Incorporating Computer Science into an Elementary School Curriculum

Maude Lemaire
McGill University
Montréal, QC, Canada

August 2014

Abstract

In Canada, computer science and related topics mainly consist of optional subject material available to students in high school and university settings . Educators in the sciences high school and elementary levels often feel intimidated by technology and thus unqualified to teach computer science. This article describes the development process behind writing and implementing a simple computer science curriculum aimed at grade five and six students. Focusing on Hypertext Markup Language (HTML), the curriculum gives the students an introduction to basic web development and the idea of syntax. Divided into eight one-hour lessons, each class incorporates an important programming component. Sixty-eight students between the ages of ten and twelve participated in a trial where three variations of the introductory curriculum were taught. In each version of the curriculum, a different number of resources were made available to the students. This article will demonstrate that the students' level of success in the course was directly affected by the availability of these resources.

1 Introduction

In a society becoming increasingly dependent on software, it is imperative that our younger generation develop the ability to think critically about technology. With a resounding push for engineering education in recent years, the availability of computer science education to students of all levels has been a growing discussion at educational ministries within the government of numerous industrialized nations, particularly in France and the United Kingdom [2]. The movement has led to the development of dozens of online platforms dedicated to educating children levels K-12 and beyond, including Codecademy (www.codecademy.com), CodeHS (www.codehs.com), Code.org (www.code.org), Scratch (scratch.mit.edu), and Alice (www.alice.org) [6]. Available in multiple languages and to anyone with an Internet connection, many of these websites offer content for teachers wishing to integrate programming into their classrooms;

most, however, focus on providing tutorials directly to students. In order to maximally benefit from these resources, those enrolling in any such program must have the self-discipline to complete the curriculum during a set period of time. Most students will also enroll with a specific goal in mind: for instance, Codecademy suggests different tracks for developing relevant employment skills, learning web-related technologies, or tackling specific languages such as Python or Ruby[4].

Although valuable, these platforms lack the active teaching component necessary to engage younger children in a computer science curriculum and encourage them to further pursue these interests. A self-motivated individual will more likely succeed in learning a new programming language by following tutorials on Codecademy than one that is less autonomous due to the fact that many of the tracks offered by the organization involve lengthy readings and report an estimated adult learning time of over ten hours[5]. With a monochrome interface barren of images, children are quickly disinterested.

A number of recently developed programs such as Code.org are beginning to feature series of "kid-friendly" lessons. However, these tend to be highly dependent on the motivation of students' school instructors or parents to first introduce the children to programming and subsequently monitor their progress[4]. Teachers with an aptitude for frequently using technology during class and a background in engineering are more open to introducing their students to programming; in fact, it was found that despite the push for more technology in the classroom within Montreal school boards, the average elementary school educator was unaware that there already exists computer science curriculum material tailored to younger students at his or her disposal. The problem grows more acute with those instructors actively reluctant to use technology[7].

With the approach taken in this study, computer science topics were introduced to elementary school students by volunteer university students and professionals in software engineering and related disciplines. These volunteer instructors each had at least two years of programming experience. Groups of fifteen to twenty students in grade five and six were taught basic HyperText Markup Language (HTML) for a period of four to eight weeks. A curriculum similar to those offered by Codecademy and Code.org was presented in a child-friendly manner and divided into eight equal-length teaching periods.

The study additionally addressed the challenges faced in incorporating computer science topics into a pre-existing elementary school curriculum with as little disruption as possible. Instructors wishing to introduce their students to computer science are heavily hindered by strict governmental guidelines and restrictions; with standards set forth both at the school board and provincial level, teachers must be able to tailor technology topics to existing competencies in order to justify their inclusion.

1.1 Students' use of computers in elementary classrooms

Students at the elementary level are already interacting with computers on a daily basis[3]. This growing generation of technology consumers has been

interacting with the Internet, tactile devices and cell phones from an early age. Recent studies show that 54% of 21st century children start using mobile devices when they are 5 to 8 years old [3]. Basic computer skills are becoming a standard requirement of most entry-level positions and educational systems have been slowly responding to meet that demand by investing more time and money in understanding how to incorporate technology in the classroom. Due to budget limitations, a student's access to technology at school can vary widely from one region to another; within the Montreal school system alone, there are schools equipped with a laptop for every child, just as there are those with only one computer lab shared across all grade levels.

In the United States, learning to type correctly has only recently become a requirement for elementary-level students. With the new national Common Core academic standards, children as early as kindergarten are learning to use a keyboard and mouse[6]. This requirement is currently lacking in most public schools across Canada with many young adults graduating high school having developed inefficient touch-typing techniques. In the frenzy to use technology in every subject, elementary-level students find themselves dragging pictures into Microsoft Word files to enhance their compositions and using search engines to find information for a research project. Although children are familiarizing themselves with keyboards, mice and websites, they lack a basic understanding of how these technologies function.

