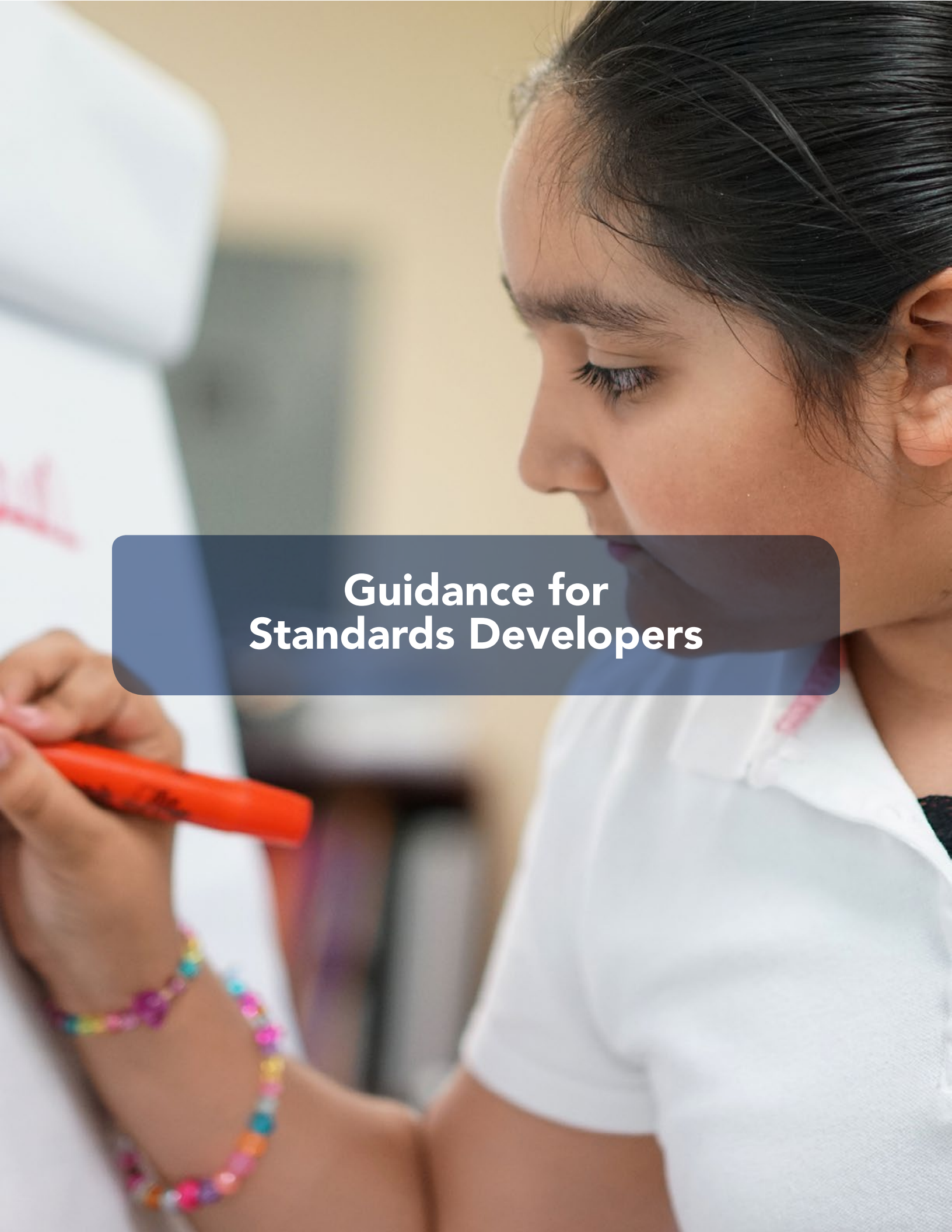
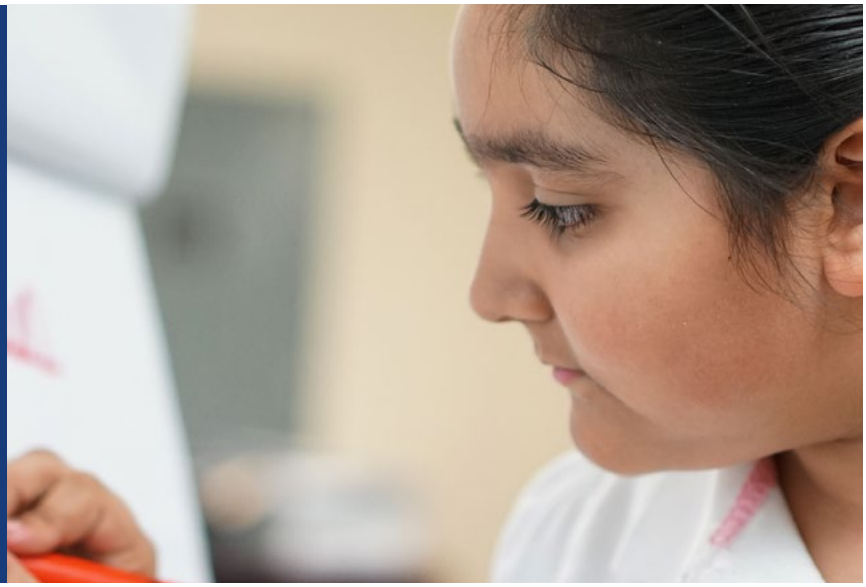


