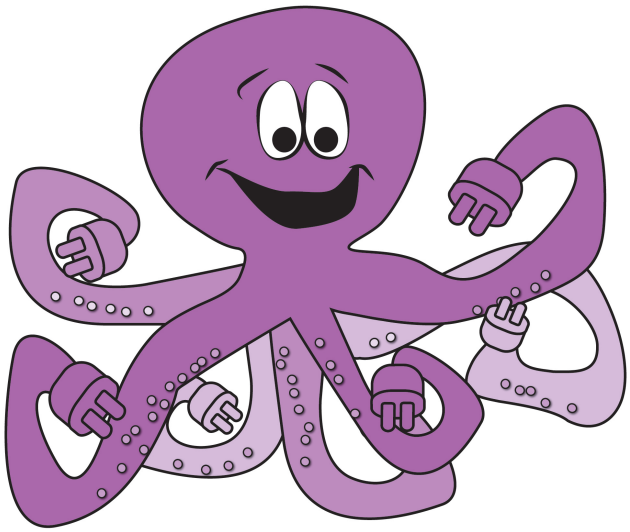
KELP-CS

Curriculum

2014-2015



UCSB
UNIVERSITY OF CALIFORNIA
SANTA BARBARA

UC SANTA BARBARA
engineering

The Gevirtz School
Graduate School of Education

Acknowledgements

Project Directors

Diana Franklin, Department of Computer Science, UC-Santa Barbara
Danielle Harlow, Department of Education, UC-Santa Barbara

Development Staff

Hilary Dwyer
Johan Henkens
Charlotte Hill
Ashley Iveland
Alexandria Killian

James Cheng-yuan Hong
Sharon Levy
Timothy Martinez
Iris-Eleni Moridis
Logan Ortega
Kenyon Prater
Jenny So
John Thomason
Rick Waltman

Elementary Teacher Consultants

Larry Kelman
Tracey Schifferns
Janis Spracher

Pilot Test Teachers

Bridget Berg-Gankas
Mary Lou Furrer
Cati Gill
Shauna Hawes
Ashleigh Lemp
Jamie Thompkins
Laurie Thorbjornsen

This project is supported in part by the National Science Foundation Grant #1240985.
Additional support provided by the University of California-Santa Barbara

Table of Contents

Introduction

Welcome to Kids Engaged in Learning Programming (KELP-CS)! KELP-CS is a modular curriculum for 4th-6th graders created by an interdisciplinary research team at UC Santa Barbara. Each of our three modules consists of 16 hours of instruction, and can be used for each grade level. The first module introduces students to block-based programming and digital storytelling. The second develops these programming skills further and teaches students about game design. And the third module enhances their computational thinking with advanced programming topics. Moreover, KELP-CS emphasizes design thinking, the process by which engineers develop innovative solutions to problems. Design thinking is a core part of new science standards for K-12 students, the Next Generation Science Standards (NGSS) and involves understanding the problem, generating ideas, selecting an idea based on multiple constraints, and improving the idea. We see design thinking overlapping with computer science in ways that allow students to access each in new and exciting ways!

Each KELP-CS module consists of activities that are completed either in the classroom or on a computer. During classroom activities, students discuss, collaborate, and engage with computer science without the computer. These off-computer activities introduce and reinforce concepts that students need to apply on the computer. On the computer, students complete small, discrete programming tasks in our interface to learn about a larger computational thinking idea such as sequential motion, event driven programming, and user interaction. As students finish these computer activities, they will transfer these programming skills directly to their design-projects. These design-projects span each module to provide students opportunities to iteratively revise and improve their programs.

For our programming environment, we use a block-based programming environment, called LaPlaya that runs through any Internet browser. The interface is user friendly and age-appropriate upper elementary school students. Instead of programming by typing individual lines of codes, students can snap individual command blocks program. As students progress through our modules, more and more blocks are introduced and at any time they can use our Sandbox area to explore the entire language.

We believe that increasing the opportunities for elementary school children to learn computer science is an essential aspect of preparing students for computer science careers as well as preparing all students to be comfortable and adept with technology. Our goal is to create a curriculum that any elementary school teacher can implement with their class during an academic day. We invite you to explore computer science with us, and help prepare the next generation of computer scientists.

