Exam hand-in - Technologies for Web and Social Media

Simon Rostami Mosen, student nb.: 20176458

June 2020

Contents

1 Introduction & Concept

Visual Weather is a website aiming to communicate real-time weather information to the user, using animated visual depictions thereof. Based on user-input and using *Open Weather Map API*[1] weather information is received and based on the received data, animated 2D-visuals are generated. The website is built using HTML to describe the structure of the page, CSS and Bootstrap for appearance, jQuery for animated scroll effects, PHP for selecting a random city as the initial value and p5.js for receiving weather data and generating visuals. The 'landing page' of *Visual Weather* can be seen below on figure 1.

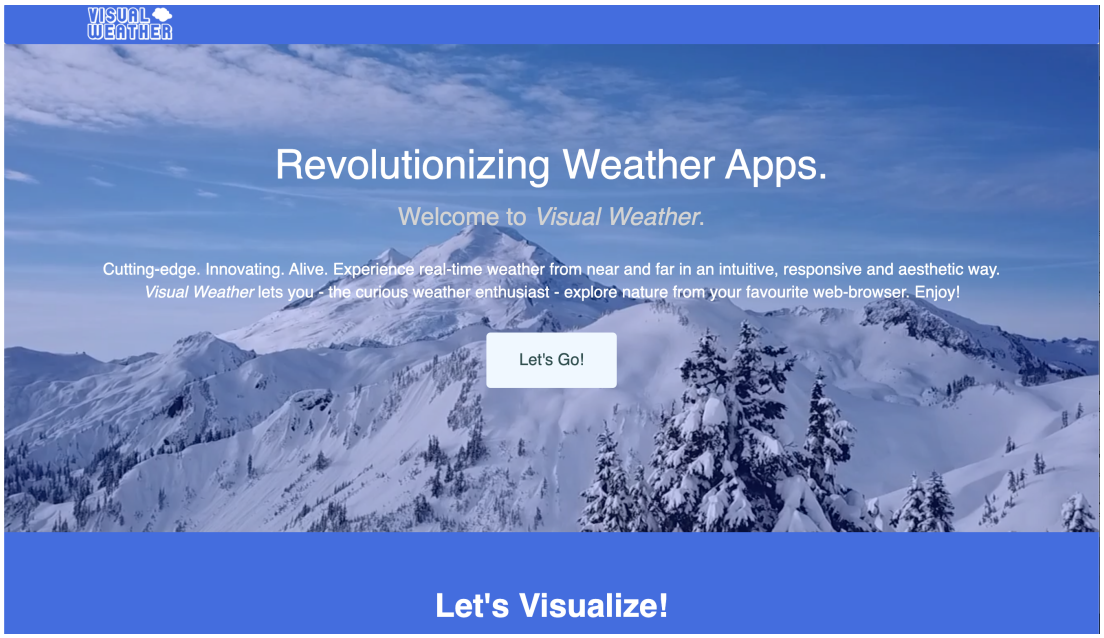

Figure 1: *Visual Weather*

[1] https://openweathermap.org/api

2 HTML & CSS

HTML and CSS is used to describe - respectively - the structure of the website and to define rules for appearance [5].

2.1 Description & Applications

HTML (Hyper Text Markup Language) is the standard markup language for website development [2]. Based on *Elements*, the web browser is told how to display different content [2]. This is often done to reflect a hierarchy of information [5].

CSS (Cascading Style Sheets) is used to specify how content of an element appears on the website [5]. It works by associating different rules with HTML elements [5].

2.2 Employment in 'Visual Weather'

HTML and CSS is widely used in the implementation of *Visual Weather*. HTML is used for structuring elements of the website while CSS is used to implement the visual appearance of these elements.

