24.10.2023
1. & 2. VL

# JavaScript (Übung)

Prof. Dr. Walter Kern


Hochschule Hof
University of Applied Sciences

Rechtliche Hinweise

# Unterlagen zur Vorlesung

Diese Präsentation ist einschließlich aller ihrer Teile und zugehöriger Vorlesungsaufzeichnungen nur für den privaten Gebrauch der Studierenden der Fakultät Informatik an der Hochschule Hof bestimmt. Eine Weitergabe an Dritte (auch von Teilen) oder eine Veröffentlichung (auch eine Veröffentlichung im Inter- oder Intranet) ist nicht zulässig. Dieses Weitergabe- und Veröffentlichungsverbot schließt auch nicht öffentliche bzw. auf bestimmte Nutzer oder Gruppen beschränkte Bereiche von Austauschplattformen, Foren oder Dienste wie Dropbox mit ein.

# 1. Übungseinheit

- **Übung 1: Erzeugen Sie mehrere (mind. 3) Personen-Objekte per Literal mit den Eigenschaften firstName, lastName, age, adresses, wobei addresses ein Array mit Objekten vom Typ Address (Eigenschaften description, location, town, postalcode) ist**

  -> Einheitliche Daten für spätere Übungen:

  - Max Mustermann (19)
  - Chris May (21)
  - Steve Clover (25)
  - Adressen haben in Description z.B. Primary Address, Secondary Address oder Secret Address

Hochschule Hof
University of Applied Sciences

# 2. Übungseinheit

- **Übung 2: Erzeugen Sie ein Array aller Personen auf mindestens 5 verschiedene Arten**
- **Übung 3: Geben Sie die Nachnamen aller Personen des Arrays mit mindestens 7 konzeptionell verschiedenen Schleifen aus**
- **Übung 4: Geben Sie den Namens jedes Schleifenelements auf funktionale Art und Weise auf mindestens 3 verschiedene Arten aus (eine Variante soll aus einer eigenen Funktion (als forEach-Ersatz) mit einem Array und einer Funktion als Eingabeparameter bestehen)**
- **Übung 5: Erzeugen Sie ein Array mit den Zahlen von 0 bis 9.**
- **Übung 6: Erzeugen Sie mit den Quadratzahlen von 1 bis 10.**

Hochschule Hof
University of Applied Sciences

# 3. Übungseinheit

- **Übung 1: Extrahieren Sie das mittige Element des Arrays und a) speichern Sie es in ein neues Array bzw. b) ersetzen Sie es durch einen neuen Wert**
- **Übung 2: Entfernen Sie das erste Element aus dem Array ohne ein neues einzufügen**
- **Übung 3: Fügen Sie eine neue Person in das Array mittig ein ohne etwas zu löschen**
- **Übung 4: Erstellen Sie ein Array mit denjenigen Personen, die älter als 20 sind (mit mindestens 3 verschiedenen Ansätzen - beginnend bei Iteration) und geben Sie die Ergebnisse jeweils über eine generische Funktion aus**

Personen, die älter als 20 sind, herausfiltern/

'-> filter(), ...

[...] = Array
{...} = Objekt

Hochschule Hof
University of Applied Sciences

# 3. Übungseinheit

- **Übung 5: Bestimmen Sie das Durchschnittsalter der Personen (iterativ und funktional) auf 2 Nachkommastellen gerundet**
  '-> Aufgabe durch Einzeiler lösen

  mit map() & reduce() arbeiten; funktioniert auch nur mit reduce()
  map(): kann für jedes Element im Ursprungsarray ein neues Element im Zielarray zurückgeben
  reduce(): kann aufsummieren, ... = Aggregat-Operationen machen

- **Übung 6: Iterieren sie mit for, for-in und for-of über das Personen-Array und geben Sie den Index der Elemente sowie den Namen mit aus (mindestens 5 Ansätze - Nicht-Index-Werte sind auszuschließen)**

- **Übung 7: Konvertieren Sie einen String mit einer Nummer 123.15 in eine Variable vom Typ number (mindestens 3 Ansätze)**

