Aufgabe 1: Wörter aufräumen

Team-ID: 00087

Team-Name: One Man Army

Bearbeiter dieser Aufgabe:
Merlin Moelter

8. September 2020

Inhaltsverzeichnis

Lösungsidee

Für alle Beispiele gilt:

- Jedes Wort im Lückentext besitzt entweder einen oder keinen vorgegebenen Buchstaben
- Alle Wörter können und müssen einmal verwendet werden

Zuerst soll mein Programm den Lückentext einlesen, sowie alle Wörter in eine Liste packen. Dann soll eine Regex alle Wörter im Lückentext und deren vorgegebene Buchstaben, falls gegeben, erkennen, und diese durch jeweils ein neues Wort ersetzen. Um ein solches Wort zu bestimmen filtert mein Programm alle Wörter aus der eingelesenen Wörterliste nach den Kriterien:

- Gleiche Länge wie das gesuchte Wort
- Übereinstimmung mit dem vorgegebenen Buchstaben

Wenn nur noch ein Wort übrigbleibt (Duplikate ausgeschlossen), so ist dieses das passende Wort für die Lücke und es wird aus der Liste der möglichen Wörter entfernt. Andernfalls ersetzt das Programm diese Lücke nicht.

Nach einer Iteration über den Text können noch einige Lücken frei sein, da zuerst eine Spätere gefüllt werden musste, um ein eindeutiges Ergebnis für die jeweilige zu erhalten. Aus dem Grund wird so oft über den Text iteriert, bis alle möglichen Wörter eingesetzt wurden.

Umsetzung

In Javascript existiert eine native Methode „replace“, die alle Treffer einer Regex auf einer Zeichenkette durch den Rückgabewert einer Funktion ersetzt. Das habe ich mir zunutze gemacht und sie auf den Lückentext mit der Regex `/_*(\w?)_*/g` angewendet. Sie lässt sich so lesen: Beliebig viele Unterstriche, gefolgt von einem oder keinem Buchstaben, gefolgt von beliebig vielen Unterstrichen.

Mit der Array-Methode „filter“ werden alle Wörter aus dem Array aller Wörter genommen, die die gleiche Länge wie das gesuchte Wort haben. Um Duplikate aus dem Ergebnis zu entfernen, wird das Array zum Set, und das Set zurück zum Array konvertiert.

Sofern ein Buchstabe im Wort vorgegeben ist, werden alle Wörter aus dem Ergebnis-Array entfernt, die an der Stelle des vorgegebenen Buchstabens einen anderen besitzen. Hierfür wird wieder „filter“ verwendet, und „indexOf“, was einem den ersten Index eines Buchstaben in einer Zeichenkette zurückgibt.

Wenn nur noch ein Element im Ergebnis-Array vorhanden ist, wird dieses aus dem Array der möglichen Wörter entfernt und zurückgegeben, wodurch das Wort im Lückentext durch dieses ersetzt wird.

Diese Prozedur wird anhand einer while-Schleife so lange wiederholt, bis keine möglichen Wörter mehr im Array vorhanden sind.

Beispiele

Das Skript wird mit der Nummer der Beispieldatei als Parameter aufgerufen

$ node index.js 0

oh je, was für eine arbeit!

$ node index.js 1

Am Anfang wurde das Universum erschaffen. Das machte viele Leute sehr wütend und wurde allenthalben als Schritt in die falsche Richtung angesehen.

$ node index.js 2

Als Gregor Samsa eines Morgens aus unruhigen Träumen erwachte, fand er sich in seinem Bett zu einem ungeheueren Ungeziefer verwandelt.

$ node index.js 3

Informatik ist die Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen, besonders der automatischen Verarbeitung mit Digitalrechnern.

$ node index.js 4

Opa Jürgen blättert in einer Zeitschrift aus der Apotheke und findet ein Rätsel. Es ist eine Liste von Wörtern gegeben, die in die richtige Reihenfolge gebracht werden sollen, so dass sie eine lustige Geschichte ergeben. Leerzeichen und Satzzeichen sowie einige Buchstaben sind schon vorgegeben.

Quellcode

```
/**
 * Textdatei einlesen und den Lückentext von den Wörtern trennen. Die
Wörter werden in ein Array separiert.
 */

const fs = require("fs")
const path = require("path")

const inputFileNumber = process.argv[2] || 0

const content = fs.readFileSync(path.join(__dirname, "beispieldaten",
`raetsel${inputFileNumber}.txt`), "utf-8")

let [text, words] = content.replace(/\r/g, "").split("\n")

words = words.split(" ")

/**
 * Lösen des Lückentextes
 */

while(words.length !== 0) {
    text = text.replace(/_*(\w?)_*/g, (match, letter) => {
        let possibleWords = words.filter(word => word.length ===
match.length)
        possibleWords = Array.from(new Set(possibleWords))

        if (letter) {
            possibleWords = possibleWords.filter(word =>
word[match.indexOf(letter)] === letter)
        }

        if (possibleWords.length === 1) {
            const result = possibleWords[0]
            words.splice(words.indexOf(result), 1)
            return result
        }

        return match
    })
}

console.log(text)
```