UNIVERSITY of LIMERICK

OLLSCOIL LUIMNIGH

Department of Computer System & Information Systems

Image to HTML/CSS Website Converter

Student Name:	**JING LIU**
Student ID:	**16060296**
Supervisor:	**Annette McElligott**
Course Code:	**LM051**
Academic Year:	**2017/2018**

Acknowledgements:

To my supervisor, Annette McElligott, for her advice and patience throughout the year.

Summary

This project helps designers to create and generate a website front-end from an image. Designers can edit their design drawings (Adobe Photoshop file only (PSD)) adding or removing object without the need to know about program code. This application can generate in real-time HTML and CSS code. If an object is modified, by the user, the corresponding HTML and CSS code will automatically be updated. The resulting HTML and CSS code can be executed to generate a web page template.

PSD layer analysis and image processing technology are used for automatic conversion to HTML and CSS code. This tool will automatically parse the layer information in the PSD structure and extract each layer of useful information, giving a parent-child relationship structure. Overall, this application will reduce a designer's work and a developer's work and assist in the production of high quality web front-ends.

Table of Contents

Summary 1

1. Introduction 1

  1.1 Background 1

  1.2 Staff Roles and Responsibilities 2

  1.3 Project Aim and Objectives 2

  1.4 Motivation 3

  1.5 Structure of Report 3

2. Research 4

  2.1 Introduction 4

  2.2 PSD File Structure 4

  2.3 Technology researched 7

    2.3.1 PSD Layer Parsing 7

    2.3.2 Image Processing 11

  2.4 Existing Application Researched 11

    2.4.1 PSD to WEB 11

    2.4.2 PSD2HTML 12

    2.4.3 Photopea 14

  2.5 Function of Proposed Tool 14

3 Technologies Used 16

  3.1 Common JS 16

  3.2 Node.JS 17

  3.3 Gulp 18

  3.4 Webpack 20

  3.5 React 21

  3.6 Canvas 25

4. Implementation 26

  4.1 Introduction 26

  4.2 Summary 26

4.3 Function Implementation........ 26
4.4 Problem Encountered........ 37
5. Testing and Evaluation........ 38
5.1. Introduction........ 38
5.2. Functional Testing........ 38
5.3. Conclusion........ 40
6. Conclusion........ 41
Reference:........ 42

1. Introduction

1.1 Background

Typically, website development comprises six phases:

(1) **Information Gathering:** The first stage of website development involves research and discovery to get a clear understanding of the purpose of the website, the target audience you want to attract and so forth. Often information gathering exercises are conducted for example, interviews may take place and questionnaires might be distributed so that the objectives of the website can be determined and adequate planning can take place to achieve these objectives. Any issues that arise are dealt with and resources are also determined at this stage.

(2) **Planning: Sitemap and Wireframe Creation:** Once it is decided to proceed with the website the developer can create a sitemap that describes the relationship between the key areas of the website. This sitemap can give the customer a better understanding of the proposed website. Also, the developer might create wireframe diagrams (cf. Figure 1), these can assist in getting approval from the customer. Wireframe diagrams are a visual representation of the user interface that will be created but they do not contain design elements such as colours, logos and so forth. They only outline the location of the graphical user interface elements that will be added to the page however, they give the customer an indication of how the website will operate from a user's perspective.

(3) **Design: Page Layouts, Review, and Approval Cycle:** At this stage, the visual content such as images, videos and so forth are created, each page layout is structured and content is added. From here, an approval cycle begins with the customer whereby they review the layout, feedback is provided and incorporated, and the design is again presented to the user for feedback. This cycle continues until the customer is satisfied.

(4) **Content Writing and Assembly:** This stage involves the customer supplying content to be communicated to the target audience. This may involve taking content from an existing website. As a rule, the customer undertakes to supply content for headings, articles and so forth that can be assembled by the developer on the new site.

(5) **Coding:** At this stage the developer finally starts creating the home page and then sub-pages are added according to agreed sitemap created earlier. The developer may use a Content Management System or a web development framework. Static elements

designed earlier are added and tested. Then interactivity and features are designed and developed. Also, the developer needs to agree keywords for search engine optimization with the customer and include these.

(6) **Testing, Review, Launch:** At this stage each page is testing, forms are tested, each script is run, a spelling-checker is run and code validation tools are also executed to check that the relevant website standards are adhered to. Further testing takes place when the website is uploaded to the server prior to the system being launched.

Once launched a feedback system will allow for errors detected to be recorded and prioritised for fixing.

1.2 Staff Roles and Responsibilities

Typically, there are four categories of staff involved in website development:

- the web manager whose responsibility is mainly about project management, planning resources such as budgets, staff supervision and other operational matters;
- the content producer who is responsible for creating and maintaining high-quality on-line content such as text, videos, podcasts, twitter updates and so forth;
- the web designer who is responsible for creating visual templates that are used for on-line communication and interaction. They design page layouts for approval and once approved they add content. They need to have a knowledge of HTML, CSS, JavaScript and have good graphic design skills and be able to use tools such as Adobe Photoshop, Acrobat Studio and so forth;
- the web developer is responsible for both the front-end and back-end development and needs to have a knowledge of web development frameworks

1.3 Project Aim and Objectives

This project targets the designer, especially the designer who has limited scripting and programming skills. However, they want to be able to modify visual templates and gain a better understanding of HTML and CSS. It is expected that the proposed tool will provide them with functionality to automatically generate HTML and CSS code from an image with which they can interact, thereby reducing their workload. Specifically, I want develop a tool that a designer can import a PSD file and have it converted to HTML and CSS so that they can modify it.

1.4 Motivation

Studies show that a well-built website will generate greater customer traffic and a good user interface is one element that is key to this. Also, I want to help a designer to user their skills to create good user interfaces. Finally, this allows me to combine several aspects of my course in particular parsing and programming.

1.5 Structure of Report

Chapter 2 will detail the structure of a PSD file, technologies that I researched and existing tools similar to my project. Chapter 3 will summarise technologies that I used for the development of the tool. Chapter 4 will outline the implementation of each function. Chapter 5 will represent the testing process and evaluation. Chapter 6 will give a conclusion of my project.

2. Research

2.1 Introduction

This chapter will detail the structure of a PSD file (cf. Section 2.2) and technologies that I researched namely, PSD Layer Parsing (cf. Section 2.3.1), the Canvas element in HTML5 (cf. Section 2.3.2), and OpenCV (cf.Section 2.3.3). In addition, I examined tools similar to my project, PSD To Web (cf.Section 2.4.1), PSD2HTML(cf.Section 2.4.2) and Photopea (cf.Section 2.4.3). Finally, this chapter presents a summary of the functions of the proposed tool (cf. Section 2.5).

2.2 PSD File Structure

A PhotoShop Document (PSD) file is a layered image file used in Adobe PhotoShop. This is the default format that Adobe Photoshop uses for saving a file. It is a proprietary file format that allows the user to work with the images of individual layers even after the file has been saved. PSD files are used for image manipulation and magazine front cover images as it is able to save layers, page formatting information and so forth. PSD supports transparency so images can be saved without a background.

The Photoshop file format is divided into five major parts (cf. Figure 2.1). The file header section (cf. Figure 2.2) contains the signature, the version number, the number of channels in the image and the number of bits per channel, the height and width of the image in pixels and the colour mode e.g. bitmap, greyscale, Indexed, RGB, CMYK.

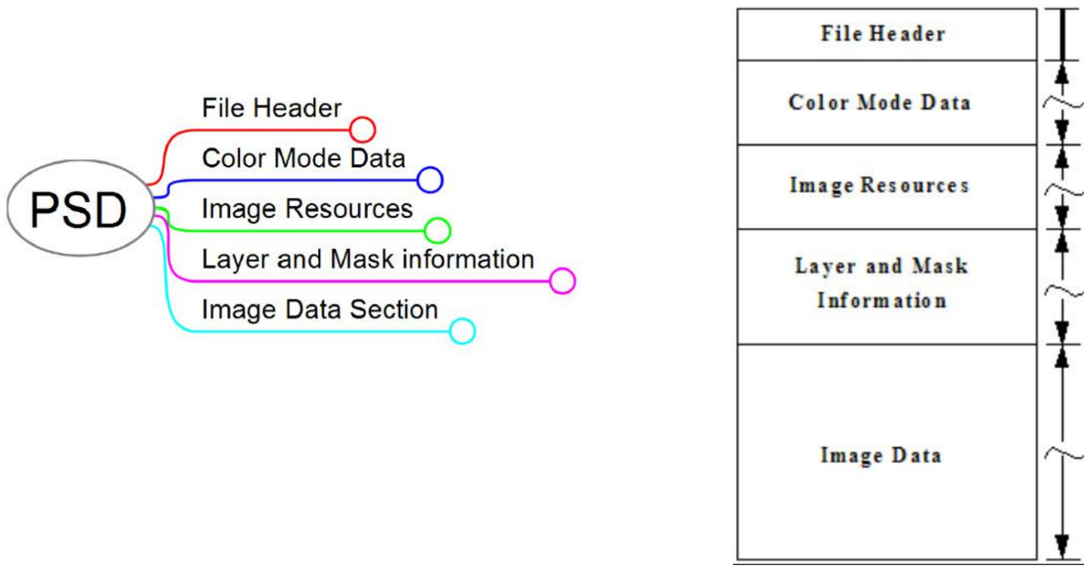
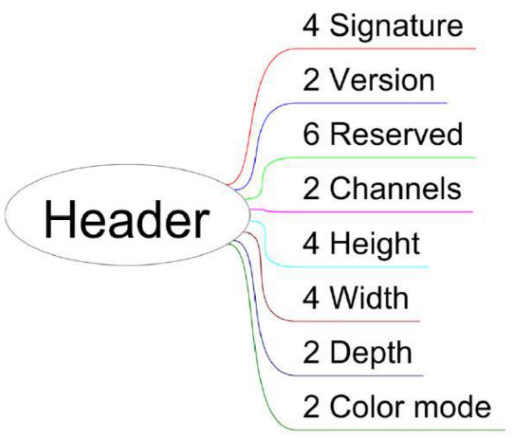


Figure 2.1: PSD File Structure



The header section looks like this:

```
{
 "sig"=>"8BPS",
 "version"=>1,
 "channels"=>3,
 "rows"=>600,
 "cols"=>900,
 "depth"=>8,
 "mode"=>3,
 "color_data_len"=>0
}
```

Figure 2.2: PSD File Header

In relation to Color Mode Data (section 2 of file header) there is only a colour mode section (cf. Figure 2.2) when the colour mode in the file header colour mode field section (cf. Figure 2.1) is set to Indexed or Duotone. Otherwise, the length of the section is set to zero and there is no data (cf. Figure 2.3).

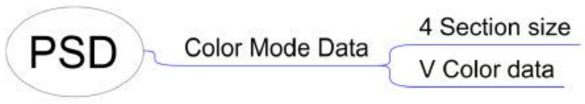


Figure 2.3: Color Mode Data

The third section of the header contains image resources (cf. Figure 2.4). It starts with the length of the image resource section, and is followed by a series of resource blocks (cf. Figure 2.5). Image resource blocks are the basic building unit of several file formats, including Photoshop’s native file format, JPEG and TIFF. Image resources are used to store non-pixel data associated with images, such as pen tool paths. Examples of resource formats are: grid and guides resource format, thumbnail resource format, colour samplers resource format, path resource format, slices resource format and so forth.

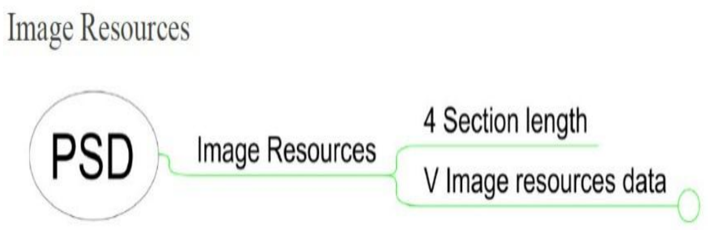


Figure 2.4: Image Resources

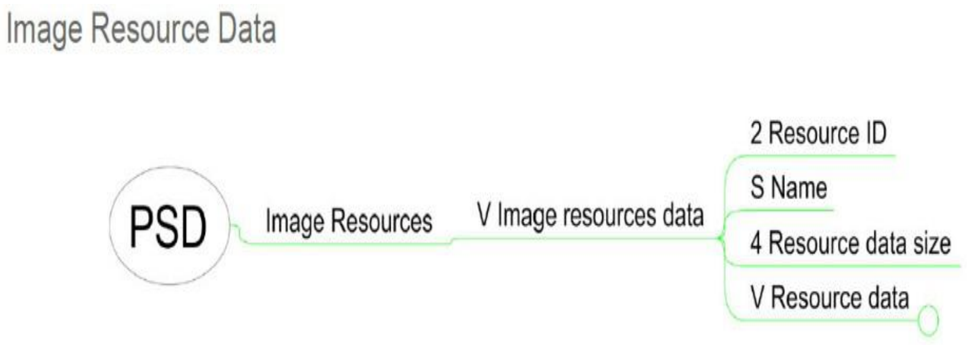


Figure 2.5: Resource Formats

The fourth section contains information about layers and masks and describes the formats of layer and mask records. The layer and mask information contains the length of the layers info section and layer count, information about each layer and channel image data. It can be divided into the subsections shown in Figure 2.6.

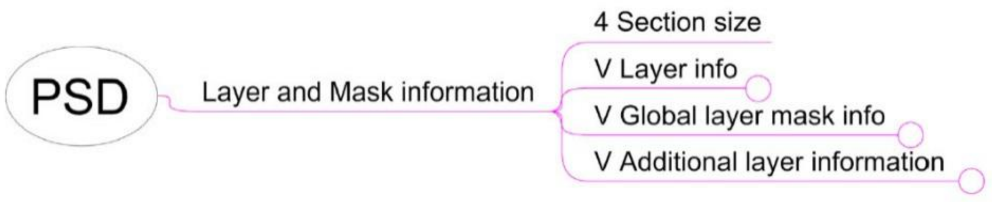


Figure 2.6: Layer and Mask Information

The last section of a Photoshop file contains the image pixel data (cf. Figure 2.7). Image data is stored in planar order: first all is the red data, then all the green data, last of all the blue data.

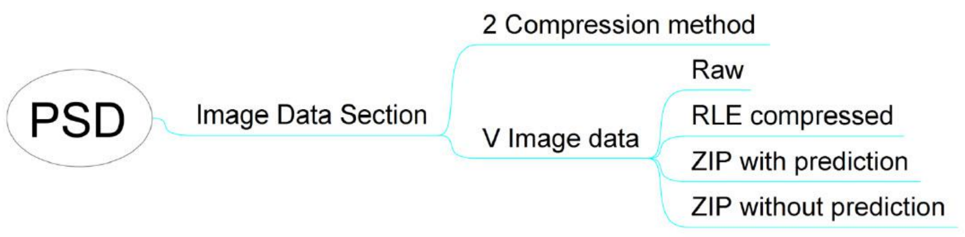


Figure 2.7: Image Data Section

2.3 Technology researched

2.3.1 PSD Layer Parsing

HTML is the main mark-up language for creating web pages that can be displayed in a web browser. The PSD file is Photoshop format contains several image layers. Generally, the primary designs are made with Photoshop in PSD format and then manually converted into HTML code. However, the first objective of my project is to convert a PSD to HTML code and CSS automatically.

In order to achieve this functionality, PSD layer parsing is the first phase of to be undertaken. The result of parsing is a layer list, where each layer has its own unique name. Each layer is then converted to a PNG image and given the same unique name as the layer e.g. layer1 and layer1.png. This same name will be given to a CSS class name for identification. The next stage is HTML and CSS generation.

