Connecticut Computer Science Implementation Guidelines

2018

CSDE
CONNECTICUT STATE
DEPARTMENT OF EDUCATION

Connecticut State Department of Education

Contents

The Connecticut State Department of Education is committed to a policy of equal opportunity/ affirmative action for all qualified persons. The Connecticut State Department of Education does not discriminate in any employment practice, education program, or educational activity on the basis of race, color, religious creed, sex, age, national origin, ancestry, marital status, sexual orientation, gender identity or expression, disability (including, but not limited to, intellectual disability, past or present history of mental disorder, physical disability or learning disability), genetic information, or any other basis prohibited by Connecticut state and/or federal nondiscrimination laws. The Connecticut State Department of Education does not unlawfully discriminate in employment and licensing against qualified persons with a prior criminal conviction. Inquiries regarding the Connecticut State Department of Education's nondiscrimination policies should be directed to: Levy Gillespie, Equal Employment Opportunity Director/Americans with Disabilities Act Coordinator, Connecticut State Department of Education, 450 Columbus Blvd, Suite 607, Hartford, CT 06103, 860-807-2071, Levy.Gillespie@ct.gov.

Connecticut Computer Science Standards Workgroup

Jon Bishop, K-12 Stem Coordinator, Canton Public Schools

Jennifer Blalock, High School Mathematics and Computer Science Teacher, Ellington Public Schools

Jacqueline Corricelli, High School Computer Science Teacher, West Hartford Public Schools

Michael Cwirka, High School Teacher, Berlin Public Schools

Elizabeth W. Dillard, High School Computer Science Teacher, CREC

Dr. Melissa Hickey, Reading/Literacy Director, Connecticut State Department of Education

Christopher J Kerr, High School Computer Science Teacher, Newington Public Schools

Dana Kinel, IB Design Technology Teacher, East Hartford Public Schools

Eric Lozaw, High School Teacher, Watertown Public Schools

Jenny Lussier, Library Media Specialist, Regional School District 13

Lanna Mack, Career & Technical Education Teacher, New Haven Public Schools

Jennifer Michalek, Education Consultant, Connecticut State Department of Education

Dario Soto, Elementary Teacher, Hartford Public Schools

Heather Sutkowski, Elementary Computer Science Teacher, CREC

Dr. Chinma Uche, President, Connecticut Computer Science Teachers Association

James Veseskis, Project Coordinator, Exploring Computer Science CT

David Weinreb, Bilingual Teacher, New Haven Public Schools

Introduction

The Connecticut State Board of Education (CSBE) believes that computer science is a key to developing and integrating 21st Century Skills (e.g., technology, communication, collaboration, critical thinking, problem solving, innovation, creativity, persistence). The CSBE further believes that all Connecticut public schools must provide for challenging and rigorous programs of study in computer science across all grade levels. As such, Connecticut recommends fully adopting the Computer Science Teachers Association K-12 Computer Science Standards as the Connecticut Computer Science Standards. This implementation guidance document articulates the lens through which to view these standards and provides guidance for implementation across the State of Connecticut.

Background

While advocating for CS education for all students, the Computer Science Teachers Association (CSTA) saw the need to define CS and provide a guiding document for its members, administrators and policy makers. Both within and outside the USA, this document provides information on how to implement CS in the K-12 space. The first CSTA K-12 Standards were developed in 2011 and were adopted by the few states in the USA that were teaching CS, and other locations abroad. As CS continued to influence technology and our world, new teaching tools for CS education were developed. It then became necessary to review the CSTA K-12 Standards and update them. A team of CS professionals (composed of teachers, administrators, and members of industry) was constituted in September 2015 to review and update the Standards for the 2015-2016 year. While the review was in progress, CSTA joined forces with other CS organizations (Association for Computing Machinery, Code.org, Cyber Innovation Center, and National Math + Science Initiative) to develop a K-12 CS Framework. This framework identified the core areas of CS that these organizations agreed needed to be represented in a K-12 CS classroom. The ideas included in the Framework were used as input into the revision of the CSTA standards. The CSTA K–12 Computer Science Standards delineate a core set of learning objectives designed to provide the foundation for a complete computer science curriculum, implemented at the K–12 level. To this end, the CSTA Standards:

- Introduce the fundamental concepts of computer science to all students, beginning at the elementary school level.
- Present computer science at the secondary school level in a way that can fulfill a computer science, math, or science graduation credit.
- Encourage schools to offer additional secondary-level computer science courses that will allow interested students to study facets of computer science in more depth, and prepare these students for entry into the workforce or college.
- Increase the availability of rigorous computer science courses for all students, especially those who are members of underrepresented groups.

