019).

BrowserStack is an automated cross-browser testing tool, it is a web based testing platform and it provides four primary services - Live, App Live, Automate and App Automate. With BrowserStack, an application can be tested on more than 2000 real devices and browsers on different operating system (browserstack.com, 2019).



App & Browser Testing Made Easy

Give your users a seamless experience by testing on 2000+ real devices and browsers. Don't compromise with emulators and simulators.

Test your websites

LIVE — Interactive cross browser testing

AUTOMATE — Instant Selenium testing

Test your mobile apps

APP LIVE — Interactive mobile app testing

APP AUTOMATE — Automated mobile app testing

Figure 8. BrowserStack (browserstack.com, 2019).

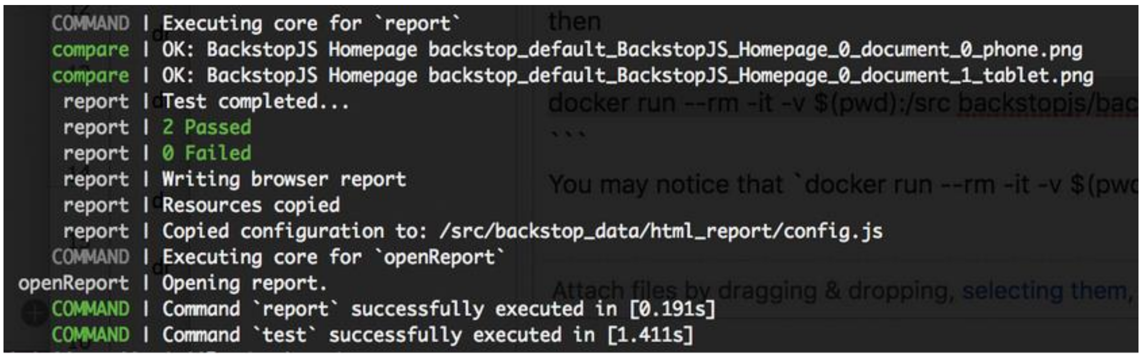
In addition to BrowserStack, there are many other cross-browser testing tools which have some extra features such as visual testing and debugging included, experitest and Ghost Inspector are among these multifunctional testing tools.

3.3.4 Visual regression test

Visual regression test is a unique type of test to front-end development, instead of testing the code, it compares the rendered result of the code – the user interface. This is typically conducted by comparing screen shots taken within a browser which runs on server. Then tools for the comparison detect the differences between the two screen shots. Based on the spotted differences, developers can quickly fix them accordingly (Sedlock, 2017).

However, the best practice for visual regression test is to perform it in the end of development, because the UI of the website keeps changing dramatically during the development, therefore, it is more efficient and beneficial to save this test for the last (Sedlock, 2017).

To perform visual regression test, an automated testing tool such as BackstopJS can be used for this purpose. It is an open-source tool that tests the responsive web UI by comparing DOM screenshots over time (github.com/garris/BackstopJS, 2019).



```
    COMMAND | Executing core for `report`
    compare | OK: BackstopJS Homepage backstop_default_BackstopJS_Homepage_0_document_0_phone.png
    compare | OK: BackstopJS Homepage backstop_default_BackstopJS_Homepage_0_document_1_tablet.png
     report |Test completed...
     report | 2 Passed
     report | 0 Failed
     report | Writing browser report
     report | Resources copied
     report | Copied configuration to: /src/backstop_data/html_report/config.js
    COMMAND | Executing core for `openReport`
 openReport | Opening report.
    COMMAND | Command `report` successfully executed in [0.191s]
    COMMAND | Command `test` successfully executed in [1.411s]
```

Figure 9. BackstopJs (github.com, 2019).

Alternative testing tools such as PhantomCSS, Percy and gremlins.js are also available for visual regression test.

3.3.5 Accessibility test

Another important test is accessibility test, the test was created to make sure that anyone under different circumstances can fully access and interact with a website. According to a WHO report, there are about 1.3 billion people live with some kind of vision impairment, including 36 million people with blindness. People with visual disabilities are not able to interact with a website properly. In addition to the people with visual disabilities, there are people with motor issues and hearing impairments as well, therefore, it is very important that a website can be accessed by people with different kinds of disabilities (Hoffmann, 2018).

However, in reality, accessibility is often ignored because companies or developers assume that people with disabilities are only minority and will not affect the amount of visitors to their websites significantly. Nonetheless, based on the WHO report, there are many people with different form of disabilities, hence, accessibility should not be ignored or skipped, it is important to assure that anyone can access a website (Hoffmann, 2018).

In order to make a website accessible, some rules need to be followed. There is a Accessibility Checklist in WCAG(Web Content Accessibility Guidelines) which is a guideline that provides detailed information on how to implement accessibility on a website. Also, there are lots of different tools for testing the accessibility of a website, such as axe. Axe is an automated accessibility testing tool and the axe-core library is open source, it can be used with different testing frameworks, tools and environments.

For instance, it can be run in functional tests, browser plugins or in the development of the application. The easiest way is to use it as a browser plugin which is very straight forward (Jelisejevs, 2017).

There are many other tools available for accessibility testing, such as pa11y, google lighthouse, accessibilityjs, tenon.io, etc.

3.4 Conclusion

Front-end testing has become more and more important as the web development gets more and more complex. In order to ensure a website operates properly, some tests have to be performed before the launch. However, not all kinds of tests are needed for the same website, only the essential tests might be enough for the assurance depends on the situation. Also, there are many different tools for each test, select the one that easily serves the purpose.

4 TESTING AS A PART OF DEVELOPMENT PROCESS

4.1 Testing is a process

As introduced in previous chapter, front-end testing is a process of testing different parts of a website to make sure it always works as intended. Therefore, it can be taken as a part of development process. However, some people would argue that testing is just a phase in the software development instead of being a process. In reality, some organizations do see testing as a phase of development. Unfortunately, the result is not always as expected, because the testing as a phase is performed in the end of development cycle before launching the product and all kinds of tests are gathered together. Therefore, all issues and bugs have to be fixed at once which will take significant amount of time, this is naturally affected by the approaching deadline and it usually leads to the result of testing not being fully conducted. Final products without being fully tested will always be carrying some potential risks regarding their performance, security, and functionalities (cigniti.com, 2019).

The solution to the issue described above is actually very simple -- take testing as part of development process instead of taking it as a phase. While planning the product development phases, identify the types of tests that are needed for each phase and then allocate specific time and resources for performing the tests. Implementing testings during each development phase will identify bugs and issues in the early stage and can be much more time and cost effective. Also, it will improve the quality of the product such as stability and reliability at the same time (cigniti.com, 2019).

In summary, testing is a process which is conducted in stages, focusing on different parts of the software development, from the internal logic of the program to the user experience. It is very important to have testing integrated into software development as it is also the key to quality assurance of software products (ag-prime.com, 2019).

4.2 Continuous Integration in the process

4.2.1 Definition of Continuous Integration

We have learned from last section that testing is a process instead of a phase in software development. However, when it comes to the details about how the whole process works in order to get the best result out of it, another concept will have to be introduced – Continuous Integration, also short for CI.

According to a software company ThoughtWorks, “Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.” (thoughtworks.com, 2019).

4.2.2 Practices of Continuous Integration

Continuous Integration has become a popular and efficient workflow in software development industry in recent years. Figure 10 below shows how Continuous Integration workflow actually works.

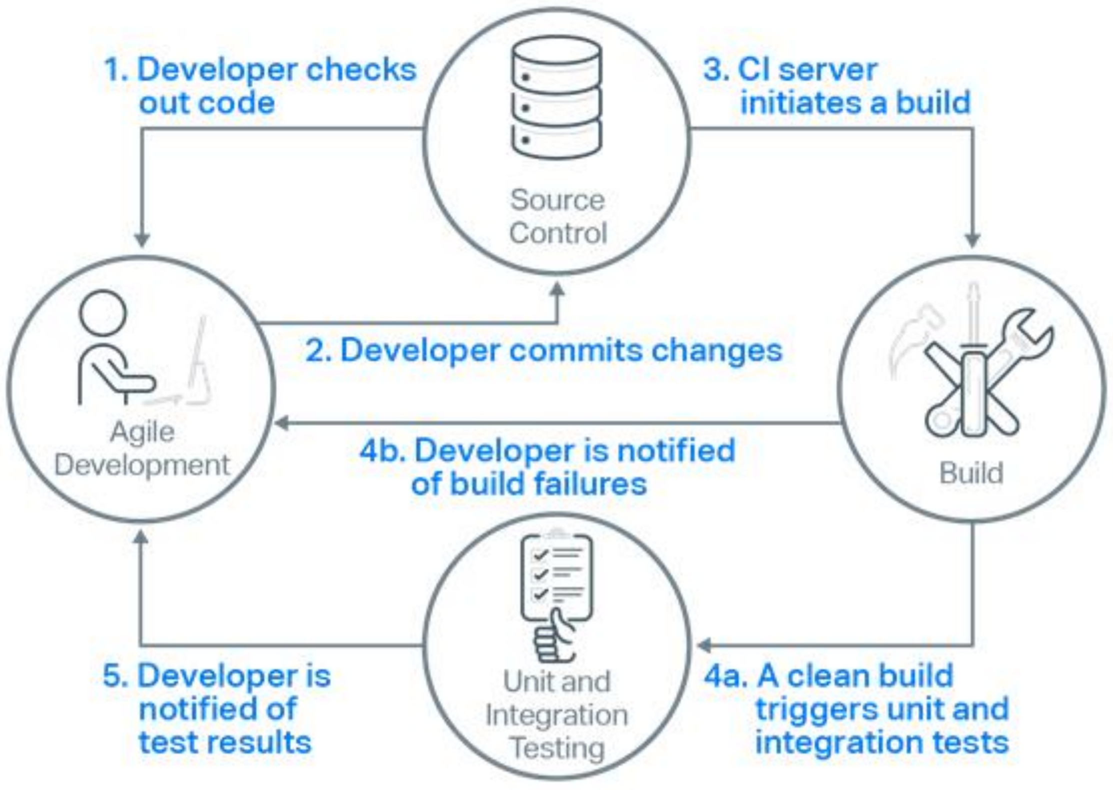

1. Developer checks out code
Source Control
3. CI server initiates a build
2. Developer commits changes
Agile Development
Build
4b. Developer is notified of build failures
5. Developer is notified of test results
Unit and Integration Testing
4a. A clean build triggers unit and integration tests

Figure 10. Continuous Integration Workflow (microfocus.com, 2019).

