Does implementing basic computer programming skills into the senior level Algebra 3-4 curriculum increase creativity and problem solving ability while also improving participation and performance of the students in the class?

By

Charles Cuddy

A Teacher Action Research Project

Presented to the Faculty of

the Teacher Education Department at Midland University

in Partial Fulfillment of Requirements

for the Degree Master of Education

in Leadership in Teaching and Learning

Under the Supervision of Carla Noerrlinger, Instructor

Omaha, Nebraska

June, 2016

Abstract

The study was conducted to determine if introducing computer markup and programming skills to high school seniors changed the students' participation, attitude, and interest in math, thus leading to a higher performance, as measured by their grades in the course. By implementing immediate applications through technology to concepts learned in Algebra 3-4, the students in this study, all of whom are below grade level in math, answered the over-asked question, "When am I ever going to use this?" by building their own websites and applications. The students were guided with a blended approach of class lectures and their use of Codecademy, a free online coding program, which has built in tutorials, projects, and resources to assist the students in their learning. Through this process, students learned more about, and improved their problem solving, logical reasoning, and creativity, that was involved in the algebra they learned. This created a more positive outlook from the students regarding the importance of math in their daily lives, which led to a higher participation rate in class and ultimately a higher grade for the students in the course.

Table of Contents

Introduction

The state of Nebraska is ranked the third fastest growing state for technology jobs in 2015, based on a hiring growth of 5.22 percent (Adams, 2015, p. 1), and yet there is little to no computer programming opportunities offered to high school students in the Omaha Public Schools (OPS). This is not out of the norm, as 90 percent of public schools nationwide did not teach computer science as of 2013 (Layton, 2013). This trend has changed recently, however, and will continue to change in the future, as multiple states have adapted a computer science curriculum for their students. The programs come highly endorsed from big names in the technology community, such as Mark Zuckerberg (creator of Facebook) and Bill Gates (founder of Microsoft), as well as major companies, like Google, who recognized the need to expose our future, our students, to programming in the public schools and have invested millions of dollars into free courses for anyone to learn how to program ("Every student in every school," 2015).

Omaha Bryan High School's population is nearly ten percent above the district average for free and reduced lunch price, a large indicator of poverty, with 83.4 percent of all students listed in this category. These students potentially lack the resources at home to take advantage of these free opportunities presented to them from the likes of Zuckerberg and Gates. Omaha Bryan also ranked below the district average on the Mathematics portion of the Nebraska State Assessment (NeSA-M) test by 4.67 percent during the 2013-2014 school year, with just over a quarter of the students considered proficient (Nebraska Department of Education's State of the Schools Report). When presented with an alternate method of learning math through computer programming, students better related to the subject, applied the material, completed more assignments, and eventually earned higher grades on their tests.

The students developed problem solving skills, logical reasoning, and creativity to apply their algebra knowledge toward building websites. Computer programming is not just a fad, and with the backing from Barack Obama, a huge advocate for coding in the classroom ("Every student in every school," 2015), coding could be finding its way into public classrooms across the country. In his State of the Union Address, Obama pointed out that,

> Growing industries in science and technology have twice as many openings as we have workers who can do the job. Think about that—openings at a time when millions of Americans are looking for work. That's inexcusable. And we know how to fix it (Koebler, 2012).

Opportunities were there for students to learn programming through open resources online, like Code.Org and CodeCademy.com, among others, while they maintained and expanded on the necessary math credits needed for graduation. Students did not just learn about programming and math, however, they also learned how to think.

Purpose of the Study

The purpose of this study was to determine if students turned in more assignments, earned better grades, and demonstrated a more positive attitude about their math class and its applications in their daily lives because of the unit on computer programming. The purpose also was to determine if seniors who were exposed to computer programming in their senior level math course possessed a heightened interest in pursuing math-related jobs or degrees upon graduation

Research Question

Does implementing basic computer programming skills into the senior level Algebra 3-4 curriculum increase creativity and problem solving ability while also improving participation and performance of the students in the class?

Language Integral to the Study

Term One: Basic Computer Programming Skills

The markup and programming languages that were taught included Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript because of their "efficiency of code and ease of maintenance" (Mills, 2014 p. 1). These are often referred to as the building blocks of the Web.

Term Two: Senior Level Algebra 3-4

Senior level Algebra 3-4 is a junior level math course that is composed of 100 percent seniors who need an extra year to complete their required math credits in order to graduate on time. All students in this class take Senior Level Algebra 3-4 because they had previously failed Algebra 3-4 or another math class in their high school career.

Term Three: Participation

In this study, participation is defined by comparing homework completion before the study, during the study, and briefly after the study.

Term Four: Performance

To determine performance as related to academic achievement, students' Algebra 3-4 grades were compared from the end of second quarter to grades from the end of third quarter to ascertain improvement.

Term Five: HyperText Markup Language (HTML)

HTML is the oldest web markup language and considered the skeleton of the web, in that it makes up the basic framework for most websites. HTML controls the presentation of data.

Term Six: Cascading Style Sheets (CSS)

CSS is the markup language that gives the HTML code it's the style, color, positions, etc. on the webpage (often referred to as the skin to HTML's bones).

Term Seven: Javascript

JavaScript is the programming language that makes the website interactive and animated.

Term Eight: Level 4 Problem

An assessment problem that involves higher level of critical thinking, including but not limited to making connections to previously learned material and expanding it beyond what was specifically taught in the class.

Literature Review

This study was conducted to determine if implementing basic computer markup and programming skills into a senior level Algebra 3-4 curriculum improved participation and performance of the students in the class. Additionally, this study examined the impact computer programming had on students' creativity, problem solving, and students' overall interest in mathematics inside and outside of the classroom. The project also introduced students to career options of which they may not have known were available. Computer programming continues to gain ground all over the country, but is especially relevant in Nebraska, as Nebraska recently ranked third in a list of fastest growing technology states in the United States (Adams, 2015, p. 1).

Computer Programming and Careers Available Upon Graduation

This study focused on teaching three markup and programming languages that make up the “basic building blocks of the web” (Mills, 2014, p.1). Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and Javascript were the three languages were selected because of their “efficiency, ease of maintenance, and accessibility” (Mills, p.1). These languages work together to create everyday websites, and each language plays a different role. HTML is thought of as the skeleton that makes up each web page; it is in this markup language that the content of a webpage is written. Every heading, paragraph, image, and link that is written into the code that makes up a webpage is done with HTML. This content is then styled using another markup language, CSS. It is with CSS that the webpages gain their color, position, and general style. Finally, Javascript is the language that brings the site to life, allowing for drop-down tabs, animation, and interaction with the user to take place (Mills, p. 1).

The world is experiencing a technological revolution, and there is a much higher need for computer programmers than ever before. “Many people in the United States and around the world lack the education and skills required to participate in the great new companies coming out of the software revolution” (Andreesen, 2011, p. 4). The purpose of the study was to not only look at computer programming as a way to improve math scores, but also as a way to introduce high school seniors to a career in a much needed field. People with a background in programming have an upper hand in obtaining desirable, well-paying jobs, because as Adams (2015) points out, “the technology unemployment rate will keep going down, making it a buyer’s market for tech jobs” (p. 2). This study introduced seniors to potential career possibilities of which they were previously unaware before they learned basic programming and its applications to mathematics.

Improving Creativity, Problem Solving, and Logical Reasoning through Programming

The introduction of programming is not just to present mathematical topics in a new way to the students, but rather to introduce a new way of thinking to the students. When students were given the opportunity to use technology in a controlled environment, to explore learning on their own, they showed improvement, and there is “considerable research that supports the prospect that individuals can enhance their creative problem solving ability through training” (Kim et al., 2013, p. 172). The structure of the study and the unit on computer programming was a great framework for this training. Kim, Chung, and Yu (2013) explained that “if creative problem solving ability is enhanced with a training program for creative problem solving based on programming, digital fluency could be improved”, in which they defined digital fluency as “abilities to solve real life problems using technology” (p. 171).

Computer programming is also linked to problem solving, as Kalelioglu and Gulbahar (2014) discovered in their study that “computer programming is an important competence for the development of problem solving in addition to logical reasoning” (p. 33). Programmers are required to think outside the box and adjust the way they develop programs to address the problems at hand. Problem solving is an important ability for any high school student to have, and it is vital for computer programmers and the general public alike. Based on these studies it is now known that problem solving is a concept that can be learned through programming. A study done by Uysal (2014) asserted that “learners’ problem solving ability, which is also an essential skill for computer programming, can be facilitated through the integration of problem solving techniques and computer technologies" (p. 198), which could play a huge role in student achievement based on the fact that "many students lack problem solving and computational thinking skills" (Kalelioglu et al., 2014, p. 34). Students not only improve their creativity and problem solving, but also seem to enjoy doing it. As Kalelioglu and Gulbahr (2014) pointed out in their study, "it can be clearly stated that all the students in the study liked programming and wanted to improve their programming" (p. 33), which would largely carry over into the mathematics classroom and into the world of improving test scores. Kim, Chung, and Yu (2013) noticed the same desire to learn in their study, reporting that "teachers observed that participating students seemed to be interested in learning, indicating that the training program of this study can affect attitudes towards learning itself" (p. 196).

Increase Student Performance with Programming in the Classroom

Eskandari and Ebrahimi (2013) shared some valuable information from their study on classroom perspective and overall results on exams taken in the classroom. As previously mentioned, students' attitudes on learning can be impacted, and there are direct links to technology that show this happening. Two groups of students from the exact same class were taught in two slightly different environments. One classroom was taught using technology and the other was a traditional classroom without basic technologies, and the differences were staggering. "Students perceived their technology-enhanced chemistry classroom more positively than the students in the traditional chemistry classroom" (Eskandari et al., p. 167), and that perception did not just apply to the material. Students who were allowed to use technology, in a classroom where the teacher used technology, felt more connected to their classmates and instructor. This finding is so valuable because “numerous research studies have revealed that student perceptions of the classroom environment account for appreciable amount of variance in the learning outcomes” (Eskandari et al., p. 161). These perceptions produced positive academic results as well: “the students in the technology-enhanced chemistry classroom learned more and received higher scores on their exams than students in traditional chemistry classrooms” (Eskandari et al., p. 167). Although strengthening students’ test scores is always a high priority, teaching is more than just numbers, and using technology encompasses it all as Eskandari and Ebrahimi concluded their study with:

> The use of Information and Computer Technology devices such as computer and educational and multimedia software can lead to better learning and better scores in [chemistry] classes among students. In addition, the positive effects of Information Computer Technology are not limited to students’ scores. The students perceived their [chemistry] classroom more positively when their classroom was enhanced with technology… such perceptions can keep them highly motivated and satisfied (p. 167).

