

Modul 323 – Funktional Programmieren

Vorwort:

Grundlage für diese Aufgabe ist die Präsentation «Funktionale Programmierung»

Dauer: 90 Minuten

Optional: 0 Minuten

Total: 90 Minuten

Nachfolgend werden Sie sich mit den Grundprinzipien von funktionaler Programmierung auseinandersetzen. Dabei werden Sie sich mit unveränderlichen Daten beschäftigen und wie diese trotzdem auf eine spezielle Art modifiziert werden können. Weiter werden Sie sich mit Funktionen auseinandersetzen.



Der aus diesem Arbeitsblatt entstehende Source Code soll in ein Github-Repository eingeecheckt werden und der Link in die folgende Text Box geschrieben werden:

https://github.com/Kaenu/ab2_modul323

Aufgabe 1)

In dieser Aufgabe beschäftigen Sie sich mit unveränderlichen Objekten und wie diese eingesetzt werden können. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**immutable_objects.js**».

Worum handelt es sich bei unveränderlichen Daten und was bedeutet das genau?

Unveränderliche (immutable) Daten beziehen sich auf Daten, die sich einmal festgelegt haben und nicht mehr geändert werden können. Dies bedeutet, dass ein bestimmter Wert oder eine bestimmte Eigenschaft eines Objekts nicht mehr geändert werden kann, nachdem es zuvor festgelegt wurde. Unveränderliche Daten sind sehr nützlich, wenn bekannt ist, dass bestimmte Daten nicht geändert werden sollen, um eine konsistente Programmlogik aufrechtzuerhalten.

Erstellen Sie eine Variable, welche einen primitiven Datentyp abbildet und nicht neu zugewiesen werden kann.

```
const immutableString = "Das ist nicht mehr veränderbar";
```

Erstellen Sie eine konstante Variable, welche ein Objekt als Wert enthält und mindestens 3 Eigenschaften aufweist.

Fügen Sie dem zuvor erstellten Objekt nach dem Prinzip der «unveränderlichen» Daten eine Eigenschaft hinzu.

```
myData.groesse = '1.67';
```

Verändern Sie eine der Eigenschaften an dem dem zuvor erstellten Objekt nach dem Prinzip der «unveränderlichen» Daten.

```
myData.job = 'Applikationsentwickler';
```

Entfernen Sie eine beliebige Eigenschaft aus dem zuvor erstellten Objekt nach dem Prinzip der «unveränderlichen» Daten.

```
delete myData.alter;
```

Aufgabe 2)

In dieser Aufgabe beschäftigen Sie sich mit unveränderlichen Arrays und wie diese eingesetzt werden können. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**immutable_arrays.js**».

Erstellen Sie einen Array mit mindestens beliebigen Elementen welche Objekte darstellen und eine Eigenschaft als eindeutiger Identifikator haben.

```
var goodPlayers = [  
  {name: 'Lionel Messi', id: '1'},  
  {name: 'Cristiano Ronaldo', id: '2'},  
  {name: 'Neymar Jr.', id: '3'},  
  {name: 'Kylian Mbappe', id: '4'}  
];
```

Erstellen Sie einen neuen Array, welcher ein zusätzliches Element enthält.

```
var goodPlayersDetails = [  
  {name: 'Lionel Messi', id: '1', age: 33},  
  {name: 'Cristiano Ronaldo', id: '2', age: 35},  
  {name: 'Neymar Jr.', id: '3', age: 28},  
  {name: 'Kylian Mbappe', id: '4', age: 22}  
];
```

Erstellen Sie einen neuen Array, dieser soll die gleichen Elemente wie zuvor enthalten doch einer der Elemente soll eine Eigenschaft aktualisiert oder verändert haben.

 Tipp: Verwenden Sie «map»

```
var goodPlayersDetailsUpdated = goodPlayersDetails.map(player => {  
  if (player.name === 'Kylian Mbappe') {  
    player.age = 23;  
  }  
  return player;  
});
```

Erstellen Sie einen neuen Array, dieser soll das Element, mit dem kleinsten eindeutigen Identifikator entfernt haben.

i Tipp: Verwenden Sie «filter»

```
var goodPlayersWithoutSmallestId = goodPlayersDetails.filter(player => {
  return player.id !== '1';
});
```

Erstellen Sie einen neuen Array aus Zahlen, welcher nachfolgende Werte enthält. Berechnen Sie danach den Durchschnitt der Werte.

const reviews = [4.5, 4.0, 5.0, 2.0, 1.0, 5.0, 3.0, 4.0, 1.0, 5.0, 4.5, 3.0, 2.5, 2.0];

i Tipp: Verwenden Sie «reduce»

```
const reviews = [4.5, 4.0, 5.0, 2.0, 1.0, 5.0, 3.0, 4.0, 1.0, 5.0, 4.5, 3.0, 2.5, 2.0];

/**
 * sum = Wert der Summe im aktuellen Element
 * cur = aktuelle Wert des Arrays
 * Der Gesamtwert wird durch die Division des Gesamtwertes durch die Anzahl der Elemente im
 * Array ausgegeben.
 */
const avgReviews = reviews.reduce((sum, cur) => sum + cur) / reviews.length;
console.log(avgReviews);
```

