Edinburgh Napier UNIVERSITY

Web Tech - Coursework 1

Manuel Alaminos Dominguez
40333504@napier.ac.uk
Edinburgh Napier University - Web Tech (SET08101)

1 Introduction

This is the first coursework of the Web Technologies module. We had to develop a website which would explain what classical ciphers are and how they were used. This involved research about specific classical ciphers and then develop online tools which would enable the user to encode their own secret messages. We were expected to **develop two ciphers** at the very least. As well as demonstrate a good knowledge of **design and usability**.

The coursework had constraints. We would **not make use of external libraries**, frameworks, or the like. Some examples of those are Favicon icons, CSS Preprocessors such as Sass, JavaScript libraries and frameworks like jQuery or React, and so on. Therefore, we had to develop our website with only HTML, CSS and JavaScript, the core technologies of the web.

This technology constraint was very positive for us, the students, as **it helps to make sure we have a firm understanding of the fundamentals** and foundation of web development before jumping into further, more abstract technologies, without a full understanding of how the core technologies actually work.

The first time I heard the word cipher was during the lecture where we were given this coursework.

Finding information about ciphers was not difficult. In fact, there is a lot of information out there on the Internet. Some of my background reading involved these websites:

learncryptography
Wikipedia

During my research I discovered small blog posts explaining how different ciphers work in various programming languages, after all, it is more about the algorithms than the language you are using to solve the problems you are facing.

I chose to do three different ciphers for my coursework, these were the following:

- **Caesar** cipher
- **Reverse** cipher
- **Atbash** cipher

The website, although not making use of external libraries, does use modern CSS and JavaScript to help with responsiveness, maintainability, scalability and readability. The end result is, a very clean UI, super lightweight website and with a good UX.

2 Software Design

From the beginning, I already had three important decisions made: what technology to use, the tools and development environment I was going to be working with and what foundation the UI design should be based on. The following sections provide an in-depth look at my decision process and my justification for their implementation.

2.1 Technology

My plan was to use the following technologies:

- **CSS Variables** (custom properties). Quite new in the realm of CSS, but after using a CSS Preprocessor like Sass, I felt this was necessary for an easy to maintain website. Specifically, it would help me to work and maintain colours and font-sizes.
- **CSS Grid**. I had decided I was not going to use a complex layout as I felt the website did not require it. However, CSS Grid would still be very useful for some of the two dimensional layouts within the website, this is, elements that have to be spread both vertical and horizontally. Although, this works perfectly well for one dimensional layouts too.
- **CSS Flexbox**. With CSS Grid and Flexbox to help with some of the one dimensional layouts, I would not have to deal with the old method of using 'hacky' floats and clear floats.
- **Semantic HTML5**. I would take advantage of semantic HTML5 tags and the usage of proper CSS naming classes. This would ensure a good level of code readability and accessibility for users.
- **CSS BEM**. The CSS naming classes would use this naming convention, very popular, very maintainable and scalable.
- **JS ES6**. This would be a good opportunity to practice some modern ES6.
- **Light/Dark theme**. For better accessibility or preferences of different users.
- **Media queries**. I knew I would not have to use many Media queries as Grids and Flexbox help towards responsiveness, but I would still use them for those elements that need extra modifications towards a better mobile experience.
- **Animation/Transitions**. Although I wanted to keep it very lightweight, I still wanted to use some animations and consistent transitions.

- **Optimised images**. I shall use optimised and compressed images to mitigate the impact of HTTP requests from the website.
- **Minimise the use of px** in CSS properties, and instead use more rem, percentages % and vh/vw. These units help towards a 'fluid' design.
- Due to these chosen technologies, I decided that **support for older or outdated browsers was not** going to be a requirement and I would rather push new technologies forward.

