**The game plan for the view**

We've just implemented the code to compute the next generation of cells, and we're storing that state in the nextGrid array. But we don't see that next generation in the view, because all we're doing so far is updating the model—the arrays where we're storing the state of the game.

There are two things we need to do to update the view. First, we need to copy the state that's in nextGrid to grid. We do that so that grid contains the *current state* of the game. Using our graph paper metaphor, it's like you're taking the 2nd sheet of graph paper---the one you just used to create the next generation of cells---and you're putting it on top of the first piece of graph paper which you don't need anymore. And, at the same time, you'll grab a fresh piece of graph paper to compute the next generation. So, we'll write a function copyAndResetGrid to copy the state from nextGrid to grid, and, at the same time, clear out the nextGrid array, setting every item to 0, so we're starting fresh.

Once we've got the new generation we just computed stored in grid, we'll use it to update the view. We'll do that in a new function, updateView, by iterating through the grid array, looking at each item in the array, and setting the class of the corresponding cell in the table to either "dead" or "live" depending on if the array is storing a 0 or 1 at a given location.