Best wishes

The KELP-CS Design Team

KELP-CS: Scope and Sequence

Module 1: Digital Storytelling

Module 1 is intended as a 4th, 5th, or 6th grader's first introduction to computational thinking and programming. Students learn general programming concepts in the context of a particular language and programming environment (LaPlaya). Some general concepts are the importance of placing things in order in a program, breaking down complex tasks into their basic components, and properly associating code with the events that trigger them, and managing the complexity of several sprites moving at once. LaPlaya programming concepts they will learn are sequential programming, event driven programming, costume changes, and scene changes. As students learn and practice these concepts, they will apply them to their animated story, satisfying design thinking standards.

Module 2: Game Design

Module 2 is geared towards 5th grade and builds off of module 1. Students learn the different skills needed to design a game in LaPlaya. Topics include loops, decision making, variables, and comparing different solutions
Module 3:

Module 3: Advanced Topics

Module 3 is geared towards 6th grade and builds off of module 1 and 2. Topics include music as numbers, parallelism in every day life, parallel sorting, and breaking down complex problems

Introduction to Computer Science

Overview for Teachers

What is Computer Science?

There is often confusion over the swirl of terms related to computing, technology, and computer science that can be daunting when trying to make decisions about what students need to learn related to computer science. Computer science, as defined by the Computer Science Teachers Association (CSTA) and the Association for Computing Machinery (ACM), is "An academic discipline that encompasses the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society".

Computer science is not just about the use of computers or computer applications, it also includes the knowledge and skills necessary to build the next generations of software and hardware tools that the world needs. The national urgency to improve science, technology, engineering, and mathematics (STEM) education is palpable, as officials have called for reforms in these areas, but what is not clear to policy makers at all levels is that computer science is frequently left out of these initiatives. Computer science drives innovation in all STEM disciplines but it is also a distinct discipline with an extensive body of knowledge.

Computer science teaching sits on a continuum from basic computing concepts that can be attained at elementary and middle school levels to deeper knowledge, skills, and practices more appropriate for secondary school. At the elementary school level students should be introduced to foundational concepts in computer science by integrating basic skills in technology with simple ideas about computational and algorithmic thinking. Learning standards developed by an ACM task force (*A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum committee*) include the following for grades three through five that are included in the KELP-CS curriculum:

- Be comfortable using keyboards and other input and output devices.
- Discuss common uses of technology in daily life and the advantages and disadvantages those uses provide.
- Use technology tools for individual and collaborative writing, communication, and publishing activities to create presentations for audiences inside and outside the classroom.
- Use online resources to participate in collaborative problem-solving activities for the purpose of developing solutions or products for audiences inside and outside the classroom.
- Use technology resources for problem-solving, self-directed learning, and extended learning activities.

Why teach Computer Science?

No other subject will open as many doors in the 21st Century, regardless of a student's ultimate field of study or occupation, as computer science. At a time when computing is driving job growth and new scientific discoveries are happening all the time, gaining a deeper knowledge of computer science and its fundamental aspects is essential not only to have a clear understanding of "what is going on under the hood" of computer software or hardware, but also to develop critical thinking skills that will serve a student throughout his or her career. The U.S. Bureau of Labor Statistics projects that the computing sector will have 1.5 million job openings over the next 10 years, making this one of the fastest growing economic fields

The knowledge and skills imparted by computer science also enables innovation and opens doors. Many fields of science and business depend on computer science. Despite the incredible diversity of the U.S. workforce, it is clear that most of today's jobs depend on some knowledge of, and skills to use computing technologies. It is also clear that this trend is growing as computing becomes embedded more deeply in everyday commerce and society. Computing touches everyone's daily lives; Securing our cyber-infrastructure, voting in elections, protecting national security, and making our energy infrastructure more efficient are among numerous issues dependent on computing and a strong computing-savvy workforce. If K–12 schools are seeking to make students college- and career-ready, computer science should be part of the core curriculum.

Increasing the opportunities for elementary school children (especially girls and other underrepresented minority groups) to learn computer science is an essential aspect of preparing students for computer science careers as well as preparing all students to be comfortable and adept with technology. Research has shown that students' career aspirations at eighth grade are strong predictors of whether they will attend college and whether they will pursue careers in science or engineering, highlighting the importance of experiences children have prior to eighth grade. Fortunately, more children are gaining experiences in computer science at younger ages. Programming environments designed for children and novices (e.g., Scratch, Alice) are easily accessible by classroom teachers and coding in K-12 education has become a huge movement.

