Coursework 1 – Report

Alexander Koenig – 40322645@live.napier.ac.uk

06/03/2019

Introduction

The scope of this coursework comprises of creating a website using HTML, CSS and JavaScript, without any use of additional libraries, templates or frameworks. I have decided to implement a total of four cyphers: The Rot-N cypher, the A1Z26 cypher, the Keyboard Cypher and the QWERTY cypher. Detailed information about these have been sourced from "Hunt a Killer with the BAU". All of these will be featured on one structured page, each being in its own "corner", the main header of the website will be in the very centre, along with basic links such as "About" and "Contact Us".

In preparation for creating the website, I spent some time looking through the W3Schools website to see what specific tags might be useful for HTML, what properties may come in handy for the CSS and what functions I will need to make the JS work.

Software Design

HTML

The HTML portion will be straight forward: One element in the centre which handles the Title and the extra information, and four separate areas for the four different cyphers in use. To make the Rot-N and Keyboard Cyphers, dropdown menus are required. For entering text, text areas will be used.

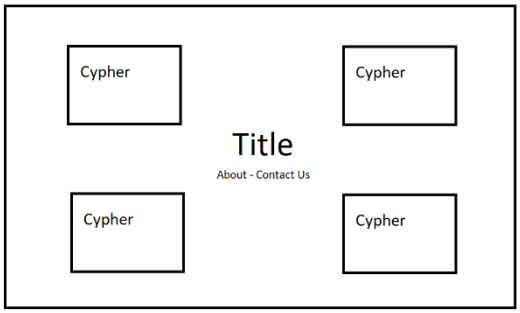

Figure 1: First basic layout idea

CSS

The main job of the stylesheet will be to ensure everything is in place correctly. Since it is an unconventional layout, extra care needs to be taken to complete this. CSS will also control the colour scheme – the idea for the layout as of right now is to have a dark grey-blueish theme. The background image should not take up more than 30% of the page, the idea is to insert it as a strip in the middle around the Title area, potentially with an opacity gradient connecting it to the top and bottom background colour parts.

JavaScript

From my estimates, the JavaScript file will be by far the largest. Essentially, there are five parts to it, which I will slightly delve into in the next few paragraphs

While I do appreciate that a Rot-13 solution was part of the lab, I rather want to see how I would do it by myself, and to add to the challenge, I have decided to implement a "Rot-N" cypher, where the user can choose which rotation it will be. The default will be set to 13. My idea is to convert the letter into a number, add the number selected by the dropdown menu and, if it is over 26 or under 1, subtract or add 26 respectively.

With the solution used in the first cypher, the A1Z26 cypher should be implemented fairly quickly, though I anticipate some issues with spaces and other special characters. One solution would be to give the output like this: "Abc def" => "1-2-3 4-5-6".

Implementing the keyboard cypher will be more difficult. The cypher itself is simple in the way I am planning on implementing it: It replaces the letter with the letter either left or right of it on a

keyboard. If there isn't a letter, the letter on the opposite side will be used. For this, I will need to create three separate maps, one for each of the three possible rows, then check where in the row the letter is and either add or subtract one from the index it is at. If it is one of the cases where the output key is on the other side, a special mechanism will have to be put in place.

Like the Keyboard Cypher, the QWERTY cypher (Q=A, W=B, E=C etc.) needs special implementation. A two-dimensional array seems to be one way to implement it, though through research I did find out that "indexOf" unfortunately does not cleanly work with a multidimensional array, which essentially will mean that I will require a for-loop within a for-loop.

All cyphers, but the A1Z26 one, also require casing to be considered, this should be simple enough using toUppercase/toLowercase, potentially requiring a way to store the casing of the letter in a Boolean at the beginning of a for loop.

Implementation

Implementing most features was fairly straight forward, though I ran into a few unfortunate problems. First and foremost, I was unable to put the "About" and "Contact Us" links where they were planned to go. The reason for this is the way I implemented the gradient opacity: To make it work, it needs to use the 'content: ""' CSS property, which makes it so nothing in the area can be interacted with. In general, I found it quite difficult to achieve such a simple effect – some research I found indicated that this potentially will only work on a limited number of browsers. To achieve full compatibility, it would require about six or seven different lines doing exactly the same. I opted to use the W3C-approved version of linear-gradient, which hasn't produced any issues using Firefox Nightly.

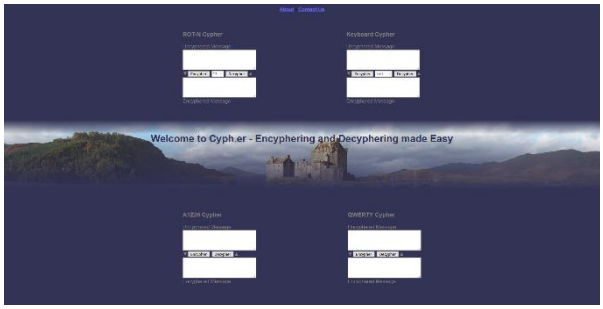

Figure 2: Finished product

Figure 3: Overlay

Beyond that, I was able to implement everything quite well. While at first, I did run into issues that made me put down everything for a few hours or even days, I was able to find everything out in the end to make it work. Furthermore, I decided to implement an overlay system for the About and Contact Us "pages" instead of having separate HTML files. Unfortunately, they don't have any fancy animation or special graphics.

All Cyphers are working as expected. I was able to follow the Software Design all the way until the end, and only had to make some minor adjustments.