2.2.a Video header

An example of HTML, CSS and Bootstrap working together is the website's video header (as seen on figure 1). As HTML5 supports video playback using the HTML `<video>` element[2], making a video header is straight forward[3]. The implementation of the video header can be seen below (video reference[4]):

[2] https://www.w3schools.com/html/html5_video.asp
[3] https://startbootstrap.com/snippets/video-header/
[4] https://storage.googleapis.com/coverr-main/mp/Mt$_B$aker.mp4

Listing 1 Video header (index.php)

```
[...]
<header>
            <div class="overlay"></div>
            <video playsinline="playsinline" autoplay="autoplay"
                muted="muted" loop="loop">
                <source
                    src="https://storage.googleapis.com/coverr-main/mp4/Mt_Baker.mp4"
                    type="video/mp4">
            </video>
            <div class="container h-100">
                            <h2  id="colheaderText">Revolutionizing Weather
                                Apps.</h2>
                            <h2 id="colsubheaderText">Welcome to <i>Visual
                                Weather</i>. </h2>
[...]
```

As seen in the implementation above, the HTML video element allows for easy implementation of video. The element has a range of attributes such as `playinline` which ensures that the video will play right where it located, where the standard for mobile users would be the video playing in full screen. The other attributes `autoplay`, `muted` and `looping` will ensure that playback is automatically triggered, the video is muted and upon the end of the video, it will restart. The container is made using Bootstrap where the attribute `h-100` defines the height of the given container to be 100% of the height of the header.

2.2.b Button styling

Another example, where the effect of CSS styling is illustrated is on the buttons. The initial button without styling and the button with CSS styling applied can be seen below on figure 2 and 3.

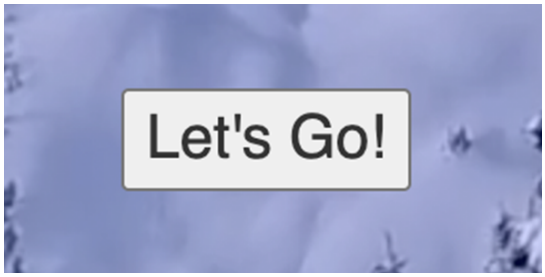

Figure 2: *Button with no CSS styling*

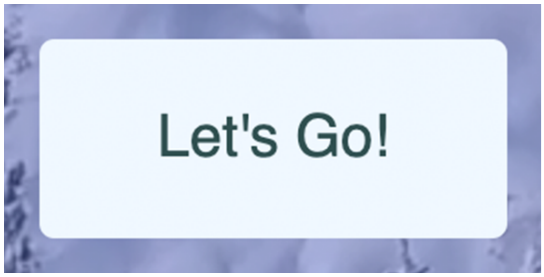

Figure 3: *Button with CSS styling*

The CSS implementation of the styling of the button, including sizing, rounded corners, background colour and text colour is found below:

Listing 2 Button styling (style.css)

```
#goToVisuals{
    margin-top: 25px;
    margin-bottom: 20px;
    border-radius: 5px;
    border:none;
    background-color: aliceblue;

    padding:20px 40px;
    color: darkslategray;
}
```

CSS is also used to create tailored style sheets for different platforms/screen sizes [1]. In order for *Visual Weather* to fit both computer screens and smaller resolution devices *CSS Media Queries* is used to hide HTML paragraphs when the screen resolution is below 620px. See the snippet below:

Listing 3 Media Queries (style.css)

```
@media screen and (max-width: 620px) {
  p {
display:none;
    }
}
```

2.3 Evaluation

HTML is a super effective tool for structuring a website, while CSS handles rules for styling in a very convenient way. The use of IDs to link elements with rules is very intuitive and allows for immediate execution of ideas. Furthermore, it is a practical tool, to use Media Queries for device-tailored style sheets.

