MASSACHUSETTS

K-12 Computer Science Curriculum Guide

EDC Learning transforms lives.

MassCAN Massachusetts Computing Attainment Network

SIGILLUM REIPUBLICÆ MASSACHUSETTENSIS

The Commonwealth of Massachusetts Executive Office of Education, under James Peyser, Secretary of Education, funded the development of this guide. Anne DeMallie, Computer Science and STEM Integration Specialist at Massachusetts Department of Elementary and Secondary Education, provided help as a partner, writer, and coordinator of crosswalks to the Massachusetts Digital Literacy and Computer Science Standards. Steve Vinter, Tech Leadership Advisor and Coach, Google, wrote the section titled "What Are Computer Science and Digital Literacy?" Padmaja Bandaru and David Petty, Co-Presidents of the Greater Boston Computer Science Teachers Association (CSTA), supported the engagement of CSTA members as writers and reviewers of this guide.

Jim Stanton and Farzeen Harunani
EDC and MassCAN
Editors

Editing and design services provided by Digital Design Group, EDC.

An electronic version of this guide is available on the EDC website (http://edc.org). This version includes hyperlinks to many resources.

TABLE OF CONTENTS

ABBREVIATIONS USED IN THIS GUIDE

AMSA: Advanced Math and Science Academy

AP: Advanced Placement

API: application program interface

Common Core: Common Core State Standards for Mathematics and English Language Arts & Literacy in History/Social Studies, Science, and Technical Subjects

CS: computer science

CSP: Computer Science Principles (AP course)

CSS: Cascading Style Sheets

CSTA: Computer Science Teachers Association

DESE: Massachusetts Department of Elementary and Secondary Education

EDC: Education Development Center, Inc.

GML: Geography Markup Language

HDMI: high-definition video device

HTML: Hyper Text Markup Language

IDE: integrated development environment

iOS: iPhone Operating System

ISTE: International Society for Technology in Education

IT: information technology

LED: light-emitting diode

MassCAN: Massachusetts Computing Attainment Network

Mass. DLCS standards: Massachusetts' Digital Literacy and Computer Science Standards

Mbps: megabits per second

MCAS: Massachusetts Comprehensive Assessment System (statewide standards-based test)

MOOC: massive open online course

NGSS: Next Generation Science Standards

NICE: National Initiative for Cybersecurity Education

NSF: National Science Foundation

OS: operating system

PD: professional development

PLTW: Project Lead the Way

SQL: Structured Query Language

STEAM: science, technology, engineering, the arts, and mathematics

STEM: science, technology, engineering, and mathematics

STEM+C: science, technology, engineering, and mathematics, plus computing

STL: Standards for Technological Literacy

USB: Universal Serial Bus

VGA: Video Graphics Array

XML: eXtensible Markup Language

INTRODUCTION

Jim Stanton & Farzeen Harunani

Education Development Center, Inc. (EDC), and the Massachusetts Computing Attainment Network (MassCAN) collaborated with the Massachusetts Executive Office of Education, the Massachusetts Department of Elementary and Secondary Education (DESE), and the Greater Boston chapter of the Computer Science Teachers Association (CSTA) to prepare this guide. The goal was to help demystify the landscape of computer science (CS) curricula options and to provide a curated collection of high-quality CS curricula for students in grades K–12. This guide was developed as part of an initiative for school districts to accelerate the creation of classroom opportunities for learning CS that are standards-based, high-quality, career-relevant, and accessible to *all* students at all grade levels.

The hallmark of the Massachusetts educational system, which has placed the Commonwealth at the forefront of education nationally, is that each school district can independently establish educational programs and graduation requirements that best serve its communities, while simultaneously providing high-quality state standards and frameworks that foster a shared understanding among districts of what constitutes a comprehensive and thorough coverage of each discipline. The recently created Massachusetts K–12 Digital Literacy and Computer Science (DLCS) Standards help districts create and shape the most effective CS educational opportunities. This guide is intended to build on that framework by identifying high-quality CS curricula that have been developed and are being offered throughout the country.[1] Equipped with this guide, school districts throughout Massachusetts (and beyond) can see and choose the options that best suit their community's needs. We also believe that one of the best ways to ensure equity across the state—with *all* students having the same exposure to CS—is to provide successful models of implementation for every community.

This guide is most effectively used in conjunction with the Mass. DLCS standards and emerging pathways for teacher licensure, all of which are key elements of a district plan for introducing CS at all K–12 levels. To help clarify the relationship between individual curricula and the DLCS standards, crosswalks are being prepared for some (if not most) of the curricula in this guide and will be available on the DESE website (http://www.doe.mass.edu/stem/dlcs/?section=planningtools)[2].

We recruited a diverse group of experts from around the state and beyond—from teachers to engineers to curriculum writers—to assemble and curate this information. A full list of these experts appears in the Acknowledgments section. We are grateful to these experts for sharing their wisdom via writing and reviewing all the materials in this guide.

An important note: EDC/MassCAN and its partners **do not** endorse any particular curriculum in this guide. In addition, a number of other CS curricula are available, but we were not able to review them due to space and time constraints.

This guide is organized as follows:

- The first section provides a high-level definition of CS and digital literacy.
- The next three sections are grouped by grade level (elementary school [K–5], middle school [6–8], and high school [9–12]), and provide the following information for the curricula we surveyed:
  - A short, high-level overview
  - A breakdown of the basic properties (full year vs. semester, whether the curriculum stands alone or can be integrated into another course, etc.)
  - Teacher training and other support resources
  - Required and recommended technology and other materials and their costs
  - Relationships to other courses and intended pathways, when applicable
  - Requirements for implementation

The final section is a consolidated properties chart that provides a side-by-side comparison of the basic properties for each curriculum in this guide, as identified by the guide writers.

1 Two curricula in the Elementary School section—Museum of Science Elementary Computer Science and STEM+C Integrated Modules—are still in the pilot stage.

2 For more advanced curricula, such as the AP Computer Science Principles and AP Computer Science A courses certified by the College Board, crosswalks will not be prepared, as these curricula are beyond the scope of the Massachusetts DLCS standards

Steve Vinter

Computer science (CS) is about designing and developing computing systems to solve problems. It is a science, so it comprises a set of ideas and principles.

Computational thinking is the heart of CS as it pertains to K–12 CS education. Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer (or human) can effectively carry it out. Computational thinking involves both skills and problem-solving techniques (which are discussed in more detail below). Applying computational thinking to problems typically results in the creation of computing systems, of which the most commonly recognized ones are computers (such as smartphones and laptops) and software applications (such as spreadsheet programs, search engines, websites, and all the applications that run on your smartphone).

Coding (also called computer programming) is the creation of instructions in a form that can be used by a computer to create a software application.

Digital literacy, in contrast to CS, refers to a person's ability to use computers and software applications (which are both designed and developed using CS) to find, evaluate, create, and communicate information. Digital literacy also includes:

- how computing affects society (for example, privacy and the security of information)
- collaboration and research using applications and other digital tools
- the ability to use computing systems, such as devices and networks

The *2016 Massachusetts Digital Literacy and Computer Science (DLCS) Curriculum Framework* can be found at http://doe.mass.edu/frameworks/dlcs.pdf.

The video Teaching Creative Computer Science by Simon Peyton Jones provides an accessible explanation of the essential difference between using computers (digital literacy) and the ideas and principles underlying computer science (CS).

EXAMPLES OF PROBLEMS THAT CS AND DIGITAL LITERACY MIGHT SOLVE

Problem 1: Each day, a bank must process its credit card transactions and identify the most likely fraudulent charges from the previous day. CS can be used to develop a solution, for a number of reasons:

- This problem involves processing enormous amounts of data, more than a human can process; how the data are represented, organized, and analyzed are key elements of the problem.
- The solution must be applied repeatedly—so often, in fact, that it is unlikely to involve human intervention.
- The solution involves a set of rules and steps (called an *algorithm*) for how data are analyzed.
- Human judgment of the rules for what constitutes a fraudulent transaction must be represented in the algorithm.

Problem 2: A presentation must be created that explains the relationship between the most popular books published in 2015 and the National Book Award winners for that same year. This problem requires digital literacy, for a number of reasons:

- The result, a presentation, is a digital artifact of information compiled *by a person* from other data (popular books published in 2015 and National Book Award winners).
- Digital tools (a search engine and a software application to compose a presentation) will be used to collect information and represent the solution.
- Human interpretation and insight about the relationship between the two collections of books (book popularity versus award-winning literature) is not represented as a set of rules to apply repeatedly and systematically to other collections, but rather as a narrative or a kind of analysis.

MORE ON COMPUTATIONAL THINKING

As we noted earlier, computational thinking involves a set of skills and problem-solving techniques, for example:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them, which includes using abstractions and pattern recognition to represent the problem in new and different ways
- Logically organizing and analyzing data
- Representing data through abstractions, such as models and simulations
- Breaking down the problem into smaller parts
- Approaching elements of the problem using programmatic thinking techniques, such as iteration, symbolic representation, and logical operations
- Applying algorithmic thinking, and reformulating the problem into a series of ordered steps
- Integrating modules that solve separate pieces of the problem into a complete solution
- Identifying, analyzing, and implementing possible solutions, with the goal of achieving the most efficient and effective combination of steps and resources
- Understanding the consequences of scale, not only for reasons of efficiency but also for economic and social reasons
- Generalizing and transferring this problem-solving process to a wide variety of problems

Five elements of computational thinking are included in the Massachusetts DLCS Curriculum Framework:

- Abstraction: A process of reducing complexity by hiding details that are irrelevant to the question at hand and bringing together related and useful details in order to focus on the main idea.
- Algorithm: A sequence of precisely defined, reusable steps to solve a particular problem.
- Data: Facts or information used to make calculations or decisions. Collecting, managing, and interpreting a vast amount of raw data is part of the foundation of our information society and economy. New tools and techniques for data collection and analysis enable us to make new insights and higher-level decisions.
- Programming and development: Programming articulates and communicates instructions in such a way that a computer can execute a task. Programming makes use of abstractions, algorithms, and data to implement ideas and solutions as executable code through an iterative process of design and debugging.
- Modeling and simulation: These allow us to represent and understand complex processes and phenomena. Computational models and simulations are created to analyze data, identify patterns, and answer questions about both real phenomena and hypothetical scenarios.

ELEMENTARY SCHOOL CURRICULA AND TOOLS

COMPUTER SCIENCE FUNDAMENTALS

» Code.org

Compiled by Maria Litvin
Instructor of Mathematics and Computer Science, Phillips Academy, and Code.org K–5 facilitator

PROPERTIES

Intended Grade Band	✔ K–5	✖ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✔ Online PD (MOOC, etc.)	✖ Meetups	✔ Local Workshops	✖ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✔ Full Semester	✔ Full Year	✔ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✖ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✖ Physical Computing	✖ Networking and Security	✖ Mobile Programming	✖ Web Development	✖ Databases	✖ Game Design
Required Technology	✔ Desktop or Laptop	✖ iPad/iPhone	✖ Android Device	✖ Chromebook	✔ Works with Any Major OS	✖ Custom Hardware Device	
Required Software	✔ Browser Only	✖ Free Software	✖ Purchasable Software				

OVERVIEW

Computer Science Fundamentals (CSF) is a comprehensive curriculum for K–5 students offered by Code.org, a nonprofit launched in 2013. Code.org is known to many as the organizer of the Hour of Code, an annual event that introduces millions of students to CS. The CSF curriculum is completely free for anyone to use, thanks to Code.org's donors, which include leading U.S. technology companies, foundations, and individuals. CSF courses can be found on the Code.org website (https://code.org/student/elementary).

CSF comprises six courses, from "Course A" to "Course F," which are aligned, respectively, to grades K–5—the sequence preferred by Code.org. However, the grade and age designations in CSF are fluid; CSF can be used in both single-grade and mixed-grade classrooms, or as an enrichment activity. Each course contains 12–20 lessons, with fewer lessons in earlier courses and more in later courses. Each lesson is designed to be 35–45 minutes.

There are alternative paths through the CSF curriculum: "Pre-Reader Express" is a condensed version of Courses A–B; the "Express Course" combines the concepts of Courses A–F into one accelerated course that can be used for older elementary, middle, and high school students without previous exposure to CSF. (An older version of CSF, "Course 1" through "Course 4," is still available online; it is ideal for international students, as it has been translated into more than 25 languages.)

In addition to elements of coding, students study fundamental CS concepts, including algorithms, iterations, conditionals, variables and functions, as well as basics of digital citizenship. The lessons are presented in a fun and age-appropriate manner, fostering students' problem-solving skills, computational thinking, collaboration, and persistence. Students develop interactive games or stories in Blockly, a block-based programming language similar to Scratch but easier to use.

CS concepts are usually introduced with an "unplugged" activity, taught without a computer. Students then solve a sequence of coding puzzles, followed by an assessment. Many sequences conclude with each student working on an open-ended project.

All required CSF materials for students and teachers are online. Teacher materials, such as teacher lesson plans, downloadable instructional videos, "unplugged" activities, and online programming puzzles (with solutions available to teachers), are accessible through a teacher account.

Code.org's comprehensive class management system allows the teacher to easily create student accounts for the whole class at once, monitor each student's progress, and move students to another teacher's CSF class while preserving students' progress records.

Code.org software engineers have implemented a feature that allows two or three students to work together on the same computer in Pair Programming mode (one student as the "driver" and another as the "navigator"), and to have the solution count for each member of the team. The Pair Programming feature allows schools to run CSF courses with fewer computers, Chromebooks, iPads, or other mobile devices than students in class.

CSF courses are aligned to ISTE, Common Core, and CSTA standards for grades K–5.

RESOURCES

Professional Development (PD):

- Code.org offers free seven-hour PD workshops for educators, led by a certified Code.org K–5 facilitator. Workshop participants receive a printed copy of the CSF curriculum guide and "unplugged" lesson plans (also available online). After completing the workshop, participants can apply to receive a free kit for "unplugged" lessons at their school. The lists of workshops and facilitators are available on the Code.org website (https://code.org/professional-development-workshops and https://code.org/educate/professional-learning/cs-fundamentals-directory).
- Schools and districts can hire a Code.org facilitator to conduct a "private" K–5 workshop at their school, at no cost to the school.
- Code.org offers a free, self-paced online course for teachers who wish to implement CSF curriculum in their classrooms (see https://code.org/educate/professional-development-online).

Vendor or PD Provider Expectations: None. Teachers do not need to have prior CS experience. CSF lessons are designed for teachers as "lead learners," learning alongside their students.

Ongoing Support: Code.org has a large online community that includes separate forums for CSF courses (http://forum.code.org/c/csf) and Technical Support (https://code.org/educate/support).

TECHNOLOGY

Required Hardware and Software and Their Costs: Any mix of laptops, Chromebooks, or mobile devices such as iPads can be used; at least one device per two students is recommended, but one-to-one is ideal.

Recommended Technology: CSF will work with any modern browser. A stable Internet connection is required for working on online puzzles. "Unplugged" activities do not require a computer.

Environments and Programming Languages Used: CSF uses Code.org's version of Blockly, a block-based programming language similar to Scratch.

KIBO ROBOT KITS
» KinderLab Robotics, Inc.

Compiled by Jason Innes
Manager of Training and Curriculum Development, KinderLab Robotics, Inc.

PROPERTIES

Intended Grade Band	✔ K–5	✖ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✔ Online PD (MOOC, etc.)	✖ Meetups	✔ Local Workshops	✖ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✔ Full Semester	✖ Full Year	✖ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✔ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✔ Physical Computing	✖ Networking and Security	✖ Mobile Programming	✖ Web Development	✖ Databases	✖ Game Design
Required Technology	✖ Desktop or Laptop	✖ iPad/iPhone	✖ Android Device	✖ Chromebook	✖ Works with Any Major OS	✔ Custom Hardware Device	
Required Software	✖ Browser Only	✖ Free Software	✖ Purchasable Software				

OVERVIEW

KIBO Robot Kits, from KinderLab Robotics, Inc., are specifically designed for young children ages 4–7. KIBO is based on over 20 years of research conducted by KinderLab's co-founder Dr. Marina Umaschi Bers, who is also the Director of Tufts University's Developmental Technologies Research Group. KIBO's development was funded in part by NSF.

The KIBO classroom program includes:

- KIBO robots (one KIBO robot per two children is recommended) and a variety of integrated add-on components
- Standards-mapped curriculum materials for a 20–40-hour robotics-based STEM unit using KIBO
- Teacher training and PD, delivered remotely or in-person, to support successful implementation

KIBO was created for an early education environment. The flexible KIBO robot is designed to be assembled and dissembled by children, thereby de-mystifying robotics. KIBO's unique programming language is based on sequencing and scanning physical wooden blocks; children do not need to work with a computer or tablet screen. KIBO is designed to be extended with arts-and-crafts and other building materials (including LEGO and compatible building bricks), providing a platform for the study of sturdy building and the engineering design process.

The 20–40-hour structured KIBO curriculum, *Creating with KIBO*, is designed for students in pre-Kindergarten through grade 2 and covers many foundational CS and engineering skills that are not often taught in early childhood. These skills are taught through a series of powerful ideas rooted around the engineering design process, robotics, programming, and sensors. Activities and materials encourage mastery of each powerful idea and the foundational academic subjects that support it.

The program culminates in a cross-curricular integrated final project, allowing students to make cross-disciplinary connections between the STEM concepts underlying robotics and other subjects, including social studies, dance, music, and art.

The lessons in *Creating with KIBO* are mapped to Common Core mathematics and literacy standards related to collaboration, planning, engineering, measuring, and estimating. A complete mapping between *Creating with KIBO* and the Mass. DLCS standards for K–2 is available on the DESE website.

RESOURCES

Professional Development (PD): KinderLab offers a variety of PD and training options, all detailed on its website (http://kinderlabrobotics.com/professional-development):

- Each classroom package purchase includes two hours of remote PD to help teachers become familiar with the robot and the *Creating with KIBO* curriculum.
- Face-to-face three- to six-hour PD workshops are offered in the Boston area, which include assistance with curriculum customization.
- School districts can contract with KinderLab for onsite PD to meet their needs.

Vendor or PD Provider Expectations: None

Ongoing Support: KinderLab Robotics, Inc., hosts a variety of support resources—video tutorials downloadable activity guides created by KinderLab, crowd-sourced activities, curriculum units, and classroom videos contributed by other teachers—on its website (http://resources.kinderlabrobotics.com).

TECHNOLOGY

Required Hardware and Software and Their Costs: The KIBO Kit is a self-contained robotic system requiring no additional software or hardware and no screens. KIBO is sold in several configurations, which differ based on the additional sensors and components; these configurations range from the KIBO-10 at $259 to the KIBO-21 at $499.

Classroom packages, including multiple KIBO kits, curriculum materials, and PD, may be purchased at a bundle discount. Prices for the KIBO-21 classroom packages are:

- Activity Center Package (supports four to six students): $1,400
- Small Classroom Package (supports 10–12 students): $2,950
- Full Classroom Package (supports 20–24 students): $5,600

KIBO is available directly from KinderLab (http://shop.kinderlabrobotics.com).

Recommended Technology: Some curriculum units benefit from having a music player available. Optionally, children can document their work through photos and videos using tablets or an equivalent.

Potential Technology Barriers: Children need to learn to scan the wooden programming blocks; most children master this during a single lesson.

Environments and Programming Languages Used: Children program KIBO using a patent-pending tangible block-based programming language, which bears some similarity to ScratchJr (also developed by Dr. Bers); each command is a wooden block, which children sequence and then scan with KIBO's barcode scanner.

SUPPLEMENTAL MATERIALS

All curriculum supplemental materials (including Engineering Design Journals for each student, an Assessment Workbook for each student, one KIBO poster set, and one "KIBO Says" programming game) are available from KinderLab (http://shop.kinderlabrobotics.com) and are included in the classroom packages.

Teachers should have a variety of arts and crafts materials on hand, such as craft paper, cardboard, recycled materials, tape, and scissors, for arts integration activities.

