# INFO1111: Computing 1A Professionalism

## 2023 Semester 1

## Self-Learning Report

## Submission number: 3

**Github link:**
`https://github.sydney.edu.au/INFO1111-2023/Self-learning-for-Yizhen-Pan`

| | |
|---|---|
| **Student name** | Yizhen Pan |
| **Student ID** | 520009998 |
| **Topic** | JavaScript |
| **Levels already achieved** | A |
| **Levels in this report** | B,C |

# 1. Level A: Initial Understanding

## 1.1. Level A Demonstration

1. Install WebStorm IDE and create a JavaScript project
2. the project will include a JavaScript function and a web page to embed the function
3. one text box, which collect the message, will be provided in the web page for JavaScript function to do.
4. three trigger based (button will be working as it is clicked) buttons will be created, one button will store the information for function to use and other two buttons will use different functions as it clicked.

## 1.2. Learning Approach

First, I install the WebStorm IDE, a good tool for me to work and demonstrate JavaScript. Then I started to look over the contents about JavaScript by googling some online courses about JavaScript, the website I use to help archive the study is w3school, [1] gives me a clear instructions on how to study and use JavaScript from the start, furthermore, [1] also provides a simple JavaScript simulator which helps me to use the knowledge I learned directly.

After learning basic knowledge to set up a JavaScript function, I googled the content about the use of different <Input> in HTML and the way to combine it with JavaScript. The website I used for this part is mdn web docs, a website teaching the contents of HTML. [2] shows different attributes of input, greatly help me understanding the use of <input>.

After all the studies, I used WebStorm IDE to create a JavaScript Project, which contains a js file and an HTML file, and demonstrate the Level A demonstration steps I implement above to show my understanding of JavaScript.

## 1.3. Challenges and Difficulties

I think the challenge parts are receiving inputs, setting the variables and the way to use them into the function.

The ways for JavaScript to receive messages from users and output results are quite different from Python or Java that I learned. As [2] and [1] mentioned, we can use <input> with different attributes to get the correct type of message that we want in the HTML; like for the rotation part, I set the attribute to only receive integer numbers from 1 to 25. However, the message part is quite different since users may implement messages that combine Upper and Lower cases or even other signatures, increasing the difficulty to rotate them correctly. I solve the difficulty by converting them to ASCII code to rotate and then converting them back to text. This way greatly helped solve the problem a lot, but I also got into another problem, which was the position to rotate. This caused me to get the wrong rotation message. finally, I set the position to the mid- point of the alphabet letters to shift.

## 1.4. Learning Sources

| Learning Source | Contribution to Learning |
|---|---|
| `https://www.w3school.com.cn/js/index.asp` | this website introduced me the basic knowledge of JavaScript |
| `https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input` | it give me some ideas of how to use different ways of getting input from users |
| `https://en.wikipedia.org/wiki/Caesar_cipher` | it describes the function I want to demonstrate and the idea of how it works |

## 1.5. Application artifacts

The application I created is a web page that does Caesar cipher encryption.

The idea of Caesar cipher described by [3] is a type of substitution cipher in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet.

The way it works in my application is by entering the message, which need to encrypt, and then enter the number to rotate, and then click encrypt button to receive the processed encrypt message.

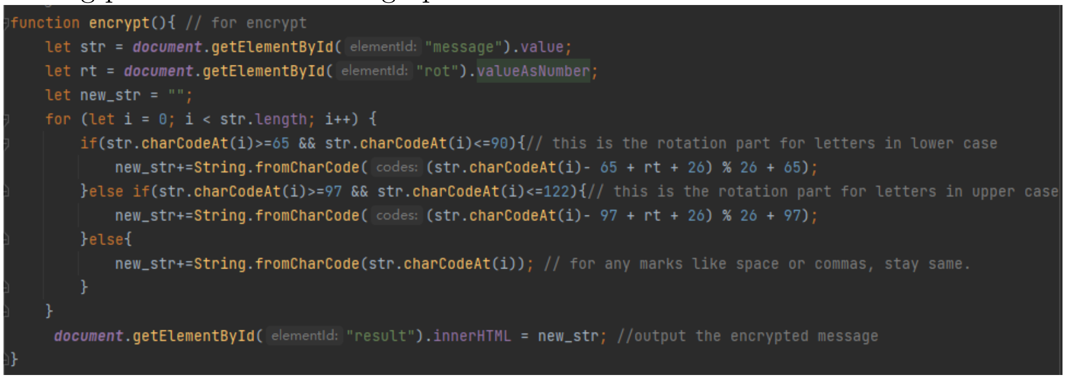
The function encrypt takes input from two HTML input elements with IDs *message* and *rot*, respectively. The input message is stored in the variable *str*, and the rotation amount is stored in the variable *rt*, then creates an empty string, $new_str$, which will store the encrypted message. The function loops through each character in the input *str* using a for loop.