2.2 Tools and Development Environment

This section was, probably, the thing I had the least doubts about. My development environment would consist of:

- **Linux** operating system. In my case, the Elementary OS distribution with the Pantheon desktop environment, its default one. A clean, light and open source operative system.
- **Visual Studio Code**. Developed by Microsoft, this text editor is the open source flagship project from Microsoft. In my opinion, it is one of the best text editors out there. Fast, lightweight, but with lots of features, such as integrated version control, terminal, debugging, etc. It is available in Linux too, so there was not any cross-platform issue.
- **Node+NPM**. This open source JavaScript back-end language and package manager would allow me to easily install a live server for easy testability and overall provide a more efficient development environment. It would be easily achievable with a simple command: sudo npm install live-server -g
- **Firefox**. This open source browser, and specifically its new Quantum version, is very fast, and its dev tools are the best for CSS Grid inspections.
- **Bash**. I would make use of bash within the integrated terminal in VS Code, to run the live server, to work with Git -I prefer bash rather than the GUI- and for general purposes, such as creating folders and files for the project. I had used it already to install Node and NPM globally in my local computer, so I did not need bash for this task this time.

2.3 Design

I had in mind some general UI design foundation priorities I wanted the website to be based on:

- **Clean** and **simple** design.
- **Consistency** between every page.
- Follow **best practices**.
- Proper **white-space** for the elements to breathe while improving readability.
- **Sans-serif** font for readability on screens, especially for low dpi screens.
- Overall **flat** design

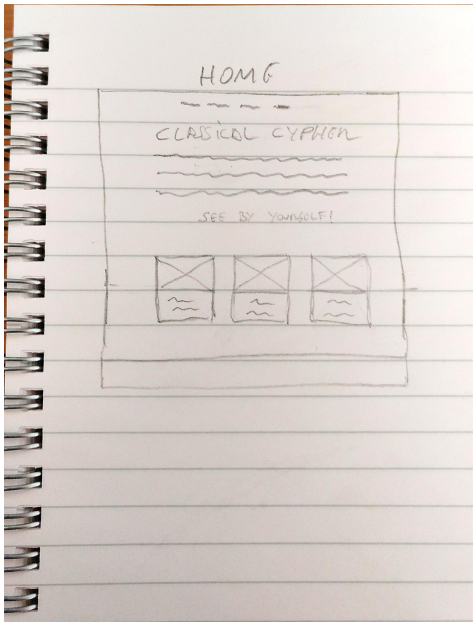

Figure 1: **Homepage** - sketch in notebook

- **Two main colours**, with a third one for call-to-actions and other elements that need user attention or action, but also to add some extra contrast in some headings. The main colours I had in mind were #FAFAFA for the background and #333 or #555 for text. I wanted these as they have less impact for the human eye than the pure white and black. The third colour was not chosen yet.

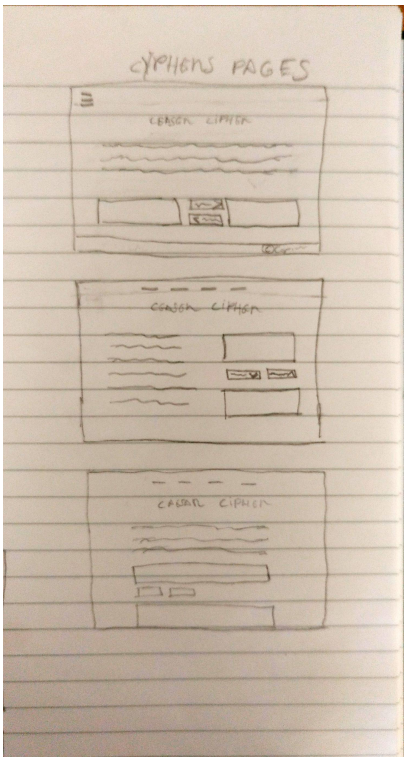

Figure 2: **Cipher page** - sketches in notebook

After all those ideas were set, it was time to **start wireframing**. I used the traditional method, **using pencil and paper**. I had to draw many sketches before starting development. Even after I started coding, I still changed some of

the design plans, e.g. the homepage currently does not have images.

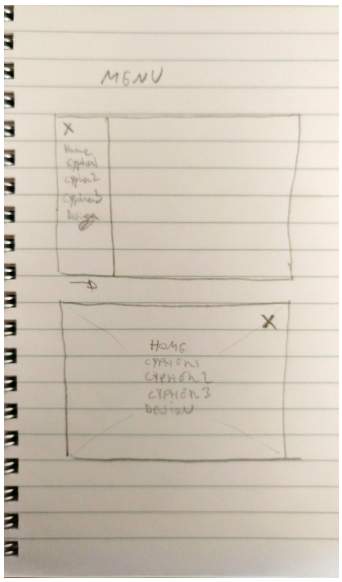

Figure 3: **Menus** - sketches in notebook

3 Implementation

The implementation went as expected. However, I did encounter some minor issues which I will explain in section four of this report.