LEGO WEDO CONSTRUCTION KIT

» LEGO Education

Compiled by Ivan Rudnicki
Computer Science Department Chair, Edward Brooke Charter Schools

PROPERTIES

Intended Grade Band	✔ K–5	✖ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✔ Online PD (MOOC, etc.)	✖ Meetups	✖ Local Workshops	✔ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✔ Full Semester	✖ Full Year	✖ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✔ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✔ Physical Computing	✖ Networking and Security	✖ Mobile Programming	✖ Web Development	✖ Databases	✔ Game Design
Required Technology	✔ Desktop or Laptop	✔ iPad/iPhone	✔ Android Device	✔ Chromebook	✔ Works with Any Major OS	✔ Custom Hardware Device	
Required Software	✖ Browser Only	✔ Free Software	✖ Purchasable Software				

OVERVIEW

The LEGO WeDo Construction kit is a set of interlocking plastic building blocks and electronic components designed for making robotic projects that can respond to computer code. Each WeDo Construction kit includes a motor, an infrared motion-distance sensor, and a six-position tilt sensor. Unlike the LEGO Mindstorms robotics kit, the WeDo kit does not have a programmable microcontroller. The code controlling a WeDo project runs on a student's computer, not on the WeDo hardware itself, so projects must remain connected to the computer as the code runs. In the newer version of the kit (WeDo 2.0), this connection is made wirelessly, via Bluetooth; in the older version (WeDo 1.0), the connection is made with a USB cable.

LEGO Education supplies a free drag-and-drop programming environment for the WeDo kits. The WeDo 2.0 software runs on Windows and Macintosh computers, as well as Chromebooks, iPads, and Android tablets; the WeDo 1.0 software requires a Windows or Macintosh computer. Programming of WeDo projects can also be done in Scratch, a widely used, browser-based, drag-and-drop coding environment. When the WeDo hardware is connected to a student's computer, new WeDo-specific programming blocks appear in the Scratch programming interface.

Connecting the WeDo motor and sensors to Scratch opens up a range of interesting applications beyond those made possible with the LEGO software. The hardware can be programmed to respond to events that happen in a Scratch

program—for example, the WeDo motor can be turned on when a key is pressed, the mouse is clicked, or two sprites come into contact on the screen. Likewise, students can create Scratch programs that respond to input from the WeDo sensors—for instance, adjusting a sprite's size or position on the screen based on the value sent to the computer from the motion-distance sensor.

For the WeDo 2.0 kit, instructions for 24 projects are integrated with the programming environment and include both video tutorials and animated depictions of project behavior. These WeDo 2.0 projects are aligned with the NGSS and relate thematically to a wide range of science and engineering topics, such as forces, structures, habitats, volcanoes, and space exploration. According to LEGO, the WeDo 2.0 curriculum offers an estimated 40 hours of instructional content.

For the WeDo 1.0 kit, LEGO offers free, step-by-step pictorial instructions for 12 different projects, organized into four themes: *Amazing Mechanisms, Wild Animals, Play Soccer,* and *Adventure Stories*. The WeDo 1.0 projects vary in complexity, but most can be assembled and programmed in 30–60 minutes each.

RESOURCES

Professional Development (PD): LEGO Education offers both face-to-face and online training programs for teachers interested in working with LEGO WeDo 2.0:

- The seven-hour face-to-face program, *Real Life Science with WeDo 2.0*, is led by a certified LEGO trainer. It is intended to familiarize teachers with the WeDo 2.0 hardware and software and to explore strategies for integrating the technology into science, engineering, and CS classes (https://education.lego.com/en-us/training/face-to-face).
- The online option, *Welcome to LEGO Education WeDo 2.0*, is self-paced and can be completed independently or with a partner (https://elearning.legoeducation.com/wedo-2-0).

Vendor or PD Provider Expectations: None

Ongoing Support: Support for WeDo 2.0 (https://education.lego.com/en-us/support/wedo-2) and the WeDo 1.0 (https://education.lego.com/en-us/support/wedo) is available online.

TECHNOLOGY

Required Hardware and Software and Their Costs: WeDo 2.0 is compatible with Windows and Macintosh computers, Chromebooks, iPads, and Android tablets. WeDo 1.0 is not tablet-compatible. The WeDo 2.0 kit costs $175.95; the LEGO WeDo 1.0 kit costs $164.95.

Potential Technology Barriers: WeDo 2.0 projects require host computers with Bluetooth 4.0 adapters installed and enabled. To program WeDo projects using Scratch, students must use the Chrome browser and must have a Scratch Device Plugin Helper installed and enabled within the browser.

Environments and Programming Languages Used: WeDo 2.0, WeDo 1.0, or Scratch.

SUPPLEMENTAL MATERIALS

All required materials are online at the LEGO Education website (https://education.lego.com/en-us).

ELEMENTARY SCHOOL COMPUTER SCIENCE

» Museum of Science, Boston

Compiled by Peter Y. Wong
Director of University Relations, Museum of Science, Boston

PROPERTIES

Intended Grade Band	✔ K–5	✖ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✔ Online PD (MOOC, etc.)	✖ Meetups	✔ Local Workshops	✔ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✖ Full Semester	✖ Full Year	✖ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✖ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✖ Physical Computing	✖ Networking and Security	✖ Mobile Programming	✖ Web Development	✔ Databases	✖ Game Design
Required Technology	✔ Desktop or Laptop	✖ iPad/iPhone	✖ Android Device	✖ Chromebook	✔ Works with Any Major OS	✖ Custom Hardware Device	
Required Software	✔ Browser Only	✖ Free Software	✖ Purchasable Software				

OVERVIEW

The Museum of Science, Boston, is creating a curriculum for grades K–5 that empowers students, teachers, and school communities to use CS to address authentic, creative, and technical challenges across disciplines. In collaboration with teachers, districts, and researchers at the DevTech lab at Tufts University, the Museum aims to enhance literacy and STEM achievement, broaden participation in CS, and strengthen the capacity for CS teaching and learning. This project is funded with support from the William and Charlotte Bloomberg Foundation.

The curriculum will comprise eight units that address different programming practices and types of problems. Each unit will require 6–10 classroom hours. The first two units—*A Fine Tune: Encoding Music and Sound to Compose a New Song* and *A Careful Plan: A Programming Approach to Making Care Packages*—should be available in June 2019.

Each unit is project-based, with a focus on authentic, creative, and technical applications of CS. The projects are related to the work of computer scientists and are easily usable by the students. Learning is anchored in meaningful, real-world contexts in order to give a broad understanding of the impact, value, and use of computing.

The units are aligned with the K–12 Computer Science Framework and the Mass. DLCS standards. Each unit also directly addresses specific national and state STEM, social studies, and English language arts standards.

The learning objectives in each unit focus on the following:

- Skills, knowledge, and habits of mind related to different aspects of abstraction and different programming practices (e.g., order and sequence, loops, conditionals, modules and functions, events, data representation and organization)
- Positive attitudes, perceptions, and beliefs toward CS
- Constructive social and emotional skills that allow students to work in groups and persist with problem solving

Unit development started in January 2017; it involves iterative cycles of classroom-based piloting and field-testing in a total of 38 classrooms in diverse districts throughout the United States. The activities, lessons, and assessments have been designed using culturally responsive approaches; a range of underrepresented audiences were engaged in all phases of feedback and development. This design approach allows the units to (1) support positive classroom cultural norms, (2) integrate meaningful collaboration and teamwork, and (3) allow students to see a diversity of people—including women, underrepresented minorities, and people with disabilities—in programming and CS roles. Universal Design for Learning principles guide the design and assessment of all curriculum materials.

To meet the varied needs of individual classrooms, lessons provide hands-on offline and online ways to engage in programming and CS. Activities and materials can be adapted for different levels and student populations. The curriculum is modular; units can be used as individual modules or sequenced within a grade or school.

RESOURCES

Professional Development (PD): Professional learning features being developed include integrated tips; online guides, activities, and planning tools; and in-person workshops and trainings priced similarly to the Museum's other curriculum product workshops.

Vendor or PD Provider Expectations: To be determined.

Ongoing Support: These resources are in development.

TECHNOLOGY

Required Hardware and Software and Their Costs: No software or hardware will be required.

Recommended Technology: A browser and Internet connection are needed for online activities.

Potential Technology Barriers: Reliable Internet access at schools.

Environments and Programming Languages Used: To be determined, but block-based programming is expected.

SUPPLEMENTAL MATERIALS

All required materials will be available online, either within the curriculum or linked from the curriculum. Recommended materials include materials kits (in development), an Ideabook (in development), and Curriculum student storybooks (in development).

PLTW LAUNCH
» Project Lead The Way

Compiled by Karine Laidley
Director of Curriculum Development, PLTW

PROPERTIES

Intended Grade Band	✔ K–5	✖ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✔ Online PD (MOOC, etc.)	✖ Meetups	✔ Local Workshops	✔ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✖ Full Semester	✖ Full Year	✔ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✔ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✖ Physical Computing	✖ Networking and Security	✔ Mobile Programming	✖ Web Development	✖ Databases	✔ Game Design
Required Technology	✖ Desktop or Laptop	✔ iPad/iPhone	✔ Android Device	✖ Chromebook	✖ Works with Any Major OS	✖ Custom Hardware Device	
Required Software	✖ Browser Only	✔ Free Software	✖ Purchasable Software				

OVERVIEW

The *PLTW Launch CS* curriculum comprises six 10-hour modules, one module per grade from Kindergarten to grade 5. These modules are part of a larger K–5 curriculum, *PLTW Launch*, which comprises 24 10-hour modules for grades K–5: 2 engineering modules, 1 biomedical science module, and 1 CS module for each grade. When you purchase *PLTW Launch*, you automatically have access to all 24 modules.

The PLTW curriculum uses an activity-, project-, and problem-based instructional approach, which offers a scaffold of learning experiences that students can then apply to solving an open-ended, real-world problem. Activities are designed for the acquisition of knowledge and skills and follow an appropriate learning progression for students. Projects provide opportunities for investigations and application of concepts and skills. Problems are designed to provide a common challenge that will typically result in unique solutions, which require the transfer of new and past knowledge and skills. Students who experience *PLTW Launch* engage in a curriculum that is hands-on and student-driven; as the teacher becomes a facilitator of learning, students learn to become independent learners. The *PLTW Launch* curriculum also teaches transferable skills, such as problem solving, persistence, creativity, collaboration, and communication

The six CS modules within *PLTW Launch* are as follows:

- Animals and Algorithms (Kindergarten): Starting with an unplugged activity, and moving on to animations and interactive games, students learn about the use of simple algorithms and collaboration as they create their own digital animations about animals and their habitats.
- Animated Storytelling (grade 1): Students explore the sequential nature of computer programs through hands-on activities, both with and without a computer. They examine key aspects of storytelling and learn how to transition a narrative from page to screen. Combining fundamental principles of CS with story-building skills, students develop animations that showcase characters, settings, actions, and events from short stories of their own creation.
- Grids and Games (grade 2): Students investigate numerical relationships while learning about the sequence and structure required in computer programs. Starting with computer-free activities and moving to tablet-based challenges, students apply addition and subtraction strategies to make characters move on a grid, as they work collaboratively to design and develop interactive games on a tablet.
- Programming Patterns (grade 3): This module introduces students to the power of modularity and abstraction. Starting with computer-free activities and progressing to programming in a blocks-based language on a tablet, students learn how to think computationally about a problem by decomposing it into smaller parts and then using functions and branching logic to solve it.
- Input/Output: Computer Systems (grade 4): In this exploration of how computers work, students are encouraged to make analogies between the parts of the human body and the parts that make up a computer. Students apply what they have learned to build their own reaction-time measurement apps on tablets. This module has strong connections to the *PLTW Launch* "Human Brain" biomedical science module.
- Infection: Modeling and Simulation (grade 5): Students investigate models and simulations and discover powerful ideas about computing. The design problem (which is related to the *PLTW Launch* "Infection: Detection" biomedical science module) has students model an infectious disease to simulate how an illness can spread through their class. Applying their new understandings, students program their own models and collect data by running simulations with different parameters.

PLTW Launch is part of the PLTW K–12 CS pathway. It is designed to fit into K–5 school schedules, as it integrates well with other disciplines and aligns to national standards, such as Common Core, NGSS, and CSTA.

RESOURCES

Professional Development (PD): *PLTW Launch* training includes three options:

- *Classroom teacher training:* This in-depth, collaborative experience prepares teachers to facilitate and deliver a transformative learning experience in their classrooms. After completing this training, teachers go through further online training that is specific to the modules they plan to teach.
  - Two-day training costs $500/teacher
  - Offered at affiliate university sites across the United States
- *Lead teacher training:* This is an enhanced train-the-trainer experience for teachers who will train other *PLTW Launch* teachers in their school or district; it is designed to develop instructional leaders and program champions.
  - Two-day training costs $700/teacher
  - PLTW will deliver this training at sites across the United States (an Online Core Training option is also available)
  - Prerequisite: Classroom teacher credential
- *District transformation training:* This option is for districts seeking to transform the learning experience system-wide by implementing *PLTW Launch* in multiple schools at once. PLTW will train a group of district teachers as classroom teachers or lead teachers, depending on the district's goals and implementation approach.
  - Two-day training at district site costs $7,500, plus expenses
  - Up to 24 participants per training session

Vendor or PD Provider Expectations: None

Ongoing Support: PLTW provides phone support to schools; the PLTW online community of teachers, master teachers, and PLTW staff for ongoing support; PLTW representation in all 50 states via PLTW's Director of

School Engagement; and PLTW online resources (including videos, tutorials, answer keys, and standards alignment documentation) to support teachers with content knowledge, facilitation, and assessment.

TECHNOLOGY

Required Hardware and Software and Their Costs:

- **Hardware:** Schools are responsible for purchasing tablets for their classrooms. One iPad or Android tablet for every two students, plus one for the teacher. Ideally, students work in pairs.
- **Software:** All apps used in the CS *PLTW Launch* modules are free or provide a free version for both iOS and Android.

The *PLTW Launch* curriculum is accessible online. If WiFi is not an option in the classroom, teachers can download the courses locally on the students' tablets ahead of time. Any videos that are available within the courses will need to be streamed on the teacher's computer. All PLTW courses are available as downloadable PDF files, if needed.

Recommended Technology:

- *Potential Technology Barriers:* The *PLTW Launch* curriculum is accessible on desktops, laptops, and mobile devices. However, most of the apps that students use run only on tablets, so having solely desktop or laptops is not sufficient.
- *Environments and Programming Languages Used:* ScratchJr, Hopscotch, and Tynker.

SUPPLEMENTAL MATERIALS

PLTW Launch offers a kit of all the supplies needed for each module, including both durable and consumable items.

The following table shows the cost of the materials needed for each CS *PLTW Launch* module. Note that this does **not** include the cost of tablets.

PLTW Launch (Grades K–5)	Year 1	Year 2	Year 3
K: "Animals and Algorithms" materials	$135	$60	$60
1: "Animated Storytelling" materials	$180	$60	$60
2: "Grids and Games" materials	$215	$60	$60
3: "Programming Patterns" materials	$220	$60	$60
4: "Input/Output Computer Systems" materials	$60	$60	$60
5. "Infection: Modeling and Simulation" materials	$60	$60	$60

OTHER NOTES

The fee for schools to purchase the 24 *PLTW Launch* modules is $750 per year. Schools may apply for funding and grants opportunities through the PLTW Grant program.

Note: There will be a grant opportunity specifically for Massachusetts public schools through the PLTW Grant program. Details should be available on the PLTW the website in mid-October 2017.

SCRATCHJR

» DevTech Research Group at Tufts University, the Lifelong Kindergarten Group at the MIT Media Lab, and the Playful Invention Company

Compiled by Meg Bednarcik
Instructor of Computer Science, Phillips Academy

PROPERTIES

Intended Grade Band	✔ K–5	✖ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✖ Online PD (MOOC, etc.)	✔ Meetups	✔ Local Workshops	✖ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✖ Full Semester	✖ Full Year	✖ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✔ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✖ Physical Computing	✖ Networking and Security	✖ Mobile Programming	✖ Web Development	✖ Databases	✔ Game Design
Required Technology	✔ Desktop or Laptop	✔ iPad/iPhone	✔ Android Device	✔ Chromebook	✔ Works with Any Major OS	✖ Custom Hardware Device	
Required Software	✔ Browser Only	✖ Free Software	✖ Purchasable Software				

OVERVIEW

ScratchJr is a free iOS, Android, and Chromebook application targeted to students in grades K–2. ScratchJr offers three stand-alone curricula that each comprise multiple lessons: *Animated Genres*, *Playground Games*, and *Reinforcing Math and Literacy*.

Animated Genres is divided into three modules. In the first module, students learn how to navigate around the ScratchJr interface, and they learn simple programming commands that animate their characters. In the second module, students learn how to define more specific behaviors for their characters, and they use tools such as speech bubbles and pages to help them create a more complex narrative. In the third module, students learn more advanced concepts in ScratchJr, such as how to program characters that interact with one another and with the user of the project.

The eight lessons in *Playground Games* provide an introduction to ScratchJr by recreating familiar children's games, such as tag, mini golf, and "monkey in the middle," using the ScratchJr characters and programming blocks.

Reinforcing Math and Literacy is designed to augment literacy and math standards. The three stand-alone modules do not build on previous ScratchJr lessons and can be woven into a K–2 literacy or math curriculum.

RESOURCES

Professional Development (PD): The Harvard Graduate School of Education team offers two primary forms of PD:

- The ScratchEd Online Community (http://scratched.gse.harvard.edu) is a venue for educators from around the world to share stories, exchange resources, and ask questions to support Scratch users in K–12 classrooms. Participation in the ScratchEd Online Community is free, and resources are licensed by Creative Commons.
- ScratchEd Meetups (http://meetups.gse.harvard.edu) are participatory professional learning experiences—opportunities for teachers to have hands-on , playful experiences with Scratch. Meetups are offered in cities across the country and are always free.

Vendor or PD Provider Expectations: None

Ongoing Support: Users can join a mailing list and receive updates about the curriculum.

TECHNOLOGY

Required Hardware and Software and Their Costs: To use ScratchJr, students need either iOS or Android devices or a Chromebook. The application can be downloaded onto their tablets and used offline.

Recommended Technology: A tablet, either Android or iOS, is recommended.

Potential Technology Barriers: Students are not able to access this tool through a browser on a PC.

Environments and Programming Languages Used: Scratch.

SUPPLEMENTAL MATERIALS

All necessary materials are available online (https://www.scratchjr.org/) as well as within the ScratchJr application on iOS or Android.

STEM+C INTEGRATED MODULES
» EDC & DESE

Compiled by Anne DeMallie
Computer Science and STEM Integration Specialist, DESE

PROPERTIES

Intended Grade Band	✔ K–5	✔ 6–8	✖ 9–12	✖ Other			
PD Model and Availability	✔ No Official PD	✖ Online PD (MOOC, etc.)	✖ Meetups	✖ Local Workshops	✖ Formal Institute and Follow-Up Sessions		
Curriculum Length	✔ Less than One Semester	✖ Full Semester	✖ Full Year	✖ Multi-Year			
Curriculum Integration	✖ Stand Alone Curriculum or Course	✔ Integrated Into Another Course					
Corresponding Pathways	✖ Software Engineering	✖ Physical Computing	✖ Networking and Security	✖ Mobile Programming	✖ Web Development	✖ Databases	✖ Game Design
Required Technology	✔ Desktop or Laptop	✖ iPad/iPhone	✖ Android Device	✖ Chromebook	✖ Works with Any Major OS	✖ Custom Hardware Device	
Required Software	✔ Browser Only	✖ Free Software	✖ Purchasable Software				

OVERVIEW

In partnership with DESE, EDC is developing and testing 17 STEM+C Integration Modules (STEM+C I-Mods) designed to integrate computational thinking into math and science lessons in grades 1–6.

Subject-matter experts in CS, science, and mathematics at the elementary school level compared Massachusetts and national math and science standards with the Mass. DLCS standards, identifying points of synergy where computational thinking could be integrated into these modules. Integration has been done on three levels:

- Concepts that *already exist* in the discipline and can be called out or presented with examples of how the concept can also relate to computational thinking
- Concepts that can be *enhanced* by adding tasks or lessons that provide a clear connection to computing concepts (where some distinction exists between the discipline and computational thinking)
- Concepts that can be *extended* by adding new lessons or sequences of lessons as a basis for computational thinking exploration, most likely involving programming activities

STEM I-Mods are being developed through a collaboration between EDC staff and Massachusetts elementary school teachers. Thus far, 10 I-Mods have been piloted in 27 classrooms in 10 schools across 10 districts in Massachusetts, and more than 70 teachers have participated in capacity-building PD and development of pilot project materials.

An online DLCS professional learning community led by DESE is being organized to support development, piloting, and implementation. I-Mods and other materials from this project will be available from the EDC and DESE websites at the conclusion of the project.

