## Scope and Content

The aim of this site is to allow the user to enter text and have it encoded/converted to their choice of code. The methods I chose are the Rot cipher and and the Morse encoder. Initially it was going to be the Rot13 cipher in specific, however I chose to allow the user to choose the value of the shift for extensibility.

## Design

Prior to coding this website I began to experiment with html and css to build my knowledge of design functions such as columns, styling, spacing, containers, colour palettes, etc. I combined all the knowledge I had picked up into the webpage, then duplicated the design elements into the coursework website, gradually stripping out or altering what I didn't need or have a use for, such as the login widget (which could be useful in future projects, but not the current one). I used online sources to generate colour palettes then extracted the colours I needed, tweaking them to make them fit better with the overall look of the page.

I had initially planned to have all content on one page,[1] however this would have rendered the navigation obsolete. It should be noted that the draft site[1] was a collection of different design elements purely for future individual reference.

In terms of code, I initially had 2 files; my index.html, page2.html and the .css. I had added my scripts at the bottom of the index.html file as I was struggling to call them as separate files. This was an issue as the code became very cluttered very quickly, later resolved by splitting up key parts of the site.

## Implementation

The plan was to do a Rot13 cipher and a Morse encoder. My first plan of action was trying to figure out how to allow the user to insert text, and have it be read as a value. Initially I had tried using inputboxes, which were small textbox fields, but I couldn't adjust the size of these and found they didn't offer the versatility I needed, so after digging around deeper I chose textareas. I wanted to go for a clean interface that was as minimalistic as possible, so I planned out drop down menus and textareas.[2]

### Drop down

The idea was to have the user select the cipher, and for the relevant options to become unhidden. For example, selecting the Rot cipher would display an additional drop-down which would allow the user to select the number of shifts they would like. This was turning into a tedious and fiddly task, so I decided to settle on a less challenging feat and come back to this design idea later if time permitted, putting both ciphers on one page and separating the textaras.[3]

### Images/modals

I decided to include some sort of proof to show the user that the converting/encoding works, by adding reference images of what the translations should be, on each side of the textarea [3]. I used modals to make the images appear in a pop up window once clicked, so the user could see their enlarged versions. I liked the organised look of this, as it filled the space nicely, but again there wasn't really an option to allow the user to choose ciphers, and wouldn't be an extensible option if more ciphers were added to the site, as they would end up stretching far down the page, so I decided to split the ciphers over several pages. I couldn't figure out how to get the modal to display different images, for example if image 2 was clicked, the modal would display image 1, so splitting the Rot and Morse over different pages was a solution to more than one problem. As the general layout of the site is centered, the navigation bar needed to be centered too [4], however, again, this particular design was an extensibility issue should more pages be added, as it took up a lot of space.

### Rot13

I started out the Rot13 code by hardcoding the number to shift by 13, so the user would not be given an option to choose how much to shift by. The function took in the values entered from the textarea, split each value up, converted it, then joined it all back together to reveal the encryption. This worked seamlessly, however moving on, there were a few major issues when trying to allow the user to choose their own cipher shift. After implementing the drop down menu to activate the encode function onclick, the values were coming up as undefined, or were coming up as the string "13" instead of the number. This was overcome by creating an event, and importing this into the function, so the value of the event (e) would be taken in (in this case, the number selected from the drop-down menu).[5]

Challenge solved, the encryption still wasn't functioning correctly. It would correctly encrypt the 1st letter, then encrypt the rest by the position of the value in the dropdown list. For example, the given select values were the numbers 11,12 and 13. If value 12 was selected, the 2nd letter entered by the user would be encrypted by 2, as 12 was the second option in the list. This boiled down to the charAt function. As shift was no longer predefined, for some reason

javascript started misreading the + and - . It would start to shift the ascii by 9 then by 7 instead of by 97. No sophisticated method of solving this was found other than using a window.alert after each character was found, to show which position in the ascii table the function was at, and adding + symbols in various parts of the line until it worked. [6]

### Morse

The cipher worked by taking in each character and using the .replace method, changing each character to global insensitive (gi) and lowercase. The same method was attempted with morse, however the function would then misread the sequences of dots and dashes. For example, if the input was ".- …", the output would be "e …", as it was reading each dot and dash separately and not as a set. This method was scrapped and instead I tried to use an array [7].

