2026 CSTA PK–12

COMPUTER SCIENCE STANDARDS

Fostering Dispositions in PK–12 Computer Science

What are Dispositions?

Dispositions are the **habits of mind, attitudes, and approaches** that shape how students engage with computer science. They influence how students think, persist, collaborate, and reflect beyond what they can code or recall. In CS, dispositions guide how students navigate challenges, debug with purpose, and build confidence in problem solving.

Fostering these dispositions helps students develop not only technical skills, but also the capacity to learn independently, stay motivated, and persist through complexity.

Key Dispositions in CS Education

In Reimagining CS Pathways (CSTA et al., 2024), the CS education community identified **seven key dispositions** essential to equitable and enduring CS learning:

	**Creativity**	Generating original and meaningful computing ideas and projects.
	**Critical thinking**	Using reasoning and evidence to analyze and refine solutions.
	**Curiosity**	Asking questions and exploring beyond assigned work.
	**Persistence**	Continuing effort despite frustration or setbacks.
	**Reflectiveness**	Connecting past experiences to inform future learning.
	**Resourcefulness**	Seeking and using tools, people, and information to solve problems.
	**Sense of belonging**	Feeling included, respected, and recognized in the CS community.

These dispositions align closely with *self-regulated learning*, as students plan, monitor, and evaluate their growth as learners of computer science.

Why These Dispositions Matter

Dispositions shape how students experience computer science—not just what they learn, but how they see themselves as learners and creators. They:

- **Advance equity and inclusion:** Belonging and creativity help all students see themselves reflected in computing.
- **Deepen understanding:** Critical thinking and reflectiveness connect conceptual learning to practice.
- **Build resilience:** Persistence and resourcefulness help students navigate challenges and turn setbacks into progress.
- **Sustain motivation:** Curiosity keeps students exploring beyond assigned tasks and supports ongoing learning.

Dispositions are the foundation of lasting CS learning. They connect technical ability to personal growth, ensuring that every student can engage meaningfully with computing. Teachers foster dispositions through intentional instructional design decisions and a consistent instructional approach that encourages their development. When classrooms intentionally emphasize these core dispositions, students develop both the confidence and competence to thrive as problem solvers, innovators, and future leaders in CS.

How to Use This Resource

This resource is designed to help teachers understand, recognize, and intentionally foster the seven key dispositions in their computer science classrooms. Each disposition includes its own two-page guide with a definition, an explanation of its purpose in CS learning, observable behaviors organized by grade band, concrete examples of what you might see or hear, research-informed strategies for fostering it, and a student progression that describes growth from emerging to demonstrating.

Use these guides flexibly: focus on one disposition at a time, reference the observable behaviors to notice and name dispositions as they emerge, or draw on the strategies when planning lessons. The progressions can support reflection, goal-setting, and conversations with students about their growth. Because dispositions develop best when consistently encouraged across tasks and grade levels, look for opportunities to weave them throughout instruction rather than treating them as isolated skills.

Creativity in CS Education

Definition

Creativity is the capacity to generate novel and valuable ideas, artifacts, or solutions through imaginative thinking and combining existing knowledge in new ways—especially within domain-specific contexts like computing.
(Adapted from Sawyer, 2012; Csikszentmihalyi, 1996; Papert, 1980)

Purpose

Creativity is essential to learning and practicing computer science, enabling learners to generate ideas, explore alternatives, and make intentional design decisions. In CS education, creativity:

- **Supports authentic computing practice** by encouraging exploration, iteration, and testing multiple solutions, similar to professional contexts (Papert, 1980; Resnick, 2007).
- **Strengthens conceptual understanding** by requiring learners to apply computing concepts in novel situations, rather than reproducing solutions (Plucker et al., 2004; Sharmin, 2021).
- **Broadens how learning outcomes are demonstrated** by valuing original design choices and purposeful variation alongside functional correctness (Sawyer, 2012).
- **Enables meaningful use of computational thinking practices**, including abstraction, decomposition, and iteration, within creative design processes (Resnick, 2007).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-1	Explores visuals, sounds, or movement in digital spaces; personalizes creations through colors, characters, or simple choices.
2–3	Combines familiar elements in new ways and makes intentional design choices to express an idea or story.
3–5	Develops original features or adaptations; customizes templates or tools to achieve a specific creative goal or audience effect.
6–8	Designs original digital artifacts that serve a clear purpose, using creativity to solve problems or communicate meaning.
9–12	Innovates within constraints; adapts tools, techniques, or systems creatively to produce impact, address real-world issues, or communicate ideas effectively.