Hochschule Hof
University of Applied Sciences

# 4. Übungseinheit

- **Übung 1: Geben Sie zurück, ob jede Person a) mindestens eine Primäradresse hat bzw. b) ob mindestens eine Person mindestens eine Secret Address hat**

prüfen, ob Element mind./überhaupt nicht in einem anderen vorkommt
,-> muss mit every() & some() arbeiten
some() = every() = some() = some()
<-- kann man mit Einzeiler returnen
<-- Satz nacheinander 1:1 übersetzen in Code

- **Übung 2: Erzeugen Sie ein neues Array, welches Personen mit folgender Struktur fasst (iterativ und funktional) – Pseudocode: [{name = "Max Mustermann", birthday = ..., addresses = 3}]**

enthält firstName & lastName
mit map()
Ursprungs-/Personenarray transformieren in neues Array, das auch Personenobjekte enthält, aber anders aufgebaut:
transformieren zu diesem Aufbau:
enthält nur noch name
- so berechnen: aktuelles Datum nehmen & age abziehen
- vorher: age
Anzahl statt Array von Adressen/ verschachtelte Adresse

- **Übung 3: Sortieren das neue Array nach dem Anfangsbuchstaben des Namens der Person absteigend (bei a) und b) jeweils ohne Vergleichsoperation!)**
  **a) Mit normaler Funktion (eigene Lösung, z.B. über BubbleSort)**
  **b) mit sort-Funktion und anonymer Funktion als Komparator**
  **c) mit sort-Funktion und mit Lambda-Funktion**

# 4. Übungseinheit

ewas schwieriger: hier geht es um
Closures = innere Funktion wird von äußeren Funktion returned & Umgebung wird gesnappshottet --> das nutzen, um funktionale Factory umzusetzen

- **Übung 4: Schreiben Sie eine Funktion Smartphone, die für einen bestimmten Hersteller (Eingabeparameter) eine Funktion zur Erzeugung spezifischer Modelle (Eingabeparameter) zurückgibt:**

```
const samsungSmartphoneFactory = Smartphone("Samsung");
const gS10=samsungSmartphoneFactory("Galaxy S10");
console.log(`${gS10.producer} ${gS10.model}`);
```

,-> hier soll etwas zurückkommen: Funktion wird returned, ...
gesnappshottete Procedure-Eigenschaft wird in der Referenz gespeichert
,-> ... der man Produktnamen übergibt --> bekommt Objekt
'-> ruft es mit () auf --> = innere Funktion
konkreter Wert wird in model gespeichert & kann darüber abgerufen
hat diese Eigenschaften

- **Übung 5 (optional): Erweitern Sie die Lösung für Übung 4 so, dass mehrere Hersteller spez. werden können und sie sich den Namen (in d. Form „producer model“) ausgeben lassen können:**

```
const smartphonesFactory=Smartphones("Samsung","Sony");
console.log(smartphonesFactory["Samsung"]
  ("Galaxy S10").getFullname());
console.log(smartphonesFactory["Sony"]
  ("Xperia 10").getFullname());
```

In Java:
kann Lambdaausdruck returnen

Hochschule Hof
University of Applied Sciences

# 4. Übungseinheit

- **Übung 6: Erstellen Sie entsprechende Konstruktorfunktionen, sodass folgende Anweisung funktioniert:**

```
const person = new Person("Max", 21, new
  Address("Teststreet 1", "Hof", 95028));
```

- **Übung 7: Erweitern Sie die Lösung für Übung 6 so, dass die obenstehende Anweisung auch ohne new-Schlüsselwort funktioniert**

- **Übung 8: Erzeugen Sie eine Instanz des in den vorherigen Übungen erstellten Person-Objekts mit Object.create und initialisieren Sie dabei alle Eigenschaften des Person-Objekts**

Hochschule Hof
University of Applied Sciences

# 5. Übungseinheit

Vererbung mit Prototypen nachbauen/
Prototyp-basierte Vererbung umsetzen =
'-> braucht man öfters
'-> dazu vielleicht etwas in Prüfung!!!