The idea was that the function would start taking in a set of values (in this case, a series of dots and dashes), until it found a space (as morse is separated by spaces). It would then treat the values it has taken in as a set, and compare this to what it has in the array, and if the value matches up to anything in the array. This approach was more successful, however still posed the problem of not treating the morse as sets. The function was left unfinished. [8]

## Critical Evaluation

### Comparison of requirements

#### Input and Output Text

The requirement of allowing the user to enter a message/text was met with the use of textareas linked to the ciphers, with read-only textareas used to output the processed message. This was met fully. A potential improvement could have been creating a flip button, which would transfer the processed text into the input box, to test the decode functionality without having to copy paste, however this was a design option and not mandatory.

#### Selecting between ciphers

The website allows the user to change between ciphers using a navigation menu at the top of the page, however a drop down menu is used to allow the user to choose the specific cipher shift they would like. A potential improvement could have been made by condensing the content into one page, however hiding all of the irrelevant elements. For example, a drop down menu could have been used to allow the user to select the method of encryption, and onclick revealing any relevant buttons or further input options required.

### Computing the Cipher

The Rot cipher can be computed by simply selecting the value to shift by from the drop-down menu, with no additional click of a button, complete with an additional reset button should the user wish to reset the content on the page without having to go into each box and backspace. There is no point asking the user to push a button after choosing the shift as no other input is required. Rot is encoded by taking in each letter via a charSet, and shifting it by the user specified number of spaces in the ascii table, returning the value it lands on. Initially I had encoded Rot using the .replace function, however realised this was not extensible if the user wanted to select different numbers to shift by.

The morse encoder/decoder comes with an encode and a decode button, as well as a reset as mentioned above. This could have been condensed with a drop down menu as mentioned in appendix 2, however the design looks more pleasing and complete with aligned buttons. A future improvement could be adding a sound button to play back the morse output as audio. The morse is encoded by splitting up each letter of the input individually, and replacing it with a specified value [9], outputting this value into the output textarea.

### Deciphering

The code behind the decode morse is incomplete [8], thus does not meet the requirement. Currently it decodes to the letter “e” and “t”, purely because of the dot and dash sequence.

Rot13 currently successfully deciphers the input text. However, it does not decipher a user specified number other than 13. This is, however, not required, as generally a user is not expected to know how many letters a cipher has been shifted by, and as the initial purpose of the cipher is to be a Rot13 cipher, the criteria is met. A future implementation could see different functions being made for each different number to shift by.

## Potential Improvement

Currently the website has no form of input validation. This could be a simple implementation in each function with if else statements.

Currently there is no sound output for morse, which would be preferred as traditionally morse is a means of audio communication. This could be implemented by taking a different approach to code, defining dots and dashes as a time in milliseconds, then translating these to sound.

## Personal Evaluation

Starting development on css and design early on in the course allowed me to quickly develop design ideas and have elements ready in advance for the coursework. This also supplied me with a personal library of information to be able to extract and use, personalising to the needs and requirements of the cipher site.

