Einführung in Javascript und Node.js

Die folgenden Erklärungen und Aufgaben dienen zur Einführung in die Programmierung mit Javascript und Node.js. Diese Aufgaben sind für den **14.10.19** vorzubereiten. Die Aufgaben behandeln dabei grundlegende Konzepte, wie Variablen, Funktionen und Schleifen.

Einrichten der Node.JS Umgebung

Um mit Javascript programmieren zu können, muss zu Beginn die Software Node.js installiert werden. Diese können Sie, zusammen mit einer Anleitung zur Installation, hier finden [1]. Eine erfolgreiche Installation wird mittels Konsole und dem folgenden Befehl überprüft.

```
$ node -v
v8.12.0
```

Im Anschluss werden Javascript-Dateien mit der Endung *.js erstellt. Diese Dateien können dann mit dem folgenden Befehl (über die Konsole) ausgeführt werden:

```
$ node grundlagen_javascript.js
```

Kommentare und Ausgabe auf der Konsole

Wie in anderen Programmiersprachen auch, kann man in Javascript Kommentare schreiben, welche entweder ein- oder mehrzeilig sind.

```
// Diese Konstante wird für die Berechnung eines Kreisumfangs benötigt
const PI = 3.141592653589793;


/*
 * Nimmt die Variable message entgegen
 * und gibt diese auf der Konsole aus
 */
console.log(message);
```

Daten können über den Aufruf console.log() auf der Konsole ausgegeben werden.

Variablen und Datentypen

Javascript ist eine dynamisch typisierte Programmiersprache. Dies bedeutet, dass Javascript zwar einige Typen, wie Strings oder Integer kennt, allerdings diese nicht während der Laufzeit prüft. Außerdem lässt sich jeder Variable auch jeden Typ zuweisen. In anderen Programmiersprachen ist dies nicht ohne weiteres möglich. Ein Beispiel dafür ist Java, welche Sie in „Algorithmen und Programmieren“ kennen gelernt haben. Bei der Deklaration einer Variable in Javascript wird kein Datentyp angegeben.

```
let radius = 10; // Diese Variable ist nun eine Zahl
radius = "10"; // Jetzt ist diese Variable ein String
```

Variablen werden mit den Schlüsselworten var, let und const deklariert. Der Unterschied bei den Deklarationen liegt in der Sichtbarkeit einer Variable. Alle Variablen die mit dem Schlüsselwort var definiert werden können an jeder Stelle in Javascript verwendet werden. Deswegen bezeichnet man diese Art von Variablen auch als „globale Variablen“. Um Variablen in ihrer Sichtbarkeit einzuschränken, sowie um Fehler durch eine globale Sichtbarkeit vorzubeugen, wurden die Variablen let und const eingeführt. Dabei handelt es sich um Variablen mit einer lokalen Sichtbarkeit, welche auf einen Bedingungs- oder Funktionsblock limitiert sind. Das folgende Beispiel macht dies deutlich:

```
// ist global und überall verfügbar
var x = 0;

If ( x == 0) {
        // gibt 0 auf der Konsole aus
        console.log(x);

        let a = 5;

        // gibt 5 auf der Konsole aus
        console.log(a);
}

// existiert nicht, da lokal sichtbar (und wirft deswegen einen Fehler)
console.log(a)
```

Variablen mit dem Schlüsselwort const verhalten sich wie let, mit dem Unterschied, dass man einer const-Variable keinen neuen Wert zuweisen kann. Insgesamt kennt Javascript fünf wichtige Datentypen mit denen Variablen initialisiert werden können:

- Boolean
- Null
- Undefined
- Number
- String

Zusätzlich existieren noch zwei weitere Datentypen (Symbol und Object). Diese werden Sie im weiteren Verlauf des Moduls kennen lernen. Weiterführende Informationen zu Datentypen sind hier zu finden [2].

Operatoren und Bedingungen

Javascript stellt für den Umgang mit Variablen eine Reihe von Operatoren bereit, welche die gleichen Aufgaben wie bei anderen Programmiersprachen erfüllen. Eine ausführliche Anleitung ist hier zu finden [3].

Wie im vorherigen Beispiel zu erkennen ist, unterstützt Javascript auch Bedingungen. Diese Bedingungen nutzen die Schlüsselwörter if, else, sowie else if und unterscheiden sich syntaktisch nicht zu Bedingungen in Java.