What is unclear to many educators is what curriculum will support this growing trend. Although we are beginning to understand how best to teach computer science at the high school level and middle school level, we know comparatively little about effective instruction at the K-5 level. Luckily, some systematic curricula, such as KELP-CS, have been developed that are targeted at helping elementary school students learn programming and computer science. This kind of curriculum is based upon the higher tiers of Bloom's cognitive taxonomy (involving design, creativity, problem solving, analyzing possible solutions to a problem, collaboration, and presenting work) and develops and extends logical thinking and problem-solving skills that can be applied to real world problems.

How will your students be learning Computer Science?

In addition to engaging in the computation thinking and problem-solving aspects of computer science, students will also be introduced to and gain a foundation in an important aspect of computer science; **programming**. Using the modified Scratch environment, called LaPlaya, to either debug existing programs or create new programs students will learn important computational thinking and programming skills including:

- Sequencing
- Breaking down actions
- Event driven programming
- Initialization
- Animation
- Scene changes

Using the KELP-CS curriculum students will also be engaging in many computer science practices such as precision, optimization, utilizing tools strategically, and switching back and forth between thinking as a software developer and thinking like a software user.

References

Goode, J., Margolis, J., & Chapman, G. (2014). Curriculum is not enough: the educational theory and research foundation of the exploring computer science professional development model. In *Proceedings of the 45th Technical Symposium on Computer Science Education (SIGCSE '14)*. Atlanta, GA: ACM.

Liberman, N., Kolikant, Y., & Beeri, C. (2012). "Regressed experts" as a new state in teachers' professional development: lessons from computer science teachers' adjustments to substantial changes in the curriculum. *Computer Science Education*, *22*(3), 257-283. doi:10.1080/08993408.2012.721663

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cuniff, D., ... & Verno, A. (2011). CSTA K-12 Computer Science Standards. *CSTA Standards Task Force*.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running on empty: The failure to teach K-12 computer science in the digital age. Association for Computing Machinery. *Computer Science Teachers Association*.

Zendler, A., & Hubwieser, P. (2013). The influence of (research-based) teacher training programs on evaluations of central computer science concepts. *Teaching & Teacher Education*, *34*130-142. doi:10.1016/j.tate.2013.03.005

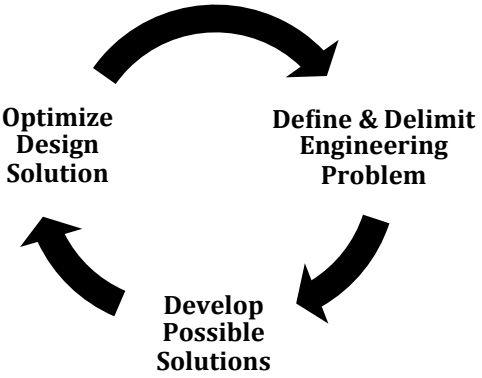
Introduction to Engineering Design Thinking

Overview for Teachers

What is Engineering Design Thinking?

Design thinking is the process by which engineers develop innovative solutions to problems and is now a core idea in new science standards for K-12 students, the Next Generation Science Standards (NGSS). The process involves understanding the problem, generating ideas, selecting an idea based on multiple constraints, and improving the idea. Even before children enter school, they use the process of design thinking to help them solve problems.

Consider a child who lets go of the string of a helium-filled balloon in her kitchen. The balloon, now on the kitchen ceiling becomes too high for her to reach, presenting a problem to solve – How to reach the balloon? She may consider the many resources available to her right in her kitchen such as chairs and cooking utensils. First, she might try using a chair to stand on. If the chair is not high enough, she might try to hold a wooden spoon in her hand to extend her reach while standing on the chair. If she then notices that she can hit the string, but not grasp it with the spoon, she stands on the chair holding a set of tongs, a solution that would allow her to reach and grasp the string. We encounter these sorts of engineering problems and engage in this sort of problem solving or design thinking every day.



The Next Generation Science Standards (NGSS) breaks design thinking into three stages: 1) Defining and delimiting an Engineering Problem, 2) Developing Possible Solutions, and 3) Optimizing the Design Solution The table below describes what students in grades 3-5 should be able to do to demonstrate understanding of these three stages.

Define/Delimit Problem.	Define a simple design problem reflecting a need or a want that includes specified criteria for success and constraints on materials, time, or cost.
Develop solutions.	Generate and compare multiple possible solutions to a problem based on how well each is likely to meet the criteria and constraints of the problem.
Optimize design solution	Plan and carry out fair tests in which variables are controlled and failure points are considered to identify aspects of a model or prototype that can be improved.