Guidance for Standards Developers

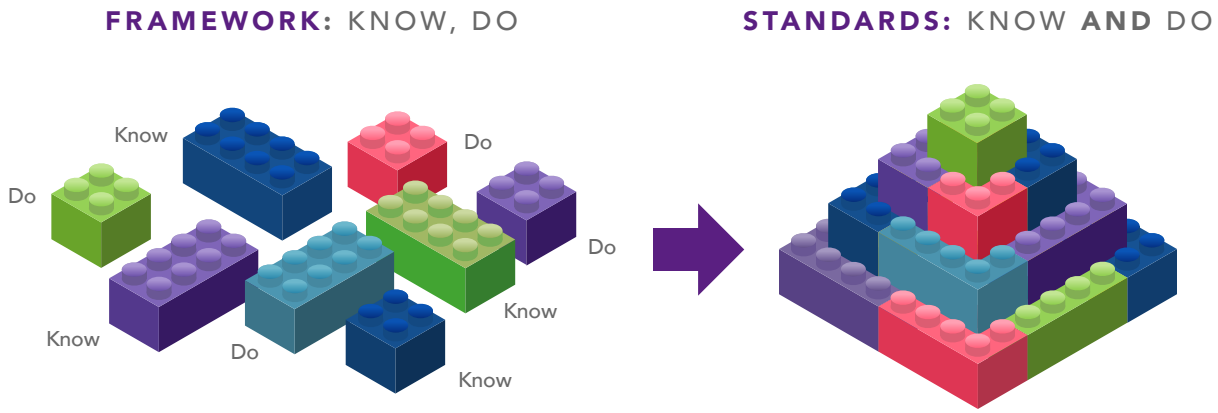
7



Guidance for Standards Developers

The K–12 Computer Science Framework is designed to serve as a foundation from which all states, districts, and organizations can develop computer science education standards for K–12 students. Standards play a vital role in achieving the vision of computer science for all students. They democratize computer science by setting learning goals for all students and the expectation that all schools will provide opportunities to achieve those goals so that all children, regardless of their age, race, gender, disability, socioeconomic level, or what school they attend, will be able to have engaging and rigorous computer science experiences. As illustrated in Figure 7.1, the framework provides the building blocks by which states can develop their own standards.

Figure 7.1: Building blocks for standards



Standards are an essential component of a larger education plan and can provide a foundation with which to align the other components, such as curriculum, instruction, professional development, and assessment, to better prepare students for success in college and the workplace. They also communicate core learning goals to policymakers, administrators, teachers, parents, and students. Computer science standards provide insight into a discipline that will be new to many teachers and offer inspirational starting points to create projects, lessons, and activities. Standards facilitate the sharing of content, such as lessons, among teachers and are a useful way to categorize that content for easy search and retrieval. Consistent standards promote alignment and connections among different districts within a state so that if a student moves to a different school, he or she will not end up with different expectations.

The purpose of this guide is to provide information and recommendations for the development of K–12 computer science education standards

- at the beginning to set the criteria and prepare standards writers,
- during the writing process with examples and exercises, and
- afterward to help evaluate the outcome.

This guide was developed in partnership with the nonprofit education organization Achieve based on recommendations for standards developers from the National Research Council (NRC, 2012). It also uses criteria and procedures Achieve has established and refined based on aspects of quality academic content standards.

These categories are described in Table 7.1.

Table 7.1: Guidance for Standards Developers summary

CRITERIA	SUMMARY
Rigor: What is the intellectual demand of the standards?	Rigor is the quintessential hallmark of exemplary standards. It is the measure of how closely a set of standards represents the content and cognitive demand necessary for students to succeed in credit-bearing college courses without remediation and in entry-level, high-quality, high-growth jobs. We recommend that standards writers establish and articulate the appropriate level of rigor in computer science to prepare all students for success in college and careers.
Focus/Manageability: Have choices been made about what is most important for students to learn and what is a manageable amount of content?	High-quality standards establish priorities about the concepts and skills that should be acquired by graduation from high school. Choices should be based on the knowledge and skills essential for students to succeed in postsecondary education and the world of work. A sharpened focus also helps ensure that the cumulative knowledge and skills students are expected to learn is manageable. We recommend grade-level standards that clearly communicate student expectations at each stage. In the case of grade-banded standards, we recommend that guidance be provided for users in creating their own grade-level standards or mapping standards to specific courses.

