Klassen in JavaScript

Aufgabe 1: Klassen und Objekterstellung

Erstelle eine Klasse `Car`, die einen `constructor` hat, der die Eigenschaften `brand`, `model`, und `year` setzt. Implementiere eine Methode `displayDetails`, die die Details des Autos anzeigt. Erstelle drei verschiedene Auto-Objekte und rufe die Methode für jedes Objekt auf.

Aufgabe 2: Getter und Setter mit Berechnungen

Erstelle eine Klasse `Circle`, die den `radius` im Konstruktor annimmt. Implementiere Getter für den Durchmesser und die Fläche des Kreises. Der Durchmesser berechnet sich als `2 * radius` und die Fläche als `Math.PI * radius^2`. Implementiere auch einen Setter, um den Radius basierend auf dem Durchmesser zu ändern.

Aufgabe 3: Vererbung und Methodenüberschattung

Erstelle eine Klasse `Employee` mit den Eigenschaften `name` und `salary`. Füge eine Methode `work` hinzu, die eine allgemeine Nachricht ausgibt. Erstelle eine Unterklasse `Manager`, die eine Methode `manage` hinzufügt und `work` überschreibt, um spezifische Informationen auszugeben.

Aufgabe 4: Statische Eigenschaften und Methoden

Erstelle eine Klasse `BankAccount`, die eine statische Eigenschaft `totalAccounts` hat, welche zählt, wie viele Instanzen der Klasse erstellt wurden. Implementiere eine statische Methode `getTotalAccounts`, die diese Anzahl zurückgibt.

Aufgabe 5: Klassen mit Getter, Setter und Vererbung

Erstelle eine Klasse `Rectangle` mit den Eigenschaften `width` und `height`. Implementiere Getter für die Fläche und den Umfang des Rechtecks. Erstelle dann eine Unterklasse `Square`, die nur eine Eigenschaft `sideLength` hat und die Getter-Methoden erbt.

Aufgabe 6: Klassen als Erstklassige Bürger

Erstelle eine Klasse `Book` mit den Eigenschaften `title` und `author`. Speichere die Klasse in einer Variablen und erstelle ein Objekt dieser Klasse. Füge eine Methode `getInfo` hinzu, die den Titel und den Autor des Buches zurückgibt.

Aufgabe 7: Klassen und Konstruktorfunktionen vergleichen

Erstelle eine Klasse `Person` mit den Eigenschaften `name` und `age`. Konvertiere diese Klasse in eine Konstruktorfunktion und vergleiche die beiden Ansätze, indem du je ein Objekt von beiden erstellst und die Methoden aufrufst.

Aufgabe 8: Klassen und Statische Methoden kombinieren

Erstelle eine Klasse `MathHelper` mit einer statischen Methode `multiply`, die zwei Zahlen multipliziert. Verwende diese Methode, um das Quadrat und das Kubik einer Zahl zu berechnen, ohne eine Instanz der Klasse zu erstellen.

Aufgabe 9: Vererbung mit Super

Erstelle eine Basisklasse `Vehicle` mit einer Methode `move`, die `"The vehicle moves"` ausgibt. Erstelle eine Unterklasse `Airplane`, die die Methode `move` überschreibt, um `"The airplane flies"` auszugeben, aber auch die Methode der Basisklasse aufruft.

Aufgabe 10: Objektinstanzierung mit Bedingung

Erstelle eine Klasse `User` mit den Eigenschaften `username` und `password`. Implementiere eine Methode `login`, die überprüft, ob das Passwort die Mindestlänge von 8 Zeichen hat. Wenn nicht, soll eine Fehlermeldung ausgegeben werden.

Aufgabe 11: Getter und Setter für Validierung

Erstelle eine Klasse `Account` mit einer versteckten Eigenschaft `balance`. Verwende Getter und Setter, um sicherzustellen, dass der Kontostand nicht negativ werden kann, und implementiere Methoden zum Einzahlen und Abheben.

Aufgabe 12: Vererbung und Methodenschatten

Erstelle eine Klasse `Device` mit einer Methode `turnOn`. Erstelle eine Unterklasse `Smartphone`, die die Methode `turnOn` überschattet, um eine spezifische Nachricht für das Smartphone auszugeben, aber auch die Methode der Basisklasse aufruft.

Aufgabe 13: Abstrakte Klasse simulieren

Erstelle eine Klasse `Shape` mit einer Methode `getArea`, die `null` zurückgibt. Erstelle zwei Unterklassen `Circle` und `Square`, die jeweils die Methode `getArea` überschreiben, um die Fläche basierend auf ihrem jeweiligen Typ zu berechnen.

Aufgabe 14: Statik und Instanzmethoden mischen

Erstelle eine Klasse `Logger`, die eine statische Methode `log` und eine Instanzmethode `warn` enthält. Verwende die statische Methode, um allgemeine Logs zu speichern, und die Instanzmethode, um spezifische Warnungen für Objekte zu speichern.

Aufgabe 15: Komplexe Vererbungshierarchie

Erstelle eine Basisklasse `Animal` mit einer Methode `speak`. Erstelle dann zwei Unterklassen `Dog` und `Cat`, die jeweils die Methode `speak` überschreiben. Füge eine weitere Unterklasse `Puppy` hinzu, die von `Dog` erbt und `speak` erneut überschattet.