The first step is to **define the problem**. An engineering problem includes **goals** (or **criteria for success)** for what should be done (e.g., build a bridge that spans a stream that is 5 meters wide and can support the weight of 5 adults) and **constraints** (e.g., a specified budget, limit on materials or time limits). The first step to solving any engineering problem is to fully understand the problem including the goals and constraints.

The second step is to **develop solutions**. This step includes brainstorming or generating several different ideas and then considering how well each idea is likely to meet the problem goals and staying within constraints.

The third step is to **optimize the solution.** Once an engineer has selected a solution, the next step is to optimize that solution. Optimize means to take an idea and make that idea as good as possible. In engineering, there is usually not a perfect solution and multiple factors are involved. Sometimes as one tries to increase optimization along one factor (e.g., using less material), it may mean decreasing performance along another (e.g., speed).

How is Engineering Design Thinking related to Computer Science & Programming?

Engineering Design Thinking is an important part of computer science and programming. When programmers create a game or an app or any other piece of software, they engage in design thinking. They consider the problem they are solving (e.g., how can I make a fun game for 4th graders to learn about math?) and then develop and optimize their solutions.

How will your students be engaging in Engineering Design Thinking?

In this design thinking activities that complement the programming activities in KELP-CS, children will be developing their own piece of software (a "Digital Story" in Module 1). They will learn tools that are useful to computer scientists like storyboards and flow charts as they develop and optimize their digital stories.

References

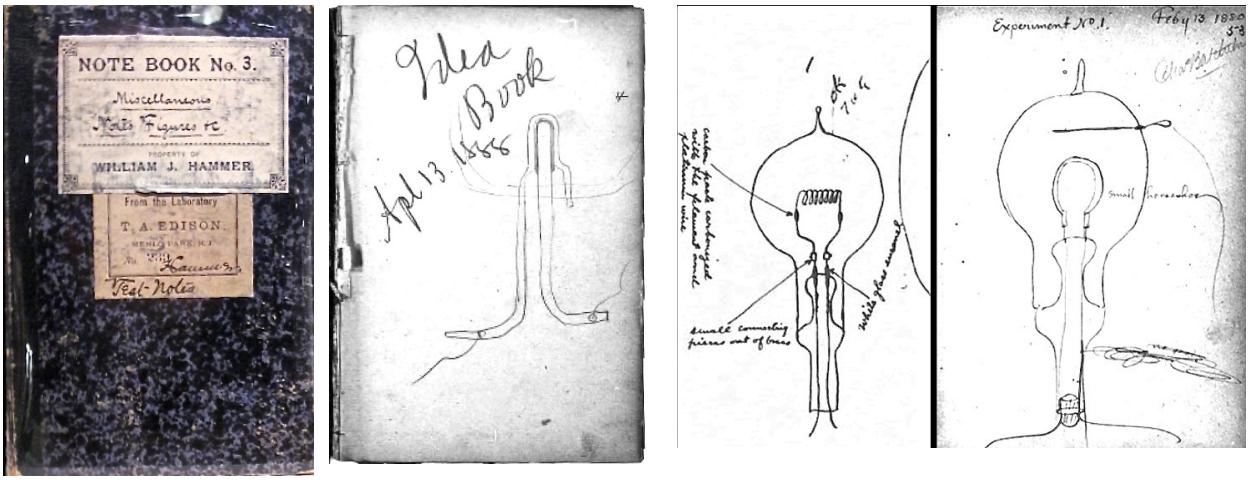
National Research Council. (2012). *A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. Washington, DC: The National Academies Press.

Achieve (2012). *Next Generation Science Standards.* Available at http://www.nextgenscience.org/

Using Design Notebooks

Why use a Design Notebook?

The idea of using a notebook to organize thoughts and ideas and keep track of work is not a new idea. Inspiring people throughout history have used notebooks like these to jot down ideas for books or movies (like Mark Twain and George Lucas), record observations of nature (as Thomas Jefferson and Charles Darwin did), keep track of business (like John D. Rockefeller), or flesh out scientific and engineering ideas and designs (as Isaac Newton and Thomas Edison did). Creating a design notebook is now a common practice in engineering, the sciences, and product design. This is a place where individuals or collectives can throw ideas out, incubate them, come back to them, and most importantly, develop them to fruition.



Thomas Edison's Design Notebook with initial sketches of the incandescent bulb.
Photos courtesy of Edison.rutgets.edu

How will your students be using a Design Notebook?

