

MIT Open Access Articles

Coding at a Crossroads

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

Citation: resnick, mitchell and Rusk, Natalie. 2020. "Coding at a Crossroads."

As Published: <https://doi.org/10.1145/3375546>

Publisher: ACM|Communications of the ACM

Persistent URL: <https://hdl.handle.net/1721.1/146140>

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Terms of Use: Article is made available in accordance with the publisher's policy and may be subject to US copyright law. Please refer to the publisher's site for terms of use.



While millions of students worldwide have enjoyed coding experiences over the last decade, the next challenge is spreading educational values and approaches.

BY MITCHEL RESNICK AND NATALIE RUSK

Coding at a Crossroads

THE EDUCATIONAL USE of coding in schools is at a crossroads.

We are at a moment of extraordinary opportunity. A decade ago, our research group wrote an article for *Communications* titled “Scratch: Programming for All.”¹⁵ At the time, our subtitle was aspirational. Now, it is becoming the reality. School systems and policymakers are embracing the idea that coding can and should be for everyone. Countries from Chile to England to South Africa to Japan are introducing coding to all students.

We are also at a moment of extraordinary challenge. In many places, coding is being introduced in ways that undermine its potential and promise. If we do not think carefully about the educational strategies and pedagogies for introducing coding, there is a major risk of disappointment and backlash.

During the past decade, we have seen that it is possible to spread coding experiences to millions of children around the world. But we have also seen that it is much more difficult to spread educational values

and approaches—that is the big challenge for the next decade.

The expansion of coding in education has been catalyzed by new types of programming interfaces (particularly block-based coding¹), a proliferation of nonprofit initiatives supporting computer-science education (such as Code.org, CSforAll, and Code Club), and a growing array of programmable devices that broaden the range of what students can code (such as micro:bit,²⁰ robotics kits,⁹ and programmable toys²³).

Our own work on Scratch (Figure 1) has both contributed to and benefitted from this broader trend. When we started developing the Scratch programming language and online community in 2002, our goal was not simply to help children learn to code. We had a broader educational mission. We wanted to provide all children, from all backgrounds, with opportunities to learn to think creatively, reason systematically, and work collaboratively. These skills are essential for everyone in today’s fast-changing world, not just those planning to become engineers and computing professionals. And these same skills are valuable in all aspects of life, not just for success in the workplace but also for personal fulfillment and civic engagement.¹³

The use of Scratch has been growing rapidly throughout the world: in the past year, more than 20 million young people created Scratch projects (Figure 2). Scratch began with use primarily in homes and informal learning settings,¹¹ but use in schools has expanded to more than half of all Scratch activity. Around the

» key insights

- In many educational settings, coding is introduced in narrow ways that focus primarily on teaching specific concepts, rather than supporting students in developing the creativity, collaboration, and communication skills needed to thrive in today’s fast-changing world.
- For students to develop computational fluency and creative thinking skills, they need opportunities to create projects, based on their passions, in collaboration with peers, in a playful spirit.