It is worth notice that this is a Continuous Integration workflow in general situation, therefore, it is not necessary to implement every activity for each project all the time, unless there are benefits for taking them into implementation. It all depends case by case in terms of efficiency and smoothness.

As shown in Figure 10, first of all, developers check out the code and make changes, then push it into the source control repository. The CI server will detect the changes since it always monitors the source control repository, then a build of the relevant sources is triggered. If the build is successful, it will trigger the unit and integration testing, The developers get notified about the test result after the testings. However, if the build is failed, the CI server will send a notification to the developers regarding the build failure so that the developers can restart the process to fix the build errors ((microfocus.com 2019).

Implementing Continuous Integration requires some necessary components such as a source control repository which allows for both local and remote source control; an automated build agent that triggers build when new code is checked into the repository; automated testing tools for the build tests (Ram, 2018).

In order to make the entire Continuous Integration workflow work, corresponding tools for each component are needed. For example, GitHub is probably the most used version control repository hosting service, there are also some other alternatives such as BitBucket and Stash, also, versioning tools like Git, Mercurial and Subversion can be used for the version control. For automated build agent, there are many options to select such as Jenkins, TeamCity, Bamboo CI, Circle CI, Travis CI, etc. Jenkins is the leading open source automation server. Some automated testing tools have been introduced in previous chapter, however, there can be more or less testing tools depending on the specific project, all in all, the selected automated testing tools need to be compatible with each other and can be integrated into the CI workflow smoothly.

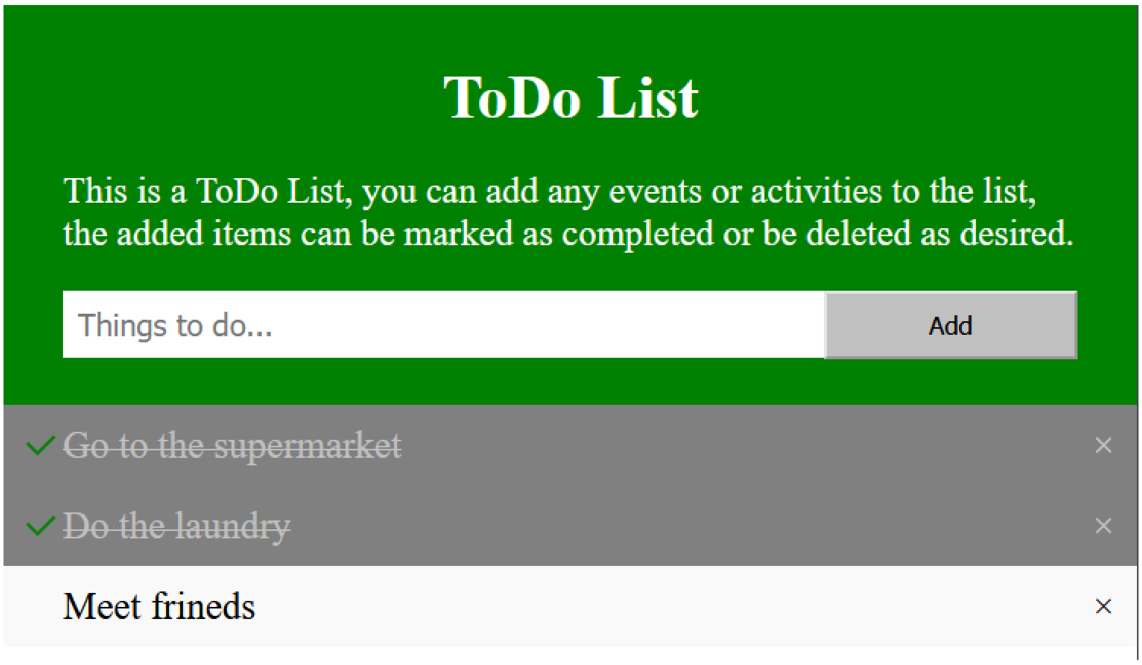
4.2.3 Conclusion

Continuous Integration is very efficient in software development, there are many benefits by implementing a CI workflow, for instance, via CI, developers can find issues early and fix them quickly. It takes less time to resolve the integration issues since each change is integrated as required. in addition, There is always a working version of the product that contains the latest changes (microfocus.com, 2019).

There are many tools for implementing Continuous Integration, however, the compatibility between the automated testing tools and the CI system has to be taken into consideration in order to make the whole workflow function efficiently.

5 CASE STUDY: TESTING A JS TODO-LIST

In this chapter, a web application - a JavaScript ToDo-List - will be tested. This web application allows users to add events to it and mark them as they are completed by clicking on the event or delete them as wish by clicking on the “x” mark. The entered event in the input field can be added to the list either by clicking the “Add” button or by pressing the Enter key on the keyboard.


ToDo List

This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.

Figure 11. The JavaScript ToDo List.

5.1 Making the testing plan

The best practice to test a software application is to make a testing plan before the actual tests. With a testing plan, the entire process of testing can be well organized and the issues found during the tests can be clearly showed and easier to fix, additionally, developpers are able to gain more confidence in deploying the project.

The testing plan for the ToDo List application is shown below:

Operating System platform: Windows 10

Browser: Google Chrome, Firefox

Testing scope: Unit test, Functional test, Cross-browser test, Visual regression test and accessibility test

Testing tools: Jasmine for unit test, TestCafe for functional test, BrowserStack for cross-browser test, BackstopJS for visual regression test and axe for accessibility test.

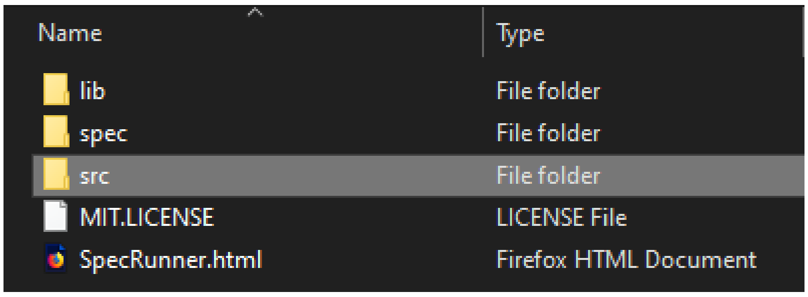
Continuous Integration platform: Travis CI.

Budget: free testing tools and non-free testing tools with free trial.

5.2 Implementing the tests

5.2.1 Unit test

In unit test, Jasmine was used for the implementation. After downloading Jasmine standalone file from official website, it was unzipped to the root folder of the testing application. There were three sub foulders and two files in Jasmine folder as shown below:

