SET08101 Coursework Report

Kinga Kieczkowska (40205426)
Edinburgh Napier University

February 26, 2018

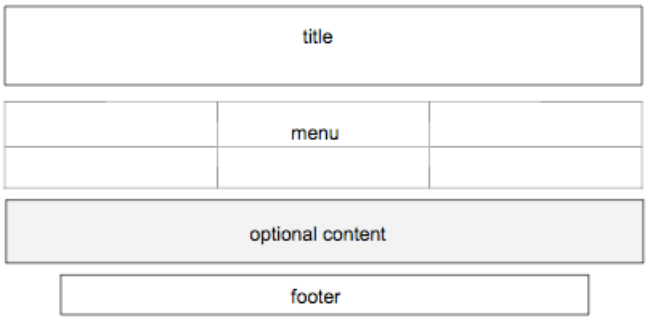
1 Introduction

The aim of the coursework was to produce a reponsive website featuring Javascript scripts encoding and decoding plain text in at least two different ciphers. The solution produced features six ciphers: ROT13, Ceasar, Polybius, Base64, Morse and Atbash. The website structure contains a main page (index.html) and six subsites corresponding to the ciphers listed above. A subsite design.html was also produced to showcase the design elements of the pages.

2 Software Design

2.1 Main page

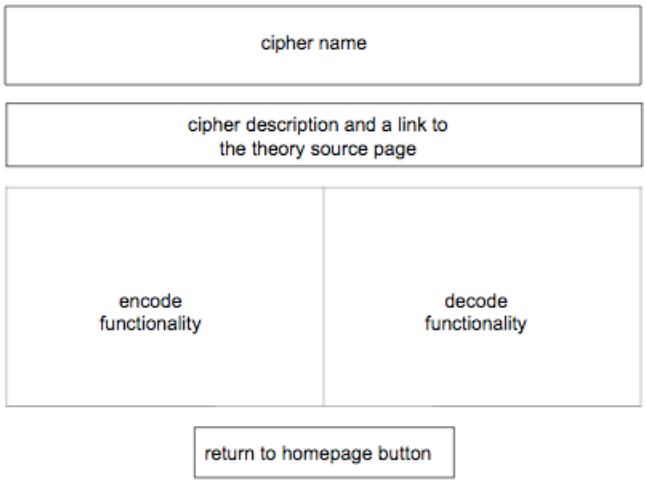
The website will consist of a main page, from where it will be possible to navigate to subpages dedicated to different ciphers, as portrayed on a sketch below. The menu is planned to be dynamic, different options changing background colour when hovered over.



Index.html will also feature a link to the design page as wekk as a footer informing of the purpose of the website.

2.2 Cipher pages

Each of the cipher pages is planned to feature a brief description of what it is and how it works as well as a hyperlink to a website containing more information on its operations. Underneath, in the form of a 1x2 table, the encode and decode functionalities will be offerred.

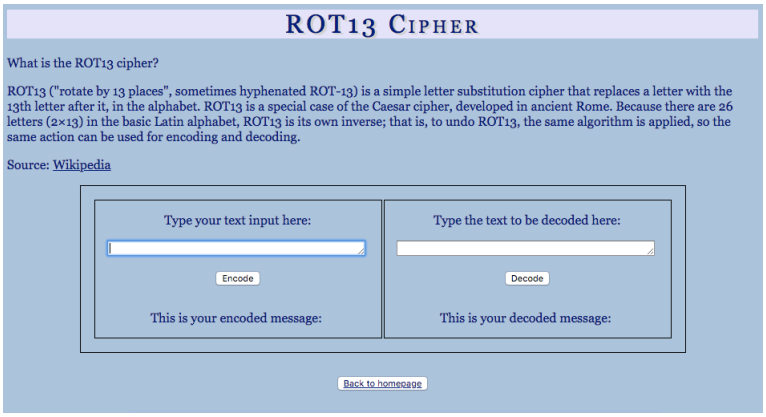


For the sake of usability, a return to homepage button will be placed at the bottom of the site to facilitate navigation back to the main page.

