Teaching

Approaches for teaching computing

What makes a good computing lesson?

There's a wealth of learning theory, academic research and professional practice, including Ofsted's expectations, that we can draw on to help address this crucial question. Good practice in computing is unlikely to be different from good practice across the primary curriculum. We can draw on what's effective in other subject areas which have much in common with computing: science, design and technology, art and design and music teaching.

What approaches are useful for teaching computing?

Educational theory can be mined for insights into how a new subject like computing might be taught. The pragmatic teacher is likely to draw on a blend of these approaches.

- **Experimenting:** Provide pupils with a chance to explore and tinker with new software or hardware when they first encounter it, so they can figure out their own mental model for how it works. This can be particularly effective with younger pupils.
- **Making:** A lot can be learnt through the process of making things to show to or share with others. This might be computer code, but it might also be PowerPoint presentations, web pages, edited video, digital photographs, etc.
- **Discussion:** Make the most of pupils' different insights, experiences and backgrounds by allowing them to share their ideas with one another and with others. Paired programming activities in class and online discussion forums are just two ways to facilitate this.
- **Connecting:** Learning from others need not be limited to the classroom: encourage pupils to explore others' solutions to problems on the Kodu or Scratch community sites, for example, or to search online for solutions to problems.
- **Direct instruction:** For some ideas in computing, the traditional, direct instruction approach can work well. Complex ideas such as variables, how the internet works or how search engines operate could be learnt using discovery-based approaches, but direct teaching is likely to be more effective.
- **Practise:** Don't assume that once pupils have demonstrated they can do something or understand an idea that their learning is secure. Provide opportunities for them to practise applying their skills, knowledge and understanding.

What does Ofsted expect?

Ofsted's expectations of good or outstanding lessons are the same irrespective of subject, and are outlined in the current edition of the *School inspection handbook* (Ofsted 2014).

The *School inspection handbook* makes clear the importance of inclusion, as discussed in relation to planning for computing (page 51). Thus, for teaching to be considered outstanding:

> *almost all pupils currently on roll in the school, including disabled pupils, those who have special educational needs, disadvantaged pupils and the most able, are making sustained progress that leads to outstanding achievement.*[1]

In his presentation on inspecting computing, David Brown made some suggestions for what good or outstanding teaching in computing might look like.

He recommended that:

- *it is informed by excellent subject knowledge and understanding of continuing developments in teaching and learning in computing*
- *it is rooted in the development of pupils' understanding of important concepts and progression within the lesson and over time; it enables pupils to make connections between individual topics and to see the 'big picture'*
- *lessons address pupils' misconceptions very effectively; teachers' responses to pupils' questions are accurate and highly effective in stimulating further thought*
- *teachers use a very wide range of innovative and imaginative resources and teaching strategies to stimulate pupils' active participation in their learning and secure good or better progress across all aspects of the subject.*[2]

When commenting on pupils' achievement in computing, David Brown suggested that this would be good or outstanding if:

- *pupils demonstrate excellent understanding of important concepts in all three strands of the computing curriculum and are able to make connections within the subject because they have highly developed transferable knowledge, skills and understanding*
- *pupils show high levels of originality, imagination, creativity and innovation in their understanding and application of skills in computing*

but would be regarded as inadequate if:

- *pupils rarely demonstrate creativity or originality in their use of computing but seem confined to following instructions.*[3]

What about beyond the classroom?

Look for ways in which pupils can take their learning further. Here are just a few examples.

- Code Club (see Further resources) run after-school coding clubs in around 2,000 primary schools and make their activities available for others to use as they wish. Typical clubs are run by volunteers, although teacher involvement is also needed. Code Club provide an introductory Scratch programming course and a more advanced Scratch coding course, as well as courses that introduce the basics of HTML coding for web-development and Python programming.
- Many schools have found it helpful to institute a system of pupil 'digital leaders', who can help teachers and other pupils with some limited tech support, as well as being the first to try out new software and hardware and even advising on the school's technology policies. Typically, schools run an open application process for these roles, inviting potential digital leaders to set out in writing why they would be well suited to the role.
- Activities such as CoderDojo and Young Rewired State (see Further resources) rely on parental support for primary aged pupils and involvement from those working in the software industry.
- Some primary pupils might, with parental permission, become active participants in online communities, such as those around Kodu, Scratch or even YouTube and blogging. Others might pursue more advanced coding skills, perhaps using online interactive tutorials such as those offered by Codecademy (see Further resources), or teaching themselves how to develop for Windows Phone, Android or iOS.

Try to find ways in which this sort of advanced learning beyond school can be brought in to class and shared with other pupils.

Further resources

- David Brown (Ofsted) 'Inspecting computing' (Barefoot Computing Conference) slides.
- Codecademy teaching resources, available at: www.codecademy.com/schools/curriculum.
- Code Club network of after-school coding clubs, available at: www.codeclub.org.uk/.
- CoderDojo network of free computing programming clubs, available at: https://coderdojo.com/.
- Cousin, G., 'An introduction to threshold concepts' (2006), available at: www.et.kent.edu/fpdc-db/files/DD%2002-threshold.pdf.
- Hattie, J., *Visible Learning for Teaching: Maximising Impact on Learning* (Routledge, 2009).
- Papert, S., *Mindstorms: Children, Computers, and Powerful Ideas* (Basic Books, 1980).
- The Sutton Trust Education Endowment Foundation Teaching and Learning Toolkit, for information on research and guidance on using resources for disadvantaged pupils, available at: http://educationendowmentfoundation.org.uk/toolkit/.
- Young Rewired State community of young digital makers, available at: www.yrs.io/.

[1] ***School inspection handbook*** **(Ofsted, 2014).**
[2] **Brown, D., Ofsted National Lead for Computing, 'Inspecting computing' slides (Barefoot Computing Conference).**
[3] **ibid.**