JS Callback-Funktionen

Lernziele

- Verstehen des Konzepts von Callback-Funktionen
- Verwenden einer anonymen Callback-Funktion
- Verwenden einer benannten Funktion als Callback-Funktion
- Wissen, was eine Higher Order Function ist

Callback-Funktionen

Eine Callback-Funktion ist eine Funktion, die **als Argument** in eine andere Funktion übergeben wird.

Die äußere Funktion kann diese Callback-Funktion zum richtigen Zeitpunkt oder mehrmals ausführen, zum Beispiel:

- wenn ein Ereignis ausgelöst wird
- wenn die abgerufenen Daten auf Ihrem Computer angekommen sind
- für jedes Element in einem Array.

Callback-Funktionen werden verwendet, wenn das Programm selbst herausfinden muss, **wann** oder **wie oft** die Funktion ausgeführt werden muss. Wir haben bereits Callback-Funktionen in **Event-Listenern** verwendet:

```
button.addEventListener("click", () => {
  console.log("Innerhalb der Callback-Funktion.");
});
```

Hier ist die Struktur wie folgt:

- äußere Funktion: `addEventListener()`
- erstes Argument: `'click'`
- zweites Argument: Callback-Funktion

```
() => {
  console.log("Innerhalb der Callback-Funktion.");
};
```

Diese Art von Funktion wird **anonyme Funktion** genannt, da sie ohne Namen deklariert wird.

Benannte Callback-Funktionen

Jede Funktion kann als Callback-Funktion verwendet werden. Sie muss nur an eine andere Funktion übergeben werden. Sie können eine normale Funktion deklarieren und dann den **Namen der Funktion**

verwenden, um sie in eine andere Funktion zu übergeben:

```
function sayHello() {
  console.log("Hey Dude!");
}

button.addEventListener("click", sayHello);
```

> ❗ Beachten Sie, dass wir die Funktion hier nicht aufrufen (wir haben `sayHello` anstelle von `sayHello()` geschrieben). Wir übergeben die Funktion nur an den Event-Listener. Die Funktion wird nur aufgerufen, wenn das Ereignis eintritt.

Higher Order Functions

Eine Higher Order Function ist eine Funktion, die eine **Callback-Funktion als Argument** annimmt und die **Callback-Funktion** innerhalb ihres Körpers **aufruft**, z.B. die `addEventListener`-Methode.

```
// Diese Funktion ruft ihre Callback-Funktion 3 Mal auf!
function myHigherOrderFunction(callback) {
  callback();
  callback();
  callback();
}
```

Wir werden diese Higher Order Functions in zukünftigen Sitzungen noch kennenlernen:

- `.then`
- `.forEach``
- `.map`
- `.filter`

> 💡 Keine Sorge, Sie müssen keine Higher Order Functions selbst schreiben, Sie wenden sie nur an, um bestimmte Probleme zu lösen.

Parameter in Callback-Funktionen

Eine Callback-Funktion kann Parameter akzeptieren. Die Werte für die Parameter werden von der Funktion bereitgestellt, die die Callback-Funktion aufruft (die "Higher Order Function").

In diesem Beispiel kann die Callback-Funktion einen Parameter akzeptieren, um Informationen über das aufgetretene Ereignis zu erhalten:

```
button.addEventListener("click", (event) => {
  console.log("Dieser Button wurde geklickt:", event.target);
});
```

Ressourcen

- MDN Callback Functions