

Aufgabe 1 [5] Konstruktoren erstellen

Implementiere die Shape-Hierarchie (Shape, Line, Rectangle, Circle) in JavaScript. Gib dabei auch Methoden `move`, `doubleSize` an.

Aufgabe 2 [4] Konstruktor und Prototype zusammenfassen

In der Vorlesung wurde angesprochen, dass sich das Definieren des Konstruktors und das Setzen der `prototype`-Property für den Konstruktor in einer Funktion zusammenfassen lassen.

Realisiere diesen Vorschlag für die Konstruktoren `NamedPoint` und `ManhattanPoint` aus der Vorlesung. Gib also Generatoren für diese Konstruktoren an, die als Argument das Prototyp-Objekt erhalten und Funktionen zurückliefern, welche sich so verhalten wie die ursprünglichen Konstruktoren.

Aufgabe 3 [8] Methoden-Update

(a) [2] Definiere einen JavaScript-Konstruktor `Cell`, der Objekte erzeugt, die einen beliebigen Wert verwalten. Die Objekte sollen Methoden `get` und `set` bereitstellen, um den Wert zu erhalten und zu setzen.

(b) [4] Definiere einen JavaScript-Konstruktor `ReCell`, der ein `Cell`-Objekt als Prototyp enthält. `ReCell` soll dabei `set` neu definieren, so dass `set` die Property `restore` auf eine Funktion setzt, die den aktuellen Wert, den `set` überschreibt, wiederherstellt.

(c) [2] Gib eine Variante an, in der die Methode `restore` neben dem Wert auch die vorherige Version von `restore` wiederherstellen und damit ein unendliches Backup realisieren.

Aufgabe 4 [8] Explizite Vererbung

Für die explizite Vererbung ist es notwendig, nur einige der Properties vom Vater-Objekt zu übernehmen. Da JavaScript nur einen Prototype für das ganze Objekt definiert, ist explizite Vererbung in JavaScript nicht möglich. Verwendet man jedoch in JavaScript Embedding (so wie in der Vorlesung mit der Funktion `cloneObj`), so lässt sich explizite Vererbung simulieren.

Gib eine Funktion `delegate` an, die als Argument ein Vaterobjekt und eine beliebige Anzahl von Property-Namen erhält und die Properties mit diesen Namen vom Vater-Objekt in das `this`-Objekt kopiert.

Teste `delegate` indem du ein benanntes `Cell`-Objekt erstellst: Füge dazu `delegate` einem leeren Objekt hinzu und übernimm dann aus einem `NamedPoint`-Objekt (siehe Vorlesung) die für die Benennung notwendigen Properties und aus einem `Cell`-Objekt die Verwaltung eines Wertes.

Hinweis: Verwende das `arguments`-Array, um eine beliebige Anzahl von Parametern zu realisieren. Arrays in JavaScript sind natürlich auch nur Objekte. Der JavaScript-Standard definiert einige nützliche Funktionen auf Arrays.

Aufgabe 5 [5] Mode-Switching

Sprachen mit Embedding realisieren Mode-Switching, indem die Objekte ihre Methoden mit den Methoden aus einem anderen Objekt überschreiben. Der Zustand des Objekts soll dabei natürlich erhalten bleiben.

Gib eine Funktion `mode_switch` an, die alle Methoden eines Objekts in ein anderes Objekt kopiert.

Gib Testfälle für Mode-Switching mit `mode_switch` an.

Hinweis: Nutze dabei aus, dass alle Funktionen als Prototype das Objekt `Function.prototype` besitzen.
