JS Array-Methoden 2

Lernziele

- [ ] Verständnis fortgeschrittener Array-Methoden
  - [ ] `includes`
  - [ ] `find` und `findIndex`
  - [ ] `sort` und `reverse`
    - [ ] Wissen, wie `slice()` verwendet wird, um eine Kopie zu erstellen
  - [ ] `some` und `every`
  - [ ] `reduce`

`includes`

Verwende `array.includes()`, um zu überprüfen, ob das Array den angegebenen Wert enthält. Wenn ja, wird `true` zurückgegeben, andernfalls `false`.

```
const colors = ["hotpink", "aquamarine", "granite"];

colors.includes("aquamarine"); // true
colors.includes("nemo"); // false
```

`find` und `findIndex``

Verwende `find()`, um **das erste Element** des Arrays zu erhalten, das die bereitgestellte Testfunktion erfüllt. Andernfalls wird `undefined` zurückgegeben.

```
const colors = ["hotpink", "aquamarine", "granite", "grey"];

colors.find((color) => color.startsWith("g")); // 'granite'
colors.find((color) => color.startsWith("b")); // undefined
```

Verwende `findIndex()`, um den Index **des ersten Elements** des Arrays zu erhalten, das die bereitgestellte Testfunktion erfüllt. Wenn kein solches Element existiert, wird `-1` zurückgegeben.

```
const colors = ["hotpink", "aquamarine", "granite", "grey"];

colors.findIndex((color) => color.startsWith("g")); // 2
colors.findIndex((color) => color.startsWith("b")); // -1
```

`sort` und `reverse``

Verwende `sort()`, um die Elemente eines Arrays zu sortieren. Du musst eine Callback-Funktion bereitstellen, um anzugeben, wie das Array sortiert werden soll.

Zahlen sortieren

```
const numbers = [4, 42, 23, 1];

numbers.sort((a, b) => a - b); // [1, 4, 23, 42]
numbers.sort((a, b) => b - a); // [42, 23, 4, 1]
```

Die Sortierreihenfolge basiert auf dem Rückgabewert von `a - b` / `b - a`:

Rückgabewert von `a - b`	Sortierreihenfolge
> 0	sortiere `a` hinter `b`
< 0	sort `a` vor `b`
=== 0	behalte die ursprüngliche Reihenfolge von `a` und `b`

> 💡 `sort()` konvertiert die Elemente in Strings und vergleicht dann deren Sequenzen von UTF-16-Codeeinheiten. Daher ist `array.sort()` ohne Callback meist wenig nützlich.

Strings sortieren

Um Strings zu sortieren, musst du der `sort()`-Methode zwei Dinge innerhalb der Callback-Funktion mitteilen:

- beide Strings vor dem Vergleich in Kleinbuchstaben umwandeln (Großbuchstaben funktionieren auch)
- mittels if-Statements explizit die Rückgabewerte abhängig vom Ergebnis des Vergleichs festlegen (`nameA < nameB` und `nameA > nameB`)

```
const strings = ["Xbox", "PlayStation", "GameBoy"];

strings.sort((a, b) => {
  const nameA = a.toLowerCase();
  const nameB = b.toLowerCase();
  if (nameA < nameB) {
    return -1;
  }
  if (nameA > nameB) {
    return 1;
  }
  return 0;
});

console.log(strings); // ['GameBoy', 'PlayStation', 'Xbox']
```

> 💡 In UTF-16 haben die Groß- und Kleinschreibweise derselben Buchstaben nicht denselben Wert. Ein Großbuchstabe 'H' hat den UTF-16-Dezimalwert 72, während der Kleinbuchstabe 'h' einen Wert von 104 hat.

> Zum Beispiel werden ein Großbuchstabe 'W' (87) und ein Kleinbuchstabe 'd' (100) hinter dem Großbuchstaben 'H' (72), aber vor dem Kleinbuchstaben 'h' (104) sortiert; das Ergebnis könnte folgendermaßen aussehen: ['H', 'W', 'd', 'h']. Daher ist es notwendig, alle Buchstaben vor dem Sortieren in Groß- oder Kleinbuchstaben umzuwandeln.