These first step is summarised by Josekutty (Josekutty, A, 2014, Para 9) as "*Default data table structure is used to store the layer information. Then the list is sorted in an increasing order. Another function is to find the relations of parents and children. After analysis, we will get some uncategorized layers, which can be sibling or confused layers. These layers will be grouped or attached to a new parent layer*".

All steps involved in PSD to HTML and CSS are shown in Figure 2.8.

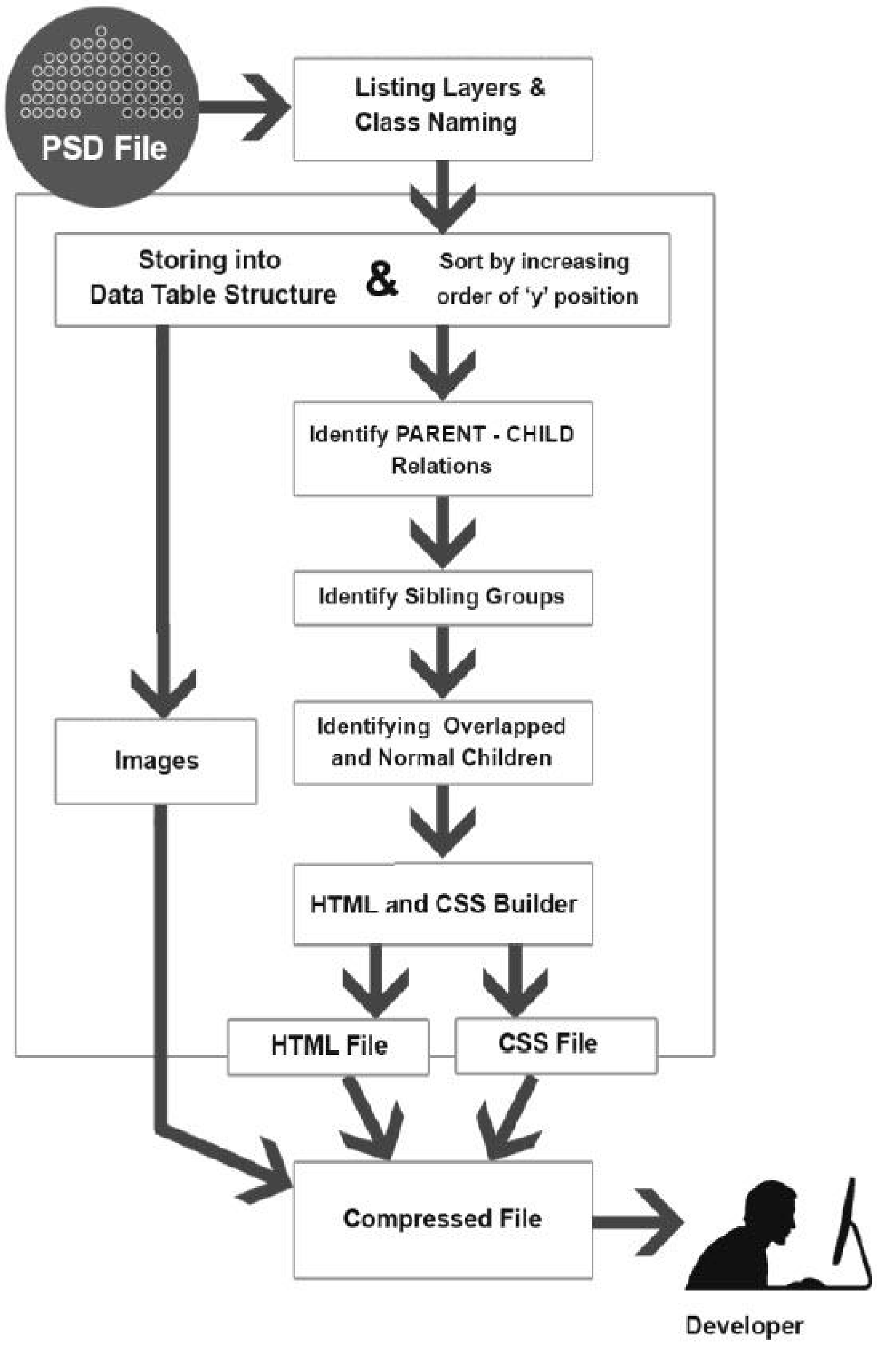
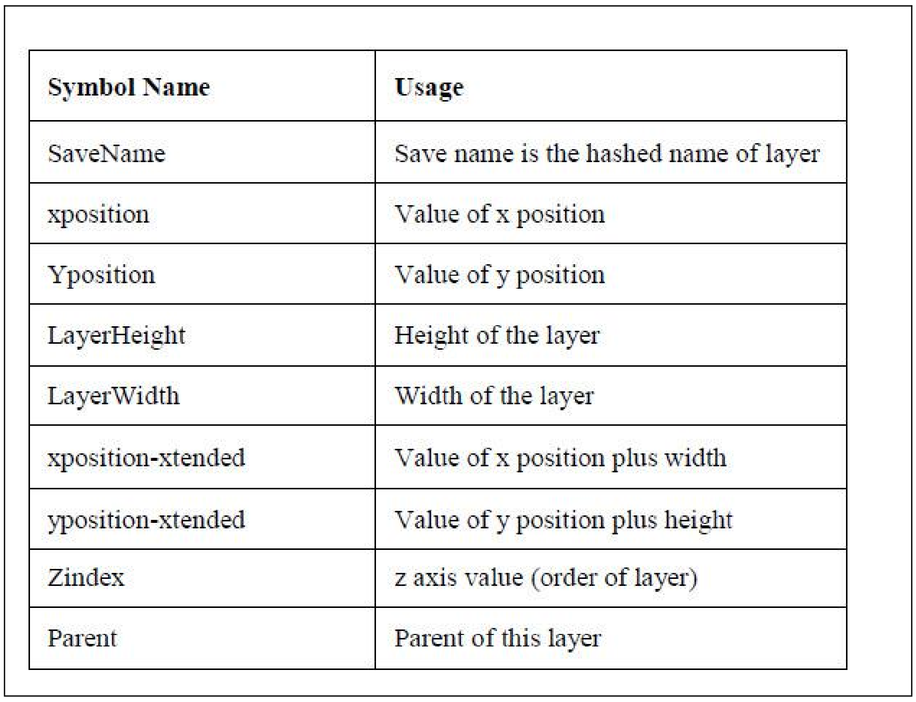

PSD File
Listing Layers & Class Naming
Storing into Data Table Structure
&
Sort by increasing order of 'y' position
Identify PARENT - CHILD Relations
Identify Sibling Groups
Identifying Overlapped and Normal Children
Images
HTML and CSS Builder
HTML File
CSS File
Compressed File
Developer

Figure 2.8: Steps to convert PSD file to HTML and CSS formats

For each layer a symbol table in created. Figure 2.9 give a summary of the attributes in such a symbol table.



Symbol Name	Usage
SaveName	Save name is the hashed name of layer
xposition	Value of x position
Yposition	Value of y position
LayerHeight	Height of the layer
LayerWidth	Width of the layer
xposition-xtended	Value of x position plus width
yposition-xtended	Value of y position plus height
Zindex	z axis value (order of layer)
Parent	Parent of this layer

Figure 2.9: Symbol Table Structure for Each Layer

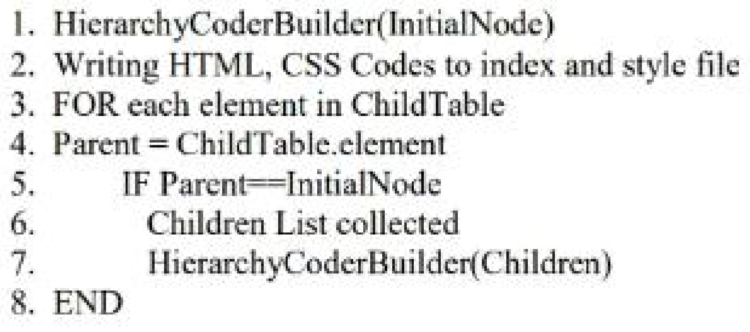
The table structure can be displayed by a JSON or XML file. "JSON is designed to be a data exchange language which is human readable and easy for computers to parse and use. It provides significant performance gains over XML, which requires extra libraries to retrieve data from DOM object, JSON is estimated to parse up to one hundred times faster than XML in modern browsers. So JSON is directly supported inside JavaScript and is best suited for JavaScript applications." (Nurzhan, Michael, Randall, Clemente, para 5).

Figure 2.10 given an example of a JSON file showing the information of each layer. The layers shown in this figure are "widget", "image" and "text". Depending on the layer, further information is given.

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "This is an example JSON file",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

Figure 2.10: JSON file example

In order to convert the JSON to HTML and CSS we need to extract the parent-child relationships from the JSON based on the information in the symbol table. An algorithm similar to that in Figure 2.11 could be used for this purpose.



```
1. HierarchyCoderBuilder(InitialNode)
2. Writing HTML, CSS Codes to index and style file
3. FOR each element in ChildTable
4. Parent = ChildTable.element
5.      IF Parent==InitialNode
6.         Children List collected
7.         HierarchyCoderBuilder(Children)
8. END
```

Figure 2.11: Algorithm to convert JSON to HTML and CSS

2.3.2 Image Processing

1.Canvas

Canvas elements introduced in HTML5 greatly simplify image processing with JavaScript. JavaScript can perform pixel-level operations on the image through canvas, and even process the binary raw data of an image. Besides, canvas provides image format conversion functions to change encoding of image quickly and easily.

Canvas can be used for image resizing, cropping an image, obtaining the color of a pixel and modifying its colour. Furthermore, it also supports for image translation and mirror transformation.

1. OpenCV

"*OpenCV is a library of programming functions mainly aimed at real-time computer vision.*" (Wikipedia). The OpenCV library contains a mix of low-level image-processing functions and high-level algorithms such as face detection, pedestrian detection, feature matching, and tracking (Adrian.R, 2014).

For some basic image manipulation tasks, there are functions in the OpenCV library that allow one to resize, crop and rotate an image which could be used in the proposed system.

2.4 Existing Application Researched

2.4.2 PSD to WEB

The application PSD to WEB converts a PSD file to HTML/CSS but it does not support layer masks. The Figure in 2.12 was uploaded to this tool and it returned the HTML with embedded CSS as shown in Figure 2.13. However, the proposed tool will keep the HTML in one file and the CSS style information in a separate file.

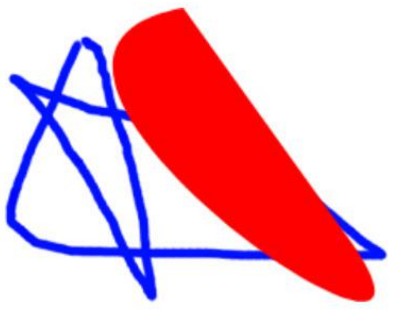
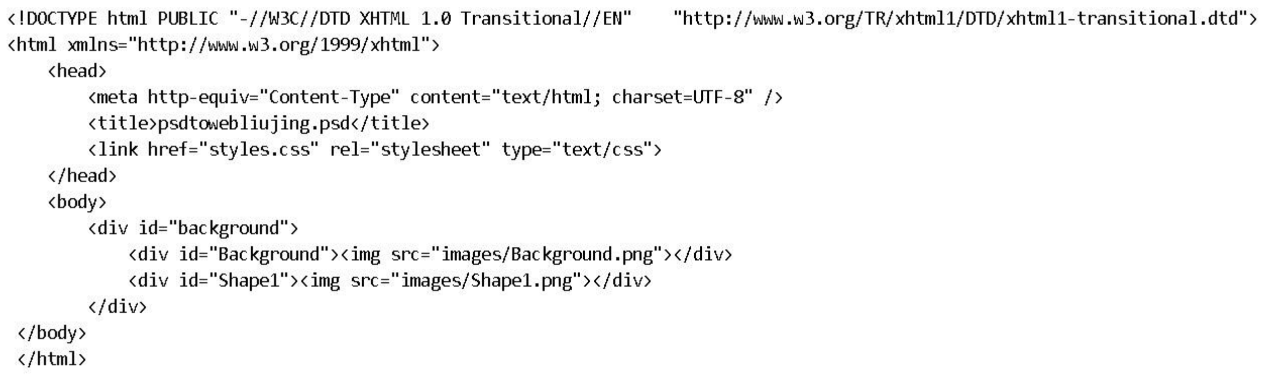