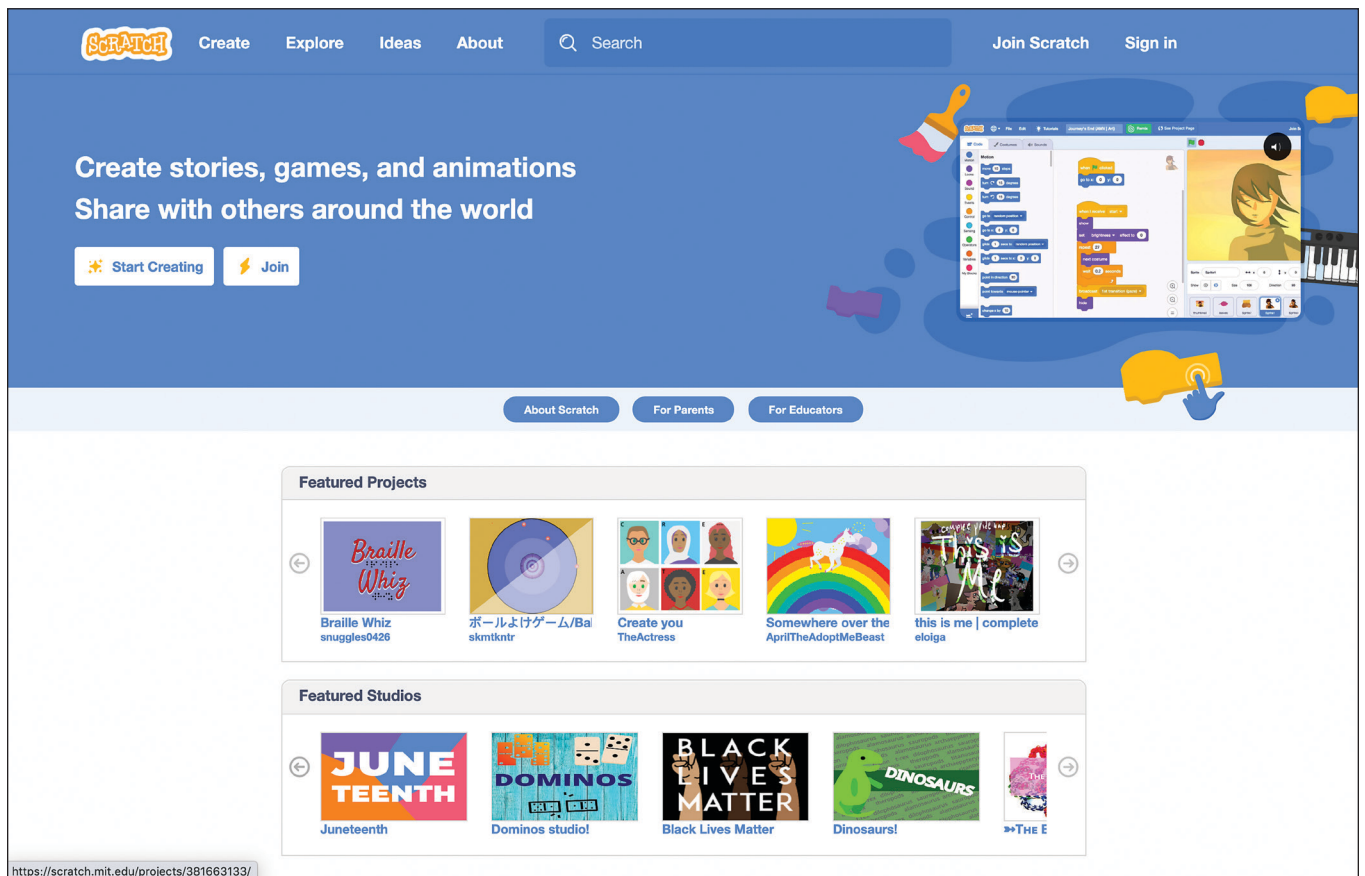


Figure 1. The Scratch website in June 2020.

world, young people are using Scratch in a wide variety of ways. For example:

- ▶ middle-school students across several countries created Scratch projects illustrating their visions for how technological innovations would transform society by the year 2050;

- ▶ thousands of young people created Scratch animations against racism and in support of the Black Lives Matter movement;

- ▶ an elementary-school teacher in Mexico integrated Scratch into a science unit on butterflies, with students creating animations of the butterfly life cycle and robotic models of butterfly motion, based on their observations of real butterflies;

- ▶ students from around the world created a studio called #ProtectOurEarth where they shared hundreds of projects highlighting issues related to climate change, including a game where you guide a polar bear across the melting Arctic ice caps.

Opportunities and Challenges

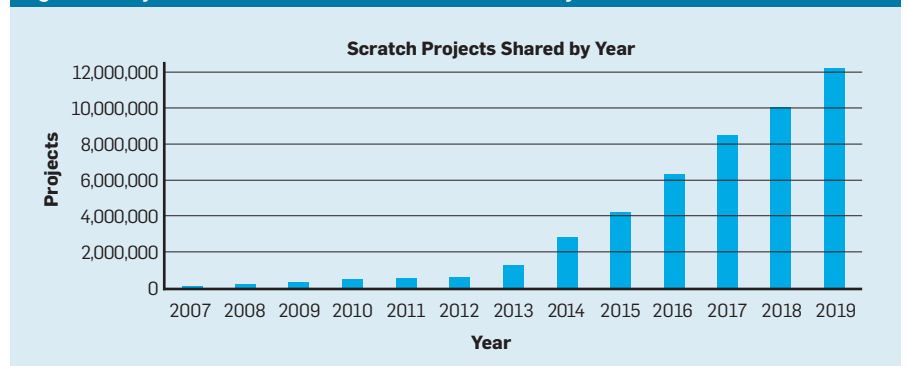
In the process of creating and sharing projects like these, students are not just

learning to code, they are coding to learn. They are not only learning important mathematical and computational concepts, they are also deepening their understanding of ideas in other disciplines and developing a broad range of problem-solving, design, collaboration, and communication skills.^{7,16}