The I-Mods will be delivered in two forms: (1) modified curriculum units (existing, complete units written for the state as part of an earlier project, or district-developed units) withe computational thinking content built in, and (2) overlay units, consisting of teaching tips, lesson enhancements, and extension activities that connect the disciplinary content (mathematics and science) with computational thinking (including, in some cases, programming activities). As the I-Mods represent integrated materials, they are meant to replace or supplement existing mathematics and science units, and thus are developed to be used in whatever sequence the discipline demands.

Current I-Mods include *Light and Shadow* (grade 1); *Build It, Fix It* (grade 2); *Survival of Organisms* (grade 3); *Weathering and Erosion* (grade 4); *Solar System* (grade 5); and *Data and Statistics* (grade 6). As this project is still under development, the complete list of I-Mods is still in process.

New overlay modules may include *Money Machine* (grade 1 math); *Measurement* (grade 2 math); *Fractions* (grade 3 math); *Electricity* (grade 4 science); *Water Cycle, Solar System,* and/or *Plants Make their Own Food* (grade 5 science); *Number Fluency and Fractions* (grade 5 math); *Density, Human Body Systems*, and/or *Earth and the Universe* (grade 6 science); and *Number System* (grade 6 math).

Additional I-Mods developed in Year 1 will be posted under Integrated Units on EDC's website.

RESOURCES

Professional Development (PD): To be determined

Vendor or PD Provider Expectations: To be determined

Ongoing Support: DESE's online DLCS professional learning community will continue to support teachers who are piloting and implementing the I-Mods.

TECHNOLOGY

Required Hardware and Software and Their Costs: Some I-Mods require Internet access, student computers, a teacher computer, and a projector.

Recommended Technology: Each STEM I-Mod will have its own requirements and recommendations. Most activities can be adapted to use technology that is available and familiar to teachers.

Potential Technology Barriers: For some extension activities, specific hardware and/or software may be suggested, but in most cases these activities are optional.

Environments and Programming Languages Used: Each STEM I-Mod will have its own requirements.

MIDDLE SCHOOL CURRICULA AND TOOLS

BOOTSTRAP

» Bootstrap

Compiled by Kathi Fisler
Bootstrap Co-Director, and Research Professor of Computer Science, Brown University

Emma Youndtsmith
Midwest Regional Manager and Facilitator, Bootstrap

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✓ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✗ Software Engineering	✗ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

Bootstrap crafts research-based curricular modules that integrate CS into other disciplines for grades 6–12. Each introductory module can be taught as part of an existing class or as part of a stand-alone CS course. The introductory modules are designed for non-CS teachers, who often have no programming experience prior to the three- to four-day PD workshops. Bootstrap strives to help both teachers and students grow in their understanding and use of computing and its connections to other disciplinary material. By integrating computing into existing courses, Bootstrap supports schools that lack the time, resources, and/or personnel for dedicated computing courses, while also providing equitable access to computing education for all students. Bootstrap is one of the largest providers of formal CS education to girls and underrepresented students nationwide.

The four Bootstrap modules can be combined into longer courses as needed:

- Bootstrap:Algebra (the best-known module) is a 20–25-hour introductory module that integrates computer programming and algebra. Students program a video game of their own design using linear functions, piecewise functions, the Pythagorean theorem, inequalities, nested functions, and more. The module focuses on solving word problems: it introduces a systematic process for breaking down complex

algebraic word problems into smaller steps in a way that is applicable to both math classes and higher-level software design methodologies.

Bootstrap has formally evaluated student learning gains in *Bootstrap:Algebra* using function composition and word problems taken from grades 7 and 8 MCAS exams. More details can be found in the "Impact" section of the Bootstrap website (www.bootstrapworld.org). *Bootstrap:Algebra* is aligned to the Common Core, CSTA standards, and Mass. DLCS standards.

As programming and STEM education continue to flourish, many curricula offer ways to teach programming alongside math. However, many popular programming languages actually undermine math instruction with variables that can be reassigned, functions that do not pass the vertical line test, and more. In contrast, *Bootstrap:Algebra* uses a mathematical, functional programming language that *reinforces* the way math is traditionally taught, and introduces problem-solving strategies appropriate to both algebra and software development.

- Bootstrap:Data Science is a 25–30-hour introductory module that teaches students to view programs as questions we ask of data—for example, *What factors make some people live longer than others?* and *Are the schools in one part of your neighborhood better than schools in another?* Answering questions such as these involves collecting and manipulating data, from sports statistics to record sales to census data. Students form their own questions about the world around them, and learn how to write small programs to analyze data critically and carefully to find answers. Activities include identifying the data required for a problem, writing short programs to manipulate tables of data (extracting or combining information), distinguishing different kinds of data (categorical, etc.), creating scatterplots and other graphical displays of data, computing basic statistics on data, and general computing-oriented data practices, such as cleaning data and testing computations on small datasets. The curriculum uses Google Sheets as a data source, so students can access data from many sources, or perform data analysis on surveys they conducted through Google Forms.

  The module contains roughly 25 hours of content and can be tailored to different needs and contexts. Business, science, and social studies teachers can use it to help students make inferences from data. Math teachers can use it to introduce foundational concepts in statistics. It also works well as a module for the AP Computer Science Principles unit on data. The module does not assume that teachers have programming experience prior to the PD.

- Bootstrap:Reactive is a 25+-hour module that builds on *Bootstrap:Algebra*. It goes deeper into programming, building events and data structures on top of the foundation laid by *Bootstrap:Algebra* and allowing students to build far more sophisticated programs. Students learn how to use data structures and event-based programming to create animations that feature two-dimensional character movement, multiple attributes of characters (such as location, orientation, and size), and more characters than in *Bootstrap:Algebra* animations. Whereas the *Bootstrap:Algebra* game template directs students to create specific, pre-designed functions, *Bootstrap:Reactive* does not involve a prescriptive game template; it allows students to design and write their own unique games and animations from scratch. Students learn to separate the visual appearance of programs from their underlying data manipulations. The course project can be tailored to individual classes or students.

  Bootstrap:Reactive is a CS module more than an algebra module. While most teachers who have taken the PD are math teachers from *Bootstrap:Algebra*, the content does not reinforce as many math concepts as *Bootstrap:Algebra*. The curriculum is aligned to the CSTA standards, the K–12 CS Framework, and the Mass. DLCS standards.

- Bootstrap:Physics integrates Bootstrap's approach to creating simulations with *Physics First*, a nationally deployed modeling-based physics curriculum for ninth-graders. *Bootstrap:Physics* is a joint project of the American Association of Physics Teachers, the American Modeling Teachers Association, and Bootstrap. This module is deeply integrated with a full-year *Physics First* curriculum, and is less of a stand-alone module than Bootstrap's other modules. Students learn how to identify the dynamic elements in a physics problem, write algebraic functions to capture how dynamic elements change over time, and create a visual simulation from information about static and dynamic elements. Physics topics covered include constant velocity, uniform acceleration, balanced and unbalanced forces, momentum, force fields, waves, and quantitative energy.

The *Algebra, Data Science*, and *Reactive* modules can be combined in various ways to form a full stand-alone CS course. Bootstrap staff consult with districts and teachers to help tailor the materials to their needs.

RESOURCES

Professional Development (PD): Since 2011, Bootstrap has offered on-demand PD workshops for school districts across the country. The PD generally lasts three days (*Data Science* PD is four to five days), and covers the entirety of one module in a hands-on, engaging manner. Workshop costs and locations are negotiated on a district-by-district basis. PD workshops can accommodate 25–40 teachers.

Vendor or PD Provider Expectations:

- There are no prerequisites for attending a *Bootstrap:Algebra* or *Bootstrap:Data Science* workshop, both of which are designed for teachers with no prior programming experience, though they do assume the ability to type and familiarity with using a Web browser. For *Bootstrap:Algebra*, familiarity with middle/high school Algebra 1 material is preferred.
- *Bootstrap:Reactive* workshops are open to teachers who have taught *Bootstrap:Algebra*. Ideally, teachers will have taken a *Bootstrap:Algebra* PD workshop, as *Reactive* builds on the same pedagogy taught in that workshop. Teachers from *Bootstrap:Data Science* should consult Bootstrap before taking the *Reactive* PD.

Ongoing Support: Bootstrap provides access to an active discussion group, moderated by Bootstrap staff and master teachers, where users ask questions and share resources. Bootstrap staff are also available for personal email and phone support (contact@bootstrapworld.org).

TECHNOLOGY

Required Hardware and Software and Their Costs: Implementation of Bootstrap modules requires access to (at least) one computer (desktop, laptop, or Chromebook) per pair of students, with access to Chrome, Firefox, Safari, or IE9+, and a Google Account (one per pair, OR one per class). All programming materials run in the browser; no software installation is required. Software licenses are built into the cost of PD. The modules do **not** require consistent access to computers, as every module also includes a substantial "unplugged" component on paper. *Environments and Programming Languages Used: Bootstrap:Algebra* uses a browser-based editing environment for the Racket programming language (see www.wescheme.org). *Bootstrap:Data Science* and *Bootstrap:Reactive* use a browser-based editing environment for the Pyret programming language (see code.pyret.org), designed by members of the Programming Languages group at Brown University (which overlaps the broader Bootstrap team) with a specific eye toward teaching beginner and intermediate-level programming. Both tools are freely accessible.

SUPPLEMENTAL MATERIALS

Each module requires a student workbook for each student (available as a free PDF or a softcover book for purchase), and a writing utensil. Recommended instructional materials are a projector, a whiteboard, and markers.

Compiled by Jennifer Rosato
Assistant Professor, Computer Information Systems, College of St. Scholastica, and Co-PI, Mobile CSP

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✓ 9–12	✓ Other			
PD Model and Availability	✓ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✓ Web Development	✗ Databases	✗ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

Codecademy provides online, self-paced mini-courses in Web development, programming in various languages, data analytics, and other CS areas. Most courses are designed to be completed within 3 to 10 hours by individual students. Each course comprises several lessons that each contain a series of exercises. The exercises include a Learn section, which includes the content of the exercise, and an Instructions section, which are the specific steps students should take to practice the content. Also included is an interactive area for writing and testing a code specific to each exercise, and links to a community forum and help pages. The code must run successfully in one exercise before advancing to the next one.

Course topics in the Web development track appropriate for grades 6–12 are HTML, CSS, JavaScript, Make a Website, and Deploy a Website. Programming courses include HTML, CSS, JavaScript, Python, and Java. Other courses (e.g., using the Watson API, SQL, and Git) may be more appropriate for advanced students who have completed several other courses or who have previous knowledge in an area.

Codecademy does not provide a complete curriculum for a course, though several courses in the Web development area could be used together to create a Web development pathway. The courses are especially helpful as an introduction to new programming languages or a review of basic concepts. Students can complete courses independently, which can be a good option for those who

want to explore CS concepts beyond the scope of their current course. For example the "Learn Java" course could be used by students as review in the AP Computer Science A course, or used toward the end of an AP Computer Science Principles course to explore other programming languages.

The site does not provide tools for teachers to create classes and monitor student progress. Students can earn points and badges for completing lessons and courses.

The courses are available on the Codeacademy website (www.codecademy.com).

RESOURCES

Professional Development (PD): None

Vendor or Provider PD Expectations: N/A

Ongoing Support: A robust online community is available with a free account. Upgrading to a Pro account includes access to an advisor via live chat, as well as learning paths, quizzes, and projects.

TECHNOLOGY

Required Hardware and Software and Their Costs: All courses run within the browser on a laptop or Chromebook and are accessible for free after creating an account.

Recommended Technology: The recommended browsers are Chrome and Firefox.

Potential Technology Barriers: A stable Internet connection is required, as content is not available offline.

Environments and Programming Languages Used: Each course uses a browser-based development environment specific to Codecademy. Languages available include HTML, CSS, JavaScript, Angular, React, Ruby, Ruby on Rails, SQL, Java, and Python.

CREATIVE COMPUTING CURRICULUM

» ScratchEd Team at the Harvard Graduate School of Education

Compiled by Laura Peters
Research Project Manager, Creative Computing Lab, Harvard Graduate School of Education

Karen Brennan
Associate Professor, Harvard Graduate School of Education

PROPERTIES

Intended Grade Band	✓ K–5	✓ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✓ Meetups	✓ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

CS and computing-related fields have long been introduced to young people in a way that is disconnected from their interests and values—emphasizing technical detail over creative potential. The *Creative Computing Curriculum* (CCC) (http://scratched.gse.harvard.edu/guide) supports the development of personal connections to computing by drawing on students' creativity, imagination, and interests, while simultaneously cultivating fluency with the knowledge, practices, and fundamental literacies that students need in order to create the types of dynamic and interactive computational media they enjoy in their daily lives. Engaging in the creation of computational artifacts prepares students for more than careers as computer scientists or programmers: it supports their development as computational thinkers who can draw on computational concepts, practices, and perspectives in all aspects of their lives, across disciplines and contexts.

CCC uses the Scratch programming environment as its central tool. Scratch (http://scratch.mit.edu) is a free computer programming language and a vibrant online community developed by researchers at the MIT Media Lab. With Scratch, users can create a wide variety of interactive media projects—animations, stories, games, and more—and share those projects with others in the online community. Students from Kindergarten through college around

the world have created and shared more than 24 million projects in the 10 years since Scratch's launch in 2007.

CCC is a collection of ideas, strategies, and activities for an introductory creative computing experience. The activities are designed to foster familiarity and increase fluency with computational creativity and computational thinking. The *Creative Computing Curriculum Guide* is organized into seven units: an initial preparatory unit and six thematic units (including animations, stories, and games). CCC culminates in a larger self-directed project. The curriculum can be used in its entirety as a semester-long computing course, or can be selectively integrated into curricular areas (e.g., math, language arts).

CCC was developed with support from Google, NSF, and the Scratch Foundation.

RESOURCES

Professional Development (PD): The Harvard Graduate School of Education team offers two primary forms of PD:

- The ScratchEd Online Community (http://scratched.gse.harvard.edu) supports Scratch educators working in K–12 classrooms. It is a venue for educators from around the world to share stories, exchange resources, and ask questions. Participation in the ScratchEd Online Community is free, and resources are licensed by Creative Commons.
- ScratchEd Meetups (http://meetups.gse.harvard.edu) are participatory professional learning experiences—opportunities for teachers to have hands-on experience with Scratch. Meetups are offered in cities across the country and are always free.

Vendor or PD Provider Expectations: CCC was released under a Creative Commons Attribution-ShareAlike license. Educators are completely free to use, change, and share this work, as long as appropriate attribution is given and others may have access to any derivative works.

Ongoing Support: Both types of PD provided offer ongoing support.

TECHNOLOGY

Required Hardware and Software and Their Costs: CCC requires computers with headphones for the computer-based design activities; an Internet connection to connect to Scratch online; and a projector or interactive whiteboard with speakers for sharing work in progress and for demonstrations. If your environment does not offer an Internet connection, a downloadable version of Scratch is available.

Recommended Technology: Computers equipped with microphones and webcams are recommended.

Potential Technology Barriers: Scratch 2.0 is not compatible with iPads or other tablets; a Flash-enabled device is required. The next major version of Scratch, Scratch 3.0, will be available on tablets. ScratchJr, a version of Scratch for younger students, is also available on tablets.

Environments and Programming Languages Used: Scratch 2.0

SUPPLEMENTAL MATERIALS

CCC encourages students to maintain a design notebook, either physical or digital.

OTHER NOTES

CCC has been translated by teachers into 10 different languages. The next version will be available in Summer 2018, coinciding with the release of Scratch 3.0.

COMPUTER SCIENCE DISCOVERIES

» Code.org

Compiled by Chad McGowan
Computer Science Teacher and Technology Coach, Ashland High School, and Code.org Facilitator

PROPERTIES

Intended Grade Band	✖ K–5	✔ 6–8	✔ 9–12	✖ Other			
PD Model and Availability	✖ No Official PD	✖ Online PD (MOOC, etc.)	✖ Meetups	✔ Local Workshops	✔ Formal Institute and Follow-Up Sessions		
Curriculum Length	✖ Less than One Semester	✔ Full Semester	✔ Full Year	✖ Multi-Year			
Curriculum Integration	✔ Stand Alone Curriculum or Course	✖ Integrated Into Another Course					
Corresponding Pathways	✔ Software Engineering	✖ Physical Computing	✖ Networking and Security	✔ Mobile Programming	✔ Web Development	✖ Databases	✔ Game Design
Required Technology	✔ Desktop or Laptop	✖ iPad/iPhone	✖ Android Device	✔ Chromebook	✔ Works with Any Major OS	✔ Custom Hardware Device	
Required Software	✔ Browser Only	✖ Free Software	✖ Purchasable Software				

OVERVIEW

Computer Science Discoveries (CSD) is an introductory CS curriculum that empowers students to create authentic artifacts and engage with CS as a medium for creativity, communication, problem solving, and fun. The CSD curriculum and tools are available online (and downloadable, in many cases) for free, and are updated as needed.

The lessons are designed to move students through units in a logical and growth-oriented direction. Units contain a mix of unplugged and plugged lessons that help illustrate and teach important introductory CS concepts through hands-on exploration. The unique learning environment provides students with easy access to online lessons, while providing full lesson plan and curriculum support to teachers, including activity guides, videos, online development environments, and rubrics.

Course units include "Problem Solving," "What Is a Computer?," "Introduction to HTML and CSS," "Introduction to JavaScript Through Animation and Games," "The Design Process," "Data and Society," and "Introduction to Physical Computing with the Adafruit Circuit Playground."

RESOURCES

Professional Development (PD): PD is delivered in two formats:

- Code.org hosts yearly national events called TeacherCons, which are designed for large regions that are new to Code.org PD and are not supported by regional partners.
- Code.org-trained and approved facilitators run local summer institutes for one week, with ongoing quarterly workshops throughout the school year.

Each type of PD is free and open to all, but space is limited and applications are used to determine acceptance into a program.

Vendor or PD Provider Expectations: Acceptance into Code.org-approved PD for CSD is prioritized by multiple factors. Administration support for the applying teacher is a strong priority. Ensuring that the full course will be offered in the following school year is also a high priority. Additional factors to be considered include recruitment and enrollment strategies, and demonstrated understanding of the importance of CS in schools.

Ongoing Support: The curriculum includes detailed lesson plans, activity guides (in multiple formats), project-based summative assessments, video tutorials throughout the course, rubrics, and other teacher-facing resources. Teachers are provided with a portal through which they can manage classes (using Google Classroom), review student work, and view assessment questions and responses.

Additionally, there is a large and active online community forum in which every aspect of the course is open to discussion, and there are active forum monitors who ensure that all questions are responded to in a prompt manner. All materials have been placed online with a Creative Commons license.

TECHNOLOGY

Required Hardware and Software and Their Costs: The CSD curriculum mixes unplugged and plugged lessons. Plugged lessons require an Internet connection and a Chromebook or better computer capable of using a full browser. Additionally, the final curriculum unit uses the Adafruit Circuit Playground (one per two students), which is a ~$20 circuit board with many built-in sensors and lights.

Recommended Technology: The recommended browser is Chrome, which works well on most high-speed connections. In cases of low bandwidth issues, watching the provided videos on a shared projector may be advised over having all students loading and watching videos. A school that has Google for Education will be able to have students and teachers seamlessly log in to Code.org.

Potential Technology Barriers: This program is not fully compatible with iPads and is not designed for their use. All videos and resources can be downloaded if resources such as YouTube are not available in the school setting. The curriculum is heavily dependent on a consistently functioning Internet connection after Unit 1.

Environments and Programming Languages Used: The environment is completely self-contained on Code.org. Three primary programming environments are used throughout the year, all within the Code.org website. Unit 2 uses a built-in Web editor to provide a development environment for HTML and CSS. Unit 3 employs a built-in development tool called Game Lab that uses JavaScript with a drag-and-drop or text-based option. Unit 6 uses App Lab, a slight variation on Game Lab, as well as the Adafruit Circuit Playground.

SUPPLEMENTAL MATERIALS

All necessary materials are listed online.

OTHER NOTES

This program was designed with grades 7–9 in mind, though it can be adapted to full middle school or high school implementation. Additionally, it was designed with a semester approach in mind so that it can be delivered over multiple years (year 1: Units 1, 2, and 3; year 2: Units 4, 5, and 6). Unit 6 integrates a physical computing device designed to limit expenses while introducing students to this strand of CS.