Table continues on next page

Table continued from previous page

Specificity: Are the standards specific enough to convey the level of performance expected of students?	High-quality standards are precise and provide sufficient detail to convey the level of performance expected without being overly prescriptive. Standards that maintain a relatively consistent level of precision ("grain size") are easier to understand and use. Those that are overly broad or vague leave too much open to interpretation, increasing the likelihood that students will be held to different levels of performance, while standards that are too prescriptive encourage a checklist approach to teaching and learning that undermines students' opportunities to demonstrate their understanding in equitable ways. We recommend that standards developers write standards that are neither too broad nor too specific and that the grain size is consistent across the standards.
Equity/Diversity: Were the standards written for all students by a diverse set of writers and reviewers? Are students able to demonstrate performance in multiple ways?	Standards, just like other aspects of education infrastructure, play a role in creating an equitable environment for all students. We recommend that diversity and equity be attended to not only in the makeup of the groups writing, advising, and reviewing the standards but also in the standards content by designing standards that can be engaged in by ALL students and are flexible enough to allow them to demonstrate proficiency in multiple ways.
Clarity/Accessibility: Are the standards clearly written and presented in an error-free, legible, easy-to-use format that is accessible to the general public?	Clarity requires more than just plain and jargon-free prose that is free of errors. Standards also must be communicated in language that can gain widespread acceptance not only by postsecondary faculty but also by employers, teachers, parents, school boards, legislators, and others who have a stake in schooling. A straightforward, functional format facilitates user access. We recommend that standards writers consider the knowledge level of users of the standards by clarifying terms and providing examples.
Coherence/Progression: Do the standards convey a unified vision of the discipline, do they establish connections among the major areas of study, and do they show a meaningful progression of content across the grades?	The way in which standards are categorized and broken out into supporting strands should reflect a coherent structure of the discipline and/or reveal significant relationships among the strands and how the study of one complements the study of another. If standards suggest a progression, that progression should be meaningful and appropriate across the grades or grade spans. We recommend that standards writers clearly communicate progressions of content and practices in the standards.
Measurability: Is each standard measurable, observable, or verifiable in some way?	In general, standards should focus on the results, rather than the processes of teaching and learning. Standards should make use of performance verbs that call for students to demonstrate knowledge and skills and should avoid using those that refer to learning activities, such as examine, investigate, and explore, or to cognitive processes that are hard to verify, such as appreciate. We recommend ensuring that each standard is measurable.
Integration of Practices and Concepts: Does each standard reflect at least one practice and one concept?	To ensure that instruction reflects both knowing and doing computer science, the core concepts of computer science should be taught alongside the practices by fully integrating them at the standards level. We recommend that standards integrate the computer science practices with the concept statements.
Connections to Other Disciplines: Are there explicit ways in which computer science is shown to be relevant in other subjects?	There are many possible areas of overlap between computer science and subject areas such as math, science, and engineering as well as humanities, including languages, social studies, art, and music. Making intentional connections between computer science standards and academic standards in other disciplines will promote a more coherent education experience. We recommend that computer science standards be written to align with and connect to (possibly via clarifying examples) state math and science standards, as well as standards from other disciplines.

Recommendations

Recommendation 1: Rigor. We recommend that standards establish and articulate the appropriate level of rigor in computer science to prepare all students for success in college and careers.

High-quality standards create foundational expectations for all students, rather than just those interested in advanced study, and prepare students for a variety of postsecondary experiences.

Standards aim to prepare students for the demands of the world they will encounter after graduation. That preparation is even more difficult when the job market changes rapidly as the influence of technology in the workforce grows steadily. It is therefore critical that standards describe rigorous expectations in computer science for all students. In addition, some students will want to specialize in computer science fields and require an even higher level of intellectual demand than is necessary for all students.

To facilitate appropriate use of the standards, differentiating between technical career standards for advanced courses and core academic standards for all students is crucial. The former may be equivalent to the expectations for specialized computer science courses—in particular, career and technical education pathways. In contrast, standards for all students describe expectations that will be important for every student to meet to help ensure their future success in any chosen field.

For example, the different standards in Figure 7.2 are based on the same practice and concept in the 9–12 grade band of the K–12 Computer Science Framework, and they compare a standard for an advanced course with a standard for all students.

Figure 7.2: Differentiating rigor for all students

Practice: Testing and Refining Computational Artifacts	Concept: Systematic analysis is critical for identifying the effects of lingering bugs. (9–12.Algorithms and Programming.Program Development)

Example 1: Test and refine software components by using unit tests to identify lingering bugs during an agile programming development cycle.

Example 2: Test and refine a program using a systematic debugging process as part of a larger iterative development process.

The first standard would be more appropriate for high school students in a specialized career and technical education course, as it calls for a product (software components) and methodologies (unit tests and agile development) that are specific to the software industry. The second standard sets a goal for all students that reflects a more general product (any computer program) yet still maintains rigor through the expectation of a systematic and iterative process.

Standards meant for all students should have sufficient rigor to help prepare students to enter and succeed in entry-level postsecondary courses that require skills such as critical thinking, problem solving, and computational literacy. Rigor applies equally to practices and concepts.

The K–12 Computer Science Framework was written to describe a vision of computer science education for all students, so most standards based on the framework could be written at a level of rigor intended for all students, rather than for students in advanced courses. Care should be taken to align the standards with grade-appropriate student abilities. It is possible that feedback or current system constraints could influence standards writers to try to limit the rigor of the standards, particularly at the elementary grade levels. However, research into students' use of sequence and iteration and practice of other aspects of computational thinking indicates that students can learn computer science at young ages (Flannery et al., 2013) when they have the support and opportunities to do so. Standards writers should be careful to keep rigor at a high enough level for younger students to ensure that all students have access to high-quality computer science education. The concept statements in the K–2 and 3–5 grade bands of the framework have been reviewed by early childhood computer science education experts and provide a blueprint for the appropriate expectations for elementary-age students. See Figure 7.3 for criteria to determine if a standard has the right amount of rigor.