Unfortunately, in many educational settings, coding is introduced in much more limited and constrained ways, so that students do not have the opportunity to experience the full conceptual and expressive powers of coding. Here are some of the challenges:

- ▶ Too often, schools are introducing students to computer science by teaching them definitions of words associated with computing, without providing them with opportunities to learn and apply computational concepts and practices in the context of meaningful activities. For example, some school districts introduce computing to elementary-school students by teaching them the definition of the word “algorithm” and the differences between hardware and software, instead of engaging students in active learning through computing activities, such as

Figure 2. Projects shared in the Scratch online community.




coding an animated story or programming a robot to dance.


► Too often, coding is introduced by telling all students to copy the exact same code, rather than encouraging them to experiment, prototype, and debug. On the Scratch website, we once saw 30 identical projects shared at the same time. At first we thought this duplication of projects was a problem with the website, but then we noticed that each project had a different username, and we realized the projects were all from a single classroom, where 30 students had followed the same instructions to make the same project with the same images and same code. Although this classroom activity may have introduced students to the basic mechanics of coding, it did not provide opportunities for creative thinking and problem solving.

► Too often, schools allocate only a brief period of time for learning to code. Within this limited time, students might learn some basic terms and concepts, but they don't have the opportunity to put the ideas to use in a meaningful way, and thus are unlikely to be able to apply the ideas in other contexts and other subjects. And in situations where coding is allocated more time, the curriculum often pushes teachers and students to shift from one coding tool to another, rather than providing time for learning a tool well enough for designing projects, solving problems, and communicating ideas. One large-scale initiative introduced Scratch to fourth-graders for one hour each week, then abruptly shifted to a different coding language. After teachers and students expressed frustration, the curriculum was revised.

► Too often, researchers and educators are adopting automated assessment tools that evaluate student programming projects only by analyzing the code, without considering the project goals, content, design, interface, usability, or documentation. For example, many are using an online Scratch assessment tool that gives students a “computational thinking score” based on the assumption that code with more types of programming blocks is an indication of more advanced computational thinking. This form of assessment doesn't take into



In our research, we have seen how coding becomes most motivating and meaningful for students when they have opportunities to create their own projects and express their own ideas.



consideration what the student's program is intended to do, how well it accomplishes the student's goals, whether the code works as intended, whether people are able to interact with it, or how the student's thinking develops over a series of projects. We see greater potential in other research and evaluation approaches, such as those that document and analyze teachers' facilitation practices and students' learning trajectories over time.^{6,8}

For coding initiatives to live up to their promise and potential, significant changes are needed in how coding is put into practice in educational systems around the world.

Computational Fluency

In most educational coding initiatives, there is a recognition that the goal should be broader than teaching specific programming techniques. Many educational initiatives are framed around the development of *computational thinking*—that is, helping students learn computer-science concepts and strategies that can be used in solving problems in a wide range of disciplines and contexts.²²

Computational thinking is certainly a worthy goal, but many initiatives focus too narrowly on teaching concepts out of context or presenting students with problems that have a single correct answer. In our research, we have seen how coding becomes most motivating and meaningful for students when they have opportunities to create their own projects and express their own ideas.¹⁸ Through these experiences, children develop as computational creators as well as computational thinkers. We use the phrase *computational fluency* to describe this ability to use computational technologies to communicate ideas effectively and creatively.

Our ideas about computational fluency have been informed and inspired by the long tradition of educational initiatives and research focused on engaging students in learning to write. Even though most students won't grow up to become professional journalists or novelists, there is a strong consensus that all students should learn to write. Through writing, students develop their ability to organize, express, and share ideas—and they begin to see themselves differently. The Brazilian

educator and activist Paulo Freire led literacy campaigns not simply to help people get jobs, but also to help people learn that “they can make and remake themselves.”⁵

We see the same potential for coding. Most students will not pursue careers as professional programmers or computer scientists but developing fluency with coding is valuable for everyone. As students create their own stories, games, and animations with code, they start to see themselves as creators, developing confidence and pride in their ability to create things and express themselves with new technologies.

Some advocates of computational thinking downplay the value of coding. They argue that there are many other ways to develop computational thinking skills. But we have found that coding can be a particularly effective way for students to become engaged with computational concepts, practices, and perspectives.² When students code their own projects, they encounter concepts and problem-solving strategies in a meaningful context, so the knowledge is embedded in a rich web of associations. As a result, students are better able to access and apply the knowledge in new situations.