3 jQuery

3.1 Description & Applications

jQuery is a simple and effective way of achieving a range of JavaScript tasks [6]. It does not exceed the possibilities of JavaScript but simplifies the way it is implemented as it wraps up extensive JavaScript code into simpler one-line methods [6]. jQuery

is cross-browser compatible and does not require any fallback code which makes it convenient for cross-platform implementations [6]. There are myriad applications of jQuery including the following:

- **Event methods:** jQuery is an effective way to respond to different events such as 'mouse moving over element', 'mouse click', etc [3].
- **Animations & movement:** jQuery can be used to add effects and animations to parts of a website. CSS properties represented by numbers are easily animated [6].
- **AJAX support:** jQuery makes it easy to create AJAX (Asynchronous JavaScript And XML) requests, a technique that allows for partial updates of a website without needing to refresh the entire page [6].

3.2 Employment in 'Visual Weather'

Since 'Visual Weather' is a single-page website, it was convenient to include smooth scroll animation when the user clicks the *Let's go!* button. Clicking the button will automatically scroll down to the input field. The smooth scroll animation is implemented using the following script[5]:

Listing 4 scroll.js

```
$("button").click(function() {
    $('html,body').animate({
        scrollTop: $(".visuals").offset().top},
        'slow');
});
```

The script above is triggered when the button is clicked. The `animate` method is used to animate/scroll the `body` element to the position of the `.visuals` class. The position of the `.visuals` class is found using the `scrollTop` method which returns the vertical position of the scrollbar and thereby defines the end location of the animation. The `animate` method, furthermore takes a second argument, being the speed of the animation - in this case, 'slow'.

3.3 Evaluation

jQuery is an efficient way of using JavaScript. It offers developers what their motto claims "write less, do more" as it wraps up extensive JavaScript code into simpler one-line methods. On the contrary, making such compact code can seem less readable to people not familiar with jQuery. As for *Visual Weather* it served the purpose of making smooth scroll animation without the need of extensive JavaScript coding.

[5] https://stackoverflow.com/questions/18071046/smooth-scroll-to-specific-div-on-click

4 PHP

4.1 Description & Applications

PHP is a very popular open source server scripting language [4]. As opposed to HTML, PHP scripts are not executed on the client-side but requires a server to be executed from (elaborated in section 4.2.a) [4]. PHP is often used to read and write files from a server as well as collecting form data and modifying database data [4]. Apart from PHP code, .php files can also contain HTML, CSS and JavaScript [4]. PHP is used by major tech companies such as Facebook[6] and Wordpress[7].

4.2 Employment in 'Visual Weather'

For *Visual Weather* PHP is used to set a random city as the initial value for the city input field. This is done by creating an array of strings (different cities) in PHP and then (also using PHP code) randomly indexing the array of cities and thereby randomly selecting a new city when the page is opened/refreshed. The implementation of PHP can be seen below:

Listing 5 Random city (index.php)

```
[...]
<?php

        $cities =array("Nuuk", "Oslo", "Marrakech", "Novosibirsk", "Athen",
            "Moscow", "Kobenhavn", "Tehran", "Cape Town", "Odense",
            "Hamburg", "Toronto", "New York", "Palma", "Gauna", "Tabriz",
            "Dori", "Reykjavik", "Beijing", "Xiamen", "Shanghai", "Hong
            Kong", "Shiraz", "Paris", "Berlin", "Aarhus");

        // get random index from array $arrX
        $randIndex =array_rand($cities);

        // output the value for the random index
        $rancity =$cities[$randIndex];
        ?>
[...]
                    <input id="textbox" type="text" value="<?php echo
                        htmlentities($rancity); ?>" />
[...]
```

As seen above, an array of strings (`$cities`), holding a range of selected cities, is created. The `array_rand()` function then returns a random key from the `$cities`

[6] https://royal.pingdom.com/the-software-behind-facebook/
[7] https://www.wpbeginner.com/glossary/php/

array and is then used to index the `$cities` array to find a random city. The random city is stored in `$rancity` and is then inserted in the 'value' attribute. The effect of the above implementation can be seen below on figure 4.