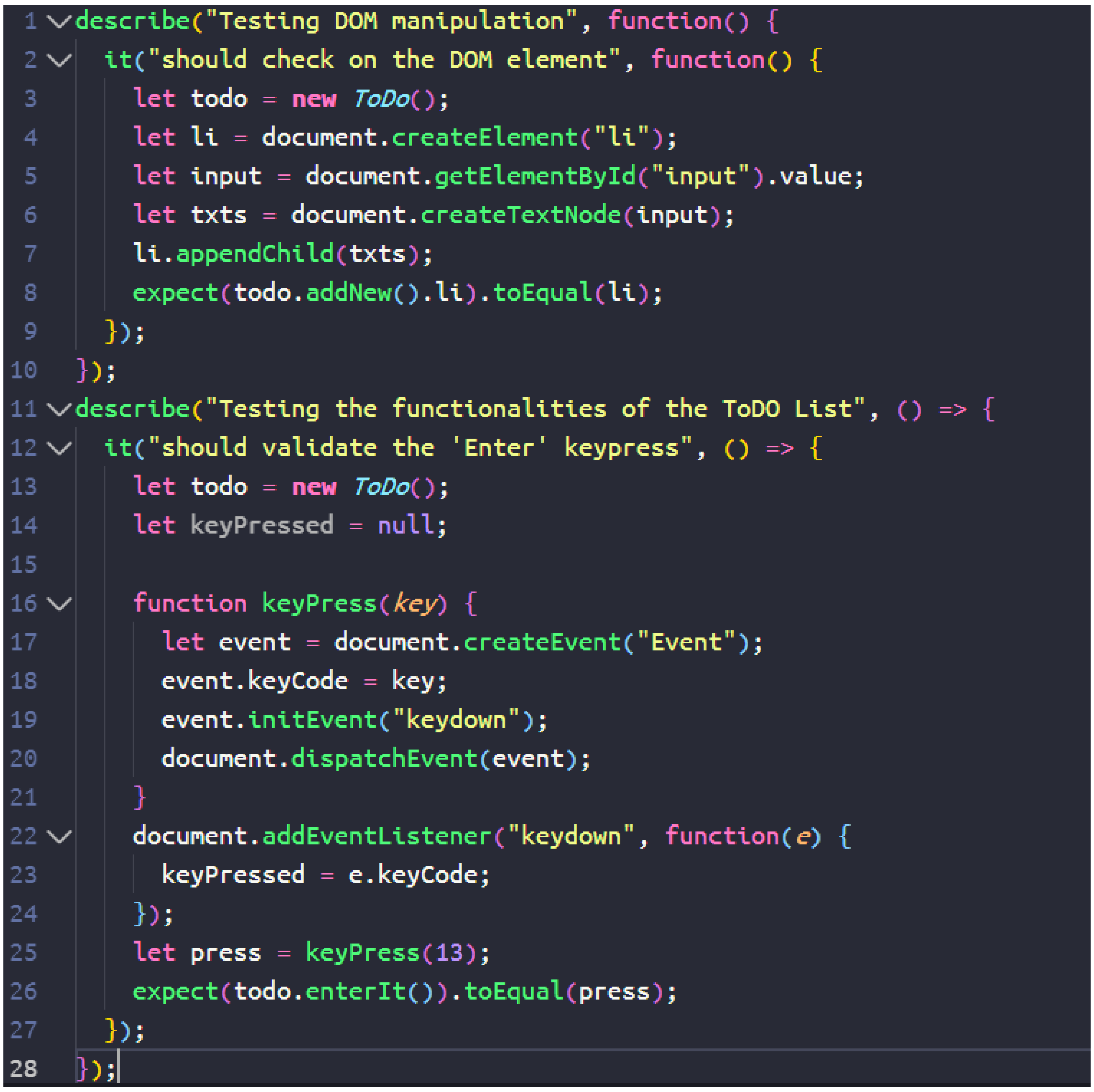


Name	Type
lib	File folder
spec	File folder
src	File folder
MIT.LICENSE	LICENSE File
SpecRunner.html	Firefox HTML Document

Figure 12. Jasmine folder content.

The src folder can be removed as it does not affect the usage of Jasmine. The most important files are SpecRunner.html and testing files in spec the folder. SpecRunner.html file will show testing results in a browser after running a test.

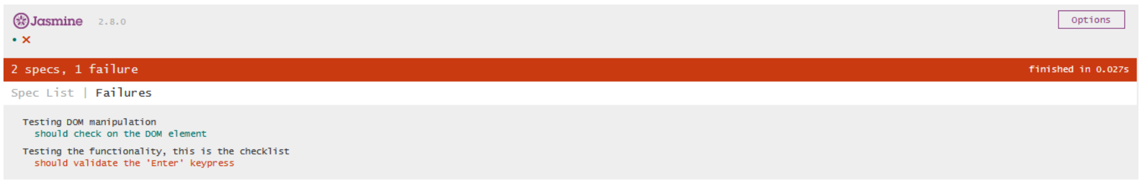
The unit test code was written into a .js file which was in the spec folder. There were two test suits for the ToDo List application, one was checking the DOM element and another one was testing Enter keypress when adding an event to the list.



```
describe("Testing DOM manipulation", function() {
  it("should check on the DOM element", function() {
    let todo = new ToDo();
    let li = document.createElement("li");
    let input = document.getElementById("input").value;
    let txts = document.createTextNode(input);
    li.appendChild(txts);
    expect(todo.addNew().li).toEqual(li);
  });
});
describe("Testing the functionalities of the ToDO List", () => {
  it("should validate the 'Enter' keypress", () => {
    let todo = new ToDo();
    let keyPressed = null;

    function keyPress(key) {
      let event = document.createEvent("Event");
      event.keyCode = key;
      event.initEvent("keydown");
      document.dispatchEvent(event);
    }
    document.addEventListener("keydown", function(e) {
      keyPressed = e.keyCode;
    });
    let press = keyPress(13);
    expect(todo.enterIt()).toEqual(press);
  });
});
```

Figure 13. Jasmine unit test.

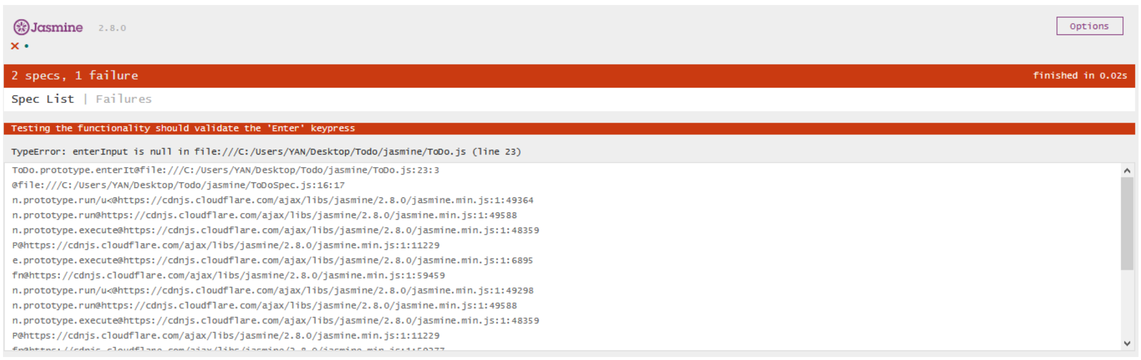
The testing result was displaying as follow after running the test. The first test was passed and the second one was failed.



```
2 specs, 1 failure                                    finished in 0.027s

Testing DOM manipulation
  should check on the DOM element
Testing the functionality, this is the checklist
  should validate the 'Enter' keypress
```

Figure 14. Jasmine testing result.

Detailed error information was given below for the failed test, it was a TypeError with enterInput function being null.



```
2 specs, 1 failure                                                    finished in 0.02s

Testing the functionality should validate the 'Enter' keypress

TypeError: enterInput is null in file:///C:/Users/YAN/Desktop/Todo/jasmine/ToDo.js (line 23)
ToDo.prototype.enterIt@file:///C:/Users/YAN/Desktop/Todo/jasmine/ToDo.js:23:3
@file:///C:/Users/YAN/Desktop/Todo/jasmine/ToDoSpec.js:16:17
n.prototype.run/u<@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:49364
n.prototype.run@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:49588
n.prototype.execute@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:48359
P@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:11229
e.prototype.execute@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:6895
fn@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:59459
n.prototype.run/u<@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:49298
n.prototype.run@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:49588
n.prototype.execute@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:48359
P@https://cdnjs.cloudflare.com/ajax/libs/jasmine/2.8.0/jasmine.min.js:1:11229
```

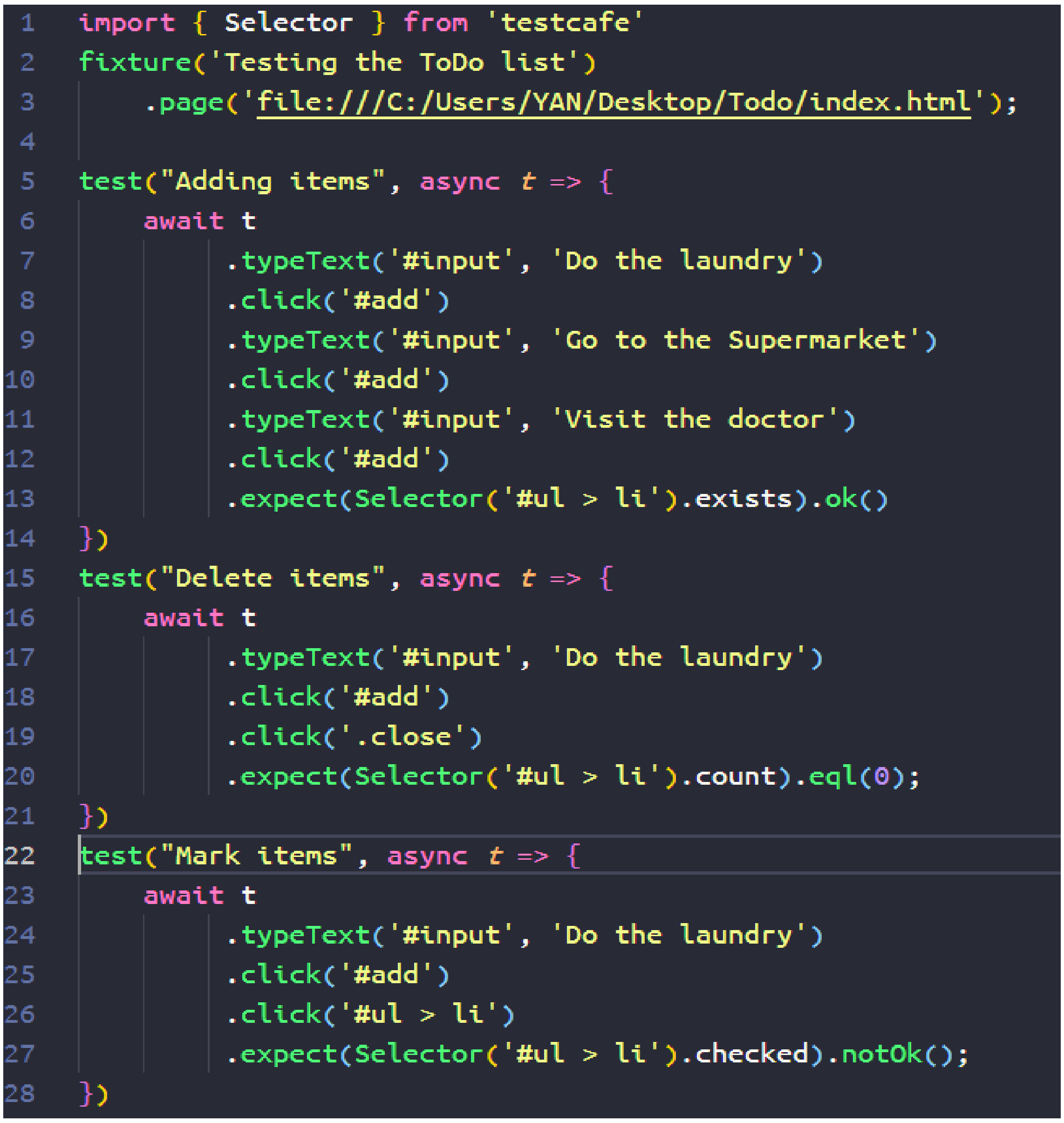
Figure 15. Error information for the failed test.