When students feel they are in safe environment, with teachers and classmates they can trust, they will learn more than students who do not experience these things in their classroom. Motivation and high interest are key drivers of learning which typically leads to higher grades and retention of the material. Technology is a significant component when it comes to creating this environment (Eskandari et al., p. 167). This study not only allowed students to use technology in their senior level Algebra 3-4 class, but it actually taught them to develop and create it on their own. This study aimed to expose students to careers they were interested in that they might not have even knew existed before, while teaching them to creatively problem solve in ways they had not tried before. The students took risks in their learning, accepted and learned from being wrong, and were inspired through programming, which ultimately led to a higher completion rate of homework and overall higher grades in the Algebra 3-4 course.

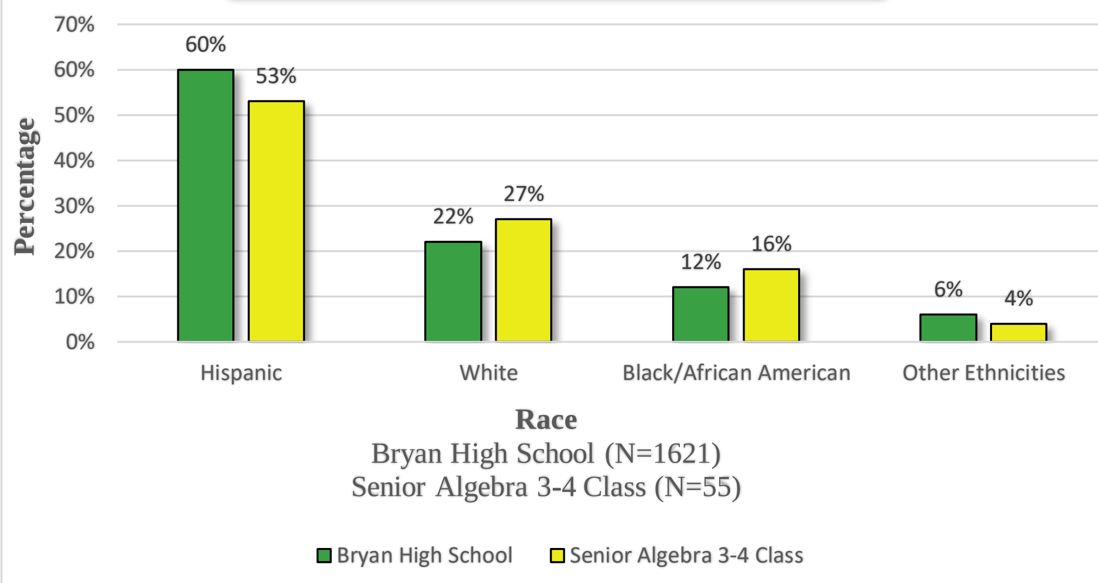
Methodology

Demographics Data and Information

This research study took place within two senior level Algebra 3-4 classrooms at Bryan High School, which is one of seven high schools in the Omaha Public Schools (OPS). OPS is the largest public school system in Nebraska. Typically seniors are placed in Algebra 3-4 if they are a year behind the general high school population in their mathematics requirements for graduation because of a previously failed math course. It should be noted that this specific senior class at Bryan High School, in general, did not meet the standards on their Nebraska State Assessment-Math (NeSA-M) that they took last year. As a whole, this class scored six percent lower than the previous class at Bryan and nearly 13 percent lower than the rest the district, and it is the only class from Bryan to score lower than the previous class in the last five years (the furthest back data is available) on the NeSA-M (Omaha Bryan High School Academic Data Department, 2015). This class being a slight outlier in comparison to previous classes could have had an impact on the results of this study.

The Nebraska Department of Education reported a student population of 52,025 in OPS for the 2014-2015 school year. Nearly three quarters of the district qualified for free and reduced price lunch last year, and Bryan High School's population had an even higher rate, at 83.2 percent qualifying for free or reduced price meals. Free and reduced price lunch participation is commonly used among the schools as an indicator of poverty. The three largest demographics at Bryan High School were Hispanic, White, and Black/African American students, who made up

Figure 1. Bryan High vs Algebra Class Demographics



60 percent, 22 percent, and 12 percent of the population, respectively. The remaining 6 percent was a combination of American Indian/Alaskan Native, Asian/Native Hawaiian or Other Pacific

The 55 students in this study had a similar demographical breakdown to the overall population of Bryan High School, shown in Figure 1, although there was a slightly lower Hispanic population, and slightly higher White and Black/African American populations in this classroom. The classroom consisted of 52.7 percent Hispanic (29 students), 27.3 percent White (15 students), 16.4 percent Black/African American (9 students), and 3.6 percent of other decent (2 students). The classroom was split in half in regards to gender, as 28 males and 27 females took part in this study.

Research Design and Data Collection Method

Preparation (December-January). A letter of consent (Appendix D) was sent home at the end of first semester in late December. The letter informed parents that the study would take place upon their child's return from winter break, that their child's identification would be kept confidential, and that they could opt out their student from the study if they so chose. No one opted out. Lesson plans were finalized at this time, based on the guide established by Codecademy (Appendix G). A pre-survey and post-survey (Appendix E) were created using the online tool, survey monkey, and tested on Ipads for use in the classroom. Also at this time, confirmation was made to ensure that all students had access to an email address, or could be given access to an email address if needed. After the instructor signed up for a free Codecademy account (only needing an email address to do so), he was then instructed to enter a class roster into Codecademy, and assign the specifically desired courses (HTML, CSS and JavaScript in this study) to each student's account. Codecademy created a unique user name for each student in order for them to track their process. The instructor set a common password for all students. It should be noted that in this study, the instructor also made two projects available to the class as an option for them to experiment with the HTML, CSS, and JavaScript that they had learned through 'Make a Website' and 'Make an Interactive Website' on Codecademy.

Introduction to the Study (End of January). The first few days of this unit involved some basic vocabulary notes (Appendix H) on HTML and CSS, and students were given a tutorial on how the basics of codecadmy.com functions. They were shown how to log in with their predetermined user name and password, and how to change it in order to personalize information once logged in. Codecademy tracked if changes were made, and passed on the information to the instructor. Students were instructed on how to save their progress, and informed they could work at their own pace from any computer that had access to the internet. Codecademy tracked every log in and completed section in the class resource section of the instructor's account, and was easily able to export to an Excels file.

Pre-Data Collection (End January – Begin February). A pre-survey (Appendix E) was administered to gauge the classes' view of math in general, if they liked it, if they thought it was important, if they talked about it with friends and family, and if they could see themselves working in a math related field after high school. Students brainstormed in small groups ways in which math could be used to create the world around them, and a discussion followed in which groups shared their thoughts. Medicine, video games, phone apps, internet, and cars were the focus of this discussion. Attendance and homework completion continued to be tracked during this time.

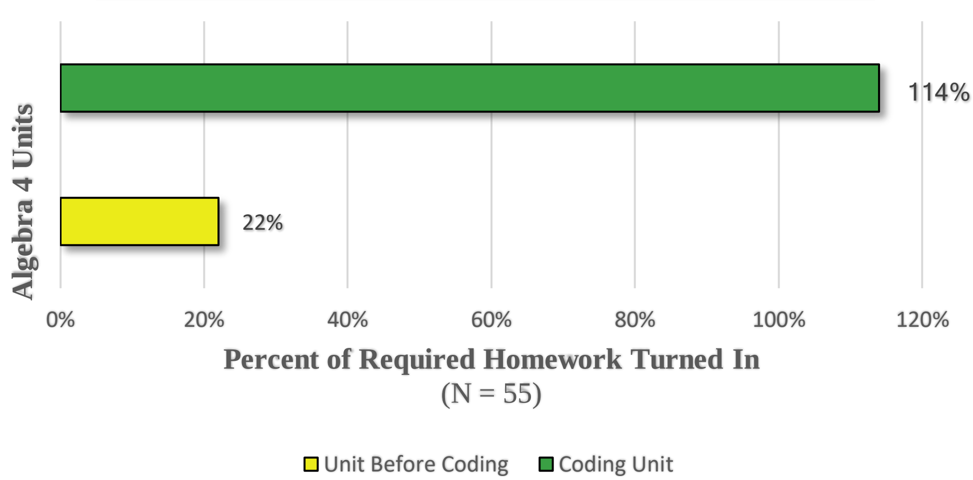
Data Collection: Start using Codecademy (February – March). Attendance and homework completion continued to be tracked. Students began working through the tutorials on Codecademy individually or in groups of no larger than three. These tutorials provided notes, examples, and asked the students to immediately apply what they had been taught. End of the day discussion took place and focused around three questions, 'What concept did I gain a deeper understanding of today?', 'What concept gave me the most trouble today?', and 'How can I build on what I learned?'. If time permitted, students who struggled with a concept that day could pair up and discuss with someone who had a better understanding of the concept. A mid-unit quiz over a few HTML and CSS concepts was given to determine retention of the information.

Post-Data Collection (Beginning of April). A post-survey (Appendix E) was administered to compare how the classes' view of math in general had changed over the course of the unit. The attendance, homework, number of students who attempted the level 4 questions on tests, and student's grades were accumulated for comparison before, during, and after the computer programming unit.

Data Analysis and Interpretation

Quantitative data was collected from the Bryan High School Senior Algebra 3-4 class and analyzed to compare several different areas. This included associating participation through homework completion, performance in the class by comparing test scores on their semester final, charting absences during each semester, and exploring a student's problem solving ability and creativity by looking at the number of higher level thinking problems solved before and after the coding unit. The results are discussed below.

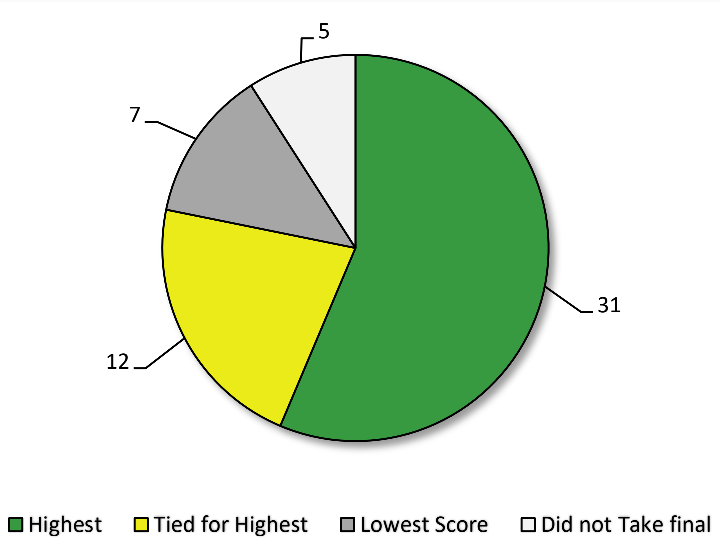
Figure 2. Senior Class Average Homework Completion Per Unit



In the section leading up to the coding unit, the senior students barely turned in but one-fifth of their assignments. Students were given time every day in class to work individually, or in small groups out of their book, and still only 22 percent of the assignments were completed. However, in the coding unit, as shown in Figure 2 above, the students could not get enough to learn. They were given the same class time to work individually, or in small groups, but as a class overall were actually requesting more assignments than were required by the instructor, and turned in 114 percent of the required work. This, nearly 420 percent, increase in homework completion exhibited a clear interest by the classes as a whole in learning how to code, and a drastic improvement in the participation within the class setting as a whole.

