OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

Übungsaufgaben Blatt 6 - JavaScript Architektur und Design Patterns

Medieninformatik Webtechnologien - Prof. Dr. Markus Heckner

Aufgabe 1 : Closures

Gegeben ist das folgende Codebeispiel:

```
function makeMultiplier(multiplicator) {
    return function(multiplyBy) {
        return multiplicator * multiplyBy;
    }
}

var multiplyBy10 = makeMultiplier(11);
var multiplyBy5= makeMultiplier(3);
console.log(multiplyBy10(11));
console.log(multiplyBy5(6));
```

Erläutern Sie die Konzepte Closure, Funktion und Sichtbarkeit anhand der Codebeispiele. Was gibt der Code aus, warum?

Aufgabe 2 : Eigene Objekte

Sehen Sie sich das folgende Codebeispiel an. Was macht der Code?

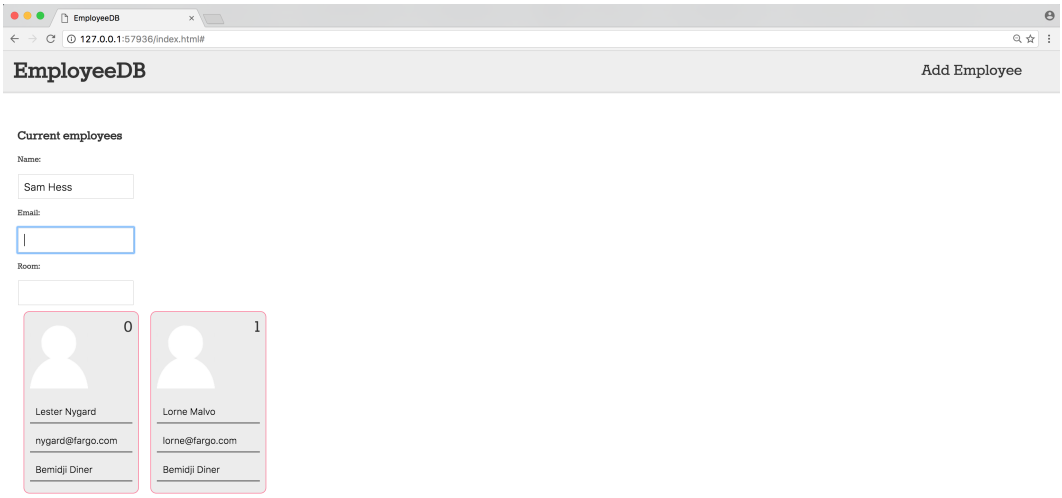
```
/* ShoppingListItem stores data for one entry in the shopping
   list */
var ShoppingListItem = function (title) {
    this.fullTitle = title;

    this.name;
    this.quantity;

    this.parseTitle();
    this.convertToUpperCase();
};

ShoppingListItem.prototype.convertToUpperCase = function () {
    var firstChar = this.name.charAt(0);
    var firstCharUpper = firstChar.toUpperCase();

    var shortenedTitle = this.name.substring(1);
    this.name = firstCharUpper + shortenedTitle;
};

```

```
20 ShoppingListItem.prototype.parseTitle = function () {
21     if (this.fullTitle.indexOf(":") !== -1) {
22         var splitTitle = this.fullTitle.split(":");
23         this.name = splitTitle[0];
24         this.quantity = splitTitle[1];
25     } else {
26         this.name = this.fullTitle;
27         this.quantity = 1;
28     }
29 };
30
31 ShoppingListItem.prototype.getTitle = function () {
32     return fullTitle;
33 };
34
35 ShoppingListItem.prototype.toString = function () {
36     var quantityString = "(" + this.quantity + ")";
37     return this.name + " " + quantityString;
38 };
```

- Erläutern Sie die Einzelbestandteile des Codebeispiels.
- Erläutern Sie das Prinzip Information Hiding anhand des Codebeispiels.
- Erstellen Sie den Client-Code zum Erstellen des Objekts und lassen Sie sich den Titel und die Stringrepräsentation des Objekts ausgeben.
- Funktioniert die Ausgabe des Titels wie geplant?

Aufgabe 3 : Mitarbeiterdatenbank - Entwicklung des *Backend*-Moduls

In dieser Aufgabe sollen Sie das *Backend* einer clientseitigen Mitarbeiterdatenbank in JavaScript implementieren.
Die Darstellung der von Ihnen verwalteten Daten wird von einem anderen Programmierer aus Ihrem Team übernommen. Da Sie sich für eine klare Trennung zwischen Darstellung und Applikationslogik mit dem *Module-Pattern* entschieden haben, können Sie jetzt Ihr Modul entwickeln, ohne auf die Fertigstellung der Arbeit des Kollegen angewiesen zu sein.
Die folgenden Abbildung zeigt einen Screenshot der fertigen Anwendung. Achtung, Sie müssen lediglich das Backend und die damit verbundene *Business-Logik* und nicht die komplette Anwendung entwickeln!



Ihr Code soll es ermöglichen, Mitarbeiter anzulegen und dabei direkt abzuspeichern. Zusätzlich kann die Datenbank die Daten eines Mitarbeiters aktualisieren, wenn sie eine korrekte (d.h. existierende Mitarbeiter ID) und die zu ändernden Daten des Mitarbeiters übergeben bekommt. Die Datenbank kann immer nur alle Daten eines Mitarbeiters, und nicht einzelne Attribute, auf einmal aktualisieren.