In order to pass the test, the fix was needed based on the error information. There can be as many unit tests as desired for one application, however, it is not necessary to unit test all the elements and functions in the application, it is sensible to focus on the important ones. Also, some code structures can be very complex to unit test either because the DOM elements are empty until the actual interaction happens or the code is not testable in some cases.

5.2.2 Functional test

TestCafe was used for functional testing, it is a free and open-source automate end-to-end web testing tool.

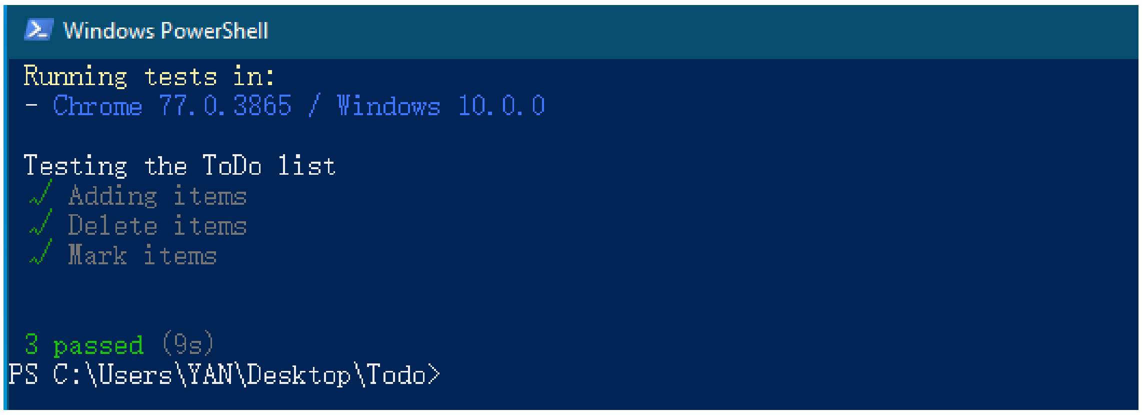
TestCafe was installed via command line `npm install -g testcafe`, after which a test file with .js extension was added to the root folder of the application, in this test, the file name was called testCafe.js. The actual test code for the ToDo List is shown below:



```
import { Selector } from 'testcafe'
fixture('Testing the ToDo list')
    .page('file:///C:/Users/YAN/Desktop/Todo/index.html');

test("Adding items", async t => {
    await t
        .typeText('#input', 'Do the laundry')
        .click('#add')
        .typeText('#input', 'Go to the Supermarket')
        .click('#add')
        .typeText('#input', 'Visit the doctor')
        .click('#add')
        .expect(Selector('#ul > li').exists).ok()
})
test("Delete items", async t => {
    await t
        .typeText('#input', 'Do the laundry')
        .click('#add')
        .click('.close')
        .expect(Selector('#ul > li').count).eql(0);
})
test("Mark items", async t => {
    await t
        .typeText('#input', 'Do the laundry')
        .click('#add')
        .click('#ul > li')
        .expect(Selector('#ul > li').checked).notOk();
})
```

Figure 16. Functional testing code in TestCafe.

The test was run via command `testcafe chrome testCafe.js`, after which the chrome browser popped up automatically and demonstrated the whole process like a real user interaction with the application. Eventually, the result was shown as follow:



```
Windows PowerShell
Running tests in:
- Chrome 77.0.3865 / Windows 10.0.0

Testing the ToDo list
 √ Adding items
 √ Delete items
 √ Mark items


 3 passed (9s)
PS C:\Users\YAN\Desktop\Todo>
```

Figure 17. Functional test result in TestCafe.

5.2.3 Cross-browser testing

An account for BrowserStack were needed in order to use it to perform cross-browser testing. After the registration, a Username and an Access Key were found in the profile settings, they were used for establishing connection between the application and their remote server. Also, there was a Local Folder URL in the settings and it was for local testing, this URL was a formatted local folder.

There were different options to build the local testing connections such as via command line, using the binary application or a browser extension. The easiest way was to use BrowserStack extension. After the installation on Chrome, the server was activated, application folder was chosen and it was formatted to the Local Folder URL(johnsmith1719.browserstack.com) in the profile settings. Then a testing page was redirected to and the Local Folder URL was entered in the input field. In the main section of the testing page, There were numerous of mobile devices and different operating system such as Windows and MacOS with all different versions available to choose. Within each version of operating system, there were many different mainstream browsers such as Chrome, Firefox, Opera, Edge, IE and Safari with different versions as well. The devices and browsers could be selected as desired.

However, since this was a free-trial version of BrowserStack, there were some limitations of the usage, such as, there were only 100 screenshots available for cross-browser compatibility testing and 25 browsers could be selected at a time. Therefore, in this testing, 25 most used browsers on different platforms were selected and the screenshots were generated below:

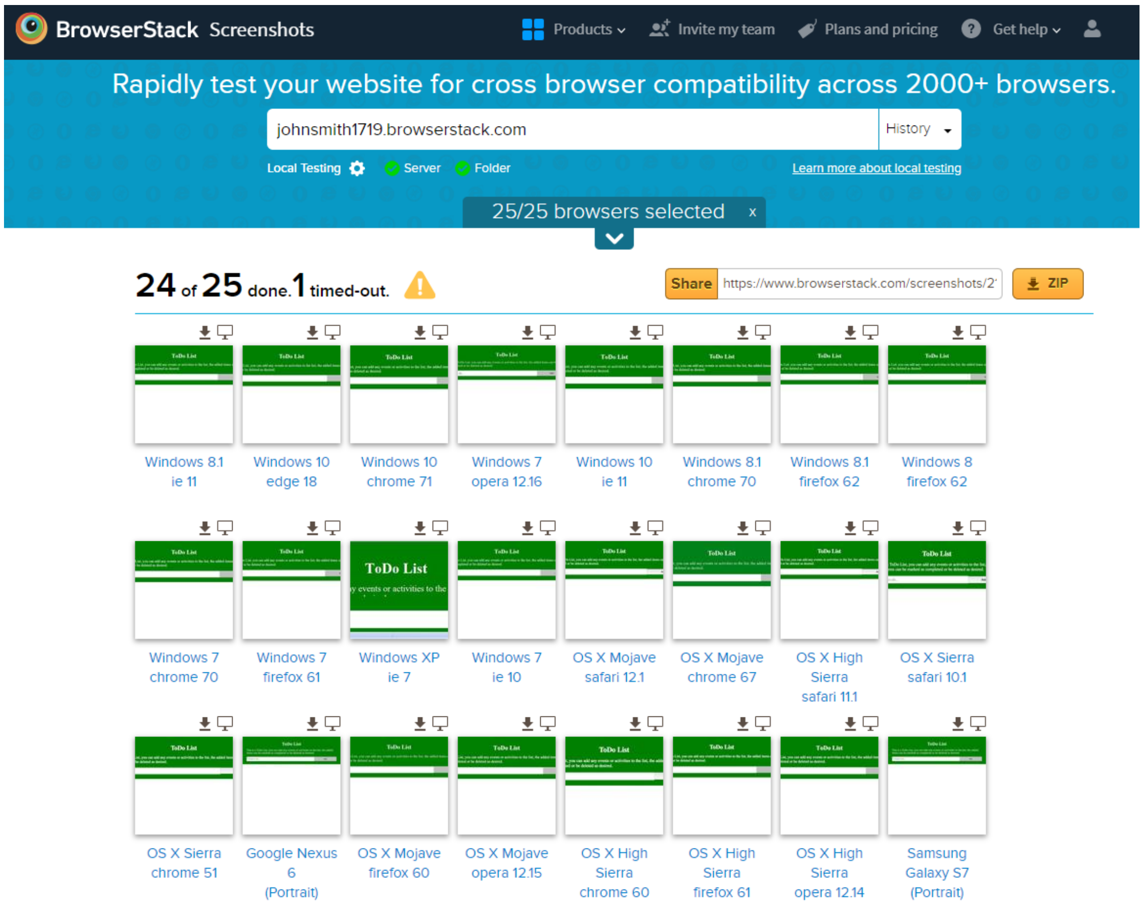
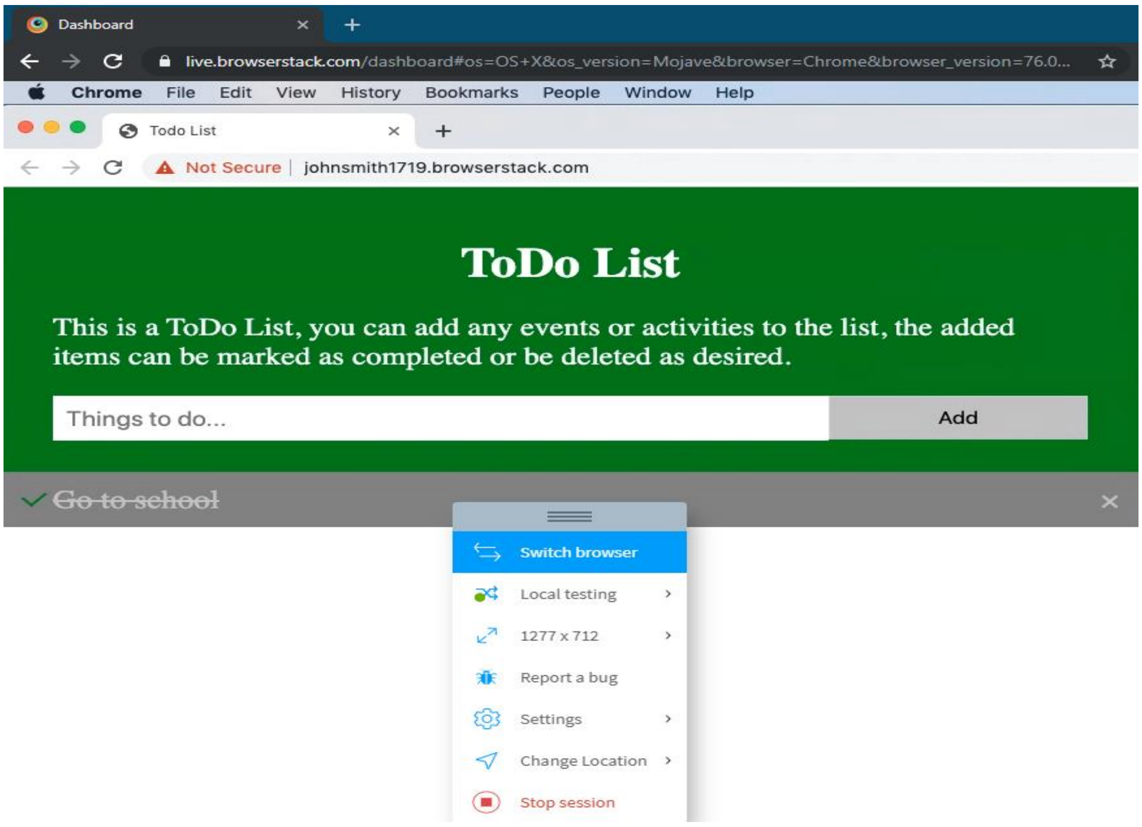