These senior students also seemed to retain the knowledge that they learned in the coding unit far better than any other unit as shown on their final semester scores. Each of the four

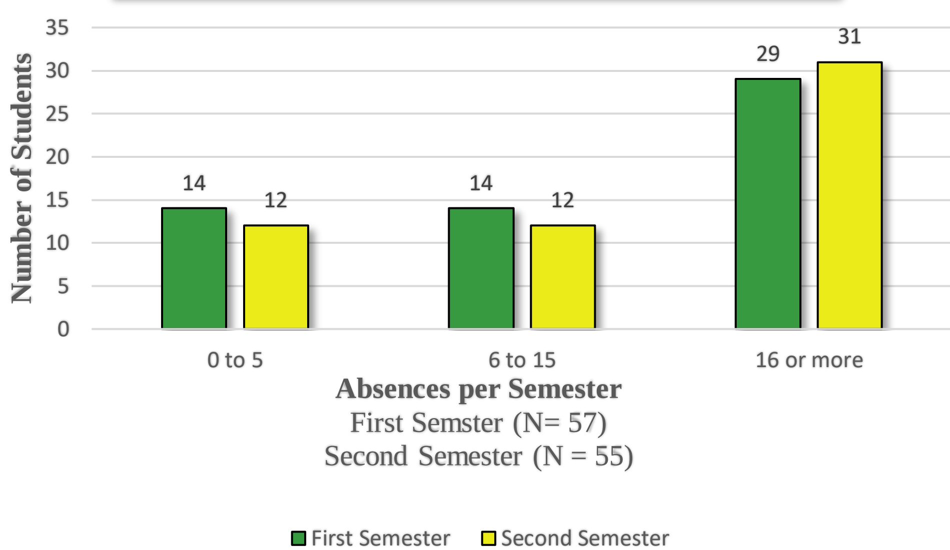
Figure 3. Coding Score Relative to Other Sections on Final Exam



second semester units was scored individually and then averaged to give the final score for the term, which allowed an analysis to be taken of the unit scores individually. Of the 55 seniors in the class, 31 scored highest on the coding unit, and 12 had a score on the coding unit that was tied with another section as their highest score. Therefore, 43 students, 78 percent of the seniors, retained the most information from the coding unit in comparison to the other three units, as shown in Figure 3. It is important to note that this unit ended in March, which means there were still two more months, in which two additional non-coding units were taught, so the high percentage of top scores cannot be attributed to the information being most recent in their minds. It should also be noted that the five students who did not take the final missed over 25 days each and were unlikely to pass any of the sections had they taken the final.

The number of absences by the senior class was the third piece of quantitative data collected during this study. It seemed that the coding unit had no effect on getting students to

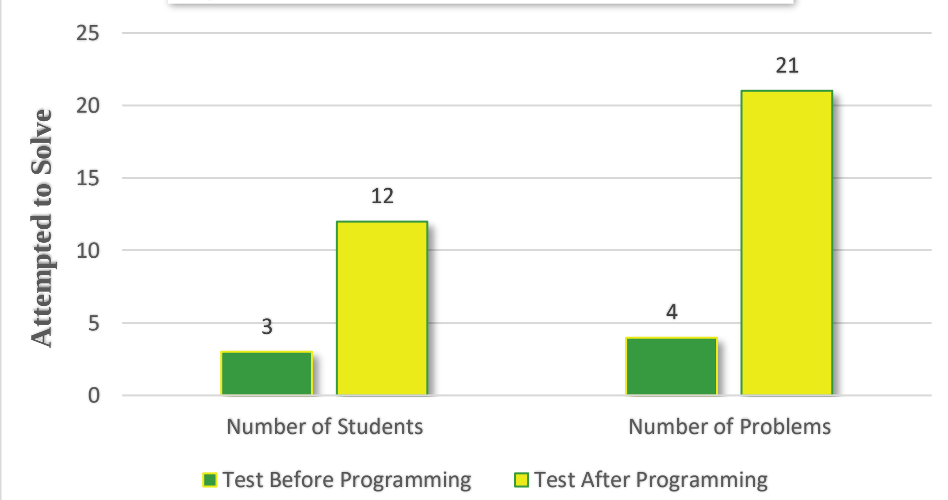
Figure 4. 1st Semester Compared to 2nd Semester Absences



attend class, which also makes the amount of completed homework for the coding unit all the more interesting. Over 50 percent, 31 students missed sixteen or more classes during second semester, and a total of 43 students missed at least six days throughout the course of the semester in which the study took place, as shown in Figure 4. Although extremely high, the number of absences was consistent with that of first semester. The average senior in this Algebra 3-4 class missed just over eleven days in both first and second semester, which is statistically significant enough to say that coding had no impact on a student's willingness to come to class. However, the sheer volume in which homework was turned in, while students missed class at the same rate as before, speaks to the increased participation and buy in to the coding unit in and out of the classroom.

On a typical Algebra 3-4 test there are one or two deeper level, problem solving and critical thinking (Level 4) type problems for students to solve. Most students in this senior level class wouldn't even attempt to solve these problems before the coding unit, only three of the 55

Figure 5. Level 4 Questions Attempted by Students



combined to attempt four Level 4 problems. However, after being encouraged to take chances at being wrong in their learning, learning from their mistakes, and thinking critically in the coding units, these same seniors had 12 different students attempt a combined 21 Level 4 problems on the test immediately following the coding unit. This growth is significant, as shown in Figure 5, and can be attributed to the emphasis placed on creativity and problem solving through challenges in the coding unit. If a student was stuck in the coding unit, they had to connect dots from previous knowledge, brainstorm ways that might work, fail and try again, and realize that there is more than one correct way to solve a single problem.

There was also a pre-survey and post-survey that collected both quantitative and qualitative responses for additional analysis. There were some expected responses with the qualitative data. For instance the most common response before the unit when asked for the student's thoughts about learning to code was "I won't be able to understand it because it is too hard." In contrast, after the unit a common response was "It was fun to learn something I can use outside of school, and it wasn't nearly as hard as I thought it would be."

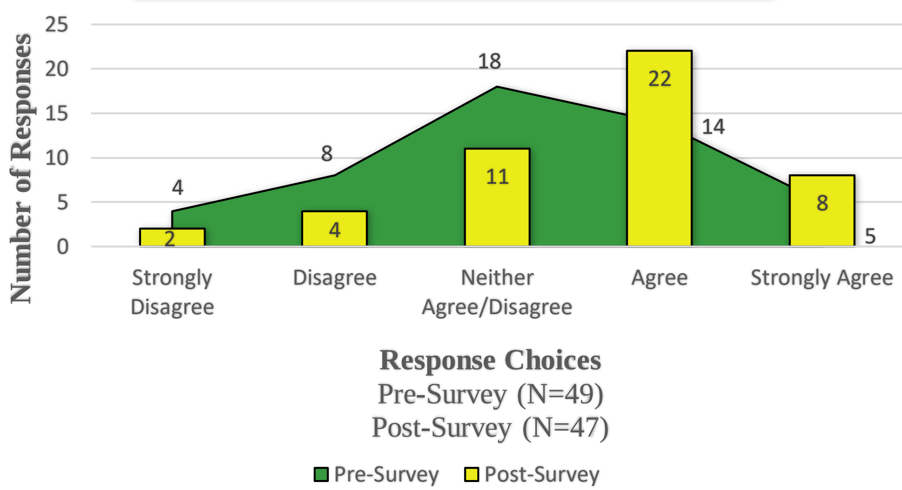
However, there were also some peculiar findings, especially when asked if students thought they would turn in more homework when working from the book (typical algebra problems), or when they are working on computers and coding. The majority, responded before the unit by saying "I don't do homework, so it won't really matter whether the assignments come from a book or computer." The peculiar finding comes with the exact same question, but in the post-survey, when students answered most commonly that they "turned in the same amount of homework from the book as they did from the computer" when in fact they turned in 418 percent more homework in the coding unit. This can be attributed to the fact that students did not feel that the coding unit was homework, but rather an activity that allowed them to learn freely.

When asked to name the part of the coding unit that they enjoyed the least and the part that they enjoyed the most the concept of working freely became obvious. The theme from their answers was that students typically did not like that they were required to think critically to solve problems because they were used to memorizing concepts and reciting; thinking was hard when they hadn't done it previously. However, this lead nicely into their favorite part which was that they were allowed to work at their own pace, which for most meant ahead of the class, and solve problems in a way that made sense to them instead of following a specific set of steps.

A quantitative piece of data from the same pre-survey and post-survey, that supported the qualitative data above, asked the students to use a rating scale to answer ten questions about math in general (Appendix E). There were four questions that showed the largest amount of change from the pre-survey to the post-survey, and those will be discussed here, however full survey results can be found in Appendix F.

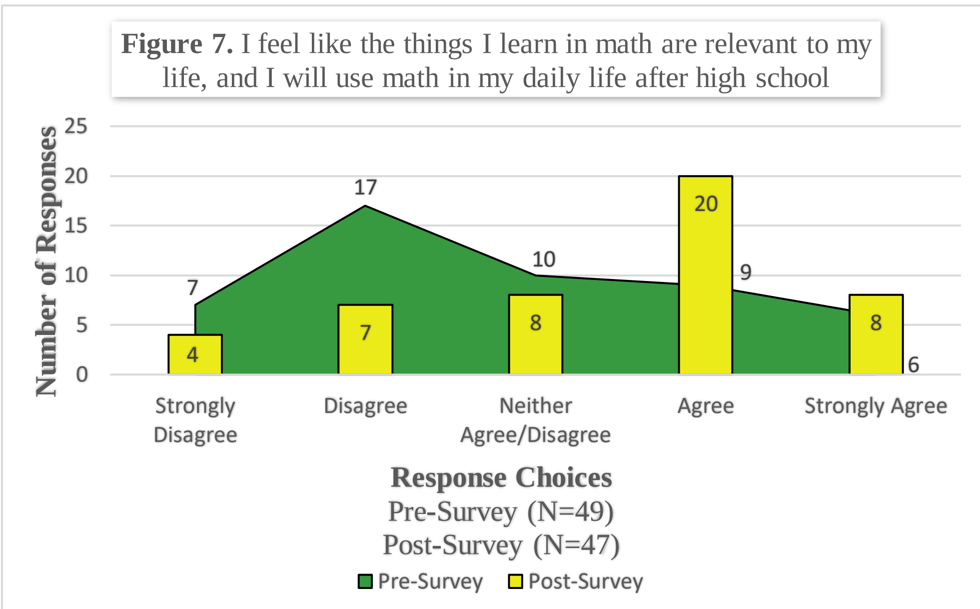
The first question with significant change was "I look forward to coming to this math class" (Figure 6). Although, few in the pre-survey strongly disagreed, or disagreed, that number

Figure 6. I look forward to coming to this math class



still decreased considerably following the unit on coding. The group that neither agreed nor disagreed dropped from 18 to 11 and most tended to agree, up from 14 to 22, or strongly agreed, up from five to eight, that they liked coming to this math class immediately following the unit on coding.