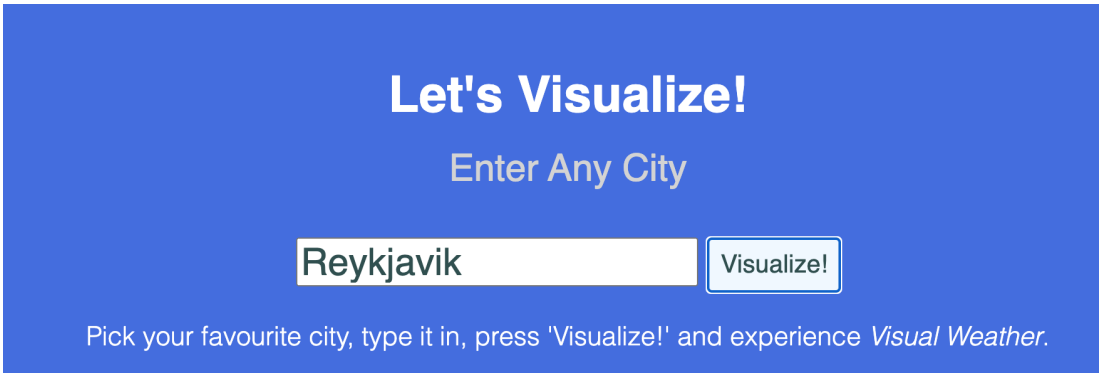

Figure 4: *Input field with random initial city*

4.2.a Server setup

As PHP can not be executed on the client-side (in the browser) a server has to be set up. This is not an issue once the website is hosted, as the host usually supports PHP, but when running/developing on a local computer, a server is needed. To run the server *XAMPP*[8] was downloaded and installed. *XAMPP* mounts a virtual drive where website files are to be executed from. Once this is setup, all files will run from the server. The *XAMPP* interface can be seen below on figure 5 and 6.

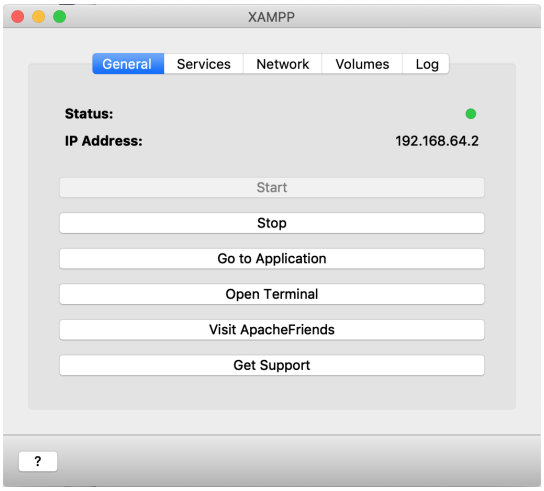

Figure 5: *XAMPP Server Interface*

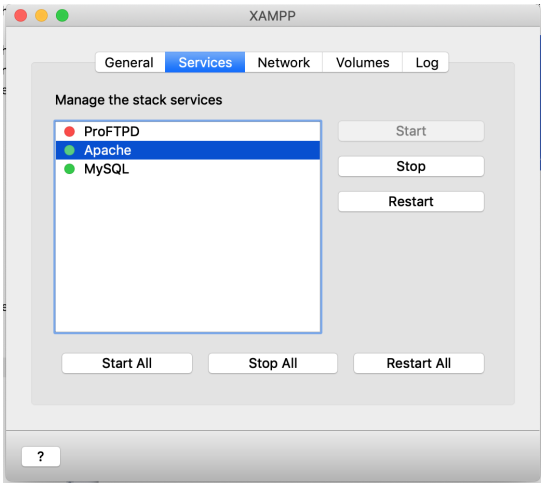

Figure 6: *XAMPP Server Interface*

[8] https://www.apachefriends.org/index.html

4.3 Evaluation

PHP is a very powerful tool for databases, reading and writing files etc. For this project, it can be argued that setting up a server and switching to PHP language is overkill considering the way it is applied to *Visual Weather* as similar results could have been achieved using JavaScript e.g. but as the nature of this assignment demands multiple web technologies, it fulfilled the task very well.