You Might Hear or See

- "Can I make mine do something different?"
- Students choose creative representations—visuals, animations, sound—to communicate an idea.
- Students express their ideas, style, humor, or personal interests through creative design choices.
- Students remix or extend examples in unexpected or original ways.

Strategies to Foster

- **Build open-endedness** into prompts (e.g., “What do you want this to do?”).
- **Introduce productive constraints** such as time limits, target audiences, required features, or specific tools to encourage inventive problem-solving.
- **Use the design thinking process** to support ideation, prototyping, testing, and revision.
- **Incorporate project showcases** or gallery walks for sharing work and peer feedback.
- **Offer choice boards or tool menus** that allow students to select how they approach a task.
- **Frame challenges with purpose-driven prompts** such as “Design something that solves a problem for someone else.”

Student Progression

Focus	Emerging	Developing	Demonstrating
Exploration and Expression	Students complete a project with given steps and little customization, showing tool use but few original decisions.	Students modify, remix, or expand a starter program with creative elements such as characters, movement, or visual style.	Designs original artifacts, making intentional creative decisions that shape story, behavior, design, and output.
Design and Iteration	Students follow a single design idea and revise only when something breaks.	Students generate multiple ideas, test variations, and refine features based on curiosity or feedback.	Students engage in purposeful revision by comparing approaches, refining through reflection, and justifying design decisions.
Cultural and Personal Relevance	Student projects reflect teacher-provided examples with minimal personal expression.	Students incorporate personal interest, style, or context into a provided project or challenge.	Students design artifacts inspired by personal identity, community, or lived experience, connecting computing and culture.

Critical Thinking in CS Education

Definition

Critical thinking is the ability to analyze problems and make reasoned decisions using evidence, logic, and computational concepts—examining how systems work, predicting outcomes, and evaluating trade-offs based on data, rules, or constraints. (Adapted from Facione, 1990; Wing, 2006; Grover & Pea, 2013)

Purpose

Critical thinking is central to how learners engage with CS, enabling students to move beyond trial-and-error by analyzing problems, anticipating outcomes, and making informed decisions about algorithms and data. In CS education, critical thinking:

- **Supports accurate problem analysis** by helping learners break down problems, identify relevant information, and recognize patterns or constraints (Wing, 2006; Grover & Pea, 2013).
- **Strengthens reasoning and decision-making** by requiring students to justify choices using evidence, logic, or observed outcomes rather than intuition alone (Facione, 1990).
- **Enables evaluation of multiple approaches** by comparing efficiency, effectiveness, or appropriateness of solutions within a given context (Wing, 2006).
- **Prepares students to examine data, systems, and tools critically**, including identifying limitations, assumptions, or potential sources of bias (CSTA et al., 2024).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-K	Sorts or sequences objects based on attributes (e.g., color, shape, order); notices when something doesn't work as expected.
1–2	Predicts what will happen in a sequence or simple system; compares options and makes choices using simple reasoning.
3–5	Identifies patterns or rules in programs, data sets, or systems; uses evidence to explain decisions; compares approaches based on observed results.
6–8	Evaluates trade-offs between solutions; traces how data or information changes through a process; uses evidence to justify design choices.
9–12	Examines how data, rules, or design choices affect how a computing system behaves, produces results, or protects information.

You Might Hear or See

- "I think this loop runs too many times—let me check why."
- Students predicting what a program, query, or system will produce before running it.
- Students comparing two data sets or solutions and identifying trade-offs between them.
- Students explaining or defending decisions using evidence, logic, or constraints.
- Students questioning whether a data source, output, or system behavior is reliable or biased.

Strategies to Foster

- **Use error analysis prompts** (e.g “What do you notice?” and “What do you think caused this?”).
- **Integrate design critiques** or peer feedback rounds that focus on reasoning and evidence rather than preference.
- **Have students map decision-making before coding** using pseudocode, flowcharts, or annotated examples.
- **Teach and model computational thinking routines**, including decomposition, pattern recognition, and abstraction.
- **Use structured instructional approaches** such as Use–Modify–Create or PRIMM to scaffold analysis before independent solution-building.

Student Progression

Focus	Emerging	Developing	Demonstrating
Analysis of Problems and Systems	Identifies key elements of a problem or system and notices when outcomes differ from expectations.	Analyzes patterns, rules, or causes within a problem or system	Evaluates solutions or systems by considering evidence, constraints, and potential impacts.
Use of Evidence and Reasoning	Make choices with limited explanation.	Explains decisions using observed results, logic, or examples.	Justifies decisions using evidence, trade-offs, or constraints.