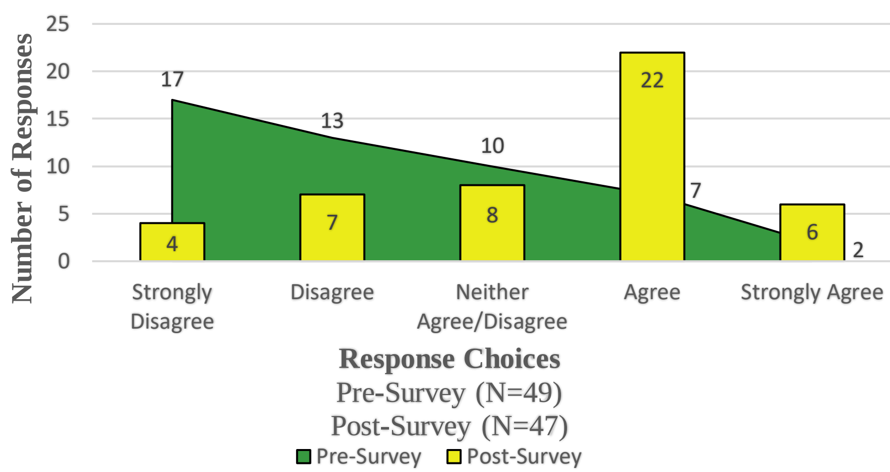
The second question that showed a growth was “I feel that the things I learn in math are relevant to my life, and I will use them in my daily life after high school” (Figure 7). The amount of students that agreed or strongly agreed at the conclusion of the coding unit, 28, is four more than who originally disagreed or strongly disagreed, 24. Simply put, coding made math meaningful



to the students, which directly related to the students looking forward to coming to class. When students felt that the information they were learning was relevant they were more likely to enjoy coming to the class. It would be of value to see if this connection between the two, if given enough time, could help address the much needed issue in regard to attendance.

Perhaps the biggest turn around between the pre-survey and post-survey was centered on the question “I talk to my family and/or friends about what I learn in math” (Figure 8). In the pre-survey this was the question that had the most amount of students, 61 percent, who either strongly disagree, or disagree with the statement. While only nine students total could either strongly agree, or agree that they talked to their family about what they learned. At the conclusion, nearly the exact opposite was true, with 59 percent stating that they strongly agreed

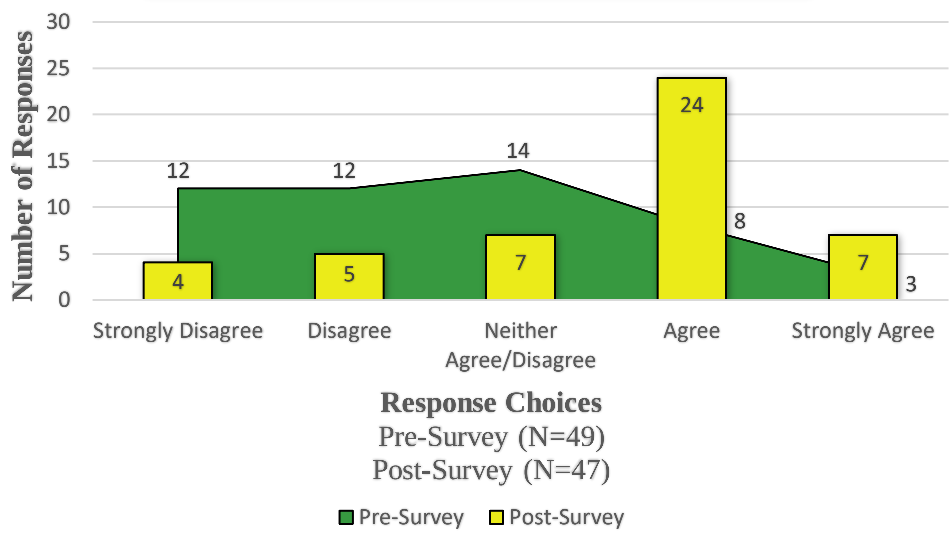
Figure 8. I talk to my family and/or friends about what I learn in math.



or agreed that they discussed what they were learning math class outside of the classroom, and the amount who disagreed in some form had dropped from 30 students to 11. This clearly shows that the coding unit was something the students were excited about, and wanted to share with their friends and family. When presented with information that they felt was relevant, the students wanted to tell others about it. It would be interesting to see the relationships between a student talking about what they've learned in class with family and friends and their score on that unit. It seems with the little research on these two specific concepts from this study, that the association seems quite strong, that those who are excited and have discussion about what they are learning score higher when tested over that subject.

Finally, the students were asked if they learned what they were supposed to learn in the unit by answering on the rating scale, “I can write the code that will create a website” (Figure 9). Before the coding unit only 11 of the 49 students surveyed felt they could write the code to make a website, and by the end that number had grown to 31. There were only nine students at the end

Figure 9. I can write the code that will create a website



of the unit that still felt that they could not write the code to make a website. Considering, as mentioned earlier, that there were 31 students who missed 16 or more classes the semester of the coding unit that number is fairly low.

The remaining questions in the pre-survey and post-survey showed little to no significant change from start to end. One question that did not show a net movement from strongly disagree or disagree to strongly agree or agree, but did show a bit of a trend was the question, “I am interested in studying a math related subject in college, or finding a career that involves math” (Appendix F). There were six fewer students who strongly disagreed with that statement, down to ten from 16, during the post-survey, but those six answers were evenly dispersed amongst the other choices resulting in a little movement toward strongly agreed or agreed, but not enough to be considered significant.

Results of the study were shared with and presented to a select group of administration, counselors, and teachers at Bryan High School, as well as to the Midland University Review Committee.

Conclusion

The intent of this study was to evaluate the impact of introducing basic computer programing skills to the senior math classes in a Title I school. The study was designed to gauge participation and performance in the math class while also looking for improvement in problem solving skills and creativity. The focus of the study was centered on the specific question: Does implementing basic computer programming skills into the senior level Algebra 3-4 curriculum increase creativity and problem solving ability while also improving participation and performance of the students in the class?

It is important to recognize that although the seniors in Algebra 3-4 course are below grade level, it is not typically because they are incapable of doing the math, but rather several other factors that caused them to fall behind, such as a feeling of disconnect from the relevance of the material being taught, outside jobs to support their family, and lack of motivation and attendance issues. Students in the study expressed concern that “math class doesn’t pay the bills” and that they would never “use a logarithm again, so why now?” and their “responsibilities outside school take priority.”

The majority of students in this class do just enough to get their credit so they can graduate high school. However, when introduced to computer programming, these seniors went above and beyond their typical effort. They requested additional material, worked on it in and outside of class, and completed it at a 420 percent increase over other assignments. Interestingly, though, the study group failed to recognize this, reporting that they felt they had completed the same amount of homework as normal. Several students indicated that they didn't feel that the computer programming unit was work, but rather "a lot of fun" and "a real life opportunity to learn something that I could use after high school." The seniors participated at a significantly higher rate than during a typical unit, and this led directly to higher test scores. Even though most didn't feel that they had done more, the response from the post-survey indicated that students could see a purpose in their learning, and that was the driving force behind the drastic increase.

Their problem solving skills for more challenging problems also increased by a large margin, and students were more willing to take risks in their learning because they better understood there is more than one way to correctly obtain an answer. Test problems that largely went unanswered before were being attempted, and although they were far from all correct, it was a step in the right direction. The seniors were more willing to apply the knowledge they had previously learned throughout a chapter, and attempt to solve a more abstract problem. This is a learned behavior from working with computer programming. One student expressed her biggest frustration became her favorite part when she said, "I hated that programming made me think about what I wanted to do, when in math I'm used to simply being told what steps to follow and completing the problem without knowing why I did what I did, but by the end that was my favorite part about programming because I only had to learn a concept one time and then manipulate it to do what I wanted it to do."

The computer programming unit had students opening up about their learning, and they discussed it outside of the classroom with friends and family at a greater rate. Their participation carried outside of the classroom and they made connections to the learning in their own lives. The students were allowed to be creative, and take the general computer programming lessons and design then create their websites about a topic of their choice. The fact students made the learning their own, took ownership, and were proud to connect it to their lives were huge factors in why the test scores were so high for this unit.

It was interesting to see that attendance was not affected at all during the unit, and the absences remained high. Nearly all students in the pre-survey mentioned math in general as being a subject they didn't enjoy because of a disconnect from the content, they would never use it again, and it was too hard, which understandably would lead to not coming to class. However, in the coding unit, and post-survey, students expressed almost unanimously how relevant they thought this was to their lives, how much easier it was than they imagined, and they could see themselves using it in their daily lives. These positive reports came in high numbers, despite that the students were still missing class at a high rate. It seems that the overwhelming reason for missing classes isn't actually related to the math itself, but to other outside factors like work, raising their own kids, and paying the bills. When in class the students enjoyed learning the computer programming materials, completed their assignments at an extremely high rate, and excelled on the tests, but the larger issue was getting them to class.

Although this population was a very good representation of the overall school demographics it was limited by a group size of 55 senior students, who all have similar mathematical backgrounds. To get a better representation in order to make a more accurate generalization, it would be beneficial to do a similar study in honors, academic, and special education classrooms of various grade levels. These students, although they had to the ability to opt out, did not pick to learn about computer programming, and there was still a lot of noticeable success in increasing participation, grades, and problem solving skills by using creativity and logic that carried over into units after we were done programming. It would be interesting to see how this would apply to an elective class in which the students openly chose to be a part of the class, and what could be accomplished in that setting. The time frame for the study was only over the course of a quarter. It would be recommended that this study be done over a longer research period to delve deeper into understanding the reason behind the increased participation, and to see if students remained motivated, or if their drive would wear off over time as the unit became more challenging. More research could be done to determine if learning to code positively impacts a student's view on what they want to study in college, because some students mentioned that they were now interested in pursuing computer programming in college or in the work force, and it would be valuable to follow up and see if this actually took place.

