Code + Stdin..

Stdin is like a file… you access it via process.stdin

Console.log goes to sdout

Find an assertion library!!!! So you don’t have to write your own…

Consider looking node.js assert

TO DO FOR YOUR PROJECT:

Figure out how to install the latest version of javascript for the compile box…

(ES 6.. let and const… and also things like promises, and react)

Figure out how to make the code look nicer (like with differnet lines, and parsing so you can see color)

Need to append code to the end (for your tests)… need to prepend code that user cannot edit!

Save your info in github repos… and then have your project just load those files up

(use the “raw” link on github, and just turn it all into a string)

use babel to parse code… get data structure.. AST

run babel as code..

use babel to make sure they are returning NOT a string literal… test things like the function must actually return a function. (curried function)

there’s a tool that visualizes the AST.. (abstract syntax tree… ) https://astexplorer.net/

---

code mirror!

Just make a white box … styling… so that part of the code they cant change…

NOTES:

Docker… remember there are IMAGES (the permanent stuff..) and the CONTAINERS (which are just instances of images, where you do whatever you want but whatever happens is evanescent)

The principle file by which you build your container:

There are 2 files in compilebox in your fellowship directory… dockerfile, and updateDocker.sh… (which actually runs the build command, taking in stuff from dockerfile)

.. those are the 2 files you edit… just remember if you change the image name (e.g., virtual machine, or bug\_slayer\_docker\_img.. or whatever).. make sure you are pointing to that image name in your api/compile.js bug slayer (the variable: vm\_name)

Also keep in mind the API folder has its own app.js and express server which you can run.. but no need to actually use it (in fact I would recommend getting rid of those things you don’t need)