Übung 05

1 Prototyping und Typescript

Erzeugen Sie den Prototypen für einen `Gegenstand` mit den Eigenschaften `bezeichner` und `gewicht` sowie den Funktionen `getBeschreibung` und `getGewicht`.

Vererben Sie die Eigenschaften und Funktionen von `Gegenstand` an einen Prototypen `Kugel` und ergänzen Sie die Eigenschaft `radius`. Außerdem soll die Funktion `getBeschreibung` so überschrieben werden, dass neben dem Bezeichner auch die Oberfläche der Kugel ausgegeben wird.

Vererben Sie die Eigenschaften und Funktionen von `Gegenstand` zusätzlich an einen Prototypen `Quader` und überschreiben Sie auch hier die Funktion `getBeschreibung` so, dass die Oberfläche des Quaders ebenfalls ausgegeben wird.

Erstellen Sie nun verschiedene Instanzen und rufen Sie für diese jeweils `getBeschreibung` und `getGewicht` auf.

Lösungsweg

Beginnen wir mit der Erstellung des ersten Prototypen `Gegenstand`. Hierfür erstellen wir ein Objekt `Gegenstand` und füllen dieses mit den Eigenschaften `bezeichner` und `gewicht`. Anschließend implementieren wir die Funktionen `getBeschreibung` und `getGewicht`, diese Funktionen sollen uns Zugriff auf die Eigenschaften des Objekts `Gegenstand` geben.

```
var Gegenstand = {
    bezeichner: null,
    gewicht: null,
    getBeschreibung: function() {
        console.log(this.bezeichner);
    },
    getGewicht: function() {
```

```
        console.log(this.gewicht, "g");
    }
}
```

Mit diesem Schritt haben wir nun unseren ersten Prototyp erstellt. Nun wollen wir einen neuen Prototyp `Kugel` erstellen. Dieser soll alle Eigenschaften und Methoden besitzen die der Prototyp `Gegenstand` besitzt und zusätzlich die Eigenschaft `radius` enthalten. Weiterhin soll die Funktion `getBeschreibung` überschrieben werden. Hierfür erstellen wir ein Objekt `Kugel` das mit `Object.create()` aus dem Prototyp `Gegenstand` erstellt wird und fügen diesem die neuen Eigenschaften und Funktionen hinzu:

```
var Kugel = Object.create(Gegenstand);
Kugel.radius = null;
Kugel.getBeschreibung = function() {
    console.log(this.bezeichner);
    console.log("Oberfläche: " + 4 * Math.PI * Math.pow(this.radius, 2));
};
```

Analog erstellen wir so auch den Prototyp `Wuerfel`:

```
var Wuerfel = Object.create(Gegenstand);
Wuerfel.seitenlaenge = null;
Wuerfel.getBeschreibung = function() {
    console.log(this.bezeichner);
    console.log("Oberfläche: " + 6 * this.seitenlaenge * this.seitenlaenge);
};
```

Testen wir nun unsere Prototypen, indem wir jeweils ein Objekt aus dem Prototyp `Kugel` und `Wuerfel` erstellen. Auch hier nutzen wir die Funktion `Object.create()` und übergeben zusätzlich zu dem Prototyp die Werte der Eigenschaften:

```
var eineKugel = Object.create(Kugel, {
    bezeichner: { value: "eine Kugel" },
    gewicht: { value: 4 },
    radius: { value: 2 }
});
```

Analog erstellen wir ebenfalls ein Objekt des Prototypen `Wuerfel`:

```
var einWuerfel = Object.create(Wuerfel, {
    bezeichner: { value: "ein Würfel" },
    gewicht: { value: 5 },
    seitenlaenge: { value: 2 }
});
```

Zuletzt lassen wir uns die Eigenschaften beider Objekte noch über unsere Funktionen ausgeben:

```
eineKugel.getBeschreibung();
eineKugel.getGewicht();
```

```
einWuerfel.getBeschreibung();
einWuerfel.getGewicht();
```

2 Aufgaben

Bearbeiten Sie nun die folgenden Aufgaben.

Aufgabe 1

Setzen Sie nun dieselbe Logik mit ES6 Klassen und Vererbung um. Definieren Sie dazu Konstruktorfunktionen für Gegenstand, Kugel und Wuerfel. Überschreiben Sie zusätzlich die Funktion `getBeschreibung` innerhalb der Klassen Kugel und Wuerfel.

Aufgabe 2

Erstellen Sie mithilfe des beigelegten `speiseplan.html`-Dokumentes eine Webseite, die es ermöglicht, Speisen einem Speiseplan hinzuzufügen und zu entfernen.

Hierbei sollen die Speisen, die bereits auf dem Speiseplan stehen, auf der Webseite angezeigt werden. Wenn eine Speise dem Speiseplan hinzugefügt wird, soll der Speiseplan auf der Webseite aktualisiert werden, dies gilt analog auch für das Entfernen einer Speise. Zusätzlich darf jede Speise nur einmal auf dem Speiseplan vorkommen, bei dem hinzufügen eines Duplikates soll eine Warnung an den User ausgegeben werden. Benutzen Sie für die Umsetzung des Speiseplans statische Methoden.