Action Plan

At the conclusion of the action research study, several additional questions presented themselves for further study. Some of these questions include: How can educators, within their strict limitations of the given district curriculum, transform the necessary material into more relevant connections for students who have a goal other than attending college after high school? Do any partnerships exist in the community, or could some be created to allow students to learn computer programming for high school credit by creating useful websites and apps for local businesses while earning a small salary that would allow them to stay in school and help support their families? Is there any other way to use technology to increase student attendance? With a sudden push by President Obama and others to make computer science and computer programming a core subject in the public schools system, where are we going to find the educators qualified to teach coding, and where is the funding going to come from for the needed technology? Does the overall way in which we teach our students in math, and other subjects, need to be reevaluated since it has remained virtually unchanged for decades in order to better reach the current generation of students?

It is clear that incorporating computer programing greatly increased the student involvement and participation in this math class. Even when students were missing class at a very high rate, although consistent with non-computer programming units, students were far more likely to stay on track, or even work ahead when learning to code than working through a standard unit. The fact that coding made school relevant to the students was the overwhelming response when they were asked what they enjoyed most about the unit, and this relevance is a vital component that should go into curriculum for both programing and non-programming units going forward.

The teacher researcher plans to propose an elective course, and curriculum centered on the findings of this study that would allow students with an interest to go deeper into the concepts of computer programming. In the meantime, the teacher researcher will continue to implement computer programming and its concepts into the core math curriculum when appropriate. An example of this blended approach would tie key concepts from the current curriculum such as plotting points on an x and y axis to the pixels on the computer screen, and how positive and negative numbers can not only move a point on graph paper, but noting that is performs the same movement on a website. These are the types of connections that will continue to build problem solving skills and increase creativity in the math classroom. When given the chance to learn freely, within a given structure, students have shown that they will push the limits of their education and take ownership of their work. The largest take away for the teacher researcher that can be applied to all classroom settings is that the students are willing to go above and beyond the required work if they can clearly see the purpose in what they are doing, if they are not being punished for making mistakes, but rather encouraged for taking a chance in applying what they already learned, and if they are allowed to make the learning their own. These concepts can be applied in non-programming units, and would help to “make school relevant” for the students.

If the study were longer in duration, it would have been interesting to see how students responded to diving deeper into JavaScript, a language that is more challenging that HTML and CSS, but offers more opportunities to create interactive and responsive websites. The process in which this coding program was implemented was efficient, but the researcher would be interested to see the impact spreading the lessons out over the course of the year might have on participation, performance and attendance in the class. Perhaps, if students were exposed to computer programming earlier in the year, and it was consistently sprinkled into units at appropriate times, then the positive effects it created in regards to problem solving and creativity could potentially root deeper within the students mind set about math and school in general.

References

Adams, S. (2015, August 18). The fastest-growing states for tech jobs in 2015. *Forbes*, p. 1. p. 2. Retrieved from http://www.forbes.com/sites/susanadams/2015/08/18/the-fastest-growing-states-for-tech-jobs-in-2015/

Andreessen, M. (2011, August 20). Why software is eating the world. *The Wall Street Journal,* p. 2. Retrieved from http://www.wsj.com

Codecademy (2015). Teacher resources. *Codecademy.* Retrieved from http://www.codecademy .com/schools/curriculum

Eskandari, Z., Ebrahimi, N. (2013). Effects of computer and multimedia software on Iranian high school students' learning and perceptions of chemistry classroom enviroments. *International Joural of Technology in Teaching and Learning 9(2),* 159-172.

Hussey, B. (2016). Ultimate web designer & developer course [Video file]. Retrieved from https://www.udemy.com/web-developer-course/learn/v4/content

Kalelioglu, F., Gulbahar, Y. (2014). The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education. 13(1),* 33-50.

Kim, S., Chung, K., Yu, H. (2013). Enhancing digital fluency through a training program for creative problem solving using computer programming. *The Journal of Creative Behavior. 47(3),* 171-199.

Koebler, J. (2012, January 25). Obama pushes STEM in State of the Union. *U.S. News.* Retrieved from http://www.usnews.com/news/blogs/stem-education/2012/01/25/obama-pushes-stem-in-state-of-the-union

Larson, Q. (2016, May 11). Let's explore the universe of programming resources together. Retrieved from https://medium.freecodecamp.com/lets-explore-the-universe-of-programming-resources-together-77ff382a4c52#.oc9dadp2i

Layton, L. (2013, December 12). Week-long 'Hour of Code' campaign lures millions of U.S. students to computer coding. *The Washington Post.* Retrieved from https://www.washingtonpost.com/local/education/hour-of-code-campaign-gets-wild-response-from-kids-trying-out-computer-science/2013/12/10/7b9913ec-61db-11e3-bf45-61f69f54fc5f_story.html

Leszczynski, K. (2014, February 2). Fastest-growing jobs in Nebraska. *World-Herald,* p. 1. p. 2. Retrieved from http://omaha.com/

Meloni, J. (2015). *Sams teach yourself HTML, CSS and Javascript all in one, Second Edition.* Indianapolis, Indiana: Pearson Education.

Moore, L. (2016, March 4). Computer science is the key to America's skill crisis. *Tech Crunch* Retrieved from http://techcrunch.com/2016/03/04/computer-science-is-the-key-to-americas-skills-crisis/

Mills, C. (2014, March 14). *The web standards model - HTML CSS and JavaScript.* Retrieved from http://www.w3.org/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript/

Omaha Bryan High School Academic Data Department. (2015). Student achievement reports. Omaha: Omaha Bryan High School.

Omaha Public Schools Research Department. (2015). Bryan High School Enrollment and Achievement School Data Book. Omaha: Omaha Public Schools.

Restuccia, D. (2016, April 13). Why should students learn to code? Opportunity. *Burning Glass.* Retrieved from http://burning-glass.com/why-should-students-learn-to-code-opportunity/

Uysal, M. (2014). Improving first computer programming experiences: The case of adapting a web-supported and well-structured problem-solving method to a traditional course. *Contemporary Educational Technology 5(3)*. 198-217.

Appendix A

OPS Research Request to Conduct Research Online Application Form

Project Title (Research Question): Does implementing basic computer programming into the senior level Algebra 3-4 curriculum increase student participation and improve grades in their math class?

Have you, or will you, complete a Request for Student Data in OPS? No, already available at Bryan High School. Contact Information: Email: charles.cuddy@ops.org Phone: 402-557-3100

Requester's Name (Include Title): Charles Cuddy, Secondary Math Teacher at Omaha Bryan High School

Address: 4700 Giles Road Omaha, NE 68157

Telephone: 402-557-3100 (work) 402-309-0853 (personal cell)

Fax: 402-557-3139 Email Address: charles.cuddy@ops.org

Project Information:

Sponsoring Organization(s): Midland University and Omaha Public Schools

Funding Source(s): N/A

Beginning Date: January 2016

Completion Date: April 2016

Describe the purpose of the research:

The purpose of this research is to determine if the use of computer programming in senior level algebra will increase the student's attitude toward math and its applications while also raising grades and participation.

Why are you conducting this research:

I am conducting this action research project as part of the Midland University's program requirements to attain a Master's degree in the Leadership in Teaching and Learning Program.

Describe the benefits to the district, students, and/or staff for participating:

All the data that is collected and analyzed from this study can be used in the future throughout the district where it applies. The students will be directly exposed to an application of math that they might not otherwise get in the typical classroom setting, and it is in a career field that is in high demand.

Methodology:

List your research questions and/or hypotheses being investigated:

Does implementing basic computer programming skills into the senior level Algebra 3-4 curriculum improve participation and performance of the students in the class?

Describe your requested participants including number, type (e.g. students, staff, parents), grade levels, building names, etc.

The participants will consist of 55-65 students all of which are in 12th grade and enrolled in one of two Algebra 3-4 courses being offered at Omaha Bryan High School in the spring of 2016.

Describe your procedures and methodology.

If you are administering a survey or instrument, please include the name of each measure. There will be a pre and post survey entitled “Value of Math” and “Current Knowledge”. The pre surveys will be administered prior to the six week study, and compared to the post survey which will be given at the completion of the study. During the 8-week study period students will be instructed in computer programming skills 1-2 period(s) per week. The online program www.codecademy.com will provide the curriculum for this unit. This project has been discussed with the Curriculum Specialist at Bryan High School and approved for implementation.

Describe the data required and how you will collect it:

Data will be collected from the pre and post surveys to determine if students perception and/or knowledge of math was impacted at all by the unit on computer programming.

Will you need data from OPS?

At this time I do not anticipate needing any data from OPS.

Describe your proposed analyses:

The data that is collected will be analyzed to determine if implementing immediate applications to the math curriculum, in this case through technology and programming, improved the seniors at Omaha Bryan perception of the importance of math in their lives leading to an improvement in their grades.

Describe the responsibilities of OPS students including the amount of time required to complete instruments or interviews.

The OPS students will be required to take a 5 minutes pre and post study survey. The students will also be working in the computer labs 1-2 days per week over the course of the study.

Describe the responsibilities of OPS teachers / staff / principals including the amount of time required to conduct interventions, distribute instruments, conduct interviews, etc.

The other OPS teachers, staff and principals will not be impacted by this study, other than the fact that the computer lab will be reserved for each of these two classes during the study.

Describe the potential interference of instructional time:

This will not have any interference with instructional time as it will take place during the NeSA-Math review that is already part of the build in to the junior level curriculum for Algebra 3-4, but since these students are seniors they will not be participating in the NeSA-M review and therefore have additional time to fill.

Will you need access to classrooms or special facilities? If so, how many and what type?

We will need access to the computer lab or the mobile lab for up to 30 students at a time, on average of once or twice a week for the six weeks.

How do you plan to use your results/outcomes (e.g. conference presentation, publication?

I will present my results to other senior level teachers at Omaha Bryan High School, the other math teachers in my building, and to the Midland Review Committee.

Appendix B

Omaha Public Schools (OPS) Request to Conduct Research Letter of Approval

OMAHA PUBLIC SCHOOLS
DIVISION OF RESEARCH
3215 CUMING STREET OMAHA, NEBRASKA 68131-2024 (402)557-2080 FAX (402)557-2049

November 16, 2015

Bryan High School
Charles Cuddy
4701 Giles Rd
Omaha NE 68157-2700

Re: Action Research Project: Does implementing basic computer programming into the senior level Algebra 3-4 curriculum increase student participation and improve grades in their math class? (2015-2016)

Dear Charles Cuddy,

Your request to conduct an action research project as a part of a course requirement through processes such as the examination and/or collection of information from surveys, direct observation, or individual interviews has been reviewed.