If you chose to have your students utilize a design notebook throughout the KELP-CS curriculum, students will be using either a composition notebook or spiral bound notebook to keep all of their worksheets, notes, and designs together for the "Design Thinking" project that they will be working on throughout each module.

This design notebook should include a title page with information such as student login information (usernames/passwords) and may include a table of contents (if you are planning on going through their design notebooks this is a good way to navigate them more easily).

Throughout the design thinking lessons students will have worksheets that they can cut and paste into the pages of their design notebooks and can then take notes on the

subsequent pages, ensuring that everything is in order and easy to find. Once students begin programming using LaPlaya they will be keeping notes on what they did on the computer, what did/didn't work, and what they changed in their program. They will also be creating basic flowcharts for their program (an essential software development skill and a required course in undergraduate computer science) and will be adding to them as they learn new, complex commands and adapt their digital stories.

A design notebook is ideal for the KELP-CS curriculum because students will be going back and forth between online and offline activities and will not always be working on their Design Thinking project (especially when you consider that they may not even work on this curriculum each day) so by having everything in one place that is easily accessible and in order students can more easily get back into the proper mindset to work on their project and can easily reference previous material if they desire.

The scope of how your students use design notebooks and engage in the design thinking activities is entirely up to the individual teacher and depends on the individual class. You may consider expanding on some of these lessons to incorporate more writing on or off the computer that would allow you to cover multiple standards at once (Common Core Writing as well as Next Generation Science Standards Engineering Design Thinking). You may also want to include more collaboration in these lessons and have students share their work with each other and can write feedback and suggestions in the margins or on the back of the page. This may also be a valuable tool for you to provide feedback to your students or should you choose to evaluate their work.

References

McKay, B., & McKay, K. (2014, January 1). The Pocket Notebooks of 20 Famous Men. Retrieved July 30, 2014.

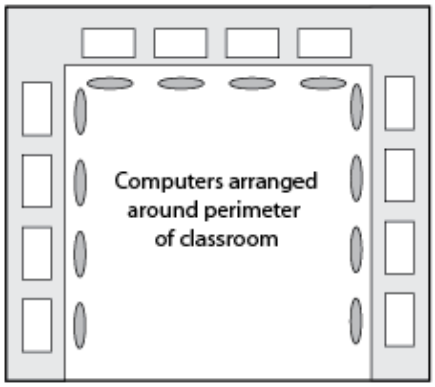
Tips for Teachers

With the help of our pilot teachers and teacher consultants we have compiled a list of tips to assist with common issues that take place in the context of teaching a computer science curriculum in the elementary school classroom such as KELP-CS. We discuss common issues that arise based on the configuration of the learning environment, the mindset of teachers and their students, the grouping of students, and the logistics of working on the computer as well as tips that may help with these issues should they arise in your classroom.

Classroom/Computer Lab Configuration

There are some basic configurations that many classrooms and computer labs fall into and with each configuration comes certain challenges to teaching. Below we will go through some of the possible configurations and common issues that teachers may run into while teaching the KELP-CS curriculum in each.

Around the Perimeter:



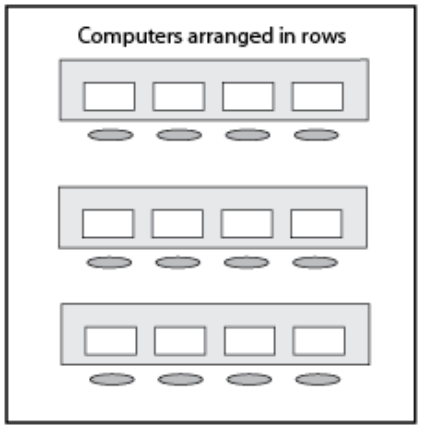
Potential Problems-

When all students face the walls when they are working on their computer it may be difficult for the teacher to get their complete attention when giving instructions are the center of the classroom.

Possible Solutions-

You might consider an easy solution; have the students turn their chairs all the way around to face the center of the room or have the students get up and sit on the floor at the center of the room. Another idea to get students to stop working on the computer and get their attention is to have them put their hands in the air (a big computer stretch) and then turn to the center.

Rows:



Potential Problems-

Because the teacher is usually at the front of the room and cannot see the students computer screens it is difficult to ensure that students are paying attention to the teacher and not distracted with their computer.