For each character, the function checks whether it is a letter or not using the $charCodeAt()$ method. If the character is an uppercase letter (ASCII code between 65 and 90) or a lowercase letter (ASCII code between 97 and 122), it is rotated by the value of rt using a simple Caesar cipher, where each letter is shifted rt positions to the right. The modulo operator % is used to wrap around the alphabet so that letters beyond Z or z wrap back to A or a. All the rotated letters are added to $new_str$.

If the character is not a letter, the function simply adds the character to $new_str$ without rotation.

Finally, the encrypted message is output to an HTML element with ID *result* using the innerHTML method.

Coding part is showed in the graph below:



```
function encrypt(){ // for encrypt
    let str = document.getElementById( elementId: "message").value;
    let rt = document.getElementById( elementId: "rot").valueAsNumber;
    let new_str = "";
    for (let i = 0; i < str.length; i++) {
        if(str.charCodeAt(i)>=65 && str.charCodeAt(i)<=90){// this is the rotation part for letters in lower case
            new_str+=String.fromCharCode( codes: (str.charCodeAt(i)- 65 + rt + 26) % 26 + 65);
        }else if(str.charCodeAt(i)>=97 && str.charCodeAt(i)<=122){// this is the rotation part for letters in upper case
            new_str+=String.fromCharCode( codes: (str.charCodeAt(i)- 97 + rt + 26) % 26 + 97);
        }else{
            new_str+=String.fromCharCode(str.charCodeAt(i)); // for any marks like space or commas, stay same.
        }
    }
     document.getElementById( elementId: "result").innerHTML = new_str; //output the encrypted message
}
```

## 2. Level B: Basic Application

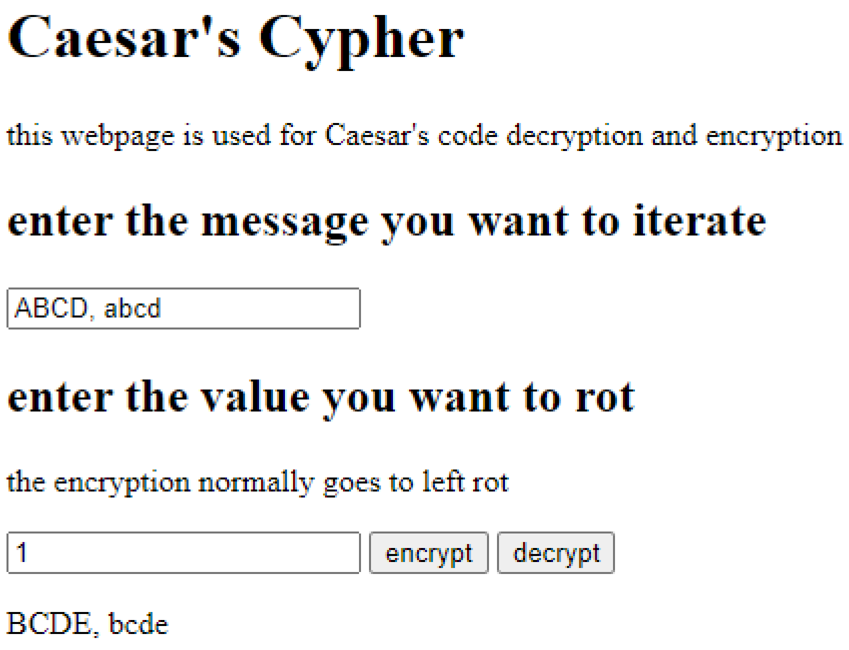
### 2.1. Level B Demonstration

The application I develop is a simple Caesar Cipher web page that can do both message encryption and decryption, the web page contains two event based triggers, encrypt and decrypt, that works as the button clicked by the user, two input text boxes–one for enter message, one for enter rotation value.

