Mobile Web Apps entwickeln

Datenstrukturen in JavaScript

Alexander Miller
alles.mil@gmail.com

Lehrgang Junior Web Frontend-/Backend-Entwickler
WIFI Salzburg

Frühjahr 2021

Version 25. Februar 2021

Datenstruktur

Eine Datenstruktur ist ein Objekt zur Speicherung und Organisation von Daten. Die Daten werden auf bestimme Art und Weise angeordnet und verknüpft, um effiziente Verwaltung und Zugriffe zu ermöglichen.[1]

[1] https://de.wikipedia.org/wiki/Datenstruktur

Häufige Datenstrukturen

- Object
- Array
  - Queue
  - Stack
- Set
- Map

Object

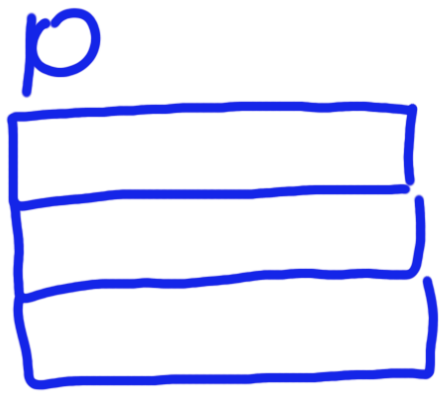
- Ein JavaScript Object ist eine Sammlung von Properties
- Eine Property ist ein Paar bestehend aus einem Namen (key) und einem Wert (value)
- Die Namen von Properties können Strings oder Integer sein
  - Zugriff auf String-Keys mit dem `.`-Operator
  - Zugriff auf Integer-Keys mit dem `[]`-Operator
- Wenn der Value einer Property eine Funktion ist, dann nennt man die Property auch "Methode"
- Fast alles in JavaScript ist ein Object

Object - Aufbau 1/8 - Initialisierung

```
let p = {};
```

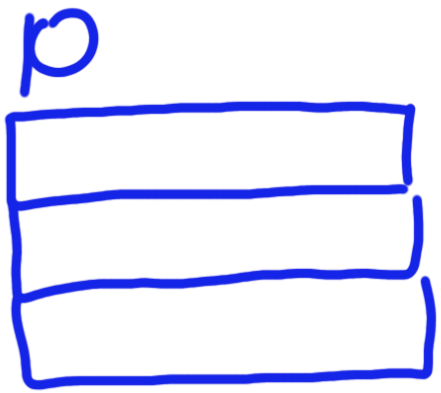
Object - Aufbau 2/8 - Initialisierung

```
let p = {};
```



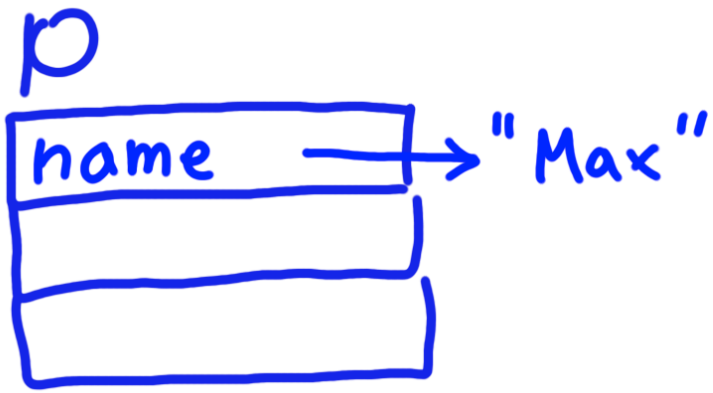
Object - Aufbau 3/8 - Property setzen

```
let p = {};
p.name="Max";
```



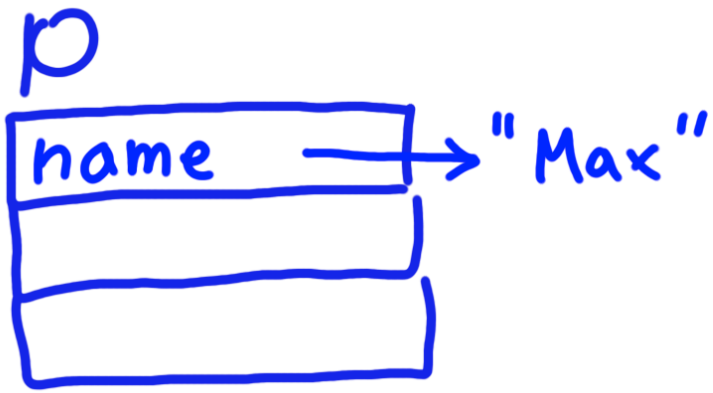
Object - Aufbau 4/8 - Property setzen

```
let p = {};
p.name="Max";
```



