Lab assignment 2

TI1506 2016/17; ti1506-students.slack.com

Introduction

In the first part of this assignment, you will add client-side JavaScript code to your application to make it *interactive*. In the second part of this assignment you will practice with a relational database instance, using both a command line interaction environment and a user friendly interface.

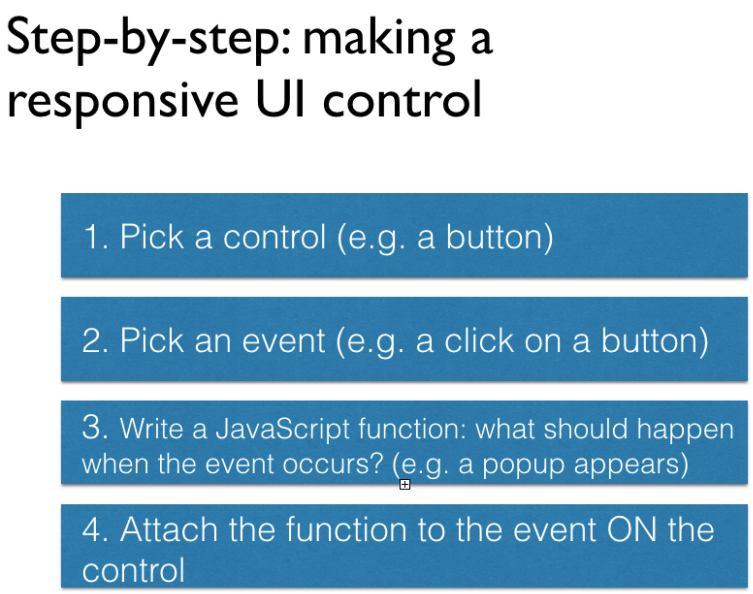
Deadline: when this assignment is due depends on the cluster your team is in. Every team is assessed biweekly. Make sure you know your cluster and what this means for your lab deadlines. All of this information is available in the course syllabus (on blackboard).

In order to pass the assessment, you have to make a valid attempt at working through the lab assignment.

Students who are already well-versed in Web technology can also attempt to make their application available offline (using our AppCache approach).

1. OVERVIEW OF ACTIONS PLAN

Recall the "Step by step" guide of the JavaScript lecture:



Create such a step by step plan for *each* UI element of your todo application that will become interactive. You must include the following interactive elements:

- Add a todo item with a due date and an importance rating
- Delete a todo item
- Set the status of a todo to "done"
- Change an existing todo item's content (i.e. todo text, due date, importance rating and status)
- Sort the items according to due date / importance

Add one more action of your choice depending on your initial design (e.g. allow a user to tag a todo item).

It should be possible to use the todo app with a keyboard alone or with a combination of keyboard and mouse (i.e. you need to implement actions for both types of events).

Ensure that the todo app behaves as we would expect based on our own experience: for instance, we expect that once a todo item has been added to the todo list, it should no longer be visible in the input element of the application.

Deliverable: a plan of actions

---

2. OBJECT-ORIENTED PROGRAMMING

Think about the design of your JavaScript code – which aspects of your action plan can you translate into the OO paradigm? For example, you might want to model each todo item as an object. Similarly, you can also think about modeling the list of all todos as an object and so on. Decide on the use of at least one of the introduced JavaScript design patterns.

Deliverable: a description of how you will aim for your JavaScript code to be object-oriented.

---

3. WRITING CODE

Now that you have made your plan and decided on the use of OO principles, start coding! Implement your plan of actions one action at a time. Reduce the redundancy in the code as much as possible. Create as few global variables as possible.

Strive to achieve a **complete separation between content and interaction**: the JavaScript-based interaction functionality should not be embedded in your HTML files. Likewise, in the next assignment the styling information should not be embedded in your HTML files. You should separate JavaScript, CSS and HTML as much as possible. While this causes more work initially, it pays off in the long run: your code can be more easily debugged, it is more readable and more easily extendable.

Figure 1: Keeping things apart. Separate interaction, presentation and content.

Deliverable: the project files (HTML and JavaScript).
Note: do not incorporate style elements yet (CSS), we will cover the style in a later lecture.

---

4. START MY SQL SERVER

Now it is time for you to have hands-on experience with a real-world relational database environment. The description of the assignment assume you are working on the provided virtual machine. If you decided to install your own database instance, then you might experience differences in the available commands.

In this exercise you will get familiar with the command line instructions required to start/stop/restart an instance of the MySQL server. To this end, open the file

`HOW TO START MYSQL.txt`

Located on the desktop of your virtual machine.

Deliverable: this exercise has no deliverable, but its successful execution will allow you to perform the following exercises

5. CONNECT TO SQL SERVER VIA COMMAND LINE

This exercise and the following require you to use the command-line interface (CLI) of your operative system. The goal is to let you familiarize with the MySQL command line client, and explore a running instance of the MySQL server.

Import the database dump contained in the `DUMP` folder into your MySQL server instance using the same procedure you used to import the IMDB database used in class.

Use the MySQL client to connect to the `ToDo` application database. Start your "conversation" with the database with

```
================================

$ mysql -u $USERNAME$ -p

================================
```

Use the following MySQL proprietary commands to check for the version and list of databases currently included in the MySQL server instance.

- `SELECT VERSION();`
- `SELECT now();`
- `HELP`
- `HELP Contents`
- `HELP Data Manipulation`
- `HELP SHOW DATABASES`
- `SHOW DATABASES;`

Deliverable: For each executed command, copy and paste in a textual file the returned response.

6. EXPLORE THE STRUCTURE OF THE TODOLIST DATABASE

In this exercise you will start interacting with the `TODO` database, using the MySQL proprietary commands.

Use the following commands to explore the schema of the `TODO` database.

- `USE TODO`
- `SELECT DATABASE();`
- `SHOW TABLES;`

For each table in the database, use the `DESC` command (as exemplified below) to show the structure of each table, and write down the returned response.

```
DESC $TABLENAME$;
```

Deliverable: Given the response obtained by executing the previous commands, provide a *visual representation* (i.e. a drawing) of the schema of the `TODO` database. Adopt the visual notation used in the *Relational Model* section of the lecture slides to denote:

- Relations
- Attributes
- Primary Keys
- Foreign Keys and Integrity Constraints

You should be able to comment and motivate each element in the database schema.