The PHP language itself is relatively straight forward and requires only limited programming experience to understand. The fact that it does not run on the client-side (such as HTML) makes the setup a bit more cumbersome, as it requires a running server.

5 p5.js

5.1 Description & Applications

p5.js is a 'dialect' of Javascript based on Processing [7]. It facilitates easy development of interactive graphics and thereby provides immediate visual feedback for the developer [7]. p5.js creates a *drawing canvas* in the window of the browser, where the visuals are shown [7]. To generate animated visuals, a typical p5.js sketch will consist of at least two functions:

Listing 6 p5.js setup

```
function setup() {

}

function draw() {

}
```

The `setup()` function runs just once when the program starts and is often used to define initial values for variables [7]. The `draw()` function should contain any code involved in animating/drawing on the canvas as it updates once per frame [7]. The nature of p5.js makes it very applicable to data visualisation. As described by McCarthy et al. [7]:

> "Data visualization is one of the most active areas at the intersection of code and graphics and is also one of the most popular uses of p5.js.
> [...] writing code to create visualization from scratch provides more control over the output and encourages users to imagine, explore, and create more unique representations of data."

As this project aims to visualise weather data abstractly and thereby invite users to abstractly interpret weather data, using a library like p5.js is very relevant.

5.2 Employment in 'Visual Weather'

In the implmenetation of *Visual Weather*, p5.js is used to facilitate both receiving weather data as well as generating visuals.

5.2.a Receiving weather data (JSON)

Weather data is received using *OpenWeatherMap API*[9]. Upon registration on the OpenWeatherMap website, a unique API key is generated. The API key is used to authorise requests from the application. An input field for city input and a "Let's Visualize!" button is implemented using HTML. When the button (or the enter key) is pressed, the `city()` function (seen below) is run.

Listing 7 city() function - sketch.js

```
function city() {
    cityVal =inputCity;
    var url="http://api.openweathermap.org/data/2.5/weather?q=" +cityVal +
        "&units=metric" +"&APPID=c10bb3bd22f90d636baa008b1529ee25";
    loadJSON(url, gotData);

}
```

The function above uses `loadJSON()` to retrieve weather data from the API and store the data as an array. The weather data is received in JSON (JavaScript Object Notation) format which is a format commonly used for storing data [7]. An example of the received JSON data from OpenWeatherMap can be seen below:

Listing 8 JSON weather data


```
{"coord":{"lon":-0.13,"lat":51.51},"weather":[{"id":300,"main"
:"Drizzle","description":"light intensity drizzle","icon"
:"09d"}],"base":"stations","main":{"temp":280.32,"pressure"
:1012,"humidity":81,"temp_min":279.15,"temp_max":281.15}
,"visibility":10000,"wind":{"speed":4.1,"deg":80},"clouds"
:{"all":90},"dt":1485789600,"sys":{"type":1,"id":5091,"message"
:0.0103,"country":"GB","sunrise":1485762037,"sunset"
:1485794875},"id":2643743,"name":"London","cod":200}
```


[9] https://openweathermap.org/api

As seen in JSON data above, a lot of information is received. In order to create visuals based on the weather information, a representative variable to simply depict weather should be found. In this case the "weather icon ID" described the weather sufficiently. To isolate the ID the data is treated as an array and indexed accordingly to retrieve the desired information. To isolate the icon ID, the following implementation was made:

Listing 9 Retriving icon ID - sketch.js

```
if(weather){
      currentWeather =weather.weather[0].icon;
  }
```

As seen above, the icon ID is stored in the `currentWeather` variable. The icon ID is a string in the format of "01d", "04n", "10d" etc. A table of the IDs and the matching weather descriptions is found below (based on ref[10]):

Icon ID day	Icon ID night	Weather description
01d	01n	Clear sky
02d	02n	Few clouds
03d	03n	Scattered clouds
04d	04n	Broken clouds
09d	09n	Shower rain
10d	10n	Rain
11d	11n	Thunderstorm
13d	13n	Snow
50d	50n	Mist

5.2.b Conditional statements

Based on the possible weather descriptions, a range of visualisations is implemented. A selection of the visualisation are seen below on figure 7.