Explanation of the process

The original 2011 CSTA K–12 CS Standards were categorized into five conceptual strands: Computational Thinking; Collaboration; Computing Practice & Programming; Computer & Communication Devices; and Community, Global & Ethical Impacts. The 2017 CSTA K–12 CS Standards are categorized into the five concepts of the K–12 CS Framework (Computing Systems, Networks and the Internet, Algorithms and Programming, Data and Analysis, and Impacts of Computing), and utilize the seven outlined computational practices (Fostering an Inclusive Computing Culture, Collaborating Around Computing, Recognizing and Defining Computational Problems, Developing and Using Abstractions, Creating Computational Artifacts, Testing and Refining Computational Artifacts, and Communicating About Computing). Both the Framework and the CSTA Standards underwent three public review periods, for two to three weeks via online forms, for scrutiny and feedback from anyone with interest or experience in K–12 computer science education. The feedback from each review period was read and addressed by the development team, ensuring that the current CSTA K-12 standards is the best peer-reviewed standard for K-12 CS education.

The Connecticut Department of Education (CSDE) convened a group of educators charged with putting forward CS standards for State Board of Education (Board) approval. These educators were divided into grade level teams, K-5, 6-8 and 9-12. Each team independently reviewed the CSTA K-12 standards. The review by each team concluded that these standards aligned to the beliefs contained with the previously adopted Position Statement on Computer Science Education for All Students K-12. It was recommended by the teams that these standards be brought forth to the Board for adoption and that feedback from stakeholders be elicited.

Stakeholder engagement

The CSTA K-12 Standards have been widely received by the computer science education and business communities, as well as policy developers. The standards are currently being used to define CS in many states across the USA. In Connecticut a survey about the standards was disseminated to a variety of stakeholders. This survey provided stakeholders the opportunity to give their feedback in regards to the standards. The survey was made publicly available and responses were collected over a six week period. Respondents included teachers, administrators, parents, higher education and business and industry. The results of the survey were favorable for adopting the standards for Connecticut.

What is Computer Science?

As the foundation for all computing, computer science is defined as “the study of computers and algorithmic processes, including their principles, their hardware and software designs, their [implementation], and their impact on society” (Tucker et. al, 2003, p. 6). Computer science is apart from and builds on computer literacy, educational technology, digital citizenship, and information technology.

These aspects of computing are distinguished from computer science because they are focused on the use of computer technologies rather than understanding why they work and how to create those technologies. Knowing why and how computers work (i.e., computer science), provides the basis for a deep understanding of computer use and the relevant rights, responsibilities, and applications. Computer science is the study of how computers and computational systems make representations of the world around them and how to apply the rules, procedures, and processes to create solutions to challenges.

What is Computational Thinking?

Also integrated throughout the Computer Science standards is the concept of computational thinking. Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent [Cuny, Snyder & Wing, 2010]. It is an approach to solving problems in a way that can be implemented with a computer. It involves the use of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts [Computer Science Teachers Association & Association for Computing Machinery]. Computational thinking practices such as abstraction, modeling, and decomposition connect with computer science concepts such as algorithms, automation, and data visualization. Beginning with the elementary school grades and continuing through grade 12, students should develop a foundation of computer science knowledge and learn new approaches to problem solving that captures the power of computational thinking to become both users and creators of computing technology.

Equity

Equity is a fundamental component in the development of Computer Science Standards. The intent of equity is to ensure that all students have the basic knowledge that will allow them to productively participate in the world and make well informed decisions about their lives. Classrooms often include students of different race, students of different genders, students of different socioeconomic status, students whose first language is not English, students with disabilities, and students with differing ways of learning. Regardless of these differences, all students have the right to high quality computer science education.