Curiosity in CS Education

Definition

Curiosity is the inclination to ask questions, explore possibilities, and seek understanding about how computing systems, tools, or ideas work. In computer science, curiosity involves wondering what might happen and investigating how things function during interaction with technology or code.
(Adapted from Berlyne, 1954; Engel, 2015; Grover & Pea, 2013)

Purpose

Curiosity drives engagement with computer science by motivating learners to explore how systems behave and why outcomes occur. In CS education, curiosity:

- **Encourages learners to ask questions** about how programs, data, or systems work rather than accepting outputs at face value (Engel, 2015).
- **Supports exploration of alternative possibilities**, inputs, or conditions within computational tasks (Berlyne, 1954).
- **Promotes deeper engagement with computing concepts** by prompting investigation and experimentation (Grover & Pea, 2013).
- **Reflects authentic computing practice**, where curiosity often drives debugging, exploration of new tools, and innovation (CSTA et al., 2024).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-K	Asks simple “what’s that?” or “why?” questions during unplugged activities or when using computing tools; explores cause-and-effect interactions through play.
1–2	Explores computing tools beyond guided instructions; asks “how does this work?” or “what if I…?” and begins predicting outcomes.
3–5	Generates specific questions during CS activities; investigates how changes to code, inputs, or data affect outcomes.
6–8	Investigates how and why computing solutions work; explores alternative approaches or configurations out of interest.
9–12	Pursues organic questioning through independent exploration, projects, or use of new tools, libraries, or technologies beyond required tasks.

You Might Hear or See

- “What happens if I change this variable?”
- “What happens if I move this block or change the order of the code?”
- Students testing different inputs or configurations just to see what will happen.
- Students pausing progress to explore an unexpected behavior or question.
- Students prompting an AI tool in different ways to see how its responses change, or asking why it produced a particular output.
- Teachers briefly pause instruction to pursue or document organic questions.

Strategies to Foster

- **Use notice/wonder protocols** when introducing new tools, systems, or challenges.
- **Leverage problem-, project-, or inquiry-based approaches** that position student questions as the driver of investigation and learning.
- **Offer sandbox or exploration** time for students to experiment with unfamiliar platforms or features.
- **Use Parsons problems** or partially completed examples to lower barriers to experimentation.
- **Frame debugging as discovery**, treating errors as opportunities to explore system behavior.
- **Normalize inquiry** and student wondering by maintaining visible spaces such as “Questions We’re Exploring” boards.

Student Progression

Focus	Emerging	Developing	Demonstrating
Questioning	Asks simple questions about what will happen.	Poses “what if” questions to explore alternatives.	Generates questions that extend or deepen the investigation.
Exploration	Explores features or actions with guidance.	Test ideas by changing inputs or conditions.	Independently explores new possibilities or related concepts.

Persistence in CS Education

Definition

Persistence is the ability to sustain effort and continue working toward a goal when faced with challenges, errors, or setbacks. Persistence involves continuing to engage with problems despite bugs, unexpected behavior, or incomplete understanding while staying focused on progress.
(Adapted from Duckworth et al., 2007; Dweck, 2006; Grover & Pea, 2013)

Purpose

Persistence supports how learners engage with the iterative, problem-rich nature of CS, helping students work through difficulty rather than abandon tasks early. In CS education, persistence:

- **Supports progress on complex or multi-step problems** by encouraging sustained engagement through errors and challenges (Grover & Pea, 2013).
- **Enables learners to view mistakes as part of the problem-solving process** rather than as signals to stop (Dweck, 2006).
- **Reflects authentic computing practice**, where developers routinely encounter bugs, unexpected outputs, and incomplete solutions that require effort over time (CSTA et al., 2024).
- **Helps students build stamina for extended tasks**, projects, or investigations common in CS learning environments (Pinto et al., 2021).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-K	Tries again when a task does not work on the first attempt; continues working with adult or peer support.
1–2	Repeats tasks with small variations after mistakes; tolerates brief frustration and continues attempting solutions.
3–5	Attempts multiple approaches to fix errors; stays focused during challenging tasks and revises work rather than abandoning it.
6–8	Uses iterative approaches (e.g., test–revise–test again); manages frustration productively while continuing to work.
9–12	Sustains effort through complex or multi-phase tasks; documents strategies used to overcome obstacles or setbacks.