- **Übung 1: Bauen Sie eine Vererbungshierarchie mit den Objekten Animal (Basisobjekt), Dog und Cat auf.**

  Realisieren Sie das ganze über den klassischen prototype-Ansatz, wobei die Objekte wie folgt genutzt werden können sollen:

'-> animal-function
'-> dog-function
'-> cat-function

```
let animal = new Animal("Tier", 5);
console.log(animal.getName()); // Tier
animal = new Dog("Benno", 10, "Bulldogge");
animal.speak(); // Benno sagt Wuff!
animal = new Cat("Minki", 15, "schwarz");
animal.speak(); // Minki sagt Miau!
console.log(Animal.instanceCount); // 3
```

Konstruktorfunktion bauen
&
Vererbungshierachie aufbauen über: Object.create(...)
&
Funktionen ergänzen

es soll Variable geben, die direkt an animal hängt & nicht an den einzelnen Instanzen

verwenden, wenn Code größtenteils funktional ist

beide Arten der Vererbung sollte man kennen

rein Prototyp-basierte Lösung

damit hat man viele Freiheiten

in Klassen-basierten Ansatz/Variante/Syntax übersetzen/
alten Code nehmen & anpassen

~ Java8-Code

verwenden, wenn Code größtenteils objektorientiert ist (z.B.: bei Angular)

- **Übung 2: Setzen Sie die vorhergehende Übung analog mit dem neuen class-Ansatz um.**

class-Syntax = Fake
'-> um es allen Entwickler, v. a. OOP-Entwickler, Recht zu machen
'-> wird als Funktion umgesetzt: typeof(Person) => function

class Person {...}

Hochschule Hof
University of Applied Sciences

# 5. Übungseinheit

String-Prototype-Iterator/
Standarditerator von String/

- **Übung 3: Überschreiben Sie den String-Iterator so, dass beim Iterieren mit for-of über einen String dieser in der umgekehrten Reihenfolge ausgegeben wird (ohne Generator).**

umgekehrt über String iterieren/

Reverse-Iterator: iterieren von hinten über die Elemente

```
let result = "";
for (const c of "hello")
  result+=c;
console.log(result); // olleh
```

= umgedrehter String/umgekehrte Reihenfolge

- das umständliche mit createIterator() muss man nicht machen
- mit einfachen Generator arbeiten

- **Übung 4: Implementieren Sie den String-Iterator von der vorhergehenden Übung so, dass nun ein Generator zum Einsatz kommt.**

Hochschule Hof
University of Applied Sciences

# 5. Übungseinheit

- **Übung 5: Destrukturieren Sie das folgende Array in 3 Variablen zahl1, string1 u. restlicheWerte (übergehen Sie das 2. Element)**

```
const arr = [1, 2, "Hallo", 3, 4, true, "ok"];
```

--> destrukturieren & dazu passende Variablen anlegen

Destrukturierung:
- gibt es auch in Kotlin, C#, Swift, C++
- gibt es nicht in Java

- **Übung 6: Vertauschen Sie n1 und n2 per Destrukturierung**

```
let n1 = "Alex"; let n2 = "Chris";
```

- **Übung 7: Weisen Sie den Variablen isbn, title, primaryStreet und town über eine Destrukturierungszuweisung die jeweiligen Werte aus book zu u. vergeben Sie für title einen Standardwert**

```
const book = { isbn: "34504350305",
  author: {
    name: "Max Mustermann",
    address: { street: "Teststreet 1", town: "Hof" }
  }} console.log(isbn, title, primaryStreet, town);
```

Hochschule Hof
University of Applied Sciences

# 5. Übungseinheit

- **Übung 8: Destrukturieren Sie in einer for-of-Schleife so, dass sie innerhalb der Schleife unmittelbar auf den authorName als Variable zugreifen können, wobei bei fehlendem Author „Unbekannt“ als Name ausgegeben werden soll**

```
const books = [
  { isbn: "34504350305", author: {
    name: "Max Mustermann",
    address: {street: "Teststreet 1", town: "Hof"}}},
  { isbn: "123453535", author: {
    name: "Chris Mustermann",
    address: {street: "Teststreet 2", town: "Hof"}}},
  { isbn: "345353453458" }
];
for (… of books)
  console.log(authorName);
```

Hochschule Hof
University of Applied Sciences

# 5. Übungseinheit

- **Übung 9: Erzeugen Sie über Spreading ein neues Array, welches die Elemente der beiden Arrays sowie die Zahlen 3 und 6 in aufsteigender Reihenfolge enthält.**
  **Machen Sie das gleiche im Anschluss über die Array.prototype.concat-Methode**

```
const arr1 = [1, 2];
const arr2 = [4, 5];
const arr3 = …;
console.log(arr3); // [ 1, 2, 3, 4, 5, 6 ]
```

