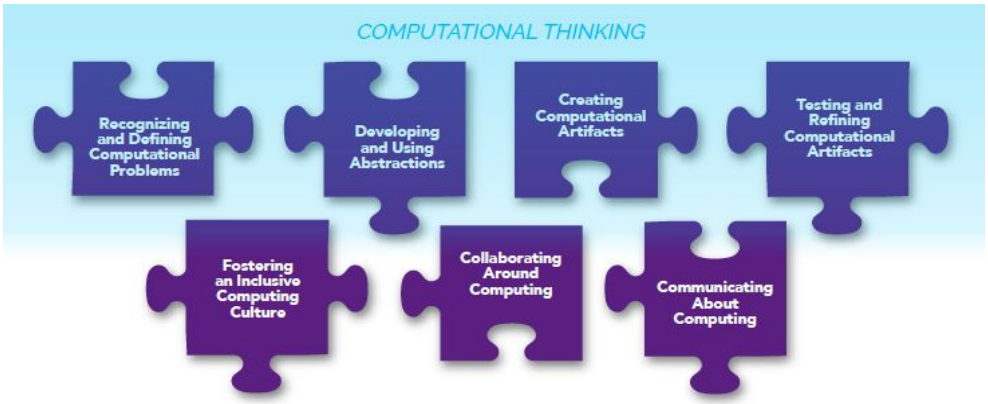
California's K-12 Computer Science Standards

unanimously adopted by the State Board of Education on September 6th, 2018

The Department of Education sets **standards** as learning goals for what students should know and be able to do at different grade levels. California's first-ever computer science (CS) standards were designed to inform teachers, curriculum developers, and educational leaders to ensure all students receive quality computer science instruction. These standards are **model** standards, meaning they are recommended but not mandated.

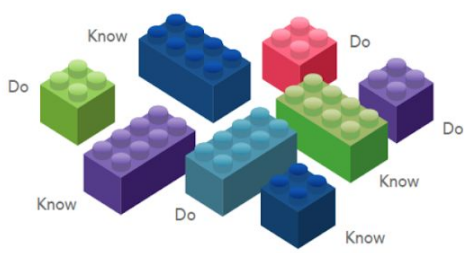
California's standards are aligned to the national ***K-12 Computer Science Framework*** (k12cs.org). The *Framework* defines computer science as **five core concepts** (what students should **know**). While some people think CS as only programming, there are four other, equally crucial concepts, including data, networks, computing systems, and impacts of computing.

Additionally, there are **seven core practices** that specify what students should be able to **do**. Four of these practices comprise **computational thinking** (CT), and the others support CT.

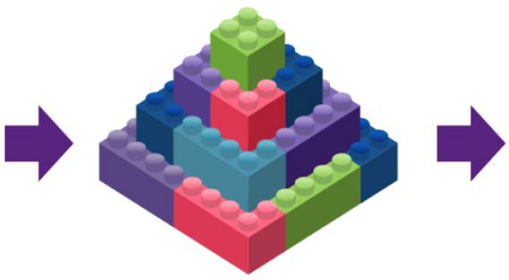


California's standards fit these concepts and practices pieces together to set specific learning goals by grade. Schools and districts choose their own **curriculum**, which specifies how students will learn and meet the standards.

FRAMEWORK: know, do



STANDARDS: know and do



CURRICULUM: how to teach

(units, courses, projects, activities)

Overview created by Bryan Twarek, co-chair of the Computer Science Standards Advisory Committee: twarekb@sfusd.edu

Standards are created by combining concepts with practices. Here is a sample standard:

Practice		Concept		Standard
Developing and Using Abstractions Model phenomena and processes and simulate systems to understand and evaluate potential outcomes.	+	**Networks & the Internet** By the end of 8th grade… Computers send and receive information based on a set of rules called protocols. Protocols define how messages between computers are structured and sent…	=	**Networks & the Internet** (6-8.NI.4) Model the role of protocols in transmitting data across networks and the Internet.

California's K-12 computer science standards are vertically aligned, coherent across grades, and designed in developmentally appropriate grade spans: K-2, 3-5, 6-8, 9-12. Because the standards are **designed for ALL students**, they are referred to as **core.**

The high school (9-12) grade span also includes an additional set of standards, referred to as **9-12 specialty**, which provides options for extending a CS pathway with content containing increased complexity and depth. The 9-12 specialty standards may be used to create more advanced courses, including career technical education (CTE), Advanced Placement (AP), dual credit, and other specialty courses (e.g., cybersecurity, robotics, web design, machine learning).