Figure 2.12: Image used to examine PSD to WEB tool



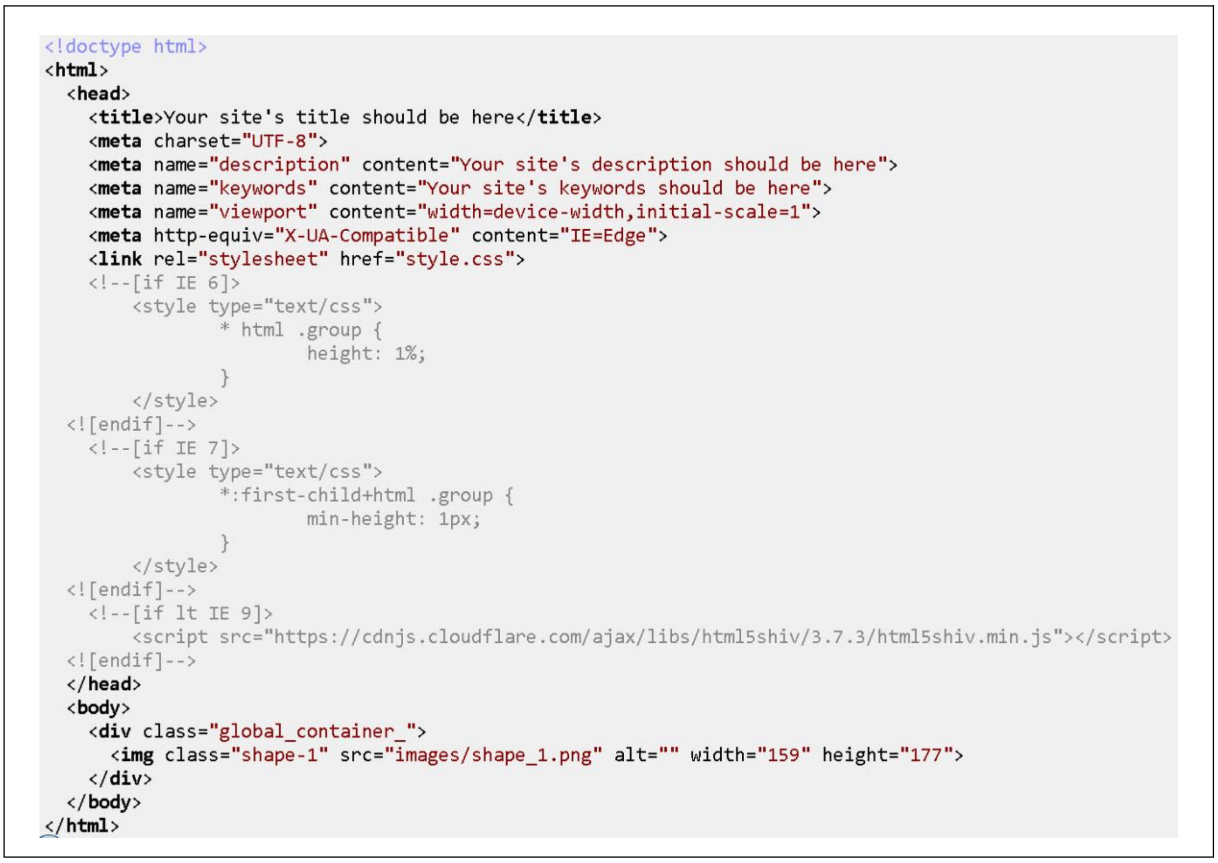
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>psdtowebliujing.psd</title>
        <link href="styles.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div id="background">
            <div id="Background"><img src="images/Background.png"></div>
            <div id="Shape1"><img src="images/Shape1.png"></div>
        </div>
 </body>
 </html>
```

Figure 2.13: Result given by PSD to WEB tool

2.4.2 PSD2HTML

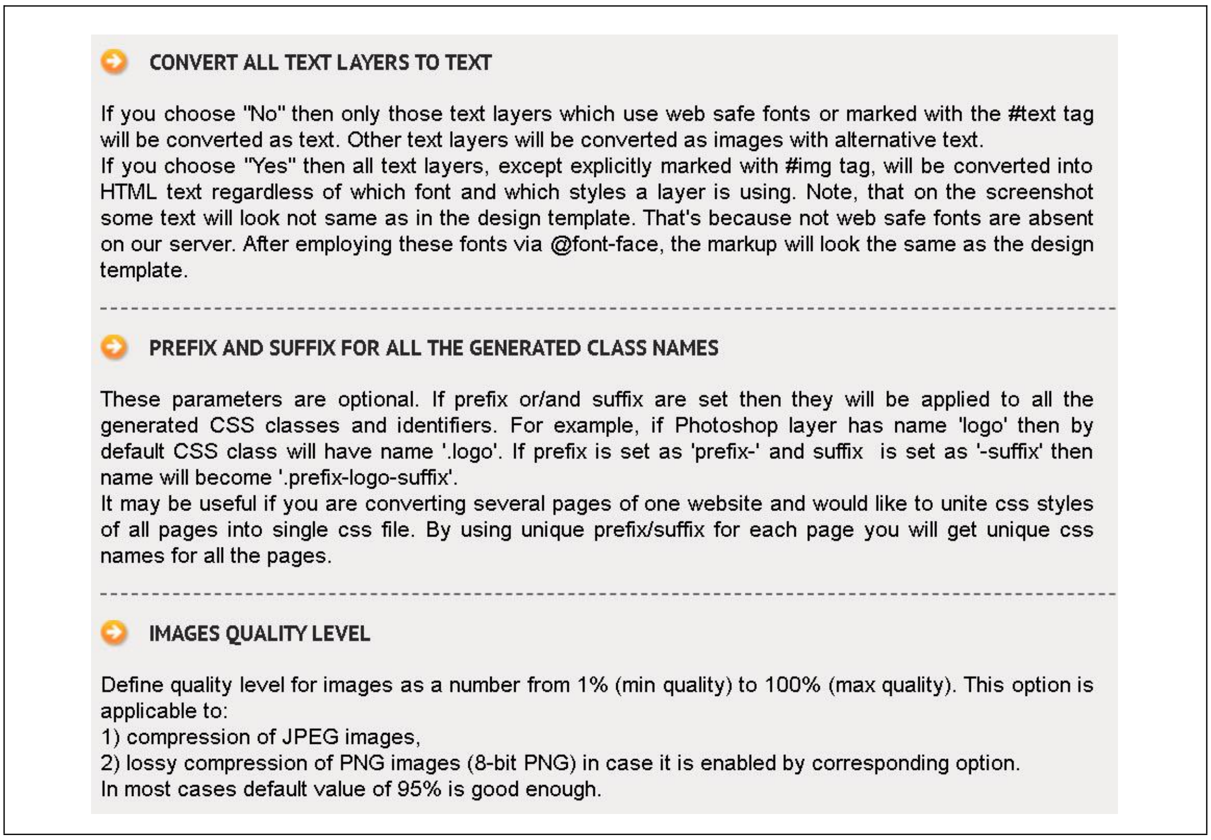
The PSD2HTML application converts a PSD file to HTML and CSS online. The uploaded PSD file limit is 400MB. It generates HTML and CSS code as two separate files and it slices the PSD file into images. For the file in Figure 2.12 when uploaded to this tool the resulting HTML is shown in Figure 2.14.

This tool allows the user to choose how they want the layers processed. It allows the user to select various options such as layout type, font size units, whether to convert all text layers to text, the image quality, whether to detect layers with fixed positioning and whether to remove unused images. A summary of these is presented in Figure 2.15 which is captured from the tool.



```
<!doctype html>
<html>
  <head>
    <title>Your site's title should be here</title>
    <meta charset="UTF-8">
    <meta name="description" content="Your site's description should be here">
    <meta name="keywords" content="Your site's keywords should be here">
    <meta name="viewport" content="width=device-width,initial-scale=1">
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <link rel="stylesheet" href="style.css">
    <!--[if IE 6]>
        <style type="text/css">
                * html .group {
                        height: 1%;
                }
        </style>
  <![endif]-->
    <!--[if IE 7]>
        <style type="text/css">
                *:first-child+html .group {
                        min-height: 1px;
                }
        </style>
  <![endif]-->
    <!--[if lt IE 9]>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shiv.min.js"></script>
  <![endif]-->
  </head>
  <body>
    <div class="global_container_">
      <img class="shape-1" src="images/shape_1.png" alt="" width="159" height="177">
    </div>
  </body>
</html>
```

Figure 2.14: Resulting HTML from PSD2TML tool



CONVERT ALL TEXT LAYERS TO TEXT

If you choose "No" then only those text layers which use web safe fonts or marked with the #text tag will be converted as text. Other text layers will be converted as images with alternative text.
If you choose "Yes" then all text layers, except explicitly marked with #img tag, will be converted into HTML text regardless of which font and which styles a layer is using. Note, that on the screenshot some text will look not same as in the design template. That's because not web safe fonts are absent on our server. After employing these fonts via @font-face, the markup will look the same as the design template.

PREFIX AND SUFFIX FOR ALL THE GENERATED CLASS NAMES

These parameters are optional. If prefix or/and suffix are set then they will be applied to all the generated CSS classes and identifiers. For example, if Photoshop layer has name 'logo' then by default CSS class will have name '.logo'. If prefix is set as 'prefix-' and suffix is set as '-suffix' then name will become '.prefix-logo-suffix'.
It may be useful if you are converting several pages of one website and would like to unite css styles of all pages into single css file. By using unique prefix/suffix for each page you will get unique css names for all the pages.

IMAGES QUALITY LEVEL

Define quality level for images as a number from 1% (min quality) to 100% (max quality). This option is applicable to:
1) compression of JPEG images,
2) lossy compression of PNG images (8-bit PNG) in case it is enabled by corresponding option.
In most cases default value of 95% is good enough.

Figure 2.15: Summary of options user can select when using PSD2 HTML tool

A limitation of this tool is that presents the HTML for you to view. This code can be copied/pasted and saved locally to a .html file. However, you cannot do this with the CSS unless you pay a fee. Therefore, one cannot view their webpage.

2.4.3 Photopea

Photopea allows the user to upload a PSD file. This tool presents the PSD as shown in Figure 2.16. The user can interact with the result via a very good graphical user interface to select, move, crop, use text tools, blur the image and so forth, however, the user cannot see the HTML and/or CSS.

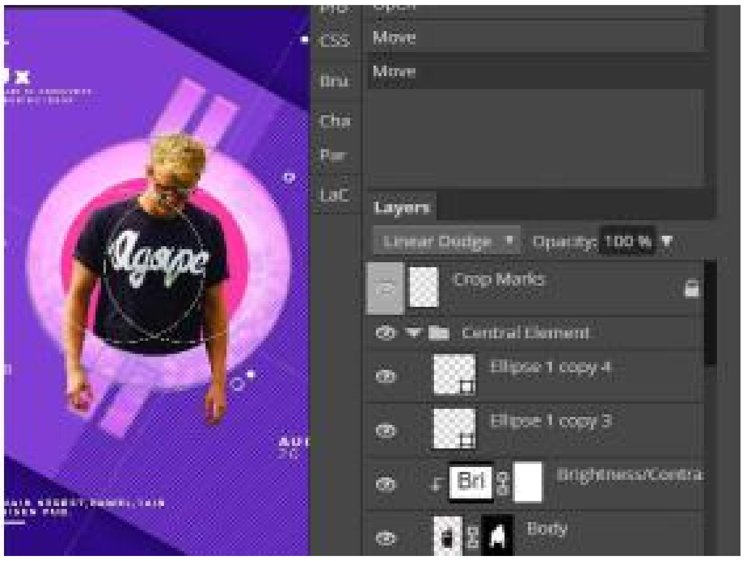


Figure 2.16: GUI of Photopea

2.5 Function of Proposed Tool

Following a review of current tools, the proposed tool should provide the user with the following functionality:

- **Open:** Allows the user to open a PSD file.
- **Conversion:** Allow user convert PSD file to HTML/CSS code, when users modify the PSD file, the HTML/CSS code will automatically change.
- **Save:** Allow the user to save the current image or save as JPG, PNG image
- **Resize:** Allows the user to define the size of the image in every layer
- **Move:** Allows the user to change the position of the object in every layer

- **Crop:** Allows the user to crop a portion of the image. It selects a portion of the image using the Selection feature and removes everything except for the selected portion.
- **Pencil:** Allows the user to draw free-hands lines on the image. The user clicks on the point where he wants to begin drawing and then dwells again on the point where he wants to stop drawing. The colour of the pencil-line can be changed using colour feature.
- **Rotate:** Allows the user to rotate the image clockwise or counter clockwise 45, 90, or 180 degrees.
- **Colour:** Allow the use to change the colour of the Pencil feature and text.
- **Undo:** Allows the user to undo his recent actions in any feature.
- **Copy:** Allows the user to copy a selected portion of the image onto the system clipboard
- **Paste:** Allows the user to paste data from system clipboard onto the current image. Such as text.
- **Add:** Allows the user to add a new object to the PSD file
- **Remove:** Allows the user to remove an object from the PSD file
- **Split:** Allows the user to split a PSD file to PNG or JPG images, every PNG or JPG image is a layer of a PSD file and it can be displayed individually.

3 Technologies Used

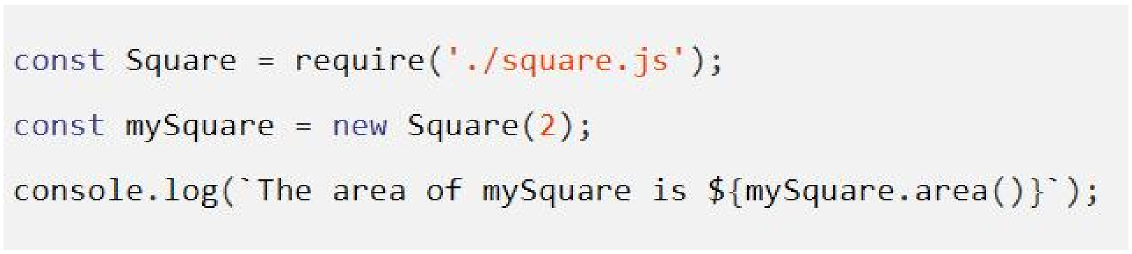
In this chapter, I will outline the technologies that I will use for the development of the proposed PSD convertor as follows:

- CommonJS (cf. Section 3.1),
- Node.js (cf. Section 3.2),
- Gulp (cf. Section 3.3),
- Webpack (cf. Section 3.4),
- React.js (cf. Section 3.5),
- Canvas (cf. Section 3.6).

3.1 Common JS

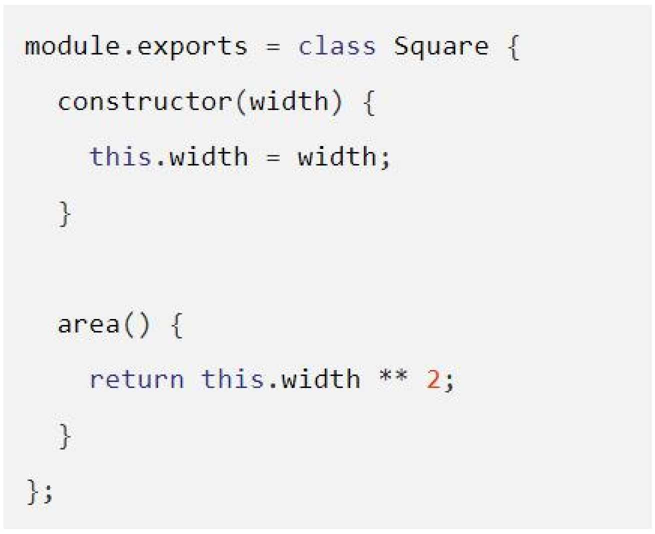
“Modular programming is a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.” (Wikipedia Para1).

Originally, JavaScript was used in client-side development (browser or user side). CommonJS (originally called ServerJS) is a simple API for declaring modules that work outside of the browser such as on the server. Basically, a CommonJS module is a reusable piece of JavaScript that exports specific objects and makes them available to any dependent code. Such a module contains two primary parts: a free variable named **exports** () that contains the objects a module wishes to make available to other modules, Figure 3.2 use exports key word to export the square class so that other mouse can use it. Figure 3.1 shows another variable **require(),** it presents that modules can be used for importing other modules.



```
const Square = require('./square.js');
const mySquare = new Square(2);
console.log(`The area of mySquare is ${mySquare.area()}`);
```

Figure 3.1: Code use require keyword to import square.js module.



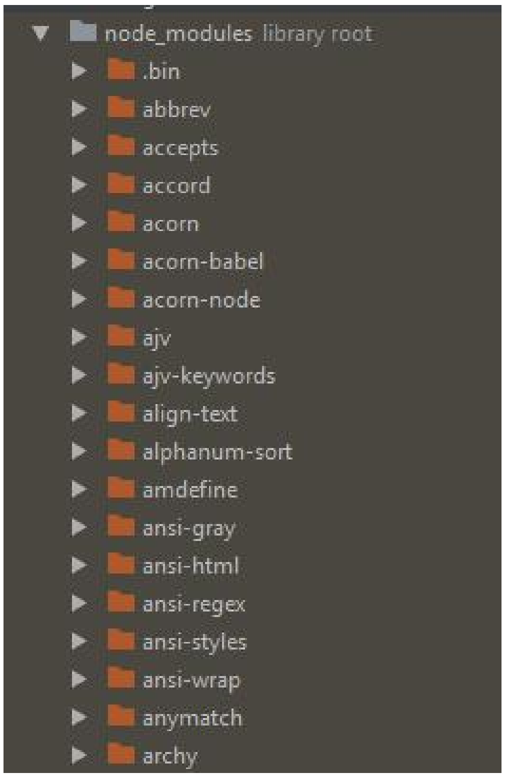
```
module.exports = class Square {
  constructor(width) {
    this.width = width;
  }