Equity is not limited to whether classes are available, but also includes how classes are taught, how students are recruited for classes or activities, and how the classroom culture supports diverse learners and promotes the retention of students. The result of equity is achieving the ability to meet the needs of diverse learners. Equity allows students set high expectations and feel capable of learning. It ensures that all students have the basic knowledge that will allow them to compete in a diverse world. Additional efforts at the state level, concerning educational policies paired with commitment at the local level with curriculum, instruction, and classroom culture are also necessary for equity to play a role in computer science education.

Computer Science Practices

The CSTA K-12 Computer Science Standards incorporate seven practices. By Grade 12, it is expected that every computationally literate student will engage with these practice behaviors as they learn the standards and develop computational artifacts. The interrelated practices are listed in the chart below in an order that simulates the developmental process taken to produce computational artifacts.

Identifier	Practice
P1	Fostering an Inclusive Computing Culture
P1.1	Include the unique perspectives of others and reflect on one’s own perspectives when designing and developing computational products.
P1.2	Address the needs of diverse end users during the design process to produce artifacts with broad accessibility and usability.
P1.3	Employ self- and peer-advocacy to address bias in interactions, product design, and development methods.
P2	Collaborating Around Computing

P2.1	Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.
P2.2	Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.
P2.3	Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.
P2.4	Evaluate and select technological tools that can be used to collaborate on a project.
P3	Recognizing and Defining Computational Problems
P3.1	Identify complex, interdisciplinary, real-world problems that can be solved computationally.
P3.2	Decompose complex real-world problems into manageable subproblems that could integrate existing solutions or procedures.
P3.3	Evaluate whether it is appropriate and feasible to solve a problem computationally.
P4	Developing and Using Abstractions
P4.1	Extract common features from a set of interrelated processes or complex phenomena.
P4.2	Evaluate existing technological functionalities and incorporate them into new designs.
P4.3	Create modules and develop points of interaction that can apply to multiple situations and reduce complexity.
P4.4	Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.
P5	Creating Computational Artifacts
P5.1	Plan the development of a computational artifact using an iterative process that includes reflection on and modification of the plan, taking into account key features, time and resource constraints, and user expectations.
P5.2	Create a computational artifact for practical intent, personal expression, or to address a societal issue.
P5.3	Modify an existing artifact to improve or customize it.
P6	Testing and Refining Computational Artifacts
P6.1	Systematically test computational artifacts by considering all scenarios and using test cases.
P6.2	Identify and fix errors using a systematic process.

P6.3	Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.
P7	Communicating About Computing
P7.1	Select, organize, and interpret large data sets from multiple sources to support a claim.
P7.2	Describe, justify, and document computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.
P7.3	Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

Organization of the Standards

Grade bands

Standards are organized into grade bands with the goal being that students will have met the expectations by the end of grade 2 (Level 1A, ages 5-7), the end of grade 5 (Level 1B, ages 8-11), the end of grade 8 (Level 2, ages 11-14) and the end of grade 10 (Level 3A, ages 14-16). Furthermore, for students who wish to study computer science in high school beyond the level required for all students, Level 3B is provided.

Strands

Standards are also organized into strands called Concepts and Subconcepts. There are five Concepts: Algorithms & Programming, Computing Systems, Data & Analysis, Impacts of Computer, and Networks & the Internet, which are further broken down into sixteen Subconcepts. The chart below provides a brief overview of each sub concept for further clarification. In addition, there are five cross-cutting topics that are interwoven within each core concept throughout the standards, but do not have stand-alone descriptions, including Abstraction, System Relationships, Human- Computer Interaction, User Inspired Software Design, Privacy and Security, and Communication and Coordination. The vertically aligned standards are intended to reflect a comprehensive instructional program and document a progression of expected achievement in each of the strands. This organization of standards also reflects the gradual progression in the development of skills.