You Might Hear or See

- “Let me try one more way to fix this.”
- Students debugging or testing changes several times before stopping.
- Students returning to a task after an error or unexpected result.
- Peer encouragement during challenging moments.
- Students breaking work into smaller steps to keep moving forward.

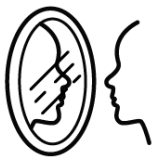
Strategies to Foster

- **Normalize errors and revision** as expected parts of computer science work, including sharing real examples of iteration and failure in technology development.
- **Design tasks that foster productive struggle**, where challenges are meaningful yet achievable.
- **Use process-focused feedback or rubrics** that emphasize effort, revision, and progress.
- **Establish visible "stuck strategies"** (e.g., code tracing, rubber duck debugging) so students know how to keep going when they encounter difficulty.
- **Use self-monitoring strategies** (e.g., Red / Yellow / Green) to help students recognize struggle levels and stay engaged appropriately.

Student Progression

Focus	Emerging	Developing	Demonstrating
Response to Challenge	Stops or seeks help immediately when encountering difficulty.	Continues working after difficulty by trying again or making small changes.	Sustains effort through repeated setbacks by adjusting approaches and continuing toward a goal.
Duration of Effort	Engages briefly with challenging tasks.	Maintains effort for moderate periods despite errors.	Maintains focus and effort over extended tasks or projects.

Reflectiveness in CS Education



Definition

Reflectiveness is the ability to think intentionally about one's learning process, decisions, and outcomes to improve future actions. In computer science, reflectiveness involves examining strategies and reasoning about how choices in code, data, or design affect outcomes over time.
(Adapted from Schön, 1983; Zimmerman, 2002; Grover & Pea, 2013)

Purpose

Reflectiveness helps learners make sense of their CS experiences by examining what happened and using it to inform next steps. In computer science education, reflectiveness:

- **Helps learners connect actions to outcomes** by examining how specific choices influenced results (Schön, 1983).
- **Supports improvement over time** by encouraging students to consider what strategies were effective and which were not (Zimmerman, 2002).
- **Strengthens understanding of computing concepts** by prompting learners to articulate reasoning and identify patterns in their work (Grover & Pea, 2013).
- **Mirrors professional computing practice**, where developers regularly review processes, test outcomes, and revise approaches based on evidence (CSTA et al., 2024).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-K	Shares reactions to a CS activity (e.g., “fun,” “hard”) and responds to simple reflection questions about what happened or what was learned.
1–2	Identifies what worked well or what was challenging during a task and suggests simple changes for next time.
3–5	Describes steps taken and outcomes achieved; reflects on what could be improved and how design choices affect results or users.
6–8	Reflects on individual and group contributions; revises approaches based on feedback or observed outcomes.
9–12	Analyzes patterns in learning or performance across projects; uses reflection to set goals or guide future strategies.

You Might Hear or See

- “Next time, I’ll write more comments in my code.”
- “Next time, I’ll organize my data differently.”
- Students identifying what they learned from a task or group challenge.
- Students referencing feedback, results, or prior attempts when revising work.
- Quiet moments of written or verbal reflection during or after activities.

Strategies to Foster

- **End lessons with brief reflective exit prompts** (e.g., "One thing I learned...", "One thing I would change next time...").
- **Use learning journals or debug diaries** to document decisions, outcomes, and revisions.
- **Invite peer feedback** using structured protocols (e.g., "I noticed...", "I wonder...").
- **Provide sentence stems** that help students articulate reflection (e.g., "I was surprised by...", "I struggled with...", "I learned that...").
- **Build in intentional pauses** for students to review progress before continuing work.

Student Progression

Focus	Emerging	Developing	Demonstrating
Reflection	Reflects on what happened during a task or activity.	Reflects on why outcomes occurred, referencing strategies or decisions.	Reflects on patterns across attempts and uses insights to plan future approaches.
Use of Reflection	Describes experiences or outcomes after completion.	Uses reflection to make specific revisions or adjustments.	Uses reflection proactively to guide planning, goal-setting, or strategy selection.