EDISON ROBOTS
» Microbric

Compiled by Jessica Jarboe
Teacher, Milton High School

PROPERTIES

Intended Grade Band	✓ K–5	✓ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✓ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✗ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✗ Software Engineering	✓ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✗ Game Design
Required Technology	✗ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✗ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

The affordable Edison robots are designed to be integrated into K–12 classrooms. Some of their functions are pre-programmed, and others can be created by programming in EdWare, EdBlocks, or EdPy software. The robots are compatible with LEGO bricks, allowing students to design robot structures with LEGOs. Ten EdWare lessons, each approximately 90 minutes, have been developed and are available on Edison's website (meetedison.com). Edison Robotics is currently working on more lessons.

Also available online are three free Edison educational books (EdBooks)—digital books written to support Edison Robotics. The EdBooks are fully illustrated, taking students through the three main concepts of controlling, programming, and building robots. EdBooks are available in 14 languages.

EdBook1 provides activities that don't require programming. It includes bar codes that can be scanned in so students can complete activities. The activities use the light sensor and clap detector on the robot.

EdBook2 offers 10 lessons that guide students through programming Edison robots using EdWare:

- Lesson 1: Get Familiar and Set Up (technology skills)
- Lesson 2: Robot Movement—Driving (an introduction to sequential programming)

- Lesson 3: Robot Movement—Turning (sequential programming and basic geometry)
- Lesson 4: Maze Challenge and Mexican Wave (a choreographed dance that reinforces learning)
- Lesson 5: Design Brief 1—My Program (creative thinking and problem solving)
- Lesson 6: Clap Sensing (an introduction to inputs [sensors])
- Lesson 7: Detect Obstacles (an introduction to the concept of obstacle detection and artificial intelligence)
- Lesson 8: Line Sensing and Tracking (industrial-like robotic behavior)
- Lesson 9: Respond to Light (environmental measurement and programming mathematics)
- Lesson 10: Design Brief 2—My Program (creative thinking and problem solving)

EdBook3 focuses on design. As Edison robots are compatible with LEGO pieces, EdBook3 describes how to build an EdDigger and EdPrinter.

RESOURCES

Professional Development (PD): None

Vendor or PD Provider Expectations: N/A

Ongoing Support: Edison offers an online forum for educators to share their experiences and offer support (www.meetedison.com/forum.) The website includes how-to videos and a troubleshooting section.

TECHNOLOGY

Required Hardware and Software and Their Costs: Desktops or laptops are required. Edison robots cost $49 each, with discounts on multiple kits. The software is free and accessible on the browser. One end of the cable plugs into the 3.5 mm (1/8 inch) audio output of your computer (included in the kit); the other end plugs into the underside of the Edison robot. This communication method is compatible with Mac, Linux, Windows, iOS, and Android operating systems.

Recommended Technology: All the programming languages (EdWare, EdBlocks, and EdPy) are free and can be accessed via a browser; no additional software is needed.

Potential Technology Barriers: The software is browser-based, so any interruptions or lack of WiFi may be a barrier. Edison robots each require four AA batteries.

Environments and Programming Languages Used: Edison robots use EdWare (a hybrid graphical programming language), EdBlocks (a blocks-based programming language), and EdPy (a text-based programming language). More languages may be coming soon.

SUPPLEMENTAL MATERIALS

All required materials are available online at the Edison website (https://meetedison.com).

OTHER NOTES

"Unit 6: Robotics" of the Exploring Computer Science curriculum uses Edison robots (exploringcs.org).

Lessons have been aligned with the Australian Curriculum (https://meetedison.com/content/Lesson-plans/Lesson-plans-links-to-Australian-Curriculm.pdf).

FINCH ROBOT

» BirdBrain Technologies

Compiled by Paula Moore
Computer Science Teacher, Westwood Public Schools

PROPERTIES

Intended Grade Band	✓ K–5	✓ 6–8	✓ 9–12	✓ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✗ Meetups	✓ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✗ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✗ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

The Finch robot, designed by Carnegie Mellon University's CREATE lab, is self-contained, with motorized wheels in a hard plastic shell. It includes motors, sound, multiple sensors for input, and a multi-color LED light on its nose.

The Finch comes with built-in light, temperature, obstacle detection (using infrared), and accelerometer sensors, which can be used to collect data, provide input to a program on the computer, or control the actions of the Finch itself as it interacts with its environment. Two attached motors can be programmed to allow the Finch to turn and to move forward and backward. The Finch's LED nose has red, green, and blue elements that allow for variations of color. It has sound output through a buzzer with programmable frequencies to produce sounds ranging from alarms to music. The Finch programming environment also provides a text-to-speech component that can play speech through the computer speakers.

The robot can be programmed using more than a dozen programming languages—from simple drag-and-drop languages (e.g., Scratch and Snap!) that are ideal for elementary and middle school-age children, to more complex text-based languages (e.g., Java and Python) for use in high school. The Finch website (FinchRobot.com) provides recommendations on languages for different age levels and installation instructions for each programming language.

The Finch robot must be tethered to a computer at all times via a 15-foot USB cable (included with the Finch), and therefore it functions best with a laptop or Chromebook. Desktops may be used with the Finch if there is enough space near the desktop for the robot to move.

RESOURCES

Professional Development (PD): BirdBrain Technologies can provide both onsite and online PD on request.

Vendor or PD Provider Expectations: None

Ongoing Support: The Finch website offers lessons and activities for grades K–12. For grades K–8, the Finch website offers 36 lessons and activities using Snap! and Scratch, which are broken down into three levels: Beginner, Intermediate, and Advanced. For grades 9–12, the Finch website offers lessons and activities broken down into content areas, such as Decisions, Loops and Recursion, and Arrays and Lists. Sample solution code is available to teachers when they register for a free teacher account. Additional content and project alignment is provided to support classroom instruction for an AP Computer Science A course using the Finch.

BirdBrain Technologies provides a loaner program each year to allow schools to borrow a classroom set of Finch robots. The application period ends by mid-May for loans occurring the following school year.

TECHNOLOGY

Required Hardware and Software and Their Costs: The Finch works with laptops or desktops using Windows OS from XP to 10, Mac OS 10.6.8 and up, Linux Ubuntu, and Chromebooks using ChromeOS. Finch robots retail at $99 but can be purchased with a 10% education discount.

Recommended Technology: The Finch robot is most flexible when used with a laptop or Chromebook. While the Finch works with a desktop, a desktop will limit the movement of the Finch to its 15-foot tether length.

Potential Technology Barriers: Chromebooks must have reliable wireless connection to enable access to the software via the Web. Windows OS, MacOS, or Linux Ubuntu laptops must be light enough for students to carry while following the Finch as it moves about the room.

Environments and Programming Languages Used: The Finch website suggests the following age groupings for the languages they support:

Grades K–2	Grades 3–6	Grades 7–9	Grades 10+
Snap! Level One Snap! Level Two	CREATE Lab Visual Programmer Snap! Level Three Snap! Level Four Scratch	Python/ Jython Processing Snap! Level Four Scratch	Java JavaScript Greenfoot Python Processing Scala C C++ C# Visual Basic Snap! Level Four

KHAN ACADEMY COMPUTING

» Khan Academy

Compiled by Patricia Clark
Teacher, Nashoba Regional High School

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✓ 9–12	✓ Other			
PD Model and Availability	✓ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✓ Full Year	✓ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✓ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✓ iPad/iPhone	✓ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

Khan Academy offers a variety of free, easily accessible lessons, videos, and text-based resources at all levels. This curriculum can also take students to more mature levels of understanding by offering resources related to object-oriented programming, interactive webpage development, simulation and game development, information theory, algorithms, and cryptography.

The Khan Academy curriculum known as Computing is browser-based and runs on all popular platforms. It consists of four modules: "Computer Programming," "Computer Science," "Animation," and "Hour of Code." Lessons from these modules can be selected and combined to support the classroom in a variety of modes:

- As a stand-alone course (lasting weeks, a semester, and/or full year)
- As a "base curriculum" that can be extended
- As enrichment for other CS curricula, such as AP Computer Science Principles
- For independent study
- As cross-curricular enrichment and for demonstrating concepts in science and math
- As resources for students learning college-level CS

The modules are described in more detail below.

- Computer Programming: The most popular and "core" curriculum in this module is Intro to JS (Java Script)

using Processing.JS language. These lessons, aimed at the middle school level, help students learn how to code in a text-based language while creating fun drawings and animations. *Intro to JS* contains about 15 hours worth of curriculum and teaches students all the basics of JavaScript programming, up to object-oriented design. The JavaScript course comprises three types of materials:

- Talk-throughs: These five-minute videos present code on the left, output on the right. Narration is given as new code gets displayed on the screen. The student can pause the talk-through at any point, change the code, and see the new output right away, which encourages interactive learning.
- Challenges: These interactive exercises are used to assess students' understanding of the concepts just taught. There is one challenge for every talk-through.
- Projects: These provide opportunities for students to be creative with the concepts they've just learned. Students are given a general set of guidelines, but they can take them in their own directions.

Khan Academy provides various tools and dashboards for teachers to track their students individually or as a whole class. Teachers can access various statistics on students' progress with challenge completion (students earn "badges" for completed projects) and talk-through watching. An online gallery is provided for each teacher to showcase all the projects created by students. An online set of resources gives helpful information to programming teachers.

Additional topics and features in the Computer Programming module are as follows:

- Introduction to HTML and CSS, SQL, Game Programming, and Natural Simulation
- Advanced JavaScript, JQuery, and Object-Oriented JS
- Comprehensive Documentation and Online Community Access

- Computer Science: This module addresses students who are seeking more in-depth understanding of CS. It acts as a repository for a number of video explanations and some interactive activities. It also includes a significant amount of material presented in textbook fashion that requires good technical reading skills. This section of the curriculum contains challenging mathematical content. Teachers could use this material as background for developing more accessible lessons for their students. Topics include algorithms, cryptography, information theory, and the Internet.
- Computer Animation: *Pixar in a Box* is a behind-the-scenes look at how Pixar artists do their jobs. Students animate bouncing balls, build a swarm of robots, and make virtual fireworks explode. This collaboration between Pixar Animation Studios and Khan Academy is sponsored by Disney.
- Hour of Code: This module comprises three units:
  - *Hour of Drawing:* Students learn how to program drawings using JavaScript by designing a snowman.
  - *Hour of Webpages:* Students learn how to make webpages with HTML tags and CSS. Their final project is to make a Web-based greeting card.
  - *Hour of Databases:* Students learn how to manipulate data in a database and make a custom store.

RESOURCES

Professional Development (PD): None

Vendor or PD Provider Expectations: None

Ongoing Support: Khan Academy offers a robust online community, videos, tutorials, and support from Khan Academy team members.

TECHNOLOGY

Required Hardware and Software and Their Costs:
No specialized hardware or software is required. This curriculum and platform work on modern browsers (Chrome, Firefox, Safari, IE9+). It also works on the iPad, but students may find that it's not as enjoyable to type on that keypad.

Recommended Technology: Headphones or earbuds are useful for students to view videos in a classroom setting.

OTHER NOTES

- The curriculum is mostly translated to Spanish, with interactive subtitles (not dubbing) for the talk-throughs. There is partial translation for Portuguese, Hebrew, Polish, and French.
- This curriculum is usable by deaf students. There is a transcript option available for the talk-throughs, the only aspect of the curriculum with audio.

LEGO MINDSTORMS EV3

» LEGO

Compiled by David C. Petty
Teacher, Winchester High School, and Co-President, CSTA-Greater Boston

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✗ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✗ Meetups	✓ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✗ Game Design
Required Technology	✓ Desktop or Laptop	✓ iPad/iPhone	✓ Android Device	✓ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✗ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

The LEGO Mindstorms EV3 CS curriculum is a set of nine coding activities that use the third-generation LEGO Mindstorms computing platform (EV3) to exemplify the big ideas of CS. The aptly named EV3 "intelligent brick" is a programmable device that can be connected to sensors, actuators, and traditional LEGO bricks. Activities are organized around using the EV3 to solve problems for autonomous vehicles.

The EV3 began its existence as a base for robotics and is still used in that context. The basic EV3 kit includes the EV3 intelligent brick, five sensors (gyro, ultrasonic, color, and 2x touch), three actuators (servo motors), assorted LEGO bricks, and various cables and chargers.

The *EV3 Coding Activities* curriculum requires that students have access to the EV3 and an external computer, Chromebook, or tablet on which to develop code.[3] Each activity includes detailed instructions (with sections titled *Connect, Construct, Contemplate,* and *Continue*), possible solutions, rubrics for assessment, and links to the NGSS and CSTA CS standards. Each activity has specific

3 The LEGO Mindstorms support page states, "You can easily program basic tasks on the EV3 Brick." However, on-brick programming will be inadequate for all but the most basic code, so an external device will be required in most cases.

objectives that cover many of the Seven Big Ideas of Computer Science (creativity, abstraction, data, algorithms, programming, input and output devices, impact). The rubrics are designed to help assess decomposition, generalization, algorithmic thinking, evaluation, and abstraction.

The curriculum provides an eight-page introduction to STEM and computational thinking. The nine *EV3 Coding Activities* focus on creative problem solving regarding autonomous vehicles, including autonomous parking, reversing safely, automatic headlights, line detection, object detection, unlocking a car, cruise control, roaming vehicles, and autonomous intersection. They are stand-alone CS activities that can make up a sequence or be incorporated individually into a CS, robotics, or engineering curriculum. Each activity includes detailed curriculum links to the NGSS and CSTA CS standards.

In addition to the *EV3 Coding Activities*, LEGO Mindstorms offers EV3 curricula titled *EV3 Design Engineering Projects*, *EV3 Science Curriculum*, and *EV3 Space Challenge Curriculum*, which include 30+ hours of activities incorporating EV3 coding. These EV3 curricula include projects that are ready to load in the Mindstorms EV3 integrated development environment, and can be incorporated into a robotics, science, or engineering curriculum.

RESOURCES

Professional Development (PD): The LEGO Education Academy offers two types of professional learning for teachers: face-to-face courses and free eLearning programs:

- There are beginner and intermediate face-to-face EV3 courses, each seven hours. According to the LEGO Education Academy, the cost of face-to-face training programs is "matched to the needs and requirements of your school. This is based on a detailed dialogue between the school and the LEGO Education trainer prior to the training."
- There are 15 free EV3 eLearning programs covering *getting started, programming and data*, and *in the classroom*.

Vendor or PD Provider Expectations: None

Ongoing Support: LEGO does not have an official EV3 support forum. However, like any technology with a near 20-year history, there is a robust community of users (many of whom are educators), including groups specific to certain topics (Scratch, robotics, RobotC, LEGO in general) and social media groups.

TECHNOLOGY

Required Hardware and Software and Their Costs: The EV3 costs $350 and is rarely discounted. Typical classroom use is one basic EV3 kit for every one to four students. In addition, a laptop, desktop, tablet, or Chromebook is required to develop code and download it to the EV3 intelligent brick. All required cables are provided with the basic EV3 kit.

Recommended Technology: LEGO provides a free LEGO Mindstorms EV3 integrated development environment to develop code for the EV3 intelligent brick, which supports a wide variety of options, including Mac OS X, Windows (7–10, including touch devices), Chromebook, iPad, Android, and Fire.[4]

Potential Technology Barriers: Users or administrators must have sufficient privileges to install the LEGO Mindstorms EV3 integrated development environment or the EV3 Programmer App on the development platform (e.g., computer, tablet, Chromebook). No other special privileges or hardware, beyond what is included in the basic EV3 kit, are required.

Environments and Programming Languages Used: The Mindstorms EV3 integrated development environment includes a block-based language for developing code for the EV3. In addition, RobotC, a third-party text-based language developed at Carnegie Mellon University, is available in free and commercial versions. For users of LabVIEW, National Instruments has a free add-on LabVIEW module for LEGO Mindstorms to control and program an EV3 with LabVIEW.

SUPPLEMENTAL MATERIALS

The *EV3 Coding Activities* do not require materials beyond those readily available in most middle school classrooms; the Automatic Headlights activity calls for a light source (e.g., a flashlight).

4 The LEGO Mindstorms support page states, "The free EV3 Programmer App—available on the App Store and from Google Play—allows you to program your robots from your tablet via Bluetooth. Compared to the programming software for PCs and Macs, the app is more simple to use and does not include some of the more advanced programming features such as data blocks and calculations."

OTHER NOTES

The following websites offer additional background and useful information:

- LEGO Mindstorms history (http://hackeducation.com/2015/04/10/mindstorms)
- EV3 support (https://www.lego.com/en-us/mindstorms/support)
- FIRST LEGO League (http://www.firstlegoleague.org/)
- LEGO Education Academy, which offers face-to-face courses and eLearning (https://education.lego.com/en-us/training)
- RobotC Intermediate Programming (http://education.rec.ri.cmu.edu/roboticscurriculum/robotc-intermediate-programming-ev3/)
- National Instruments LabVIEW Module for LEGO Mindstorms (http://sine.ni.com/nips/cds/view/p/lang/en/nid/212785)

MICRO:BIT'S INTRO TO CS

» Douglas and Mary Kiang, Microsoft

Compiled by Terry Dash
Teacher, Arlington Public Schools

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✗ 9–12	✗ Other			
PD Model and Availability	✓ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✗ Networking and Security	✓ Mobile Programming	✗ Web Development	✗ Databases	✗ Game Design
Required Technology	✓ Desktop or Laptop	✓ iPad/iPhone	✓ Android Device	✓ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

Micro:bit is a pocket-size programmable computer with 25 LEDs, buttons and pins for input and output, Bluetooth wireless communication, and the ability to sense temperature, light, acceleration, and magnetism. Students can develop programs on the Web, iOS, Android, Chromebooks, and laptop or desktop computers using graphical or text-based languages. Micro:bit may be powered via USB or by a proprietary battery pack. Developed by BBC Learning, micro:bit is distributed to all 11–12-year-old students in the United Kingdom.

Intro to CS is a semester-long, project-driven curriculum that explores micro:bit's capabilities. Each of the 12 lessons begin with an unplugged activity that investigates a big idea (for example, algorithms or iteration), followed by a coding activity that explores micro:bit's hardware capabilities and programming commands. Lessons conclude with maker-style projects where students identify and then solve problems they find interesting—for example:

- A micro:pet that responds when "petted" (touched or moved)
- A digital fidget cube that does different things when buttons are pressed; when the micro:bit is shaken, moved, or rotated; or when pins (0–2) are pressed

- Pedometers that count students' steps
- Calculators that perform arithmetic operations
- Musical instruments that play a series (array) of musical notes
- Games where one light dodges other lights
- An alarm system that notifies students when a door opens (a "sentry" micro:bit on a door communicates wirelessly with a "listener" micro:bit in another room)

Pictures of maker projects are included in the curriculum.

Each lesson requires approximately 3.5 hours, except the final lesson, which requires 13.5 hours.

The Microsoft Block Editor (https://makecode.microbit.org/) provides a Web-based environment for easy program development. As in Scratch, color-coded commands may be dragged and dropped to create programs. Commands vary in complexity, from simple blocks that activate lights or receive input to more complex functions, arrays, and objects. The Block Editor also provides an advanced block-to-text interface where dragged blocks convert into JavaScript text; later lessons in *Intro to CS* explore this mode.

To install their programs on micro:bit, students connect micro:bit via USB (no drivers necessary) and then drag programs onto it, or they pair micro:bit wirelessly via Bluetooth and Flash programs. (Micro:bit supports a wide variety of text and graphical languages, but *Intro to CS* uses only the Microsoft Block Editor.)

RESOURCES

Professional Development (PD): None. However, some Microsoft partners offer training on micro:bit (see, for example, https://www.fairchancelearning.com/professional-development).

Vendor or Provider Expectations: None

Ongoing Support: The searchable micro:bit Support page (https://support.microbit.org/support/home) includes frequently asked questions in a number of categories.

TECHNOLOGY

Required Hardware and Software and Their Costs: The minimum hardware per student (or student pair) includes one micro:bit, USB cable, and battery pack with two AAA batteries, available for $16.50 at Adafruit (https://www.adafruit.com/product/3362) or SparkFun (https://www.sparkfun.com/products/14336). In the United States, micro:bit is available for free for DonorsChoose.org projects (http://microbit.org/en/2017-08-18-donorschoose/).

Micro:bit hardware and software are compatible with laptops/desktops running Macintosh OSX, Windows, or Linux; Chromebooks; or handhelds running iOS or Android.

Environments and Programming Languages Used: Students may develop software in the Web-based Microsoft Block Editor using either its graphical or JavaScript (text) interfaces. Alternatively, they may develop software in MicroPython via a Web-based editor (http://python.microbit.org/v/1) or the downloadable Mu editor (https://codewith.mu/). They may also develop software in C++ using the mbed developer environment (https://developer.mbed.org/platforms/Microbit/). Macintosh users may also program micro:bit through ScratchX.

