M183 Applikationssicherheit Implementieren

Tutorial zum Lab Encryption Cesar Cipher

Version 1	30.11.2017	Jürg Nietlispach

Contents

Idee

In diesem Lab soll eine .Javascript Applikation erstell werden, welche Hilft, Geheimtexte, die mit der Cesar Cipher bzw. Monoalphabetischer Substitution erstellt wurden, zu decodieren.

Herangehensweise Cryptoanalysis

1. Index.html File erstellen
2. GUI-Elemente erstellen
3. Eventhandling und Routinen für Verschlüsselung und Entschlüsselung erstellen

Leeres index.html File erstellen mit folgendem Header-Informationen

```
<!doctype html>
<html class="no-js" lang="">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />

<body>
    <div class="container">
```

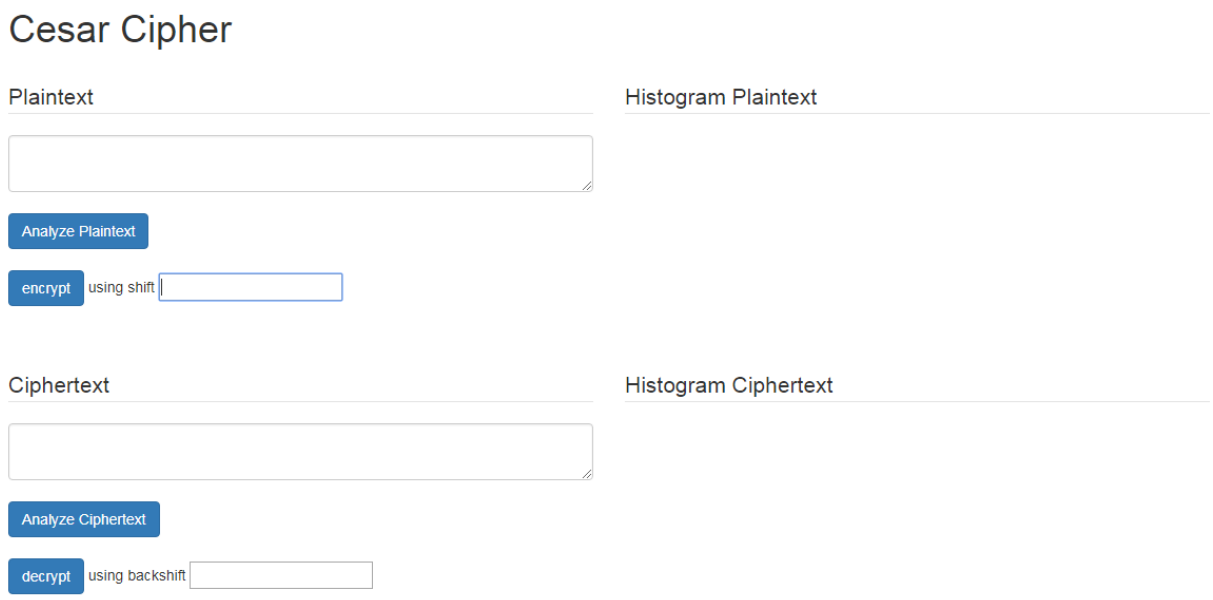
Innerhalb des Containers werden die Elemente dann hinzugefügt (Verschlüsselungsbereich)

```
<div class="row">
    <div class="col-md-12">
        <h1>Cesar Cipher</h1>
    </div>
</div>

<div class="row">

    <div class="col-md-6">
    <br>
        <legend>Plaintext</legend>
        <textarea id="plaintext" class="form-control"></textarea><br>
        <button class="btn btn-primary" id="analyze_plaintext">Analyze Plaintext</button> <br><br>
        <button class="btn btn-primary" id="encrypt">encrypt</button> using shift <input name="shift" id="shift" value="" type="text" />
    </div>

    <div class="col-md-6">
    <br>
        <legend>Histogram Plaintext</legend>
        <div id="chart_plaintext"></div>

    </div>

</div>
```

