



# Reduced Seminar 4 (2016), JavaScript

Internet Applications, ID1354

## 1 Goal

- Learn JavaScript.
- Learn to use jQuery to facilitate event handling and updates to the DOM.
- Learn to use a framework, for example Knockout, to improve architecture and design of the JavaScript client.

## 2 Grading

The grading is as follows:

**0 points** The mandatory tasks are accepted and you have passed the seminar.

**1 point** The mandatory tasks and one higher grade task are accepted. You have passed the seminar and have also gained one point to improve the final course grade, see course plan for details on final grade.

**2 points** The mandatory tasks and both higher grade tasks are accepted. You have passed the seminar and have also gained two points to improve the final course grade, see course plan for details on final grade.

To pass the LAB1 sub course you must pass all four seminars. If you fail this seminar you have to report it again at the end of the course, at the fifth seminar. You can also report higher grade tasks at the fifth seminar.

## 3 Auto-Generated Code and Copying

**All HTML, CSS, PHP and JavaScript code must be well designed and you must be able to explain and motivate every single part. You are *not* allowed to copy entire files or classes from the sample chat application, even if you understand it and/or change it.**

However, you are allowed to write code very similar to the chat application. You are also allowed to copy HTML and CSS from any web site and to use any web development tool, you do not have to write HTML and CSS by hand. In particular, you are encouraged to get inspiration from (or use) free design templates.



## 4 Reduced Mandatory Task

Note that you choose to do **either mandatory task 1 or optional task 1**. If you solve the mandatory task, you pass but do not get any higher grade points. If instead you solve the optional task, you pass and also get one higher grade point. This is because to do optional task 1 you have to learn one more framework.

### Reduced Task 1, JavaScript Client

Use JavaScript and AJAX for, at least, reading and writing and deleting recipe comments. To navigate between index, recipe and calendar pages, and to login/logout, may still be done by loading entire HTML documents. You are, however, allowed to use AJAX also for this, if you wish. You are not required to implement a viewmodel layer. Instead, you are allowed to store all data, like for example recipe comments, in HTML elements in the DOM tree. You are encouraged to use jQuery, since it is normally easier, but it is not a requirement.

- The report must show that AJAX is used for reading and writing and deleting recipe comments.
- The report must show how recipe comments are stored in the client (browser).
- The report must include parts of JavaScript code with explanation. You may also include PHP code if you wish. Remember to treat codes as figures. They shall be clearly separated from the text and have number and caption (*figurtext*). Include only small and interesting parts of the code. Do *not* include long code dumps.

## 5 Reduced Optional Tasks

### Reduced Optional Task 1, Use a JavaScript Framework

Write the functionality described in mandatory task 1, but using Knockout (or another JavaScript framework) for the viewmodel layer. You are *not* allowed to store recipe comments in HTML elements in the DOM tree. Instead, this data must be stored in a JavaScript viewmodel, which is managed by the framework. Your JavaScript code shall not read comments from the DOM, or write comments to the DOM.

- The report must show that there is a viewmodel, and that it is managed by Knockout (or another JavaScript framework).
- The report must include parts of JavaScript code, with explanation. You may also include PHP code if you wish. Remember to treat codes as figures. They shall be clearly separated from the text and have number and caption (*figurtext*). Include only small and interesting parts of the code. Do *not* include long code dumps.



## Optional Task 2, Long Polling

The client shall use long polling to update the viewmodel, and thereby the view. This means that whenever a recipe comment is written by a user, it shall immediately be visible to all other users watching the page where the comment was written.

- The report must show how long polling is used to communicate with the server.
- The report must include parts of JavaScript code, with explanation. You may also include PHP code if you wish. Remember to treat codes as figures. They shall be clearly separated from the text and have number and caption (*figurtext*). Include only small and interesting parts of the code. Do *not* include long code dumps.