WBE-Praktikum 3

Objekte und Arrays

Aufgabe 1: Node REPL

Zunächst wieder ein paar Versuche auf der Node REPL:

```
> let student1 = { name: "Bob", age: 21, grades: [4.5, 5.0, 4.5, 5.5] }
> let student2 = Object.assign({}, student1)
> let student3 = student1
> let student4 = {...student1}
> [student1===student2, student1===student3, student1===student4]
> [typeof student1, typeof student1.grades, Array.isArray(student1.grades)]
> student1.age = 25
> [student1, student2, student3, student4].map(s => s.age)
> student1.grades[3] = 6.0
> [student1, student2, student3, student4].map(s => s.grades)
> student4.grades[6] = 3.0
> student4.grades.length

> const birthday = (stud) => ({...stud, age: stud.age+1})
> birthday(student3)
> student3 = birthday(student3)
```

In den Slides hat es noch einige Anregungen, was Sie noch ausprobieren könnten.

Noch ein paar Überlegungen zu den obigen Anweisungen:

a. Wie sind die Ergebnisse der Vergleiche der verschiedenen *student*-Objekte (fünfte Zeile) zu erklären? Beachten Sie, dass {…} sowohl in *student2* als auch in *student4* neue Objekte erzeugen.

b. Welche Anforderungen müssen vom Argument der *birthday*-Funktion erfüllt werden, damit die Funktion wie erwartet arbeitet (nehmen wir an, dass die Erwartungen auch ohne weitere Spezifikation der Funktion ähnlich sind)?

   Noch ein Hinweis zur *birthday*-Funktion: die Funktion verändert das Argument nicht, sie erzeugt ein neues Objekt. Daher muss das Ergebnis zugewiesen werden.

Aufgabe 2: Objekte vergleichen (Abgabe)

Der Operator === überprüft auf Gleichheit. Bei Objekten werden aber die Referenzen verglichen. Schreiben Sie eine Funktion *equal*, welche *true* liefert, wenn die beiden Argumente === sind, aber auch, wenn es sich um zwei Objekte gleichen Inhalts handelt. Die Attributwerte der Objekte sollen dabei selbst mit === verglichen werden, das heisst wir berücksichtigen nur die oberste Ebene der Objekte und vergleichen keine verschachtelten Strukturen

Hier ein paar Beispiele:

```
> equal(16, 16)
true

> equal("hi", "hi")
true

> equal({}, {})
true

> equal({a:1, b:2}, {b:2, a:1})
true

> equal({a:1, b:2}, {c:3, b:2, a:1})
false

> equal({a:{}}, {a:{}})
false

> let emptyObj = {}
undefined

> equal({a:emptyObj}, {a:emptyObj})
true
```

Hinweise:

Überprüfen Sie zunächst, ob die beiden Argumente === sind. Falls das nicht der Fall ist, überprüfen Sie, ob beides Objekte sind (*typeof*-Operator). Wenn ja können Sie diese vergleichen.

Object.keys(obj) liefert ein Array der Attributnamen von *obj*. Mit der *includes*-Methode von Arrays können Sie überprüfen, ob ein Wert im Array enthalten ist. Achtung: *typeof* liefert für *null* ebenfalls 'object'.

a. Implementieren Sie die Funktion *equal* und testen Sie anhand einiger Beispiele, ob die Funktion korrekt arbeitet. Arrays sind ebenfalls Objekte. Ihre *equal*-Funktion sollte daher auch den Inhalt von Arrays vergleichen.

b. Fakultativ: Erweitern Sie Ihre Funktion zu einer Funktion *deepEqual*, welche auch den Inhalt verschachtelter Strukturen vergleicht. Funktionen können in JavaScript rekursiv sein, sich also selbst aufrufen.

Abgabe

Geben Sie die Funktion *equal* ab.

Abgabeserver:	https://radar.zhaw.ch/python/UploadAndCheck.html
Name Praktikum:	WBE2
Dateiname:	equal.js
Funktionsname:	equal
Export im Script:	`module.exports = { equal }`

Aufgabe 3: Übung zu Strings (Abgabe)

Schreiben Sie eine Funktion *findTag*, welche aus einem String das erste Tag (also ein Text in spitzen Klammern, darf keine Leerzeichen enthalten) findet und den Tagnamen zurückgibt:

```
> findTag("<header>Text</header")
"header"
> findTag("blabla <br> blabla")
"br"
> findTag("123245 </header> bla")
"/header"
> findTag("123245 <hea der> bla")
undefined
> findTag("123245 <hea<der> bla")
"der"
> findTag("123245 <hea<der bla")
undefined
```

Sie können mit `"blabla"[n]` auf das n-te Zeichen des Strings zugreifen und eine for-Schleife verwenden (es gibt aber auch andere Möglichkeiten).

Abgabe

Geben Sie die Funktion *findTag* ab.

Abgabeserver:	https://radar.zhaw.ch/python/UploadAndCheck.html
Name Praktikum:	WBE3
Dateiname:	find-tag.js
Funktionsname:	findTag
Export im Script:	`module.exports = { findTag }`