### 2.2. Application artifacts

In HTML part, JavaScript functions are activated when user click the button decrypt or encrypt, and the encrypt() and decrypt() functions in the JavaScript get the message and rot value entered by the user after the click using document.getElementById("message").value and document.getElementById("rot").valueAsNumber. The result will be showed after the line of button.

Example output



Caesar's Cypher

this webpage is used for Caesar's code decryption and encryption

enter the message you want to iterate

ABCD, abcd

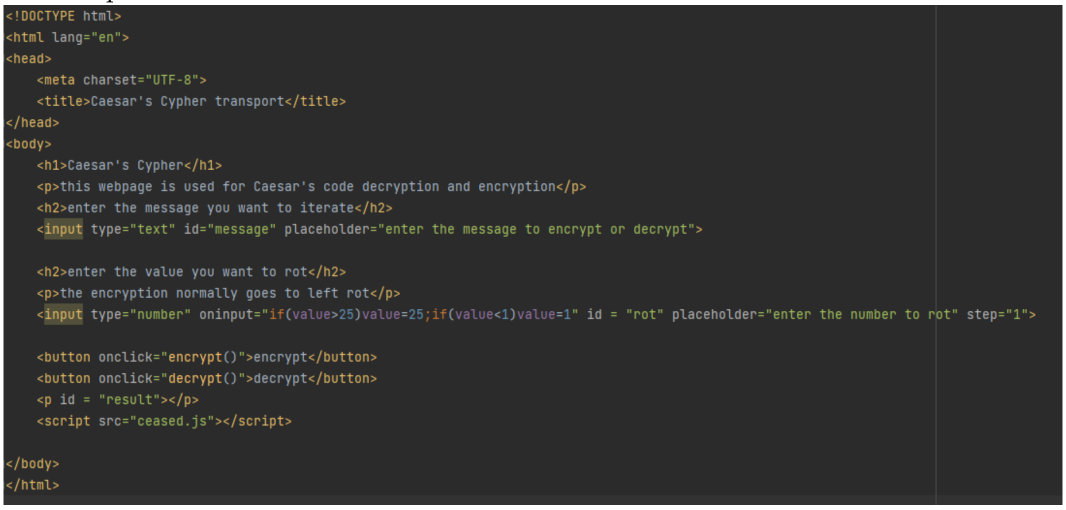
enter the value you want to rot

the encryption normally goes to left rot

1

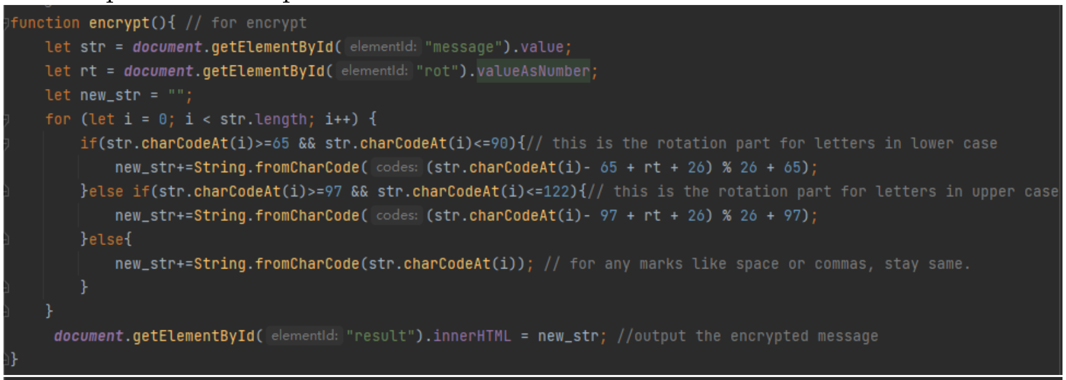
BCDE, bcde

the complete HTML codes are showed below

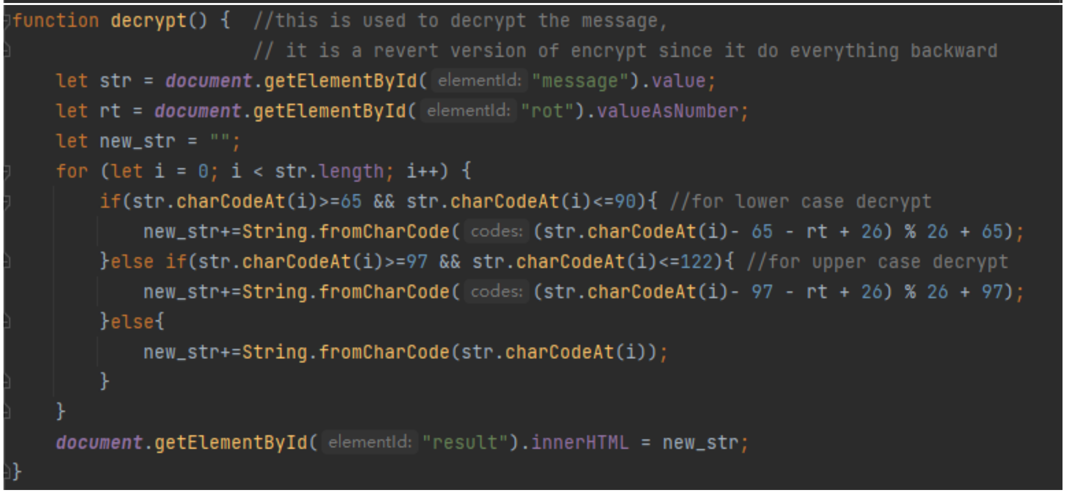


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Caesar's Cypher transport</title>
</head>
<body>
    <h1>Caesar's Cypher</h1>
    <p>this webpage is used for Caesar's code decryption and encryption</p>
    <h2>enter the message you want to iterate</h2>
    <input type="text" id="message" placeholder="enter the message to encrypt or decrypt">

    <h2>enter the value you want to rot</h2>
    <p>the encryption normally goes to left rot</p>