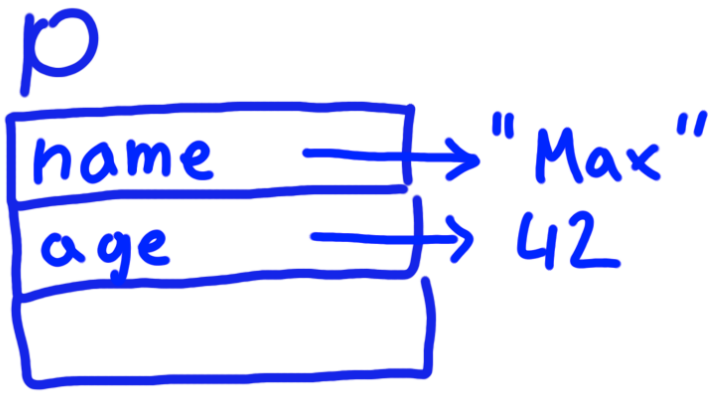
Object - Aufbau 5/8

```
let p = {};
p.name = "Max";
p.age = 42;
```



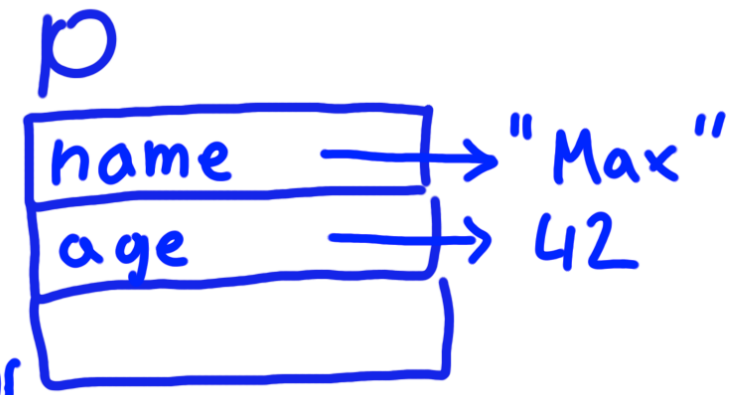
Object - Aufbau 6/8

```
let p = {};
p.name = "Max";
p.age = 42;
```



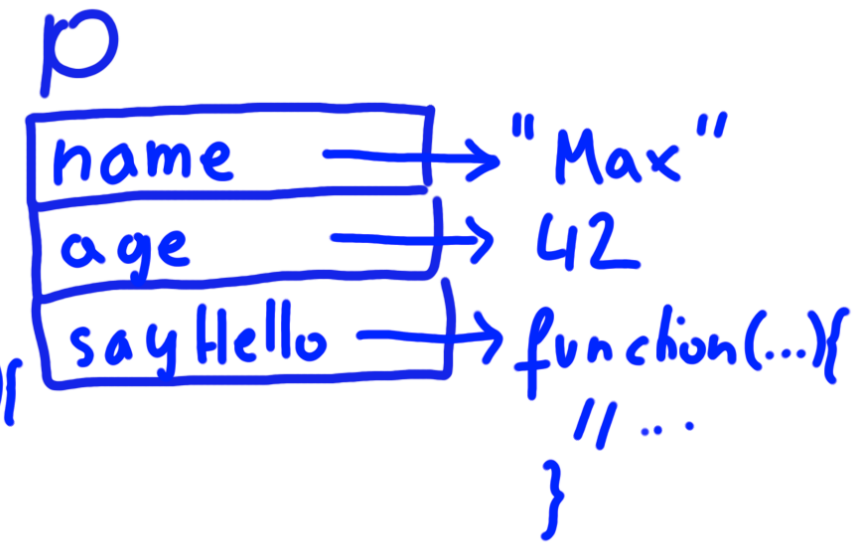
Object - Aufbau 7/8

```
let p = {};
p.name = "Max";
p.age = 42;
p.sayHello = function(other_name){
  //...
}
```