Possible Solutions-

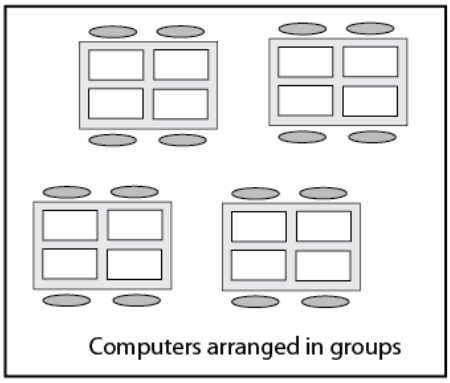
There are several potential solutions for this

configuration including having the teacher give directions at the back of the room and having students turn around (like in the "around the perimeter" configuration). You can also ensure that students' screens don't distract them by having them turn the monitor sideways (if possible), having them turn the monitor off (which shouldn't affect their work only turns off the display), or use a simple piece of paper (shown below) to cover the screen when needed and can be flipped up when students are working on the computer. A piece of paper taped to the top of the computer screen is a great way to ensure that when you're talking students cannot look at their computers. And you can easily see if any student doesn't have their screen covered because it flips to the back where you can see it from the front of the room.

Groups:

Potential Problems-

Like in the "rows" configuration, it is difficult for the teacher to see all students' computer screens at one time and students may be distracted when they should be listen to directions. Another potential problem that may arise is that students will work together more frequently (which is a good thing), but some students may feel less comfortable working with some other students and may take a back seat and let another students do all the work.



Computers arranged in groups

Possible Solutions-

If using personal computers, only allow students to have their computers on their desk when they will be using them. You may even ask students to shut their computers or put them away all together if the class is having difficulty paying attention to instructions. To help with group work you may want to circulate around the room while students are working on the computer and make sure no students is being left out. You may also consider setting some guidelines for

students ensuring that all students work on their own computer and must complete their own work unless explicitly stated they work in groups.

Individual Desks:

Potential Problems-

Sometimes students get antsy and move around a lot in their desk. If they are using a personal computer at their individual desk and it is not stable there is a chance that the computer could fall off their desk (which may damage or break it).

Possible Solutions-

You can just warn students about moving their desk too much because the computer might fall or you may consider using a different classroom configuration with multiple desks put together, which would stabilize them and prevent the desks from moving around much.

"Fixed" vs. "Growth" Mindset

Research has shown that there is a continuum of beliefs about where success comes from. The ends of this continuum are called "Fixed Mindset" and "Growth Mindset." Students with a fixed mindset believe that success comes from innate abilities, while those with a growth mindset believe that success comes from hard work and training. Students with a fixed mindset are likely to fear failure and are less likely to try things that they do not already know they will be successful with. In contrast, students with a growth mindset are likely to continue to work at tasks that they find challenging.

Fixed Mindset	**Growth Mindset**
• Intelligence is innate and does not change	• Intelligence can be developed
• Desires to look smart	• Desires to learn more
• Avoids challenges	• Embraces challenges
• Gives up easily when faced with obstacles	• Persists when faced with obstacles
• Worse performance	• Better performance

Students are likely to come into computer science activities with a wide range of experiences. Students with fixed mindsets may see other students (who have more experience) being successful and determine that they are "not good at computer programming." It is important to encourage a **growth mindset** and remind students that, through experience, they will get better.

Encouraging your students to have a growth mindset
Praise effort, "I see you worked really hard on that task"
Help students see that there are things they can learn even when they are unsuccessful.

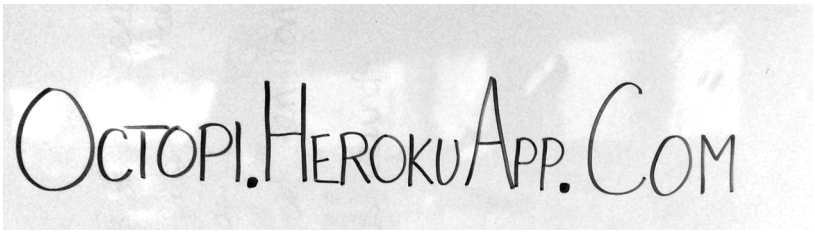
For more information, watch the video
https://www.youtube.com/watch?v=xs9fddMg71o

Reference: Dweck, C., (2006). Mindset: The new psychology of success. Random House: New York.

Computer Logistics

Navigating the Internet

Students often struggle with typing in web addresses because of their limited typing skills and their lack of attention to detail. Many students would add spaces or extra punctuation to a web address, which resulted in an error message and much frustration and chaos in the classroom.