The standards include many **interdisciplinary connections** and examples that demonstrate diverse means of implementation. For example:

3-5.AP.19: Compare and refine multiple algorithms for the same task and determine which is the most appropriate.

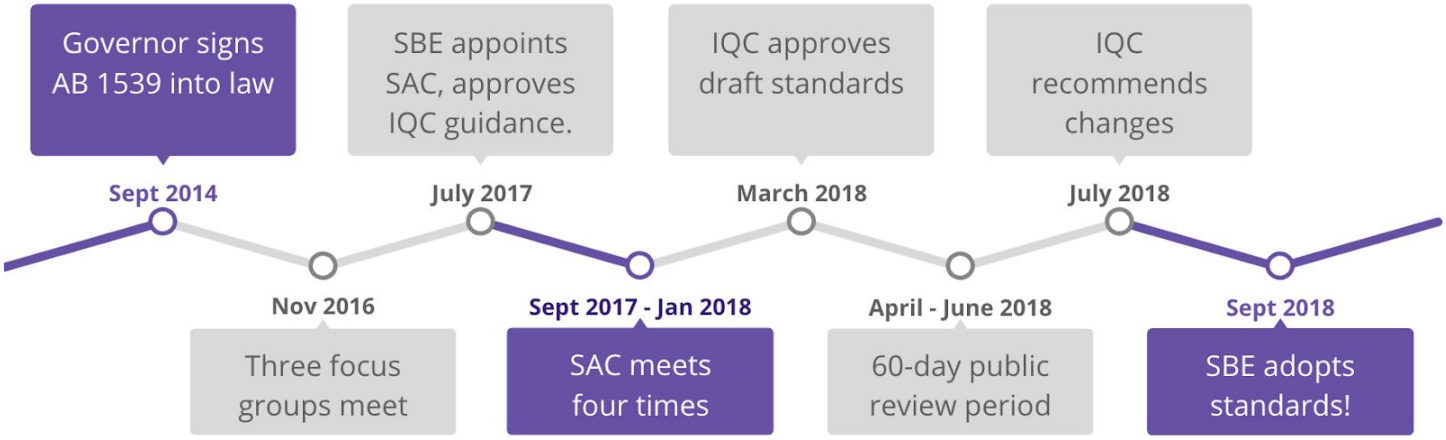
Descriptive Statement: Different algorithms can achieve the same result, though sometimes one algorithm might be more appropriate for a specific solution. Students examine different ways to solve the same task and decide which would be the better solution for the specific scenario.

Mathematics Example: Students could identify multiple algorithms for decomposing a fraction into a sum of fractions with the same denominator and record each decomposition with an equation (e.g., 2 1/8 = 1 + 1 + 1/8 = 8/8 + 8/8 + 1/8). Students could then select the most efficient algorithm (e.g., fewest number of steps). (CA CCSS for Mathematics 4.NF.3b)

English Language Arts Example: Students could compare algorithms that describe how to get ready for school and modify them for supporting different goals including having time to care for a pet, being able to talk with a friend before classes start, or taking a longer route to school to accompany a younger sibling to their school first. Students could then write an opinion piece, justifying with reasons their selected algorithm is most appropriate. (CA CCSS for ELA/Literacy W.3.1, W.4.1, W.5.1)

History/Social Studies Example: Students could use a map and create multiple algorithms to model the early land and sea routes to and from European settlements in California. They could then compare and refine their algorithms to reflect faster travel times, shorter distances, or avoid specific characteristics, such as mountains, deserts, ocean currents, and wind patterns. (HSS.4.2.2)

Development Process: The standards were developed by California educators who were appointed to the CS Standards Advisory Committee (SAC) by the State Board of Education. The committee adapted the Computer Science Teachers Association (CSTA) K-12 standards (CSteachers.org/standards) to be relevant and accessible to each and every student in California. Here is a high-level summary of the development process:



Reference: IQC = Instructional Quality Commission. SAC = Standards Advisory Committee. SBE = State Board of Education.

Themes: The standards contain significant themes, as referenced in the *K-12 CS Framework*:

- **Equity:** Issues of equity, inclusion, and diversity are addressed in concepts and practices, the standards, and in examples of ways to broaden participation in computer science education listed in the appendix.
- **Powerful ideas:** The concepts and practices evoke authentic, powerful ideas that can be used to solve real-world problems and connect understanding across multiple disciplines.
- **Computational thinking (CT):** CT is the human ability to formulate problems so that their solutions can be represented as computational steps or algorithms to be executed by a computer. CT practices such as abstraction, modeling, and decomposition intersect with computer science concepts such as algorithms, automation, and data visualization.
- **Breadth of application:** Computer science is more than coding. It involves physical systems and networks; the collection, storage, and analysis of data; and the impact of computing on society. This broad view of computer science emphasizes the range of applications that computer science has in other fields.