- **Übung 10: Was ist der Unterscheid der beiden folgenden Zuweisungen?**

```
const person = { name: "Max Mustermann", age: 21 };
const extendedPerson1 = { person };
const extendedPerson2 = { ...person };
```

Hochschule Hof
University of Applied Sciences

# 5. Übungseinheit

- **Übung 11: Lagern Sie die Klassen Animal, Dog und Cat aus einer früheren Übung aus auf ein Modul namens Exercise5_Classes. Erstellen Sie ein Modul Exercise5_Utils, indem sie eine Funktion getAnimalType(animal) definieren, die den Typ des übergebe-nen Objekts via animal.constructor.name zurückgibt. Binden Sie die Module ein, sodass folgender Code (im Firefox) funktioniert:**

```
let animal = new Exercise5_Classes.Animal("Tier", 5);

animal = new Exercise5_Classes.Dog("Benno", 10,
  "Bulldogge"); animal.speak(); // Benno sagt Wuff!
animal = new Exercise5_Classes.Cat("Minki", 15,
  "Siamkatze"); animal.speak(); // Minki sagt Miau!

console.log(animal.getName()+" is a "+getAnimalType(
  animal).toLowerCase() + "!"); // Minki is a cat!
```

# 5. Übungseinheit - Kleine Anwendung bauen

in Übungen bauen wir
immer so eine CRUD-App

- **Übung 12: Erstellen Sie eine Anwendung, die in einer Liste eine Menge von Personen anzeigt und per über Eingabefelder und einen Button die Ergänzung weiterer Personen erlaubt sowie über Buttons neben den jeweiligen Einträgen der Liste ein Entfernen und Bearbeiten ermöglicht.**

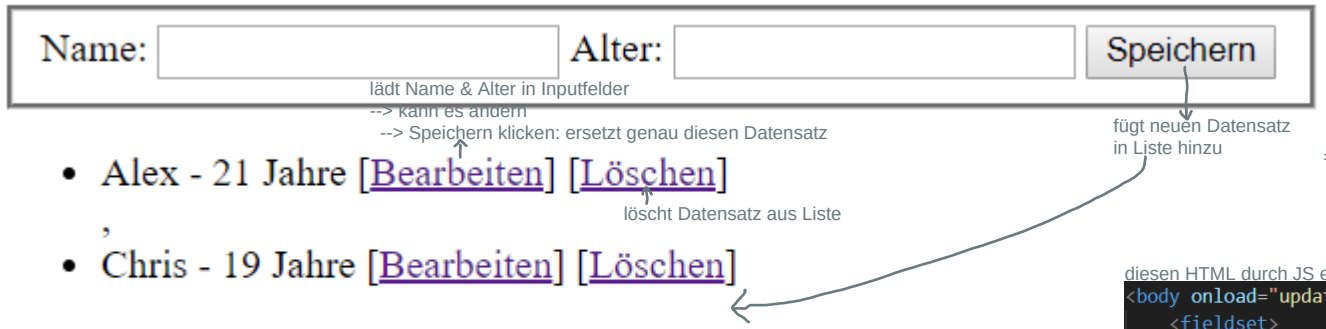
  Die Daten zur dargestellten Liste sollen in einem Array mit Person-Objekten verwaltet werden.

so eine umfangreiche Aufgabe kommt
in Klausur nur in React dran

Anwendung/
Application
= CRUD-App / Personlist-App: hinzufügen/bearbeiten/löschen von Person mit purem JS mit modernen Ansatz
ohne Framework <-- kommt später

,-> ohne Framework
alter Ansatz: DOM-Manipulation: document.getElementById(...), document.querySelectorAll(...), ...

jQuery: kann $-Selector durch Einzeiler ersetzen
heute: sollte React oder modernere Ansätze benutzen mit denen man direkt Daten im DOM-Baum manipuliert




diesen HTML durch JS erweitern:

```
<body onload="updateList();">
    <fieldset>
        <input id="id" type="hidden" value="">
        <label for="name">Name</label>: <input type="text" id="name">
        <label for="age">Alter</label>: <input type="text" id="age">
        <button onclick="saveItem();">Speichern</button>
    </fieldset>
    <ul id="persons"></ul>
    <a href="#" onclick="createItem();">Neu</a>
</body>
```