Computer Science Applies to Many College Majors and Careers

Business:
Business professionals can apply processes learned in computer science to expand a business and optimize for efficiency.

Music:
Musicians can design sounds, effects, and filters. They can create a system to control music using gestures to manipulate sounds and visuals for a live show.

Biology:
Researchers can analyze a database of genetic sequences for genes similar to a known cancer gene.

Sports:
Coaches can create algorithms to analyze the performance of athletes as a training tool or develop strategies using real-time data on the field.

Figure 7.3: Determining the right amount of rigor for a standard

A standard should meet all three of these criteria:

- Does the standard require an appropriate level of conceptual understanding?
- Does it require application of that concept?
- Does it require engagement with a practice?

To help ensure that standards set expectations that prepare students for success in entry-level postsecondary courses and careers, feedback from employers and faculty members, including from two-year institutions, is crucial. The involvement of reviewers with a perspective on student preparation for postsecondary courses and careers will provide valuable information about the rigor necessary in the standards.

Recommendation 2: Focus/Manageability. We recommend that standards be limited in number, focus on the content and practices described in the framework, and be written for individual grade levels or courses.

High-quality standards prioritize the concepts and skills that should be acquired by students. A sharpened focus helps ensure that the knowledge and skills students are expected to learn are important and manageable in any given grade or course.

A clear focus within standards helps teachers see and prioritize learning experiences for students. Therefore the framework was developed to describe a core set of concepts and practices, which were selected using criteria developed by the writing team and vetted by the computer science community during review periods. Standards based on the framework should focus on the set of concepts and practices described here, rather than incorporating additional topics that could be included in advanced computer science courses.[1] See Table 7.2 for examples of important topics that are essential or not essential for all students to learn.

1 Additional topics would be appropriate for standards for advanced courses, if they are clearly designated as such and not as standards for all students (see Recommendation 1).

Table 7.2: Examples of essential and non-essential topics

IMPORTANT AND ESSENTIAL FOR ALL STUDENTS	IMPORTANT BUT NOT ESSENTIAL FOR ALL STUDENTS
• Troubleshooting strategies	• Operating systems
• Searching and sorting	• Algorithmic efficiency
• Digital data representations	• Relational databases
• Basic online security measures	• Cryptography methods

This focus will help ensure that the limited time available for computer science education throughout K–12 is concentrated on those areas that are priorities for all students. Additional standards could be added for elective computer science courses, but those should be noted as elective and not for all students. Figure 7.4 provides an example of a standard appropriately focused on the concept.

Figure 7.4: Focusing on the concept

Practice: Recognizing and Defining Computational Problems
Concept: Different software tools used to access data may store the data differently. The type of data being stored and the level of detail represented by that data affect the storage requirements. (3–5.Data and Analysis.Storage)

Standard that focuses on the concept: Evaluate the appropriateness of different ways to store data based on the type of data and the level of detail.

Standard that includes extraneous concepts: Evaluate the appropriateness of binary, octal, and hexadecimal representations of data and convert between bits and bytes.

Another aspect of appropriate focus is that standards are developed for either specific grade levels or courses. Although the framework's statements are written for grade bands (i.e., K–2, 3–5, 6–8, 9–12) and more accurately, grade-band endpoints, standards developed from the framework should be written for individual grade levels. For example, the framework's expectations by the end of 5th grade (Grades 3–5) may inform standards in all three grade levels—Grades 3, 4, and 5—or in Grade 5 only. If grade level standards are not possible, guidance should be provided about how users of the standards can create their own grade level or course-specific student expectations. Narrowing the focus of student goals at each grade level or course—either by standards writers or by district and state administrators—will enable alignment across the education system and help ensure that teachers have the support they need to focus on particular standards during a course or grade level.

Recommendation 3: Specificity. We recommend that standards writers attend to the specificity of the standards to ensure that they are neither too broad nor too specific and that the grain size, when possible, is consistent across the standards.

High-quality standards are precise and provide sufficient detail to convey the level of performance expected without being overly prescriptive. Those that maintain a relatively consistent level of precision tend to have consistent interpretation and use. Conversely, those that are overly broad or vague leave too much open to interpretation and are implemented inconsistently, and overly specific standards reduce students' opportunities to demonstrate their understanding in flexible ways.

Writing standards to a useful level of specificity requires a balance between being too vague and too specific (see Figure 7.5). A consistent and appropriate level of specificity will help ensure that teachers have the understanding and support they need to help students reach the standards. When standards are too broad, a teacher must interpret the intent of the standards—to decide what types of connections are to be understood and what depth of complexities of problems are to be solved. Useful specificity can often be added with boundary statements, which specify what content is not expected, clarifying the scope of material to be taught. For example, students by the end of eighth grade should know that network protocols exist to allow different computers to communicate with one another but not the structure of messages sent using a specific protocol, such as HTTP (Hypertext Transfer Protocol).