1.2 Effects of computer use in elementary classrooms

Use of laptops and tablet computers have increased student engagement by providing a fun learning environment. These technologies can be tailored to each students' individual learning style through games and story-telling applications. Student engagement can perpetuate beyond the classroom; with access to a wide range of apps from any mobile device, students can continue their learning beyond the classroom into the home. Homework may no longer feel like an obligation. A study recently conducted by Everyday Family demonstrates that 77% of parents acknowledge that usage of tablets increase children's learning and creativity[3].

On the other hand, some parents and educators have criticized the use of technology in the classroom as it reduces physical activity[3]. With motion-sensor technology readily available to consumers, however, children can interact with apps and games while being physically active. Gaming consoles such as the Microsoft Kinect and Nintendo Wii provide this feature at a consumer-friendly price[1]. Unmonitored use of computers often result in students accessing social media websites and playing online games decreasing their productivity[3]. By keeping a watchful eye over younger students' use of technology, yet facilitate access to educational resources, the benefits can outweigh the distractions[3].

1.3 Research questions

In an elementary school curriculum, the central questions under investigation were the following: (1) What sorts of resources do children at the grade five and six levels need to learn HTML? (2) What factors and/or resources help motivate these students to continue their interest in programming? (3) Is having an active educator a significant advantage? (4)

2 Methods

The examination of students' ability to learn basic HTML and implement a simple web page was pursued using two methods: a student self-evaluation and a graded evaluation of the students' projects. Participants that completed the program submitted a self evaluation after their final lesson and their final web page was evaluated by the instructors according to a set rubric.

2.1 Participants

Study participants included fifth and sixth grade students from three public Montreal school districts: Commission scolaire de Montreal, Lester B Pearson school board, and the English Montreal school board. Arrangements were made with individual elementary school administrators to choose an appropriate classroom for the study. These groups ranged from 12 to 30 students for a total of 68 participants. Students were taught in their usual language of instruction (either French or English depending on the school board). Parental consent was obtained prior to enrolling the students in the program, none of which received compensation.

2.2 Materials and procedure

2.2.1 Self Evaluation

A self-evaluation was designed and distributed to each student following the their final computer science lesson. The evaluation consisted of 18 statements; the students were instructed to rate the level with which they agreed with each statement on a 5-point scale corresponding to the following descriptions: I strongly disagree, I somewhat disagree, I neither agree nor disagree, I somewhat agree, and I strongly agree. Some statements gauged the student's interest in continuing to create HTML content, others measured student confidence in identifying key portions of an HTML tag, and others yet sought to evaluated whether sufficient resources were provided for the student. A full list of questions provided on the self-evaluation is provided below:

(1) I learned a great deal from this course; (2) I think computer science has some exciting career opportunities; (3) I discovered a new way of expressing myself creatively through HTML; (4) Because of this course, I want to learn more about computer science and/or web development; (5) I was provided with

enough resources and guidance to keep programming on my own; (6) I am excited to make web pages of my own at home; (7) I learned a great dela from the instructor(s) for this course; (8) I feel confident making pages on my own; (9) I feel comfortable collaborating with others to fix my code or others' code; (10) I understand the components of an HTML tag; (11) I understand the components of a style attribute; (12) I understand the relationship between a browser and HTML; (13) I can format `divs` using padding, margins, and borders; (14) I can confidently create both an ordered and unordered list; (15) I feel confident debugging my code on my own; (16) I understand the value in developing incrementally and checking my progress consistently; (17) I enjoy creating content using HTML; (18) I believe learning about technology can help me in other aspects of my education.

2.2.2 Project Evaluation Rubric

A four-point rubric was used to evaluate completed student projects. The rubric was divided into six key in order to group similar requirements and assign an overall value to the student's execution of each category. The categories consisted of understanding sequence, student expression and creativity, iterative and incremental development, testing and debugging, basic HTML syntax, and formatting. For each category, a student could receive a minimum score of 1 and a maximum score of 4; partial points were awarded only within a 0.5 range. Scores per category were average for each group, and likewise students were given an average overall score.

The progress of any given project was not monitored; only the final result was evaluated using the rubric. As students were encouraged to work on their projects outside class time, it is unknown whether a project may have benefited from additional, individualized attention beyond that provided by the curriculum instructors.