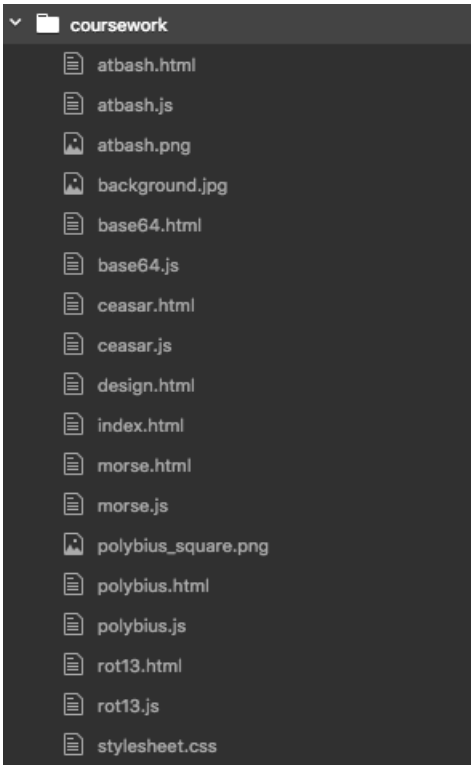
3 Implementation

The software design ideas have been implemented to produce the following main page:

and six subpages following the design suggested above - as an example, the ROT13 subpage:



The implementation of the website is based on an organisational model of having a dedicated, separate .html and .js file for each of the ciphers. The .html files feature the structural and some visual details of each subsite, while the .js files consist two functions: `xyz_encode()` and `xyz_decode()` (xyz being replaced with the name of each cipher). The application home directory is therefore as follows:



3.1 Algorithms' design details

Each of the cipher scripts were implemented with as much clarity and standardization as possible in mind. As portrayed by the `morse.js` script below, the variable names such as `plain_text`, `cipher_text` or `letter` are consistent between the two functions (as well as between different ciphers).

```
function morse_encode()
{
  var plain_text = document.getElementById("encode_message").value.toLowerCase();
  var cipher_text = [];
  var alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z',
  '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
  var morse = ['.–', '–...', '–.–.', '–..', '.', '..–.', '––.', '....', '..', '.–––', '–.–', '.–..', '––', '–.', '–––', '.––.', '––.–', '.–.', '...',
  '–', '..–', '...–', '.––', '–..–', '–.––', '––..', '–––––', '.––––', '..–––', '...––', '....–', '.....', '–....', '––...', '–––..', '––––.'];

  for (var idx=0; idx < plain_text.length; idx++)
  {
    var letter = plain_text[idx];
    var index = alphabet.indexOf(letter);
    cipher_text.push(morse[index]);
  }
  document.getElementById("encode_output").innerHTML = cipher_text.join(" ");

}
function morse_decode()
{
  var cipher_text = document.getElementById("decode_message").value;
  cipher_text = cipher_text.split(" ");
  var alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'];
  var morse = ['.–', '–...', '–.–.', '–..', '.', '..–.', '––.', '....', '..', '.–––', '–.–', '.–..', '––', '–.', '–––', '.––.', '––.–', '.–.', '...',
  '–', '..–', '...–', '.––', '–..–', '–.––', '––..', '–––––', '.––––', '..–––', '...––', '....–', '.....', '–....', '––...', '–––..', '––––.'];
  var plain_text = [];

  for (var idx=0; idx < cipher_text.length; idx++)
  {
    var letter = cipher_text[idx];
    var index = morse.indexOf(letter);
    plain_text.push(alphabet[index]);

  }
  document.getElementById("decode_output").innerHTML = plain_text.join("");

}
```

Listing 1: The morse.js file

The same approach was applied to naming the text areas in .html files to increase possible reusability of code between different ciphers.

3.2 Design

A CSS stylesheet was created and linked to all pages in order to standardize the their appearance. A number of devices such as `div ids`, `table` classes and `button ids` were used to ensure design integrity across subpages.

```
...
#back {
  margin-top: 25px;
}

#encode_button {
  margin-top: 15px;
  margin-bottom: 15px;
}

#decode_button {
  margin-top: 15px;
  margin-bottom: 15px;
}
...
```

Listing 2: A snippet of stylesheet.cs featuring the buttons' design

4 Critical Evaluation

The functionality of the proposed solution features six different ciphers, which is more than required and therefore offers a more rewarding user experience. All required functionalities have been implemented and a standardized style with a deliberately chosen colour palette applied to all presented subpages.

4.1 Future improvements

In the future with more time resources available the website could be improved by adding more interactive elements and dynamicity. Placing a more intuitive way to navigate directly between cipher subpages on each cipher subpage would surely add to the usability of the portal as well.

5 Personal Evaluation

This project has most importantly allowed me to improve my Javascript skills, as prior to this module my main programming language was Python. I also feel that I improved my code organisation skills significantly, as with so many different ciphers it would be impossible to keep organised if the code structure was not standardized across files. Lastly, I was also eager to learn design principles in practice to increase the "user-friendliness" of both the coursework website and programs I create outside of university classes and the lecture material has proven to be invaluable.