The Scratch programming language and online community are designed specifically to support the development of computational fluency. Of course, it takes time for students to develop fluency. Many projects in the Scratch online community are very simple or poorly structured, created by students who are just starting to explore the possibilities of coding. But when students have the necessary time and support for developing their fluency, we see how they can grow as both computational thinkers and computational creators.

As an example, we would like to share the story of a Scratch community member named Taryn, who was first introduced to Scratch at her school in South Africa when she was 10 years old. A few years later, in a science class, Taryn used Scratch to program an interactive simulation of the water cycle, including two sliders for controlling the evaporation rates over the sea and over the land. In all, Taryn created a dozen different variables for the project (Figure 3).

Figure 3. Taryn’s Scratch project modeling the water cycle.

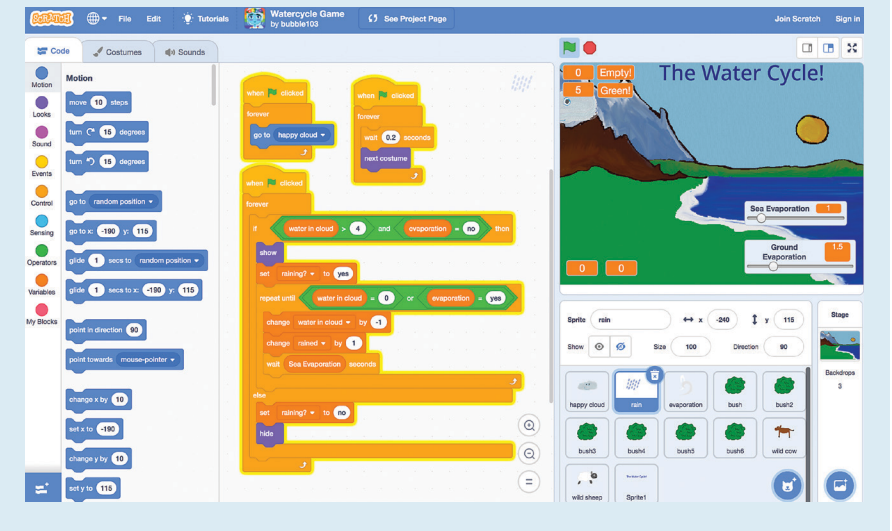


Figure 4. Taryn’s tutorial on how to use variables.

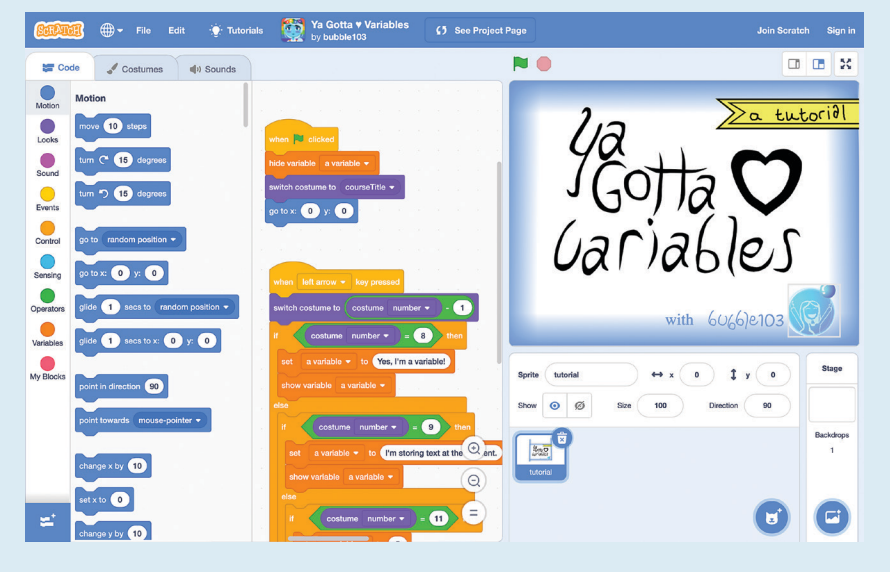


Figure 5. One of Taryn’s Colour Divide animated stories.



Through working on this project, Taryn became inspired to help others learn about variables. She decided to create a tutorial project called *Ya Gotta ♥ Variables* and shared it in the Scratch online community (Figure 4). As she explained in the notes that accompany the project: “I love variables! They’re extremely useful in programming, and I wouldn’t have been able to make most of my projects without them. However, they’re a bit tricky to understand—that’s where this tutorial can help you!” Taryn also encouraged others to experiment: “Have fun playing around and experimenting with variables and booleans! The more you experiment (and fail!), the more you will understand and the easier it will be for you to use variables to make your projects awesome!”