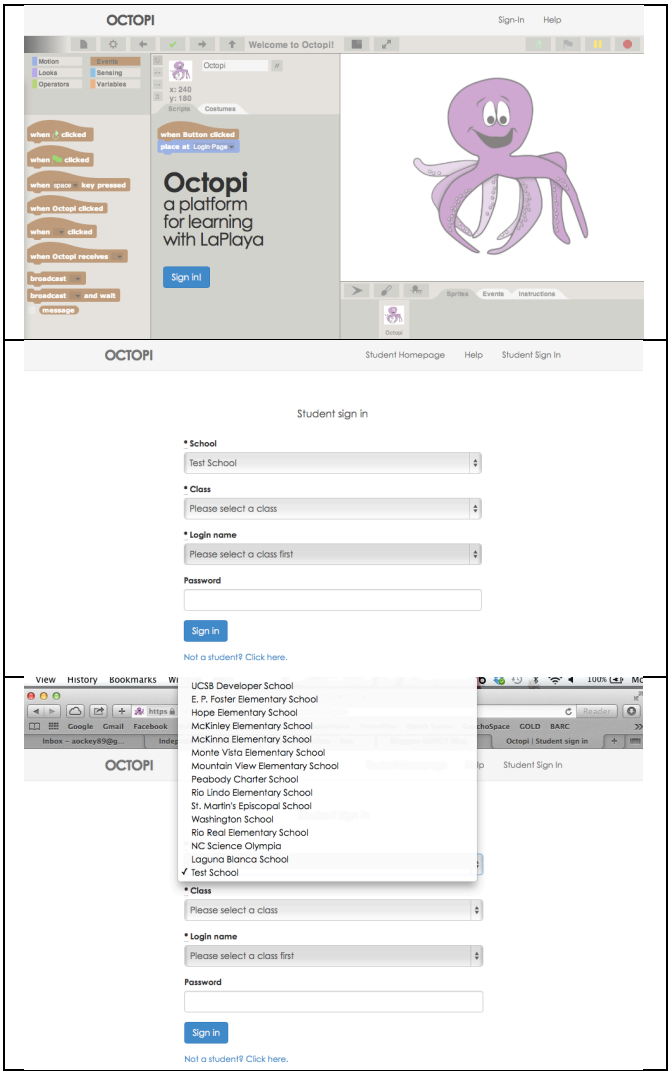
To help remedy this situation you can write the web address on the board in LARGE, clear print (You can even write it in all CAPS because you will navigate to the same page regardless of if the address is upper- or lower-case) and emphasize that there are NO spaces in web addresses.

This is also a good time to implement a system for what students should do if they have questions (if you have not already established this in your classroom) because this process usually results in multiple students having difficulty and since there is usually only one teacher students will have to wait before continuing on with the lesson.

Logging In to LaPlaya

Student Login-

Students should save their login username and password (you can have them write it inside their design notebook or somewhere else easily accessible). You may also want the students to bookmark the web address in their browser (Safari, Firefox, Google Chrome, etc.) so they can easily come back to the website without having to type the address in each time.



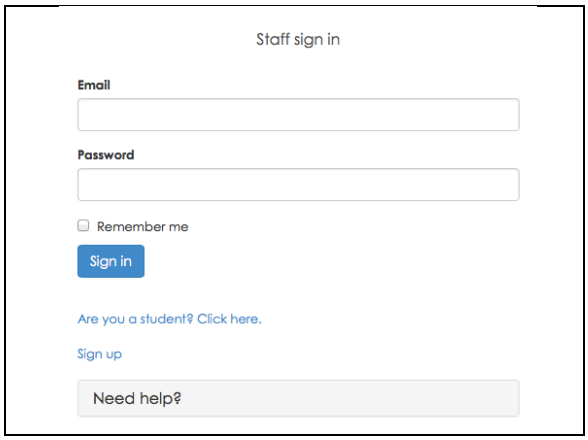
Once you get to the website you will see the LaPlaya interface in the background (top picture to the left). Students will click on the blue "Sign In" button to sign in to their account.