Resourcefulness in CS Education

Definition

Resourcefulness is the ability to identify, select, and use available tools, information, and support effectively to solve problems or complete tasks. Resourcefulness involves knowing when and how to leverage documentation, examples, peers, and technical resources to move learning forward.
(Adapted from Zimmerman, 2002; OECD, 2018; CSTA et al., 2024)

Purpose

Resourcefulness helps learners navigate complexity in CS, where unfamiliar tools push students to find and use resources independently. In CS education, resourcefulness:

- **Enables learners to work productively** in open-ended or unfamiliar contexts by identifying appropriate tools or sources of information (Zimmerman, 2002).
- **Strengthens problem-solving** by promoting strategic help-seeking and effective use of documentation, examples, and support systems (OECD, 2018).
- **Reflects authentic computing practice**, where programmers routinely consult references, collaborate with others, and adapt existing resources to meet new needs (CSTA et al., 2024).
- **Supports efficient learning** by helping students evaluate which resources are useful and how to apply them appropriately to a task (Kennett & Keefer, 2006).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-K	Uses classroom supports such as visual cues, routines, or modeled examples; imitates peers to complete tasks
1–2	Chooses between provided tools or options; seeks familiar supports such as peers or guides when needed.
3–5	Independently uses available resources (e.g., help menus, charts, peers); begins distinguishing useful from unhelpful information.
6–8	Selects and uses peers, teachers, and technical resources strategically; monitors whether a resource is helping and adjusts as needed.
9–12	Integrates multiple tools or sources of information; adapts or combines resources to meet goals and supports peers in using them effectively.

You Might Hear or See

- "I looked up a tutorial to see how this works."
- Students using help menus, hints, documentation, or example code without prompting.
- Students asking peers targeted questions after attempting to use available resources.
- Creative use of limited tools or materials to achieve a desired outcome.
- Students switching strategies or resources when a single approach is ineffective.

Strategies to Foster

- **Model how to locate and use reference materials**, help menus, and documentation.
- **Teach independent search skills**, including how to identify safe, relevant, and reliable information.
- **Encourage structured help-seeking strategies** (e.g., "Try three resources before asking the teacher").
- **Design tasks that enable student choice** among tools or resources rather than providing a single path.
- **Acknowledge and celebrate** effective or creative use of resources during project sharing or reflection.

Student Progression

Focus	Emerging	Developing	Demonstrating
Use of Resources	Uses provided supports when prompted.	Selects and applies appropriate resources independently.	Strategically integrates multiple resources and adapts them to new contexts.
Help Seeking	Asks for help without first using available resources.	Uses available supports before seeking assistance.	Seeks targeted help after evaluating and attempting multiple resources.

Sense of Belonging

Definition

Sense of Belonging is a student's perception of being accepted, valued, and included as a legitimate member of the CS learning community. It reflects whether students see themselves—and are recognized by others—as participants whose ideas, identities, and contributions matter in computing space.
(Adapted from Anant, 1967; Veilleux et al., 2013; Krause-Levy et al., 2021)

Purpose

Sense of belonging is foundational to sustained participation in CS, influencing whether students engage fully, contribute ideas, and continue in CS over time. In CS education, sense of belonging:

- **Supports equitable participation** by ensuring students feel welcomed and included regardless of background, prior experience, or identity (Krause-Levy et al., 2021).
- **Influences students' perceptions of their place in computing** by shaping whether they view CS as "for people like me" (Veilleux et al., 2013).
- **Strengthens engagement in collaborative computing practices** by fostering trust, respect, and shared responsibility within learning communities (Hansen et al., 2023).
- **Contributes to retention in CS pathways** by reducing feelings of isolation and marginalization, particularly for students historically underrepresented in computing (CSTA et al., 2024).

Observable Behaviors

Grade Band	Observable Behaviors
PreK-1	Participates in shared CS activities and is comfortable working alongside peers.
2–3	Engages regularly in collaborative CS tasks and shares ideas or responses with peers.
4–5	Contributes ideas during group work and acknowledges the ideas of others.
6–8	Recognizes their role in team success and participates in shared problem-solving.
9–12	Actively supports inclusive collaboration and values diverse perspectives in computing spaces.

You Might Hear or See

- Students offering help or encouragement to one another during CS activities.
- Students referencing shared goals, roles, or team agreements.
- Students acknowledging or building on one another's ideas.
- Students using inclusive language naturally during collaboration (e.g., "we," "our project").

Strategies to Foster

- **Use collaborative protocols** (e.g., pair programming, structured group roles, think-pair-share) to ensure all students participate.
- **Establish and revisit shared norms** for respectful communication and teamwork.
- **Highlight and celebrate diverse contributions** through student showcases, classroom visuals, or examples from multiple perspectives.
- **Design tasks that require interdependence**, where success depends on collaboration rather than individual competition.