Object - Aufbau 8/8

```
let p = {};
p.name = "Max";
p.age = 42;
p.sayHello = function(other_name){
  //...
}
```



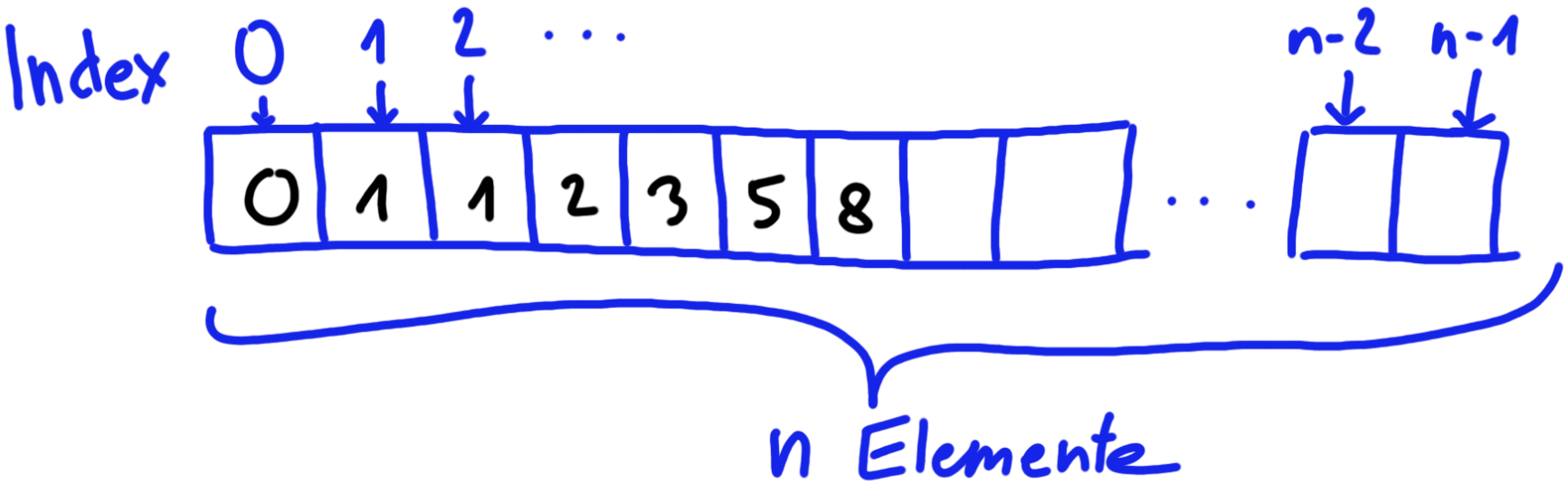
Object - Beispiele

```
let o = {}; // empty object

// Object mit den Properties name und age
o = { name: "Max Mustermann", age: 42 };

// Aufruf der Methode 'querySelector' vom Object 'document'
document.querySelector('body');
```

Array

- Eine Liste von 0 oder mehr Werten der Länge `length`.
- Elemente können über den Index, beginnend bei 0, angesprochen werden.
- Index `0` ist der Beginn der Liste (‘vorne’), Index `length - 1` ist das Ende der Liste (‘hinten’).

Array - Aufbau



Arrays - Erzeugung

Erzeugung mit einem *Array Literal*.

```
const list1 = []; // empty array
const list2 = [0, 1, 1, 2, 3, 5];
```

Arrays - Lesen

```
list1[0]; // Gibt das erste Element zurück
list1[list1.length - 1]; // Gibt das letzte Element zurück
```

Zugriff auf ein ungültiges Element gibt `undefined` zurück.

