Coursework Report

Luke Reeder
40277803@napier.ac.uk
Edinburgh Napier University - Web Tech (SET08101)

1 Introduction

The assignment given was to create a website to showcase the functions of a few different types of ciphers. This was to be done by implementing the style and functionality of the website through the use of HTML, CSS and Javascript with no additional libraries or framework. The site is to have at least two ciphers, an index page and a design page documenting the styles and presentation elements used throughout.

2 Software Design

The first thing I started on with my design was researching different types of ciphers. This was done to choose ciphers that seemed feasible to write in Javascript, and also find different ciphers with similar styles of encryption. By doing so, this allows a showcase of how simple a cipher can be, while also proving their effectiveness due to the many possibilities of encryption available. After research, I chose to use the Rot13 Cipher, the Caesar Cipher and the AtBash Cipher.

The Rot13 Cipher was chosen due to its simple nature along with its fairly straightforward implementation. Because the Rot13 is symmetrical in terms of its encoding as its decoding, only one function had to be written for the cipher. It also served as a comparison to show how ciphers can evolve and become more complex by leading then into the Caesar Cipher.

The Caesar Cipher was chosen as a follow on from the Rot13 due to it being essentially the same, but with a user key rather than a set shift. This was useful as it allowed similar design to be used but also with the addition of a separate input for the user to be able to test how different keys affected their message.

The third cipher I chose to use was the AtBash Cipher. Another substitution cipher that works by inverting each letter of the message, making it so A becomes Z, B becomes Y and so forth. Again, being a substitution cipher this would allow for a similar design to be used for both the javascript and also the html/css.

By choosing these three ciphers I could then start the initial design of the style of the web page and how each individual cipher page would look and feel. I thought of what each page would consist of, then drew a brief sketch as a very loose basis to work towards when implementing the design in CSS. I wanted to keep the main pages and theme of the site similar to keep it feeling consistent and smooth when switching between each page.

3 Implementation

When it came to actually building the site, I first started with the basic framework of the HTML. I made the basic pages with the list of index pages and anchors allowing navigation between each page. The next step I took was to make a quick text area and button to allow work and testing on the cipher functions. I started on just working on each of the encode functions at first, using the Rot13 as a basis.

To code the ciphers, I used the method of taking the input from the user, loop through each letter of the input, and then convert the character to its ASCII value. This then fed into a check to see if the value was an upper case or lower case letter. Once the value had been found, I used the formula (((indexCode - 97 + 13) mod 26) + 97) for lower case letters for the Rot13 Cipher. This worked by taking away the lowest value of the ASCII alphabet, in this example 97 from the code of the letter fed into the formula, then shifting it along 13 spaces. By then taking the modulus of this number, if the letter would be out of the range of the alphabet, it would then loop back round to the start. Finally, the 97 is re-added to then be converted back to a string character and appended to an output string.

While working on the Caesar cipher 1 I used the same formula of the Rot13 but edited it to also find the Shift input, and replaced the 13 with the user shift. This then allowed the cipher to work with a user input, but not encode the message if the user does not enter what shift they want to use.

The AtBash cipher 2 had a slightly different formula, but I used the same basis of using the ASCII characters to manipulate the letters and feed it back into a string at the end of the encoding. This time the formula; (indexCode + (122 - indexCode - (indexCode - 97))) was used. This formula works by taking the initial code, then taking the upper limit minus the code, then take away the code minus the lower limit. This value is then added back to the original index code to eventually have the inverse between the limits. This formula along with the Rot13 and Caesar have their respective limits replaced with the upper case alphabet limits when a capital letter is detected.

After the encode functions were finished and tested, the decode functions were then implemented. For the Rot13 and AtBash ciphers, their decode functions were the same as the encode functions due to the symmetrical nature of the encryption. The Caesar cipher however was an exception, mainly because of the user input for the cipher key. The same approach was once again taken to keep the javascript consistent and easier to maintain. The formula was altered to then take the inverse of the shift by adding; (26 - shift) to the it.

