

## Do Digital Literacies Include Programming?



ISSUE ORIENTED

By Anita McAnear

*Anita McAnear is Le&Ls acquisitions editor and national program chair for NECC. A former middle school math and language arts teacher, McAnear has been with ISTE since 1983.*

Starting back in the early days when all you could do with a computer was program it, through the early days of computer software, many of us thought that to take full advantage of this tool we would all need to develop programming skills. As word processors, databases, and spreadsheets developed, computer educators started to question this idea. “You don’t need to be a car mechanic to know how to drive one,” we reasoned.

Soon, there was so much that one could do with a computer that the concerns shifted to developing infrastructure in schools and classrooms, providing access, exploring effective use, and educating students, teachers, and administrators about these tools. I, for one, would still like to see all students have the chance to experience programming, especially now that we have so many fun environments in which they can do so. Tools such as Lego Mindstorms and Crickets let students see immediately if their programming is successful, and they get to experience math, science, and engineering concepts in action.

I was surprised at TCEA in February to hear Marc Prensky claim again that all students should learn to program. Then I read his February/March *Edutopia* article, “Programming: The New Literacy,” where he argues that the skill set of an educated person will soon include programming fluency.

Prensky defines *programming literacy* as “the ability to make digital technology do whatever, within the possible, one wants it to do—to bend digital technology to one’s needs, purposes, and will, just as in the present we bend words and

images.” His definition of programming at the lowest level is operating an on/off switch.

That sounds a little broader than what I have always thought of as programming, which is using a computer language to instruct the machine to do something, whether directly (i.e., using machine code) or using a high-level programming language such as Flash or HTML.

W. T. Tsai and his coauthors (page 28), however, describe a new computer science paradigm called Service-Oriented Computing, in which programmers construct software using reusable services, or components with standard interfaces. This type of programming does start to sound more like Prensky’s description of programming languages that consist of menus and choices. I would have said that many of his examples of programming were just following directions. But our devices have so many choices that it is impossible to have a set of directions for all possibilities.

Prensky doesn’t believe that students will want to stop at this level. They will go on to learn programming languages such as XML, PHP, and languages for creating games. As we move through the 21<sup>st</sup> century, he believes this is the level that will approach programming fluency for an educated person.

One thing is clear: As technology keeps advancing, the bar is raised for teachers. Accordingly, the new NETS for Teachers, which will debut in late June at NECC, also raise the bar. Some have argued that it is too high, given where most teachers are with their technology skills, but what choice do we really have? Students and society won’t wait! ■