Automated grading of Web Development Labs

Jean-Claude Dufourd

[1] LTCI, Télécom Paristech, Université Paris-Saclay, Paris, France
dufourd@telecom-paristech.fr

Abstract. We relate our experience with automated grading of web development lab work, implemented using web technologies.

Keywords: automated grading, web technologies.

1 Objectives

Our initial objectives were to create a Web Development course, teaching HTML, CSS, JavaScript and their common uses in web applications, for a large number of students and evaluated mostly from lab work. We first converted 2 labs to automated grading, and noted that average grade of automatically graded labs was significantly higher than the average grades of manually graded labs. We then converted all labs of the course to automated grading.

We chose to implement our grading server with node.js. The students have unlimited access to the grading server. There are no limits on the number of grading runs.

2 Grading situations

In this paragraph, we will explore the different grading situations. Lab1 teaches the basics of HTML and CSS. Lab2 teaches basic JavaScript. Lab3 teaches JavaScript in the browser. Lab4 and Lab5 teaches server programming with node.js.

The students create files in their environment and test them. The lab text specifies names to allow automated grading: file names, function names, HTML ids, CSS class names, server URLs. They zip their files and upload the archive to the grading server.

Grading Lab1 requires exploring the DOM tree and the CSS text to check if the work is coherent with the assignment. The grader reads the files and does a set of checks. Lab1 uses JSDOM and Promise and its complexity comes from the number of ways to achieve something in HTML/CSS.

Grading Lab2 requires testing plain JS code, which can be done with plain JS unit testing. The grader is integrated with the JS unit reporting. Lab2 is the easiest to change.

Grading Lab3 required testing JS code running in the browser, which means injecting grading JS, then sending results over the net to the grading server. This grading is sensitive to the browser used while grading. Lab3 is really ad-hoc code and is difficult to rationalize.

Grading Lab4/5 requires reproducing the execution context of a node.js server, testing that server and then stopping it. The grader sends requests to the student server, gets async responses, sometimes needs to chain requests. Lab4/5 are rather complex but follow a constant pattern of using request and Promise.

3 Problems

Problems include:

- the whole process of grading is built from multiple asynchronous tasks happening in at least two places, the grading server and the student code under scrutiny; we have made a choice of using HTTP communication to send grading decisions and sequencing triggers; we are thinking of switching to WebSockets in a next version;
- HTTPS is imposed by authentication and needs to be used for all communications with students; this increases the technical complexity of the grading system: HTTP exchanges need to be proxied;
- the grading decisions must be communicated to a central logging part of the grader; which is fragile from a security perspective; it should be as difficult as possible for the students to cheat;
- there are a few points we did not know how to grade automatically with a reasonable effort, so for 2 of the 5 labs, 20% of the lab is graded manually: after the end of the module, when making a final grading run, the grader admin is presented with a subset of the student work and a set of grading choices, for each student.

4 Evaluation

After two years of usage, here are some thoughts:

- **Lab description**: There is a diversity of ways to implement the same assignment; the lab assignment needed to be completely rewritten in a more precise fashion to be compatible with automated grading. Students are also very inventive to find new ways to understand the assignments.
- **Execution context**: The student testing contexts are different from the automated grading context; this sometimes requires convoluted explanations in the lab text.
- **Student motivation**: Many of the students will keep working until they have the maximum grade! Motivation seems significantly increased.
- **Pedagogy**: By alternating programming (**applying**) with **analyzing** their work, the students keep going one stage up in Bloom's taxonomy.
- **More pedagogy**: Some students send detailed reports of problems with the grading, and propose solutions, fixes and improvements. Their level of analysis is excellent. They are two stages higher in Bloom's taxonomy: **evaluating**.
- **Usage time**: The time not used for manual grading is more than compensated by the need for software maintenance of the grader.
- **Design time**: The creation of a new lab or the modification of an older lab requires a lot of time and testing. Rewriting and documenting the software to open it to other teachers is required as a next step.