  area() {
    return this.width ** 2;
  }
};
```

Figure 3.2: Code uses exports keyword to export square.js as a module.

3.2 Node.JS

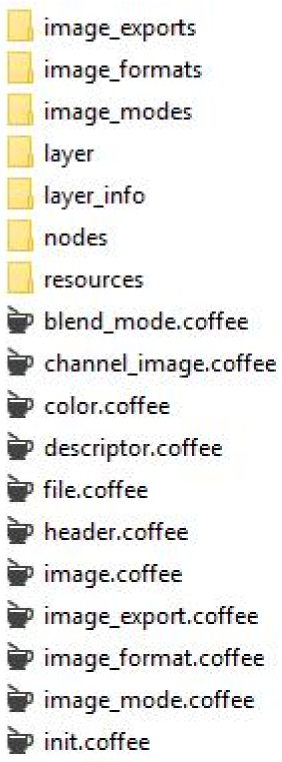
“Node.JS is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side.” (Wikipedia). As we know, JavaScript was originally used for client-side scripting, in which scripts written in JavaScript were embedded in the user’s web page. So using Node.JS we can write both the back-end and front-end of a web application entirely in JavaScript.

Node.js contains lots of APIs, which shown in Figure 3.3, called node_modules. On my project, the PSD parser used lots of encapsulating modules, this Figure 3.4 represent a part of files that PSD parser have to import. such as image format, image modes, layer, layer information, nodes and so on. Figure 3.5 is an example that uses “psd” module in the node modules when uploading a PSD file. Besides, NodeJS environment is the premise of using react framework.



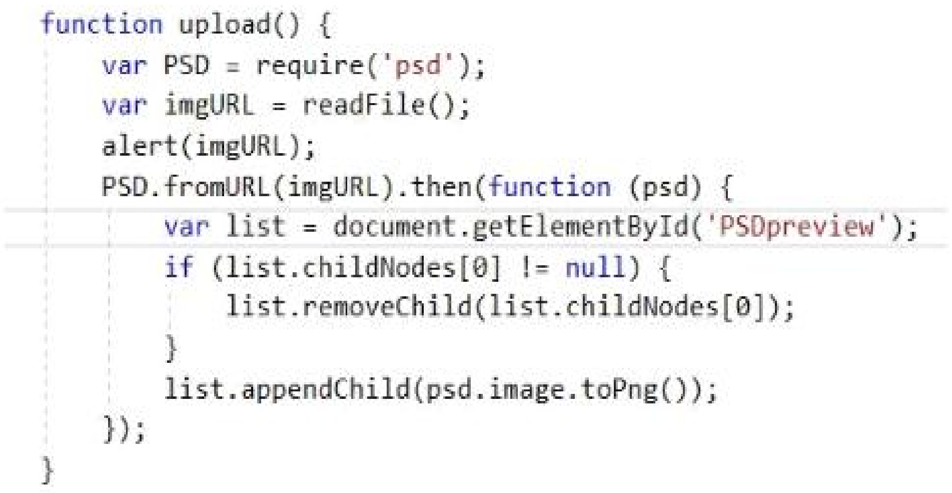
```
▼ node_modules library root
    ▶ .bin
    ▶ abbrev
    ▶ accepts
    ▶ accord
    ▶ acorn
    ▶ acorn-babel
    ▶ acorn-node
    ▶ ajv
    ▶ ajv-keywords
    ▶ align-text
    ▶ alphanum-sort
    ▶ amdefine
    ▶ ansi-gray
    ▶ ansi-html
    ▶ ansi-regex
    ▶ ansi-styles
    ▶ ansi-wrap
    ▶ anymatch
    ▶ archy
```

Figure 3.3: node_modules



```
image_exports
image_formats
image_modes
layer
layer_info
nodes
resources
blend_mode.coffee
channel_image.coffee
color.coffee
descriptor.coffee
file.coffee
header.coffee
image.coffee
image_export.coffee
image_format.coffee
image_mode.coffee
init.coffee
```

Figure 3.4: PSD module



```
function upload() {
    var PSD = require('psd');
    var imgURL = readFile();
    alert(imgURL);
    PSD.fromURL(imgURL).then(function (psd) {
        var list = document.getElementById('PSDpreview');
        if (list.childNodes[0] != null) {
            list.removeChild(list.childNodes[0]);
        }
        list.appendChild(psd.image.toPng());
    });
}
```

Figure 3.5: Code of upload PSD file using “psd” module.

3.3 Gulp

“*Gulp is a streaming JavaScript build system built with Node.JS that leverages the power of streams and code-over-configuration to automate, organizes, and run development tasks very* *quickly and efficiently. By simply creating a small file of instructions, Gulp can perform just about any development task you can think of”* (‘Getting started with gulp’, Page7). Gulp’s working style is in a configuration file; specify the specific steps for similar tasks such as compiling, combining, and compressing certain files, then the build tool can automatically complete these tasks, Figure 3.6 is a workflow of gulp.

1. Gulp Workflow

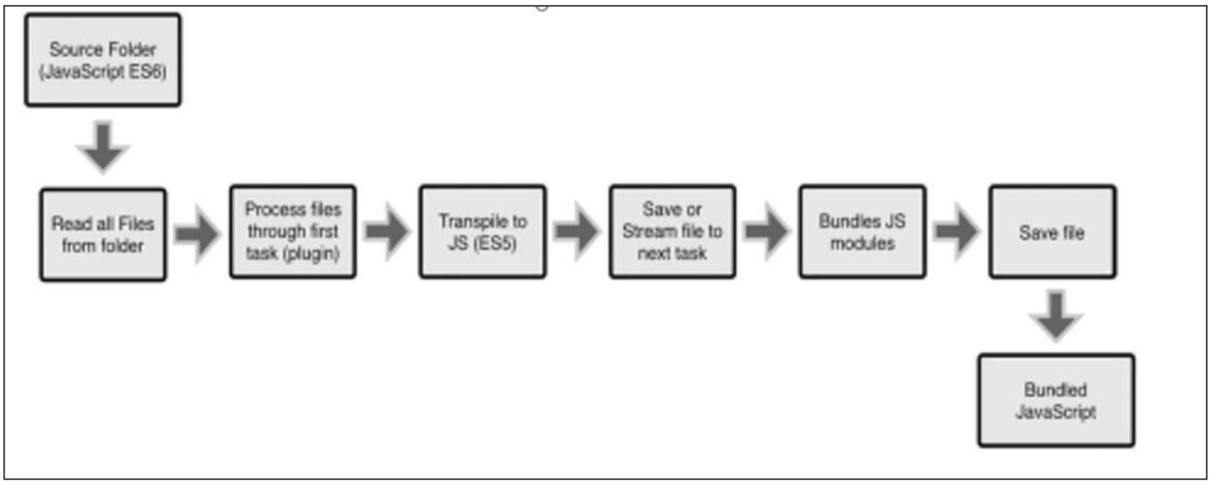

Source Folder (JavaScript ES6)
Read all Files from folder
Process files through first task (plugin)
Transpile to JS (ES5)
Save or Stream file to next task
Bundles JS modules
Save file
Bundled JavaScript

Figure 3.6: Gulp Workflow

2. Why use Gulp?

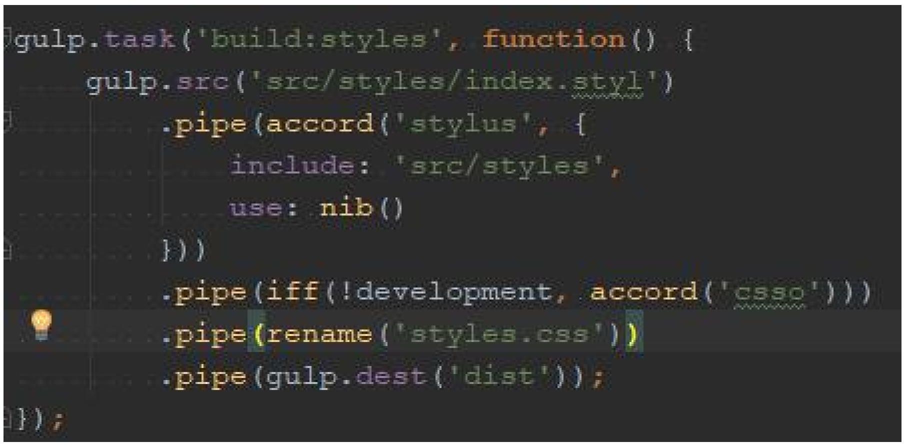
There are lots of advantages when the programmer uses gulp, such as merge files, file compression, vendor prefix, unit testing and code analysis and so on. However, there are two important reasons for why I use gulp in my project:

(1). Merge Files

The browser needs to download multiple JS files, and the browser has concurrency restrictions, so the performance of the web pages will be very poor, so we need to merge multiple files to reduce the number of files. Figure 3.7 is a piece of the code segment that comes from gulp file in my project. Using gulp to merge all CSS files (all .styylus file under.stype branch) and using Pipe() output the files to a file (style.js).

(2). File Compression

We know that the larger the file, the slower the download. The spaces, newlines inside JavaScript and CSS just in order to get a better understand to code for the programmer, but it has no effect for the computer. So Gulp can decrease files size by compressing the file and improve application performance.Figure 3.7 using Dest() function to compressing the SRC files, which can make sure no newlines and semicolon in the compressed file.



```
gulp.task('build:styles', function() {
    gulp.src('src/styles/index.styl')
        .pipe(accord('stylus', {
            include: 'src/styles',
            use: nib()
        }))
        .pipe(iff(!development, accord('csso')))
        .pipe(rename('styles.css'))
        .pipe(gulp.dest('dist'));
});
```

Figure 3.7: Code for merging CSS files and compress CSS files

3.4 Webpack

"Webpack is a module bundler. Webpack can take care of bundling alongside a separate task runner"(What is Webpack, Para1). What it does is analyze your project structure, then convert and package JavaScript modules and other extension languages (Scss, TypeScript) that the browser cannot run directly into a suitable format for the browser to use.

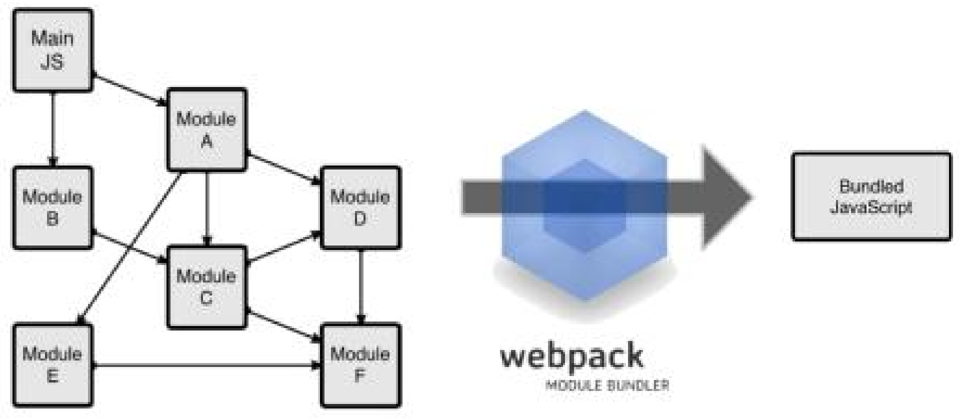


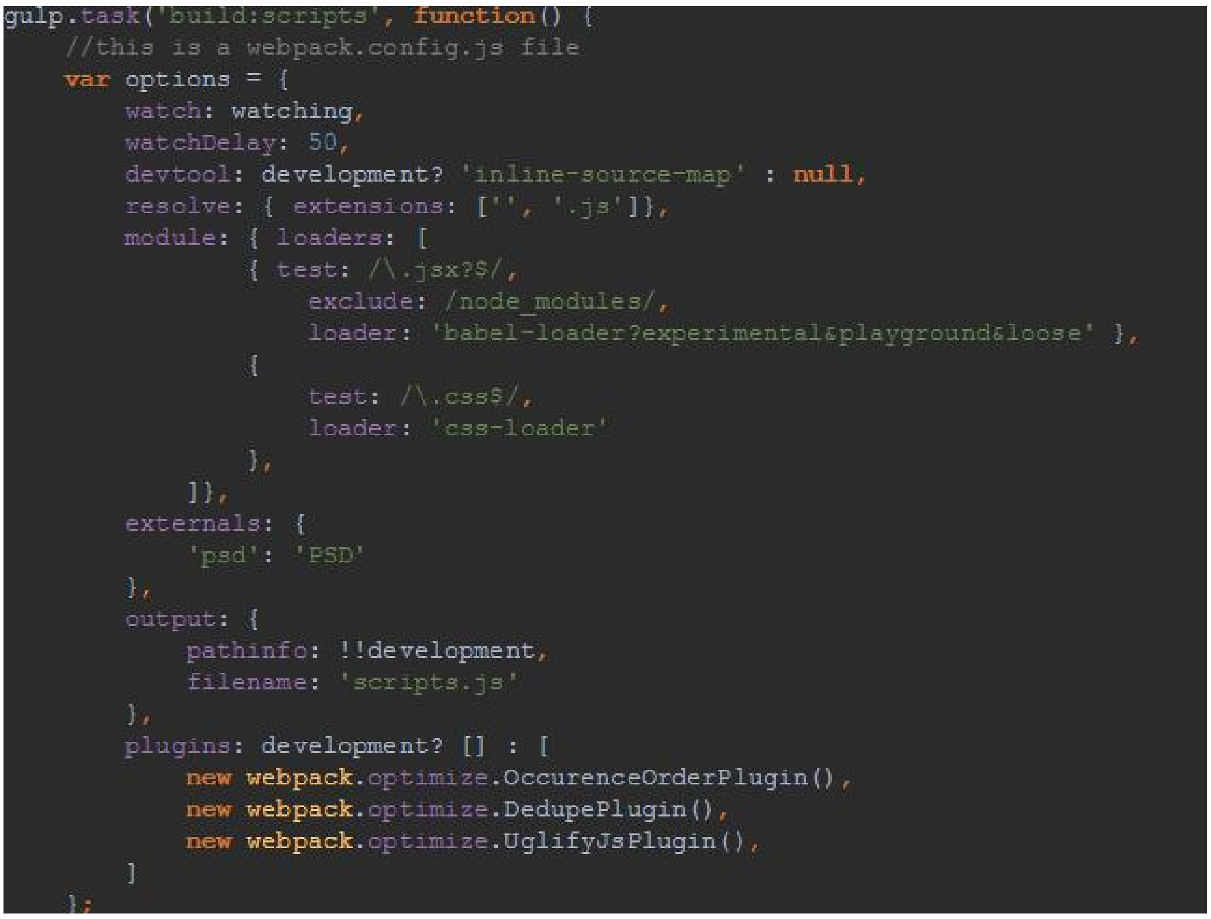
Figure 3.8: Webpack workflow

1. Webpack workflow

Figure 3.8 shows the webpack workflow, The way webpack works is taking the project as a whole, then giving master file (index.js), webpack will start from this file to find all dependencies of your project, use loaders to process them, and finally packaged as one(or more) browser-recognizable JavaScript files.

2. Why use Webpack

Gulp is task runner, webpack is module bundler. However, the line between bundler and task runner has become blurred thanks to community developed webpack plugins. Sometimes these plugins are used to perform tasks that are usually done outside of webpack, such as cleaning the build directory or deploying the build. My project use webpack plugins inside gulpfile and use React as the front-end framework, for React development, suitable loaders can convert JSX files used in React to JS files. Figure 3.8 displays a piece of the code segment that converts JSX to JS or convert style to CSS when using different loaders.