Figure 18. Screenshots of cross-browser testing with BrowserStack.

As the result shown above, there were 24 screenshots of the chosen browsers generated successfully and 1 screenshot timed-out, this was due to BrowserStack system setup, there was a two-minute processing time limit for a screenshot to be generated to prevent the queue from stalling. Nonetheless, among the generated screenshots of those browsers, each one of them could be clicked to view a larger image and spot the issues easier. In this test, most of the browsers in different platforms were working fine except for IE7 on Windows XP. After which the spotted issue could be fixed accordingly.

BrowserStack provides Live cross-browser testing which allows users to interact with their applications directly within different browsers, this is even easier to find the issues. Although, it can be time-consuming if there are many browsers need to be tested. With Live testing, any platform and browser could be selected, then the application would be tested in it. There is a panel menu in the testing screen and it allows users to switch browsers, change screen resolution and report bugs, etc.


Dashboard
live.browserstack.com/dashboard#os=OS+X&os_version=Mojave&browser=Chrome&browser_version=76.0...
Chrome File Edit View History Bookmarks People Window Help
Todo List
Not Secure | johnsmith1719.browserstack.com
Things to do...
Add
Switch browser
Local testing
1277 x 712
Report a bug
Settings
Change Location
Stop session

ToDo List

This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.

~~Go to school~~

Figure 19. Live testing with BrowserStack.

5.2.4 Visual regression testing

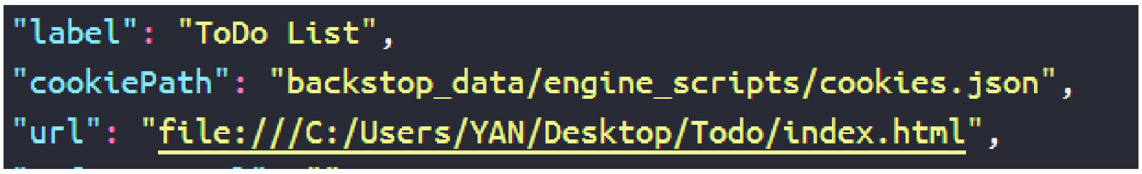
BackstopJS was used for visual regresstion test for the application. The tool was installed by running command:

```
npm install -g backstopjs
```

In the root folder of the application, a configuration file was created by running command:

```
backstop init
```

There were lots of code in the configuration file, however, the actual tests were set up in the “scenarios” section only like below:



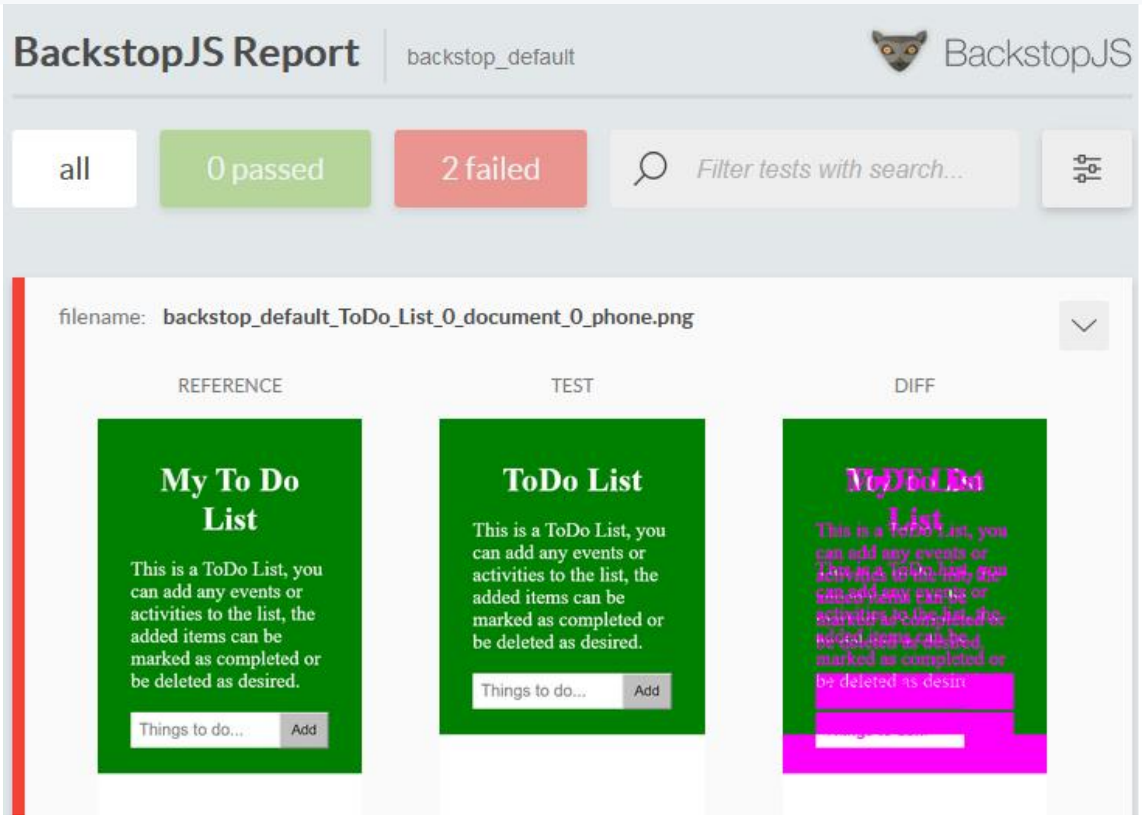
```
"label": "ToDo List",
"cookiePath": "backstop_data/engine_scripts/cookies.json",
"url": "file:///C:/Users/YAN/Desktop/Todo/index.html",
```

Figure 20. Configuration file of BackstopJS.

The test was run via command:

```
backstop test
```

The result was shown as follow:


BackstopJS Report | backstop_default | BackstopJS

all | 0 passed | 2 failed | Filter tests with search...

filename: backstop_default_ToDo_List_0_document_0_phone.png

REFERENCE

My To Do List

This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.

Things to do... | Add

TEST

ToDo List

This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.

Things to do... | Add

DIFF

Figure 21. Visual regression testing with BackstopJS(Failed).

The figure above was showing some UI diffenences, hence, the test was failed, after some fixes, the test was run again and eventually passed.