SUPPLEMENTAL MATERIALS

Additional useful materials include alligator clips (two kinds: standard, and with a male jumper end for servos), headphones or earbuds, a servo motor, and various craft supplies (e.g., scissors, boxes, markers, glue, cellophane tape, duct tape, copper tape, popsicle sticks, colored construction paper, pipe cleaners, stickers, feathers, and string). Additional, optional hardware components are suggested on the micro:bit website (http://microbit-accessories.co.uk/).

OTHER NOTES

Intro to CS is one of two curricula that have been developed for micro:bit. The second, *Computer Science for Innovators and Makers* by Project Lead the Way, is addressed elsewhere in this guide.

The following websites offer additional background and useful information on teaching with micro:bit:

- Technical information about micro:bit (https://tech.microbit.org)
- Microsoft project tutorials, examples, and documentation (https://makecode.microbit.org/docs)
- Getting Started with micro:bit (SparkFun—four videos) (https://www.youtube.com/watch?v=kaNtg1HGXbYandlist=PLBcrWxTa5CS0mWJrytvii8aG5KUqMXvSk)

MIDDLE SCHOOL PATHWAYS IN COMPUTER SCIENCE

» University of Massachusetts Lowell, TRITEC Inc., and Medford Public Schools

Compiled by Fred Martin
Professor, Computer Science, University of Massachusetts Lowell

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✗ 9–12	✗ Other			
PD Model and Availability	✓ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✗ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✗ Software Engineering	✗ Physical Computing	✗ Networking and Security	✓ Mobile Programming	✗ Web Development	✗ Databases	✗ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✓ Android Device	✓ Chromebook	✗ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

Middle School Pathways in Computer Science (CS Pathways) is a 20-hour middle school curriculum that builds students' digital literacy skills, CS knowledge, and awareness of CS-related careers. Students design mobile apps for social good, using MIT's App Inventor technology. The curriculum was developed with funding from NSF in collaboration with teachers in Everett and Medford, Massachusetts.

CS Pathways takes a "projects first" approach to teaching computing, inspiring kids to be makers of mobile software with a social impact. The curriculum immediately engages students in app-building, starting with the creation and manipulation of digital media, such as images and sound clips, and then continuing into CS concepts, such as events and conditionals. A career-awareness component encourages students to think broadly about CS as a tool they can use in many different careers.

In *CS Pathways*, students develop facility with Android tablets and learn how to author image, sound, and movie files. They learn foundational CS skills in a blocks-based programming environment, such as the selection and layout of app components and the use of event handlers and conditional logic. In every project, students are encouraged to engage in an iterative design process, including needs assessment, design sketches, debugging, testing with peers, and documentation. In terms of pedagogy, *CS Pathways* encourages experimentation, pair programming, and inclusive teaching practices.

The curriculum was primarily designed to be integrated into middle school technology/IT literacy courses. However, with its grounding in digital literacy, media, and communications, the curriculum also accomplishes digital literacy learning outcomes (e.g., students build apps instead of creating PowerPoint presentations).

CS Pathways has been integrated into technology, engineering, science, math, library, and art classes. In each case, teachers find overlap between their instructional goals and the *CS Pathways* curriculum—for example, computational skills are part of engineering, logical thinking supports mathematical reasoning, and digital creation skills are useful in making art.

RESOURCES

Professional Development (PD): None

Vendor or PD Provider Expectations: N/A

Ongoing Support: Materials related to the curriculum, including lesson plans, examples of students work, presentation slides, and research publications, are available on the project website (cspathways.org).

TECHNOLOGY

Required Hardware and Software and Their Costs: App Inventor requires the use of Android devices. For classroom use, every two students should be provided with an Android tablet (typically $50–150 each). Also recommended is a classroom cart (about $500), and heavy-duty rubber cases for the tablets (about $15 each).

Recommended Technology: App Inventor runs on Chromebooks, Macs, or PCs running the Chrome Web browser. Students must have Google accounts, as noted above.

Potential Technology Barriers: Support from a school's IT department may be required to configure computers and tablets. App Inventor requires peer-to-peer connectivity between the computer running Chrome and the tablet, which may require relaxing firewall rules. To support whole-classroom use (e.g., 20 browsers and 10 tablets in concurrent operation), the network needs to have good underlying performance.

Environments and Programming Languages Used: The curriculum is based on MIT App Inventor 2, a browser-hosted, blocks-based environment for creating mobile apps. App Inventor is free to use. It requires each student to have a Google account. In schools that use Google Apps for Education, the student email accounts provided by Google work as App Inventor accounts.

MIDDLE YEARS COMPUTER SCIENCE
» Harvey Mudd College

Compiled by Shaileen Crawford Pokress
K–12 Computer Science Education Consultant

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✓ No Official PD	✓ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✓ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✓ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

Middle Years Computer Science (MyCS) is an introductory CS course aimed at students in middle or early high school. MyCS emphasizes the creative aspects of CS, and offers a balance between theory and practice. The stated goal is that every student taking the course should come to believe that "CS is something that people like me do."

MyCS is adapted from Harvey Mudd College's introductory course for first-year college students. Originally designed for the classroom, it is also available online as a free six-week edX course. The six units in the curriculum are modular. Teachers can choose the pieces that fit their particular students to create a customized course. Teachers may also give students access to the online version of the course.

Students taking MyCS learn the foundational concepts and skills of CS. They explore how to use the power of computers to solve big, real-world problems. Students build knowledge through "unplugged" activities and then apply that understanding as they create projects in the Scratch programming environment.

Here is a sample course outline:

- **Unit 1: What is Computer Science?**
  - » Answer broad questions about the role of computers and the goals of computer scientists.

- Explore the definition of intelligence as it relates to computers.

Unit 2: A-maze-ing Scratch

- Learn the basics of Scratch programming through a series of pre-made mazes of increasing difficulty.

Unit 3: Data and Codes

- Encode and decode information using a variety of methods.
- Represent numbers in binary.
- Connect these concepts to data and information processing—the core of CS.

Unit 4: Projects in Scratch

- Create personally meaningful stories and games using Scratch.
- Apply design principles to create unique programming projects.

Unit 5: Problem Solving and Algorithms

- Build intuition for how people and computers solve problems differently.
- Learn basic algorithms for searching and sorting information, as well as how to compare algorithms.

RESOURCES

Professional Development (PD): In-person workshops to prepare teachers to teach MyCS are sometimes offered. These vary each year, so it is best to check the website in the winter or spring for upcoming PD opportunities. Teachers may also use MyCS edX MOOC, a free edX course, to prepare themselves to teach the course (https://www.edx.org/course/mycs-computer-science-beginners-harveymuddx-cs001x).

Vendor or Provider PD Expectations: None

Ongoing Support: The curriculum and accompanying teacher resources are available online (https://sites.google.com/a/g.hmc.edu/mycs/).

TECHNOLOGY

Required Hardware and Software and Their Costs: Computers with a stable Internet connection are needed.

Recommended Technology: All of MyCS can be done within a browser on any computer, including Chromebooks. There is no recommended operating system.

Potential Technology Barriers: School technology policy must be flexible enough to allow for student accounts to connect and log in to the Scratch programming environment. If the instructor chooses to have students use the edX course materials online, students must also be able to connect to the edX website.

Environments and Programming Languages Used: Scratch programming environment (free online at scratch.mit.edu); edX

SUPPLEMENTAL MATERIALS

There are variations of the MyCS course online, from which teachers may pull alternative materials—for example, the University of San Francisco's MyCS (https://sites.google.com/a/sfusd.edu/my-cs/).

STEM: EXPLORE, DISCOVER, APPLY

» National Integrated Cyber Education Research Center (NICERC)

Compiled by Dr. Chuck Gardner
Director of Curriculum, NICERC

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✗ 9–12	✓ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✓ Meetups	✓ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✓ Full Year	✓ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✓ iPad/iPhone	✓ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✗ Purchasable Software				

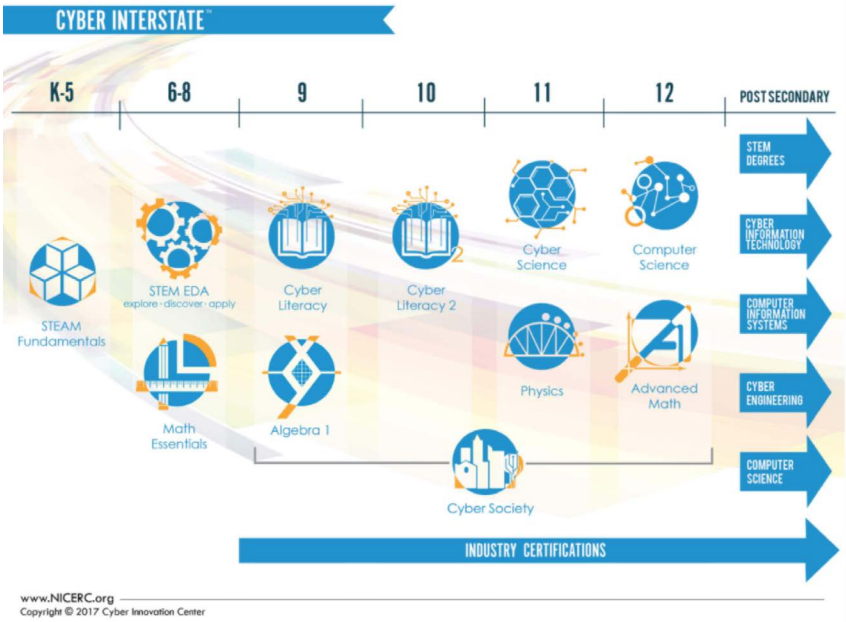
OVERVIEW

The National Integrated Cyber Education Research Center (NICERC) provides free classroom curriculum and PD opportunities for educators across the country. Currently, NICERC offers curricula for students in grades 6–12 on such topics as STEM (grades 6–8), cyber engineering (grades 9–11), physics and advanced mathematics (grades 11–12), and CS (grade 12). Teachers have reported that many of the courses can be taught outside of these grade bands.

STEM: Explore, Discover, Apply (STEM: EDA) engages middle school students through a series of hands-on collaborative projects that help improve their problem-solving and critical-thinking skills. All projects integrate the engineering design process, which allows students to creatively explore STEM through design. In grades 6 and 7, students *Explore* and *Discover* (respectively) fundamental STEM concepts, and then *Apply* them in grade 8. This Explore/Discover/Apply model applies across the 12 "threads" that comprise *STEM: EDA*:

Egg Drop	Alternative Energy	Roller Coasters
Volcanoes	Electricity	Cars
Music	Aerospace	Genetic
Catapults	Earthquakes	Coding

Each icon on NICERC's Cyber Interstate (see graphic) indicates a distinct course that includes 180 hours of content but is also modular. Teachers can implement *STEM: EDA* as a stand-alone course, choose individual



components and apply them where they may have relevance in their own classrooms, or offer *STEM: EDA* as an afterschool program.

The *STEM: EDA* modules are mapped to applicable NGSS and Common Core standards; frequent correlations to NICE's Cybersecurity Workforce Framework are also apparent.

RESOURCES

Professional Development (PD): Through a grant from the Department of Homeland Security (DHS), NICERC is able to provide PD at no cost to schools and districts, as long as there are at least 20 participants and the host organization arranges for a training site and any additional costs for participants. The PD models are customizable to the host organization; they can run from a few hours of content introduction to a four- or five-day model that takes a deeper dive into the content. Sessions can be held concurrently or end to end. The initial step of hosting a PD workshop is to complete the online PD form (https://nicerc.org/pd/).

Vendor or PD Provider Expectations: NICERC requires that host organizations (school, district, conference organization, state, etc.) arrange the local logistics, including securing a facility, arranging for any employee stipends or substitute teacher budgets, and providing meals or meal expenses for participants. NICERC grant funding covers NICERC personnel's travel, food, and lodging, and all workshop materials. Host organizations should discuss with NICERC the possibility of including take-home technology in the workshop, as some of those costs can be covered by NICERC.

Ongoing Support: Through the DHS funding, NICERC's content library is available at no cost to users. Access to the library includes a variety of materials needed to present the *STEM: EDA* (and other curricula) content to students, including student worksheets, homework pages, teacher master notes, PowerPoint presentations, tests, study guides, solution guides, rubrics, and national standards information. A network of more than 7,000 teachers are currently enrolled in NICERC's Canvas learning management system, where they can access the entire library, engage with other users, and engage directly with NICERC's team of subject-matter experts and master teachers. Additionally, NICERC connects to users through a variety of social media platforms, including Facebook, Twitter, and Instagram, and an ever-growing library of instructional and assistive videos on their YouTube channel. Lastly, NICERC staff and subject-matter experts are available all year via email and telephone to directly assist teachers with implementation concerns or questions or to help them customize content for their particular classroom, including providing assistance with local standards and administrative requirements.

TECHNOLOGY

Required Hardware and Software and Their Costs: Technology requirements vary with the content being implemented. Information and pricing for all NICERC curriculum kits can be found on the NICERC store website (https://nicerc.org/shop/).

Recommended Technology: Throughout NICERC content, students are constantly being challenged with research questions that are best handled through Internet-connected technology. Teachers should be prepared to use whatever connected devices are best for their audience, be it laptops, tablets, or cell phones. WiFi should be available for all participants. While the bandwidth needs are negligible for the formal curriculum, 1 Mbps is sufficient for networking activities.

Environments and Programming Languages Used: The *STEM: EDA* "Discover: Coding" module includes instruction on Scratch programming, a free software from MIT.

SUPPLEMENTAL MATERIALS

STEM:EDA makes use of "household" items to enhance instruction. A partnership with Nasco, a national educational supply vendor, provides teachers with pre-packaged kits ($30–170, depending on the module) containing everything they need to engage a class of students in the content. Each *STEM: EDA* module also includes a supply list that allows teachers to source the materials on their own, should they choose to do so.

Each lesson within NICERC's library of content includes a supply list indicating what materials may be required to complete the lesson, such as robotics platforms ($130–150 per pair of students), Raspberry Pi kits ($85 per student), and the aforementioned Nasco kits. For a significant portion of the content, only "typical" classroom supplies (e.g., markers, pencils, colored pencils, rulers, scissors, tape, and glue) are needed. Teachers should preview lessons before presenting them in the classroom to ensure that they have the appropriate materials on hand.

OTHER NOTES

NICERC is currently preparing an eighth grade math essentials curriculum that is scheduled for release in the spring or summer of 2018.

PLTW GATEWAY
» Project Lead The Way

Compiled by Karine Laidley
Director of Curriculum Development, PLTW

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✗ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✓ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✗ Networking and Security	✓ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✓ Android Device	✗ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

The *PLTW Gateway* curriculum is targeted for middle school students and comprises a number of engineering, biomedical, and CS units. The CS units include such concepts as algorithms (linear, conditional, and repetitive), abstraction, programming, and data, as well as tracing and debugging, pair programming, and computational thinking skills. Students who experience *PLTW Gateway* learn to become independent learners and engage in a curriculum that is hands-on and student-driven, as the teacher becomes a facilitator of learning. In addition to CS knowledge and skills, *PLTW Gateway* fosters transferable skills, including problem solving, persistence, creativity, collaboration, and communication.

Each unit is designed for a total of 45 days of instruction, 45 minutes per class. There are two CS units offered in this program:

- **Computer Science for Innovators and Makers:** Throughout the unit, which is targeted to sixth-graders, students learn about programming for the physical world by blending hardware design and software development, allowing students to discover CS concepts and skills by creating personally relevant, tangible, and shareable projects.
- **App Creators:** This unit, targeted to grades 7 and 8, exposes students to CS as a means of computationally analyzing and developing solutions to authentic problems through mobile app development. It also

conveys the positive impact of the application of CS to other disciplines and to society.

PLTW curricula use an activity-, project-, problem-based instructional approach, which offers a scaffold of learning experiences that students can then apply to solving an open-ended problem. Activities are designed for the acquisition of knowledge and skills and follow an appropriate learning progression for students. Projects provide opportunities for investigations and application of concepts and skills. Problems are designed to provide a common challenge that will typically result in unique solutions, which require the transfer of new and past knowledge and skills. Students learn about the design process as they collaborate to creatively design and develop solutions to engaging, real-world problems.

PLTW Gateway is part of the PLTW K–12 CS pathway and aligns to national standards, such as Common Core, NGSS, STL, and CSTA.

RESOURCES

Professional Development (PD): *PLTW Gateway* training options include the following:

- **Core Training:** This in-depth, collaborative experience is offered for each *PLTW Gateway* CS unit. It covers the unit content, pedagogy, and facilitation support, and is designed to prepare teachers to deliver a transformative learning experience in their classrooms. Five-day training for each unit costs $1,200/teacher and is offered at affiliate university sites across the United States.
- **Online Core Training:** PLTW offers online training for both CS units. This option offers a participant-centered and job-embedded experience. It is designed to develop teachers' understanding of course content and instructional practices, as well as to help teachers reflect on what's happening in their classroom as they teach the unit. Opportunities to participate and collaborate in a professional learning community will also provide support and feedback to help teachers implement the unit in their classrooms. The online course for each *PLTW Gateway* CS unit includes two-hour live online coach-led weekly meetings, for 10 weeks; participants also collaborate in small groups during the online weekly sessions. The cost is $1,200/teacher.

Vendor or PD Provider Expectations: None

Ongoing Support: Regardless of which training teachers receive, PLTW provides phone support to schools; the PLTW online community of teachers, master teachers, and PLTW staff for ongoing support; PLTW representation in all 50 states via PLTW's Director of School Engagement; and PLTW online resources (including videos, tutorials, answer keys, and standards alignment documentation) to support teachers with content knowledge, facilitation, and assessment.

TECHNOLOGY

Required Hardware and Software and Their Costs: The PLTW curriculum is accessible online and can also be downloaded locally on the student tablets ahead of time. All PLTW courses are also available as downloadable PDF files, if needed.

Hardware costs for *Computer Science for Innovators and Makers* is included in the materials cost. (Please see the Supplemental Materials section below.) For *App Creators*, schools are responsible for purchasing tablets for their classroom. One Android tablet for every two students, and one for the teacher, is recommended, as students will work in pairs. For both modules, schools are responsible for purchasing computers (desktops/laptops) for their classrooms. All software used in the *PLTW Gateway* CS units is free.

Potential Technology Barriers: *PLTW Gateway* is online and accessible on desktops/laptops, Chromebooks, and mobile devices. Any videos that are available within the courses need to be streamed and require Internet connectivity. Both *Gateway* CS units require Internet connectivity for accessing their online programming environments.

Environments and Programming Languages Used: *Computer Science for Innovators and Makers* uses the Microsoft Programming Experience Toolkit (micro:bit programming). App Creators uses MIT App Inventor.

SUPPLEMENTAL MATERIALS

The table below shows the cost of the materials (both durable and consumable items) needed for each CS unit, which are available at the PLTW Store (https://www.pltw.org/mypltwresources). These costs are based on a class of 20 students. Note that this does **not** include the cost of the tablets required for *App Creators* or the computers required for either unit.

PLTW Gateway CS Units (Grades 6–8)	Year 1	Year 2	Year 3
Computer Science for Innovators and Makers	$1,171	$60	$60
App Creators	$145	$60	$60

OTHER NOTES

The fee for schools to participate in *PLTW Gateway* is $750 per year. This covers participation in all Gateway units for that school. Schools may also apply for funding and grants opportunities through the PLTW Grant program.

Note: There will be a grant opportunity specifically for Massachusetts public schools through the PLTW Grant program. Details should be available on the PLTW the website in mid-October 2017.

Compiled by:
Irene Lee
Research Scientist, MIT's Scheller Teacher Education Program, and PI, Project GUTS

Melody Hagaman
Project Manager, Project GUTS

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✗ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✗ Meetups	✓ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✗ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

Project GUTS (Growing Up Thinking Scientifically)' middle school *Computer Science in Science* program, developed in partnership with Code.org, consists of four curricular units and PD designed to situate CS practices and concepts within the context of life, physical, and Earth sciences and to prepare students to pursue formal, year-long courses in CS during high school. Project GUTS uses CS as a tool to more deeply explore STEM concepts while addressing course standards. The modules leverage years of research funded by NSF. All curriculum resources—daily lesson plans, slide decks, videos, and supplemental extension resources for teachers—are provided at no cost to schools.