Figure 7.5: A spectrum of specificity in standards

	Standard	Comments
Too vague	Use conditionals in a program.	This standard lacks context and is too vague to be assessed.
Balanced	Design an algorithm that efficiently uses conditional statements to represent multiple branches of execution.	This standard specifies the type of product and a level of rigor yet allows for multiple contexts in which to demonstrate performance.
Too specific	Create an app to help friends decide between watching a comedy, action, or science fiction movie by using three if-statements.	The context for this standard is too specific and does not allow for a range of demonstrations of performance.

Consistency in the level of specificity across the standards is also important (see Figure 7.6). In practice, standards within the same document may be interpreted to have equal levels of specificity and may thus be allotted equal amounts of instructional time. It is more difficult for educators, curriculum designers, and assessment developers to use standards that vary in scope across grade levels.

Figure 7.6: Calibrating specificity across standards writers

Create a set of three to five standards that vary in specificity and have different standards writers (as small groups or individuals) put them in order and compare. Discuss the differences and characteristics of each standard, then select the one or two examples of specificity that the group should be aiming for when writing standards.

Recommendation 4: Equity/Diversity. We recommend that diversity and equity be attended to by developing standards that allow for engagement by ALL students and allow for flexibility in how students may demonstrate proficiency. The makeup of the groups of stakeholders writing and reviewing the standards should be diverse.

The framework is based on the belief that all students, regardless of race, gender, socioeconomic class, or disability, when given appropriate support, can learn all of the concepts and practices described in the framework.

Equitable standards create expectations for students with a variety of college and career interests, allow for flexible demonstrations of performance, do not assume out-of-school preparation, and are written by stakeholders with diverse perspectives.

Standards that are created for all students focus on the core aspects of computer science that are applicable to a wide range of college and career choices, rather than extraneous content with narrow application. The concepts and practices of the K–12 Computer Science Framework represent literacy in computer science for all students, not just students interested in majoring in the field or pursuing technical careers.

If computer science education is expected of all students, it must also be equitable and allow students to demonstrate their knowledge and skill in multiple ways. When a standard is particularly prescriptive, such as when it resembles the scope ("grain size") of an assessment item, it prescribes a particular way that students should demonstrate their understanding, creating the potential for an inequitable classroom environment. Equitable standards are not biased for or against students from a particular background. This includes making standards accessible to students with special needs or English language learners.

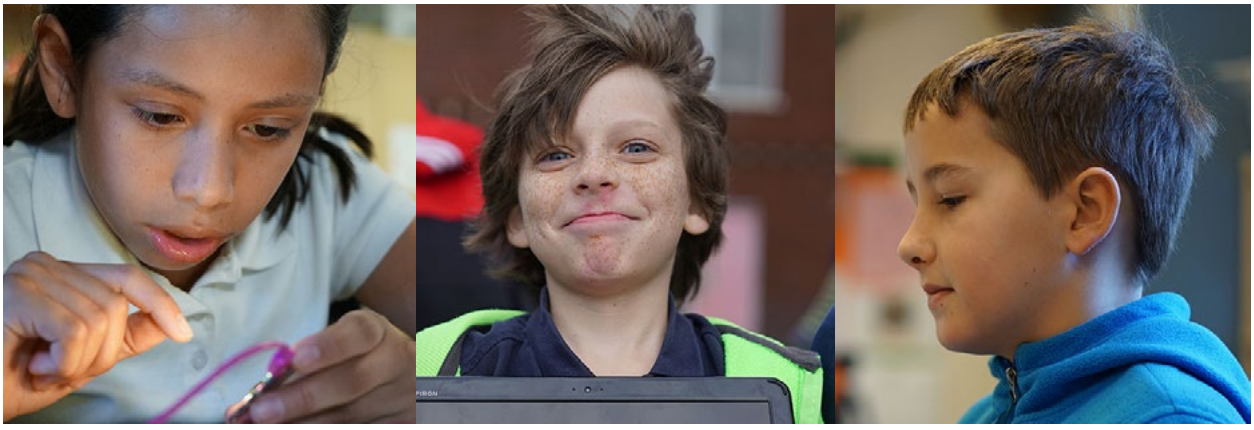
Equitable standards do not presuppose content knowledge, and therefore a level of preparation, in computer science but instead include all key stages in a learning progression. Incomplete learning progressions require out-of-school opportunities to fill in gaps in knowledge, putting students without these experiences at a disadvantage.

Developing equitable standards requires diverse stakeholders. The writers and reviewers involved in developing the standards should include diverse representation from two- and four-year institutions; the research community; industry; and most important, K–12 education, including expertise in early childhood, English language learners, and students with disabilities. This diversity will help ensure that different perspectives and areas of expertise are involved in each standard's development decision and that writers and reviewers can review each statement and example for possible bias. For example, creating standards that require specific equipment or software that is not readily accessible will disadvantage certain groups, such as rural or poor communities.

Recommendation 5: Clarity/Accessibility. We recommend that standards writers clarify standards for the average user of the standards, including defining terms and providing examples.

High-quality standards are clearly written and presented in an error-free, legible, easy-to-use format that is accessible to both the targeted instructors and the general public.

Writing clear and accessible standards is challenging. As content experts, writers may tend to drift into technical language. Additionally, computer scientists may use terms in different ways than many of the users of the standards. Precision in meaning is important but so is an awareness of the audience that will be reading and implementing the standards. In all cases, standards writers must attend to the technical understanding of the user as well as the actual content of the standard.