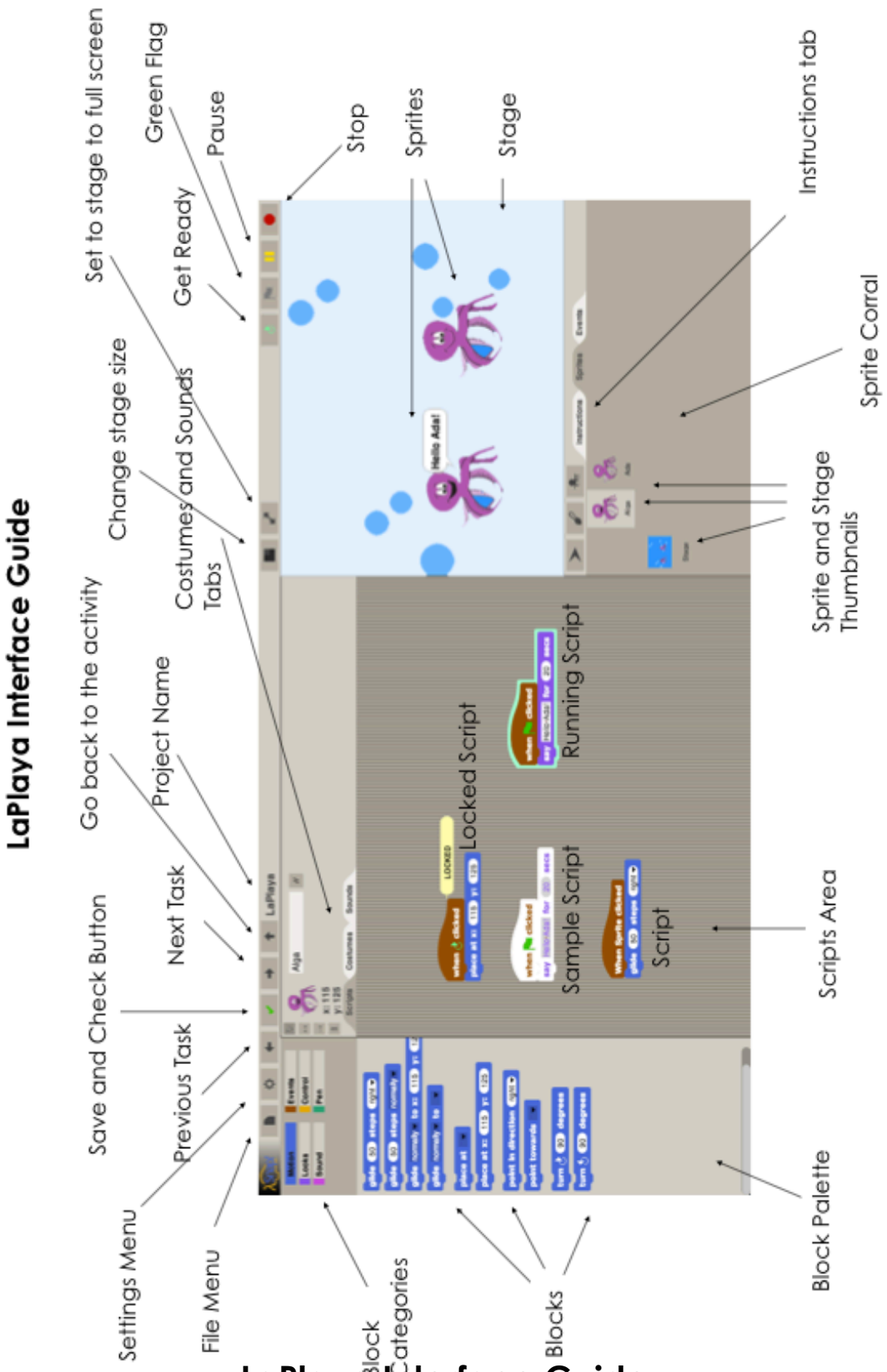
This will take you to a sign in page (middle picture to the left)where students will select their school from the dropbown menu (shown on the bottom picture to the left) and then select their class and login name (which will be assigned to them).

Students will also be given a password that they will need to type in. Their password should not be too difficult, but this is a good time to begin talking about how computer need you to give precise instructions and that it is important to type exactly what they are supposed to or the computer will not recognize it.

Teacher Login-

To sign in to a teacher account you will go to the same web address and click on the blue "sign in" button, but once you reach the log in page you will click on the blue writing at the bottom of the page that says, "Not a student? Click here." This will redirect you to the staff sign in page (shown to the right), where you will enter you email and password. Once signed in you have access to the curriculum, LaPlaya, and information about your class(es).





LaPlaya Interface Guide