There are currently no plans for a state framework for computer science education. Thus, the Standards Advisory Committee included many supplementary resources with the standards:

Introduction
1. Vision 2. Why Computer Science? 3. Issues of Equity 4. Problem Solving and the 4 Cs 5. What is Computer Science?

Appendices
1. Guide for Leadership 2. Guide for Flexible Implementation 3. Guide for Instructional Practices Alignment 4. Interdisciplinary Connections 5. CTE Connections 6. Connections to Postsecondary Education

Sample K-12 Progression for *Algorithms and Programming > Variables:*

K-2	**Model the way programs store data.** *Information in the real world can be represented in computer programs. Students model the digital storage of data by transforming real-world information into symbolic representations that include text, numbers, and images.* *For example, after identifying symbols on a map and explaining what they represent in the real world, students could create their own symbols and corresponding legend to represent items on a map of their classroom (HSS.K.4.3, 1.2.3, 2.2.2)* *Alternatively, students could invent symbols to represent beat and/or pitch. Students could then modify symbols within the notation and explain how the musical phrase changes. (VAPA Music K.1.1, 1.1.1, 2.1.1, 2.2.2)*
3-5	**Create programs that use variables to store and modify data.** *Variables are used to store and modify data. Students use variables in programs they create. At this level, students may need guidance in identifying when to create variables (i.e., performing the abstraction).* *For example, students could create a game to represent predators and prey in an ecosystem. They could declare a "score" variable, assign it to 0 at the start of the game, and add 1 (increment) the score each time the predator captures its prey. They could also declare a second "numberOfLives" variable, assign it to 3 at the start of the game, and subtract 1 (decrement) each time a prey is captured. They could program the game to end when "numberOfLives" equals 0. (CA NGSS: 5-LS2-1) (CA CCSS for Mathematics 5.OA.3)* *Alternatively, when students create programs to draw regular polygons, they could use variables to store the line size, line color, and/or side length. Students can extend learning by creatively combining a variety of polygons to create digital artwork, comparing and contrasting this to another work of art made by the use of different art tools and media, such as watercolor or tempera paints. (CA CCSS for Mathematics 3.G.1) (VAPA Visual Arts 3.1.4)*
6-8	**Create clearly named variables that store data, and perform operations on their contents.** *A variable is a container for data, and the name used for accessing the variable is called the identifier. Students declare, initialize, and update variables for storing different types of program data (e.g., text, integers) using names and naming conventions (e.g. camel case) that clearly convey the purpose of the variable, facilitate debugging, and improve readability.* *For example, students could program a quiz game with a score variable (e.g. quizScore) that is initially set to zero and increases by increments of one each time the user answers a quiz question correctly and decreases by increments of one each time a user answers a quiz question incorrectly, resulting in a score that is either a positive or negative integer. (CA CCSS for Mathematics 6.NS.5)* *Alternatively, students could write a program that prompts the user for their name, stores the user's response in a variable (e.g. userName), and uses this variable to greet the user by name.*
9-12	**Create more generalized computational solutions using collections instead of repeatedly using simple variables.** *Computers can automate repetitive tasks with algorithms that use collections to simplify and generalize computational problems. Students identify common features in multiple segments of code and substitute a single segment that uses collections (i.e., arrays, sets, lists) to account for the differences.* *For example, students could take a program that inputs students' scores into multiple variables and modify it to read these scores into a single array of scores.* *Alternatively, instead of writing one procedure to find averages of student scores and another to find averages of student absences, students could write a single general average procedure to support both tasks.*
9-12 Specialty	**Compare and contrast fundamental data structures and their uses.** *Data structures are designed to provide different ways of storing and manipulating data sets to optimize various aspects of storage or runtime performance. Choice of data structures is made based on expected data characteristics and expected program functions. Students = compare and contrast how basic functions (e.g., insertion, deletion, and modification) would differ for common data structures including lists, arrays, stacks, and queues.* *For example, students could draw a diagram of how different data structures change when items are added, deleted, or modified. They could explain tradeoffs in storage and efficiency issues.* *Alternatively, when presented with a description of a program and the functions it would be most likely to be running, students could list pros and cons for a specific data structure use in that scenario.*