Taryn became well known in the Scratch community through a series of projects called *Colour Divide*, set in a fantasy dystopian world where people are subjected to a test that determines their place in society (Figure 5). Taryn collaborated on the initial *Colour Divide* project with five other students who she met in the online community. For Taryn, the project was a way to explore important social issues. When we interviewed Taryn, she explained: “Growing up, I’ve definitely seen the scars that apartheid has left on my country and the people. I’m really exploring that through the different characters that are a part of this story.”

Taryn described the important role that collaboration played in the development of *Colour Divide*. “I set it up so that other Scratchers could contribute faces and voices and scenery and music. It felt less like something that I was making, more like something that we were making together,” she said. “I’ve just been constantly blown away by the kind of support and collaboration and sharing that happens in the community. That’s one of the main things that keeps me coming back to Scratch every day.”

Through her work on Scratch, Taryn has shifted the way she approaches learning. “I’ve become more confident to try new things and express myself—and more comfortable with taking risks and making mistakes,” she explained. “In other languages, you are

almost too scared to get something wrong and type the wrong thing and be judged. But Scratch it’s like playing, it’s like chucking things together, if they don’t work, that’s fine. And being able to make mistakes is part of the thing that develops creative confidence.”

For us, Taryn’s work serves as an example of how students, through their work on Scratch projects, can develop as both computational creators and computational thinkers. We have seen many other students in the Scratch community go through similar learning trajectories. But many students don’t receive the opportunities or support they need to become fluent with computation and develop as creative thinkers. How can we help more students experience the joys and possibilities of computational fluency?

Four Guiding Principles

In our research group, we have developed four guiding principles for supporting creative learning and computational fluency. We call these principles the Four Ps of Creative Learning: *Projects*, *Passion*, *Peers*, and *Play*.¹⁴

These principles provide a framework to guide the design of technologies, activities, curriculum, communities, and spaces to support coding and learning. Here, we explore the Four Ps of Creative Learning through examples from the Scratch community.

Projects. *Provide students with opportunities to work on meaningful projects (not just puzzles or problem-solving activities), so they experience the process of turning an initial idea into a creation that can be shared with others.*

To us, it seems natural to introduce coding to young people in a project-oriented way, so that they learn to express themselves creatively as they learn to code. But many introductions to coding take a very different approach, presenting students with a series of logic puzzles in which they need to program animated characters to move from one location to another. When students successfully solve one puzzle, they can move on to the next. Students undoubtedly learn some useful computational concepts while working on these puzzles. But learning to code by solving logic puzzles is somewhat like learning to write by solving crossword puzzles. That’s not the way to become truly fluent. Just as

students develop fluency with language by writing their own stories (not just playing word games), students develop fluency with coding by creating projects (not just solving puzzles).

Increasingly, schools are shifting to a project-based approach to coding. In one school, for example, fourth-grade students created Scratch projects about the book *Charlotte’s Web*, rather than writing traditional book reports. In one of the projects, a student programmed a pig to move within the scene. To make the pig look further away, the student programmed it to become smaller, applying the art concept of perspective and using mathematical calculations to adjust the size of the pig. The project cut across the curriculum, integrating ideas from language, art, math, and computer science. In other schools, students have designed projects in many different subject areas—creating games about ancient Egypt in history class, modeling DNA replication in biology, and creating animations of haiku poems in language arts.

For teachers, it might be easier to introduce coding through puzzles that tell students whether they have correctly solved the problem or where they went wrong. Managing a project-based classroom can be more challenging, since different students will create different types of projects. Yet it is precisely this opportunity for developing an idea from initial conception to shareable project that enables young people to develop as creative thinkers and problem solvers.¹⁴

Passion. *Allow students to work on projects connected to their interests. They will work longer and harder—and learn more in the process.*

We designed Scratch to support a wide range of projects and interests—from art, music, and animations, to games, stories, and simulations. We also made sure students can customize and personalize their projects, by bringing in their own images and sounds.

Why is this important? Different children have different interests, come from different cultures, and think in different styles. Supporting diverse pathways into Scratch is important to ensure that all children, from all backgrounds, can work on Scratch projects that are relevant and meaningful to them. On the Scratch website, you can see a wide

diversity of projects, everything from interactive newsletters to dance tutorials to historical dress-up games to musical beat machines. That's an indication that Scratch is supporting students with a wide range of different interests and passions. Similarly, when evaluating Scratch classes or workshops, we use diversity of projects as a measure of success—an indication that children are working on projects they care about.