Verwenden Sie die Werte von der Vorherigen Aufgabe. Erstellen Sie eine Funktion, welche die Werte gruppiert und zählt. Diese sollen folgendermassen eingeordnet werden:

- 4 oder grösser = good
- 2.5 oder grösser = ok
- kleiner = bad

i Tipp: Verwenden Sie «reduce»

```
const countReviews = reviews.reduce((sum, cur) => {
  if (cur >= 4) {
    sum.good++;
  } else if (cur >= 2.5) {
    sum.ok++;
  } else {
    sum.bad++;
  }
  return sum;
}, {
  good: 0,
  ok: 0,
  bad: 0
});
```

```
console.log(countReviews); // { good: 5, ok: 4, bad: 5 }
```

Aufgabe 3)


In dieser Aufgabe beschäftigen Sie sich mit «Currying» von Funktionen und verwenden Funktionen als «First Class Objects». Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**currying.js**».

Schreiben Sie eine Funktion, welche den Nachfolgenden Array von diesem Format:

```
const studentGrades = [  
  {name: 'Joe', grade: 88},  
  {name: 'Jen', grade: 94},  
  {name: 'Steph', grade: 77},  
  {name: 'Allen', grade: 60},  
  {name: 'Gina', grade: 54},  
];
```

In dieses Format transformiert:

```
const studentFeedback = [  
  'Nice Job Joe, you got an b',  
  'Excellent Job Jen, you got an a',  
  'Well done Steph, you got an c',  
  'What happened Allen, you got an d',  
  'Not good Gina, you got an f',  
];
```

 Tipp: Verwenden Sie «currying» und «map»

```
const studentGrades = [  
  {name: 'Joe', grade: 88},  
  {name: 'Jen', grade: 94},  
  {name: 'Steph', grade: 77},  
  {name: 'Allen', grade: 60},  
  {name: 'Gina', grade: 54},  
];  
  
const gradeLetter = (grade) => {  
  if (grade >= 90) return 'a';  
  if (grade >= 80) return 'b';  
  if (grade >= 70) return 'c';  
  if (grade >= 60) return 'd';  
  return 'f';  
}  
  
const gradeComment = (name, grade) => `Nice Job ${name}, you got an ${gradeLetter(grade)}`;  
  
const studentFeedback = studentGrades.map(student => gradeComment(student.name,  
student.grade));  
  
console.log(studentFeedback);
```

Laden Sie sich sich «Ramda» herunter und speichern Sie diese in einer Datei «ramda.min.js». Importieren Sie danach die Library und verwenden Sie diese nun für die folgende Aufgabe. Erstellen Sie einen Array, welcher aus mehreren Elementen des Typen String besteht. Erstellen Sie danach eine «Curry» Funktion, welche vor jedes Element ein bestimmtes Präfix setzt. Rufen Sie danach die Funktion entsprechend auf, sodass Sie bei der Übergabe in map keinen Parameter mehr mitgeben müssen.

<https://cdnjs.cloudflare.com/ajax/libs/ramda/0.25.0/ramda.min.js>

i Tipp: Verwenden Sie «Partial Application» und «map»

```
const r = require('./ramda.min.js');
const array = ["Messi", "Neymar", "Vinicius"];
const prefixFunction = r.curry((prefix, item) => prefix + item);
const prefixedArray = r.map(prefixFunction('Meine Lieblingsspieler sind: '), array);
console.log(prefixedArray);
```

Erstellen Sie eine einfache Funktion, welche die Länge eines Arrays zurückgibt, welche aus Ihrer Sicht «Impure» ist. Schreiben Sie diese dann entsprechend um, damit sie «Pure» wird.

```
// Impure function
function arrayLength(array) {
  return array.length;
}

// Pure function
function arrayLength(array) {
  let length = 0;
  for (let i = 0; i < array.length; i++) {
    length++;
  }
  return length;
}
```

Aufgabe 4)

In dieser Aufgabe beschäftigen Sie sich mit «Composition» von Funktionen. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**functions_compositions.js**».

Schreiben Sie eine Komposition von Funktionen, welche in nachfolgendem Satz:

```
const sentence = 'PechaKucha is a presentation style in which 20 slides are shown for 20 seconds each (6 minutes and 40 seconds in total).';
```

Die Anzahl von numerischen Zeichen zählt. Das erwartete Ergebnis ist 7.

i Tipp: Verwenden Sie «isNan('.')» sowie «R.compose» oder «R.pipe»

--

Aufgabe 5) (Optional)

Falls Sie mit dem Arbeitsblatt bereits fertig sind, können Sie sich den nachfolgenden Onlinekurs anschauen und selbstständig anfangen die Übungen zu erledigen. Das kann Ihnen zusätzlich helfen, in das ganze Thema hereinzukommen.

<https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/>

💡 Um die Übungen und den Fortschritt speichern zu können, müssen Sie einen gratis Account auf freecodecamp erstellen.