The *Computer Science in Science* curriculum connects CS to science through computer modeling and simulation. All lesson resources are aligned to the NGSS. Based on a crosswalk identifying areas of overlap between the NGSS and CSTA K–12 CS Standards, the modules address performance expectations in both standards.

There are four *Computer Science in Science* modules:

- Module 1: Introduction to Computer Modeling & Simulation
- Module 2: Water as a Shared Resource
- Module 3: Ecosystems as Complex Systems

- Module 4: Chemical Reactions

Modules 2–4 are designed to replace existing modules in Earth, life, and physical science, respectively, without the need for additional class time.

Each module consists of five lessons that augment traditional science instruction by including computational thinking in the process of modeling and simulation. The modular design allows for a range of classroom implementation time (from 10 to 25 hours).

RESOURCES

Professional Development (PD): Project GUTS' year-long PD program prepares middle school science teachers to implement the four modules and supports their growth as science teachers. Teachers are guided through each of the five-hour instructional sequences; they learn basic CS concepts and computer programming, follow the "Use-Modify-Create" progression while engaging in scientific inquiry, and discuss effective practices and solutions to potential roadblocks. Workshop costs and locations are negotiated on a case-by-case basis (with either schools or districts) and can accommodate 25–40 teachers. The program consists of three major components:

- Pre-workshop online preparation, consisting of introductory videos and guided tutorials (roughly one day)
- An intensive PD workshop (three days, face-to-face)
- Mini-workshops each semester (two one-day workshops, either face-to-face or virtual) and online support

Vendor or PD Provider Expectations: Project GUTS requires the sponsoring district to provide the location, materials, meals or meal vouchers, and stipends, and to recruit and register participants. Sponsoring districts contract with veteran Project GUTS facilitators and cover their transportation and stipends for offering the workshop. There are no prerequisites to attend the workshop other than interest in learning.

Ongoing Support: Project GUTS offers an online PD course (guts-2017.appspot.com) and hosts an online PD network (TeacherswithGUTS.org) for teachers to ask questions and share resources. Both are moderated by Project GUTS staff and veteran Project GUTS teachers.

TECHNOLOGY

Required Hardware and Software and Their Costs:

- **Hardware:** A PC or Mac with a modern browser and Adobe Flash enabled, or a Chromebook running Chrome OS (Acer or HP preferred). StarLogo Nova 2.0 uses HTML5/JavaScript, so it will also run on iPads.
- **Software:** *Computer Science in Science* uses the free online StarLogo Nova software for modeling and simulation. Note: The current version (1.0) uses Flash, but the newer version (2.0, available November 2017) will not require Flash.)

Recommended Technology: Internet connection with bandwidth capable of supporting downloads of 5 megabits per student using the software during a class period.

Environments and Programming Languages Used: *Computer Science in Science* uses the visual block-based language StarLogo Nova.

SUPPLEMENTAL MATERIALS

Project GUTS' *Computer Science in Science* curriculum binder, which is available online for free (TeacherswithGUTS.org) and for purchase in print format (https://marketplace.mimeo.com/projectguts), is highly recommended for all participants. Other materials needed include dice, plastic cups, colored pencils, big pads of sticky paper, and markers. A projector and screen are needed for the PD workshops.

OTHER NOTES

Project GUTS is unique in integrating CS in science through agent-based modeling and simulation at the middle school level. Project GUTS curriculum and PD were offered in districts across the United States through Code.org's CS in Science program (2014–2017). Project GUTS also offers numerous modules developed for afterschool programs that focus on issues of community relevance, such as opinion dynamics, climate change impacts on local agriculture, emergency egress, pollution, shared resources, social networks, sustainability, and traffic patterns.

ZULAMA GAME DESIGN FUNDAMENTALS

» Zulama, Inc.

Compiled by Nikki Navta
CEO, Zulama

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✓ Meetups	✓ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✗ Full Semester	✗ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✓ Mobile Programming	✓ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✓ Purchasable Software				

OVERVIEW

The Zulama Game Design Fundamentals package contains nine 15–20-hour courses that cover all aspects of game design, from storytelling to mobile app development to programming. At that critical time in middle school when many students lose interest in STEAM subjects, students engage in the content through fun, interest-driven video game projects while learning a wide range of technical skills, including drag-and-drop interfaces, block-based coding, and text-based programming. Course titles include "Games from American History," "Coding with GameMaker," "Storytelling in Games," "Math Game Design," "Making Mobile Games," "The Business of Games," "Arcade Game Design," "Gamestar Mechanic Game Design," and "Science Game Design."

The curriculum was developed with faculty from the Entertainment Technology Center at Carnegie Mellon University and is being offered in over 100 districts nationwide.

The courses are aligned with the national CSTA standards, state and national curriculum standards, NGSS, and 21st Century Skills. Crosswalks with learning standards, including the Massachusetts state standards, can be found on the Zulama website (http://zulama.com/our-program/standards/).

RESOURCES

Professional Development (PD): Zulama offers professional learning that is interactive, project-based, and participant-driven. A variety of opportunities are available, for example:

- Online self-paced teacher training, approximately four hours per course. Cost: $500 per course per teacher.
- Two-hour implementation meetings for teachers, administrators, and staff, online or onsite. Cost: $500 per session, for up to 50 participants.
- One-day in-person, hands-on, and project-based teacher training. Cost $2,200 per day, for up to 50 participants.

Vendor or PD Provider Expectations: Zulama requires that any interested schools include Zulama on the school's master schedule for all students, use the correct course code, publicize the course, and ensure adequate enrollment.

Ongoing Support: Teachers do not need prior programming experience to teach these courses. The Zulama Learning and Content Management System provides an active, robust online forum that connects teachers and gives them access to an extensive database of "how to" videos and other resources. Teacher notes are included throughout all lessons to provide even more explanation and background. A completed project file ("answer key") is provided for every lesson. Teachers maintain access to all online course-specific training while using the curriculum, so they can brush up on their skills at any time.

TECHNOLOGY

Required Hardware and Software and Their Costs: Unlimited student and teacher access to all nine courses costs $1,950 per building.

- **Hardware:** Internet-connected desktops or laptops are required, a minimum of one per classroom and ideally one device per student.
- **Software:** A modern browser, either Mac OS or Windows, is required. GameMaker Studio ($25 per device) is required for "Coding with GameMaker," GameSalad ($24 per device) is required for "Making Mobile Games," Scratch (free) is used in "Arcade Game Design," Gamestar Mechanic (free) is used in "Gamestar Mechanic Game Design," and one or more Hummingbird Duo base kits (approximately $100 each) are used for "Arcade Game Design."

Environments and Programming Languages Used: "Coding with GameMaker" uses GameMaker Studio and GML (a proprietary programming language similar to Python); "Making Mobile Games" uses GameSalad; "Gamestar Mechanic Game Design" uses Gamestar Mechanic; and "Arcade Game Design" uses Scratch.

OTHER NOTES

More information, including course syllabi, scope and sequence, and examples of student work, can be found on the Zulama website (http://zulama.com/our-program/short-courses/).

HIGH SCHOOL CURRICULA AND TOOLS

AP COMPUTER SCIENCE A

» The College Board

Compiled by Padmaja Bandaru
Computer Science Teacher, AMSA Charter School, and Co-President, CSTA-Greater Boston

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✓ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✓ Networking and Security	✓ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✗ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✗ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

The College Board is a nonprofit organization that helps students prepare for a successful transition to college through programs and services in college readiness, including the Advanced Placement (AP) Program.

AP Computer Science A introduces students to fundamental topics in CS, such as problem solving, design strategies and methodologies, organization of data (data structures), approaches to processing data (algorithms), analysis of potential solutions, and the ethical and social implications of computing. It is meant to be the equivalent of a first-semester course in CS. This engaging course underscores the importance of communicating solutions appropriately and in ways that are relevant to current societal needs.

The course uses Java as its programming language. The AP Computer Science A Exam uses a subset of Java. (See "AP Computer Science A Frequently Asked Questions" [https://apcentral.collegeboard.org/courses/ap-computer-science-a/course/frequently-asked-questions?course=ap-computer-science-a] for more information.)

Exam topics, requirements, and course descriptions are periodically updated. Visit the College Board AP website (https://apstudent.collegeboard.org/apcourse/ap-computer-science-a) or see "AP Computer Science A" (http://media.collegeboard.com/digitalServices/pdf/ap/ap-course-overviews/ap-computer-science-a-course-overview.pdf) for the most up-to-date information.

RESOURCES

Professional Development (PD): The College Board sponsors weeklong summer institutes taught by College Board-certified consultants. The cost varies depending on the provider. Upcoming CS workshops and conferences are listed on the College Board's AP Professional Development Event Calendar (http://eventreg.collegeboard.org/events/Calendar/Calendar.aspx?cal=54ae034d-96ef-4609-8458-c7c7a76ad3b9).

Vendor or PD Provider Expectations: None

Ongoing Support: The College Board hosts an active AP Computer Science A Teacher Community forum (https://apcommunity.collegeboard.org/web/apcompsci), where users can post questions that are answered in detail by experienced AP CS A teachers. There is also an online social media group (https://www.facebook.com/groups/APComputerScienceTeachers/), open to members only.

TECHNOLOGY

Required Hardware and Software and Their Costs: A CS lab with network access that can run a Java IDE (integrated development environment), such as Eclipse, Blue J, jGRASP, or Netbeans, is needed; browser-based IDEs may not support some labs with graphical user interfaces. Each student should have access to a computer for at least three hours a week during class.

Recommended Technology: Network-connected PCs or Macs with relatively recent technology.

Environments and Programming Languages Used: Any IDE that allows students to create, edit, compile, and execute Java programs.

SUPPLEMENTAL MATERIALS

Many free websites offer lab ideas and online practice, for example:

- Practice-It! (https://practiceit.cs.washington.edu/), developed by Stanford professor Marty Stepp, gives students practice with solving Java programming problems online.
- CodingBat (http://codingbat.com/java), developed by Stanford professor Nick Parlante, offers a number of problems to build students' coding skills in Java and Python.
- "Java Review for the AP CS A Exam," offered by Runestone Interactive (http://interactivepython.org/runestone/static/JavaReview/index.html), provides a complete guide to Java.

COMPUTATIONAL THINKING AND PROBLEM SOLVING

» University of Massachusetts Boston, Bunker Hill Community College

Compiled by Deborah Boisvert
Executive Director and PI, BATEC Center for Computing and Information Technologies

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✗ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✓ Networking and Security	✓ Mobile Programming	✓ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✗ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✗ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

Computational Thinking and Problem-Solving (CTPS) is designed to be a year-long class in computational thinking and creative problem solving, preparing students to advance to the AP Computer Science Principles course, Career and Technical Education IT coursework, and work-based learning opportunities. CTPS advances students' computational thinking skills and introduces an array of CS and IT competencies through a problem-based approach for students to apply learning in more relevant ways through authentic real-world scenarios.

The course is approachable and appropriate for all students regardless of their previous knowledge, skills, and abilities, although students may benefit from having foundational CS coursework, such as *Exploring Computer Science* or Code.org's *Computer Science Discoveries*. There is a strong focus on "soft skills" (problem solving, critical thinking, collaboration, resilience, and communication) and on solving complex problems. In addition, the course emphasizes the teamwork, reflection and metacognition, writing and presentation skills, and cohort-building skills that are important to student retention and advancement.

CTPS is organized as a set of six projects that seek to engage students and give them a sense of ownership and a maker mentality, using topical areas such as designing a gaming lounge, reducing the technology carbon footprint, developing a mobile application for a local student or community-based organization, creating a Web presence for a local nonprofit, and exploring security issues. The

problems themselves reflect different IT domains, feature realistic situations, and involve increasingly complex steps. Problems offer either well- or ill-defined goals and outcomes, and include either a limited scope with concrete tasks or an infinite scope requiring self-determined constraining as students discover their resource limitations.

As they work through challenges in the course, students identify what they already know, what they need to know, and how and where to access new information that may lead to resolution of the problem. Students learn how to (1) analyze, critique, and select from alternative sources of information and courses of action, (2) think about problems that have multiple solutions, (3) work together with those of differing views, and (4) confront and address problems and situations in constructive and creative ways.

CTPS was developed by BATEC (headquartered at the University of Massachusetts [UMass], Boston) through an NSF grant to better understand computational thinking in the information technologies. It is presently offered at UMass and Bunker Hill Community College as a freshman-level course, affording an opportunity to explore dual enrollment options. It is also currently offered as dual enrollment course within the Boston Public Schools and Chicago Public Schools.

The curriculum, along with all necessary teacher resources such as lesson plans, student supports and rubrics, is available at http://batec.org/curriculum/highschool/ct-ps/.

RESOURCES

Professional Development (PD): CTPS teachers participate in an intensive one-week, pedagogy-focused summer face-to-face institute. This is followed by technical assistance meetups throughout the year to help teachers facilitate the CTPS challenges. The primary PD fixed costs are the facilitators (two for groups greater than 20; a minimum of 12 teachers is required) and the curriculum binder; variable costs include teacher stipends and meals. Issues of teacher stipends are dependent on districts and any PD sponsors.

Vendor or PD Provider Expectations: In Massachusetts, BATEC is the primary PD provider. BATEC requests that any interested schools include CTPS on the school's master schedule for all students, publicize the course, and ensure adequate enrollment. Districts should work with BATEC if they are interested in exploring dual enrollment options. Facilitators often need to be scheduled well in advance.

Ongoing Support: BATEC will work with districts to provide technical support and assistance for teachers who have adopted the course in their programs of study.

TECHNOLOGY

Required Hardware and Software and Their Costs: Desktops or laptops with Microsoft Excel, Word, and PowerPoint are required. Each student should have access to earbuds or headsets. Module 2 requires Kill-A-Watt meters (one per team; available for less than $20 each) and *The Human Face of Big Data* video (one for the class; available for $10). Module 4 benefits from Android tablets or phones. All required materials are online, either within the CTPS curriculum or linked from the curriculum online.

Recommended Technology: Google Drive (or other online storage space) is recommended for student products.

Potential Technology Barriers: Tablet use depends on reliable WiFi access.

Environments and Programming Languages Used: Excel, App Inventor

SUPPLEMENTAL MATERIALS

Non-technology materials needed include three-ring binders for each student (for exit tickets). Module 0 requires *Think Like a Freak* books for each student, and Marshmallow Challenge Kits (spaghetti, string, marshmallows, masking tape, and a paper lunch bag), one kit per team. Module 4 requires Water Tower Challenge kits: playing cards (1 pack/team), scissors (1/team), Scotch tape (1 roll/team), Gummi Bears (2/team), and marshmallows (1/team).

OTHER NOTES

Additional scenarios are being developed with the assistance of BATEC's business partners and will be added to the curriculum site in the future.

THE COLLEGE BOARD[5]

Course Details

AP Computer Science Principles introduces you to the foundations of computer science with a focus on how computing powers the world. Along with the fundamentals of computing, you will learn to analyze data, create technology that has a practical impact, and gain a broader understanding of how computer science impacts people and society.

The AP CSP course is organized around seven **big ideas**, which are essential to studying computer science.

Big Idea 1: Creativity

Computing is a creative activity. In this course, you will use the tools and techniques of computer science to create interesting and relevant digital artifacts (e.g., a video, animation, infographic, audio recording or program) with characteristics that are enhanced by computation.

Big Idea 2: Abstraction

Abstraction is a central problem-solving technique in computer science. In this course, you'll use abstraction to model the world and communicate with people and machines.

Big Idea 3: Data and Information

Data and information facilitate the creation of knowledge. Managing and interpreting an overwhelming amount of raw data is part of the foundation of our information society and technology. In this course, you will work with data to better understand the many ways in which data is transformed into information and knowledge.

Big Idea 4: Algorithms

Algorithms are used to develop and express solutions to computational problems. They are fundamental to even the most basic everyday task. In this course, you will work with algorithms in many ways: You will develop and express original algorithms, implement algorithms in a language, and analyze algorithms analytically and empirically.

Big Idea 5: Programming

Programming enables problem solving, human expression, and creation of knowledge. It results in the creation of software, and it facilitates the creation of computational artifacts, including music, images, and visualizations.

In this course, you'll learn the fundamental concepts of programming that can be applied across a variety of projects and languages. You will create programs, translating human intention into computational artifacts.

Big Idea 6: The Internet

The Internet and systems built on it have a profound impact on society. It pervades modern computing. In this course, you will: gain insight into how the Internet operates; study characteristics of the Internet and systems built on it; and analyze important concerns, such as cybersecurity.

Big Idea 7: Global Impact

Computation has changed the way people think, work, live, and play. In this course, you'll become familiar with many ways in which computing enables innovation. You will analyze the potential benefits and harmful effects of computing in a number of contexts.

ABOUT THE EXAM

The AP Computer Science Principles Assessment consists of two components: a through-course assessment and the end-of-course AP Exam. Both of these parts will measure your achievement of the course learning objectives.

Through-Course Assessment

For the through-course assessment, you will upload digital artifacts and written responses via a Web-based digital application. You will be asked to describe or analyze your work, whether it includes research, the creation of an artifact (e.g., a video, spreadsheet, graph, or electronic slide show), or the creation of a program.

End-of-Course AP Exam

The end-of-course AP Exam is a paper-and-pencil written exam. It is 2 hours long and includes approximately 74 multiple-choice questions. There are two types of multiple-choice questions:

- Single Select Multiple-Choice: you select 1 answer from among 4 options
- Multiple Select Multiple-Choice: you select 2 answers from among 4 options

5 Note from editors: This description of CSP is from the College Board's website (https://apstudent.collegeboard.org/apcourse/ap-computer-science-principles). © 2017 The College Board. *Mobile CSP, Beauty and Joy of Computing,* Code.org's AP Computer Science Principles, and PLTW's Computer Science Principles have all been formally endorsed by the College Board as AP CSP providers; the description here applies to all four courses, as well as to the Zulama Computer Science Program of Study.

BEAUTY AND JOY OF COMPUTING
» EDC and University of California, Berkeley

Compiled by Mary Fries
Curriculum and Instructional Design Associate, EDC

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✓ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✓ Meetups	✗ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✗ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✓ Networking and Security	✓ Mobile Programming	✗ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✗ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

The *Beauty and Joy of Computing* (BJC) is designed to attract nontraditional computing students (especially women and minorities) to the breadth and depth of ideas in modern CS. BJC uses Snap!, a visual programming language based on Scratch that was designed especially for this course. Unlike most CS Principles courses, BJC is programming-heavy, with the aim of giving students a taste of the beauty and joy of programming itself, not just the products of programming.

BJC takes a project-based learning approach (contextualized in games, art, mathematics, language, etc.), rather than a topical approach. The goal is to help students develop computational habits of mind, including abstraction and modularity, an understanding of the power of algorithms and the logic of predicates, the value of modeling, and the use of conditionals and other control structures. The course takes the social issues of computing very seriously, with labs on privacy, copyright, artificial intelligence, networking, cybersecurity, number systems, and logic gates. And, in developmentally appropriate ways, it introduces powerful CS ideas beyond the AP requirements not encountered in most high school courses, including higher order functions, advanced data structures, and recursion.

The BJC curriculum is endorsed by the College Board as preparation for the AP Computer Science Principles Exam. Self-check quizzes and project-based end-of-unit assessments are built into the curriculum, which helps to

prepare students for the AP Performance Tasks and Exam. The curriculum includes a corresponding Teacher Guide and example solutions.

RESOURCES

Professional Development (PD): BJC offers a three-week summer PD workshop n locations across the United States for teachers intending to teach BJC the following year. The cost of the summer PD is $2,000, but U.S. public school teachers may raise the tuition and an optional teacher stipend of $1,000 through DonorsChoose.org (and thus attend the workshop for free). The workshop is open to all teachers who successfully complete the application process, and to others who support these teachers.

In 2017, PD trainings were offered in California, Colorado, New Jersey, New York, Pennsylvania, and South Carolina. BJC also offers an extended PD model for New York City in partnership with the state Department of Education.

Vendor or PD Provider Expectations: During the summer PD, teachers are expected to:

- Complete the three-week PD course, approximately 60 hours (15–20 hours pre-PD preparation, 40 hours face-to-face during Week 2, ~5 hours post-PD)
- Work through a selection of the course's eight content units (the same units the students will work on)
- Maintain an online portfolio of their work on the lessons

During the school year, teachers are expected to:

- Teach the BJC course, ideally throughout the entire academic year, but half-year courses (in cases of block scheduling) are also acceptable
- Participate in periodic online gatherings with project leaders and mentor teachers
- Complete a teacher survey at the end of the semester and/or academic year
- Work with the project research team to gather data from students (student and parental consent forms, a pre- and post-course questionnaire, completed rubrics for the student performance tasks, final exam, etc.)