The final **third colour** chosen in addition to the #FAFAFA and #333 was a golden #B28451.

I also used two different shades for every colour, or rather a lighter version:

- #FFF for a lighter version of #FAFAFA
- #555 for a lighter version of #333
- #C69963 for a lighter version of #B28451

The reason I chose this colour was because a classical cipher reminded me of ancient times, for example, the Roman Empire when parchment was used as a medium for written texts. More importantly, it provides a **good contrast** against the other two colours.

Another important note is the font used. Although I first used Helvetica, something in the visual aesthetic of the website was missing. I decided to make use of the **Quicksand** font, available on Google Fonts. It really improved the visual appeal of the website by a great margin. It was still sans-serif, so it would be still very readable for low dpi screens.

The implementation of the cipher was quite fun, and I very much **enjoyed overcoming some of the challenges** that came with it, even for the more basic reverse cipher.

I tried to use best practice and naming convention for JavaScript. Some of the very useful resources I found was Google JavaScript Guide

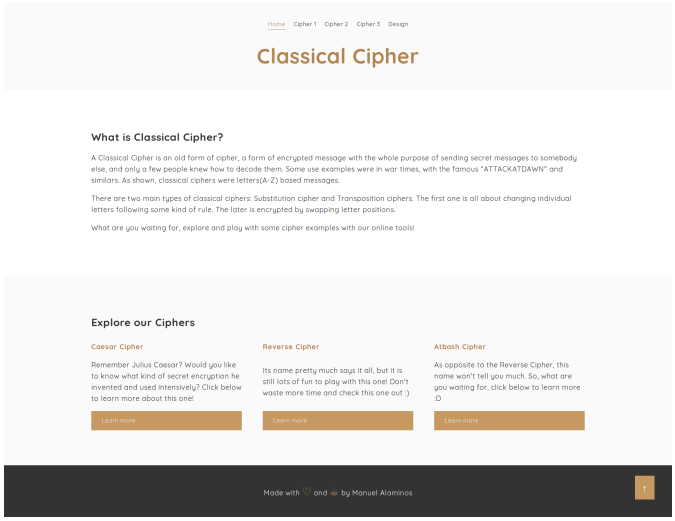

Figure 4: **Homepage** - in laptop screen at FullHD

Listing 1: JavaScript - ES6

```
1  // REVERSED CIPHER
2  const REVERSE_INPUT = document.getElementById("↩
     reverse-cipher-input");
3  const REVERSE_OUTPUT = document.getElementById("↩
     reverse-cipher-output");
4
5  function reverseCipher() {
6      let str = REVERSE_INPUT.value;
7      let encrypted = str.split("").reverse().join("");
8      REVERSE_OUTPUT.value = encrypted;
9  }
10
11 function clearReverse() {
12     REVERSE_INPUT.value = "";
13     REVERSE_OUTPUT.value = "";
14 }
15
```

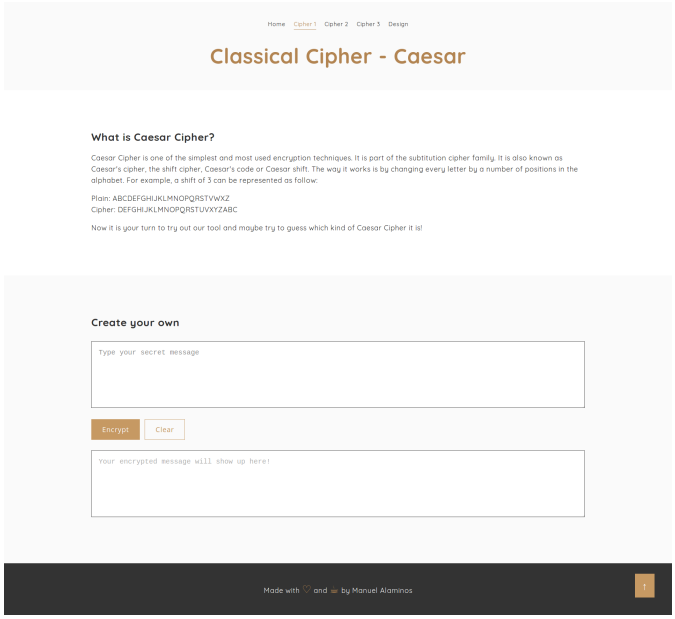

Figure 5: **Ciphers page** - in laptop screen

With the help of CSS Grid, Flexbox, Media Queries and the use of rem, percentages % and vh/vw, **I did not need to write many lines of code to make it mobile friendly.**

