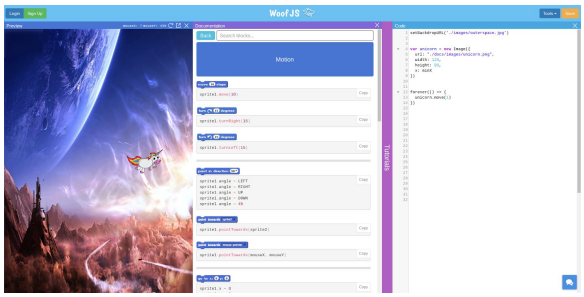
Top-down programming assistant

There are many ways for a new programmer to get “stuck” while learning to code, and require the help of a more experienced programmer to get them “unstuck.” For example, it’s virtually impossible for a new programmer to spot a missing closing-bracket. That’s why a block-based coding platform like Scratch is so wonderful. By eliminating syntax errors, it allows new programmers to spend more time learning, being productive, and enjoying coding. One way to optimize a programming environment is minimizing the time learners are stuck.

In this vein, I’d like to discuss a problem that my students encounter while coding in WoofJS, and a prototype that I’m working on to solve it that I call “WoofJS Workflow”. For reference, these students are aged 8-14, and attend classes at my after school program, The Coding Space. We built WoofJS (woofjs.com) to be “the next step after Scratch” for our more advanced students. It’s a JavaScript framework and IDE for students to make games and animations. For each block in Scratch, there’s an equivalent JavaScript command in Woof that does the same thing. In this way, WoofJS is an alternative to ProcessingJS.



As simple as it sounds, I’ve found that the most common way students get stuck is when *they lose track of what they’re working on*. Thus when any student asks for help, we first ask, “What are you trying to accomplish here?” More often than not, the student will have entirely forgotten what she’s trying to do, and respond with a shoulder shrug.

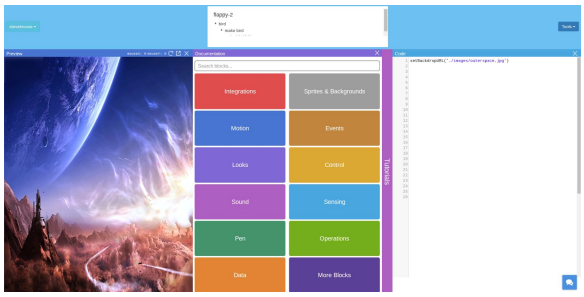
If the student is working through one of our guided tutorials, which have numbered steps, we can then follow up with, “What number step are you on?” and then, “Could you read it aloud?” Sometimes, simply re-focusing the student’s attention on the problem at hand allows her to solve her issue herself with no further teacher input. But even if the student needs more guidance, the conversation is much more productive now that teacher and student are aligned on the goal.

However, when students are working on creative projects of their own devising, there is no numbered list of steps for them to refer back to. While I’m constantly begging students to write their steps out for themselves on paper, it’s a losing struggle that almost never happens. Many creative student projects never see the light of day because students get stuck, frustrated, and quit, going back to our guided projects.

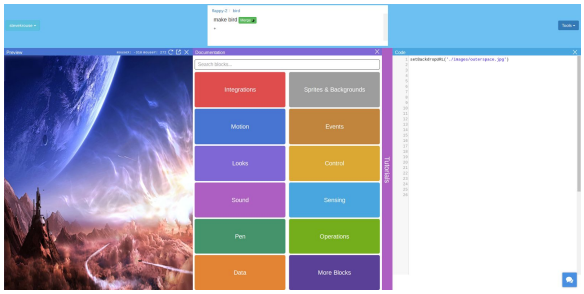
While *staying on task* is vital to many constructive endeavors, it’s particularly difficult in programming because our tasks have subtasks with subtasks with subtasks with subtasks. When a student gets stuck on a subtask 3 levels deep, they often lose sight of the larger picture and what they’re even trying to accomplish in the first place. Luckily, programming has a technique for dealing with breaking down problems into subproblems - *top-down programming*.