Und für den Decodierungsbereich

```
<div class="row">

    <div class="col-md-12"><br><br></div>
</div>
<div class="row">

    <div class="col-md-6">
    <br>
    <legend>Ciphertext</legend>
        <textarea id="ciphertext" class="form-control"></textarea><br>
        <button class="btn btn-primary" id="analyze_ciphertext">Analyze Ciphertext</button> <br><br>
        <button class="btn btn-primary" id="decrypt">decrypt</button> using backshift <input name="backshift" id="backshift" value="" type="text" />
    </div>

    <div class="col-md-6">
    <br>
        <legend>Histogram Ciphertext</legend>
        <div id="chart_ciphertext"></div>

    </div>
</div>
```

Wurden alle GUI Elemente korrekt hinzugefügt, sieht das dann so aus im Browser:



Nun kommt die Applikationslogik. Es soll ein Event für die Analyse des Plaintextes registriert werden (Button-Click):

```
document.getElementById("analyze_plaintext").addEventListener("click", function (e) {

    e.preventDefault();
    e.stopPropagation();

    var text = document.getElementById("plaintext").value;
    var histogram = createHistogramValues(text);

    document.getElementById("chart_plaintext").innerHTML = createHTMLTable(histogram);

});
```

Ebenfalls für die Verschlüsselung mit dem angegebenen Shift Faktor

```
document.getElementById("encrypt").addEventListener("click", function (e) {

    e.preventDefault();
    e.stopPropagation();

    var plaintext = document.getElementById("plaintext").value;
    var shift = document.getElementById("shift").value;

    document.getElementById("ciphertext").innerHTML = createAlphabeticalShift(plaintext, shift);

});
```

Für die Generierung des Histograms wird folgende Funktion benötigt – die untere generiert dasselbe Histogram in Tabellenform

```
function createHistogramValues(text)
{
    var histogram_prepare = [];

    for (var i = 0, len = text.length; i < len; i++) {

        var letter = text[i];
        if (letter.match(/[a-z]/i))
        {
            histogram_prepare[letter] = (histogram_prepare[letter] || 0) + 1;
        }
    }

    histogram = histogram_prepare.sort(function(a, b) {
        a = a[1];
        b = b[1];

        return a < b ? -1 : (a > b ? 1 : 0);
    });

    return histogram;
}

function createHTMLTable(histogram)
{
    var html = "<table>";
    for (var key in histogram) {
        html += "<tr>";
        html += "<td>" + key + ": </td><td> </td>";
        html +=  "<td>" + histogram[key] + "</td>";
        html += "</tr>";
    }
    html += "</table>";
    return html;
}
```

Für ExpertInnen: bitte Bar-Chart Library einbinden!

Die Verschlüsselungs (und Entschlüsselungs) Routine sieht dann folgendermassen aus:

```
function createAlphabeticalShift(text, shift)
{
    shift = parseInt(shift);
    if(shift < 0) shift = 26 + shift;

    // Make an output variable
    var output = '';

    // Go through each character
    for (var i = 0; i < text.length; i ++) {

        // Get the character we'll be appending
        var c = text[i];
        var tempchar = "";

        // If it's a letter...
        if (c.match(/[a-z]/i)) {

            // Get its code
            var code = text.charCodeAt(i);

            // Uppercase letters
            if ((code >= 65) && (code <= 90))
                tempchar = String.fromCharCode(((code - 65 + shift) % 26) + 65);

            // Lowercase letters
            else if ((code >= 97) && (code <= 122))
                tempchar = String.fromCharCode(((code - 97 + shift) % 26) + 97);

        }
        // Append
        output += tempchar;
    }
    // All done!
    return output;
}
```

Für den Decryption-Teil werden dann noch folgende Events benötigt:

```
document.getElementById("analyze_ciphertext").addEventListener("click", function (e) {

    e.preventDefault();
    e.stopPropagation();

    var text = document.getElementById("ciphertext").value;
    var histogram = createHistogramValues(text);

    document.getElementById("chart_ciphertext").innerHTML = createHTMLTable(histogram);

});

document.getElementById("decrypt").addEventListener("click", function (e) {

    e.preventDefault();
    e.stopPropagation();

    var ciphertext = document.getElementById("ciphertext").value;
    var shift = document.getElementById("backshift").value;

    document.getElementById("ciphertext").innerHTML = createAlphabeticalShift(ciphertext, shift);

});
```