1.3 - JSA: Methoden und das Schlüsselwort `this`

Einleitung

- Wie Funktionen als Methoden definiert werden können
- Wie `this` in Methoden funktioniert und sich unterscheidet
- Wie Getter und Setter zum kontrollierten Zugriff auf Eigenschaften verwendet werden
- Welche Besonderheiten bei Kurzschreibweisen oder bei Arrow Functions zu beachten sind

---

Funktionen als Methoden

1. Normale Funktionszuweisung

```
const user = {
  name: "Lena",
  greet: function() {
    console.log("Hallo, ich bin " + this.name);
  }
};

user.greet(); // "Hallo, ich bin Lena"
```

Die Funktion `greet` ist hier eine Methode des Objekts `user`. Sie verwendet `this`, um auf andere Eigenschaften des Objekts zuzugreifen.

2. Kurzschreibweise für Methoden (ES6)

```
const user = {
  name: "Lena",
  greet() {
    console.log("Hallo, ich bin " + this.name);
  }
};
```

Dies ist funktional identisch mit dem vorherigen Beispiel, aber kürzer und moderner.

3. Methoden per Zuweisung nachträglich definieren

```
const user = {
  name: "Lena"
};

user.sayBye = function() {
  console.log("Tschüss von " + this.name);
};

user.sayBye(); // "Tschüss von Lena"
```

Das Keyword `this`

Was bedeutet `this`?

`this` ist ein Kontextverweis innerhalb von Funktionen. In Methoden bezieht es sich auf das Objekt, auf dem die Methode aufgerufen wird.

```
const car = {
  brand: "BMW",
  getBrand: function() {
    return this.brand;
  }
};

console.log(car.getBrand()); // "BMW"
```

Achtung bei Arrow Functions

Arrow Functions haben **kein eigenes** `this`. Sie erben den Kontext aus dem umgebenden Scope:

```
const user = {
  name: "Lena",
  greet: () => {
    console.log("Hallo, ich bin " + this.name);
  }
};

user.greet(); // "Hallo, ich bin undefined"
```

Warum? `this` zeigt hier nicht auf `user`, sondern auf das äußere Scope (z. B. `window` in Browsern).

➡️ Verwende **niemals Arrow Functions** für Methoden, bei denen du `this` brauchst.

Getter und Setter

Getter – Zugriff kontrollieren

Ein **Getter** ist eine spezielle Methode, die wie eine Eigenschaft verwendet wird:

```
const person = {
  firstName: "Tom",
  lastName: "Müller",
  get fullName() {
    return this.firstName + " " + this.lastName;
  }
};

console.log(person.fullName); // "Tom Müller"
```

Obwohl `fullName` eine Funktion ist, wird sie ohne Klammern aufgerufen.

Setter – Werte setzen mit Logik

Ein **Setter** definiert, was passiert, wenn eine Eigenschaft gesetzt wird:

```
const person = {
  firstName: "Tom",
  lastName: "Müller",
  set fullName(value) {
    const parts = value.split(" ");
    this.firstName = parts[0];
    this.lastName = parts[1];
  }
};
```

```
person.fullName = "Anna Schmidt";
console.log(person.firstName); // "Anna"
```

Getter/Setter ohne `get`/`set` – als Methoden

Alternativ kannst du auch ganz normale Methoden verwenden, die wie Getter oder Setter funktionieren:

```
const account = {
  balance: 0,
  getBalance: function() {
    return this.balance;
  },
  setBalance: function(amount) {
    if (amount >= 0) this.balance = amount;
  }
};

account.setBalance(100);
console.log(account.getBalance()); // 100
```

Diese Methode ist etwas weniger elegant, aber expliziter.

Praxisbeispiele

```
const dog = {
  name: "Bello",
  speak() {
    return this.name + " bellt!";
  }
};

console.log(dog.speak());
```

```
const rectangle = {
  width: 10,
  height: 4,
  get area() {
    return this.width * this.height;
  }
};

console.log(rectangle.area); // 40
```

Übungsaufgaben

1. Methode erstellen

Erstelle ein Objekt `cat` mit der Eigenschaft `name` und einer Methode `meow`, die den Namen verwendet.

2. Unterschiedliche Methodenformen

Erzeuge ein Objekt `calculator` mit Methoden in allen drei Varianten: klassische Funktionszuweisung, Kurzschreibweise und nachträgliche Definition.

3. Fehler mit Arrow Function

Erzeuge ein Objekt `user` mit einer Arrow Function als Methode und beobachte das Verhalten von `this`.

4. Getter und Setter einsetzen

Erstelle ein Objekt `temperature` mit der Eigenschaft `celsius`. Verwende einen Getter `fahrenheit` (Umrechnung) und einen Setter `fahrenheit`, der zurückrechnet.

5. Getter/Setter als Methoden

Baue ein Objekt `bankAccount` mit Methoden `getBalance()` und `setBalance()`, die mit `this` auf ein internes Attribut zugreifen.

Micro-Projekt – Methoden in Objekten

Projekt: Geräte-Fernbedienung

Erstelle ein Objekt `remoteControl`, das verschiedene elektronische Geräte simuliert und über Methoden gesteuert werden kann.

Anforderungen:

- Definiere mindestens drei Methoden (`powerOn`, `powerOff`, `changeChannel`) direkt im Objekt
- Verwende `this`, um Gerätezustände innerhalb der Methoden zu ändern oder abzufragen
- Ergänze einen Getter `status`, der den aktuellen Zustand als String zurückliefert

Bonus:

- Verwende zusätzlich die ES6-Methodensyntax für mindestens eine Methode
- Simuliere eine zweite Fernbedienung mit ähnlichen Funktionen, aber anderem Objektkontext