Web-Entwicklung: Übungsblatt 3

Aufgabe 1: Konstruktorfunktionen

Erstellen Sie eine Konstruktorfunktion `Container` und implementieren Sie den dazugehörigen Prototypen. Ein Container-Objekt aggregiert verschiedene andere Objekte und verwaltet diese intern in einem Array.

Anforderungen an Ihre Lösung sind:

- Bei der Erstellung eines Container-Objekts kann der initiale Inhalt optional angegeben werden.
- Ein neues Element kann mit der Methode `push(element)` hinzugefügt werden.
- Die Methode `getAll()` liefert den Inhalt des Containers als Array zurück. Achten Sie darauf, keine Referenz auf die interne Datenstruktur auszuliefern, sondern eine (flache) Kopie.
- Eine Methode `getInstances()` liefert ein Array mit allen bisher erstellten Container-Objekten.

Stellen Sie durch entsprechende Testanweisungen fest, ob sich Ihre Implementierung an diese Spezifikation hält.

Aufgabe 2: Revealing Module-Entwurfsmuster

In der Vorlesung haben Sie das Entwurfsmuster *Revealing Module* zur Kapselung von Prototypen mittels sog. IIFEs (*Immediate Invoked Function Expressions*) kennengelernt.

Kapseln Sie Ihre in Aufgabe 1 erstellte Konstruktorfunktion gemäß dieses Entwurfsmusters.

Verwenden Sie den gleichen Code wie in Aufgabe 1, um die Korrektheit Ihrer Implementierung zu testen.

Aufgabe 3: CommonJS-Modul

In der Vorlesung haben Sie die Konventionen für Module gemäß der CommonJS-Spezifikation kennengelernt.

Kapseln Sie Ihre in Aufgabe 1 erstellte Konstruktorfunktion in einem CommonJS-Modul. Verteilen Sie so das Modul und die Testanweisungen auf zwei separate Dateien.

Aufgabe 4: Vererbung 1 (TypedContainer)

Realisieren Sie einen spezialisierten `TypedContainer`, der nur Objekte mit einem bestimmten Prototyp aufnimmt. Der Prototyp soll als Konstruktorargument angegeben werden können. Wird versucht ein Objekt einzufügen, das nicht diesen Prototyp aufweist, soll ein Fehler geworfen werden.

Fügen Sie die Konstruktorfunktion dem CommonJS-Module aus Aufgabe 3 hinzu und ergänzen Sie die Testanweisungen in der separaten Datei.

Hinweis: Zum Loggen von Fehlern sollte die Methode `console.error()` verwendet werden, da diese unabhängig von normalen Log-Ausgaben in eine separate Datei weitergeleitet werden kann.

Aufgabe 5: ES6-Klassensyntax

In der Vorlesung haben Sie die neue Syntax zur Erzeugung von Prototypen ("ES6-Klassen") kennengelernt.

Wenn Sie die neue Klassensyntax auf das in Aufgabe 3 und 4 erstellte CommonJS-Modul an.

Verwenden Sie den gleichen Code wie in Aufgabe 3 und 4, um die Korrektheit Ihrer Implementierung zu testen.

Hinweis: Gerne dürfen Sie hier auch die übrigen in der Vorlesung kennengelernten ES6-Sprachkonstrukte einsetzen, z.B. die Schlüsselworter `let` und `const`, Template Strings, Arrow Functions oder die Objekt-Desktrukturierung.

Bonus-Aufgabe 6: Vererbung 2 (SecureContainer)

Serverseitig können sensible Daten vorliegen, die bei einer Anfrage nicht mitgesendet werden sollen. Realisieren Sie einen weiteren spezialisierten Container `SecureContainer` als ES6-Klasse im CommonJS-Modul, dem Sie ein Array von Funktionen als Konstruktorargument übergeben. Diese Funktionen soll beim Aufruf von `getAll()` auf die intern verwalteten Elemente angewendet werden, um so sensible Daten in der Rückgabe zu filtern. Die intern verwalteten Daten sollen dabei unverändert bleiben.

In der Datei `dataset.json` finden Sie einen Datensatz, der sensible Nutzerdaten (nämlich die E-Mail-Adressen) enthält. Lassen Sie diesen Datensatz von einem SecureContainer-Objekt verwalten und sorgen Sie dafür, dass bei Aufruf von `getAll()`

1. die Eigenschaft `email_is_public` aus jedem Element entfernt wird und
2. die Eigenschaft `email` aus allen Elementen entfernt wird, bei denen die Eigenschaft `email_is_public` den Wert `false` hat.