We believe your project has merit and permission is granted for you to proceed under the following conditions:

- Midland University approves of your project.
- The principal of your building agrees to your project and use of building resources (e.g., computers, computer lab, gymnasium, etc.).
- You will notify parents/guardians of students of the nature of your study and direct them to contact you if they have questions regarding their child's participation (if applicable).
- Parents of students in the study must consent to their child's participation in the study (if applicable)
- The consent form will be translated into Spanish (if applicable).
- Students who do not have parent consent/permission should not participate (if applicable).
- Staff/teachers agree to your study (if applicable).
- In the reporting of the data/results, teachers, students, schools, and district will not be personally identifiable. The identity of students should not be revealed.
- You will be willing to share results of your study with your principal/supervisor.
- The collected data will be kept in a locked cabinet or under password protection.
- The findings of the study must be shared with the Research Review Committee for review before being presented or released for publication.
- It is important to note that the Research Review Committee sees this as a limited study of this topic, and that results may not be generalizable.

Thank you for your interest and support in meeting the needs of our students.

Best wishes,

Sincerely,

Janet Zahm

Janet Zahm
Instructional Research Administrator
JZ/bf

pc: Robert Aranda, Carla Noerrlinger

Appendix C

Midland University Application for Action Research Approval

Rev. November 6, 2014

MIDLAND UNIVERSITY

Masters of Education in Leadership in Teaching And Learning
Application for Action Research

Applicant's Name: Charles Cuddy
Address: 4700 Giles Road Omaha, NE 68157
Phone number: 402-557-3100
School: Omaha Bryan High School
Advisor: Carla Noerrlinger
Research Support Team Members:

Project Title: Does implementing basic computer programming skills into the senior level Algebra 3-4 curriculum improve participation and performance of the students in the class?

Purpose:
The purpose of this research is to determine if the use of computer programming in senior level algebra will increase the student's attitude toward math and its applications while also raising grades and participation.

Potential Benefit to my Teaching (or your current role):
The students will be directly exposed to an application of math that they are currently learning about and might not otherwise get in the typical classroom setting. The study focuses on a career field that is in high demand, and an area that is on the rise. Increasing student participation and homework completion will lead to higher grades. Indirectly students will also learn valuable skills such as logical thinking, problem solving and will be allowed to use their creativity.

Data Collection Methods:
Data will be collected from the pre and post surveys to determine if students perception and/or knowledge of math was impacted at all by the unit on computer programming.

Describe how confidentiality will be assured:
Surveys are anonymous. A Passive Consent Letter was provided to parents allowing them to "opt out" their student from the study's findings.

Rev. November 6, 2014

9/9/15

1. Applicant's Signature and date

Carla Noerrlinger 09/23/15

2. Advisor OR EDU 697 Instructor Signature and date

9/2/15

3. Principal's Signature and date

4. Letter of District Approval Attached (date) 11/16/15

5. Approved by Midland University Teacher Education Committee: (date) Sue J Evanich, 1/14/16

Please obtain the signatures in the order listed above. After you have District approval send the signed MU Application to Evanich@midlandu.edu for the final TEC approval. Sue will email you once TEC has approved your proposal.

Appendix D

Action Research Consent Form

Dear Parent/Guardian:

I am currently in a graduate program in which we are conducting research. My Action Research Project is entitled: Does implementing computer programming into the senior level Algebra 3-4 curriculum increase student participation and grades in their math classes? The reason that I chose this research topic is because I want to try and find an alternative way to expose the student, your child, to math in different fields. I will be looking to see if students develop skills will prepare them for their next educational experience, or open up the possibilities for a career. I also want to gain insights regarding improvements that could be implemented to better serve our students' needs. Our common goal is that your child/my student can experience academic success, stay on track academically, and earn their math credit.

Please be assured that I will not be using any student names, or other identifying information, when reporting data and findings for my study. My data will be collected through student surveys, credit-completion data, and grade comparisons. Student confidentiality is of the utmost importance to me, and to the district.

This form only needs to be signed and returned if you do NOT want your child to participate in my research project.

Please Note: Your child will still receive regular classroom instruction regardless of his/her participation in the research project.

Sincerely,

Charles E. Cuddy

Questions or concerns may be directed to the teacher/researcher:

Name: Charles Cuddy

Job Title: Secondary Math Instructor, Omaha Public Schools

University: Midland University

Phone Number: 402-557-3100

Email Address: charles.cuddy@ops.org

________I DO NOT want my child to participate in this research project.

Child's Name____

Parent/Guardian Signature____

Date_________________________________

Return to teacher, by January 4th, 2016, only if you

DO NOT want your child to participate in the research project.

Appendix E

Pre-Survey (used www.surveymonkey.com) and Post-Survey

Pre-Survey

Please pick one of the following that best describes you feel about each question.

Strongly Disagree, Disagree, Neither Agree/Disagree, Agree, Strongly Agree

1. **I like Math**
2. **I am good at Math**
3. **I regularly do my homework for this math class**
4. **I talk to my family and/or friends about things I learn in math.**
5. **I can teach someone else what I've learned in math**
6. **I think learning about math is important**
7. **I look forward to coming to this math class**
8. **I feel like the things I learn in math are relevant to my life, and I will use math in my daily life after high school**
9. **I am interested in studying a math related subject in college, or finding a career that involves math**
10. **I can write the code that will create a website**

Post-Survey

Please pick one of the following that best describes you feel about each question.

Strongly Disagree, Disagree, Neither Agree/Disagree, Agree, Strongly Agree

1. **I like Math**
2. **I am good at Math**
3. **I regularly do my homework for this math class**
4. **I talk to my family and/or friends about things I learn in math.**
5. **I can teach someone else what I've learned in math**
6. **I think learning about math is important**
7. **I look forward to coming to this math class**
8. **I feel like the things I learn in math are relevant to my life, and I will use math in my daily life after high school**
9. **I am interested in studying a math related subject in college, or finding a career that involves math**
10. **I can write the code that will create a website**

Appendix F

Pre Survey and Post Survey Results

(PRE: N=49 POST: N= 47)

	Pre Survey	**Post Survey**	**Difference**
I like Math			
Strongly Disagree:	7	7	0
Disagree:	10	5	-5
Neither Agree/Disagree:	15	13	-2
Agree:	14	19	+5
Strongly Agree:	3	3	0
I am good at Math			
Strongly Disagree:	6	6	0
Disagree:	10	10	0
Neither Agree/Disagree:	21	16	-5
Agree:	10	12	+2
Strongly Agree:	2	3	+1
I regularly do my homework for this math class			
Strongly Disagree:	8	4	-4
Disagree:	11	6	-5
Neither Agree/Disagree:	12	16	+4
Agree:	15	18	+3
Strongly Agree:	3	3	0

I talk to my family and/or friends about things I learn in math.

Strongly Disagree:	16	7	-9
Disagree:	7	7	0
Neither Agree/Disagree:	14	6	-8
Agree:	10	22	+12
Strongly Agree:	2	5	+3

I can teach someone else what I've learned in math

Strongly Disagree:	7	2	-5
Disagree:	8	6	-2
Neither Agree/Disagree:	10	10	0
Agree:	21	22	+1
Strongly Agree:	3	7	+4

I think learning about math is important

Strongly Disagree:	5	2	-3
Disagree:	2	4	+2
Neither Agree/Disagree:	11	8	-3
Agree:	25	21	-4
Strongly Agree:	6	12	+6

I look forward to coming to this math class

Strongly Disagree:	4	2	-2
Disagree:	8	4	-4
Neither Agree/Disagree:	18	11	-7
Agree:	14	22	+8
Strongly Agree:	5	8	+3

I feel like the things I learn in math are relevant to my life, and I will use math in my daily life after high school

Strongly Disagree:	7	4	-3
Disagree:	17	7	-10
Neither Agree/Disagree:	10	8	-2
Agree:	9	20	+11
Strongly Agree:	6	8	+2

I am interested in studying a math related subject in college, or finding a career that involves math

Strongly Disagree:	16	10	-6
Disagree:	13	14	+1
Neither Agree/Disagree:	11	12	+1
Agree:	7	7	0
Strongly Agree:	2	4	+2

I can write the code that will create a website

Strongly Disagree:	12	4	-8
Disagree:	12	5	-7
Neither Agree/Disagree:	14	7	-7
Agree:	8	24	+16
Strongly Agree:	3	7	+4

Appendix G

Codecademy Lesson Guides

UNIT OVERVIEW: HTML/CSS Part 1

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
1	1. HTML Basics	1 - 8	• Understand what HTML and CSS are used for • Define some basic HTML terminology • Explain and implement <head><title> <body> and <p> tags • Create all types of headings <h1> to <h6>	L1 algorithms L1 programming L1 hardware AL	20
2	1. HTML Basics	9 - 14	• Demonstrate knowledge of tags so far including <title>, <h3> and <p> • Define hyperlinks, explain their use and create a link using <a> • Understand and create an image tag <img> • Construct a linked image	L1/2 communication AL EV	20
3	2. Build Your Own Webpage	1 - 6	• Summative assessment of learning from HTML lessons 1 & 2 • Create a webpage with an HTML frame, including a header, paragraphs, images and links in either images or text	L2 communication L2/3 information AB AL EV	20
4	3. HTML Basics II	1 - 6	• Recognise HTML can be used to create lists • Recognise indentation and explain why it is used • Describe and create ordered and unordered lists • Demonstrate how to use nesting with lists	L1 algorithms L1 programming AL	20
5	3. HTML Basics II	7 - 16	• Describe what inline CSS is • Use style attributes (including font-color, font-family, text-align) • Demonstrate how to bold and italicise text		20
6	4. Social Networking Profile	1 - 7	• Summative assessment of learning from HTML lessons 4 & 5 • Create a 'social networking profile', including lists of their interests and styling"	L5 algorithms AB AL	30
7	5. HTML Basics III	1 - 5	• Recognise structural tags <table>, <div> and <span> • Create a table with rows and columns and explain why we use them • Construct a table with multiple rows and columns"	L2 data AB DE	20
8	5. HTML Basics III	6 - 15	• Design table formatting using style attributes • Illustrate how to use a <div> tag and use it to create a link • Construct a paragraph with selective styling using the <span> tag"		25
9	6. Clickable Photo Page	1 - 7	• Summative assessment of learning from HTML lessons 8 & 9 • Create a 'clickable photo page': a table with multiple rows and columns containing images that link to external websites	L2 communication AL EV	30