Student Progression

Focus	Emerging	Developing	Demonstrating
Participation	Students participate when prompted but may remain hesitant to share ideas.	Students contribute ideas and engage with peers during structured activities.	Students actively participate, initiate collaboration, and support group engagement.
Collaboration	Students work alongside peers with limited interaction.	Students collaborate with peers and respond to shared goals or feedback.	Students engage in collaborative problem-solving and value diverse contributions.
Community Membership	Students view computing as an individual task.	Students recognize themselves as part of a learning group.	Students demonstrate ownership of and responsibility for an inclusive CS community.

References

Anant, S. S. (1967). Belongingness and mental health: Some research findings. *Acta Psychologica, 26,* 391–396.

Berlyne, D. E. (1954). A theory of human curiosity. *British Journal of Psychology. General Section, 45*(3), 180–191. https://doi.org/10.1111/j.2044-8295.1954.tb01243.x

Csikszentmihalyi, M. (1996). *Creativity: Flow and the psychology of discovery and invention.* Harper Collins.

CSTA, IACE, ACM, Code.org, College Board, CSforALL, & ECEP Alliance. (2024). *Reimagining CS Pathways: Every student prepared for a world powered by computing.* Association for Computing Machinery. https://reimaginingcs.org

Duckworth, A. L., Peterson, C., Matthews, M. D., & Kelly, D. R. (2007). Grit: Perseverance and passion for long-term goals. *Journal of Personality and Social Psychology, 92*(6), 1087–1101. https://doi.org/10.1037/0022-3514.92.6.1087

Dweck, C. S. (2006). *Mindset: The new psychology of success.* Random House.

Engel, S. (2015). *The hungry mind: The origins of curiosity in childhood.* Harvard University Press.

Facione, P. A. (1990). *Critical thinking: A statement of expert consensus for purposes of educational assessment and instruction.* American Philosophical Association.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Hansen, M. J., Palakal, M. J., & White, L. J. (2023). The importance of STEM sense of belonging and academic hope in enhancing persistence for low-income, underrepresented STEM students. *Journal for STEM Education Research.* https://doi.org/10.1007/s41979-023-00096-8

Kennett, D. J., & Keefer, K. (2006). Impact of learned resourcefulness and theories of intelligence on academic achievement of university students: An integrated approach. *Educational Psychology, 26*(3), 441–457. https://doi.org/10.1080/01443410500342062

Krause-Levy, S., Griswold, W. G., Porter, L., & Alvarado, C. (2021). The relationship between sense of belonging and student outcomes in CS1 and beyond. *Proceedings of the 17th ACM Conference on International Computing Education Research,* 29–41. https://doi.org/10.1145/3446871.3469748

OECD. (2018). *The future of education and skills: Education 2030* (OECD Education Policy Perspectives No. 98). OECD Publishing. https://doi.org/10.1787/54ac7020-en

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books.

Pinto, J. D., Zhang, Y., Paquette, L., & Fan, A. X. (2021). Investigating elements of student persistence in an introductory computer science course. *CEUR Workshop Proceedings, 3051.*

Plucker, J. A., Beghetto, R. A., & Dow, G. T. (2004). Why isn't creativity more important to educational psychologists? Potentials, pitfalls, and future directions in creativity research. *Educational Psychologist, 39*(2), 83–96. https://doi.org/10.1207/s15326985ep3902_1

Resnick, M. (2007). All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten. *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition (C&C '07),* 1–6. https://doi.org/10.1145/1254960.1254961

Sawyer, R. K. (2012). *Explaining creativity: The science of human innovation* (2nd ed.). Oxford University Press.

Schön, D. A. (1983). *The reflective practitioner: How professionals think in action.* Basic Books.

Sharmin, S. (2021). Creativity in CS1: A literature review. *ACM Transactions on Computing Education, 22*(2), 16:1–16:26. https://doi.org/10.1145/3459995

Veilleux, N., Bates, R., Allendoerfer, C., Jones, D., Crawford, J., & Floyd Smith, T. (2013). The relationship between belonging and ability in computer science. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education,* 65–70. https://doi.org/10.1145/2445196.2445220

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Zimmerman, B. J. (2002). Becoming a self-regulated learner: An overview. *Theory into Practice, 41*(2), 64–70.

Acknowledgements

This work is supported by the National Science Foundation under Awards No. 2311746 and 2118453. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

We are grateful to the 2026 CSTA PK–12 Standards Writers and the many people who contributed to the *Reimagining CS Pathways* project.

Suggested Citation

Computer Science Teachers Association. (2026). *Fostering Dispositions in PK–12 Computer Science.* https://csteachers.org/pk12standards/resources

csteachers.org/pk12standards/resources