Die Datenbank speichert die Einträge lediglich in einem Array, Sie müssen sich nicht damit befassen, wie Sie die Einträge permanent speichern sollen.

Achtung:
Für diese Aufgabe gibt es ein Starterprojekt in Grips, das Sie nutzen können. Dort finden Sie alle Dateien, sowie eine HTML-Datei, die den Code *zusammenhält*. Sie müssen lediglich die Funktionalität implementieren.
In dieser Aufgabe sollen Sie die folgenden Dateien wie folgt implementieren:

- `Employee.js` als Objekt mit Konstruktorfuntion und `prototype`.
  Der Mitarbeiter hat die folgenden Eigenschaften...
  - id
  - name

  - email
  - room

  und die folgenden Methoden:

  - getId
  - getName
  - getEmail
  - getRoom
  - toString - Diese Methode gibt den Mitarbeiter als String aus, z.B. wie folgt: ID: 1 Name: Lester Nygard email: nygard@fargo.com room: Office

- **`EmployeeDatabase.js`** nach dem *Revealing Module Pattern*
  Private Eigenschaft:

  - Array zur Speicherung der Mitarbeiter

  Öffentliche Methoden:

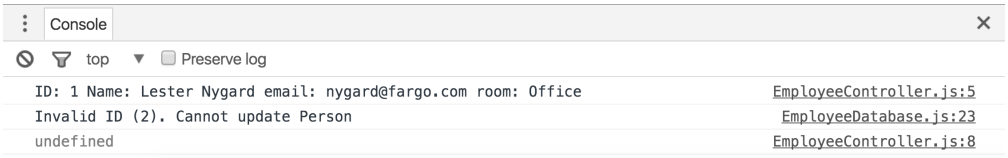
  - createEmployee. Diese Methode erwartet **`name, email, room`** als Parameter und speichert einen Mitarbeiter in der Datenbank (=*Array*). Die Methode gibt den gerade erstellten Mitarbeiter zurück (**`return`**). Hier müssen Sie ein Objekt *Employee* erzeugen. Mitarbeiter erhalten automatisch eine ansteigende ID, d.h. der erste Mitarbeiter erhält die ID 0, der zweite ID 1 usw.
  - updateEmployee Diese Methode erwartet **`id, name, email, room`** als Parameter. Die Methode prüft zunächst, ob ein Mitarbeiter mit der entsprechenden id in der Datenbank vorhanden ist. Wenn nicht, wird eine entsprechende Konsolenausgabe erzeugt (z.B. Invalid ID (2). Cannot update Person) und **`undefined`** zurückgegeben. Wenn ja, dann wird der Mitarbeiter entsprechend in der Datenbank aktualisiert und der gerade geänderte Mitarbeiter zurückgegeben (**`return`**).

- **`EmployeeController.js`** - Diese Datei dient zum Testen Ihrer Datenbank. Sie müssen keinen eigenen Code ergänzen.

Kommentieren Sie den Code in der Datei **`EmployeeController.js`** ein, und testen Sie Ihr Programm mit dem dort bereits bestehenden Testcode:

```
1 employeeDatabase.createEmployee("Lester Nygard", "nygard@fargo.
    com", "Bemidji Diner");
2 employeeDatabase.createEmployee("Lorne Malvo", "lorne@fargo.com
    ", "Bemidji Diner");
3
4 var resultUpdateSuccess = employeeDatabase.updateEmployee(1, "
    Lester Nygard", "nygard@fargo.com", "Office");
5 console.log(resultUpdateSuccess.toString());
6
7 var resultUpdateFail = employeeDatabase.updateEmployee(2, "
    Lester Nygard", "nygard@fargo.com", "Office");
8 console.log(resultUpdateFail);
```

Ihr Programm sollte in Chrome auf der Konsole die folgende Ausgabe erzeugen:



Ergänzen Sie abschließend eine Methode, die alle Mitarbeiter auf einmal als Array aus Strings zurückgibt.

Aufgabe 4 : **Mitarbeiterdatenbank - Integration des *Backends* und der Darstellung auf der Seite**

Mittlerweile ist auch Ihr Kollege mit der Entwicklung des Frontends fertig. Ihr Kollege hat das Frontend gegen ein *Dummymodul* entwickelt, das immer mit statischen Daten auf Anfragen antwortet aber keine echte Funktionalität bietet. Laden Sie den Code aus Grips herunter (*Integration des Moduls in fertige HTML Seite (Erweiterung)* und testen Sie die Seite im Browser.

Gehen Sie anschließend wie folgt vor:

- Integrieren Sie Ihren Code aus der vorhergehenden Aufgabe so, dass die Anwendung wie gewünscht, d.h. wie auf dem Screenshot der vorhergehenden Aufgabe funktioniert.
- Versuchen Sie anschließend nachzuvollziehen, was der Kollege implementiert hat und warum die Seite funktioniert!

Erweitern Sie jetzt die Funktionalität wie folgt: Bei Klick auf einen Mitarbeiter soll der Mitarbeiter aus der Mitarbeiterdatenbank gelöscht werden und auch auf der Anzeige der Webseite verschwinden. **Tipp**: Damit Sie einen Mitarbeiter aus der Datenbank löschen können, müssen Sie herausfinden, welche ID der geklickte Mitarbeiter hat. Die folgende Website zeigt Ihnen eine Standardmöglichkeit zur Lösung dieses Problems:
https://developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use_data_attributes.