Arrays - Hinzufügen

```
list1.unshift('A'); // 'A' wird vorne angefügt
list1.push('B'); // 'B' wird hinten angefügt

// Füge vor dem Element an Index 1 den String 'C' ein:
list1.splice(1, 0, 'C');
```

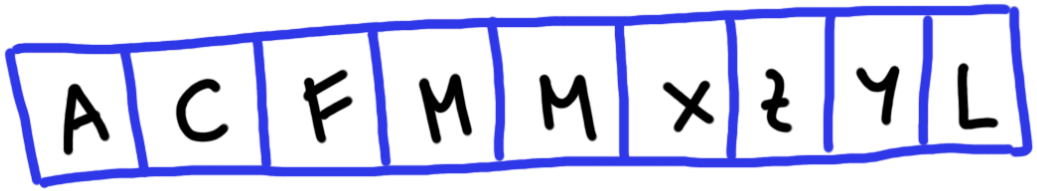
Arrays - Iteration

```
// Iteration über Index
for (let i = 0; i < list1.length; i++) {
  // doSomething(list1[i]);
}

// Iteration über Werte
for (let item of list1) {
  // doSomething(item);
}
```

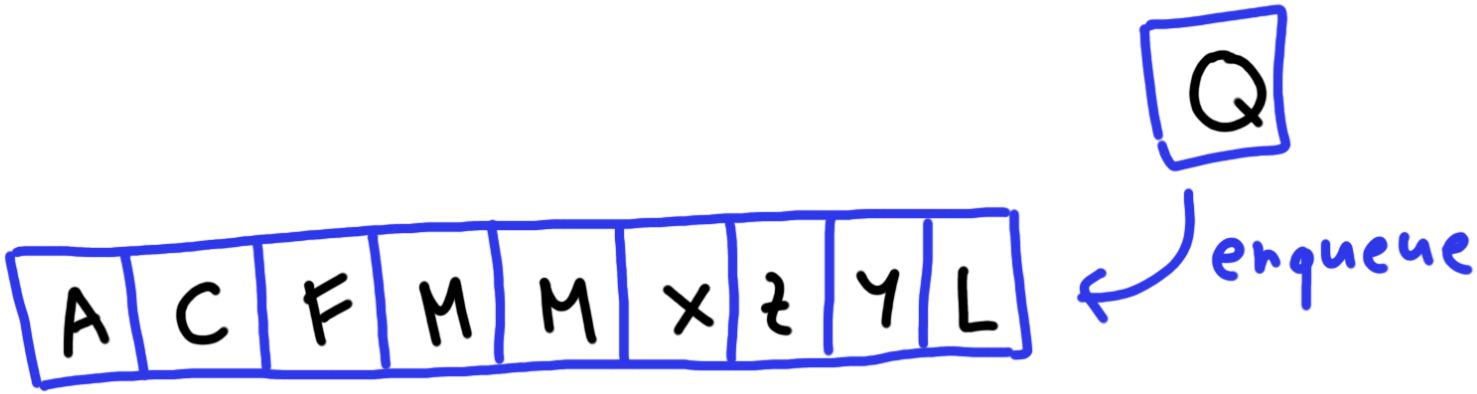
Queue

- "Warteschlange"
- FIFO: First In First Out
  - `enqueue(element)` stellt ein Element hinten an die Queue
  - `dequeue()` holt das vorderste Element von der Queue
- Hinzufügen nur an einem Ende, Herausnehmen nur am anderen Ende
- Keine eigene Klasse in JS, stattdessen kann ein Array mit den Methoden `push` und `shift` verwendet werden.