The Rot-N encipher goes through all the characters of the input, which are taken from the textarea. If the character is a letter (this is checked by comparing to a regular expression using .test – this is used in all following cyphers), it saves the casing for later. Then, it converts the letter into a number based on the A1Z26 cypher. Next, it adds the number selected in the dropdown menu to that number. If the number is higher than 26, 26 will be deducted. Then, the number will be converted back into a letter, and also back into a lowercase character if that applies. All of this will only be performed on letters, any other characters will just be added to the output. For deciohering, the process is the same, the only difference being that the dropdown number will be deducted, and 26 added if it's lower than 1.

The A1Z26 encipher, also, goes through all characters from the input text area. If a character is detected, it will be converted into a number, and a "-" added. If any character other than a letter is detected, the last "-" in the output string will be removed, and a space will be added. This means that any non-letter characters will not be transferred during enciphering. After going through the whole input string, the last character in the output will be removed if it is a "-". For deciphering, the input is first split by spaces, then each "word" will be processed, with everything other than a number being a delimitator. With this, spacing will be correct in the output result.

The Keyboard encipher first goes through all characters. If the character is a letter, save its casing. Next, it will determine in which row of keys the letter is and save this row to a variable. If it's not a letter, it sets the row variable to null. Next, it will check if row isn't null. If it isn't, it will, depending on the selection from the dropdown menu, move either left or right in the row array. If it would be beyond the end or beginning of the array, special if statements handle this. For some reason, this requires an extra variable, as using the "row.indexOf(input.charAt(i).toUpperCase())" is not recognized in the if statement itself. This will then be added to the output, of course with casing in mind. If the character is not a letter, it will simply print the character to the end of the output. The deciphering process simply reverses this whole process.

The Qwerty encipher also observes casing throughout it. At its core, there is a "map", a two-dimensional array which has each letter mapped to its corresponding letter within the Qwerty cypher. First, the function iterates through all characters of the input. If it is a letter, it will check which is its corresponding letter in the map. As mentioned in the Software Design section, this is done by iterating through the map within iterating through the input. However, the loop breaks as soon as it finds the corresponding map. It saves the resulting array entry into a separate variable, and adds the corresponding letter to the output. If the character is not a letter, it will simply add it to the output without changing anything. The reverse process of this simply changes so that the second for-loop checks against the second letter of the arrays in the map instead of the first one.

Critical Evaluation

Overall, the website turned out the way I envisioned it. As expected, the JavaScript file makes up nearly three quarters of all the code – though this is partially on how I implemented the letter to number and reverse function. There could easily have been some improvement there, plus it is most likely not the best performing solution. However, I am happy with how well it is working right now, and both of those functions are used for two of the cyphers.

As for the layout, as mentioned in the implementation, I wasn't able to fully implement the header area, especially in regards to the placement of the "About" and "Contact Us" links. While researching on how to solve this, I found a few solutions, but using external libraries or overly complex code, which I didn't just want to copy and paste.

I do also see some issues with the text areas and buttons potentially being confusing for some users, though I hope that the arrows next to the buttons bring some sense into this for those users.

Finally, as mentioned, the overlay is – at least for my taste – a bit too simple, however, I was not quite sure how to improve anything about that.

Personal Evaluation

My previous experience with the Triade of HTML, CSS and JS, as well as my experience with Java has certainly helped completing the coursework. I do personally find that JavaScript is a lot easier to handle than Java or similar program languages, and it allows for more wiggle room when it comes to certain common programming issues.

Throughout the completion process of the coursework, I had to frequently use references like W3Schools, as well as to find solutions to problems on websites like Stackoverflow. I will list some of the most useful sites in a section at the end.

One of the largest challenges were a few parts of the keyboard cypher – first, it kept outputting any non-letter character as "undefined" even though I tried to specifically make sure that non-letter characters don't go through the same algorithm as letters. Also, the jumping from one end to the other on the right-side algorithm simply did not work. Both made me put down the whole thing for about a day, and after this break – with a fresh look on the problem – I figured it out right away, both were simple issues on my end.

Overall, for the most part at least, this coursework reminded me of how rewarding creating a website can be. We also never did JavaScript that in-depth back in school, so I've certainly learned quite a few things throughout the process. As for HTML and CSS, it was a great refresher on the extended basics, and I certainly feel equipped for the second coursework.

Significant References Used

- W3Schools.com (https://www.w3schools.com/) – Various pages, especially useful:
    - CSS How-TO – Overlay (https://www.w3schools.com/howto/howto_css_overlay.asp)
    - JavaScript Array (Both Tutorial (https://www.w3schools.com/js/js_arrays.asp) and Reference (https://www.w3schools.com/jsref/jsref_obj_array.asp))
    - JavaScript Regular Expression (https://www.w3schools.com/jsref/jsref_obj_regexp.asp)
    - HTML Colour Picker (https://www.w3schools.com/colors/colors_picker.asp)
- Stackoverflow (https://stackoverflow.com) – Various pages, especially useful:
    - CSS: Linear gradient transparency on both side of image (https://stackoverflow.com/questions/28409395/css-linear-gradient-transparency-on-both-side-of-image)
    - JS: Chop/slice/trim off last character in string (https://stackoverflow.com/questions/952924/javascript-chop-slice-trim-off-last-character-in-string)
- Hunt A Killer With The BAU (https://www.huntakillerwiththebau.com/cyphers2/) – For the Cyphers