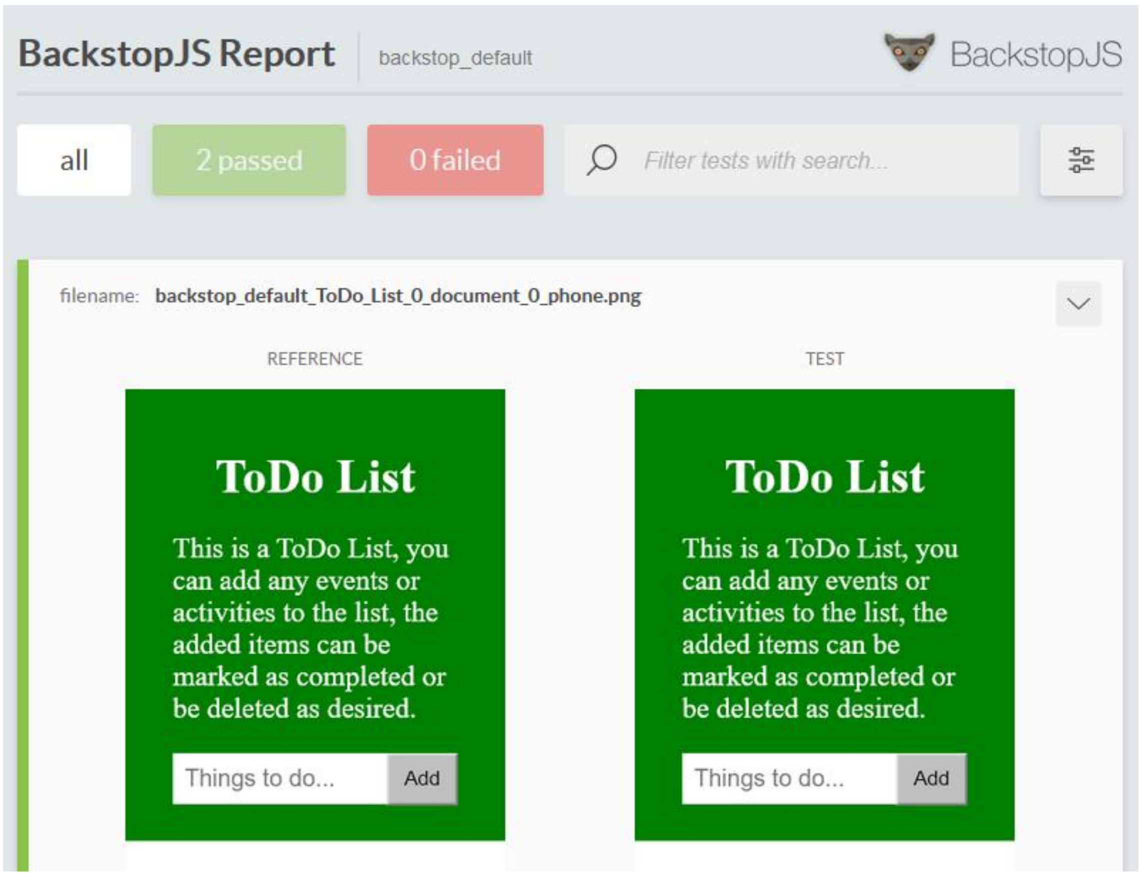
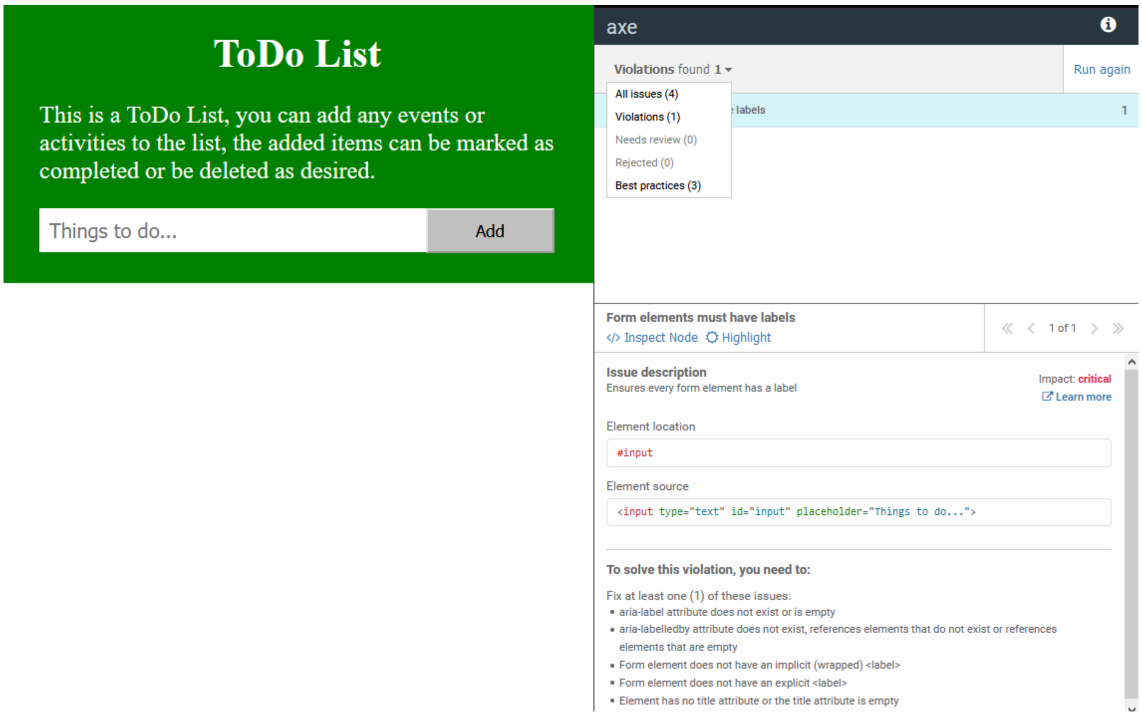

BackstopJS Report
backstop_default
BackstopJS
all
2 passed
0 failed
Filter tests with search...
filename: backstop_default_ToDo_List_0_document_0_phone.png
REFERENCE
TEST
ToDo List
This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.
Things to do...
Add
ToDo List
This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.
Things to do...
Add

Figure 22. Visual regression test with BackstopJS(Passed).

5.2.5 Accessibility test

Accessibility test was performed with axe which is an automated testing tool. In this test. Axe was used as a browser plugin as it was easier and faster to spot the issue. The axe plugin was installed in Firefox. In the ToDo List web page, Firefox Inspector was opened by right-click on the page and selected “Inspenct Elements”, after which navigated to the axe tab and clicked it, the axe interface appeared, then started the test by clicking on “ANALYZE” button, the result then was showing as follow:


ToDo List

This is a ToDo List, you can add any events or activities to the list, the added items can be marked as completed or be deleted as desired.

Things to do... | Add

axe

Violations found 1 | Run again

All issues (4)
Violations (1)
Needs review (0)
Rejected (0)
Best practices (3)

labels 1

Form elements must have labels
Inspect Node | Highlight | 1 of 1

Issue description
Ensures every form element has a label
Impact: critical
Learn more

Element location
```
#input
```

Element source
```
<input type="text" id="input" placeholder="Things to do...">
```

To solve this violation, you need to:

Fix at least one (1) of these issues:
- aria-label attribute does not exist or is empty
- aria-labelledby attribute does not exist, references elements that do not exist or references elements that are empty
- Form element does not have an implicit (wrapped) <label>
- Form element does not have an explicit <label>
- Element has no title attribute or the title attribute is empty

Figure 23. Accessibility testing with axe.

The figure above displayed the issues found in the application, there were one violation and three best practice suggestions .These issues could be fixed according to the detailed suggestions given in the last section.

5.2.6 Continuous Integration with Travis

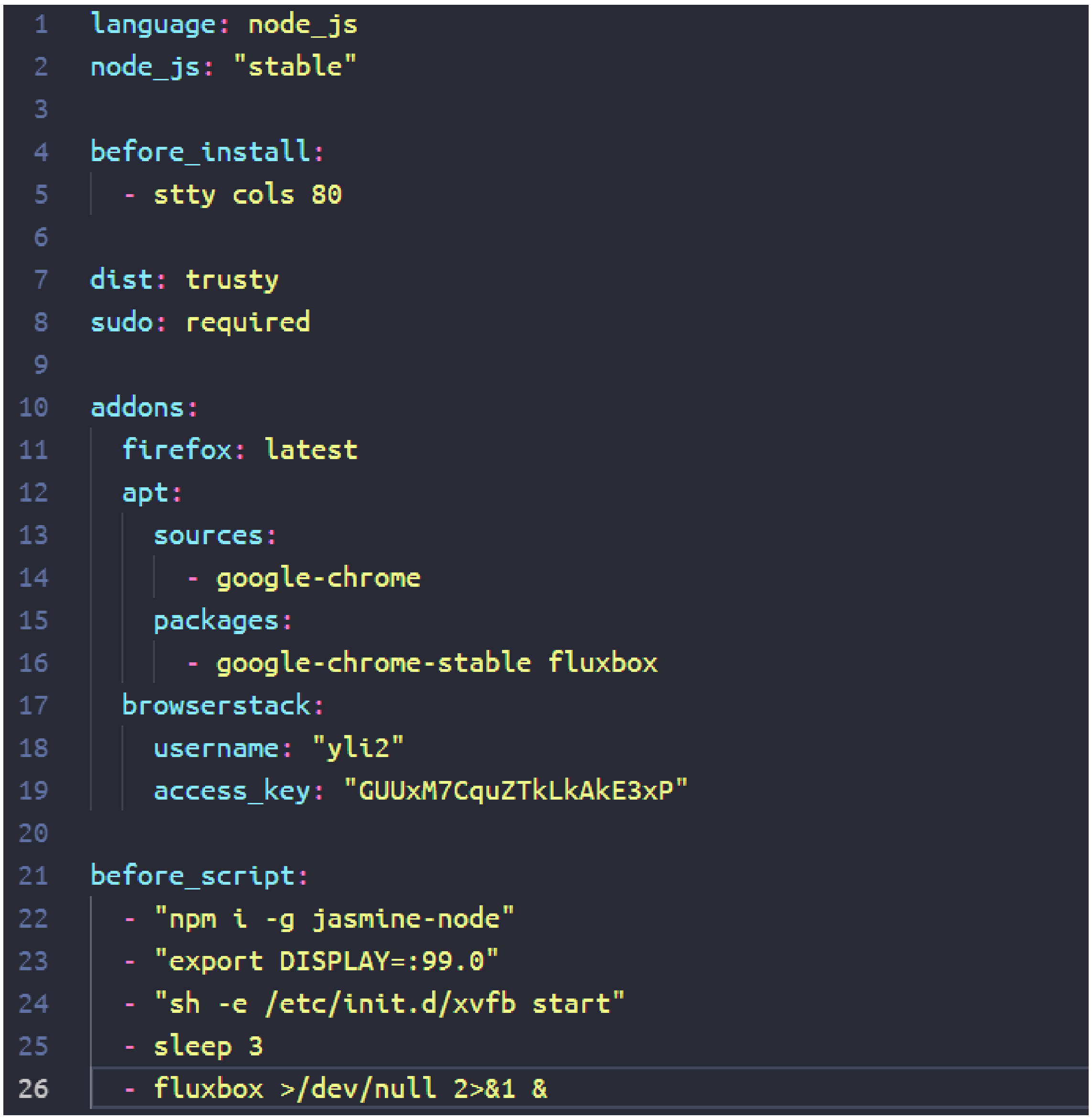
As introduced in chapter 4, continuous integration workflow is very popular and more efficient nowadays. For the purpose of demonstration, Travis CI was used for the test. Jenkins would be a great option for a larger project, since the ToDo List was a small Single Page Application, Travis was a sensible chose for simplicity and efficiency.