In an influential paper from the 1990s, Sherry Turkle and Seymour Papert emphasized that encouraging diverse styles of thinking and programming is essential for promoting equity and developing a more inclusive computer culture.²¹ As they wrote:

“The computer is an expressive medium that different people can make their own in their own way ... The diversity of approaches to programming suggests that equal access to even the most basic elements of computation requires accepting the validity of multiple ways of knowing and thinking, an epistemological pluralism.”

We often refer to this idea with the phrase “many paths, many styles.” Some students make elaborate plans, others explore and tinker. Some students enjoy telling stories, others enjoy making patterns. Some students are excited about animals, others are excited about sports. To ensure coding is for all, it is important to support these diverse entry points and approaches.

Peers. *Encourage collaboration and sharing, and help students learn to build on the work of others.*

When our research group launched the Scratch programming language in 2007, we launched the Scratch online community at the same time. We wanted to support the social side of learning, providing students with opportunities to learn with and from one another. The online community has grown into a dynamic space where young people collaborate with one another, sharing more than one million projects and posting more than three million comments each month.

We have learned from Scratchers just how important the online community is for motivating their ongoing participation.¹⁸ As one Scratcher explained: “I would've quit earlier, but then I made friends ... Of course, I had friends in real life, but having friends in other



The online community has grown into a dynamic space where young people collaborate with one another, sharing more than one million projects and posting more than three million comments each month.



countries with the same interests kept me coming back to talk to them.”

Young people talk about multiple reasons why the Scratch online community matters to them:

- ▶ The community provides *audience*: When young people share projects they have made, they get feedback, encouragement, and suggestions from peers in the community.

- ▶ The community provides *inspiration*: By looking at other projects on the website, young people get new ideas for their own projects.

- ▶ The community provides *connection*: Young people make friends and meet others with shared interests from other cities and countries.

As a young person in the online community reflected:

“When I used the website, I got interested in the projects of others. This is largely how I learned Scratch: through remixing and sharing and creating. I made many friends here, who remix my projects, give comments, and have taught me new things.”

As participation in the Scratch community has grown, young people have collaborated in ways beyond what we had originally anticipated. More and more young people have taken the initiative to connect, coordinate, and collaborate on projects and activities. About a quarter of all projects on the Scratch website are remixes, in which students modify or add code to existing projects.⁴ Some students form collaborative groups to create complex games and animations that none could have created on their own. Other students have learned how to create projects through crowdsourcing, asking others in the community to contribute code, images, or sound clips.¹⁷


A few years ago, a college physics professor told us his children had become actively involved in the Scratch community. We expected he would go on to tell us about the coding skills and computational ideas they were learning. But that's not what interested him most. Rather, he was excited that his children were participating in an open knowledge-building community. “It's like the scientific community,” he explained. “Kids are constantly sharing ideas and building on one another's work. They're learning how the scientific community works.”

Play. *Create an environment where students feel safe to take risks, try new things, and experiment playfully.*


Scratch is designed to encourage playful experimentation and tinkering. As with LEGO bricks, it is easy to snap together Scratch programming blocks to try out new ideas, and it is also easy to take them apart to revise and iterate. Just click on a stack of Scratch blocks, and the code runs immediately. There are no error messages in the Scratch programming editor. Instead, many children learn new coding strategies by playfully experimenting with different combinations of Scratch blocks, seeing what happens when their code runs, iteratively revising their code, and looking at code in other projects. We view “play” not as an activity but as an attitude: a willingness to experiment, take risks, and try new things.

When we have interviewed long-time Scratchers, we have found that many became engaged in coding by “messing around” with Scratch.¹⁶ For example, a long-time Scratcher explained that he learned about variables, events, and other coding concepts “just by experimenting.” Although it might seem more efficient to teach concepts through direct instruction, we have seen that many students become more engaged and gain a greater sense of agency and confidence when they learn through playful experimentation and exploration. We do offer tutorials on the Scratch website, but the tutorials are designed to encourage students to incorporate their own ideas and make their own variations, not just follow step-by-step instructions.

The Scratch community guidelines emphasize the importance of being respectful and friendly, and clearly state that Scratch “welcomes people of all ages, races, ethnicities, religions, abilities, sexual orientations, and gender identities.”¹⁹ Respectful communication and inclusiveness have become norms that experienced participants communicate to newcomers and others.¹⁰ A respectful community is essential for accomplishing our goals with Scratch. When people feel they are surrounded by caring, respectful peers, they are much more likely to play—that is, to try new things and take the risks that are an essential part of the creative process.