2.2.3 Course and curriculum design

The introductory HTML curriculum was divided into 8 lessons, offered on a weekly basis for a total of 8 hours of class time over a 2 month period. The time of day at which the course was taught was left to the discretion of the classroom teacher. It is imperative to note, however, that in all three groups excluding the control group, students were taught between the hours of 10:00 am and 1:00 pm.

In order to seamlessly incorporate HTML instruction into public elementary schools, core Quebec educational competencies were matched to each lesson. These include (1) to propose explanations for or solutions to scientific or technological problems; (2) to make the most of scientific and technological tools, objects and procedures; (3) to communicate in the language used in science and technology; (4) to use information and communications technologies; (5) cooperates with others.

The curriculum revolves around the creation of a single HTML page viewable

in any browser. By default, it was suggested to have students write about three of their interests[1]. For each of these interests, students were required to include a short paragraph and incorporate images, videos, links to external web pages.

Each lesson follows the same format: first, there is a series of revision questions to help students recall the previous week's content. A new `tag` is then introduced with multiple examples. Next, students incorporate the new tag into their project. Finally, a half-dozen revision questions conclude the lesson.

The topics covered are as follow in the order given: (1) introduction to computer science, anatomy of a `tag`; (2) body, headers, bold, italic and underlined text, links, paragraphs, line breaks; (3) style attributes; (4) images; (5) debugging; (6) adding layout including lists, tables, divs; (7) embedding video, basic CSS; (8) HTML review game. The full curriculum is available for download upon request via Kids Code Jeunesse (http://www.kidscodejeunesse.org). Sample "About Me" projects can be viewed on the General Vanier Elementary School website (http://www.emsb.qc.ca/generalvanier/KCJeunesse.asp).

2.3 Classroom Groups

From early January 2014 to late June 2014, four sets of students participated in basic HTML lessons:

The first group, taught in English, was given the first iteration of the curriculum this included using JSBin as the primary development tool, and having access to two instructors in the classroom for questions. A "cheat sheet" was provided as quick reference for tag names and examples of how to use them.

The second group was given the first version of the curriculum, but in written form. The students were not given access to an active teaching component, nor provided with help from the instructors. The corresponding "cheat sheet" was provided.

The third group was given the second version of the curriculum - this included using Mozilla Thimble as the primary development tool, added an introduction to HTML by navigating web pages using Mozilla X-Ray Goggles, and reorganized some content so as to introduce images and styling earlier. A second version of the "cheat sheet" was provided with more in-depth examples.

Finally, the fourth group was given the second version of the curriculum, but with added examples to which they had access. For each lesson, an example project was provided so that material could be easily introduced in the manner of a live coding session. The code associated with these examples was printed and given as a reference tool to the students.

[1] One participating instructor chose to model the curriculum around developing a web page about Earth Day. The content is made to be easy extensible and applicable to variety of topics.

3 Results and Discussion

3.1 Do students learn computer science topics best with a teacher present?

This inquiry is best answered by comparing the second group to the other three student groups. A total of 12 students were observed in the second group without guidance from an instructor. Group one was of identical size and received active instruction from a volunteer teacher and were able to ask questions. The mean total score of a project completed by a student in Group 2 was 2.420; the mean total score of a project completed by a student in Group 1 was nearly 70% higher at 3.486. This indicates that without a doubt, students benefited from having an active educator present in the classroom to both lead each lesson and help individual students. More detailed results from both of these groups are available in Tables 1 and 2.

Group 1													**Averages**
Sequence	4	4	4	4	3.5	4	4	4	3.5	4	4	4	3.917
Expressing	2	4	4	2	2	3.5	3	1	3	4	4	4	3.042
Iterative	3	3.5	4	4	2.5	3	4	2	3	4	4	4	3.417
Debugging	3	3	4	3	3	3	4	3	3	3.5	4	4	3.375
Basic HTML	4	4	4	3	2.5	4	4	4	4	4	4	4	3.792
Formatting	4	3	4	2	2	4	4	2	3.5	4	4	4	3.375
Student Avg	3.33	3.583	4	3	2.583	3.582	3.83	2.66	3.33	3.916	4	4	3.486

Table 1: Individual student results by category for Group 1, taught in January 2014.

Group 1													**A**
Sequence	2	3	1	4	2.5	3	2	4	3.5	4	4	3	3.0
Expressing	2	1.5	1	2	1	1.5	1.5	4	2	4	1.5	1.5	1.9
Iterative	2	2	1	2.5	1.5	2.5	1.5	3.5	2	3	3	2.5	2.2
Debugging	2	2.5	1.5	2.5	2	2	2	4	2.5	3	3	2	2.4
Basic HTML	3	3	2	3	2	3	3	4	3	3	3.5	3	2.9
Formatting	2	1.5	2	2	1	2.5	1	4	2	2	2	2.5	2.0
Student Avg	2.16	2.25	1.416	2.66	1.66	2.416	1.83	3.916	2.4583	3.16	2.83	2.416	2.4

Table 2: Individual student results by category for Group 2, taught in February 2014.