Travis CI is a continuous integration service that helps developers build and test software projects, it is free for open source projects, therefore, projects need to be pulic on a repository service platform in order to use it for free. Also, a versioning tool and a repository hosting service are needed to make the entire Continuous Integration work. In this test, Git and Github were selected to serve this purpose.

The ToDo List application was pushed to Github via Git and made it public, Then logged into Travis CI with Github account on Travis CI website, a permission request page from

Travis CI to access Github account was directed to, after pressing Authorize application button, Travis CI detected all projects hosted on the Github, ToDo List project was then activated for building and testing.

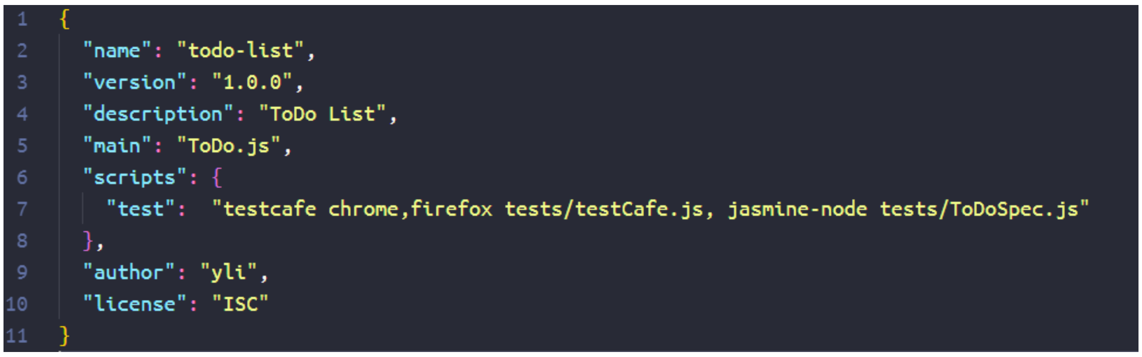
Travis CI will not run without a .travis.yml file, it is a configuration file that needs to be in the root directory of the project repository. The .travis.yml file specifies the project programming language and the relevant test information, basically, it tells Travis what to do.



```yaml
language: node_js
node_js: "stable"

before_install:
  - stty cols 80

dist: trusty
sudo: required

addons:
  firefox: latest
  apt:
    sources:
      - google-chrome
    packages:
      - google-chrome-stable fluxbox
  browserstack:
    username: "yli2"
    access_key: "GUUxM7CquZTkLkAkE3xP"

before_script:
  - "npm i -g jasmine-node"
  - "export DISPLAY=:99.0"
  - "sh -e /etc/init.d/xvfb start"
  - sleep 3
  - fluxbox >/dev/null 2>&1 &
```

Figure 24. Travis CI .travis.yml file.

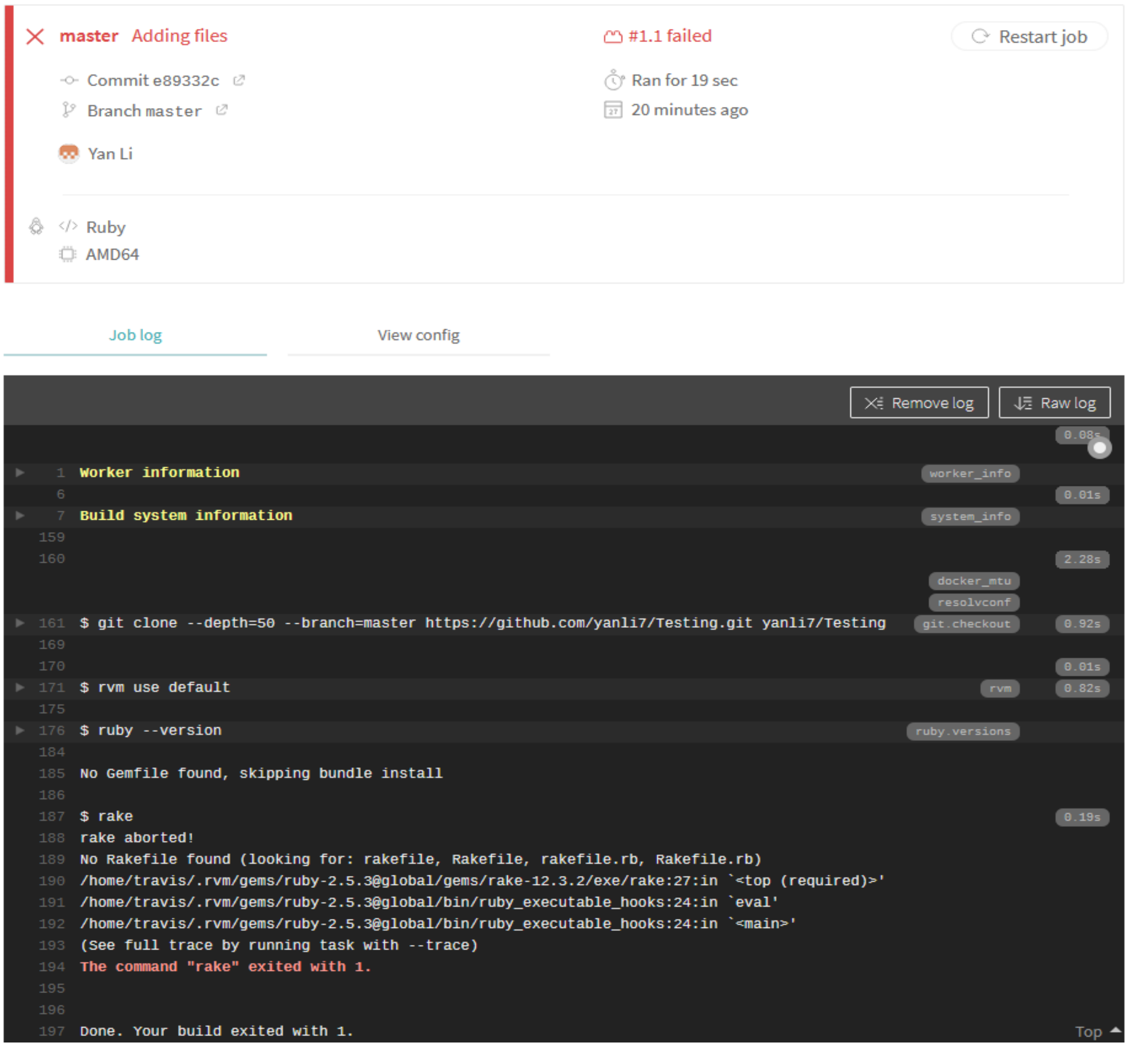
The .travis.yml file above contains programming language, end-to-end testing and cross-browser tesing information. Additionally, another file called package.json was also needed to let Travis know where the tests were and run them. The package.json file contains metadata related to the testing project.



```json
{
  "name": "todo-list",
  "version": "1.0.0",
  "description": "ToDo List",
  "main": "ToDo.js",
  "scripts": {
    "test":  "testcafe chrome,firefox tests/testCafe.js, jasmine-node tests/ToDoSpec.js"
  },
  "author": "yli",
  "license": "ISC"
}
```

Figure 25. The package.json file.

Travis CI was running automatically after pushing the .travis.yml file and package.json file to the ToDo List repository on Github. The build took 19 seconds to complete and the result was shown below:



```
Worker information                                                     worker_info   0.08s

Build system information                                               system_info   0.01s


                                                                                      2.28s
                                                                       docker_mtu
                                                                       resolvconf
$ git clone --depth=50 --branch=master https://github.com/yanli7/Testing.git yanli7/Testing   git.checkout   0.92s


                                                                                      0.01s
$ rvm use default                                                      rvm           0.82s

$ ruby --version                                                       ruby.versions

No Gemfile found, skipping bundle install

$ rake                                                                                0.19s
rake aborted!
No Rakefile found (looking for: rakefile, Rakefile, rakefile.rb, Rakefile.rb)
/home/travis/.rvm/gems/ruby-2.5.3@global/gems/rake-12.3.2/exe/rake:27:in `<top (required)>'
/home/travis/.rvm/gems/ruby-2.5.3@global/bin/ruby_executable_hooks:24:in `eval'
/home/travis/.rvm/gems/ruby-2.5.3@global/bin/ruby_executable_hooks:24:in `<main>'
(See full trace by running task with --trace)
The command "rake" exited with 1.


Done. Your build exited with 1.
```

Figure 26. Travis CI build result.

The build was failed and an email was received immediately about the build result. This is very convenient and efficient as every fix and push to the repository will make Travis CI run automatically and inform developers the building result and every red area will turn green with a checkmark in front of master branch if the build passes.

5.3 Final result

The ToDo List application was tested with unit test, functional test, cross-browser test, visual regression test and accessibility test. Each type of test was very important and common in front-end testing. During the implementation of these tests, most of the tests passed for the first time, however, there were some failed tests and compatibility issues in unit test and cross-browser test, also, there were some issues found in the accessibility test including one violation and three best practice suggestions. The failed tests appeared with error information and the issues could be fixed based on the information in order to pass the tests.