Concept	Sub concept	Overview
Algorithms and Programming	Algorithms	People evaluate and select algorithms based on performance, reusability, and ease of implementation. Knowledge of common algorithms improves how people develop software, secure data, and store information.
	Control	Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures.

Algorithms and Programming	Modularity	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures or independent, but interrelated, programs. Modules allow for better management of complex tasks.
	Program Development	Diverse teams can develop programs with broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs. The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.
	Variables	Data structures are used to manage program complexity. Programmers choose data structures based on functionality, storage, and performance tradeoffs.
Computing Systems	Devices	Many everyday objects contain computational components that sense and act on the world. In early grades, students learn features and applications of common computing devices. As they progress, students learn about connected systems and how interaction between humans and devices influences design decisions.
	Hardware and Software	Computing systems use hardware and software to communicate and process information in digital form. In early grades, students learn how systems use both hardware and software to represent and process information. As they progress, students gain a deeper understanding of the interaction between hardware and software at multiple levels within computing systems.
	Troubleshooting	When computing systems do not work as intended, troubleshooting strategies help people solve the problem. In early grades, students learn that identifying the problem is the first step to fixing it. As they progress, students learn systematic problem-solving processes and how to develop their own troubleshooting strategies based on a deeper understanding of how computing systems work.
Data and Analysis	Collection, Visualization, and Transformation	Data is collected with both computational and non-computational tools and processes. In early grades, students learn how data about themselves and their world is collected and used. As they progress, students learn the effects of collecting data with computational and automated tools.
Data and Analysis	Inference and Models	Data science is one example where computer science serves many fields. Computer science and science use data to make inferences, theories, or predictions based upon data collected from users or simulations. In early grades, students learn about the use of data to make simple predictions. As they progress, students learn how models and simulations can be used to examine theories and understand systems and how predictions and inferences are affected by more complex and larger data sets.
	Storage	Data can be composed of multiple data elements that relate to one another. For example, population data may contain information about age, gender, and height. People make choices about how data elements are organized and where data is stored. These choices affect cost, speed, reliability, accessibility, privacy, and integrity.
Impacts of Computing	Culture	The design and use of computing technologies and artifacts can improve, worsen, or maintain inequitable access to information and opportunities.

Impacts of Computing	Safety, Law and Ethics	Laws govern many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights. International differences in laws and ethics have implications for computing.
	Social Interactions	Many aspects of society, especially careers, have been affected by the degree of communication afforded by computing. The increased connectivity between people in different cultures and in different career fields has changed the nature and content of many careers.
Networks and the Internet	Cybersecurity	Transmitting information securely across networks requires appropriate protection. In early grades, students learn how to protect their personal information. As they progress, students learn increasingly complex ways to protect information sent across networks.
	Network Communication and Organization	Computing devices communicate with each other across networks to share information. In early grades, students learn that computers connect them to other people, places, and things around the world. As they progress, students gain a deeper understanding of how information is sent and received across different types of networks.

Implementation Models

In the following examples, a computer science experience can range from a few hours a week to a semester- or year-long course. *Integrated or as a stand-alone based on student and district readiness*

SAMPLE K-12 COMPUTER SCIENCE PATHWAYS

	Broad and Deep Exposure	Moderate Exposure	Basic Exposure
Elementary School	Independent Special (Similar to Music, Art, etc.)	Integrated into the general classroom	Integrated into the general classroom
Middle School	Integrated into math, science, other subjects + Independent course at a particular grade level	Independent course at a particular grade level	Integrated into math, science, other subjects
High School	Introductory course + AP Computer Science + Specialized courses	Introductory course + Specialized courses	Introductory course

Elementary and Middle School

Computer Science (CS) at the K-2, 3-5, and 6-8 grade bands can be embedded within the curriculum and/or offered as a ‘standalone’ course, depending on the school’s program. This flexible implementation allows schools the choice to determine their own timeline on how they will ensure that all students will have the opportunity to learn CS. All certified staff members and subject areas are encouraged to integrate computer science instruction into their classrooms.