`reverse`

Um ein Array umzukehren, verwende einfach `array.reverse()`. Dies kann auch mit `sort()` kombiniert werden:

```
const numbers = [4, 42, 23, 1];

const reversedNumbers = numbers.reverse(); // [1, 23, 42, 4]
```

`slice`

Es ist wichtig zu beachten, dass einige Array-Methoden, wie `sort()`, kein neues Array erstellen, sondern das Original-Array verändern.

```
const numbers = [4, 42, 23, 1];

console.log(numbers); // [4, 42, 23, 1]

const sortedNumbers = numbers.sort((a, b) => a - b);

console.log(sortedNumbers); // [1, 4, 23, 42]
console.log(numbers); // [1, 4, 23, 42]

// Was passiert, wenn sortedNumbers umgekehrt wird?

const reversedSortedNumbers = sortedNumbers.reverse();

console.log(reversedSortedNumbers); // [42, 23, 4, 1]
console.log(sortedNumbers); // [42, 23, 4, 1]
console.log(numbers); // [42, 23, 4, 1]
```

Dieses Verhalten führt häufig zu Fehlern. Um dies zu vermeiden, kannst du einfach eine Kopie des Original-Arrays mit `slice()` erstellen.

```
const numbers = [4, 42, 23, 1];
```

```
console.log(numbers); // [4, 42, 23, 1]

const sortedNumbers = numbers.slice().sort((a, b) => a - b);

console.log(sortedNumbers); // [1, 4, 23, 42]
console.log(numbers); // [4, 42, 23, 1]
```

`some` und `every`

Verwende `some()`, um zu testen, ob **mindestens ein Element** im Array den bereitgestellten Test besteht.

```
const colors = ["hotpink", "aquamarine", "granite"];

colors.some((color) => color.startsWith("g")); // true
colors.some((color) => color.startsWith("i")); // false
```

Um zu überprüfen, ob **alle Elemente** den Test bestehen, verwende `every()`.

```
const colors = ["hotpink", "aquamarine", "granite"];

colors.every((color) => color.length > 5); // true
colors.every((color) => color.length < 3); // false
```

`reduce`

`Array.reduce()` ist eine Array-Methode, um eine Liste von Werten in einen einzigen Wert zu reduzieren.

Es hat die folgenden Kernmerkmale:

- Es wird von Anfang an die Callback-Funktion auf jedes Element des Arrays ausgeführt.
- Der Rückgabewert jeder Berechnung wird an die nächste Berechnung weitergegeben (d.h. er wird zum neuen Startwert für die nächste Iteration durch das Array).
- Das Endergebnis ist ein einzelner Wert.

Die Hauptanwendung besteht darin, die Summe eines Arrays von Zahlen zu berechnen.

```
const numbers = [4, 42, 23, 1];

const sum = numbers.reduce((a, b) => a + b);

console.log(sum); // 70
```

> ! Wenn du feststellst, dass du mit reduce etwas Komplexeres machst (wie das Reduzieren eines Arrays auf ein Objekt usw.), solltest du versuchen, eine andere Lösung für dein Problem zu finden.

> Komplexe reduce-Funktionen sind sehr schwer zu lesen und daher fehleranfällig.

> Beispiel für das Reduzieren eines Arrays auf ein Objekt ohne `reduce()`:

```
const myArray = [
  { foo: 1, bar: "hi" },
  { foo: 4, bar: "hey" },
  { foo: 2, bar: "ho" },
];
const myObject = {};
myArray.forEach((element) => {
  myObject[element.bar] = element.foo;
});

console.log(myObject); // {hi: 1, hey: 4, ho: 2}
```

Resources

- Searching Arrays (javascript.info)
- sort (javascript.info)
- reduce (javascript.info)