JavaScript - Kurzüberblick

Variablen

In Google Apps Script können globale Variablen in der ersten Datei im Projektordner abgelegt werden. Globale Variablen werden per Konvention GROSS geschrieben.

Beispiel:

```
const GLOBALE_VARIABLE = 'hallo'; // String-Wert, immer in Anführungszeichen
let myVar = 1; // Zahl
```

const und let werden verwendet, um Variablen zu definieren. `const` können nicht geändert werden.

Funktionen

Einfache Funktionen

```
function meineFunktion() {
    Logger.log('irgendwas');
    let e = 'test';
    let f = 1;
}
```

'Logger' - Wird verwendet, um Informationen in der Konsole (CLI) zu loggen.

Übergabe von Werten an Funktion

```
function meineFunktion2(a, b) {
    Logger.log(a, b);
}
```

Funktion, die eine andere Funktion aufruft

```
function meineFunktion3() {
    const a = 1;
    const b = 'Test';
    meineFunktion2(a, b);
}
```

Funktion mit Rückgabewerten

```
function meineFunktion4(a, b) {
    Logger.log(a,b);
    return `${a} ${5 +5}`;
```

Objekte und Arrays

Objekte

Objekte --> mit geschweiften Klammern, im JSON-Format. Anführungsstriche be

```
const obj = {name:'Laurent'};
const obj2 = {'name':'Laurentia'};
const obj3 = {vorname: 'Laurenz', nachname : 'Meier'};

// der Nachname kann dann z.B. ausgegeben werden..
Logger.log(obj3.nachname);
// alternativ ...
Logger.log(obj3['nachname']);
```

Objekte mit Funktionen

Das Keyword "this" verweist auf das Objekt selbst.

```
const meinObjekt = {
    'vorname' : 'Markus',
    'nachname' : 'Schmidt',
    'vollstaendigerName' : function(){
        Logger.log('Der komplette Name: ' );
        return this.vorname + ' ' + this.nachname;
    }
}

let vollstName = meinObjekt.vollstaendigerName();
Logger.log(vollstName);
```

Arrays

Index startet bei 0.

```
const meinArray = ['Markus', 'Fatima', 'Sabine'];

// Ausgabe des ersten Eintrags.
Logger.log(meinArray[0]);

// ein Element hinzufügen
meinArray.push('Zania');
```

```
// Array andersherum
meinArray.reverse();

// sortieren
meinArray.sort();

// letztes Element löschen
const letztes = meinArray.pop();

// letzte ... hinzufügen
meinArray.push('neues letztes');

// erstes ... löschen
const erstes = meinArray.shift();

// erstes ... hinzufügen
meinArray.unshift('neues erstes');

//
const laenge = meinArray.length;
```

Schleifen und Verzweigungen

For- und While-Schleife

```
function output1() {
    for  (let i =0; i<10;i++){
        Logger.log('i: '+i);
    }

    let j = 0;
    while(j < 10) {
        Logger.log('j:' +j);
        j++;
    }
}
```

Durch ein Array iterieren.

```
const arr = ['Müller', 'Meier', 'Schulze'];
function output2() {
    for  (let i = 0; i<arr.length; i++){
        Logger.log('Array-Wert: ' + arr[i]);

    }
}
```

Die folgende Methode kann zum Iterieren durch ein Array verwendet werden. Dabei wird eine Funktion hinter dem => definiert, die auf jedes Element angewandt wird.

```
arr.forEach((element)=>{
    Logger.log(element);
})

// Ausdrucken auch des Indexes
arr.forEach((, index)=>{
    Logger.log(element, index);
})

// Arrays mit Objekten füllen
const myArray = []; // leeres Array
  for(let q=0; q<10; q++){
    const temp = {
      'first' : 'Myname'+q,
      'last' : 'Laster'+q
    };
    myArray.push(temp);
  }

  myArray.forEach((person)=>{
    Logger.log(person.first + person.last);
  })
```