1.2 - JSA: Referenzen und Objektkopien

Einleitung

In JavaScript ist es wichtig zu verstehen, wie **Daten übergeben, gespeichert und kopiert** werden – besonders bei Objekten. Während primitive Werte wie Zahlen oder Strings direkt kopiert werden, verhält es sich bei Objekten grundlegend anders: Hier wird nicht das Objekt selbst kopiert, sondern nur eine **Referenz** auf dessen Speicherort.

Primitive Werte vs. Referenzen

Primitive Werte (z. B. Number, String, Boolean)

```
let a = 5;
let b = a;

b = 10;
console.log(a); // 5
console.log(b); // 10
```

Ergebnis: Beide Variablen enthalten **eigene Kopien** des Werts.

Objekte – Referenzen

```
let obj1 = { name: "Anna" };
let obj2 = obj1;

obj2.name = "Tom";
console.log(obj1.name); // "Tom"
```

Ergebnis: `obj1` und `obj2` verweisen **auf dasselbe Objekt** im Speicher.

Objektkopien

1. Manuelles Kopieren (flach)

```
const original = { x: 1, y: 2 };
const copy = {};

for (let key in original) {
  copy[key] = original[key];
}
```

2. `Object.assign()` (flach)

```
const copy = Object.assign({}, original);
```

3. Spread Operator `{ ...obj }` (flach)

```
const copy = { ...original };
```

❗ Einschränkung bei flachen Kopien:

Bei **verschachtelten Objekten** bleibt die Referenz auf tieferer Ebene erhalten:

```
const original = {
  name: "Anna",
  address: { city: "Berlin" }
};

const copy = { ...original };
copy.address.city = "Hamburg";

console.log(original.address.city); // "Hamburg"
```

➡️ **Tiefer liegende Objekte werden nicht mitkopiert.**

Tiefe Kopien (Deep Cloning)

1. Mit JSON (funktioniert nur bei einfachen Objekten)

```
const deepCopy = JSON.parse(JSON.stringify(original));
```

Achtung:

- funktioniert **nicht** mit Funktionen, Symbolen, `undefined`, Date-Objekten etc.

2. `structuredClone()` – moderner Standard (ab ES2021)

```
const deepCopy = structuredClone(original);
```

- Unterstützt komplexe Strukturen, auch z. B. Arrays, Maps, Sets
- **Nicht in allen Browsern oder Node-Versionen verfügbar**

Typische Fehlerquellen

1. Objektkopien überschreiben sich gegenseitig

```
const a = { val: 1 };
const b = a;
b.val = 2;
console.log(a.val); // 2
```

2. Tiefe Objekte werden nur scheinbar kopiert

```
const original = { data: { value: 42 } };
const copy = { ...original };
copy.data.value = 99;
console.log(original.data.value); // 99
```

➡️ Nur die **oberste Ebene** wird kopiert.

Praxisbeispiele

```
const user = {
  name: "Lisa",
  contact: {
    email: "lisa@example.com"
  }
};

const shallowCopy = { ...user };
shallowCopy.contact.email = "neu@example.com";
console.log(user.contact.email); // "neu@example.com"
```

```
const deepUser = structuredClone(user);
deepUser.contact.email = "tief@example.com";
console.log(user.contact.email); // bleibt "neu@example.com"
```

Übungsaufgaben

1. Referenz vs. Wert

Erzeuge eine Zahl `a` und kopiere sie in `b`. Ändere `b` und prüfe, ob `a` sich verändert.

2. Objektreferenz verstehen

Erzeuge ein Objekt `obj1` und weise es einer zweiten Variablen `obj2` zu. Ändere `obj2` und prüfe `obj1`.

3. Flache Kopie mit Spread

Nutze den Spread Operator, um ein Objekt `person` zu kopieren. Ändere eine verschachtelte Eigenschaft und beobachte das Ergebnis.

4. Tiefe Kopie mit JSON

Kopiere ein Objekt `config` per `JSON.parse(JSON.stringify(...))` und prüfe, ob Änderungen am Kopieobjekt das Original beeinflussen.

5. Deep Copy mit `structuredClone`

Nutze `structuredClone()` für ein verschachteltes Objekt `settings` und vergleiche die Tiefe der Trennung.

Micro-Projekt – Referenzen und Objektkopien

Projekt: Konfigurationsvorlage

Du sollst ein Konfigurationsobjekt klonen und anpassen, ohne das Original zu verändern.

Anforderungen:

- Definiere ein Objekt `defaultConfig` mit verschachtelten Strukturen
- Erstelle mindestens zwei angepasste Varianten:
  - eine per Spread-Operator (flache Kopie)
  - eine per Deep Clone (`structuredClone` oder `JSON`)
- Wechsle bewusst zwischen flacher und tiefer Kopie und demonstriere den Unterschied

Bonus:

- Schreibe eine Vergleichsfunktion `compareConfigs(a, b)` die überprüft, ob beide denselben Inhalt haben (flach)