Computer science standards writers should consider the potential technical understanding of the average user, given the current scenarios in which computer science is taught. Rather than decreasing rigor, writers should consider how to frame standards language so that it is accessible to educators who are teaching computer science outside of their primary area of certification and may not have a computer science background. In most elementary schools, teachers are generalists, with no special training in computer science. Policymakers and community members also need to understand the educational priorities communicated by the standards. It is therefore critical for computer science standards to be accessible to many different audiences.

Precise use of language is very helpful in creating a common understanding of student outcomes among varied users, such as educators, curriculum developers, and assessment designers. Clarifications could come via boundary statements that describe the limits of standards; parenthetical notes within the standards themselves; or separate, nonassessable statements that accompany the standards. This is particularly true when words like abstraction, parallelization, and even algorithms may be used differently in different disciplines. Technical terms should be defined and, as often as possible, plain language restatements added so that the readers, particularly teachers, will be able to understand and apply the standards consistently for both curriculum and assessment. Explanations, simpler language, and/or detailed descriptions would be helpful to ensure consistent application of the standards (see Figure 7.7).

Figure 7.7: Example of technical terms versus simple language in standards

Standard 1: Use an API by calling a procedure and supplying arguments with appropriate data types to efficiently employ high-level functionality.

Standard 2: Select and use a procedure from a library of procedures (API) and provide appropriate input as arguments to replace repetitive code.

The second standard retains "API" (application programming interface), adds more accessible wording such as "library of procedures," and prefaces the specific programming term "argument" with the more general "appropriate input." The second standard continues to use the terms "procedure" and "arguments" as these are necessary terms that provide clarity. Accessible standards use key terminology to provide clarity and avoid extraneous terms and technical jargon.

Examples are very useful to communicate the intent of the standards to users. However, when examples are used, we recommend that multiple examples always be present. The use of single examples can often seem to be a limiting factor or inadvertent prescription of curriculum (Achieve, 2010).

Recommendation 6: Coherence/Progression. We recommend that standards be organized as progressions that support student learning of content and practices over multiple grades.

Coherence refers to how well a set of standards conveys a unified vision of the discipline, establishing connections among the major areas of study and showing a meaningful progression of content across grade levels and grade spans.

Research on student learning indicates that students need explicit help to connect new ideas to ideas that have been learned previously (Marzano, 2004). To support teachers as they help students make these connections, standards should describe developmentally appropriate levels of a learning progression, and the learning progressions embedded in standards must be made apparent to users. This is true for both the content and the practices, as students' facilities with each of the practices change and deepen over time when they are provided adequate instructional opportunities. Separate displays that show the progression of each dimension through K–12 have been very useful to educators in implementing standards.

The framework writers were careful to describe coherent progressions of content and skills across grade bands. Standards based on the framework, however, may be written for individual grade levels. In that case, care should be taken to ensure that the progression from grade level to grade level is coherent and research-based as much as possible and that student knowledge and practice will build on the foundation of content and skills learned previously. The progressions in the framework revolve around a central subconcept in each core concept area and reflect developmentally appropriate milestones that grow in sophistication from kindergarten to Grade 12 (see Figure 7.8).

Figure 7.8: Example learning progression

Computing Systems.Hardware and Software

By the end of Grade 2: A computing system is composed of hardware and software. Hardware consists of physical components, while software provides instructions for the system. These instructions are represented in a form that a computer can understand.

By the end of Grade 5: Hardware and software work together as a system to accomplish tasks, such as sending, receiving, processing, and storing units of information as bits. Bits serve as the basic unit of data in computing systems and can represent a variety of information.

By the end of Grade 8: Hardware and software determine a computing system's capability to store and process information. The design or selection of a computing system involves multiple considerations and potential tradeoffs, such as functionality, cost, size, speed, accessibility, and aesthetics.

By the end of Grade 12: Levels of interaction exist between the hardware, software, and user of a computing system. The most common levels of software that a user interacts with include system software and applications. System software controls the flow of information between hardware components used for input, output, storage, and processing.

Recommendation 7: Measurability. We recommend ensuring that each standard is objective and measurable.

Standards should focus on the results, rather than the processes of teaching and learning. They should make use of performance verbs that call for students to demonstrate knowledge and skills, with each standard being measurable, observable, or verifiable in some way.

To be effective for teaching and learning, standards must be observable and measurable. What the standard intends a student to understand or be able to do should be clear. Accordingly, teachers need to be able to clearly determine if the expectation has been met to know whether students need further help with these concepts.

However, standards do not necessarily need to be written such that they could be tested on a large-scale summative assessment. They simply need to be observable by some measure, including by a classroom teacher. Careful selection of the verbs used in each standard, along with specificity of content, will help ensure that the standard is observable and measurable (see Table 7.3).

Table 7.3: Examples of verbs that assist with measurability

VERBS THAT REFER TO OBSERVABLE PERFORMANCE OR RESULTS	VERBS THAT REFER TO LEARNING ACTIVITIES	VERBS THAT REFER TO COGNITIVE PROCESSES
Create Develop Test Refine Communicate	Examine Explore Observe Discover	Know Understand Appreciate

Recommendation 8: Integration of Practices and Concepts. We recommend that standards integrate the computer science practices with the concept statements.