We have been encouraged to see a growing number of teachers and schools are finding ways to integrate creative, expressive approaches to coding into their classroom practices.



Putting the Four Ps into Practice

From our observations of Scratch activities around the world over the past decade, we have seen the value of Projects, Passion, Peers, and Play in supporting the development of computational fluency. But we have also seen that it is not easy to put these four principles into practice within the realities of today’s standards-based, assessment-driven classrooms.

We have been encouraged to see a growing number of teachers and schools are finding ways to integrate creative, expressive approaches to coding into their classroom practices. In a public high school in Tacoma, WA, for example, computer-science teacher Jaleesa Trapp wanted to provide her students with an opportunity to learn computational concepts in the context of projects that would be meaningful to them. Jaleesa noticed that many of her students enjoyed watching how-to videos online, so she proposed that they use Scratch to create their own how-to tutorials.

The students created a wide range of projects: how to crochet, how to use a 3D printer, and how to make a video game, among others. The students designed their projects to make them accessible to users with diverse abilities. To create their projects, students needed to research their topics, develop prototype tutorials, test out their prototypes with other students, revise their projects, and finally present their projects to friends and family, as well as sharing with a broader audience online.

This activity was well-aligned with the four Ps, since students were working on projects based on their passions, in collaboration with peers, in a playful spirit. But the activity was also well-aligned with computer science and engineering standards, since it involved iterative design, testing, debugging, and refinement of computer programs.^{3,12} Students gained an understanding of important computational concepts and practices (such as using control structures and improving usability) through working on their projects.

Jaleesa also wanted an assessment method that would be meaningful to the students. So, before they started designing, she asked the students to help develop a rubric for evaluating their projects. They began by identifying the features of how-to videos that

they valued and decided together which criteria were most important to include in the rubric. By contributing to the criteria for assessment, the students developed a shared understanding of the goals, and they were invested in meeting them.

Jaleesa noted that many computer-science initiatives evaluate students based on how many different programming blocks they use in their projects. Jaleesa worried that focusing on this metric might lead students to simply add programming blocks to fulfill a requirement, without understanding the purpose of the different blocks. Instead, the students in Jaleesa’s class used a wide variety of programming blocks in an authentic way. Because students were designing how-to projects to support accessibility, they naturally needed to coordinate multiple events, incorporate multiple types of media, and respond to different types of user input.

The Next Decade

We are at a moment of great opportunity but also great challenge. Even as new technologies have flowed into schools and as new coding initiatives have been adopted, the core structures of most educational institutions have remained largely unchanged. If new technologies and new coding initiatives are to live up to their promise, we must break down structural barriers in the educational system.

We need to break down barriers across disciplines, providing students with opportunities to work on projects that integrate science, art, engineering, and design. We need to break down barriers across age, allowing people of all ages to learn with and from one another. We need to break down barriers across space, connecting activities in schools, community centers, and homes. And we need to break down barriers across time, enabling children to work on interest-based projects for weeks or months, rather than squeezing projects into the constraints of a class period or curriculum unit.

Breaking down these structural barriers is difficult. It requires a shift in the ways people think about education and learning. People need to view education not as a way to deliver information, but rather as a way to support stu-

dents in exploring, experimenting, and expressing themselves, so that students can develop the creativity, collaboration, and communication skills that are needed to thrive in today’s fast-changing world.

These changes in structures and mindsets will require efforts by many people, in many places, at many levels. There are already teachers, schools, and even entire districts that are implementing new, creative approaches to coding and learning. We need to build on these examples to support broader change. No individual policy or individual school or individual technology can bring about change on its own. We need a movement in which people in all parts of the educational ecosystem—educators, administrators, researchers, curriculum developers, toolmakers, and policymakers—think about coding in new ways and think about learning in new ways.

We are at a crossroads. Ten years from now, we hope we can look back and report on a decade of educational change, in which schools have provided students with the time, space, support, and encouragement they need to become fluent with new technologies, so that they can help shape tomorrow’s society.