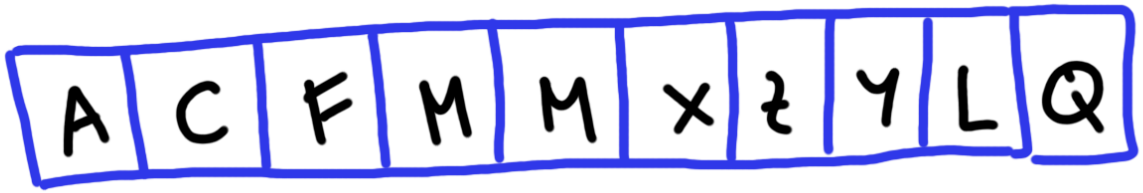
Queue - Beispiel 1/4



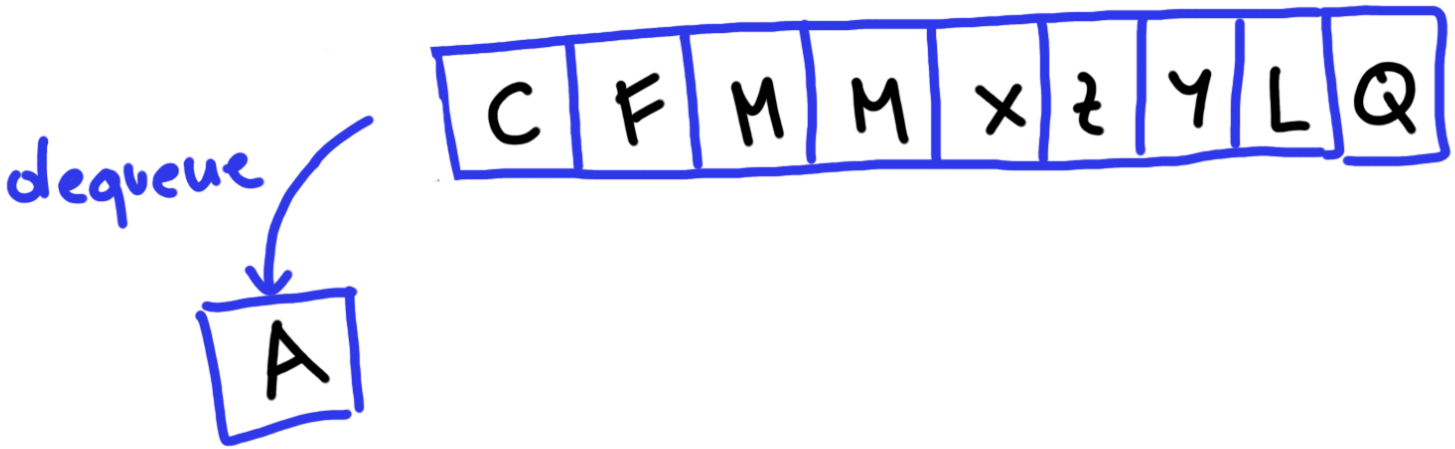
Queue - Beispiel 2/4 - enqueue



Queue - Beispiel 3/4



Queue - Beispiel 4/4 - dequeue



Übung - Queue

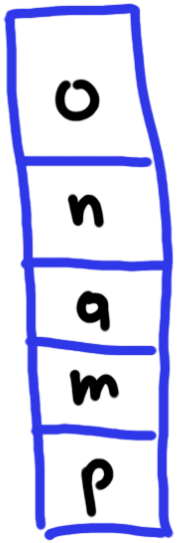
Erstellen Sie eine Web App, die eine Queue von Strings und ihre Operationen darstellt.

- Die User können beliebigen Text in die Queue einfügen (`enqueue`).
- Die User können Elemente aus Queue holen (`dequeue`).
- Geben Sie den Usern auch Feedback, falls Fehler auftreten.

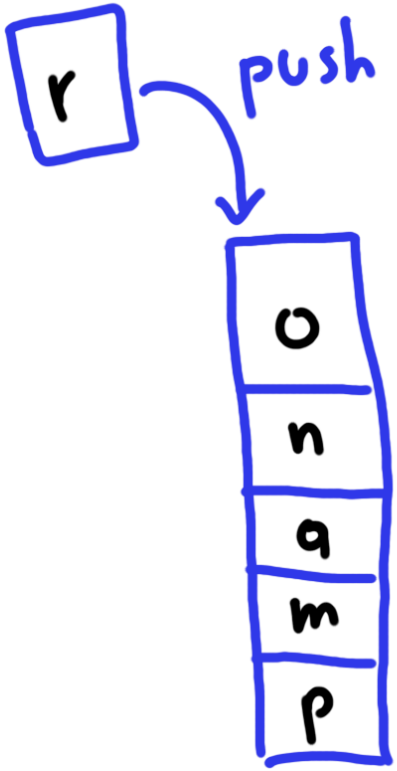
Stack

- "Stapel"
- LIFO: Last In First Out
  - `push(element)` legt ein Element auf den Stack
  - `pop()` nimmt ein Element vom Stack herunter
- Keine eigene Klasse in JS, stattdessen kann ein Array mit den Methoden `push` und `pop` verwendet werden.