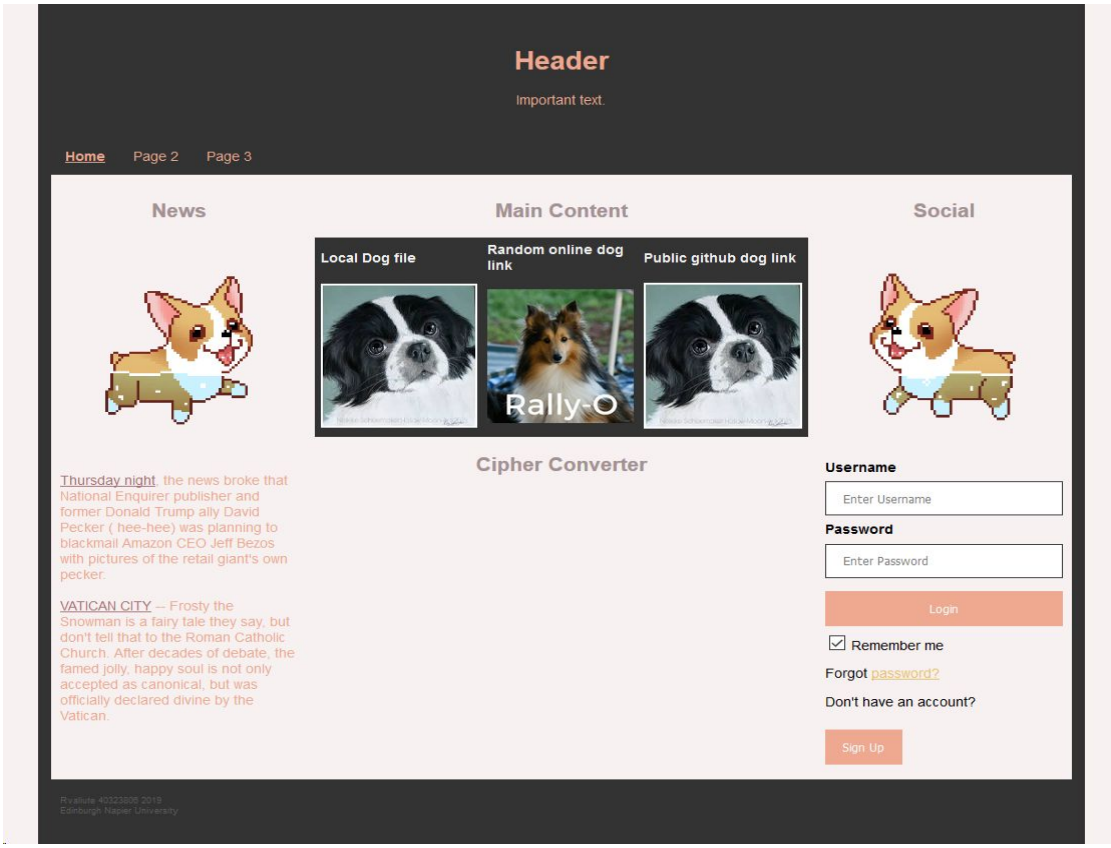
The actual coursework, however, was started very late. I feel like with more time, I could have dedicated myself more to the javascript and hopefully made much further progress with the decoding and optional extras, as I feel like I definitely have the capability to do so, just not in the time space I had given myself.

## References


| | |
|---|---|
| CSS Columns | https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_float |
| Drop Down Value Issues, general reading, no specific answer | https://stackoverflow.com/questions/43099742/change-javascript-variable-depending-on-selected-dropdownlist<br><br>https://stackoverflow.com/questions/12265596/html-select-change-selected-option-by-value-using-javascript |
| Modal Styling | https://www.w3schools.com/howto/howto_css_modals.asp |
| Approach to Morse Decoding, general reading to try come up with an approach | https://stackoverflow.com/questions/43726344/js-decoding-morse-code |
| Morse Image src | https://www.history101.com/telephone-morse-code-messages/ |
| Rot14 Image src | https://en.wikipedia.org/wiki/ROT13 |
| Dog Gif src | https://gfycat.com/stickers/search/8bit |