The teacher's school is expected to:

- Schedule at least one section of the BJC course during the academic year
- Recruit students with an aim toward enrolling representative numbers of the targeted demographics (females, underrepresented minority students, and students with disabilities).

Ongoing Support: There is an active online community of new and master BJC teachers, PD providers, and curriculum developers. A version of the course is also available on edX as a MOOC for independent learners and as a SPOC (small private online course) for classroom use.

TECHNOLOGY

Required Hardware and Software and Their Costs: One computer with Internet access and a relatively up-to-date browser is required for every two students. The minimum specifications for running Snap*!* are outlined on the BJC website (http://bjc.edc.org/bjc-r/cur/specifications.html).

Potential Technology Barriers: The curriculum uses videos, so teachers should ensure that the videos work on their school computers.

Environments and Programming Languages Used: BJC uses the Snap*!* programming language, which was developed specifically for this curriculum. Its visual, drag-and-drop design is based on Scratch, so it's accessible to a wide audience and not intimidating; however, the language itself is extended with the abstraction mechanisms needed for serious CS: first-class procedures for control abstraction and first-class lists for data abstraction. These capabilities are embodied in carefully chosen visual metaphors, so that ideas traditionally considered difficult can be understood and enjoyed by beginners.

SUPPLEMENTAL MATERIALS

BJC uses the book *Blown to Bits* (Abelson, Ledeen, and Lewis, 2008) as a reference text. Students do not need their own copy of this book, as the text is available online (bitsbook.com).

OTHER NOTES

"Comparison with CSP Framework" on the BJC website (http://bjc.edc.org/bjc-r/cur/compare.html) provides more information on what makes BJC special.

CODE.ORG COMPUTER SCIENCE PRINCIPLES

» Code.org

Compiled by Kaitie O'Bryan
Math and Computer Science Teacher, Mounds View Public Schools, Minnesota; Code.org Facilitator and Forum Moderator

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✓ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✗ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✓ Networking and Security	✓ Mobile Programming	✓ Web Development	✓ Databases	✗ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✗ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✓ Browser Only	✗ Free Software	✗ Purchasable Software				

OVERVIEW

Code.org's AP *Computer Science Principles* curriculum was designed to support students and teachers who are new to the discipline. The curriculum includes daily lesson plans comprising inquiry-based activities, videos for both students and teachers, assessments, and computing tools, allowing teachers to guide and learn alongside students as they discover core computing concepts. The curriculum is completely free for anyone to use.

Curriculum features:

- Daily instructional lesson plans that include inquiry- and equity-based pedagogy and background content
- Formative and summative assessments, project exemplars, and rubrics
- Widgets and simulators for exploring computing concepts, such as text compression and the Internet
- Concept and tutorial videos for students, and teaching tips-and-tricks videos for teachers
- Code Studio—a learning platform where students interact with lesson materials and tools, and where teachers access a dashboard to see student work and progress
- App Lab—a JavaScript programming environment in Code Studio, designed for creating event-driven Web apps with block-to-text workspace and debugging capabilities

Many of the projects, assignments, and activities in this curriculum encourage students to be creative, to express themselves, and to share their creations with others. While certain lessons focus on learning and practicing new skills, the goal is always to enable students to transfer these skills to creations of their own.

The course includes five units:

- **Unit 1: The Internet:** Students learn how the multi-layered systems of the Internet function. They collaboratively solve problems and puzzles about encoding and transmitting data, both "unplugged" and using Code.org's Internet Simulator.
- **Unit 2: Digital Information:** Students use a variety of digital tools to look at, generate, clean, and manipulate data to explore the relationship between information and data.
- **Unit 3: Algorithms and Programming:** Students learn JavaScript with Turtle programming in Code.org's App Lab, and learn general principles of algorithms and program design.
- **Unit 4: Big Data and Privacy:** Students research current events around the complex questions related to public policy, law, ethics, and societal impact.
- **Unit 5: Building Apps:** Students continue to learn how to program in the JavaScript language by creating a series of Web applications that live online.

This course is fully aligned to the College Board's AP *Computer Science Principles Course and Exam Description* (https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf). Additionally, in the curriculum materials, each lesson names the CSTA standards aligned to the lesson for those teachers and schools who may decide not to teach the course as an AP course.

While this course may be part of a larger CS sequence, there are no CS prerequisites.

RESOURCES

Professional Development (PD): A free week-long PD training is available for teachers in Massachusetts, and four quarterly workshops will be held on Saturdays throughout the school year. Contact Code.org (teacher@code.org) for information about dates. Code.org's PD is offered to teachers at no cost.

Vendor or PD Provider Expectations: Participants who plan to attend PD are expected to attend every day.

Ongoing Support: Each lesson comes with a carefully written overview, agenda, and teaching guide aligned to the objectives. Code.org also offers a moderated online forum available to teachers.

TECHNOLOGY

Required Hardware and Software and Their Costs:

- **Hardware:** A Chromebook, desktop, or laptop computer is required for each student in the course.
- **Software:** Access to YouTube and Flash player is needed for Code.org-hosted videos. (If needed, the YouTube videos can be downloaded ahead of time and used offline.)

Recommended Technology: An Internet connection of at least 15 MB/sec is recommended.

Environments and Programming Languages Used: Students use Code.org's App Lab to write programs. App Lab is a JavaScript programming environment, designed for creating apps with block-to-text workspace.

SUPPLEMENTAL MATERIALS

Many lessons have recommended handouts that are designed to guide students through activities; these are available as PDFs or in Google Docs. One unit requires some craft materials.

OTHER NOTES

Additional information can be found in the *Code.org CS Principles Curriculum Guide* (https://code.org/files/CSP_CurriculumGuide_2017_forWeb.pdf).

MOBILE COMPUTER SCIENCE PRINCIPLES

» The Mobile CSP Project

Compiled by Ralph Morelli
Professor of Computer Science Emeritus, Trinity College, and Co-PI, Mobile CSP Project

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✓ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✓ Meetups	✓ Local Workshops	✗ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✗ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✓ Networking and Security	✓ Mobile Programming	✗ Web Development	✓ Databases	✗ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✓ Android Device	✓ Chromebook	✓ Works with Any Major OS	✗ Custom Hardware Device	
Required Software	✗ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

Mobile Computer Science Principles (Mobile CSP) (https://course.mobilecsp.org) is one of the College Board-endorsed curricula for AP Computer Science Principles. As its name suggests, it is organized around a mobile computing theme, in which students learn the principles of CS through the process of designing and building socially useful mobile apps in App Inventor. In this way, *Mobile CSP* meets students where *they* hang out (i.e., on their mobile devices) and draws them into the process of creating apps that they care about.

The *Mobile CSP* curriculum is project-based and emphasizes collaborative learning and creativity. Students are engaged in CS topics through their desire to create mobile apps that benefit their school, family, neighborhood, or broader community. Students maintain a portfolio of all coursework.

The course is hosted entirely online. It is designed to be delivered by a classroom teacher through multiple modalities (slide presentations, videos, readings). The course consists of 60 lessons, approximately half of which are app-building lessons. Programming is taught through a three-step, pair-programming model: Students follow a hands-on tutorial that introduces new App Inventor constructs and programming concepts. Students then receive a set of problem-solving tasks and programming challenges (solutions are provided). Finally, students are encouraged to come up with their own ideas for apps or enhancements to an existing app.

In addition to the College Board's required Create Task, students must work in pairs on a project of their own design and implementation. Many of the CSP (non-programming) lessons are also project-based and use a process-oriented group-inquiry learning (POGIL) approach that encourages students to collaborate on solutions to computing problems.

Accompanying the course is a complete, parallel resource site for teachers (https://course.mobilecsp.org/teach_mobilecsp) that includes all lesson plans, rubrics, presentations, quizzes, exams, and pedagogy resources. Each lesson is mapped to the specific learning objectives, essential knowledge, and enduring understandings of the CS Principles framework.

There are eight units:

- **Unit 1:** Getting Started: Preview and Setup
- **Unit 2:** Introduction to Mobile Apps and Pair Programming
- **Unit 3:** Creating Graphics and Images Bit by Bit
- **Unit 4:** Animation, Simulation, and Modeling
- **Unit 5:** Algorithms and Procedural Abstraction
- **Unit 6:** Using and Analyzing Data and Information
- **Unit 7:** Communication Through the Internet
- **Unit 8:** AP CS Principles Exam Prep

A detailed curriculum overview is available online in Google Docs (https://docs.google.com/document/d/1iFHenUMq8-VLF5EQ8mG9IwaFfm2elg_ox5_IzzjTW14).

Mobile CSP is College Board-endorsed and covers the AP CSP framework. It also satisfies the Common Core Reading Standards for Informational Texts.

This is a stand-alone course that can be taken as a first course in CS. It would also work well in a two-course sequence (*Mobile CSP* and AP Computer Science A) or a three-course sequence (*Exploring Computer Science, Mobile CSP,* and AP Computer Science A).

RESOURCES

Professional Development (PD): The Mobile CSP project offers various models of PD, including an online option. In 2017 the subscription fee was $2,000, which included 120 hours of PD training and one-on-one mentoring during the school year by a Master Teacher. For details on the 2018–19 PD program, visit the Mobile CSP wesbite (mobilecsp.org).

Vendor or PD Provider Expectations: None

Ongoing Support: *Mobile CSP* teachers are supported by a large, active online community. Support during the school year also includes monthly webinars that focus on CS pedagogy and content.

TECHNOLOGY

Required Hardware and Software and Their Costs: The curriculum, including App Inventor, and the teacher resource site are available through a browser. All course and teacher materials, including App Inventor, are free and openly licensed (Creative Commons). While the course could be taught using a free Android emulator, it works best with Android mobile devices. Suitable Android tablets (not provided) cost between $50 and $100. One Android device for every two students is recommended.

Potential Technology Barriers: The course requires reliable WiFi access that can support peer-to-peer connections between the computer and the Android device. Students also require access to various Google products (Gmail, Sites, and YouTube or TeacherTube).

EXPLORING COMPUTER SCIENCE

» UCLA Graduate School of Education and Information Studies

Compiled by Farzeen Harunani
Coordinator of District and Teacher Engagement, MassCAN, EDC

PROPERTIES

Intended Grade Band	✗ K–5	✓ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✓ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✗ Networking and Security	✗ Mobile Programming	✓ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✗ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✗ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

Exploring Computer Science (ECS) is a widely used introductory high school CS curriculum developed with funding support from NSF. The course was designed to expose early high school students to a wide breadth of what CS has to offer. It features inquiry-based instruction and emphasizes both equity and the societal impacts of computing in all aspects of CS. By concentrating on the three pillars of equity, inquiry, and content, ECS has made important inroads at bringing equity into CS. ECS encourages the think-pair-share model, small-group work, journaling, and reflection.

The course comprises six units that are intended to be taught in order:

- **Unit 1: Human-Computer Interaction** focuses on how people interact with computers, models of intelligent behavior, and the Internet.
- **Unit 2: Problem Solving** teaches algorithms, abstraction, and connections between mathematics and CS.
- **Unit 3: Web Design** teaches Web design and development, along with more about the Internet.
- **Unit 4: Introduction to Programming** reinforces the ideas of algorithms and abstraction and teaches programming.
- **Unit 5: Computing and Data Analysis** also teaches programming, but focuses more on data and information.

- **Unit 6: Robotics** allows students to reinforce their programming knowledge by actually seeing their work in physical action.

While the first four units are expected to be taught in all ECS courses, the last two are optional. Usually, all six units are taught as a one-year course. ECS may be offered as a one-semester course, covering the first four units.

Due to its introductory nature, ECS has no prerequisites and could be easily followed with a programming course, an AP Computer Science Principles course, a Web design course, or a robotics/hardware course.

The curriculum, along with all necessary teacher resources, is available on the ECS website (www.exploringcs.org).

RESOURCES

Professional Development (PD): Beginning ECS teachers participate in an intensive one-week, pedagogy-focused summer face-to-face institute. This is followed by four content-focused quarterly workshops offered face-to-face on four Saturdays throughout the year to help teachers prepare to teach upcoming ECS units. ECS teachers who are unable to attend the face-to-face workshops can access this PD support through an online option consisting of four two-week online experiences.

The primary PD fixed costs are the facilitators (two for groups greater than 15) and the curriculum binder; variable costs include teacher stipends and food. In Massachusetts, ECS has an official partner (the Massachusetts Exploring Computer Science Partnership) that administers the PD, and EDC is the primary PD coordinator.

Vendor or PD Provider Expectations: There is a minimum requirement of 15 teachers in order to access ECS PD. Facilitators often need to be scheduled well in advance. ECS requires that any interested schools include ECS on the school's master schedule for all students, use the correct course code (10012), publicize the course, and ensure adequate enrollment. Issues of teacher stipends are dependent on the district and any PD sponsors.

Ongoing Support: Because of its nationwide popularity, ECS boasts the largest number of robust online communities. Look for a community of interest on the CS for All Teachers website (https://csforallteachers.org/groups?qt-all_groups=1 - qt-all_groups); here you can also find assessment materials for Units 1–4 (https://csforallteachers.org/exploring-computer-science) and other key resources. Within Massachusetts, there is a community of over 100 ECS teachers, many of whom are part of either the Greater Boston or the Western Massachusetts CSTA chapters.

TECHNOLOGY

Required Hardware and Software and Their Costs: All required materials are online, either within the ECS curriculum or linked from the curriculum.

- **Hardware:** Internet-connected desktops or laptops are required. For Unit 6, the Edison robot is used, which can be purchased online (www.meetedison.com) for $49 apiece, or in quantities up to 30 for $990 ($33 apiece).
- **Software:** A modern browser (preferably Firefox or Chrome), a basic HTML/CSS editor, Scratch (which can be run in a browser), a spreadsheet and document viewer/editor (which can be accessed in Google Drive in a browser), and EdWare (which can be run in a browser) are needed.

Recommended Technology: All of ECS can be done within a browser, so Chromebooks can be used, and there is no "recommended" operating system. A stable Internet connection is recommended.

Potential Technology Barriers: In Unit 6, the Edison robots connect to the computers via the computer's audio jack.

Environments and Programming Languages Used: Unit 3 uses a blogging website, HTML/CSS, Flash (which is optional), and JavaScript (also optional). Unit 4 uses Scratch. Unit 6 uses EdWare.

SUPPLEMENTAL MATERIALS

It is recommended that each student have a journal. Legos are recommended for Unit 6.

» National Integrated Cyber Education Research Center (NICERC)

Compiled by Dr. Chuck Gardner
Director of Curriculum, NICERC

PROPERTIES

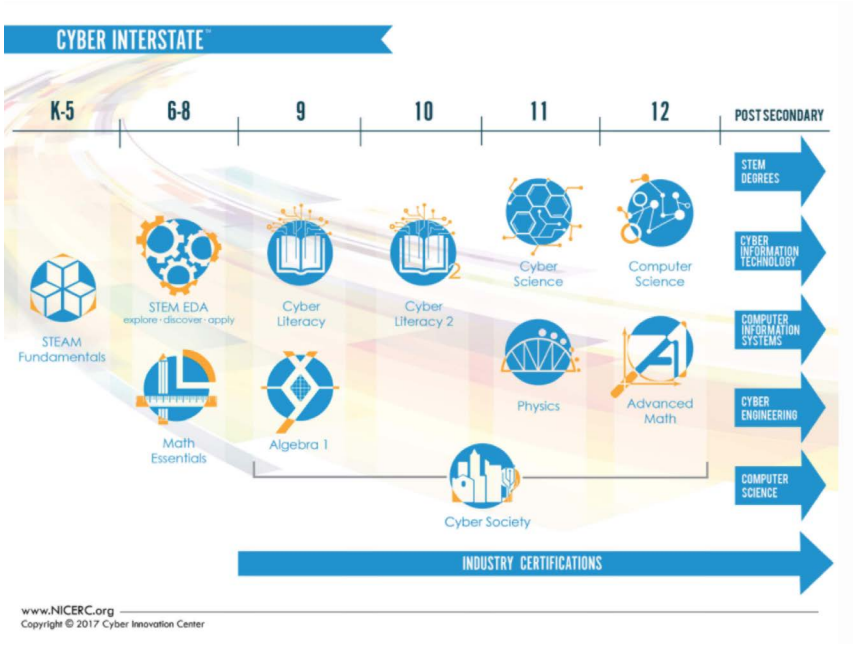
Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✗ Online PD (MOOC, etc.)	✓ Meetups	✓ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✓ Less than One Semester	✓ Full Semester	✓ Full Year	✓ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✓ Networking and Security	✗ Mobile Programming	✗ Web Development	✗ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

The National Integrated Cyber Education Research Center (NICERC) provides free classroom curriculum and PD opportunities for educators across the country. Currently, NICERC offers curricula for students in grades 6–12 on such topics as STEM (grades 6–8), cyber engineering (grades 9–11), physics and advanced mathematics (grades 11–12), and CS (grade 12). Teachers have reported that many of the courses can be taught outside of these grade bands.

- **Cyber Literacy 1** (grade 9) introduces students to cyber engineering by blending robotics, programming, electricity, and elements of liberal arts. Students learn about the opportunities, threats, responsibilities, and legal constraints associated with operating in cyberspace.
- **Cyber Literacy 2** (grade 10) takes students into a deeper exploration of cyber by blending real-world robotics with significant investigations into the liberal arts and humanities. The robotics component is approached as a series of increasing systems-engineering challenges. An investigation of the Fourth Amendment helps students understand the relevance of privacy vs. security in today's digital society as well as some key components of proper search and seizure.
- **Cyber Science** (grade 11) uses a robotics platform to teach important cyber concepts and fundamentals. Students are engaged in a systems-level approach to problem-solving using robotics and CS in the context of the humanities; they make meaningful connections between STEM and liberals arts while learning how to become better cyber-citizens.

- **Computer Science** (grade 12) This course provides a broad understanding of CS concepts by exposing students to computer programming through relevant projects that use unique hardware and software platforms.
- **Cyber Society** (grades 9–12) enables teachers to foster cyber awareness among high school students who will help develop the pipeline of a future cyber workforce. The lessons within each module are easily customizable by teachers; topics include Cyber Law, Cyber Ethics, Cyber Terrorism, Cyber Communications, and Cyber Business.



Each icon on NICERC's Cyber Interstate (see graphic) indicates a distinct course that includes 180 hours of content but is also modular. Teachers can implement each unit as a stand-alone course, choose individual components from each course and apply them where they may have relevance in their own classroom, or offer the units as afterschool programs.

RESOURCES

Professional Development (PD): Through a grant from the Department of Homeland Security (DHS), NICERC is able to provide PD at no cost to schools and districts, as long as there are at least 20 participants and the host organization arranges for a training site and any additional participant costs. The PD models are customizable to the host organization; they can run from a few hours of content introduction to a four- or five-day model that takes various deep dives into the content. Sessions can be held concurrently or end to end. The initial step in hosting a PD workshop is to complete the online PD form (https://nicerc.org/pd/).

Vendor or PD Provider Expectations: NICERC requires that host organizations (school, district, conference organization, state, etc.) arrange the local logistics, including securing a facility, arranging for any employee stipends or substitute teacher budgets, and providing meals or meal expenses for participants. NICERC grant funding covers NICERC personnel's travel, food, and lodging, and all workshop materials. Host organizations should discuss with NICERC the possibility of including take-home technology in the workshop, as some of those costs can be covered by NICERC.

Ongoing Support: Through the DHS funding, NICERC's content library is available at no cost to users. Access to the library includes a variety of materials needed to present the content to students, including student worksheets, homework pages, teacher master notes, PowerPoint presentations, tests, study guides, solution guides, rubrics, and national standards information. A network of more than 7,000 teachers are currently enrolled in NICERC's Canvas learning management system, where they can access the entire library, engage with other users, and engage directly with NICERC's team of subject-matter experts and master teachers. Additionally, NICERC connects to users through a variety of social media platforms, including Facebook, Twitter, and Instagram, and an ever-growing library of instructional and assistive videos on their YouTube channel. Lastly, NICERC staff and subject-matter experts are available all year via email and telephone to directly assist teachers with implementation concerns or questions or to help them customize content for their particular classroom, including providing assistance with local standards and administrative requirements.

TECHNOLOGY

Required Hardware and Software and Their Costs: Information and pricing for all NICERC curriculum kits can be found on the NICERC store website (https://nicerc.org/shop/). WiFi should be available for all participants. While bandwidth needs are negligible for the formal curriculum, 1 Mbps is sufficient for networking activities.