WoofJS Workflow

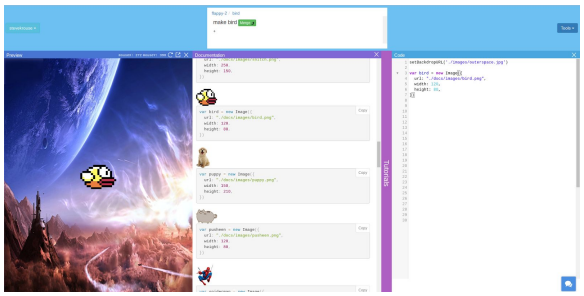
While top-down programming is indisputably a core programming competency, our tooling doesn't lend itself well to it, which is why we often resort to whiteboards, pieces of paper, or github issues. What would it look like to integrate a top-down programming assistant into an IDE?



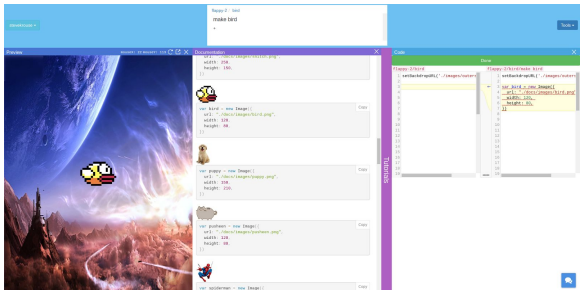
I think it would look like a nested To Do list right at the top of the screen. That way, what the student is working on now is always top of mind. Even better, the breadcrumb trail explaining how what they're working on now plays in the larger picture is also right there.



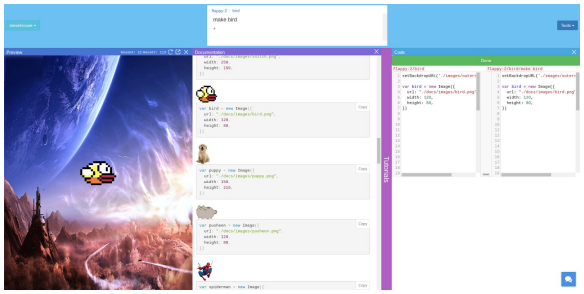
Here's where it gets interesting. Each To Do item represents a separate isolated branch of code. When you drill into a To Do item (by clicking its bullet), you see the code only for that branch and can edit it without affecting the other branches.



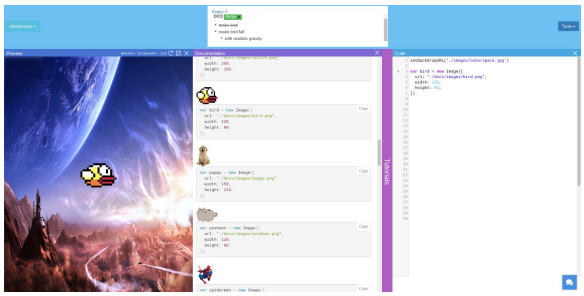
Then, when you've completed a To Do task, you simply hit "Merge" to merge this branch with its parent branch.



The merge process shows you the parent and child versions of the code side by side and allows you to selectively move sections from the child over to the parent.



When you are done merging, you hit "Done" to complete the merge. (Have you ever thought it strange that in git you first decide what to work on, then do it, and at the end write about what you did? Doesn't that seem backwards?)



As you can see, this workflow skips the “git add” and “git push” steps. It also makes it much easier to switch between branches -- no need to stash changes before checking out a new branch. Those changes will be waiting for you at this branch whenever you come back to it.

While this tool is technically no more powerful than git and github, I think it would enable greater collaboration, because by encouraging you to break your problems down explicitly, it allows teammates to go directly to a sub-problem 6 levels deep and help you there. Currently, we only collaborate with pull requests one level deep, but I think a tool like this could change this, enabling deeper collaboration without much thought for coordination.