Grid was used in the features/explore section of the homepage. Grid is tremendously useful for two dimensional layouts. Although the 'cards' of the explore section are one dimensional, as soon as you shrink the width of the window, the 'cards' will start jumping to a new line, making it a two dimensional layout.

Listing 2: CSS Grid

```
1   .features {
2       display: grid;
3       grid-template-columns: repeat(auto-fit, minmax(25rem,↩
        1fr));
4       grid-column-gap: 6rem;
5   }
6
```

Flexbox also helped in the cards, but for the elements within the cards, aligning the title and button on top and bottom respectively. Keeping the elements within the cards aligned across all cards regardless of the length of the text in the middle.

Listing 3: CSS Flexbox

```
1   .feature {
2       display: flex;
3       flex-direction: column;
4       justify-content: space-between;
5   }
6
```

Flexbox was also used to optimise the navigation menu in mobile, in a column direction rather than in a row.

Listing 4: CSS Flexbox for mobile nav

```
1   @media all and (max-width: 450px) {
2       html {
3           font-size: 40\%;
4       }
5
6       .nav {
7           display: flex;
8           flex-direction: column;
9       }
10  }
11
```

The need to develop a CSS/Design documentation page was a very good point of this coursework. It was something completely new to me and the way I see it, should start doing this to every project I work on. This is not only useful to other developers but also to ourselves, when perhaps going back to work on a previous project. It also helps to see if I am being consistent with my design choices as I develop the website.

4 Critical Evaluation

Comparing the final result with my initial plan, I can see that there are a few things missing, but I also added others.

4.1 Missing

The first thing is, I have **not added any image** to the website. My obsession to keep it lightweight and the lack of very good reasons -at the moment of developing- to use them, pushed me to discard this idea.

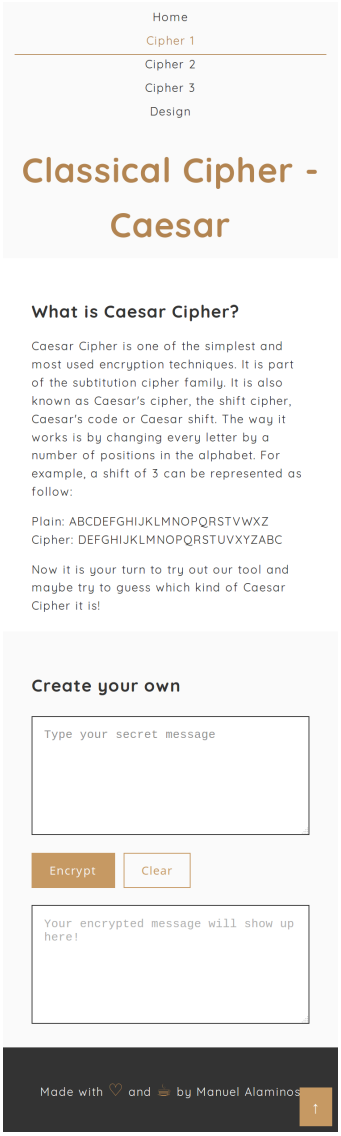

Figure 6: **Cipher page** - in mobile screen

Another important **feature missing** is the **light-dark theme** I wanted to develop. Like a simple button in which the colours would have changed to match a dark theme. Unfortunately, this have not been a priority in the development of the website.

4.2 Added

In regards to my development environment set up, I had to use Chrome to overcome **differences between browsers**. For example, Chrome style the auto-focus of form elements, while Firefox does not. I also used Windows and the Edge browser for testing purposes, but to my surprise, everything was working wonderfully.