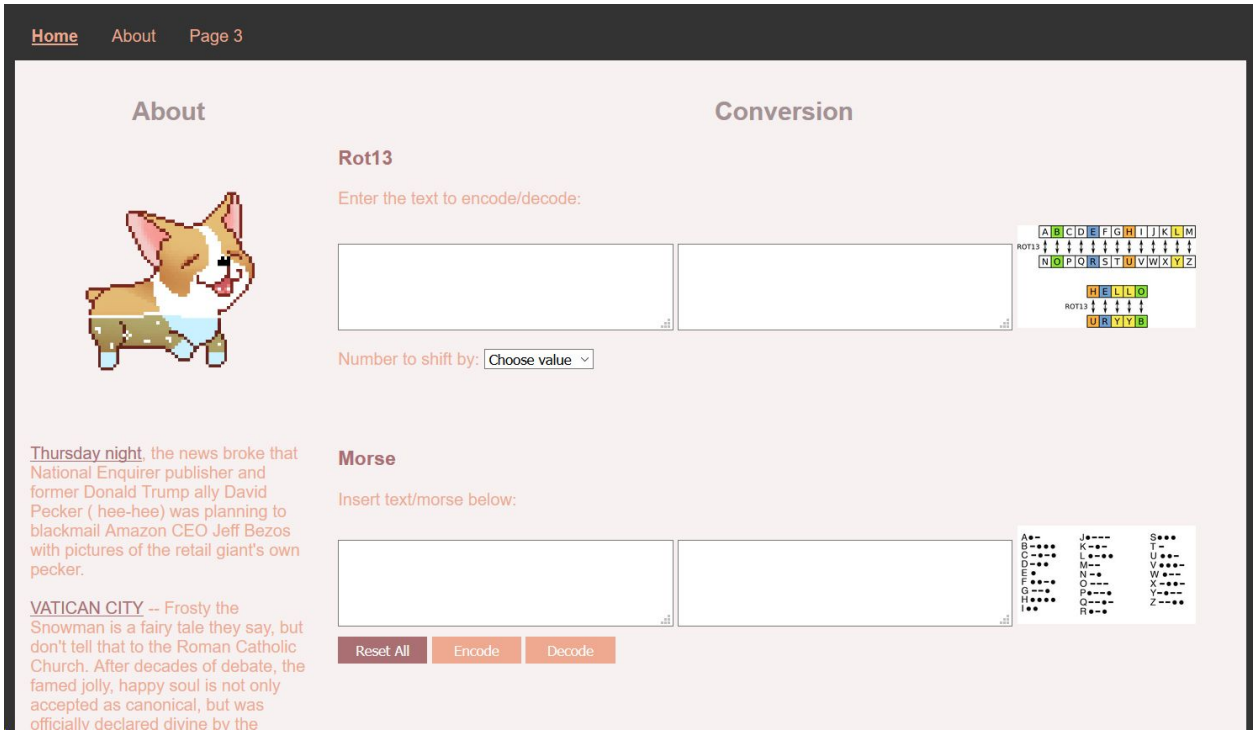
# Appendix

## a.1. Initial Plan, design.




## a.2. Initial Plan, drop down user select.

a.3. Separating text areas, images.




a.4. Initial nav menu, not extensible.


a.5. Rot13 encode development, changing from hardcoded value.

```
function Rot13Encode(e) {
        var str = textarea.value.toLowerCase().split(" ").sort().join(" ");
        var shift = document.getElementById("shiftSelect").value = e.target.value
        /* var shift = "13" */
        var output = "";
```

a.6. Rot13 shifter issues, java picking up incorrect values.

```
    for (var i = 0; i < str.length; i ++) {
        var character = str[i];
        var code = str.charCodeAt(i);

            character = String.fromCharCode(((+code - +97 + +shift) % 26) + +97);
            /*window.alert((+code - +97 + +shift) % 26)*/

        output += character;
    }

    textarea2.value = output;
}
```

a.7. Morse Decode array development.

```
function MorseDecode() {
    var output="";
    var convert = textarea3.value
    var morse = {
        ".- ": 'a',
        "-... ": "b",
        "-.-. ": "c",
        "-.. ": "d",
        "." : "e"

        };
```

a.8. Morse decode array development.

```
for (var i = 0; i < convert.length; i++) {
    if (convert === " ") { /*if text entered is a space */
            /* compare string and return value /*
            /* else */
            /* keep recording value entered */
        }

        output += morse[convert.split(" ").join(" ").charAt(i)];
    }

textarea4.value = output;
```

a.9. Morse Encoder .replace function

```
function MorseEncode() {

  textarea4.value = textarea3.value.split(" ").sort().join(" ")
  .replace(/a/gi, ".- ")
  .replace(/b/gi, "-... ")
  .replace(/c/gi, "-.-. ")
  .replace(/d/gi, "-.. ")
  .replace(/e/gi, ". ")
  .replace(/f/gi, "..-. ")
  .replace(/g/gi, "--. ")
  .replace(/h/gi, ".... ")
  .replace(/i/gi, ".. ")
  .replace(/j/gi, ".--- ")
  .replace(/k/gi, "-.- ")
  .replace(/l/gi, ".-.. ")
  .replace(/m/gi, "-- ")
  .replace(/n/gi, "-. ")
  .replace(/o/gi, "--- ")
  .replace(/p/gi, ".--. ")
  .replace(/q/gi, "--.- ")
  .replace(/r/gi, ".-. ")
  .replace(/s/gi, "... ")
  .replace(/t/gi, "- ")
  .replace(/u/gi, "..- ")
  .replace(/v/gi, "...- ")
  .replace(/w/gi, ".-- ")
  .replace(/x/gi, "-..- ")
  .replace(/y/gi, "-.-- ")
  .replace(/z/gi, "--.. ")
 }
```