[10] https://openweathermap.org/weather-conditions

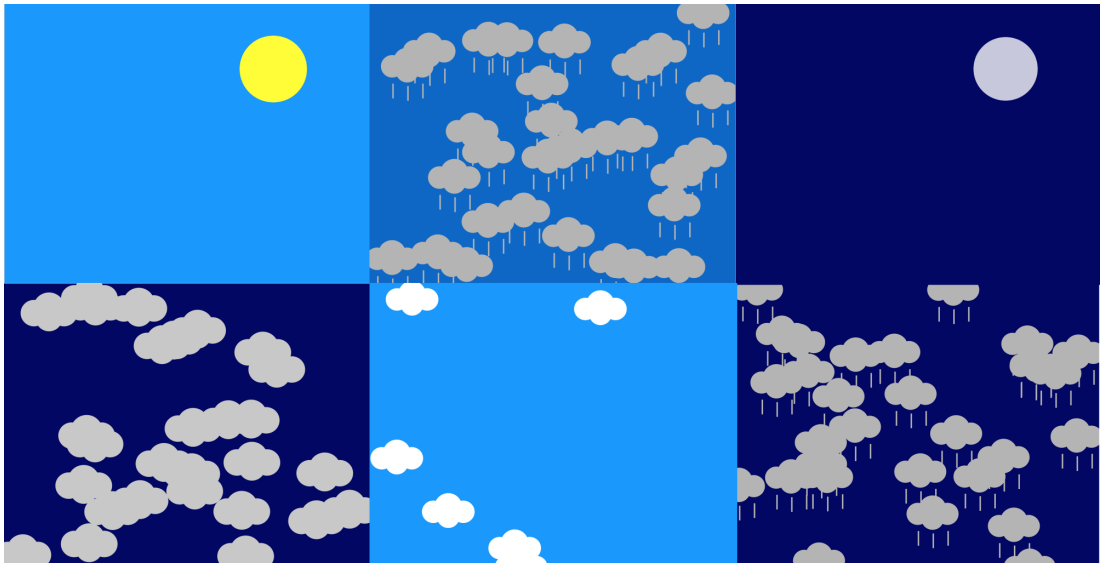

Figure 7: *Visualisations of different weather conditions*

To select the correct visualisation, a range of conditional statements are implemented. These conditional statements are based on the icon IDs in the table in section 5.2.a. The implementation is shown below:

Listing 10 Conditional statements - sketch.js

```
[...]
 if(currentWeather=="01d"){
        //day: clear sky
        background(0,150,255);
        //sun
        noStroke();
        fill(255,255,0); /
        ellipse(windowWidth/2+100,windowHeight/6,100+movement,100+movement);

    } else if (currentWeather=="02d"){
        //day: few clouds
        background(0,150,255);
        //clouds
        noStroke();
        fill(255);
        for(let i=0; i <6;i++){
            clouds[i].display();
        }
[...]
```

All visualisations are created using a combination of geometric shapes. Simple elements such as sun and moon, are simply created using a coloured ellipse, whereas the more complex clouds are created using a 'cloud' class.