    <input type="number" oninput="if(value>25)value=25;if(value<1)value=1" id = "rot" placeholder="enter the number to rot" step="1">

    <button onclick="encrypt()">encrypt</button>
    <button onclick="decrypt()">decrypt</button>
    <p id = "result"></p>
    <script src="ceased.js"></script>

</body>
</html>
```

the complete JavaScript codes are showed below



```
function encrypt(){ // for encrypt
    let str = document.getElementById( elementId: "message").value;
    let rt = document.getElementById( elementId: "rot").valueAsNumber;
    let new_str = "";
    for (let i = 0; i < str.length; i++) {
        if(str.charCodeAt(i)>=65 && str.charCodeAt(i)<=90){// this is the rotation part for letters in lower case
            new_str+=String.fromCharCode( codes: (str.charCodeAt(i)- 65 + rt + 26) % 26 + 65);
        }else if(str.charCodeAt(i)>=97 && str.charCodeAt(i)<=122){// this is the rotation part for letters in upper case
            new_str+=String.fromCharCode( codes: (str.charCodeAt(i)- 97 + rt + 26) % 26 + 97);
        }else{
            new_str+=String.fromCharCode(str.charCodeAt(i)); // for any marks like space or commas, stay same.
        }
    }
     document.getElementById( elementId: "result").innerHTML = new_str; //output the encrypted message
}
```