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

UNIT OVERVIEW: HTML/CSS Part 2

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
10	7. CSS an Overview	1 - 12	• Explain what CSS is and the reason it is seperate from HTML • Create a link to a CSS stylesheet • Describe 'selectors', 'properties' and 'values' and implement property-values • Illustrate CSS syntax and comments	L1 hardware L1 programming L4 algorithms AB AL	25
11	7. CSS an Overview	13 - 21	• Explain why hexadecimal values are used in CSS • Explain why 'em' units are needed and the need for default fonts as backups • Create backgrounds and borders and style them		20
12	7. CSS an Overview	22 - 26	• Plenary: Build a basic HTML index page and linked CSS stylesheet	L3 I.T. AB	20
13	8. Design a Button for Your Website	1 - 6	• Plenary: Build a button by styling divs and links		15
14	9. CSS Selectors	1 - 9	• Describe branching, children, parents and siblings • Demonstrate and understand how to format nested selectors and directly nested selectors • Explain which selectors will override others	L3 data AL	25
15	9. CSS Selectors	10 - 18	• Compare, Create and Implement Classes and IDs • Recognise and implement pseudo-class selectors for links (link, visited, hover) • Understand and use a 'first-child' and 'Nth-child' pseudo-class selectors	L3 data AL	30
16	9. CSS Selectors	19 - 23	• Plenary: Construct HTML document and CSS stylesheet with selectors		15
17	10. Sorting Your Friends	1 - 8	• Plenary: Synthesising Module 9: CSS Selectors	L3 algorithms AB	40
18	11. CSS Positioning	1 - 12	• Explain the box model and its importance in positioning • Demonstrate and understand the main display properties and their differences • Explain and Implement margins, borders and padding • Describe how negative values impact the positioning of an element	L4 algorithms AB AL	25

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

UNIT OVERVIEW: HTML/CSS Part 3

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
19	12. CSS Positioning	13 - 25	• Explain and demonstrate the float and clear properties • Explain static, absolute, relative and fixed positioning • Plenary: Compose a webpage using CSS positioning	L5 communication AL AB	35
20	13. Build a Resume	1 - 8	• Plenary: Synthesising Module 11: CSS Selectors	L5 algorithms L5 communication L5 programming AL AB	60

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

https://codecademy-school.s3.amazonaws.com/uk-curriculum/web.pdf

UNIT OVERVIEW: Make a Website

codecademy

Lesson	Course	Exercises (Cards)	Objectives	Progression Pathways	Time (min)
1	1. Structure your page	(1 – 13) 1 - 9	• Understand what HTML & CSS are used for • Understand various HTML elements & implement them • Understand div elements and wrap elements with them	L1 hardware L1/2 data AB AL GE	30
2	2. Style your text	(1 - 13)	• Understand CSS rules & implement them • Understand selectors, classes & properties • Edit properties	L1 hardware L4 algorithms AL	10
3	3. Style your elements	(1 – 18) 1 - 10	• Understand & practice editing properties • Create CSS rules to style elements		30
4	4. Organise your page	(1 - 7) 1 - 4	• Understand display, position & float properties • Implement display, position & float properties		20
5	5. Kickstart your webpage	(1 - 13) 1 - 14	• Understand & implement Bootstrap • Understand & implement tabs, pills and jumbotrons • Use Bootstrap's grid to organise & create new sections • Edit CSS rules	L4 algorithms L5 communication L5 programming AB AL	40

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

https://codecademy-school.s3.amazonaws.com/uk-curriculum/make-a-website.pdf

UNIT OVERVIEW: jQuery

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
1	Introduction to jQuery	1 - 13	• Understand what jQuery is used for, how it is linked to HTML/CSS and DOM • Understand that jQuery is a library of JavaScript and the basic syntax such as <script> • Understand what $(document).ready() is • Understand the syntax of a function and implement the .mouseenter(), .mouseleave(), .slideDown and .fadeTo() keywords	L2/4 data L5 programming AB AL GE	30
2	jQuery Functions and Selectors	1 - 13	• Understand variables, how they are used in functions and how they are conventionally named • Implement event handlers and effects • Recognise that functions can select elements, classes and IDs from HTML • Recognise that functions can pass other functions and multiple selectors can be inputs	L3/5 programming AB AL	30
3	Modifying HTML Elements	1 - 5	• Understand how HTML elements can be dynamically added using .append(), .prepend() and .appendTo() keywords • Understand how to move elements and specify locations of new elements using the .before() and .after() keywords	L3 programming AB	15
4	Modifying HTML Elements	6 - 14	• Recognise and implement how to add, remove and toggle classes from a selector using .addClass() , .removeClass() and .toggleClass() • Understand that .height(), .width() and .css() event handlers can be used to modify CSS elements • Understand that HTML elements can be modified and implement using .html() and .val() • Understand and implement the .on() general handler	L7 data AB	25
5	jQuery Events	1 - 12	• Review jQuery event handlers such as .click() and learn .hover() and .dblclick() • Recognise that multiple functions with effects can be passed into a single eveny handler • Practice the .fadeOut(), .addClass() and removeClass() effects • Understand and implement the .focus() event handler to a CSS element • Practice adding a CSS element using .css() keyword • Recognise and implement the keyDown() event handler and the .animate() effect	L7 data AB	30
6	jQuery Effects	1 - 14	• Understand the jQuery UI library and the event handler .effect() on images and <div>s • Understand and implement the following inputs to .effect() : 'explode', 'bounce', 'slide', 'draggable', 'resizable'. • Recognise and implement UI event handlers that pass lists : .selectable(), sortable() • Practice editing the HTML doc and implementing the .accordion() handler to an ID	L5 programming AB AL	30

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

https://codecademy-school.s3.amazonaws.com/uk-curriculum/jquery.pdf

UNIT OVERVIEW: Make an Interactive Website

codecademy

Lesson	Course	Exercises (Cards)	Objectives	Progression Pathways	Time (min)
1	1a. Your first program	(1 – 10)	• Understand what JavaScript & jQuery are used for • Understand and create script elements • Understand and implement $(document).ready() • Debug the code	L2 algorithms L2 programming AL	15
2	1b. Push menu	1 - 8	• Understand what jQuery is • Create methods containing functions to respond to user actions	L4 programming AB AL	25
3	2. JavaScript	1 - 13	• Understand what JavaScript is • Understand, create, edit and compare variables • Understand if and if/else statements • Understand functions and that they take parameters as inputs	L3 programming AB	15
4	3a. Events	(1- 8)	• Understand events and event handlers • Create event handlers • Understand and implement the .keypress() method	L4/6 programming AB AL	15
5	3b. Newsreader	1 - 10	• Create and edit event handlers • Create if/else if/else statements	L4 programming AL	30
6	4a. DOM manipulation	(1 – 19)	• Select exisiting elements and create new elements • Understand and create methods to add, hide and remove elements • Recognise the DOM is useful to represent relationships between elements • Understand and select children & siblings of elements	L6 programming AB AL	15
7	4b. Status update	1 - 9	• Create event handlers • Replace the text within an element • Create an if/else if/else statement	L4/5 programming AL	30
8	5a. Effect	(1 – 10)	• Understand and create jQuery methods to animate HTML elements • Understand the .animate() method allows you to create your own animations	L6 programming AB AL	15
9	5b. Flipboard	1 - 12	• Create and edit event handlers • Create if statements	L4/5 programming AB AL	30

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

https://codecademy-school.s3.amazonaws.com/uk-curriculum/make-an-interactive-website.pdf

UNIT OVERVIEW: JavaScript Part 1

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
1	1. Getting Started with Programming	1 - 11	• Understand what JavaScript is and what its used for • Understand and create comments • Demonstrate basic mathematical operations • Understand and create confirm and prompt dialogs • Understand data types in JavaScript: strings, numbers and Booleans	L1 programming L1 algorithms L2 data AB AL GE	20
2	1. Getting Started with Programming	12 - 20	• Demonstrate console.log and recognise comparison operators ('<', '>', '==', '!==') • Understand and construct if / else statements using comparison operators • Understand and practice debugging • Review data types, comparators, if / else statements and maths operations • Understand and implement modulo (%) using an if / else statement	L2/4 programming L3 algorithms AL	20
3	1. Getting Started with Programming	21 - 28	• Understand and practice implementing the substring keyword • Understand and practice using variables • Review variables and manipulation of numbers & strings • Practice using variables in if / else statements	L3/5 programming AB AL	20
4	2.Choose Your Own Adventure	1 - 7	• Plenary activity synthesising Module 1: Getting started with programming • Assign prompt to a variable, construct if / else statements with comparison operators and use console.log	L3 programming L4 algorithms AB AL DE	30
5	3. Introductions to Functions in JS	1 – 8	• Understand what a function does and how it works • Practice creating and debugging functions • Recognise and implement the return keyword	L2/6 programming L6 algorithms AB AL GE	20
6	3. Introductions to Functions in JS	9 – 13	• Recognise and implement functions with more than one parameter • Understand and Evaluate Global and local variables • Practice functions with if / else statements	L6/7 programming AB AL DE GE	15
7	4. Build "Rock, Paper, Scissors"	1 – 9	• Plenary activity synthesising Module 3: Introductions to functions in JS • Understand and implement Math.random() method and use else / if statement • Create functions using multiple if / else statements • Extension: Improve game with the skills previously acquired	L4 algorithms L5/6 programming AB AL DE	20
8	5. Introduction to 'For' Loops	1 – 8	• Understand 'for' loops, how they are helpful and the general syntax • Understand how to initiate, control and end a for loop • Practice a 'for' loop counting down	L4 algorithms AB AL	20

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

UNIT OVERVIEW: JavaScript Part 2

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
9	5. Introduction to 'For' Loops	9 - 13	• Understand what an array is and how to create one • Understand how to access an element of an array • Practice creating and accessing all elements of an array using a for loop	L2/6 data L6 programming AB GE	20
10	6. Search Text For Your Name	1 – 7	• Plenary exercise synthesising Module 5: Introduction to 'For' Loops • Recognise and implement text wrapping and the .push() method for arrays • Create a searching program using for loops, if statements and console.log • Extension: Fix problems highlighted	L2 algorithms L2 programming L3/4 data AL	20
11	7. Introduction to 'While' Loops in JS	1 – 11	• Understand what a 'while loop is useful for and general syntax • Recognise infinite 'while' loops and the use of Booleans • Compare and evaluate 'while' and 'for' loops • Understand and implement a do / while loop • Practice all types of loops	L4/8 programming AB AL GE	20
12	8 . Dragon Slayer	1 – 6	• Plenary activity synthesising Module 7: Introduction to 'While' loops in JS • Create an interactive game, implement the math.floor method • Use a while loop and multiple if / else statements • Extension: Improve game and develop console.log statement	L3 algorithms L4/L6 programming AL GE	30
13	7. More on Control Flow in JS	1 – 9	• Review if / else statements and for and while loops • Understand and implement the inNaN method • Understand and implement a switch statement, adding cases and a default	L4/5 algorithms AB AL GE	20
14	7. More on Control Flow in JS	10 – 14	• Understand and implement the 'And, 'Or', and 'Not' logical operators	L5 programming AL	15
15	8. Choosing Your Own adventure 2	1 – 6	• Plenary activity synthesising Module 7: More on Control Flow in JS • Understand and use .toUpperCase() and .toLowerCase() • Implement a switch statement, if / else statements and logical operators • Extension: Add further cases to expand game	L5 algorithms L5 programming AB AL DE	20
16	9. Arrays and Objects in JS	1 – 8	• Review arrays and practice looping through elements of arrays • Recognise a heterogeneous, two dimensional and jagged arrays	L5 algorithms L5/8 programming AB AL DE	20