There were a few other features I had to add that were not planned. One of them was simply the **story telling** capacity of the website, it was further down in the development stage when I stopped to think more about the user facing the website for first time. Thanks to that, I changed some texts for more meaningful ones.

Another feature that I added later down the road was better **SEO capabilities**. Once my website was pretty much done, I put it against some online tools that score your website

depending on its SEO, speed or security capabilities. Some of those tools were a new Chrome extension in beta state that you can find in Checkbot . And also Google Lighthouse, that you can run by simply opening the Chrome dev tools and clicking in Audit.

Then, I realised that I had to improve my meta tags to add more information like descriptions, author and tags to further improve the SEO of the site. And finally, I added myself as the Webmaster of the site with Google Search Console, to further analyse the performance of the website in the near future.

The latest feature I added was the **button to get back to top**. I found myself constantly missing that feature, so it was about time when I realise I finally had to add it.

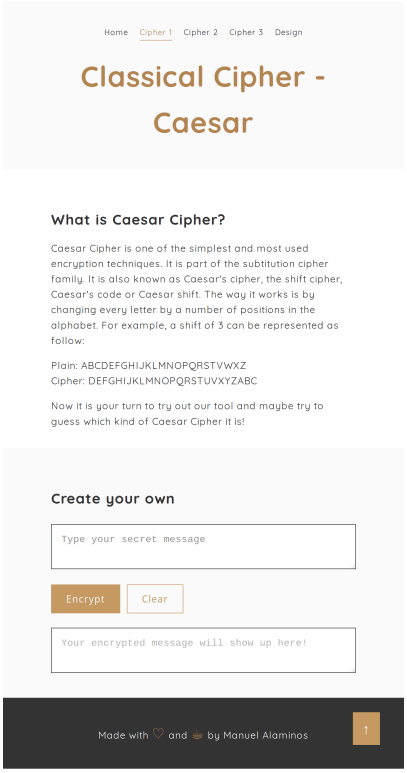

Figure 7: **Landscape** - in mobile screen

4.3 Improvements

Obviously, there is a lot of room for improvements on the site. The way I see it, the homepage needs something visually more appealing. I know I wanted the UI to be simple and clean, and get out of the way of the user. But the use of a **hero image** with a golden filter on top or something similar would have been a great addition.

I also think that more complicated and **advanced ciphers** could have been developed, but for me, as they are now, they were quite of a challenge already.

And last, but definitely not least, **a svg logo** to improve the branding of the page, making it more personal an unique as well.

5 Personal Evaluation

Personally, I think I improved existing skills more than learned new things. There were a few things I still learned though.

5.1 Learning new things

The first thing I learned about was **what a cipher is**, I did not know anything about this prior to the coursework.

Secondly, writing the **design documentation** was very insightful to me, it is very useful, as I mentioned above.

Third, I am also writing this report in **LaTex**, something that I had never used previously.

Development wise, I used **CSS Custom Properties** (variables) for first time and **CSS Grid** for first time in a personal project and not in an online course. Same could be said about **JavaScript ES6**.

5.2 Improving skills

It was a long time since I had developed a website. Since last trimester we did not have any web related module and I did not have any personal project during that time. I almost forgot some basic HTML and CSS features, so I really had to sharpen my web development tool kit.

I think it has been positive to have done some C# and Java(Android) within the last few months, as I had forgotten some JavaScript syntax. What also helped me improve my JavaScript knowledge was the online course JavaScript: The Weird Parts from Udemy, which further helped me understand what happens under the hood of JavaScript and, although very similar to other languages externally, it works very different internally. But I owned a lot of my JavaScript knowledge to the best online course I have ever done about programming, which is Practical JavaScript from WatchAndCode, which really focuses in the fundamentals of JavaScript.

This is not all about learning new things though, as we also have to go through small challenges and build stuff to really sink our knowledge. And this coursework was perfect for that, small challenges where I had to think really hard, at the same time as building a site, with a real context.

All in all, I think I performed very well. Not especially good at any one thing, but good all around, this is, UI design, UX, use of new technologies (CSS Grid, Flexbox and Variables, JS ES6), keeping mobile in mind as they cover now most of the internet browsing usage, and most of all, DRY, clean, readable, consistent, maintainable and easily scalable code, thanks to semantic HTML, CSS class naming convention with the BEM method, and JavaScript best practices.