```
function decrypt() {  //this is used to decrypt the message,
                      // it is a revert version of encrypt since it do everything backward
    let str = document.getElementById( elementId: "message").value;
    let rt = document.getElementById( elementId: "rot").valueAsNumber;
    let new_str = "";
    for (let i = 0; i < str.length; i++) {
        if(str.charCodeAt(i)>=65 && str.charCodeAt(i)<=90){ //for lower case decrypt
            new_str+=String.fromCharCode( codes: (str.charCodeAt(i)- 65 - rt + 26) % 26 + 65);
        }else if(str.charCodeAt(i)>=97 && str.charCodeAt(i)<=122){ //for upper case decrypt
            new_str+=String.fromCharCode( codes: (str.charCodeAt(i)- 97 - rt + 26) % 26 + 97);
        }else{
            new_str+=String.fromCharCode(str.charCodeAt(i));
        }
    }
    document.getElementById( elementId: "result").innerHTML = new_str;
}
```

# 3. Level C: Deeper Understanding

## 3.1. Strengths

According to [4] and my personal experience from developing the project, the key strengths are listed below:

1. JavaScript is very fast since it executes on the client side and this also reduces the server load since it can produce an error message before any information is transmitted to the server.

2. JavaScript is used everywhere on the web and can be execute in any modern browsers and produce equivalent results.

3. JavaScript plays nicely with other languages and utilized in an enormous sort of applications.

4. There are a lot of open-source projects and online courses about JavaScript, making it become one of the easiest IT languages to learn.

## 3.2. Weaknesses

The weaknesses of JavaScript is also very obvious too according to [4]

1. JavaScript codes are always visible to the client side, as we can simply use F12 in the browser to look at the source code. This can open up avenues of attack for malicious parties looking to gain access to related systems, do damage, or deny others access.

2. Compared to other programming languages, JavaScript is less strict on conventions and rules. For example, Java has String, int, float, char, and Boolean types to initialize the variables, but JavaScript may only use var to initialize all the types of variables needed to use, which may lead developers to bad habits if they aren't careful about applying rules of their own.

3. the websites powered by JavaScript can be displayed differently in different browsers. This makes it somewhat complex to read and write cross-browser code.

## 3.3. Usefulness

One such scenario I think JavaScript could be useful is in web development. According to [5]'s description, JavaScript is used when a webpage is to be made dynamic and add special effects on pages like rollover, roll out and many types of graphics since JavaScript will be executed immediately when user enter the web page.

## 3.4. Key Question 1

when to use it:

Based on [6] and [5] descriptions, one of the right time to use JavaScript is in web development, especially the user need to have their website communicate with the browser and watch for events like clicks and mouse enters since JavaScript is available for having the website interact with the browsers and users immediately.

not to use it:

JavaScript is not appropriate to use in creating Security-Sensitive Applications since one of the weaknesses of JavaScript is the codes are always visible to the client side, which will cause significant cybersecurity issues since we don't want users to access the sensitive information that might be included in the codes.

## 3.5. Key Question 2

According to [7] and [8]'s descriptions, JavaScript Frameworks are collections of different JavaScript code libraries that are compiled together to make it easy for a developer to refer to and use the scripts; It can greatly help developers to create websites easier. In the meantime, developers can choose the proper JavaScript frameworks to suit their needs, whether is the right size and performance, the one most compatible in the backend, or the right level of complexity.

# 4. Level D: Evolution of skills

## 4.1. Level D Demonstration

## 4.2. Application artifacts

## 4.3. Alternative tools/technologies

## 4.4. Comparative Analysis

# Bibliography

[1] w3school.com.cn, “JavaScript Tutorial.” [Online]. Available: https://www.w3schools.com/js/DEFAULT.asp

[2] “<input>: The Input (Form Input) element - HTML: HyperText Markup Language | MDN,” 4 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

[3] W. contributors, “Caesar cipher,” *Wikipedia*, 4 2023. [Online]. Available: https://en.wikipedia.org/wiki/Caesar_cipher

[4] GeeksforGeeks, “pede and Dispede of JavaScript,” *GeeksforGeeks*, 1 2023. [Online]. Available: https://www.geeksforgeeks.org/pede-and-dispede-of-javascript/

[5] P. Pedamkar, “Uses of JavaScript,” *EDUCBA*, 3 2023. [Online]. Available: https://www.educba.com/uses-of-javascript/

[6] C. Coyier, “You Know You Should Use JavaScript When... | CSS-Tricks,” 9 2009. [Online]. Available: https://css-tricks.com/you-know-you-should-use-javascript-when/

[7] C. Brewster, “JavaScript Frameworks: What Are They and How Do They Work? | Trio Developers.” [Online]. Available: https://www.trio.dev/blog/javascript-framework

[8] Codemotion, “17 JavaScript Frameworks that You Should Know About - A Comprehensive Guide,” *Codemotion Magazine*, 3 2023. [Online]. Available: https://www.codemotion.com/magazine/frontend/javascript/javascript-frameworks-guide/