Sizing Exercises Correctly

Adrian Kosmaczewski

2023-03-10

I have often learned technical subjects online or in person with oversized exercises. By that, I mean sample code or applications that are needlessly complex and contrived, to the point that their complexity hides the main subject of the class. Such examples are hard to install, run, and understand, and teachers need to spend more time helping their students to run the code than actually explaining the subject.

By making this mistake, online teachers destroy the purpose of their learning proposition, their main point buried beneath lots of unnecessary code.

What is the right size of a code example? The answer to this question depends on various factors. I do not have a rule of thumb other than the following: make it as small as possible and then some more.

Code samples for online learning must fulfill the following characteristics:

Simple to install. Learners must be able to install and run the code in seconds unless the course is about the installation process itself. If the examples are complex to set up, provide an installer, a "curl-pipe-bash" script, a Helm chart, or any other mechanism that makes deploying code as fast as possible. Your students are here to learn the gist of the code, not to spend the first ten minutes of a 45-minute session setting up their environment. Unless, of course, you're teaching "Kubernetes the Hard Way" or some similar subject.

Short. One hundred lines of code are too much. 50 is too much. Five is not enough. Ten to twenty is about right. Remove any boilerplate code required for the code to run and place it in a different file, on a separate module, or in a library. Make those available to students, but keep the example itself neat. Unless, of course, you're teaching software architecture or some bigger-than-life subject where size is the main topic.

Relevant. This is largely a consequence of the previous point. Do not mix and match clever tips and tricks around the API call or design pattern you want to show. Concentrate on the thing you want to teach; ask yourself, what message conveys this code? What problem is it solving? If the answer is longer than expected, break the code sample into smaller pieces.

No eye-candy. Many teachers spend a lot of time "pinping up" their code examples with eye-catching features, when sometimes just a white screen with some text is enough to get the point across. The visuals on your demos and

sample code do not need to win an award. Unless, of course, you're teaching accessibility, design, or some other related subject.