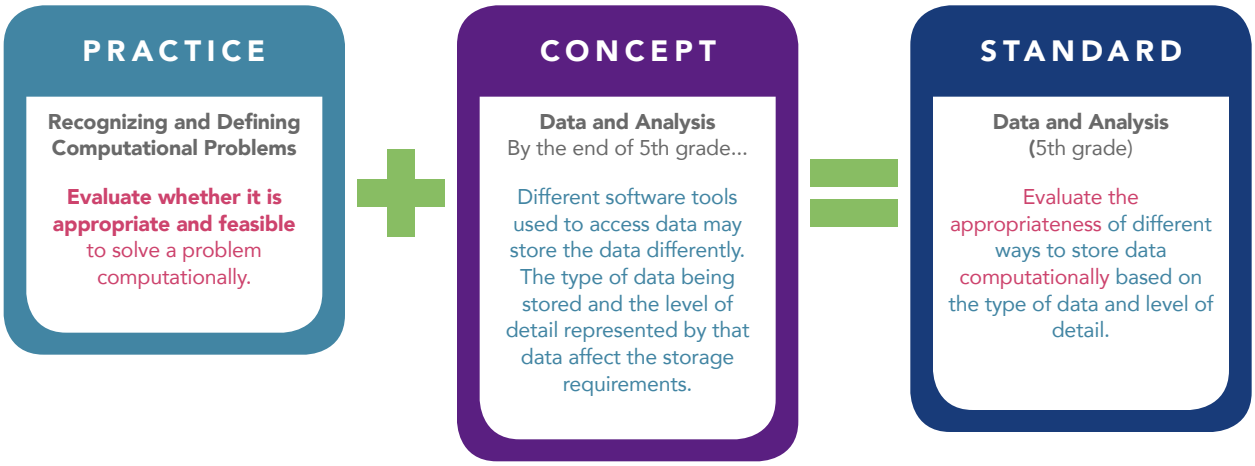
To realize the vision described in this framework and to ensure that all students can become proficient users of computer science knowledge and practice, the practices and concepts should be integrated in the standards, as well as in curriculum and instruction.

Previous sets of education standards in many different disciplines included separate practice and content standards. However, because teachers and curriculum designers were more familiar and comfortable with the content standards, the practice standards were very rarely implemented. They were separate, so they were typically left out or "covered" in the first week of school and then forgotten, or they were used irregularly. One efficient way to help ensure that practices are included throughout instruction is to integrate them completely with the content standards.

More important, part of the vision for computer science education is that students will become proficient at using and applying knowledge—not just memorizing it. If application and deep understanding is indeed the goal, education standards should be written to reflect that goal. By combining a practice with each concept statement to create a standard, the resulting standards more closely describe the behavior, abilities, and deep knowledge we want students to have.

Figure 7.9 below shows an example of how to integrate a computer science practice with a concept statement from the framework.

Figure 7.9: Example of integrating a practice and concept to create a standard

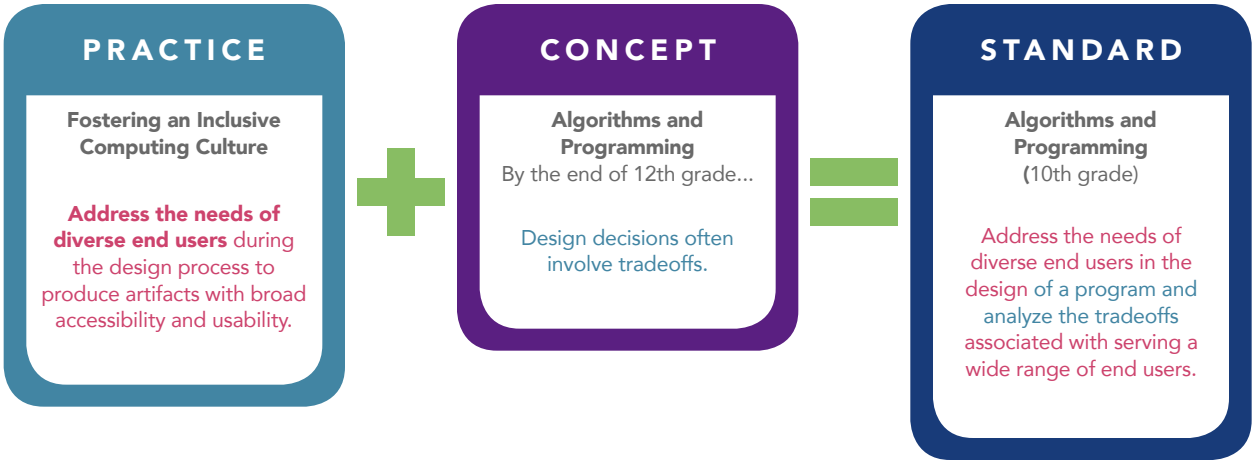


The following steps were taken to create this example.

1. The writer chose a specific practice statement within Practice 3: *Recognizing and Defining Computational Problems*.
2. The writer selected a portion of the *Data and Analysis* concept statement as a context for applying the practice.
3. The practice and concept were combined to create an observable performance expectation that calls for the application of the practice within the context of the concept. The bolded verb stem in the practice statement helped to focus the action in the standard.