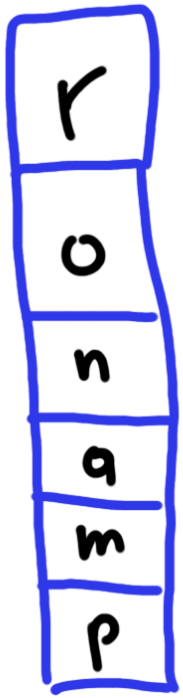
Stack - Beispiel 1/4



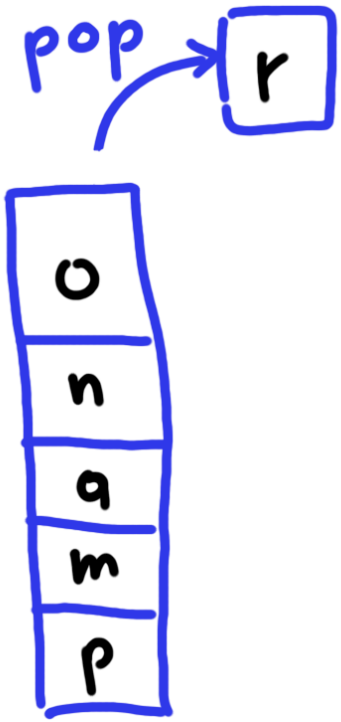
Stack - Beispiel 2/4 - push



Stack - Beispiel 3/4



Stack - Beispiel 4/4 - pop



Übung - Stack

Erstellen Sie eine Web App, die einen Stack von Strings und seine Operationen darstellt.

- Die User können beliebigen Text auf den Stack legen (`push`).
- Die User können das oberste Element vom Stack holen (`pop`).
- Geben Sie den Usern auch Feedback, falls Fehler auftreten.

Übung

Erstellen Sie eine Web App, die ein Array mit zufälligen ganzzahligen Werten und einstellbarer Länge erzeugen kann. Die User sollen eine untere und obere Schranke für die Werte und die Länge des Arrays angeben können.
Nach der Erstellung sollen die User nach einer Zahl suchen können und die Web App zeigt an, ob und wie oft diese Zahl im Array vorkommt.
Wie verhält sich die Laufzeit der Suchanfrage für Arrays der Größe 100, 1.000, 10.000, ...?

Set

- Menge von Werten ohne Reihenfole und ohne Duplikate
- Teil von ES6 (`https://caniuse.com/mdn-javascript_builtins_set`)
- Erzeugung: `let s = new Set();`
- Hinzufügen: `s.add('A');`
- Existenz eines Elements prüfen: `s.has('A');`

Map

- "Abbildung" von Keys zu Values
- Keys können von beliebigem Datentypen sein, anders als bei Objects
- Keine Duplikate bei den Keys
- Drei Methoden
  - `set(key, value)` um ein Key-Value-Paar in die Map zu schreiben
  - `get(key)` um für einen Key den Value aus der Map zu lesen
  - `has(key)` ob ein Key bereits in der Map existiert

Stack - Beispiel 1/8 - Initialisierung

```
let m = new Map();
```

Stack - Beispiel 2/8 - Initialisierung

```
let m = new Map();
```

Stack - Beispiel 3/8

```
let m = new Map();
m.set(303, {name:"Roland"});
```

Stack - Beispiel 4/8

```
let m = new Map();
m.set(303, {name:"Roland"});
```

Stack - Beispiel 5/8

```
let m = new Map();
m.set(303, {name:"Roland"});
m.set('A', 42);
```

Stack - Beispiel 6/8

```
let m = new Map();
m.set(303, {name:"Roland"});
m.set('A', 42);
```

Stack - Beispiel 7/8

```
let m = new Map();
m.set(303, {name:"Roland"});
m.set('A', 42);
```


```
m.get(303); ⟶ {name:"Roland"}
```

Stack - Beispiel 8/8

```
let m = new Map();
m.set(303,{name:"Roland"});
m.set('A', 42);
```


```
m.get(303); ⟶ {name:"Roland"}
m.get(42);  ⟶ undefined
```

Map - Iteration

```
for (let [key, value] of m) {
// console.log(key + ": " + value);
}
```

Übung

Verwenden Sie Set und Map, um die Suchanfrage von der vorhergehenden Übung zu beschleunigen.