Acknowledgments

Many people have contributed to the design, development, and support of Scratch, particularly members of the Lifelong Kindergarten Group at the MIT Media Lab and the Scratch Team at the Scratch Foundation. We are grateful to the National Science Foundation for supporting the initial research and development of Scratch, and to the Siegel Family Endowment, LEGO Foundation, and other supporters for making it possible to make Scratch available for free for young people and educators around the world. C

References

1. Bau, D., Gray, J., Kelleher, C., Sheldon, J. and Turbak, F. Learnable programming: blocks and beyond. *Commun. ACM* 60, 6 (Jun. 2017), 72–80; <https://dl.acm.org/citation.cfm?doi=3098997.3015455>
2. Brennan, K. and Resnick, M. Using artifact-based interviews to study the development of computational thinking in interactive media design. *Annual Meeting of the American Educational Research Association*, Vancouver, B.C. 2012.
3. Computer Science Teachers Association. *CSTA K-12 Computer Science Standards*, 2017; <http://www.csteachers.org/standards>
4. Dasgupta, W.H., Monroy-Hernández, A. and Hill, B.M. Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-*

Supported Cooperative Work & Social Computing (2016). ACM, New York, 1438–1449. <https://doi.org/10.1145/2818048.2819984>

5. Freire, P. *Pedagogy of Indignation*. Paradigm, Boulder, CO, 2014.
6. Israel, M., Pearson, J.N., Tapia, T., Wherfel, Q.M., and Reese, G. Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, 82 (Mar. 2015), 263–279; <https://doi.org/10.1016/j.compedu.2014.11.022>
7. Kafai, Y.B. and Burke, Q. *Connected Code: Why Children Need to Learn Programming*. MIT Press, Cambridge, MA, 2014.
8. Ke, F. An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education*, 73 (Apr. 2014), 26–39.
9. Khine, M.S. *Robotics in STEM Education*. Springer, 2017; <https://doi.org/10.1007/978-3-319-57786-9>
10. Lombana-Bermudez, A. Moderation and sense of community in a youth-oriented online platform. 2017; <https://bit.ly/2NfpxEl>
11. Maloney, J., Peppler, K., Kafai, Y., Resnick, M., and Rusk, N. Programming by choice: Urban youth learning programming with Scratch. *ACM SIGCSE Bulletin* 40, 1 (Mar. 2008), 367–371.
12. NGSS Lead States. *Next Generation Science Standards: For States, by States*. National Academies Press, Washington, D.C., 2013.
13. National Research Council. *Education for Life and Work: Developing Transferable Knowledge and Skills in the 21st Century*. National Academies Press, Washington, D.C., 2013
14. Resnick, M. *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. MIT Press, Cambridge, MA, 2017.
15. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. Scratch: Programming for all. *Commun. ACM* 52, 11 (Nov. 2009), 60–67.
16. Roque, R. and Rusk, N. Youth perspectives on their development in a coding community. *Info. Learning Sci.* (Apr. 2019); <https://doi.org/10.1108/ILS-05-2018-0038>
17. Roque, R., Rusk, N., and Resnick, M. 2016. Supporting diverse and creative collaboration in the Scratch online community. *Mass Collaboration and Education*. U. Cress, H. Jeong, and J. Maskaliuk (Eds.) Springer, Cham, Switzerland. 241–256; https://doi.org/10.1007/978-3-319-13536-6_12
18. Rusk, N. Motivation for making. *Makeology: Makers as Learners*, K. Peppler, E. Rosenfeld Halverson, and Y.B. Kafai (Eds.). Routledge, New York, NY, 85–108.
19. Scratch Community Guidelines. 2018; http://scratch.mit.edu/community_guidelines/
20. Sentance, S., Waite, J., Hodges, S., MacLeod, E., and Yeomans, L. ‘Creating cool stuff’: Pupils’ experience of the BBC micro:bit. In *Proceedings of the 2017 ACM SIGCSE* (Seattle, WA) 531–536.
21. Turkle, S. and Papert, S. Epistemological pluralism: Styles and voices within the computer culture. *SIGNS*: 16, 1 (1990), 128–157.
22. Wing, J.M. Computational thinking. *Commun. ACM* 49, 3 (Mar. 2006), 33–35.
23. Yu, J. and Roque, R. A review of computational toys and kits for young children. *Int’l J. Child-Computer Interaction*. (Jul. 2019); <https://doi.org/10.1016/j.jcci.2019.04.001>

Mitchel Resnick (mres@media.mit.edu) is Professor of Learning Research at the MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA.

Natalie Rusk (nrusk@media.mit.edu) is Research Scientist in the Lifelong Kindergarten group at the MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA, USA.

Copyright held by author/owner.
Publication rights licensed to ACM.



Watch the authors discuss this work in the exclusive *Communications* video. <https://caem.acm.org/videos/coding-at-a-crossroads>