The technology requirements vary according to the content being implemented:

- *Cyber Literacy 1:* The assumption is that students will use a programmable robotics platform to get the most from the content. Essentially, the robot becomes the textbook, in that students interact with the robotic platform almost daily to learn the course content. Through a partnership with Parallax, an international educational and technology supply vendor, teachers receive discounted pricing on NICERC-compliant products. The Parallax Boe-Bot or the Parallax Shield-Bot with Arduino are two platforms that can be used in *Cyber Literacy 1*, and each are less than $150. NICERC suggests that students use the robotic platforms in pairs, so 10 robotic kits would be needed for a class of 20. Additionally, students need a laptop to program the bots with free, downloadable software, available from a variety of sources.
- *Cyber Literacy 2, Cyber Science,* and *Cyber Physics* also use the Parallax Boe-Bot platform.
- *Computer Science:* This course uses the Raspberry Pi platform as its textbook. NICERC suggests that students use the RasPi platforms in a one-to-one configuration, meaning that a class of 20 would require 20 RasPi kits. Through a partnership with CanaKit, an international technology supply vendor, teachers receive discounted pricing on NICERC-compliant RasPi kits. There are two options available from CanaKit:
  - The "Basic" kit contains just the RasPi and accessories to create circuits; it costs $85. This kit is adequate for schools with an excess of VGA or HDMI monitors and keyboard/mice combos.
  - The "Ultimate" kit includes the Basic kit, a 7" LCD touchscreen, and a keyboard/trackpad combo; it costs $197. For schools that don't have extra tech available, this kit is a better option.

Recommended Technology: Throughout NICERC content, students are constantly challenged with research questions that are best handled through Internet-connected technology. When programming the Boe-Bot or Shield-Bot with Arduino platforms, the product vendors suggest that classrooms use full-laptop computers or desktop computers. At this time, tablets and cell phones cannot be used to program these devices. Currently, each of the three major operating systems (Windows, MacOS, and Chrome) can be used to program the robotic platforms.

Since the Raspberry Pi platform can be considered a complete computing system, laptops or desktop computers are not required for NICERC's *Computer Science* course, but they are recommended to help with any heavy processing-required activities.

Potential Technology Barriers: Typically, the installation of Parallax's programming environments for the Boe-Bot or the Shield-Bot with Arduino require administrative privileges.

Environments and Programming Languages Used: *Cyber Literacy 1, Cyber Literacy 2, Cyber Science*, and *Cyber Physics* use the Parallax programming environment. The official programming language of Raspberry Pi, used in the *Computer Science* unit, is Python.

SUPPLEMENTAL MATERIALS

Each lesson within NICERC's library of content includes a supply list indicating what materials may be required. NICERC has partnered with Amazon to provide supply list items to teachers presenting the course. Links to the online-sourced resources are presented within the content's learning management system.

For a significant portion of the content, only "typical" classroom supplies (e.g., markers, pencils, colored pencils, rulers, scissors, tape, and glue) are needed. Other items may be needed; for example, the *Cyber Literacy* homemade battery project requires vinegar, copper wire, and steel nails. Teachers should preview lessons before presenting them in the classroom to ensure that they have the appropriate materials on hand.

OTHER NOTES

NICERC is currently preparing a ninth grade Algebra 1 curriculum that is scheduled for release in the spring or summer of 2018.

PLTW HIGH SCHOOL COMPUTER SCIENCE
» Project Lead The Way

Compiled by Karine Laidley
Director of Curriculum Development, PLTW

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✗ Meetups	✗ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✗ Full Semester	✓ Full Year	✓ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✗ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✓ Physical Computing	✓ Networking and Security	✓ Mobile Programming	✓ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✓ Android Device	✗ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✗ Purchasable Software				

OVERVIEW

The PLTW High School Computer Science program empowers students in grades 9–12 to become creators, instead of merely consumers, of the technology all around them. Students who experience PLTW High School CS learn to become independent learners and engage in a curriculum that is hands-on and student-driven, as the teacher becomes a facilitator of learning. In addition to CS knowledge and skills, the program teaches transferable skills, such as problem solving, persistence, creativity, collaboration, and communication.

PLTW High School CS comprises three courses:

- **Computer Science Essentials (CSE):** This full-year course exposes students to a diverse set of computational thinking concepts, fundamentals, and tools, allowing them to gain understanding and build confidence. Students use visual, block-based programming and seamlessly transition to text-based programming with languages such as Python to create apps and develop websites, and learn how to make computers work together to put their design into practice. They apply computational thinking practices, build their vocabulary, and collaborate just as computing professionals do to create products that address topics and problems important to them. Students without prior computing experience are encouraged to start with this course.

- **Computer Science Principles (CSP):** Using Python as a primary tool and incorporating multiple platforms and languages for computation, this full-year course aims to develop computational thinking, generate excitement about career paths that use computing, and introduce professional tools that foster creativity and collaboration. CSP helps students develop programming expertise and explore the workings of the Internet. Projects and problems include app development, visualization of data, cybersecurity, and simulation.
- **Computer Science A (CSA):** This full-year course focuses on further developing computational thinking skills through the medium of Android App development for mobile platforms. The course uses industry-standard tools, such as Android Studio, Java programming language, XML, and device emulators. Students collaborate to create original solutions to problems of their own choosing by designing and implementing user interfaces and Web-based databases.

PLTW curricula uses an activity-, project-, problem-based instructional approach, which offers a scaffold of learning experiences that students can then apply to solving an open-ended, real-world problem. Activities are designed for the acquisition of knowledge and skills and follow an appropriate learning progression for students. Projects provide opportunities for investigations and application of concepts and skills. Problems are designed to provide a common challenge that will typically result in unique solutions, which require the transfer of new and past knowledge and skills.

PLTW High School CS is part of the PLTW K–12 CS pathway and aligns to national standards and frameworks, such as CSTA and the College Board AP frameworks. PLTW is recognized by the College Board as an endorsed provider of curriculum and PD for AP Computer Science Principles. This endorsement affirms that all components of PLTW's CSP course are aligned to the AP Curriculum Framework standards and the AP CSP assessment. In addition, PLTW's CSA course aligns with the College Board's AP Computer Science A Framework and is also endorsed by the College Board.

RESOURCES

Professional Development (PD): PLTW high school training options include the following:

- *Core Training:* This in-depth, collaborative experience is offered for each High School CS course (CSE, CSP, and CSA). It covers course content, pedagogy, and facilitation support, and is designed to prepare teachers to deliver a transformative learning experience in their classrooms. Two-week (10-day) training for each unit costs $2,400/teacher and is offered at affiliate university sites across the United States.
- *Online Core Training:* Available for CSE only, this option offers a participant-centered and job-embedded experience. It is designed to develop teachers' understanding of course content and instructional practices, as well as to help teachers reflect on what's happening in their classrooms as they teach the course. Opportunities to participate and collaborate in a professional learning community will also provide support and feedback to help teachers implement the course in their classrooms. The online course includes two-hour live online coach-led weekly meetings, for 14 weeks; participants also collaborate in small groups during the online weekly sessions. The cost is $2,400/teacher.

Vendor or PD Provider Expectations: None

Ongoing Support: Regardless of which training teachers receive, PLTW offers phone support to schools; the PLTW online community of teachers, master teachers, and PLTW staff for ongoing support; PLTW representation in all 50 states via PLTW's Director of School Engagement; and PLTW online resources (including videos, tutorials, answer keys, and standards alignment documentation) to support teachers with content knowledge, facilitation, and assessment.

TECHNOLOGY

Required Hardware and Software and Their Costs: PLTW curricula are accessible online and can be downloaded locally on student tablets ahead of time. All PLTW courses are also available as downloadable PDF files, if needed.

- **Hardware:** The cost of the hardware for CSE is included in the materials cost (see the Supplemental Materials section below). For CSE, CSP, and CSA, schools are responsible for purchasing the tablets for their classrooms. One Android tablet for every two students, and one for the teacher, is recommended, as students will mostly work in pairs.
- **Software:** All software used in PLTW High School CS is free.

Potential Technology Barriers: PLTW curricula are online and accessible on desktops/laptops, Chromebooks, and mobile devices. Any videos that are available within the courses need to be streamed and require Internet connectivity. All High School CS courses require Internet connectivity for accessing their programming environments.

Programming Environments/Software Used: The PLTW High School CS course use MIT App Inventor, Canopy, Scratch, BlueJ, Android Studio, Cloud9, and VEX programming studio.

SUPPLEMENTAL MATERIALS

The table below shows the cost of the materials (both durable and consumable items) needed for each CS course, which are available at the PLTW Store (https://www.pltw.org/mypltwresources). These costs are based on a class of 20 students. Note that this does **not** include the cost of tablets or computers required for the classroom.

High School CS (Grades 9–12)	Year 1	Year 2	Year 3
CSE	$3,500	$75	$75
CSP	$750	$60	$60
CSA	$146	$0	$0

OTHER NOTES

The fee for schools to participate in High School CS is $2,000 per year. This covers participation in all three courses for that school. Schools may also apply for funding and grants opportunities through the PLTW Grant program.

Note: There will be a grant opportunity specifically for Massachusetts public schools through the PLTW Grant program. Details should be available on the PLTW the website in mid-October 2017.

ZULAMA COMPUTER SCIENCE PROGRAM OF STUDY

» Zulama, Inc.

Compiled by Nikki Navta
CEO, Zulama

PROPERTIES

Intended Grade Band	✗ K–5	✗ 6–8	✓ 9–12	✗ Other			
PD Model and Availability	✗ No Official PD	✓ Online PD (MOOC, etc.)	✓ Meetups	✓ Local Workshops	✓ Formal Institute and Follow-Up Sessions		
Curriculum Length	✗ Less than One Semester	✓ Full Semester	✓ Full Year	✗ Multi-Year			
Curriculum Integration	✓ Stand Alone Curriculum or Course	✓ Integrated Into Another Course					
Corresponding Pathways	✓ Software Engineering	✗ Physical Computing	✗ Networking and Security	✓ Mobile Programming	✓ Web Development	✓ Databases	✓ Game Design
Required Technology	✓ Desktop or Laptop	✗ iPad/iPhone	✗ Android Device	✓ Chromebook	✓ Works with Any Major OS	✓ Custom Hardware Device	
Required Software	✓ Browser Only	✓ Free Software	✓ Purchasable Software				

OVERVIEW

The Zulama Computer Science Program of Study leads students through the software development process using engaging, interest-driven, and project-based learning as students design, code, and continually improve their own digital games. Students investigate algorithms, learn how computers manage large amounts of data, explore the global impact of technology and the Internet, develop computational thinking skills, and acquire career-ready programming skills.

Three courses comprise this program of study:

- Introduction to Computer Science Through Game Design make CS relatable and help students see immediate results in what they're coding. This course introduces students to such concepts as loops, variables, conditionals, and arrays in an easy, step-by-step method that builds their confidence. Students develop original, portfolio-worthy games that can be used to apply for higher education or employment opportunities.
- Advanced Placement (AP) Computer Science Principles develops the computational thinking practices students use to problem-solve and accomplish goals. Students learn to code by completing fun, rigorous game projects. They plan, design, code, and test software using GameMaker Studio. Students who successfully complete this course will be prepared to take the AP CS Principles Exam. The course syllabus is pre-approved by the College Board.

- Unity 3D Programming gives students an opportunity to learn the Unity 3D interface and how to program in either C# or JavaScript. Students apply their growing Unity 3D knowledge to games and to immersive real-life projects. Students explore art and animation concepts that are integral to designing game levels and game environments. Students who successfully complete this course will be prepared to take the Unity Developer Certification Exam.

The curriculum was developed with faculty from the Entertainment Technology Center at Carnegie Mellon University and is being offered in over 100 districts nationwide.

The courses are aligned with the national CSTA Standards, the AP CS Principles Curriculum Framework, and the Unity Developers Exam. Crosswalks with learning standards, including the Massachusetts state standards, can be found on the Zulama website (http://zulama.com/our-program/standards/).

All required materials are online. More information, including course syllabi, scope and sequence, and examples of student work can be found on the Zulama website (http://zulama.com/how-it-works/computer-science/).

RESOURCES

Professional Development (PD): Zulama offers professional learning that is interactive, project-based, and participant-driven. A variety of professional learning opportunities, for example:

- Online self-paced teacher training, approximately four hours per course. Cost: $500 per course per teacher.
- Two-hour implementation meetings for teachers, administrators, and staff, online or onsite. Cost: $500 per session, for up to 50 attendees.
- One-day in-person, hands-on, and project-based teacher training. Cost: $2,200 per day, for up to 50 participants.

Vendor or PD Provider Expectations: Teachers do not need prior programming experience to teach these courses. Zulama requires that any interested schools include Zulama on the school's master schedule for all students, use the correct course code, publicize the course, and ensure adequate enrollment.

Ongoing Support: The Zulama Learning and Content Management System provides an active, robust online forum that connects teachers and gives them access to an extensive database of "how to" videos and other resources. Teacher notes are included throughout all lessons to provide even more explanation and background. A completed project file ("answer key") is provided for every lesson. Teachers maintain access to all online course-specific training while using the curriculum, so they can brush up on their skills at any time.

TECHNOLOGY

Required Hardware and Software and Costs: The per-classroom cost for each course is $1,500. Unlimited student and teacher access to all three courses costs $2,450.

- **Hardware:** Internet-connected desktops or laptops are required, a minimum of one per classroom and ideally one device per student.
- **Software:** A modern browser, either Mac OS or Windows, is required. GameMaker Studio ($25 per device) is required for the "Introduction to Computer Science Through Game Design" and "AP Computer Science Principles" courses. Unity 3D (free) is required for the "Unity 3D Programming" course.

Environments and Programming Languages Used: "Introduction to Computer Science Through Game Design" and "AP Computer Science Principles" use GameMaker Studio and GML (a proprietary programming language similar to Python). "Unity 3D Programming" uses Unity 3D, C#, and JavaScript.

CONSOLIDATED PROPERTIES CHART

Curriculum Name	Intended Grade Band						PD Model & Availability				Curriculum Length		Curriculum Integration		Corresponding Pathways							Required Technology					Required Software				
	K - 5	6 - 8	9 - 12	Other	No Official PD	Online PD (MOOC, etc.)	Meetups	Local Workshops	Formal Institute/Follow Ups	Less than One Semester	Full Semester	Full Year	Multi-Year	Stand alone Curriculum/Course	Integrated into Another Course	Software Engineering	Physical Computing	Networking & Security	Mobile Programming	Web Development	Databases	Game Design	Desktop or Laptop	iPad/iPhone	Android Device	Chromebook	Works with Any Major OS	Custom Hardware Device	Browser Only	Free Software	Purchasable Software
AP Computer Science A			X						X		X	X		X		X		X					X				X		X	
AP Computer Science Principles (Code.org)			X					X				X		X		X		X		X			X						X	
Beauty and Joy of Computing (AP CSP)			X	X		X	X		X			X		X		X		X		X			X						X	
Bootstrap		X	X					X		X					X							X				X		X		
Codecademy		X	X	X				X		X				X		X						X				X				
Computational Thinking and Problem Solving (CTPS)			X						X			X		X		X		X		X			X						X	
Computer Science Discoveries		X	X					X			X	X		X		X		X					X						X	
Creative Computing		X	X			X	X				X	X		X		X							X						X	
CS Fundamentals			X				X	X			X	X	X	X		X							X						X	
Edison Robots		X	X		X					X					X		X												X	
Exploring Computer Science (ECS)		X	X						X		X	X		X		X	X	X		X			X						X	
Finch Robot		X	X		X			X		X				X		X	X						X						X	
Khan Academy - Computing		X	X	X						X	X	X	X	X		X				X			X						X	
KIBO Robot					X			X		X	X		X	X		X													X	
LEGO WeDo Construction Kit		X			X				X	X	X			X		X													X	
LEGO Mindstorms EV3			X					X		X	X			X		X													X	
Micro:bit's Intro to CS			X		X					X	X			X		X		X											X	
Middle School Pathways in Computer Science		X			X					X					X								X		X					
Middle Years Computer Science (MyCS)		X	X		X						X			X		X				X			X			X			X	
Mobile Computer Science Principles (Mobile CSP)			X			X	X		X			X		X		X		X		X			X			X			X	
MOS Elementary Computer Science		X			X			X		X				X		X							X			X			X	
NICERC Cyber and Computer Science for HS			X				X		X		X	X	X	X		X		X					X			X			X	
NICERC STEM: Explore, Discover, Apply for MS		X		X			X		X		X	X	X	X		X							X			X			X	
Project Lead The Way (PLTW) Gateway		X			X						X		X			X		X					X			X			X	
Project GUTS CS in Science Curriculum		X				X		X		X				X		X							X			X			X	
Project Lead The Way (PLTW) High School CS			X		X				X			X		X		X		X		X			X			X			X	
Project Lead The Way (PLTW) Launch		X			X			X			X		X			X								X						
ScratchJr		X					X	X			X			X		X							X			X			X	
STEM+C Integrated Modules		X			X					X					X								X							
Zulama Computer Science Program of Study			X			X	X		X			X		X		X		X		X	X		X						X	X
Zulama Game Design Fundamentals		X				X	X		X		X							X												X

ACKNOWLEDGMENTS

Padmaja Bandaru
Computer Science Teacher, AMSA Charter School, and Co-President, CSTA-Greater Boston

Meg Bednarcik
Instructor of Computer Science, Phillips Academy

Deborah Boisvert
Executive Director, BATEC at University of Massachusetts Boston

Karen Brennan
Associate Professor, Harvard Graduate School of Education

Patricia Clark
Teacher, Nashoba Regional High School

Terry Dash
Teacher, Arlington Public Schools

Anne DeMallie
Computer Science and STEM Integration Specialist, DESE

Damian DeMarco
Computer Science Teacher, Revere Public Schools

Kathi Fisler
Bootstrap Co-Director, and Research Professor of Computer Science, Brown University

Mary Fries
Curriculum and Instructional Design Associate, EDC

Dr. Chuck Gardner
Director of Curriculum, National Integrated Cyber Education Research Center (NICERC)

Melody Hagaman
Project Manager, Project GUTS

Farzeen Harunani
Coordinator of District and Teacher Engagement, MassCAN, EDC

Jason Innes
Manager of Training and Curriculum Development, KinderLab Robotics, Inc.

Jessica Jarboe
Teacher, Milton High School

Audra Kaplan
Digital Learning Coach, Groton-Dunstable Regional School District

Karine Laidley
Director of Curriculum Development, Project Lead The Way (PLTW)

Irene Lee
Research Scientist, MIT's Scheller Teacher Education Program, and PI, Project GUTS

Maria Litvin
Instructor of Mathematics and Computer Science, Phillips Academy, and Code.org K–5 Facilitator

Joyce Malyn-Smith
Director of Strategic Initiatives in Workforce and Human Development, EDC

Fred Martin
Professor, Computer Science, University of Massachusetts Lowell

Chad McGowan
Computer Science Teacher and Technology Coach, Ashland High School, and Code.org Facilitator

Paula Moore
Computer Science Teacher, Westwood Public Schools

Ralph Morelli
Professor of Computer Science, Emeritus, Trinity College, and Co-PI, Mobile CSP

Nikki Navta
CEO, Zulama

Kaitie O'Bryan
Math and Computer Science Teacher, Mounds View Public Schools, Minnesota; Code.org Facilitator and Forum Moderator

Laura Peters
Research Project Manager, Creative Computing Lab, Harvard Graduate School of Education

David C. Petty
Teacher, Winchester High School, and Co-President, CSTA-Greater Boston

Shaileen Crawford Pokress
K–12 Computer Science Education Consultant

Jennifer Rosato
Assistant Professor, Computer Information Systems, College of St. Scholastica, and Co-PI, Mobile CSP

Ivan Rudnicki
Computer Science Department Chair, Edward Brooke Charter Schools

Mark Sherman
Postdoctoral Research Associate, MIT App Inventor

Jim Stanton
Executive Director, MassCAN, and Senior Project Director, EDC

Susanne Steiger-Escobar
Professor, Computer Science, Massachusetts Bay Community College

Steve Vinter
Tech Leadership Advisor and Coach, Google

Peter Y. Wong, Ph.D.
Director of University Relations, Museum of Science, Boston

Emma Youndtsmith
Midwest Regional Manager and Facilitator, Bootstrap

EDC
43 Foundry Avenue
Waltham, MA 02453

edc.org

Boston | Chicago | New York | Washington, D.C.