Figure 7.10 provides another example of integrating a practice and concept to create a standard. By using the checklist provided in Recommendation 1: Rigor, we see that this standard requires an appropriate level of content understanding, as reflected in the concept portion [highlighted in blue] and engagement with a practice [highlighted in magenta], which facilitates the application of the content [the standard as a whole].

Figure 7.10: Second example of integrating a practice and concept to create a standard



It is not expected, or recommended, that each concept statement be combined with statements from all of the practices to form multiple standards. For example, although there are a total of 68 concept statements and seven practices (each of which has multiple statements), a K–12 standard set should not expect to have 476 standards (i.e., 68 multiplied by 7). Only the practice statements that are most relevant to a concept statement should be considered. In addition, remember that integrating practices with concept statements often introduces more rigor to the student performance expectation than would be seen in the concept statement on its own because students now will have to do something with that conceptual knowledge. Care should be taken to ensure that the particular combination of practices and concepts does not introduce a higher level of rigor than is appropriate for the grade band. Figure 7.11 provides an exercise for standards developers using these considerations.

Figure 7.11: Exercise in standards creation

1. As a group, pick the same concept and practice and create a standard from the pairing.
2. Compare each other's proposed standard.
3. Ask:
   a. Is the rigor appropriate for the grade band?
   b. Is the performance expectation clear?
   c. Does it accurately reflect components of the concept and practice?
   d. Is this an appropriate standard for all students or just those going on to extended study?

Recommendation 9: Connections to Other Disciplines. We recommend that computer science standards be written to align with and connect to other academic standards, such as mathematics and science.

There are many possible ways computer science can connect with other subjects, like math, science, and engineering, as well as humanities, such as languages, social studies, art, and music. Making intentional connections between computer science standards and academic standards in other disciplines will help teachers understand how computer science can connect with their implementation of standards in other subjects and promote more coherent education experiences for students. While related, technology/digital literacy and computer science are distinct subjects.

With limited time in the classroom, students' education should be as coherent as possible. When content in different disciplines is related or connecting, it is important to point out those connections to educators and to facilitate them through standards. When potential alignments are not recognized in standards, extra instructional time may be required to cover everything. For example, if a core math concept is required for third grade computer science standards but is not included in math standards until fifth grade, third grade teachers would need to add that concept into their computer science curriculum, or they might end up ignoring the computer science content due to an impression that it is too overwhelming.

In addition to aligning grade-level expectations, it can also be helpful to include clarifying examples that align and connect to math, science, and engineering standards (see Figure 7.12).

Figure 7.12: Example of a computer science standard that connects with a science standard

Standard: Test and refine a program using a wide range of inputs until criteria and constraints are met.

This standard connects with Next Generation Science Standard MS-ETS1-2 Engineering Design: Evaluate competing design solutions using a systematic process to determine how well they meet the criteria and constraints of the problem.

Attention should also be given to connections to subjects outside of science, technology, engineering, and math, such as language arts literacy standards for technical subjects. Since computer science is not currently required or assessed in most states, illustrating how the standards connect to and help meet existing standards in other subjects will be very useful. These connections can be made through ancillary materials like crosswalks and examples and can be used as a tool to integrate content from other subjects into computer science or embed computer science content into other subjects. This is particularly true for Grades K–8, as budget constraints may not allow for separate computer science teachers in elementary and middle schools.

In 2010, the Association for Computing Machinery's *Running on Empty* reported that "there is deep and widespread confusion within the states as to what should constitute and how to differentiate technology education, literacy and fluency; information technology education; and computer science as an academic subject" (p. 9). While it is plausible to combine digital/technology literacy standards with academic computer science standards, care should be taken so as not to confuse addressing one with addressing the other. For example, while a digital presentation can be used to communicate a team's software development process, the creation of the digital presentation, or the general use of office productivity software, is not a computer science activity. Again, *Running on Empty* reported that "consistent with efforts to improve 'technology literacy,' states are focused almost exclusively on skill-based aspects of computing (such as using a computer in other learning activities) and have few standards on the conceptual aspects of computer science that lay the foundation for innovation and deeper study in the field (for example, develop an understanding of an algorithm)" (p. 7). If combining digital literacy and computer science into one set of standards, it is important that the distinction be kept clear through separately identifiable strands.

References

Achieve. (2010). *International science benchmarking report.* http://www.achieve.org/files/InternationalScienceBenchmarkingReport.pdf

Association for Computing Machinery & Computer Science Teachers Association. (2010). *Running on empty: The failure to teach K–12 computer science in the digital age.* Retrieved from http://runningonempty.acm.org/fullreport2.pdf

Flannery, L. P., Kazakoff, E. R., Bontá, P., Silverman, B., Bers, M. U., & Resnick, M. (2013, June). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children* (pp. 1–10).

Marzano, R. J. (2004). *Building background knowledge for academic achievement: Research on what works in schools.* Alexandria, VA: Association for Supervision and Curriculum Development.

National Research Council. (2012). *A Framework for K–12 science education: Practices, crosscutting concepts, and core ideas.* Committee on a Conceptual Framework for New K–12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.