```
gulp.task('build:scripts', function() {
    //this is a webpack.config.js file
    var options = {
        watch: watching,
        watchDelay: 50,
        devtool: development? 'inline-source-map' : null,
        resolve: { extensions: ['', '.js']},
        module: { loaders: [
                { test: /\.jsx?$/,
                    exclude: /node_modules/,
                    loader: 'babel-loader?experimental&playground&loose' },
                {
                    test: /\.css$/,
                    loader: 'css-loader'
                },
            ]},
        externals: {
            'psd': 'PSD'
        },
        output: {
            pathinfo: !!development,
            filename: 'scripts.js'
        },
        plugins: development? [] : [
            new webpack.optimize.OccurenceOrderPlugin(),
            new webpack.optimize.DedupePlugin(),
            new webpack.optimize.UglifyJsPlugin(),
        ]
    };
```

Figure 3.9: Code of using webpack plugin in gulp file.

3.5 React

React is a JavaScript framework. React was originally create by engineers at Facebook to solve the challenges involved when developing complex user interfaces with data sets that change over time. Figure 3.1.6.1 is a life cycle for initial a component. It passed component “will receive props”, “should Component Update”, “component will update”, and “render (return a component)”, and “component did update”.

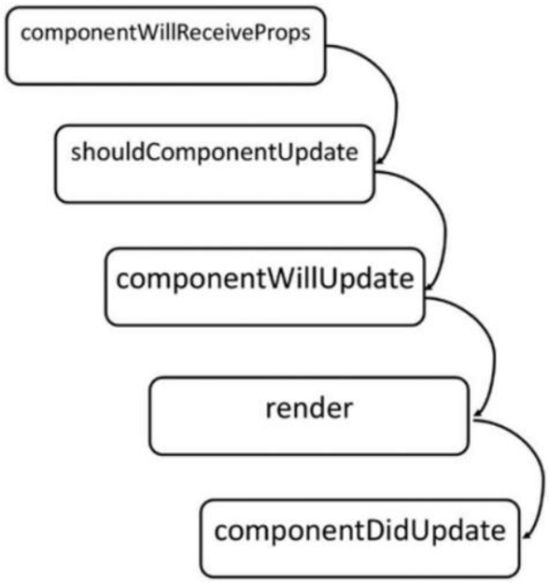
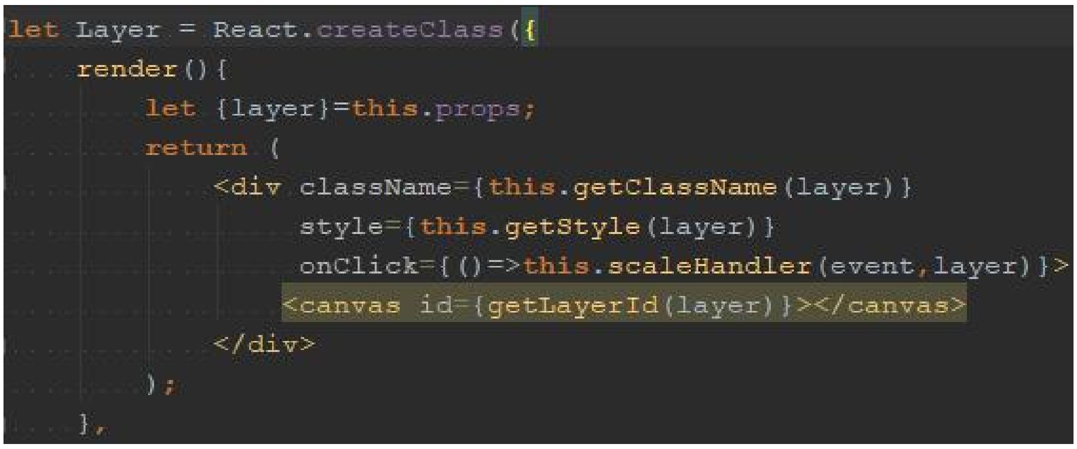


Figure 3.10: Life cycle for initial a component

1. **CreateClass**

A ReactClass is created by the React.createClass() method. The createClass method will create a new component class in React. createClass can be created with an object, which must have a render() function. Every ReactClass must have render method which returns a new component. Figure 3.11 presents how to create a class and it must includes render function.

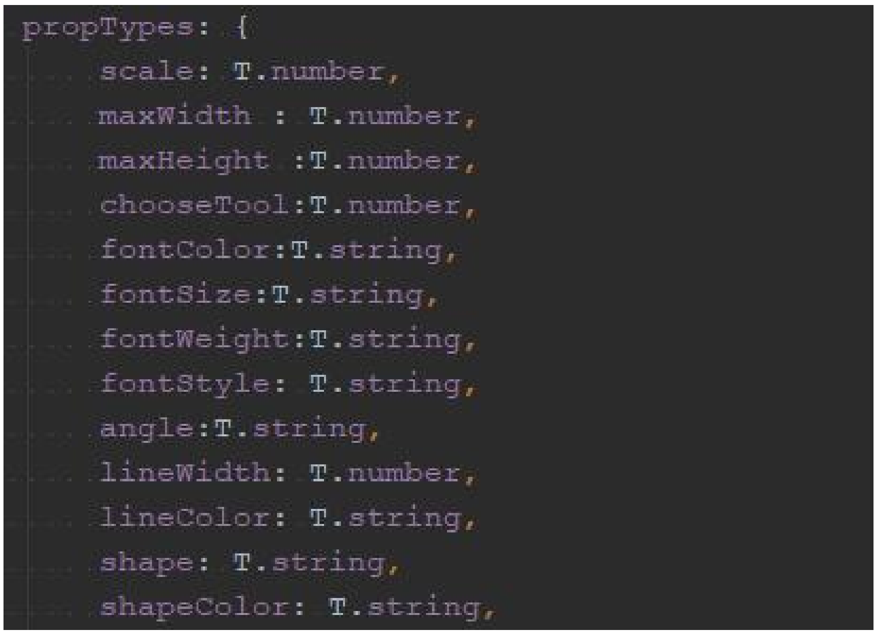


```
let Layer = React.createClass({
    render(){
        let {layer}=this.props;
        return (
            <div className={this.getClassName(layer)}
                 style={this.getStyle(layer)}
                 onClick={()=>this.scaleHandler(event,layer)}>
                <canvas id={getLayerId(layer)}></canvas>
            </div>
        );
    },
```

Figure 3.11: Create class using react

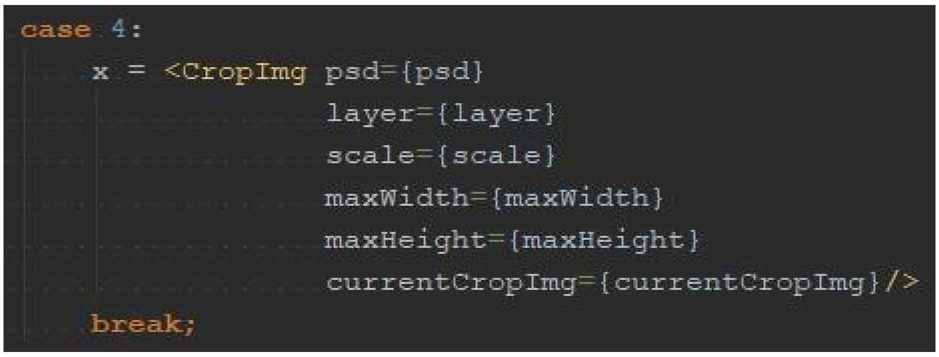
2. **The Props Parameter and propTypes**

The props parameter is a JavaScript object passed from a parent element to a child element, Figure 3.13 shows how to pass a parameter from parent element to child element while Figure 3.14 shows how to get the parameter in child element using props, propTypes (shown in Figure 3.12) is an object that can ass checks for types for each of the props passed to the component.



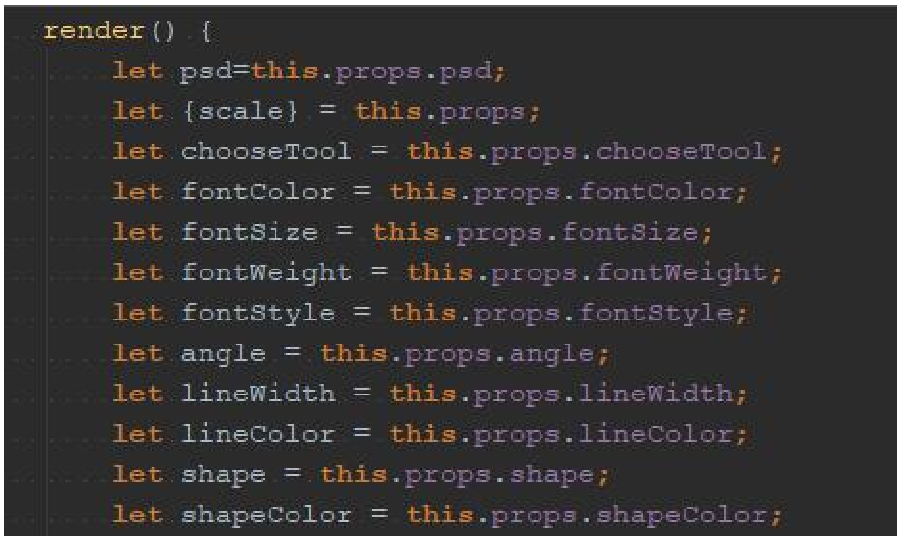
```
propTypes: {
    scale: T.number,
    maxWidth : T.number,
    maxHeight :T.number,
    chooseTool:T.number,
    fontColor:T.string,
    fontSize:T.string,
    fontWeight:T.string,
    fontStyle: T.string,
    angle:T.string,
    lineWidth: T.number,
    lineColor: T.string,
    shape: T.string,
    shapeColor: T.string,
```

Figure 3.12: Code of variable declaration using propTypes.



```
case 4:
    x = <CropImg psd={psd}
                 layer={layer}
                 scale={scale}
                 maxWidth={maxWidth}
                 maxHeight={maxHeight}
                 currentCropImg={currentCropImg}/>
    break;
```

Figure 3.13: Code of pass parameters.

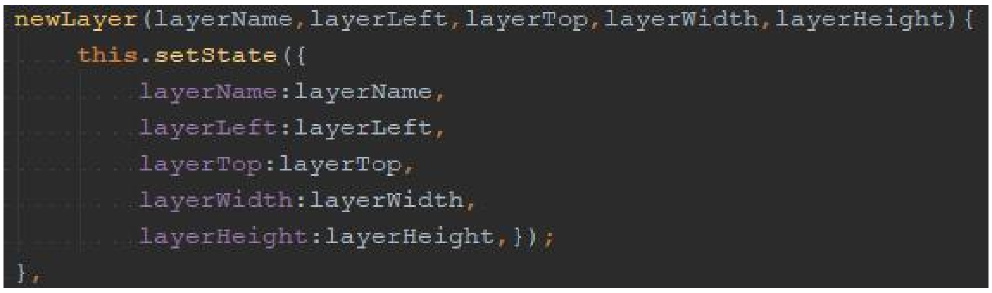


```
render() {
    let psd=this.props.psd;
    let {scale} = this.props;
    let chooseTool = this.props.chooseTool;
    let fontColor = this.props.fontColor;
    let fontSize = this.props.fontSize;
    let fontWeight = this.props.fontWeight;
    let fontStyle = this.props.fontStyle;
    let angle = this.props.angle;
    let lineWidth = this.props.lineWidth;
    let lineColor = this.props.lineColor;
    let shape = this.props.shape;
    let shapeColor = this.props.shapeColor;
```

Figure 3.14: Code of obtain parameter value through props.

3. **State**

The state is a very important concept and it is set on each component because it is initialized and is also altered throughout the life cycle of a component. When you change an object's state, the setState is called, the new object will queuing into the React update queue, which is the mechanism React uses to control when this is changed. Once the state is ready to change, the new state object will be merged with the remainder of the components state. Figure 3.15 is an example in my code when user adds a new layer, the state of the layer name, position, size will be changed, once setState is triggered, the code of layer name, position, size in HTM L and CSS will be changed automatically as well.

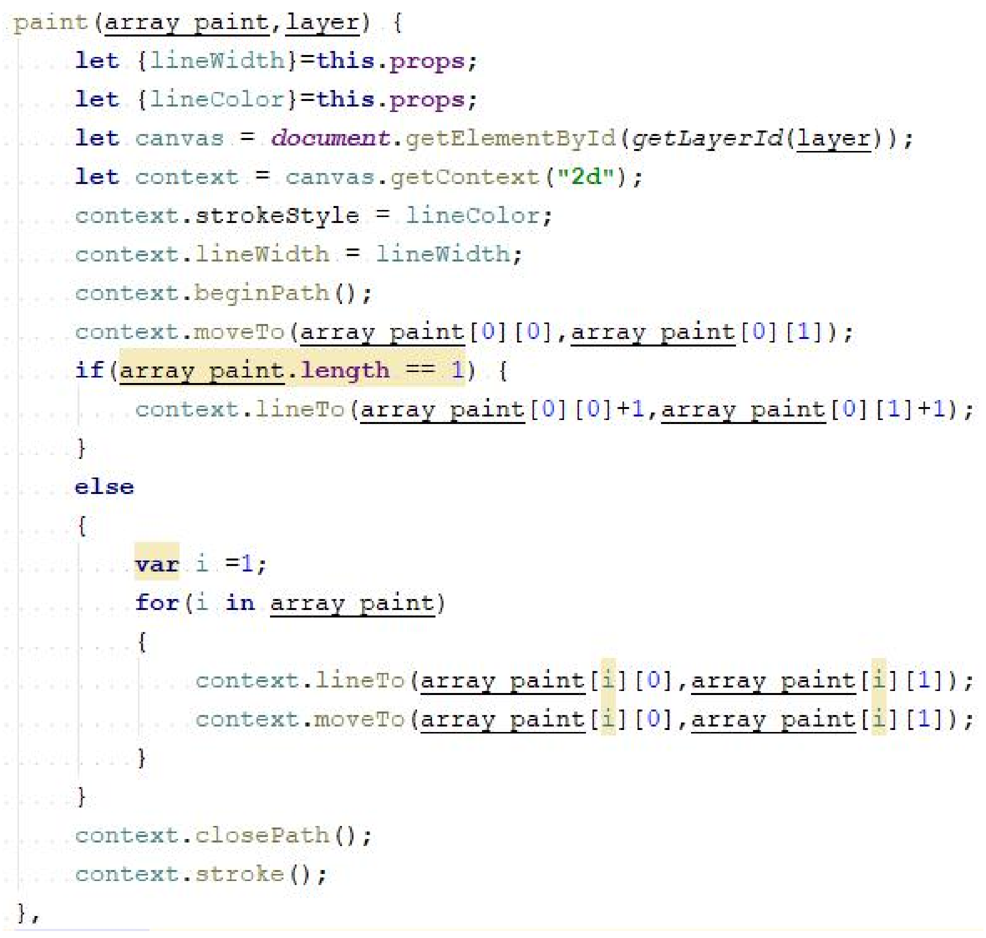


```
newLayer(layerName,layerLeft,layerTop,layerWidth,layerHeight){
    this.setState({
        layerName:layerName,
        layerLeft:layerLeft,
        layerTop:layerTop,
        layerWidth:layerWidth,
        layerHeight:layerHeight,});
},
```

Figure 3.15: Core function of the set state.

3.6 Canvas

"HTML5 canvas is an immediate mode bit-mapped area of the screen that can be manipulated with JavaScript. Immediate mode refers to the way the canvas renders pixels on the screen" (html5_canvas, page1). The basic HTML5 canvas API include a 2D context, which allows the programmer to draw various shapes render text and display image directly onto a defined area of the browser window. The programmer can apply colors; rotations; alpha transparencies; pixel manipulation; draw shapes, text, and image and so forth. Figure 3.16 is a piece of code segment about how to paint on canvas.The function of lineTo() is the start point of the line, and the moveTo() is the end point of the line, both of them have two essential parameters, it's X coordinate and Y coordinate. Two reasons for why use canvas in all layer image edition: (1) Canvas is a rich image processing API, (2) Canvas can generate a new image through creating URL so that the image state can be constantly updated.