With the functionality finished for the site, I then started finishing the HTML sections, mainly being adding the description and headings to each of the ciphers. Once all of the HTML was finished and each section had appropriate ID tags, I moved onto the CSS. I decided to make the background a light color gradient, leading from a light blue to a pale white color. The main sections of the ciphers were then centered to highlight them, along with adding pictures to help describe the functionality. The index was the last part to be styled. I went for a outset border with a pale colored background to distinguish the different sections available to the user. This was then finished off by making sure that all of the text was consistent throughout by making it all Ariel font and appropriate sizes for each section.

4 Critical Evaluation

At first glance the website looks simple and light, although some parts do appear slightly plain in certain points, such as towards the bottom of the pages. The design of the page does cover each of the points listed in the specification, although as previously mentioned it is a bit simple and more could be done to add a bit more character to the page to help its visual appeal.

As for usability, I believe the website is very straightforward with clear indexing at the top of each page to navigate. Each page has a brief description of what each cipher does, and how it works. Also included is a picture to help give a visual aid to explanation of the ciphers workings. The text input box is clear and has a placeholder message to indicate where to enter a message to be encoded, and the output is also marked on the page. Possible improvements for this would be to make the output more distinct, such as having a dedicated box to display the encoded message in. Other possible improvements could be to have separate message boxes for encoding messages to decoding messages, for as it is currently there is only a single message box for both uses. This keeps the page simple but some users may prefer having separate message boxes and output spaces to see clearly how each function works side by side.

The functionality of each cipher is fully implemented; each cipher has both a encode and decode function on each page. A part that could be improved on is having some more complex ciphers to give more examples of how different ciphers work and showcase other types rather than only substitution ciphers. The description for each cipher is brief but gives an understanding of each, however the background of each could be explored and possibly set aside into a separate web page to help give the site some more depth to it overall.

5 Personal Evaluation

During the coursework I did come across a number of problems, many of which can and were solved by time and research into the language. I also spent a while trying to fully understand each error in the console as they came up to help future debugging. One such problem was an error in the encryption of the Caesar cipher messages, while the cipher worked perfectly with the letter A and any shift, any other letter was incorrectly being translated. One example of this was when the letter B was attempted to be encoded with a shift of 1. The expected output was C, however the actual output came out as N. After some testing and debugging from the console, it was found that the cause of the problem was the shift input was incorrectly parsing as a string which caused unexpected results. After a quick search a fix was implemented to take the input in from the user as a string, then parse the string as an int to pass into the equation.

Another fairly simple problem I had was again for the Caesar cipher was when it came to coding the function for decoding messages. For the encoding solution, I used the formula (((indexCode - 97 + shift) mod 26) + 97). When it came to decoding, I at first tried just reversing the shift to be a minus rather than a plus. This then caused problems when decoding past the letter A, as the ASCII code would then end up being less than the 97 which is the start of the lower case alphabet and into other character symbols. I found a fix to this to be adding in a separate bracket to do 26 - shift to give (((indexCode - 97 + (26 - shift)) mod 26) + 97). This then forced the equation itself to esentially loop back to the end of the alphabet rather than ending up as non letter characters.

Other problems were caused due to simple mistakes such as mistyping a variable during javascripts or referencing the wrong section of a sheet. I found the best way to recover from these mistakes and start making progress once more was to take a break if I did not find what I was looking for causing the error, and coming back to the problem with a fresh set of eyes. As stated, some of the time it was a simple error that I glossed over while looking for something else.