In addition to the individual test performed above, the practice of Continuous Integration was also implemented to demonstrate how this workflow actually worked. The implementation suggested that Continuous Integration workflow was very intuitive and efficient for the testing purpose, every change made to the code in the repository triggered the build automatically and the build result received immediately via email after the build finished.

However, the visual regression test and accessibility were not included in the Continuous Integration workflow because they were mainly testing the front-end UI and DOM elements, in reality, the UI and DOM element can be changed time to time before the final production, therefore, the best practice is to leave the visual regression test and accessibility for the last before launching the product.

6 CONCLUSION

Web technology has become more and more powerful and the front-end development has been evolved dramatically. Websites we visit today are very different from years ago, they have evolved from simply presenting information for users to allowing users to interact with them constantly, such as watching videos, checking maps, booking tickets, chatting online, filling forms and submitting them. Also, the user interface of the websites are more polishing and modern. All of these attractive features and powerful functions exist thanks to the fast development of the web technology. However, it is very complicated to develop such features behind the scenes. More features means more code and more complex structures and any error in the code or some conflicts between the code could break the feature or even the website. Therefore, front-end developers are carrying a great responsibility for the development. In order to make sure the entire website work as expected, the developers need to be very careful during the development, however, unexpected issures always appear no matter how careful they are and it is quite inevitable. The effective way of avoiding issues and assuring the quality of the front-end development is testing.

This thesis introduced front-end development and front-end testing. The importance of front-end testing was widely discussed in order to assure the quality of the development, including why front-end testing was needed, what should be tested and how to implement these tests. Also, a popular development workflow – Continuous Integration was also introduced.

During the implementation of the tests, a web application called ToDo List was built with some HTML, and mostly CSS and JavaScript. It was a list which allows users to add any items or events to it and mark them as they were completed, also, the events on the list could be deleted as well. A testing plan was created before the actual tests, including operating system, browsers, tools for different types of test, a Continuous Integration platform and a budget plan for the tests.

After the testing plan, the actual test for each type of testing was started. The testing included unit test, functional test, cross-browser test, visual regression test and accessibility test with testing tools such as Jasmine, TestCafe, BrowserStack, BackstopJS and Axe accordingly. During these tests, most of the tests passed successfully for the first time and some tests failed with errors that needed to be fixed.

Additionally, the practice of utilizing Continuous Integration workflow was also conducted, Travis CI was selected as the CI platform, Git and Github were chosen as a version control tool and repository hosting service. After pushing the .travis.yml file and package.json file to the Github repository of the application, Travis started the build automatically and the build result was sent out immediately once the build was completed. It turned out that the Continuous Integration workflow was very efficient and convenient for development and testing.

The purpose of the thesis was to introduce front-end testing, emphasize the importance of front-end testing and demonstrate how to implement the tests. There were many other tests could be performed as well in addition to the ones conducted in this thesis, however, the other tests might not be necessary or not as important for this specific application.

Front-end development has evolved dramatically and it will keep evolving constantly, to assure the quality of front-end development, front-end testing will keep being involved in the development process.

REFERENCES

ag-prime.com. (2019). Why is Testing part of the development process? [online] Available at: https://www.ag-prime.com/blog/why-testing-part-development-process [Accessed 04.08.2019]

angular.io. (2019). Introduction to the Angular Docs. [online] Available at: https://angular.io/docs [Accessed 25.11.2019]

Brennan, B. (2017). Libraries vs. Frameworks. [image] Available at: https://medium.com/datafire-io/libraries-vs-frameworks-626cdde799a7 [Accessed 09.06.2019]

bumblebee.co.za. (2019). Introduction to Angular. [online] Available at: http://bumblebee.co.za/2019/03/25/introduction-to-angular/ [Accessed 09.06.2019]

cigniti.com. (2019). Testing is a Process, not just a Phase. [online] Available at: https://www.cigniti.com/blog/testing-is-a-process-not-just-a-phase/ [Accessed 10.08.2019]

codeutopia.net. (2015). What are Unit Testing, Integration Testing and Functional Testing? [online] Available at: https://codeutopia.net/blog/2015/04/11/what-are-unit-testing-integration-testing-and-functional-testing/ [Accessed 20.07.2019]

Copes, F. (2018). A developer’s introduction to React. [online] Available at: https://jaxenter.com/introduction-react-147054.html [Accessed 09.06.2019]

flaviocopes.com. (2018). Introduction to the JavaScript Programming Language. [online] Available at: https://flaviocopes.com/javascript-introduction/ [Accessed 02.06.2019]

frontend.turing.io. Introduction to Unit Testing with JavaScript. [Online] Available at https://frontend.turing.io/lessons/module-1/introduction-to-testing-javascript.html [Accessed 14.07.2019]

github.com. (2019). BackstopJs. [online] Available at: https://github.com/garris/BackstopJS [Accessed 04.08.2019]

guru99.com. (2019). Software Configuration Management in Software Engineering. [online] Available at: https://www.guru99.com/software-configuration-management-tutorial.html [Accessed 27.10.2019]

guru99.com. (2019). What is JavaScript? Complete Introduction with Hello World! Example. [online] Available at: https://www.guru99.com/introduction-to-javascript.html [Accessed 02.06.2019]

guru99.com. (2019). What is Quality Assurance(QA)? Process, Methods, Examples. [online] Available at: https://www.guru99.com/all-about-quality-assurance.html [Accessed 26.10.2019]

guru99.com. (2019). What is Software Testing? Introduction, Definition, Basics & Types. [online] Available at: https://www.guru99.com/software-testing-introduction-importance.html [Accessed 28.10.2019]

Hoffmann, M. (2018). Why A Good Frontend Developer Should Care About Web Accessibility. [online] Available at: https://dev.to/mokkapps/why-a-good-frontend-developer-should-care-about-web-accessibility-545m [Accessed 09.10.2019]

hyperthot.com. (2019). Requirements Definition. [online] Available at: http://www.hyperthot.com/pm_reqs.htm [Accessed 27.10.2019]

insights.stackoverflow.com. (2018). Developer Survey Results 2018. [online] Available at: https://insights.stackoverflow.com/survey/2018#technology [Accessed 25.11.2019] jasmine.github.io. (2019). Jasmine sample code. [online] Available at: https://jasmine.github.io/ [Accessed 27.07.2019]

Jelisejevs, P, (2017). Automated Accessibility Checking with aXe. [online] Available at: https://www.sitepoint.com/automated-accessibility-checking-with-axe/ [Accessed 09.10.2019]

Lindley, C. (2018). What Is a Front-End Developer? [online] Available at: https://frontendmasters.com/books/front-end-handbook/2018/what-is-a-FD.html [Accessed 02.06.2019]

madasamy, M. (2018). JavaScript brief history and ECMAScript(ES6,ES7,ES8,ES9) features. [online] Available at: https://medium.com/@madasamy/javascript-brief-history-and-ecmascript-es6-es7-es8-features-673973394df4 [Accessed 08.06.2019]

microfocus.com. (2019). Continuous Integration Workflow. [image] Available at: https://www.microfocus.com/documentation/visual-cobol/VC40/EclWin/GUID-70375FE3-745F-4FA5-B28A-6F65953E562B.html [Accessed 11.08.2019]

microfocus.com. (2019). Introduction to Continuous Integration. [online] Available at: https://www.microfocus.com/documentation/visual-cobol/VC40/EclWin/GUID-B2D23B42-ECFE-464A-AC5B-03B4FED140C5.html [Accessed 11.08.2019]

nightwatchjs.org. (2019). Nightwatch demo test. [online] Available at: https://nightwatchjs.org/ [Accessed 03.08.2019]

Ram, P. (2018). How to implement Continuous Integration and Continuous Delivery in your organization. [online] Available at: https://hackernoon.com/how-we-transitioned-from-traditional-agile-to-agile-ci-cd-2ef9b7af5bf0 [Accessed 11.08.2019]

Reactjs.org. (2019). Tutorial: Intro to React. [online] Available at: https://reactjs.org/tutorial/tutorial.html [Accessed 25.11.2019]

Rouse, M. (2019). Quality Assurance (QA) . [online] Available at: https://searchsoftwarequality.techtarget.com/definition/quality-assurance [Accessed 26.10.2019]

Sedlock, A. (2017). An introduction to frontend testing. [online] Available at: https://www.creativebloq.com/how-to/an-introduction-to-frontend-testing [Accessed 30.05.2019]

softwaretestinghelp.com. (2019). What Is Cross Browser Testing And How To Perform It: A Complete Guide [online] Available at: https://www.softwaretestinghelp.com/how-is-cross-browser-testing-performed/ [Accessed 21.07.2019]

softwaretestinghelp.com. (2019). What is Software Quality Assurance (SQA): A Guide for Beginners [online] Available at: https://www.softwaretestinghelp.com/software-quality-assurance/ [Accessed 26.10.2019]

thoughtworks.com. (2019). Continuous Integration. [online] Available at: https://www.thoughtworks.com/continuous-integration [Accessed 11.08.2019]

vuejs.org. (2019). The Progressive JavaScript Framework. [online] Available at: https://vuejs.org/ [Accessed 25.11.2019]

wibas.com. (2019). Product Integration (PI) (CMMI-DEV). [online] Available at: https://www.wibas.com/cmmi/product-integration-pi-cmmi-dev [Accessed 28.10.2019]

Wozniewicz, B. (2019). The Difference Between a Framework and a Library. [online] Available at: https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/ [Accessed 08.06.2019]