Addendum

Portfolio 1 - browser_cli

Nathan Karasch

- **The following three links (also listed in the back of the write-up) will be especially helpful during your evaluation:**
  - View the live demo at https://krashleviathan.github.io/browser_cli/
  - View the hyperlinked source code at https://www.crossdart.info/p/browser_cli/1.0.0-alpha.3
  - View the full API Documentation at https://www.dartdocs.org/documentation/browser_cli/1.0.0-alpha.3/ . If you click on the libraries on the left hand side, you can see all the public classes in that library. If you click on the classes, you can see all the public API within that class. By clicking on the methods within the class, you can also see the source code for each method.

- **Demo**
  - Type `help` to get some basic information about using the interface.
  - Type `help --list` or `commands` to see a list of commands.
  - Try using tab completion on a command. What happens when there are multiple commands that could be completed? Press tab twice to see!
  - Try cycling through previous commands with UP and DOWN.
  - Try setting a temporary variable.
    - `dog=”Mishka”`
  - Try recalling the variable in the `echo` comformand.
    - `echo My dog’s name is $dog.`
  - Try setting a persisting variable with the `export` command. Not sure how? Type `help export`.
  - Print out all the environment variables with `printenv`. What happens when you refresh the page? (cookies must be enabled)
  - Try using `unset` on a variable and printing the environment variables again
  - Try out the `testinput` command.

- **Code Evaluation**
  - The easiest way to evaluate the code will be in the hyperlinked source code link given above
  - Otherwise, if you plan on evaluating the code in the zip file, you should probably add a plugin to your IDE to give you syntax highlighting for dart. (Emacs has dart-mode package, Sublime has something else, WebStorm has it preinstalled I think…)
  - Start by glancing at web/main.dart. This shows how easy it is for someone to set up browser_cli for their own project. Then they just make some processes of their own similar to the ones in the lib/src/processes folder, and make sure to register them in main.dart.
  - I think most of the meat of the code is in the following files:
    - lib/
      - command_line_interface.dart
      - evironment_variables.dart
      - process_manager.dart
    - src/
      - command_line_interface/
        - key_binding_manager.dart (simple, but effective!)
    - process_manager/
      - process.dart (the factory pattern works wonderfully!)
  - If you want to glance at the concrete implementations of the Process class, here are some of the more interesting ones:
    - help_process.dart
    - testinput_process.dart