Exercises React Day-1

1) Complete React's basic intro tutorial: https://facebook.github.io/react/tutorial/tutorial.html

2) Also, spend some time with this tutorial: http://buildwithreact.com/tutorial

3)

a) Create a simple create-react-app generated app, which renders a table with the data given below

b) Add a input element, used to input a filter value, so the table only should show artists with a rating above the value entered for the filter.

```
let musicians= [
      {id: 1, name: "James Hetfield", stars: 8},
      {id: 2, name: "Tina Turner", stars: 6},
      {id: 3, name: "Chris Martin", stars: 8},
      {id: 4, name: "Madonna", stars: 5},
      {id: 5, name: "Emmelie de Forest", stars: 1}
 ]
```

4) Complete part 1-4 of the third-semester exam exercise given below (don't focus on the graded part).

React, Components ~~and Mobx~~

General part

A common way to make a system easier to maintain, is to break it up into smaller, sometimes reusable, parts.

- Explain how java programs can be divided up into smaller parts
- Explain how React app's are composed by Components
- Explain how the JavaScript array methods, like `filter`, `map` and (`reduce`) can be used to generate dynamic HTML structures (tables, ul's etc.)
- Explain about the Observer pattern, and where you have used it, both with Java, JavaScript and Mobx.
- Explain the differences in designing a Component as an ES6 class versus a pure JavaScript function.

Practical part

Clone this project https://github.com/Lars-m/startCodeExamPrepMobx2.git *(type **npm install** to fetch dependencies and **npm start** to run)* and open the project in your favourite IDE. It contains *a create-react-app* generated project, ejected, and modified to provide start code (and ES7-decorator support) for this exercise.

The file *`dataModel.js`* contains a hardcoded data-model which you must use to create a table as sketched below (please note that values in the Average columns are derived (calculated) values, and not found in the data model :

	Basic Programming	Advanced Programming	DataBase Intro	Average
Peter Hansen	10	12		11.00
Jan Olsen	7	10		8.50
Gitte Poulsen	7	7		7.00
John McDonald	10		7	8.50

- The data model contains an array: *`headers`*, which hold all courses a student can take (the top row)
- It also contains a list of all `students` and their `grades`.

You can assume that when the model was built, it was done so that each student will have a number of grades

matching the number of courses found in the `headers` array, and in that order. If a student has not yet taken a course, it is added as an empty grade-item ({}).

Spend a few minutes with the data-model, and make sure you understand how it maps to the table visualized above.

1. Add the necessary code to create the header row (first column is empty, last is the hardcoded value "Average")
2. Add the necessary code to render the rows with name and grades (leave out the average column in this part)
3. Add the necessary code to render the rows with name, grades and the **average** grade for the student
4. It is assumed that the table will be used in more than one view. Refactor the table-code into a separate React component (`StudentTable`), and include this component in your App.js

~~5. Up until now, everything above has not been very reactive, since the table-view does not re-render if we change the list of students (add, remove etc. students).~~

~~Use mobx[1] to implement the necessary changes to the `dataModel` and other files, to make your app reactive (your view updates, if you change the model).~~

~~Hint: An easy way to test this is via the Chrome console, since the `dataModel` adds the data-structure to the window-object (see code): Test like this:~~

~~`info.students.push({studentId:34, name:"spiderman", grades:[{grade:"12" },{},{grade: "10" }]})`~~

How this would be graded:

2-4	**To fall into this range you must:** Give a minimal to fair performance related to the topics stated in the "General Part" And Have completed, ex 1+2 in the practical part with only a few weaknesses
4-7	**To fall into this range you must:** Give a fair to good performance related to the topics stated in the "General Part" And Have completed, ex 1+2 in the practical part with none or only minor weaknesses Have completed ex-3 with only minor weaknesses or alternatively 3+4 with some major/minor weaknesses.
7-10	**To fall into this range you must:** Give a very good performance related to the topics stated in the "General Part" And Have completed, ex-1 +2 in the practical part with none or only a few minor weaknesses Have completed , ex 3 +4 in the practical part with only minor weaknesses
10-12	**To fall into this range you must:** Give an excellent performance related to the topics stated in the "General Part" And Have completed, ex-1 +2 in the practical part with no or only a few minor weaknesses Have completed , all steps in ex-3 +4 in the practical part with only a few minor weaknesses Have completed ex5 with only a few minor weaknesses

[1] In this exercise dependencies are added to package.json. For the real exam-exercise you are expected to do this by yourself