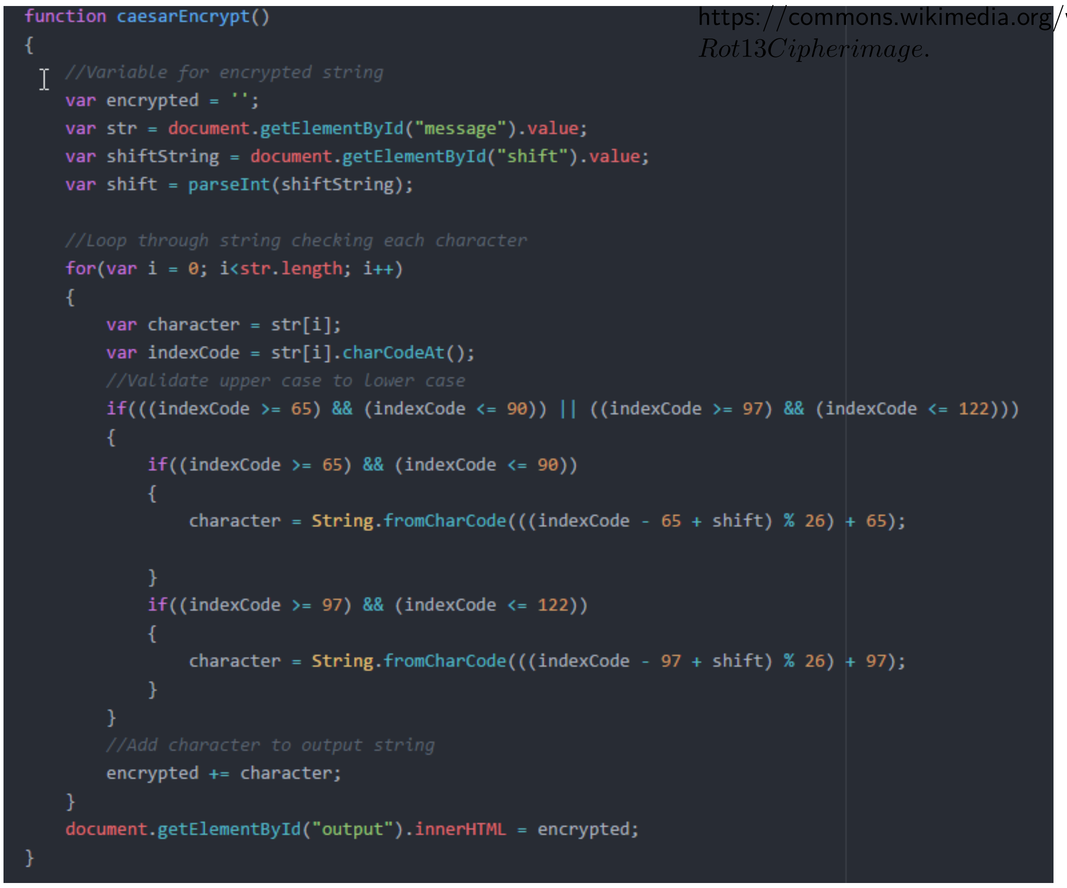
Thankfully the AtBash cipher was straightforward to implement especially once the other ciphers were fully functioning, as I used consistent styles by converting the strings to ASCII characters before encoding them, then converted it back to a string for the output.

If I were to repeat the coursework, I would like to spend more time on researching css techniques to allow my site to look more professional and less plain. I would also like to try more complex ciphers to help improve my knowledge of javascript and it's relationship with HTML.

References

https://www.pinterest.co.uk/pin/17592254775359300/ - Caesar Cipher image.

https://goo.gl/images/NCQwgf - AtBash Cipher image.