```
paint(array_paint,layer) {
    let {lineWidth}=this.props;
    let {lineColor}=this.props;
    let canvas = document.getElementById(getLayerId(layer));
    let context = canvas.getContext("2d");
    context.strokeStyle = lineColor;
    context.lineWidth = lineWidth;
    context.beginPath();
    context.moveTo(array_paint[0][0],array_paint[0][1]);
    if(array_paint.length == 1) {
        context.lineTo(array_paint[0][0]+1,array_paint[0][1]+1);
    }
    else
    {
        var i =1;
        for(i in array_paint)
        {
            context.lineTo(array_paint[i][0],array_paint[i][1]);
            context.moveTo(array_paint[i][0],array_paint[i][1]);
        }
    }
    context.closePath();
    context.stroke();
},
```

Figure 3.16: Code for painting using Canvas

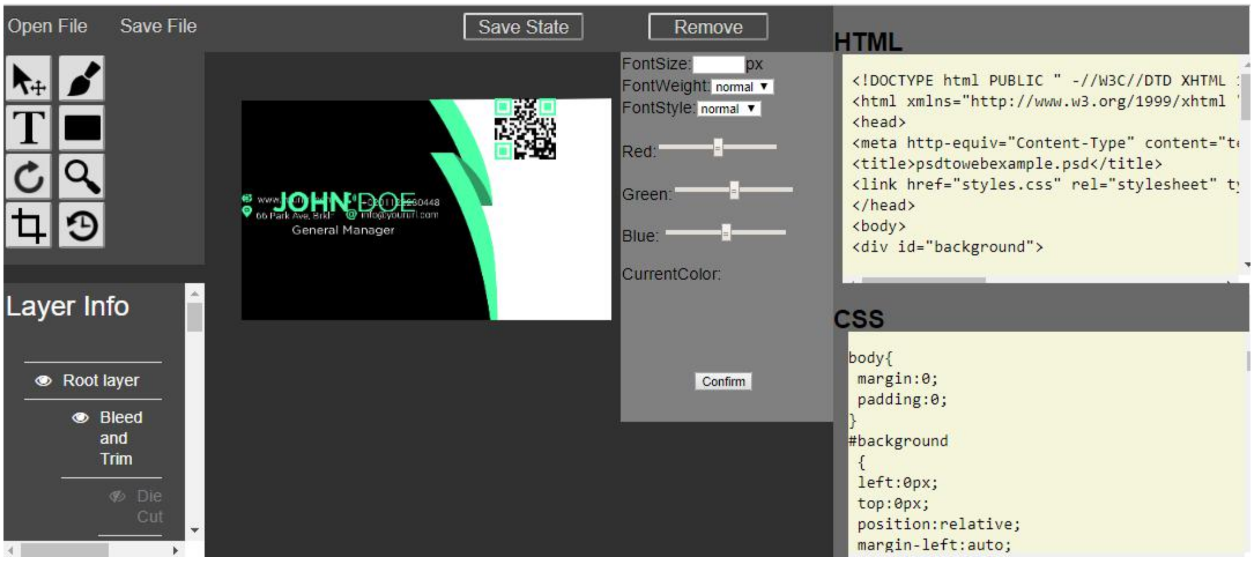
4. Implementation

4.1 Introduction

This chapter will present a summary of the tool (cf. Section 4.2) and then detail how each function was implemented (cf. Section 4.3), next, summarize some problems I encountered during developing my project(cf. Section 4.4).

4.2 Summary

This product is a pure front-end project using React.js as a framework (cf. Figure 4.1). On the left-hand side of the user interface the user can preview and edit the PSD image.   On the right-hand side, the user can see the corresponding HTML and CSS code.   It uses the Canvas API and a PSD parser (that was downloaded from GitHub) to present the image.



```
<!DOCTYPE html PUBLIC " -//W3C//DTD XHTML
<html xmlns="http://www.w3.org/1999/xhtml
<head>
<meta http-equiv="Content-Type" content="t
<title>psdtowebexample.psd</title>
<link href="styles.css" rel="stylesheet" t
</head>
<body>
<div id="background">
```

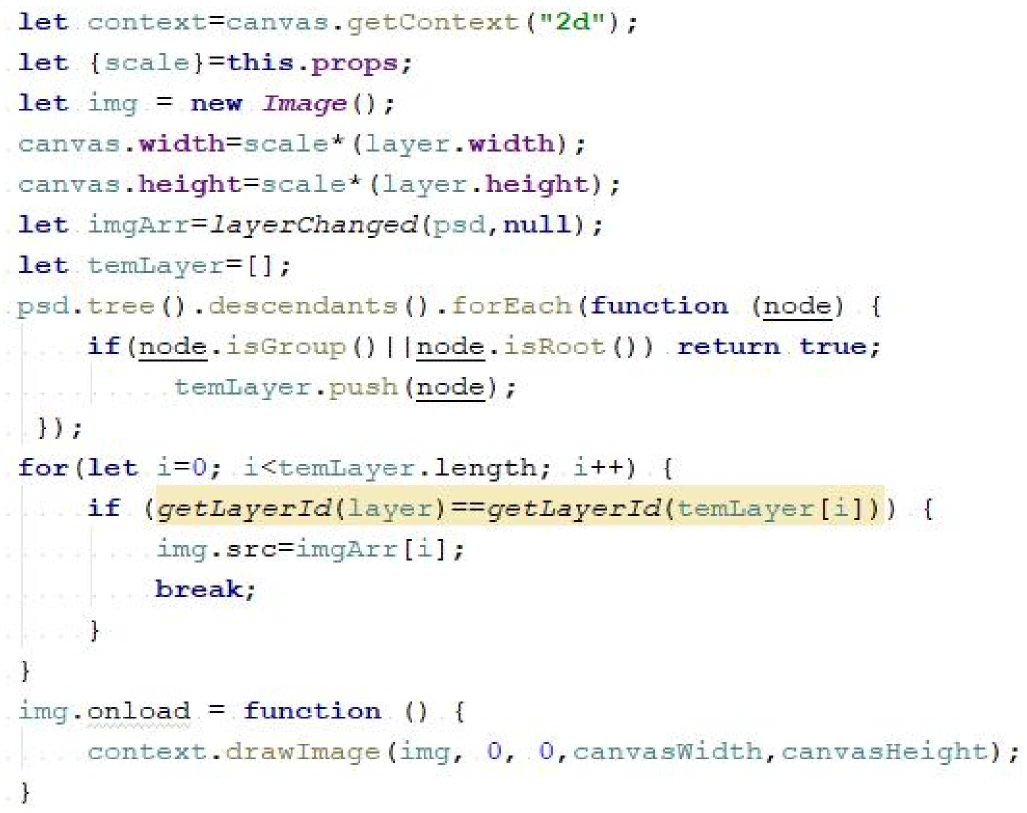
```
body{
 margin:0;
 padding:0;
}
#background
 {
 left:0px;
 top:0px;
 position:relative;
 margin-left:auto;
```

Figure 4.1: GUI of the developed tool

4.3 Function Implementation

(1). Upload a PSD file

The PSD file upload was achieved using the drawImage() function of the Canvas API. The PSD parser gets the base64 of each layer and creates a URL for each layer and the drawImage() function then displays the image so that the user can interact with it.   The code written to upload the image is shown in Figure 4.2.

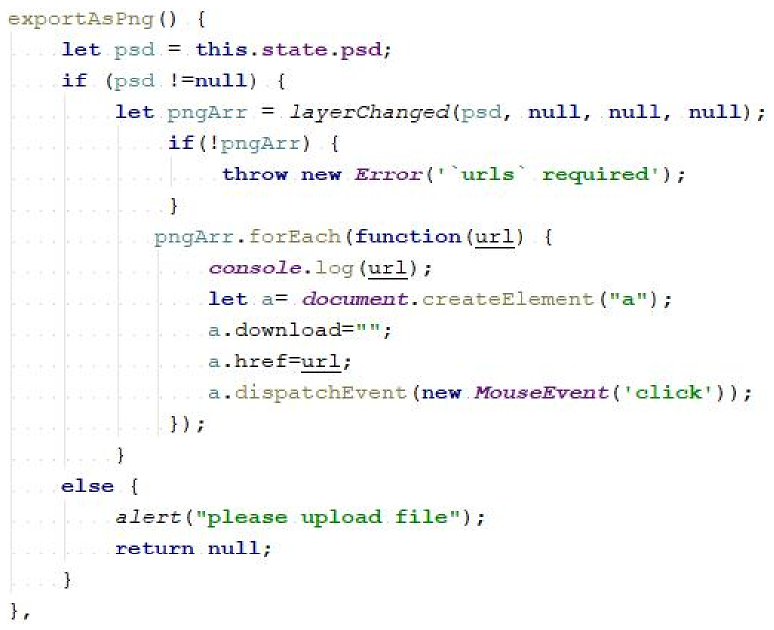


```
let context=canvas.getContext("2d");
let {scale}=this.props;
let img = new Image();
canvas.width=scale*(layer.width);
canvas.height=scale*(layer.height);
let imgArr=layerChanged(psd,null);
let temLayer=[];
psd.tree().descendants().forEach(function (node) {
    if(node.isGroup()||node.isRoot()) return true;
        temLayer.push(node);
});
for(let i=0; i<temLayer.length; i++) {
    if (getLayerId(layer)==getLayerId(temLayer[i])) {
        img.src=imgArr[i];
        break;
    }
}
img.onload = function () {
    context.drawImage(img, 0, 0,canvasWidth,canvasHeight);
}
```

Figure 4.2: Code to Upload PSD file

(2).Split it into PNG files

A PSD file that was uploaded to my tool can split a PSD file to PNG files, and then the user can download these PNG images and used it to HTML code website component. Just like logo image, button image, table image and so on. Figure 4.3 is a core function of batch image download, using <a> element to download all images at one time. Figure 4.4 shown PNG images that were downloaded from this tool.



```
exportAsPng() {
    let psd = this.state.psd;
    if (psd !=null) {
        let pngArr = layerChanged(psd, null, null, null);
            if(!pngArr) {
                throw new Error('`urls` required');
            }
          pngArr.forEach(function(url) {
              console.log(url);
              let a= document.createElement("a");
              a.download="";
              a.href=url;
              a.dispatchEvent(new MouseEvent('click'));
          });
        }
    else {
        alert("please upload file");
        return null;
    }
},
```

Figure 4.3: Code to Split PSD and Download PNG images

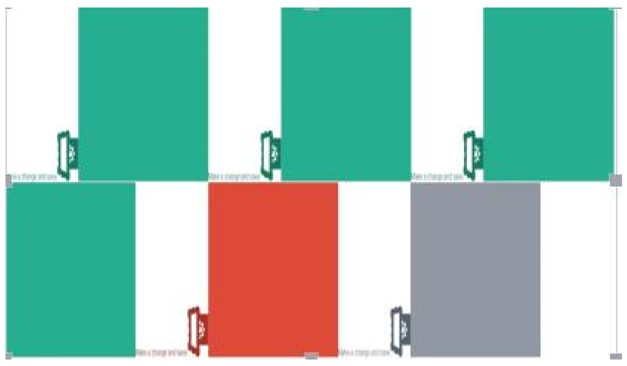
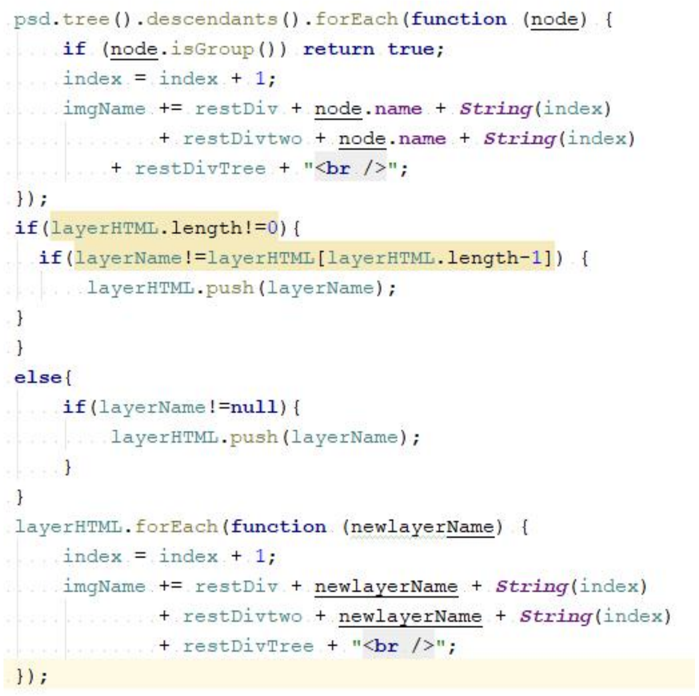


Figure 4.4: PNG images

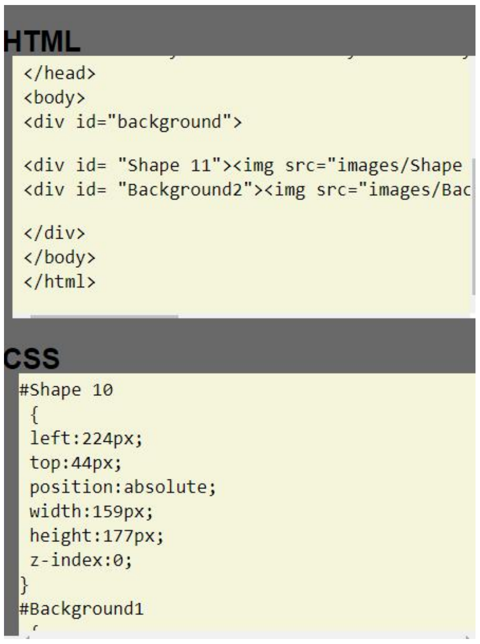
(3). Convert it to HTML and CSS

HTML and CSS code conversion functionality is mainly divide two sections: HTML section and CSS section. Figure 4.4 display the HTML and CSS code that is converted by this tool, HTML code basically consist of <DIV>elements and CSS code consists with each layers' information.   Figure 4.5 is a core function of HTML conversion, this function can obtain all layers information and extract the useful layer information (such as: layer name and layer image) to generate HTML code. (CSS conversion function is similar with HTML conversion).