Schleifen

Auch Schleifen lassen sich ähnlich wie in Java definieren. Auch hier gibt es die zwei klassischen Beispiele for- und while-Schleifen. Darüber hinaus gibt es noch die for-in-Schleife, for-of-Schleife sowie die do-while-Schleife. Beispiele für diese zusätzlichen Schleifen können Sie hier finden [4]. Die Syntax für for- und while-Schleifen sieht wie folgt aus:

```
for (let x = 0; x < 10; x++ {
        console.log(x);
}

let y = 0;
while ( y < 10) {
        console.log(y);
        y++;
}
```

Funktionen

Funktionen werden in Javascript üblicherweise über das Schlüsselwort function definiert. Wie in anderen Programmiersprachen auch können entsprechende Parameter übergeben und ein Wert zurück gegeben werden. Auch hier werden keine Typen definiert, wie man es vergleichsweise bei Methoden aus Java kennt. Ein wichtiger Unterschied zu anderen Sprachen ist die Tatsache das eine Funktion der Parameter einer anderen sein kann. Dies bezeichnet man als Callback. Das folgende Beispiel zeigt die Syntax von Funktionen und wie eine Funktion übergeben werden kann:

```
const log = function(message) {
        console.log(message);
};

const min = function(a, b, callback) {
        if ( a > b ) {
                callback(„a ist größer b“);
        } else if (a < b) {
```

```
                callback(„b ist größer a“);
        } else {
                callback(„die Werte sind gleich“);
        }
}

// gibt aus: „die Werte sind gleich“
min(5, 5, log);
```

Eine alternative Syntax für die Deklaration von Funktionen sind sogenannte „Arrow Functions“. Diese werden Sie im weiteren Verlauf des Modules kennen lernen.

Eingaben über die Konsole

Für die Eingabe über die Konsole empfiehlt sich das Module „readline“. Durch Module können Sie auf bereits fertige Funktionen zurückgreifen. Durch Module werden verschiedene Funktionen gruppiert und sind daher mit Namespaces zu vergleichen.

```
// Einbinden des readline moduls
const readline = require(‚readline');
const rl = readline.createInterface({
        input: process.stdin,
        output: process.stdout
});

rl.question('Was ist der Radius des Kreises?', function(answer) {
        console.log(`Answer: ${answer}`); // Hier geben wir die eingegebene answer aus
        rl.close();
});
```

Aufgaben

Aufgabe 1

Richten Sie NodeJS auf ihrem Arbeitsgerät ein und erstellen Sie eine Datei mit dem Namen: grundlagen_javascript.js

Öffnen Sie diese Datei und schreiben Sie Code, welcher ihnen ihren Namen auf der Konsole ausgibt und führen Sie diese Datei aus. Fangen Sie hier schon an, ihren Code zu kommentieren.

Aufgabe 2

Deklarieren Sie eine Konstante, welche die maximale Höhe einer möglichen Bewertung einer App für das Smartphone angibt (zum Beispiel Sterne im App Store). Außerdem deklarieren Sie zwei weitere Variablen, eine welche die aktuelle Anzahl der Bewertungen beinhaltet, und eine die die Bewertung selbst speichert. Weisen Sie diesen Variablen und ihrer Konstante nun Werte zu, mit denen sie in den folgenden Aufgaben arbeiten. Geben Sie diese Variablen auf der Konsole aus. Zusätzlich simulieren Sie einmal eine Bewertung und lassen die veränderten Werte wiederum ausgeben. Was macht Javascript, wenn Sie eine der Variablen einen anderen Typ zuweisen? Was passiert, wenn Sie ihrer Konstante, nachdem Sie diese deklariert haben, einen neuen Wert zuweisen?

Aufgabe 3

Erweitern Sie ihr Programm so, dass Sie die Bewertung einer App durch einen Benutzer simulieren. Achten Sie darauf, ob die abgegebene Bewertung mit ihrer Konstante korreliert. Lesen Sie die Bewertung über die Konsole ein. Testen Sie ihr Programm mit verschieden Werten, nehmen Sie auch non-Integer Werte. Geben Sie geeignete Fehlermeldungen auf der Konsole aus.

Aufgabe 4

Erweitern Sie ihr Programm so, dass Sie die Bewertung n mal berechnen, wobei Sie bei jeder Berechnung eine zufällige Bewertung verwenden. Geben Sie jedesmal die Anzahl der Bewertungen, die aktuelle Bewertung sowie die abgegebene Bewertung aus.

Aufgabe 5

Nun sollen Sie ihr Programm so erweitern, dass Sie die Berechnung der Bewertung in eine Funktion packen.

Quellen und Links

[1] Node.js - Download, Dokumentation und Installation (https://nodejs.org/en)
[2] JS Datenstrukturen: (https://developer.mozilla.org/de/docs/Web/JavaScript/Datenstrukturen)
[3] JS Ausdrücke und Operatoren (https://developer.mozilla.org/de/docs/Web/JavaScript/Guide/Ausdruecke_und_Operatoren)
[4] JS Schleifen (https://developer.mozilla.org/de/docs/Web/JavaScript/Guide/schleifen_und_iterationen)