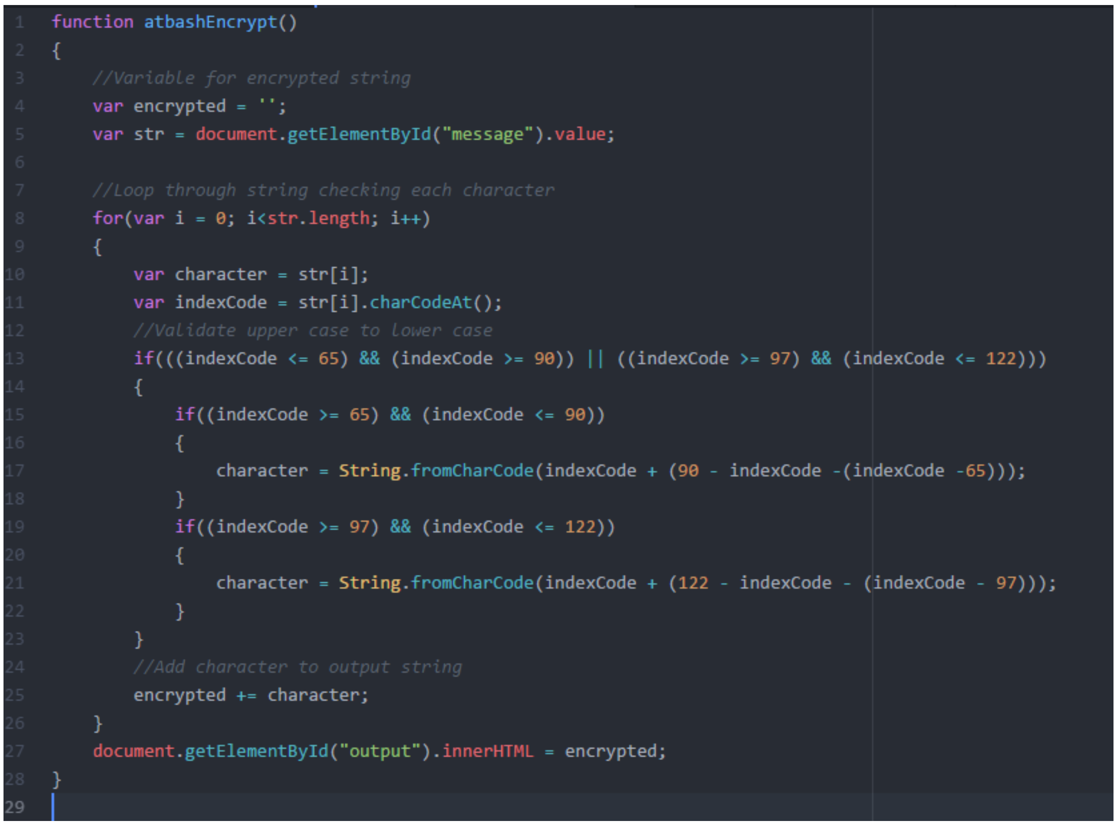
```
function caesarEncrypt()
{
    //Variable for encrypted string
    var encrypted = '';
    var str = document.getElementById("message").value;
    var shiftString = document.getElementById("shift").value;
    var shift = parseInt(shiftString);

    //Loop through string checking each character
    for(var i = 0; i<str.length; i++)
    {
        var character = str[i];
        var indexCode = str[i].charCodeAt();
        //Validate upper case to lower case
        if(((indexCode >= 65) && (indexCode <= 90)) || ((indexCode >= 97) && (indexCode <= 122)))
        {
            if((indexCode >= 65) && (indexCode <= 90))
            {
                character = String.fromCharCode(((indexCode - 65 + shift) % 26) + 65);

            }
            if((indexCode >= 97) && (indexCode <= 122))
            {
                character = String.fromCharCode(((indexCode - 97 + shift) % 26) + 97);
            }
        }
        //Add character to output string
        encrypted += character;
    }
    document.getElementById("output").innerHTML = encrypted;
}
```

https://commons.wikimedia.org/wiki/File:ROT13$_table.svg-Rot13Cipherimage.$

Figure 1: Screen shot showcasing the Caesar Cipher Java script.



```
function atbashEncrypt()
{
    //Variable for encrypted string
    var encrypted = '';
    var str = document.getElementById("message").value;

    //Loop through string checking each character
    for(var i = 0; i<str.length; i++)
    {
        var character = str[i];
        var indexCode = str[i].charCodeAt();
        //Validate upper case to lower case
        if(((indexCode <= 65) && (indexCode >= 90)) || ((indexCode >= 97) && (indexCode <= 122)))
        {
            if((indexCode >= 65) && (indexCode <= 90))
            {
                character = String.fromCharCode(indexCode + (90 - indexCode -(indexCode -65)));
            }
            if((indexCode >= 97) && (indexCode <= 122))
            {
                character = String.fromCharCode(indexCode + (122 - indexCode - (indexCode - 97)));
            }
        }
        //Add character to output string
        encrypted += character;
    }
    document.getElementById("output").innerHTML = encrypted;
}
```

Figure 2: Screen shot showcasing the AtBash Cipher Java script.