```
psd.tree().descendants().forEach(function (node) {
    if (node.isGroup()) return true;
    index = index + 1;
    imgName += restDiv + node.name + String(index)
            + restDivtwo + node.name + String(index)
        + restDivTree + "<br />";
});
if(layerHTML.length!=0){
  if(layerName!=layerHTML[layerHTML.length-1]) {
      layerHTML.push(layerName);
  }
}
else{
    if(layerName!=null){
        layerHTML.push(layerName);
    }
}
layerHTML.forEach(function (newlayerName) {
    index = index + 1;
    imgName += restDiv + newlayerName + String(index)
            + restDivtwo + newlayerName + String(index)
            + restDivTree + "<br />";
});
```

Figure 4.5: Code to Generate HTML



HTML

```
</head>
<body>
<div id="background">

<div id= "Shape 11"><img src="images/Shape
<div id= "Background2"><img src="images/Bac

</div>
</body>
</html>
```

CSS

```
#Shape 10
 {
 left:224px;
 top:44px;
 position:absolute;
 width:159px;
 height:177px;
 z-index:0;
}
#Background1
```

Figure 4.6: HTML and CSS generation

(4). Layer hidden/visible

The layer information area on left-hand side of the screen can change the layer state (hide or visible). For example, if users click on the eye icon, it displays in Figure 4.8, the red layer is hidden, which showed in Figure 4.9. If users click the eye icon again, the red layer will appear. Figure 4.7 represents that red layer appears when clicking on the eye icon.

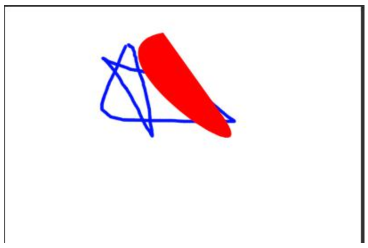
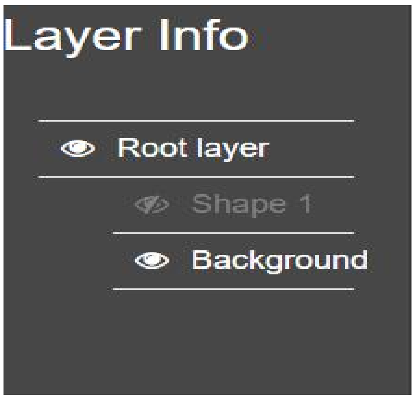


Figure 4.7: Image of making the red layer visible



Layer Info

Root layer

Shape 1

Background

Figure 4.8: Layer information component

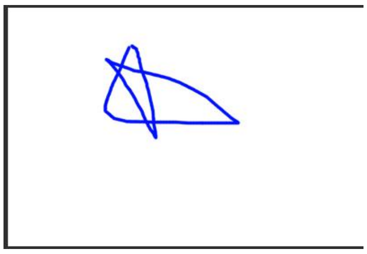


Figure 4.9: Image of hiding the red shape layer

(5). Move Layer

Move layer is the function that allows the user to move a specific layer when users click on a specific layer of the uploaded PSD file. If users use move tool, the layer position of PSD file will be changed, an example is shown in Figure 4.10, which shows the PSD file contrast of before moving the layer and after moving layer. In addition, the CSS code will be changed as well, it presents in Figure 4.11, which shows the CSS code contrast of before moving layer and after moving layer. The Figure 4.12 shown the key function for the implementation of moving the layer, the layer moving position can be obtained through mouse down, mouse move and mouse up event.

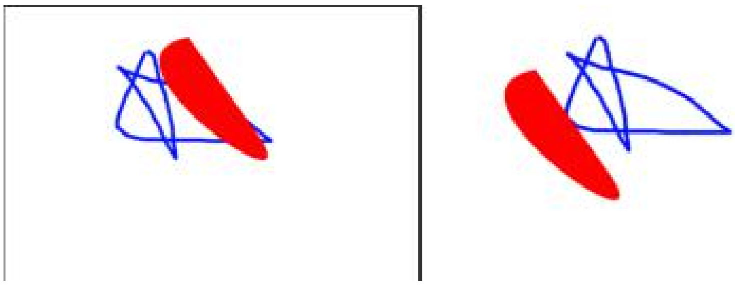
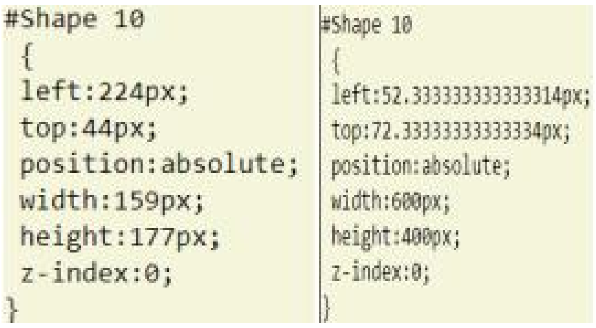


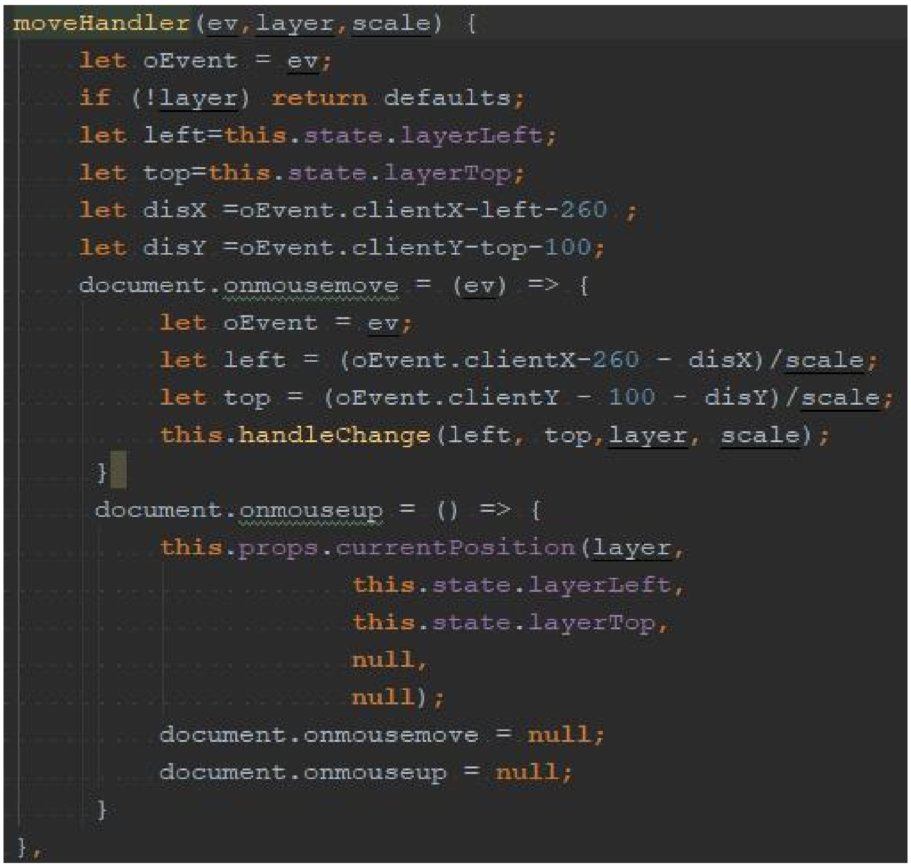
Figure 4.10: PSD contrast of before moving layer and after moving layer



```
#Shape 10
 {
 left:224px;
 top:44px;
 position:absolute;
 width:159px;
 height:177px;
 z-index:0;
}
```

```
#Shape 10
 {
 left:52.333333333333314px;
 top:72.33333333333334px;
 position:absolute;
 width:600px;
 height:400px;
 z-index:0;
}
```

Figure 4.11: CSS code contrast of before moving layer and after moving layer



```
moveHandler(ev,layer,scale) {
    let oEvent = ev;
    if (!layer) return defaults;
    let left=this.state.layerLeft;
    let top=this.state.layerTop;
    let disX =oEvent.clientX-left-260 ;
    let disY =oEvent.clientY-top-100;
    document.onmousemove = (ev) => {
         let oEvent = ev;
         let left = (oEvent.clientX-260 - disX)/scale;
         let top = (oEvent.clientY - 100 - disY)/scale;
         this.handleChange(left, top,layer, scale);
     }
     document.onmouseup = () => {
         this.props.currentPosition(layer,
                     this.state.layerLeft,
                     this.state.layerTop,
                     null,
                     null);
         document.onmousemove = null;
         document.onmouseup = null;
     }
},
```

Figure 4.12: Code of move layer

(6). Scale Layer

Scale layer means that user can change the resize the layer when they click on a specific layer of the uploaded PSD file, it is showed in Figure 4.13 and the red layer is smaller than the previous one. The width and height attribute in CSS code are obviously modified after resizing the red layer as Figure 4.14.

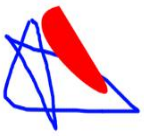
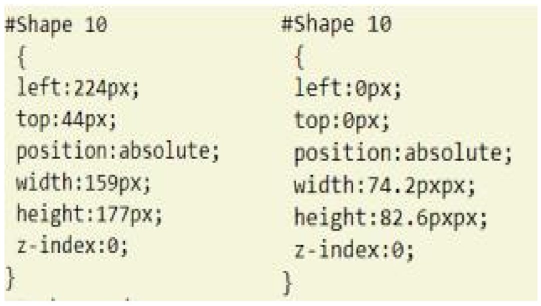


Figure 4.13: Image after resizing the red layer



```
#Shape 10                    #Shape 10
 {                             {
 left:224px;                   left:0px;
 top:44px;                     top:0px;
 position:absolute;            position:absolute;
 width:159px;                  width:74.2pxpx;
 height:177px;                 height:82.6pxpx;
 z-index:0;                    z-index:0;
}                             }
```

Figure 4.14: CSS code contrast of before resizing layer and after resizing the layer

(7). Rotating tool

The user can rotate the layer when they click on rotate layer button, a specific layer will be rotated if they click on the layer. If the user saves the state, the layers' states will become the left image of Figure 4.1.5.

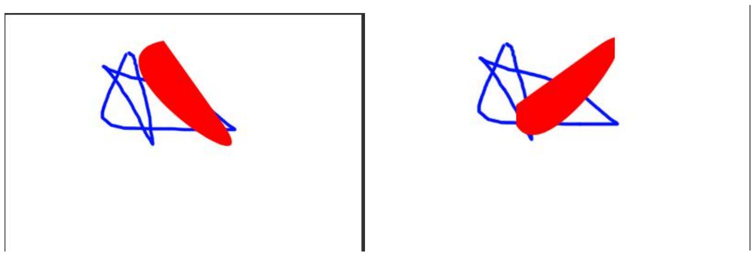


Figure 4.15: Image contrast of before rotating red layer and after rotating red layer.

(8). Painting tool

Paint layer means the user can draw anything on the PSD file, the function allows users to select various options such as the pen color, pen width and so forth. If the user saves the state, this layer state will be changed to Figure 4.16.

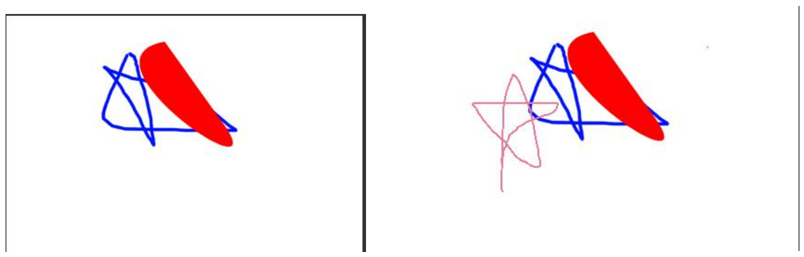


Figure 4.16: Image contrast of before painting and after painting.

(9). Shape tool

Shape tool means that user can add a new layer through adding a shape to PSD file, which shown as Figure 4.17; the shape options can be a line, circle, square, rectangle, round rectangle and so on. When user adds a shape to PSD file and save this state, the HTML and CSS will be changed, Figure 4.18 show an example of the "shape03" was added to HTML and CSS automatically when user adds a square to PSD file and save the state.

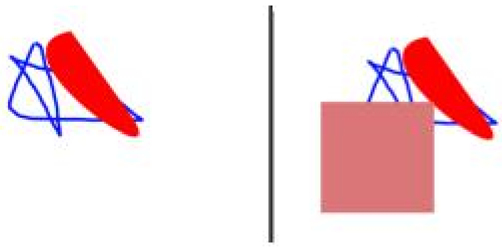
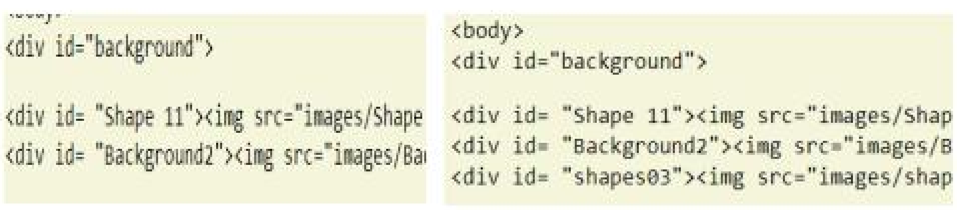


Figure 4.17: Image contrast of before adding new shape and after adding a new shape.



```
<div id="background">

<div id= "Shape 11"><img src="images/Shape
<div id= "Background2"><img src="images/Ba
```

```
<body>
<div id="background">

<div id= "Shape 11"><img src="images/Shap
<div id= "Background2"><img src="images/B
<div id= "shapes03"><img src="images/shap
```

Figure 4.18: HTML and CSS code after adding a new shape.

(10). Text tool

Text tool means that user can add a new layer through adding a text to PSD file, which shown as Figure 4.19; The text attribute options have text color, font weight, font family, font style and so forth. When user adds a new text to PSD file and save this state, the HTML and CSS will be modified, Figure 4.20 shows an example of the “text03” was added to HTML and CSS automatically when user adds a “HELLO WORLD” text layer to PSD file and save this state. During the implementation of text tool, mouse down, mouse move and mouse up events decide the text area position and size, the Figure 4.21 presents the process of three events, and then Figure 4.22 shows how to achieve filling text on canvas using fillText() function and save this state.

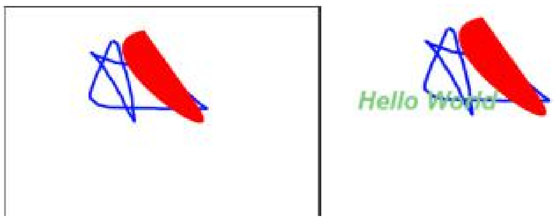
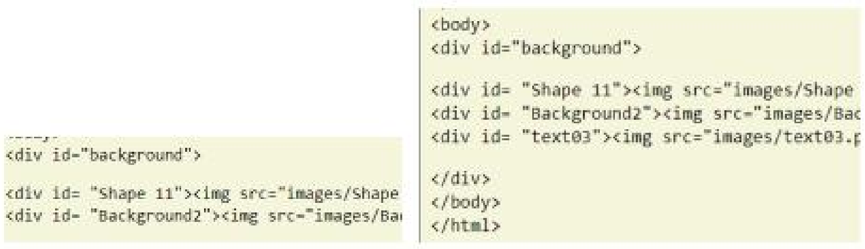


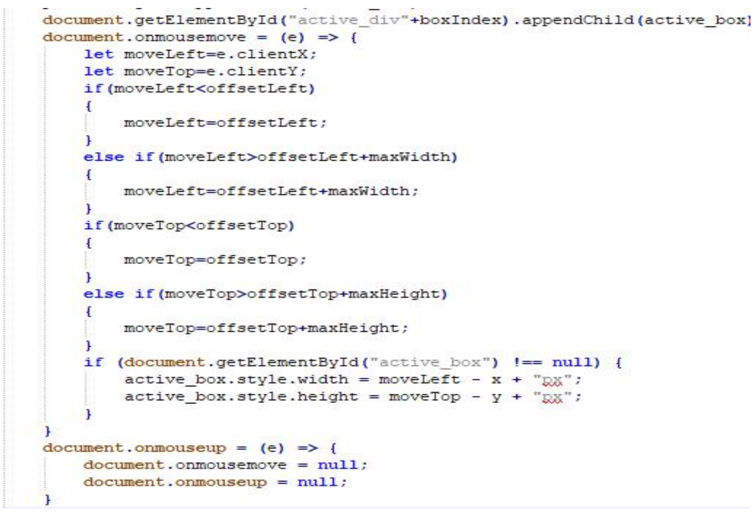
Figure 4.19: Image contrast of before adding new text and after adding new text.



```
<div id="background">

<div id= "Shape 11"><img src="images/Shape
<div id= "Background2"><img src="images/Ba
```

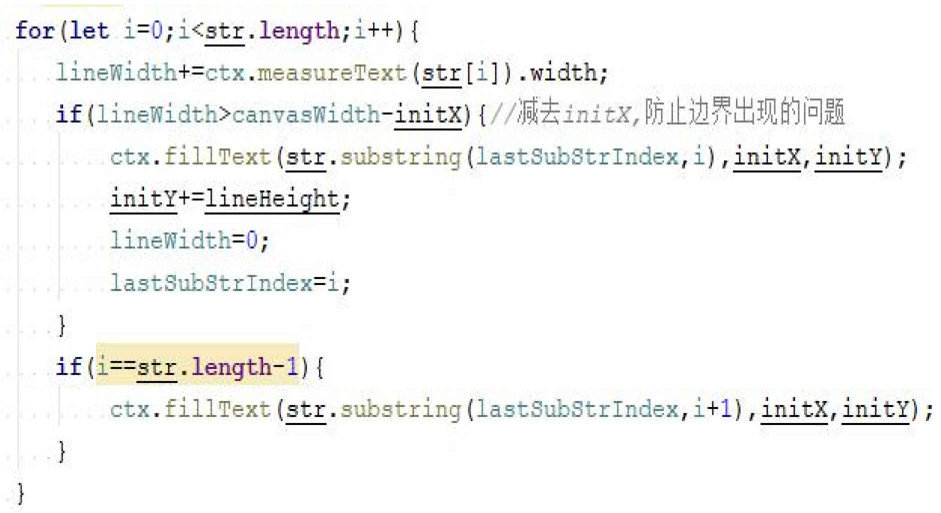
```
<body>
<div id="background">

<div id= "Shape 11"><img src="images/Shape
<div id= "Background2"><img src="images/Bac
<div id= "text03"><img src="images/text03.p

</div>
</body>
</html>
```

Figure 4.20: HTML and CSS Code after adding new text.