Aufgabe 3

Erstellen Sie mit Typescript eine Klasse Kunde, die das Attribute Name enthält. Implementieren Sie weiterhin eine Klasse Bank mit den Attributen Name, Bankleitzahl und Kunden. Hierbei soll das Attribut Kunden ein Array aus Objekten der Klasse Kunde sein und leer initialisiert werden. Um nachträglich Kunden einer Bank hinzuzufügen, implementieren Sie eine Funktion `addKunde`, die einen Kunden als Parameter erhält und diesen dem Kundenarray der Klasse Bank hinzufügt. Implementieren Sie ebenfalls für alle Attribute Funktionen, mit denen Sie auf die Attribute der Klassen zugreifen können (getter-Methoden). Achten Sie weiterhin darauf, das Ihre Attribute sinnvoll typisiert sind.

Zum Testen ihrer Klassen erstellen Sie drei Kunden-Objekte und ein Bank-Objekt. Fügen Sie die Kunden der Bank hinzu und lassen Sie sich alle Kundennamen der Bank in der Konsole ausgeben. (Hinweiß: Vergessen Sie nicht ihre Typescript Datei vorher zu einer JavaScript Datei zu transpilieren)

Lösungs zu den Aufgabes

Lösung 1

Listing 1

```
class Gegenstand {
  constructor(bezeichner, gewicht){
    this._bezeichner = bezeichner;
    this._gewicht = gewicht;
  }

  getBeschreibung(){
    console.log(this._bezeichner);
  }

  getGewicht(){
    console.log(this._gewicht, "g");
  }
}

class Kugel extends Gegenstand{
  constructor(bezeichner, gewicht, radius){
    super(bezeichner, gewicht);
    this._radius = radius;
  }

  getBeschreibung(){
    console.log(this._bezeichner);
      console.log("Oberfläche: " + 4 * Math.PI * Math.pow(this._radius, 2));
  }
}

class Wuerfel extends Gegenstand{
  constructor(bezeichner, gewicht, seitenlaenge){
    super(bezeichner, gewicht);
    this._seitenlaenge = seitenlaenge;
  }

  getBeschreibung(){
    console.log(this._bezeichner);
      console.log("Oberfläche: " + 6 * this._seitenlaenge * this._seitenlaenge);
  }

}

var eineKugel = new Kugel("Eine Kugel", 4, 2);
var einWuerfel = new Wuerfel("Ein Wuerfel", 5, 2);

eineKugel.getBeschreibung();
eineKugel.getGewicht();
einWuerfel.getBeschreibung();
einWuerfel.getGewicht();
```

Lösung 2

Listing 2

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Speiseplan</title>
    </head>
    <body>
      <h3>Speiseplan</h3>
      <table>
        <tr>
          <th><p>Geben Sie eine Speise ein</p></th>
        </tr>
        <tr>
          <td><input type="text" id="speiseEingabe"></td>
          <td><input type="button" value="Hinzufügen" id="hinzufuegen"></td>
                <td><input type="button" value="Entfernen" id="entfernen"></td>
        </tr>
      </table>
        <p id="Speiseplan"></p>
        <script src="speiseplan.js"></script>
    </body>
</html>
```

Listing 3

```
class Speiseplan{

  static menu = [];

  static addItem (item){
    if(Speiseplan.isInMenu(item)){
      alert("Speise ist bereits im Menü");
    }
    else{
      Speiseplan.menu.push(item);
    }

  }

  static removeItem (item){
    if(Speiseplan.isInMenu(item)){
      Speiseplan.menu.splice(Speiseplan.menu.indexOf(item),1);
    }
    else{
      alert("Speise ist nicht im Menü");
    }

  }

  static isInMenu(item){
    return Speiseplan.menu.includes(item);
  }
  static printMenu(){
    var speiseListe = document.querySelector('#Speiseplan');
    speiseListe.textContent = '';
    for(let item in Speiseplan.menu) {
      speiseListe.textContent += Speiseplan.menu[item] + ', ';
    }
```

```
  }

}

var speiseEingabe = document.querySelector('#speiseEingabe');
var hinzufuegen = document.querySelector('#hinzufuegen');
var entfernen = document.querySelector('#entfernen');

hinzufuegen.addEventListener('click', function() {
  let item = speiseEingabe.value;
  Speiseplan.addItem(item);
  Speiseplan.printMenu();
});

entfernen.addEventListener('click', function() {
  let item = speiseEingabe.value;
  Speiseplan.removeItem(item);
  Speiseplan.printMenu();
});
```

Lösung 3

Listing 4

```
class Kunde {
  name: string;

  constructor(name: string){
    this.name= name;
  }

  getName(){
    return this.name;
  }
}

class Bank  {
  name: string;
  blz: number;
  kunden: Kunde[];

  constructor(name: string, blz: number){
    this.name = name;
    this.blz = blz;
    this.kunden = [];
  }

  getName(){
    return this.name;
  }

  getBlz(){
    return this.blz
  }

  getKunden(){
    return this.kunden;
  }
```

```
35
    addKunde(kunde: Kunde){
      this.kunden.push(kunde);
    }

40  }

    var alice = new Kunde("Alice");
    var bob = new Kunde("Bob");
    var charlie = new Kunde("Charlie");
45
    var sparkasse = new Bank("Sparkasse", 43000);

    sparkasse.addKunde(alice);
    sparkasse.addKunde(bob);
50  sparkasse.addKunde(charlie);

    var sparkassenKunden = sparkasse.getKunden();
    for(let kunde in sparkassenKunden){
      console.log(sparkassenKunden[kunde].getName());
55  }
```