Below are various suggestions for implementation:

- Integrate CS into a particular subject area (i.e. math, science, technology) on a weekly or biweekly basis within elementary classrooms.
- Plan districtwide and schoolwide participation in the annual "Hour of Code" for all grade levels.
- Offer CS in a particular grade level and then expand the program to additional grade levels in subsequent years.
- Provide small group instruction.
- Provide a weekly "specials"/ "unified arts" course designed to specifically teach the CS standards.
- Integrate CS instruction into existing Library/Media time.
- Incorporate CS in a similar fashion as Maker Spaces, Genius Hour etc.

High School

Implementation at the high school level is best achieved through course offerings specific to CS. All high schools should offer at least one rigorous computer science course. Ideally high schools develop CS pathways for students to explore based on need and interest.

Resources

A sampling of resources is provided to assist districts in making curriculum decisions related to the implementation of CS. This is not an all-inclusive list of resources and they are not endorsed by CSDE. They are presented to districts as guidance about quality CS curriculum. Implementing CS curriculum requires many decisions.

- Schools may decide to teach a specific curriculum or combine multiple resources to deliver CS instruction.
- Computer science instruction can be implemented with limited access to technology. Students are encouraged to work together and can share devices.
- Computer science can be individualized and collaborative.
- "Unplugged" lessons are available and districts are encouraged to use a combination of 'plugged" for an authentic CS experience.
- 20 hours of Computer Science instruction per year will provide a rigorous and well-developed experience for students at the elementary and middle school levels.
- One credit or its equivalent in computer science at the high school level will best prepare students to be college and career ready.

Elementary School

Organization	Curriculum
Apple	The lessons in the Get Started with Code Teacher Guides, which are part of the Everyone Can Code Curriculum, are designed to help you bring coding into the early primary classroom.
Bee-Bot	Bee-Bot Lessons contains 100 detailed lesson plans, with accompanying images, for using Bee-Bot to teach across the curriculum. Problem-Solving with Bee-Bot provides 150 sequential student challenges that use Bee-Bot to develop problem-solving, critical-thinking, and decision-making skills.
codeSpark Academy	Ignite interest in computer science and turn programming into play.
Code Studio(Code.org)	Computer Science Fundamentals is comprised of 6 courses of about 15 lessons that may be implemented as one unit or over the course of a semester.
Code Monkey	The Code Monkey game is accompanied by a curriculum guide which includes 35 detailed lesson plans with both online and offline activities.
Computer Science for All in SF	Creative Computing Curriculum for K – 2 and 3 – 5 introduces computer science as a creative, collaborative, and engaging discipline across 15 – 20 lessons at each grade level.
Kodable	Courses for every grade K – 5 enabling students to learn foundational skills in computer science preparing them for the next step in their learning.

Organization	Curriculum
Project Lead The Way	PLTW Launch modules engage students and build knowledge and skills in the area of computer science.
ScratchEd	Activities are designed to support familiarity and increasing fluency with computational creativity and computational thinking using Scratch. Units can be used as a semester-long computing course or as part of other curriculum areas.
Tynker	Seven coding courses designed for students K – 5.

Middle School

Organization	Curriculum
Apple	The lessons in the Learn to Code Teacher Guides, which are part of the Everyone Can Code Curriculum, are designed to help students learn fundamental coding concepts.
Bootstrap	Teach algebra through video-game programming, with a module to go alongside or inside a math class.
CodeHS	CodeHS helps schools and districts build a comprehensive Middle School computer science program starting with introductory level block-based programming courses. There are courses available for all grades 6-8.
Code.org	Computer Science Discoveries is an introductory computer science course recommended for grades 6 - 10 that empowers students to