Unsurprisingly, upon analyzing the results from these students' self-evaluations, it was observed that the students that received the least guidance felt more less confident in their programming ability. In fact, when comparing results for the tenth statement in the self evaluation, "I understand the components of an HTML tag", the percentage of students from Group 2 (Control) which said that they "agreed" or "strongly agreed" was lower by 16 points to that from

Group 1 (Classroom). Similar statements which were aimed at measuring a student's confidence level with different technical aspects of the curriculum showed a similar trend and are highlighted in yellow in Figure 1.

Other significant observations which can be made from the results are highlighted in green. The first is the difference between the two groups in those interested in learning more about computer science and/or web development. Although the students that received instructional guidance and a more active learning experience were still eager (at 80%) to continue to learn about topics discussed in the curriculum, a significant amount more students from Group 2 showed an interest in pursuing these studies further. The second indicates the drastic difference in student-teacher interaction between Group 1 and Group 2.

3.2 Which educational resources, excluding active educators, are necessary for facilitating a successful learning environment?

With each iteration of the HTML curriculum, better student engagement and a higher quality of final projects was anticipated. In order to evaluate the effect of each iteration, a stronger emphasis was placed on project evaluations rather than self-evaluations. As these changes were more subtle, it was decided that the most objective rubric results would shed light on the development of the curriculum.

By opting for a more kid-friendly development tool, it was hypothesized that the quality of projects would increase as less time would be spent hunting for mistakes in the student's code and there would be fewer mishaps with accidentally losing one's work. This change occurred between the January and May groups, but the average student score was quite similar; in fact, student scores in testing and debugging dipped by 0.24 points as can be seen in Figure 2.

With the adoption of another Mozilla tool to help students visualize how tags are being used in the context of website they frequently visit, X-Ray Goggles was added to the second iteration of the curriculum. A change was anticipated in the Basic HTML category, where students are marked on their syntax and the variety of tags they adopt. A large decrease of 0.79 points in the Basic HTML category was observed, due to the fact that students consistently used the same tags throughout their page after having seen emerging patterns through the X-Ray Goggles tool.

It is hypothesized that the more detailed, colorful examples introduced to the students within the June iteration helped level the playing field with the initial January version of the HTML curriculum. Figure 2 displays the increase in average scores within all categories, most significantly within the Basic HTML column. These results suggest that the examples tailored to the program are more relevant to the students' individual projects and thus more beneficial than using the Mozilla X-Ray Goggles to view the structure of example web content.

Unsurprisingly, in each of the six categories, the February group displayed lower performance. Not only were the students less confident about their pro-

gramming ability as was displayed in their student evaluations, but the projects they created were of lesser quality. In fact, having an instructor actively teaching the material and helping students apply the intended skills has the most important effect on project quality. This can also be viewed in Figure 3 where the average overall score is compared between each iteration of the curriculum.

The final results indicate that the most successful version of the curriculum was that taught in January. One would hypothesize that other important factors including language, teaching experience of the volunteer instructor, and the importance the classroom instructor placed on including HTML into their students' schedule could also have impacted student performance.

4 Conclusion

There are dozens of existing platforms tailored to teach students grades K-12 basic programming concepts and beyond. Although valuable, these platforms lack the active teaching component necessary to engage younger children in a computer science curriculum and encourage them to further pursue these interests. The results of this study indicate that a highly tailored curriculum taught in a classroom setting by enthusiastic instructors yields to superior student involvement and better understanding of the subject material. Unfortunately, we are now faced with the challenge of integrating computer science topics within government-sanctioned educational policies. Perhaps with added interest in the field of computer science, and a stronger push for computer science education within classrooms instead of in the homes of students, the next generation of Canadians will be more technologically literate and ready for modern careers.

References

[1] Miller Andrew. Kinect in the classroom, September 2012.

[2] Miles Berry. *Computing in the national curriculum.* British Library Cataloguing in Publication Data, September 2013.

[3] Santosh Bhaskar. Impact of technology in elementary classrooms, September 2013.

[4] Codecademy. Codecademy, August 2014.

[5] Fred Guterl. Can children teach themselves?, February 2013.

[6] Matt Richtel. Reading, writing, arithmetic, and lately, coding, May 2014.

