LU01c - Funktionen

Funktionen sind in JavaScript fundamentale Bausteine und werden verwendet, um wiederverwendbare, strukturierte Blöcke von Code zu schreiben, die eine bestimmte Aufgabe erfüllen. Sie sind ein zentraler Bestandteil der Programmierlogik und können als First-Class Citizens (Erstklassige Objekte) betrachtet werden, was bedeutet, dass Funktionen wie andere Datenstrukturen behandelt werden können: Sie können Variablen zugewiesen, als Argumente an andere Funktionen übergeben und sogar von anderen Funktionen zurückgegeben werden.

Funktionserklärungen (Function Declarations)

Die klassische Methode, eine Funktion zu definieren, ist die Funktionserklärung (Function Declaration). Dies ist der traditionellste Weg, eine Funktion zu schreiben. Eine Funktionserklärung beginnt mit dem Schlüsselwort function, gefolgt vom Namen der Funktion und einer Liste von Parametern in Klammern.

```
function addiere(x, y) {
  return x + y;
}
```

In diesem Beispiel wird eine Funktion addiere erstellt, die zwei Parameter, x und y, erwartet. Sie gibt die Summe dieser beiden Parameter zurück.

Besonderheit

Funktionserklärungen werden gehoisted, was bedeutet, dass sie im gesamten Gültigkeitsbereich verfügbar sind, noch bevor der eigentliche Code ausgeführt wird. Du kannst die Funktion also schon aufrufen, bevor sie im Code definiert wurde.

```
console.log(addiere(3, 4)); // Funktion wird vor ihrer Definition aufgerufen

function addiere(x, y) {
  return x + y;
}
```

Das obige Beispiel funktioniert, weil JavaScript Funktionserklärungen beim Start des Codes automatisch „nach oben verschiebt“.

Funktionsausdrücke (Function Expressions)

Ein Funktionsausdruck ist eine alternative Methode, um eine Funktion zu definieren. Dabei wird die Funktion als Wert einer Variablen zugewiesen. Anders als bei Funktionserklärungen werden Funktionsausdrücke nicht gehoisted. Das bedeutet, die Funktion ist erst verfügbar, nachdem der Code die Zuweisung erreicht hat.

```
const multipliziere = function(x, y) {
  return x * y;
};

console.log(multipliziere(3, 4)); // 12
```

Hier wird die anonyme Funktion (eine Funktion ohne Namen) der Variablen multipliziere zugewiesen. Beachte, dass du diese Funktion erst nach der Definition aufrufen kannst.

Arrow-Funktionen (Arrow Functions)

Arrow-Funktionen sind eine modernere, kürzere Syntax, um Funktionen zu schreiben. Sie wurden mit ES6 (ECMAScript 2015) eingeführt und unterscheiden sich in mehreren Aspekten von herkömmlichen Funktionen. Eine Arrow-Funktion verwendet das ⇒-Symbol und verzichtet auf das Schlüsselwort function.

```
const addiere = (x, y) => {
  return x + y;
};
```

Dieses Beispiel ist funktional identisch mit der vorherigen Funktionsdefinition, nur mit einer kürzeren Syntax. Wenn eine Funktion nur einen Rückgabewert hat, kann sie noch weiter vereinfacht werden:

```
const addiere = (x, y) => x + y;
```

Eigenschaften von Arrow-Funktionen

- **Kürzere Syntax:** Arrow-Funktionen sind kompakter und erfordern kein function-Schlüsselwort.
- **Kein eigenes this:** Arrow-Funktionen haben kein eigenes this. Sie übernehmen das this des umgebenden Kontexts, was sie besonders nützlich in Objektmethoden oder in Callbacks macht.

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/ffit/js/learningunits/lu01/funktionen?rev=1729745946

Last update: **2024/10/24 06:59**