```
document.getElementById("active_div"+boxIndex).appendChild(active_box)
document.onmousemove = (e) => {
    let moveLeft=e.clientX;
    let moveTop=e.clientY;
    if(moveLeft<offsetLeft)
    {
        moveLeft=offsetLeft;
    }
    else if(moveLeft>offsetLeft+maxWidth)
    {
        moveLeft=offsetLeft+maxWidth;
    }
    if(moveTop<offsetTop)
    {
        moveTop=offsetTop;
    }
    else if(moveTop>offsetTop+maxHeight)
    {
        moveTop=offsetTop+maxHeight;
    }
    if (document.getElementById("active_box") !== null) {
        active_box.style.width = moveLeft - x + "px";
        active_box.style.height = moveTop - y + "px";
    }
}
document.onmouseup = (e) => {
    document.onmousemove = null;
    document.onmouseup = null;
}
```

Figure 4.21: Code of adding new text to PSD file.



```
for(let i=0;i<str.length;i++){
    lineWidth+=ctx.measureText(str[i]).width;
    if(lineWidth>canvasWidth-initX){//减去initX,防止边界出现的问题
        ctx.fillText(str.substring(lastSubStrIndex,i),initX,initY);
        initY+=lineHeight;
        lineWidth=0;
        lastSubStrIndex=i;
    }
    if(i==str.length-1){
        ctx.fillText(str.substring(lastSubStrIndex,i+1),initX,initY);
    }
}
```

Figure 4.22: Code of adding new text by using Canvas.

(11). Crop Image

This function mainly used for cropping the whole PSD file or each element of PSD file, Figure 4.23 shows a cropped image, Figure 4.24 can save and download the cropped image through <a> label and download attribute.

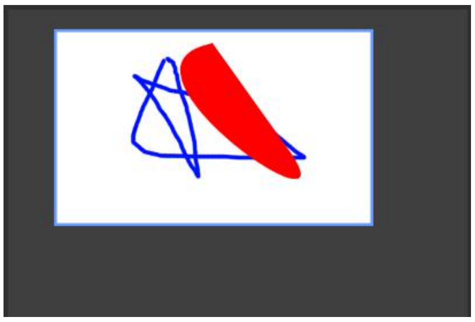


Figure 4.23: Image of crop PSD file

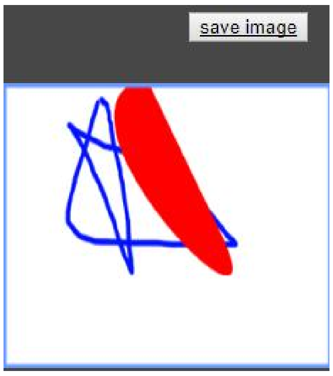


Figure 4.24: Save as an image

(12). History Record

History Record has two functions: firstly, user can view what tools they have used, Figure 4.25 shown the process of what tools users have used, like first use type tool, then use the move tool and so on. Second, user can undo the state and return to a previous state. Like the example of Figure 4.36, users can return to the state before painting the pentagram when they click on the first paint tool in history table.So history can record and undo layer state.

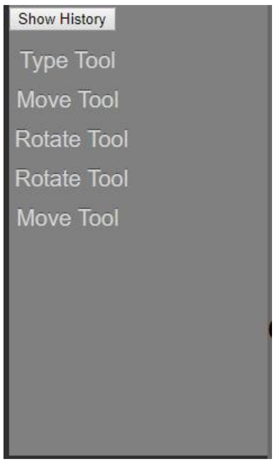


Figure 4.25: History table component

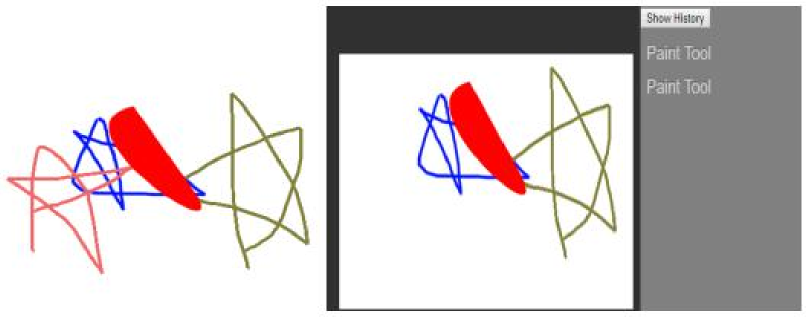


Figure 4.26: Image contrast of before undo and after undo

4.4 Problem Encountered

During the implementation, I encountered two big problems

(1) The first program is about "add new layer" function, such as shape tool and text tool. When I try to add a new layer, I found the PSD parser does not have any interfaces to add a new layer to the existing PSD file. and each layer structure has hundreds of attribute. The way I solve this problem is using canvas to replace the layers, which means I can save the "new layer state" and convert it to HTML and CSS code. But I can not save it to PSD file and export the PSD file.

(2) The second problem is about "crop tool" function, in the beginning, I researched the way can implement crop function is using react-cropper API, but there always have an error about "cannot read getImageDateAttribute". Finally, I achieved this function without using any API.

5. Testing and Evaluation

5.1. Introduction

This chapter will introduce the functional testing (cf . Section 5.2) and the testing process and evaluation (cf. Section 5.3) of converter condition.

5.2. Functional Testing

Functional Testing also calls black box testing, it tests each function can be used normally by testing every function. In the testing, it regards the program as a block box without considering the internal structure and internal characteristics of the program at all. what the testing focuses on is if the program produces correct output information when users input data. Therefore, the black box test focuses on the external structure of the program and it does not consider the internal logical structure. It mainly tests software interfaces and software functions. Due to too many functions of this tool, the author just picks some of the functions to testing.

(1) Move Tool

Test 1:

Input Event: Users click on a specific layer and drag it.

Expected Result: Layer position and CSS code will be changed.

Result: Layer can follow the mouse move event to move to a correct position and CSS code will be changed as well when mouse up event is triggered.

Verdict: Successful test.

Test 2:

Input Event: Users click on the save state button.

Expected Result: The current layer position will be saved and PSD file will be updated.

Result: If click on the save state button and then use another tool, the state is the state after moving file, if users don’t click on save state button, then the layer state will return last state (before moving layer).

Verdict: Successful test.

(2) Crop Tool

Test 1:

Input Event: Users click on the right-bottom corner and drag it.

Expected Result: The crop image size will be changed, but the pixel of cropped imaged didn't change.

Result: The crop image size changes as mouse moves and image pixel will not be changed.

Verdict: Successful test.

Test 2:

Input Event: Users click on the crop area (except the image corner).

Expected Result: The crop area will move as mouse move, if the mouse moves out of the image area (for example, the position of right greater than the rightmost limit, the position of crop area will default to rightmost position), the position of crop area will default to the biggest (rightmost, bottommost) or smallest (leftmost, topmost) position.

Result: The cropped image moves when mouse moved, the cropped image always inside the image area. When the mouse moves out of image area, the cropped image only moves to the biggest or smallest limit position (same with expected result).

Verdict: Successful test.

Test 3:

Input Event: Users click on save crop image.

Expected Result: The cropping image will be downloaded.

Result: The PNG format of crop image is downloaded.

Verdict: Successful test.

(3) Type Tool

Test 1:

Input Event: Users click on the mouse and move the mouse on the preview container area.

Expected Result: The text area size will be changed as mouse position changes, if there is an existing but not saved or removed text on the preview container area, users are not allowed to create the second text area, Besides, the text area size cannot exceed the preview container size. If the mouse moves out of preview container, the size of text area default to the size of preview container.

Result: The text area size is changed when the user moves mouse, and text can be saved or removed when users click on save or remove button. The user can generate second text area when first text is saved or removed, otherwise, they cannot create a new text. The size of the text area is equal to the preview container size if the mouse moves out of preview container area.

Verdict: Successful test.

Test 2:

Input Event: Users click the save state/remove button.

Expected Result: If users click on save state, the text will be saved, and new HTML and CSS code will be added to the HTML and CSS area. If users click on remove button, the text will be removed.

Result: Users click on save sate button, the text state can be saved and the code about the text is added to HTML/CSS area automatically when user click on remove button, the text is removed and HTML/CSS code did not change.

Verdict: Successful test.

5.3. Conclusion

The result of black box testing suggests that this tool can be used for editing the PSD file and generating HTML/CSS code with these basic functions successfully.

6. Conclusion

The image to HTML/CSS converter can help designers and programming beginners to create a static web page without writing any code. In general, this tool has achieved a basic PSD file edition function and HTML/CSS conversion function, but it still can be improved by adding richer functions. For example, it can introduce PNG or JPG images as a new layer (web page element), or add a basic component such as button, label, table, list etc. When these components is added to PSD file as new layers, and then HTML will add <button> label or <table> label or <textarea> label and so forth.

Finally, in the process of developing this application, the author gained rich experience in developing a front-end application. The author knows about node.JS, react.JS, webpack technologies, and applies this knowledge to the project. Developing this project also improve the author's programming skill and self-research/self-study ability. This report also lets the author report writing and evaluation skills have a great enhancement. Overall, the completion of this project let her confident in developing a website or documenting a successful practical application in the future.

Reference:

PSD2WEB [online], available: http://psdtoweb.de/ [accessed 05 Oct 2017]

PSD2HTML converter [online], available: https://psd2htmlconverter.com/en/[accessed 05 Oct 2017]

Photopea editor [online], available: https://www.photopea.com/[accessed 06 Oct 2017]

Sverlana.G, 'Website Development Process: Full Guide in 7 Steps' [online], 14/12/2015, available: https://xbsoftware.com/blog/website-development-process-full-guide/ [accessed 10 Oct 2017]

'A new description of Role & Responsibilities' [online], available: http://www.diffily.com/articles/webteamroles.html[accessed 10 Oct 2017]

'Why is website design so important' [online], available: http://sunriseprowebsites.com/website-design-important/[accessed 10 Oct 2017]

'Why website design is important' [online], available: https://www.bopdesign.com/bop-blog/2011/09/why-website-design-is-important/[accessed 11 Oct 2017]

'Adobe Photoshop File Formats Specification August 2016' [online], 1991-2016, available:https://www.adobe.com/devnet-apps/photoshop/fileformatashtml/[accessed 18 Dec 2017]

'PSD File Format' available: http://whatis.techtarget.com/fileformat/PSD-Adobe-Photoshop-default[accessed 18 Dec 2017]

Curs, G., Ferrer, R. (2016-2017) 'Improving Photoshop PSD file format support in the free software image editor GIMP' [online], available: https://repositori.upf.edu/bitstream/handle/10230/33036/Rocafort_2017.pdf?sequence=1&isAllowed=y [accessed 19 Dec 2017]

Josekutty, A., Prof, M., (2014) 'Intelligent HTML Code Analyser and Builder from PSD Layers'[online],available:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.652.67&rep=rep1&type=pdf [accessed 20 Dec 2017]

Nancy, M. (2017) 'How to View and Edit Photoshop PSD file Online' [online], available:

https://www.makeuseof.com/tag/view-edit-photoshop-psd-files-online/ [accessed 20 Dec 2017]

Nurzhan, N., Michael, P., Randall, R. ‘Comparison of JSON and XML Data Interchange Formats a Case Study’ [online], available: https://pdfs.semanticscholar.org/8432/1e662b24363e032d680901627aa1bfd6088f.pdf[accessed 20 Dec2017]

Adrian.R, ‘Basic Image Manipulations in Python and OpenCV: Resizing (scaling), Rotating, and Cropping’ [online], 20/01/2014, available: https://www.pyimagesearch.com/2014/01/20/basic-image-manipulations-in-python-and-opencv-resizing-scaling-rotating-and-cropping/ [accessed 26 Dec 2017]

‘Basic graphics operation and processing with JavaScript’ [online], 03/09/2014, available:

http://www.ituring.com.cn/article/121428 [accessed 28 Dec 2017]

Kim.W, Kwan.K, Fedyuk.I, Margrit. B, ‘Camera Canvas: Image Editor for People with Severe Disabilities’ [online], 2008, available: https://open.bu.edu/bitstream/handle/2144/1703/2008-010-camera-canvas.pdf?sequence=1 [accessed 30 Dec2017].

‘PSD viewer tool’ [online], available: https://github.com/zenoamaro/psd-viewer.git [accessed 30 Dec 2017]

‘PSD parser tool’ [online], available: https://github.com/meltingice/psd.js [accessed 30 Dec 2017]

Michael.M, ‘The best website builders of 2018’ [online], 2017, available: http://uk.pcmag.com/web-site-hosting-services/42131/guide/the-best-website-builders-of-2018[accessed Feb 2018]

Best Program for Web Design-no programming [online], 2013/07/05, available: https://forums.adobe.com/thread/1248635[accessed 5 Feb 2018]

Browserify Summary [online], available: https://www.cnblogs.com/lovesong/p/5861932.html[accessed 7 Feb 2018]

Introduction to Browserify [online], available: https://writingjavascript.org/posts/introduction-to-browserify[accessed 7 Feb 2018]

Node.JS Modules [online], available: http://www.tutorialsteacher.com/nodejs/nodejs-modules[accessed 7 Feb 2018]

'Node.JS Wikipedia' [online] available: https://en.wikipedia.org/wiki/Node.js[accessed 8 Feb 2018]

Louis.S, 'Node.JS is the New Black' [online], 2010/07/03, available: https://www.sitepoint.com/node-js-is-the-new-black/[accessed 8 Feb 2018]

JS Modular Programming-- CommonJS and AMD/CMD [online], available: https://www.cnblogs.com/chenguangliang/p/5856701.html [accessed 9 Feb 2018]

Travis.M, 'Getting stated with gulp' [online], 2017, available: https://books.google.ie/books?hl=en&lr=&id=3UEwDwAAQBAJ&oi=fnd&pg=PP1&dq=gulp+file&ots=9MXJuKUtRr&sig=4mpX_fzYF5rifOtMYnQuZsuMpRE&redir_esc=y#v=onepage&q=gulp%20file&f=false [accessed 3 Mar 2018]

'Why use gulp' [online], available: http://blog.csdn.net/xllily_11/article/details/51320002 [accessed 4 April 2018]

'What is webpack'[online], available: https://survivejs.com/webpack/what-is-webpack/ [accesses 5April 2018]

Cory.G, 'React to Introduction' [online], available: http://pepa.holla.cz/wp-content/uploads/2016/12/Introduction-to-React.pdf [accessed 10 April 2018]

'React is Essential' [online], available: http://kartolo.sby.datautama.net.id/PacktPub/9781783551620-REACTJS_ESSENTIALS.pdf [accesses 10 April 2018]

O'Reilly, 'HTML5 Canvas' [online], available: https://the-eye.eu/public/Books/IT%20Various/html5_canvas.pdf [accessed 10 April 2018]