**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 werdenp Sie sich mit Funktionen auseinandersetzen.

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

|  |
| --- |
|  |

**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?**

|  |
| --- |
| Sind Daten, die nicht mehr geändert werden können, wenn sie einmal erstellt wurden. |

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

|  |
| --- |
|  |

**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.**

|  |
| --- |
|  |

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

|  |
| --- |
|  |

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

|  |
| --- |
|  |

**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 ein Array mit mindestens zwei beliebigen Elementen welche Objekte darstellen und eine Eigenschaft als eindeutiger Identifikator haben.**

|  |
| --- |
|  |

**Erstellen Sie ein neues Array, welches ein zusätzliches Element enthält.**

|  |
| --- |
|  |

**Erstellen Sie ein neues Array, dieses soll die gleichen Elemente wie zuvor enthalten doch einer der Elemente soll eine Eigenschaft aktualisiert oder verändert haben.**

ℹ️ Tipp: Verwenden Sie «map»

|  |
| --- |
|  |

**Erstellen Sie ein neues Array, dieses soll das Element, mit dem kleinsten eindeutigen Identifikator entfernt haben.**

ℹ️ Tipp: Verwenden Sie «filter»

|  |
| --- |
|  |

**Erstellen Sie ein neues Array aus Zahlen, welches 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];

ℹ️ Tipp: Verwenden Sie «reduce»

|  |
| --- |
|  |

**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**

ℹ️ Tipp: Verwenden Sie «reduce»

|  |
| --- |
|  |

**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 das Nachfolgende 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',

];

ℹ️ Tipp: Verwenden Sie «currying» und «map»

|  |
| --- |
|  |

**Laden Sie sich sich «Rambda» herunter und speichern Sie diese in einer Datei «rambda.min.js». Importieren Sie danach die Library und verwenden Sie diese nun für die folgende Aufgabe.**

**Erstellen Sie ein Array, welches 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>

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

|  |
| --- |
|  |

**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.**

|  |
| --- |
|  |

**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.**

ℹ️ 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.