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

UNIT OVERVIEW: JavaScript Part 3

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways	Time (min)
17	9. Arrays and Objects in JS	9 – 17	• Understand what objects are a compare the syntax for the two notations of declaring objects • Practice creating a heterogeneous and multidimensional array with an object • Practice creating and editing objects	L5/8 programming AB AL DE	20
18	10. Contact List	1 – 8	• Plenary activity synthesising Module 11: Arrays and Objects in JS • Create objects within objects with various types of properties • Implement a for / in loop to search properties of object • Recognise and implement conventional formatting	L5 algorithms L8 programming AB AL DE	20
19	11. Introduction to Objects I	1 – 12	• Review data types, comparator, arrays, functions and switch statements • Practice nested conditional in a for loop • Review and practice creating objects in literal and constructor notation • Review and practice accessing properties using dot and bracket notation	L4 algorithms L6 data AB AL	20
20	11. Introduction to Objects I	13 – 19	• Understand what a method is and practice creating methods • Understand the 'this' keyword and how a method can be called by multiple objects • Practice creating methods that are called by multiple objects	L6/7 programming AB AL DE GE	30
21	11. Introduction to Objects I	20 – 25	• Understand and create custom & method constructors • Practice constructors	L6 programming AB AL	20
22	11. Introduction to Objects I	26 – 33	• Recognise an array of objects • Understand how objects can be passed into functions • Review creating objects, custom constructors and methods	L6 programming AB	15
23	12. Building an Address Book	1 – 6	• Plenary activity synthesising Module 12: Introduction to Objects I • Practice creating and storing objects in arrays • Create a function to access properties of objects • Create a for loop to call a property for all objects in the array • Create a function to search for a property of an array of object and add objects	L8 programming AB DE	20
24	13. Introduction to Objects II	1 – 10	• Review literal and constructor notation for object amd methods • Review the 'this' keyword and calling a property value using bracket notation • Understand the 'typeof' and "hasOwnProperty" keywords • Review a for / in loop with bracket notation	L5 algorithms AL	20

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

UNIT OVERVIEW: JavaScript Part 4

codecademy

Lesson	Course	Exercises	Objectives	Progression Pathways		Time (min)
25	13. Introduction to Objects II	11 – 21	• Understand what a class is and why it is useful • Understand what a prototype is and how the keyword can be used to add methods to a predefined class • Understand inheritance and recognise what is does • Implement prototype keyword to add methods to classes	L8 programming	AB DE	20
26	13. Introduction to Objects II	22 – 26	• Understand what a public property and private variable is • Understand how to access a private variable using a public method • Understand how to call a private method using a public method	L7 programming	AB AL	20
27	13. Introduction to Objects II	27 – 30	• Practice using for / in loop and typeof keyword • Practice using prototype to add a method to a class • Recognise that all objects have 'hasOwnProperty' as a property • Compare private and public properties	L5 algorithms L7/8 programming	AB AL DE	20
28	14. Building a Cash Register	1 – 7	• Plenary activity synthesising Module 14: Introduction to Objects II • Practice custom constructors, calling methods in an object and recognise that method can be used to call another method • Add functionality to methods that allow the void of transactions and quantities of products	L8 hardware	AB AL DE GE	30

Computational thinking concepts: AB - Abstraction GE - Generalisation AL - Algorithms EV - Evaluation DE - Decomposition

https://codecademy-school.s3.amazonaws.com/uk-curriculum/javascript.pdf

Appendix H

HTML and CSS Introduction to Vocabulary Notes

WHAT IS HTML? THE ULTIMATE WEB DESIGNER • BRAD HUSSEY

HTML (HyperText Markup Language) is a language used to create documents on the web. HTML is meant for meaning and structure of a web page’s content. HTML contains tags that organize and structure text, include images, create forms & tables, and link to other documents or web pages all across the Internet.

HTML Syntax The basic syntax of HTML is made up of Tags, Attributes and Elements.

Tags

HTML tags are tag names surrounded by angle brackets, like so: <html>

HTML tags usually come in pairs, like so: <body> </body>

The first tag is called the opening tag and the second tag is called the closing tag.

Attributes

Attributes provide extra information about an element and are always specified in the opening tag, like so: <article attribute= “value”> </arcticle>

Attribute values should be enclosed within quotation marks. You may come across something like: <tag attribute = value>

But using quotation marks is more common, cleaner, and best practice.

Elements

An element is a tag and the content it wraps around, for example: <p> element </p>

HTML Parent / Child Structure

HTML tags are like nested containers. Like Tupperware, or those stackable Russian dolls.

It’s also called parent/child relationships, where the containing element is referred to as the parent of the element contained within it.

WHAT IS CSS? THE ULTIMATE WEB DESIGNER • BRAD HUSSEY

CSS (Cascading Style Sheets) is a language used to create documents on the web. CSS is meant for the presentation of a web page's content, or rather, how a web page is meant to look. We can use CSS to modify any HTML tag, so as to deliver a branded experience in graphical browsers. Users accessing the pages with more primitive devices will still be able to understand and use the site. This is called graceful degradation.

The Style Rule CSS allows the author to specify formatting & presentation of HTML elements using CSS Style Rules. A CSS Style Rule looks like this:

```
h1 {
    font-size: 24px;
    font-family: Helvetica;
    color: blue;
}
```

Selectors

Selectors are used in CSS to style a specific HTML element, or multiple HTML elements.
h1 one in the above example is the selector

Properties

Properties are located within the curly braces of a CSS selector.
font-size, font-family, and color are examples of properties in the example above

Values

A value is assigned to a property.
24px, Helvetica, and blue are the values in the example above

Appendix I

Examples of Student Work

Unit 6 HTML and CSS. Codecademy

For problem 1, what is the difference between HTML and CSS? [L2]

HTML: language meant for meaning and structure of a web page's content

1. CSS: language used to create documents on the web. Meant for presentation of a web page's content (how the web page is supposed to look)

For problem 2, what is the meaning of a <h1> </h1> tag in your HTML code? [L2]

2. header

For problem 3, what is the difference between a <ol> tag and <ul> tag in your HTML code? [L2]

3. ol means ordered list (1... ,2... 3...) ul means unordered list (•... ,•...)

For problem 4, fill in the code with the missing parts, and add your own content to fit the following criteria. [L3]

[Title, Heading, Link, 2 List Items, Closing all tags]

```
<DOCTYPE! html>
  <html>
    <head>
        <title> I am a title </title>
    </head>
      <body>
        <h1> I am a header </h1>

        <a href= "www.google.com" > go to this website </a href>

        <ul>

          <li> I am an unordered list </li>

        </ul>

      </body>
    </~~head~~ html >
```

For problem 5, describe what each of the following tags did in your code, be specific. [L4]

<html> started my web page so I can write on it

<title> gave my page a title (doesn't actually show up)

<body> signal, that everything on my page starts showing up

<a> means that I insert a link or image that can link to another website

href made the text a link to another website

 created an unordered list on my page

 an element in my list

Unit 6 HTML and CSS. Codecademy

For problem 1, what is the difference between HTML and CSS? [L2]

CSS (cascading style sheets) is a language used to create documents on the web CSS is meant for presentations of a web pages content, rather how it suppose to look

1. HTML (Hyper Text markup language) is a language used to create documents on the web. It's meant for meaning and structure of a web page

For problem 2, what is the meaning of a <h1> </h1> tag in your HTML code? [L2]

2. h1 tag is meaning that a new header / headline ~~is cta~~ was created

For problem 3, what is the difference between a <ol> tag and <ul> tag in your HTML code? [L2]

3. ol tags are standing for an orderd list. ul tags stand for ~~undo~~ unordered list

For problem 4, fill in the code with the missing parts, and add your own content to fit the following criteria. [L3]

[Title, Heading, Link, 2 List Items, Closing all tags]

```
<DOCTYPE! html>
  <html>
    <head>
      <title> Frog </title>
    </head>
      <body>
        <h1> Frogs are awesome</h1>

        <a href= "www.google.com" > frog </ahref>

        <ul>

          <li> benefits of frogs</li>


        </ul>

      </body>
   </html >
```

For problem 5, describe what each of the following tags did in your code, be specific. [L4]

<html> html tag open our code. it is ~~comparable~~ like nested containers

<title> title is giving the web page a name

<body> body is created to write information inside. it can be extended with paragraphs and lists.

<a> a tag is used to link a sentence to a web page or a picture

href used to complete the a tag when you want to link something on a webside

 ul tag is standing for an ~~o~~ unordered list~~, which was created to write about the benefits of frogs, in this case~~

 li is the list tag, where we are writing the containt in, which will be shown on the web page

Unit 6 HTML and CSS. Codecademy

For problem 1, what is the difference between HTML and CSS? [L2]

CSS: you put style to your webpage like color, font.

1. html: allows you to put images, text, links and other pieces of contents in a webpage.

For problem 2, what is the meaning of a <h1> </h1> tag in your HTML code? [L2]

2. <h1> means a header.

For problem 3, what is the difference between a <ol> tag and <ul> tag in your HTML code? [L2]

3. stands for ordered list and <ul> means unordered list.

For problem 4, fill in the code with the missing parts, and add your own content to fit the following criteria. [L3]

[Title, Heading, Link, 2 List Items, Closing all tags]

<DOCTYPE! html>
<html>
<head>
<title> Gladys☺ </title>
</head>
<body>
<h1> I love Math. **</h1>**
**** favorite website
<ul>
**
** favorite food. **
**
**
**
</body>
</html>

For problem 5, describe what each of the following tags did in your code, be specific. [L4]

<html> tells that its an html document.

<title> to put a title or heading on your page.

<body> your starting a paragraph

<a> stands for when your going to put a link on the page.

href Indicates the location of the link

 Unorderedlist; and they can go in any specific order.

 stands for list item and puts bulleted list.