Organization	Curriculum
	create authentic artifacts and engage with computer science as a medium for creativity, communication, problem solving, and fun.
Code Monkey	The Code Monkey game is accompanied by a curriculum guide which includes 35 detailed lesson plans with both online and offline activities.
Codesters	Range of courses where students use Python to build projects through structured lessons, then modify their code to create custom projects.
Computer Science for All in SF	MyCS intended for grade 6 and App Inventor for grade 7 highlight the personal relevance of computer science to middle school students and attempt to present CS as a fun, creative, and collaborative discipline.
Edhesive	Explorations in Coding course is specifically designed for middle school classrooms, this blended online course covers foundational concepts and skills of computer science.
Globaloria	Standalone and core subject integration pathways.
GUTS	In partnership with Code.org, Middle School CS in Science includes four modules each consisting of five or six lessons.
Project Lead The Way	PLTW Gateway units include units that engage students in computer science through robotics, hardware and software development, and mobile app development.
Pythonroom	Pythonroom's curriculum is designed to teach students problem-solving and algorithmic thinking while introducing computer science to young students

Organization	Curriculum
ScratchEd	Activities are designed to support familiarity and increasing fluency with computational creativity and computational thinking using Scratch. Units can be used as a semester-long computing course or as part of other curriculum areas.
Tynker	Seven coding courses designed for students 6 - 8.
UC Davis C-STEM	Multiple academic year-long courses on computing in math, programming, robotics, and film production.

High School

Organization	Curriculum
Apple	The Intro to App Development with Swift and App Development with Swift curricula were designed to teach high school students with little or no programming experience how to be app developers, capable of bringing their own ideas to life.
Beauty and Joy of Computing	Introductory computer science curriculum intended for high school juniors and seniors that is aligned to the AP CS Principles course.
Bootstrap	Teach algebra through video-game programming, with a 20-hr module to go alongside or inside a math class
CodeHS	4-year high school CS pathway. Intro CS JavaScript, Intro CS Python, AP CS Principles, AP CS in Java, Computing Ideas, Web Design and more.

Organization	Curriculum
Code.org	Two years of Computer Science courses for beginners. The first course, Computer Science Discoveries, is appropriate for grades 6-10 and the second, Computer Science Principles, can be implemented as an AP course or an introductory course.
Edhesive	Year-long AP Computer Science courses.
Exploring Computer Science	Year-long introductory high school course aimed at broadening participation in CS.
Globaloria	Standalone and core subject integration pathways.
Mobile CSP	Mobile CSP is a College Board-endorsed AP Computer Science Principles curriculum
Project Lead The Way	PLTW Computer Science engages students in true-to-life activities like creating an online art portal or developing problem-solving apps.
ScratchEd	Activities are designed to support familiarity and increasing fluency with computational creativity and computational thinking using Scratch. Units can be used as a semester-long computing course or as part of other curriculum areas.
TEALS	TEALS helps high schools build and grow sustainable computer science programs by pairing experienced and trained software engineer professionals with classroom teachers. TEALS has two standard high school course offerings and offers support for additional courses.

Organization	Curriculum
UC Davis C-STEM	Multiple academic year-long courses on computing in math, programming, and robotics.
UTeach CS Principles	A classroom-ready curriculum that is fully aligned with the College Board's AP Computer Science Principles framework and endorsed by the College Board.

Additional resources to support CS curriculum and instruction can be accessed on the Connecticut Computer Science Teachers' Association website.

References

Computer Science Teachers Association (2017). CSTA K-12 Computer Science Standards, Revised 2017. Retrieved from CS Teachers.

"K–12 Computer Science Framework." *k12cs.Org*, Computer Science Teachers Association, k12cs.org/.

Education, Virginia Department of. "Computer Science." *VDOE :: Computer Science Standards of Learning Resources*, Virginia Department of Education, Nov. 2017, www.doe.virginia.gov/testing/sol/standards_docs/computer-science/index.shtml.

"Anybody Can Learn." *Code.org*, Code.org, code.org/.

"CSTA." *CSTA*, csteachers.org/.

"Proposed Nevada K-12 Computer Science Standards." www.doe.nv.gov/uploadedFiles/nde.doe.nv.gov/content/Standards_Instructional_Support/Nevada_Academic_Standards/Comp_Tech_Standards/DRAFTNevadaK-12ComputerScienceStandards.pdf.

"3rd Party Educator Resources." *CSEd Week*, csedweek.org/educate/curriculum/3rd-party.

"ISTE - International Society for Technology in Education - Home." *ISTE - International Society for Technology in Education - Home*, www.iste.org/.