5.2.c The 'cloud' class

In order to simplify the implementation of the cloud, a `cloud` class is created. By using object-oriented programming (OOP), extensive hard-coding of visuals is avoided. The implementation of the `cloud` class can be seen below:

Listing 11 Cloud class - sketch.js

```
class Cloud {
    constructor(offsetX, offsetY) {
        this.offsetX =offsetX;
        this.offsetY =offsetY;
    }

    display(){
        ellipse(this.offsetX,this.offsetY,30+noisy,30+noisy);
        ellipse(this.offsetX-30, this.offsetY+10, 20+noisy, 20+noisy);
        ellipse(this.offsetX+30, this.offsetY+10, 20+noisy, 20+noisy);
        ellipse(this.offsetX, this.offsetY+20, 20+noisy, 20+noisy);
    }

    rain(){
        rect(this.offsetX+30, this.offsetY+45,3, 30);
        rect(this.offsetX, this.offsetY+50,3, 30);
        rect(this.offsetX-30, this.offsetY+45,3, 30);
    }
[...]
```

The constructor of the `cloud` class takes two arguments for x and y position of the cloud. The class has three methods: `display()`, `rain()` and `snow()`. The `display()` method is used for the creation of the cloud itself and consists of four ellipses at a fixed size, and a relative positioning to each other. The `offsetX`/`offsetY` variables from the constructor are used to locate the cloud at a given position. The `rain()` and `snow()` functions add, respectively, rain drops (elongated rectangles) and snow (small ellipses) under the clouds.

5.2.d The 'movement' variable

To introduce movement to the visualisations, the `movement` variable seen below, was implemented.

Listing 12 Movement variable - sketch.js

```
[...]
movement=map(noise(xoff),0,1,10,50);
xoff+=0.007;
[...]
```

The `noise` function generates one-dimensional Perlin noise as a linear function of time (in this case the `xoff` variable) [8]. Perlin noise, in contrast to the `random` function, creates a more organic behaviour by producing a naturally ordered - and thereby - smooth sequence of pseudo-random numbers [8]. The `movement` variable is added to the size of the ellipses and thereby creates movement in the visualisations.

5.3 Evaluation

p5.js is a very versatile library for creating interactive experiences. It is very effective and by its visual nature it gives the developer immediate feedback. As for this project, p5.js served as the perfect platform for receiving data and generating animated data visualisations in an intuitive programming language, while still allowing for powerful interplay with HTML and CSS.

6 Conclusion

Based on this project it can be concluded that a website communicating real-time weather data can be successfully implemented using a combination of HTML, CSS, PHP, jQuery and p5.js. Furthermore, it can be concluded that each of these web-technologies has its specific advantages. HTML was found super effective at structuring a website, while CSS handles rules for styling elements in a very convenient way. jQuery was found to be an efficient way of using JavaScript as it wraps up extensive JavaScript code into simpler one-line methods. Even though PHP in this project was found to be a slightly excessive solution to a relatively simple task, it was found to be a very powerful tool for databases, reading and writing files etc. Lastly, p5.js was found to be a very creative and efficient way of developing a responsive and interactive website experience.

7 References

[1] CSS Media Queries - Examples. `https://www.w3schools.com/css/css3_mediaqueries_ex.asp`. Accessed: 2020-03-19.

[2] HTML Introduction. `https://www.w3schools.com/html/html_intro.asp`. Accessed: 2020-03-19.

[3] jQuery Event Methods. `https://www.w3schools.com/jquery/jquery_events.asp`. Accessed: 2020-03-19.

[4] PHP Introduction. `https://www.w3schools.com/php/php_intro.asp`. Accessed: 2020-03-19.

[5] J. Duckett. *HTML and CSS: Design and Build Websites*. Wiley, 2011.

[6] J. Duckett. *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley, 2014.

[7] L. McCarthy, C. Reas, and B. Fry. *Getting Started with P5.js: Making Interactive Graphics in JavaScript and Processing*. Make: Technology on Your Time. Maker Media, Incorporated, 2015.

[8] D. Shiffman. *Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction*. The Morgan Kaufmann Series in Computer Graphics. Elsevier Science, 2015.