[7] Sheena Vaidyanathan. We need coding in schools, but where are the teachers?, December 2013.

Control Group	Classroom	Difference	Corresponding Statement
Total % of students that said "They agreed" or "Strongly agreed" with the statement	Total % of students that said "They agreed" or "Strongly agreed" with the statement		
81.82	100.00	18.18	I learned a great deal from this course.
81.82	88.00	6.18	I think computer science has some exciting career opportunities.
81.82	88.00	6.18	I discovered a new way of expressing myself creatively though HTML.
90.91	80.00	-10.91	Because of this course, I want to learn more about computer science and/or web development.
81.82	84.00	2.18	I was provided with enough resources and guidance to keep programming on my own.
90.91	88.00	-2.91	I am excited to make web pages of my own at home.
81.82	96.00	14.18	I learned a great deal from the instructor(s) for this course.
72.73	64.00	-8.73	I feel confident making pages on my own.
81.82	88.00	6.18	I feel comfortable collaborating with others to fix my code or others' code.
63.64	80.00	16.36	I understand the components of an HTML tag.
45.45	80.00	34.55	I understand the components of a style attribute.
72.73	72.00	-0.73	I understand the relationship between a browser and HTML.
36.36	74.00	37.64	I can format "div"s using padding, margins and borders.
72.73	76.00	3.27	I can confidently create both an ordered and unordered lists.
54.55	80.00	25.45	I feel confident debugging my code on my own.
81.82	84.00	2.18	I understand the value in developing incrementally and checking my progress consistently.
90.91	88.00	-2.91	I enjoy creating content using HTML
90.91	92.00	1.09	I believe learning about technology can help me in other aspects of my education.

Figure 1: Self evaluation results and comparison for Group 1 and Group 2.

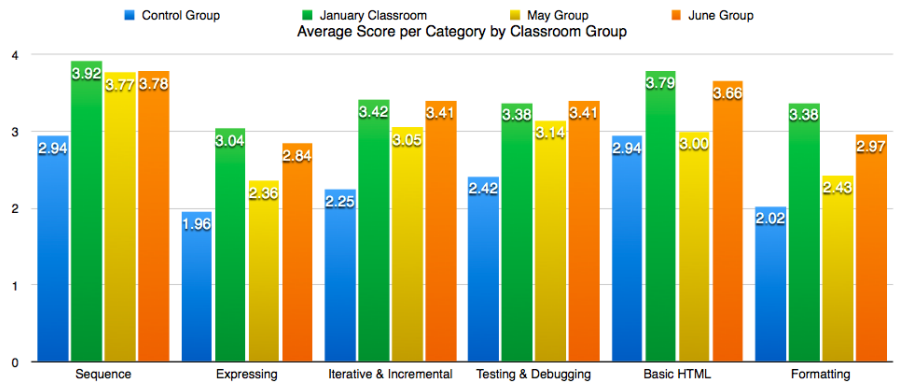

Figure 2: The average student scores by individual category for each classroom group.

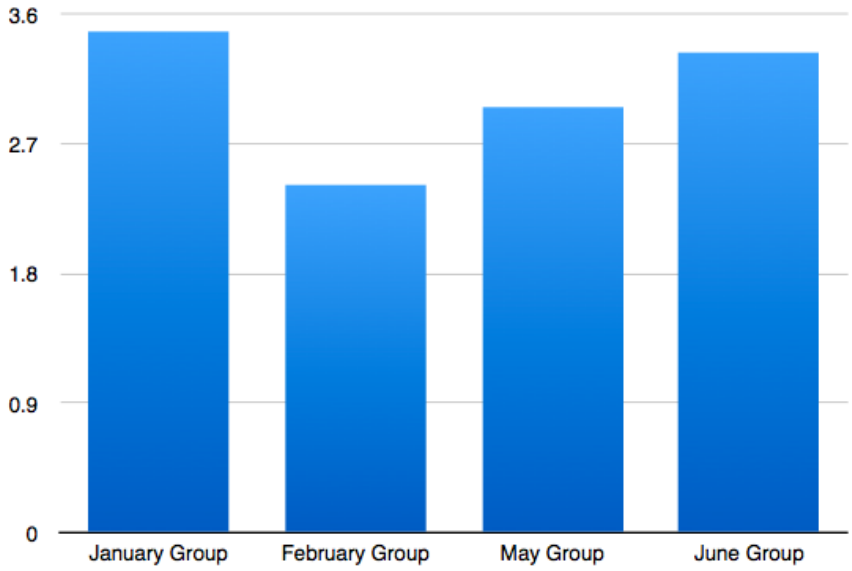

Figure 3: The average scores over all categories by classroom group.