Localstar: Notebook programming in browser-driven local-first executables

Grant Hames
Software Engineering | hamesg@uvic.ca

Dylan Brown
Software Engineering | dylanbrown@uvic.ca

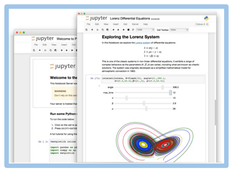
Neil Ernst
SENG 499 Supervisor | nernst@uvic.ca

SENG 499
Spring 2021

Introduction

Notebooks are emerging as a critical part of education worldwide, primarily through the Jupyter Notebook system.

Unfortunately it brings many pitfalls and difficulties to students and instructors. Is there a brighter future for this technology?



- Difficult and large installation
- Many moving parts include the CLI and package managers
- Limited graphics support

What does a better experience look like?

- Local-First: Lives on your laptop; private and secure
- Flexible: Evolves with you through plugins
- Explorable: Learn from any resources on the web. Copy/Paste snippets, share notebooks, and find community

Project Goals

Deliver a smoother notebook programming experience for use by students, instructors, and researchers in education.

Create a hackable platform that enables people to develop the system into the future for their needs. Embrace modern web technologies to deliver highly interactive visualizations.

Design Objectives

Observing the target audience of this project, we have defined a set of principles from which our design objectives are derived.

- Low-setup
- Deployable
- Available
- Private
- Interoperable
- Modern
- Future-proof

- Lower is better: bandwidth usage; disk usage; memory usage; complexity when installed on disk; number of dependencies; number of installation steps or interactivity to install; number of launch steps or interactivity to launch

- Higher is better: real world applicability to users; programming language availability; activity, stability, and health of dependencies; portability; interoperability

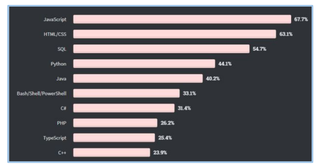
Objectives

1. < 30MB download for a single executable binary.
2. No installation; self-contained.
3. Double click to run.
4. Optionally could run remotely like Jupyter.
5. Memory usage comparable to Jupyter.
6. Supports the most popular and applicable programming languages.
   a. Full Python support.
   b. Full support for `.ipynb` Jupyter.
7. Built with stable technologies:
   a. Either stable/finished or actively maintained with funding.
   b. Codebase is approachable by the target audience of our platform, should they need to dive into it one day.

Literature Survey

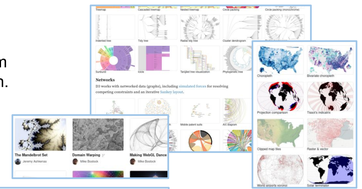
Based on what people will like to work in, and what they would like to build

- Textbooks / Courses
- Visualizations with animation and interactivity
- Choice of language
- Plugin-like flexibility

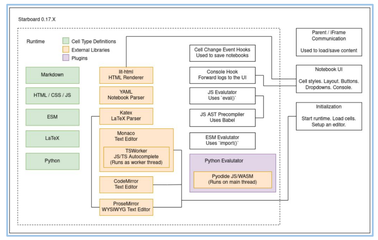


Supporting the web platform is a great start; so is Python.

Possible to have both?



Yes. With **Starboard**



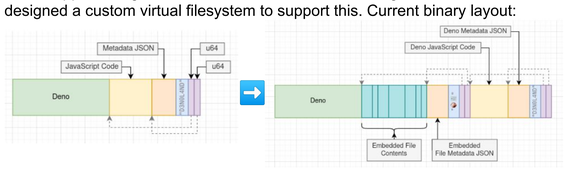
Installation free?
Yes: **Deno & Browsers**

Running Python in a browser? Yes: **WASM**.

Design Methodology

Executable

Deno supports single-file executables but not embedding files. We have designed a custom virtual filesystem to support this. Current binary layout:



Server

- HTTP REST server. Relays built-in and computer files
- No dependencies. Kept lightweight.
- Ready to support arbitrary code execution for custom plugins and notebook cell types

Client

Technologies

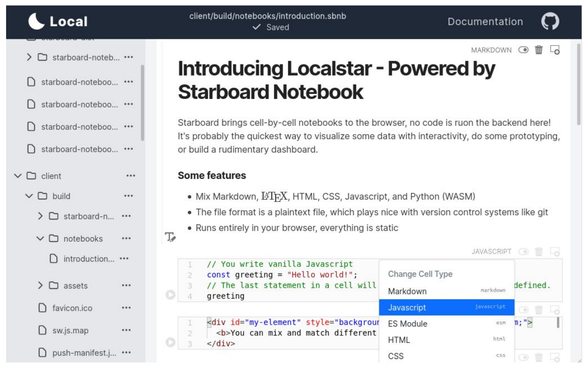
- Preact JavaScript framework for UI
- Redux for state management
- TailwindCSS for styling

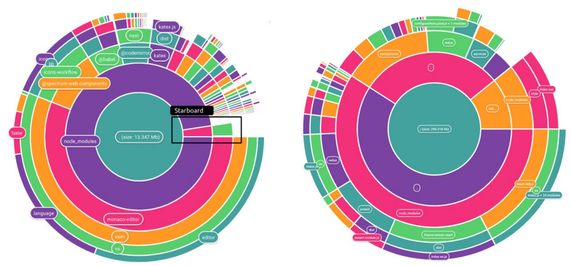
Implementation

- Integrate Starboard notebook through iframe and accompanying web-standard communication mechanism
- Provide file system navigator to support file and directory creation, deletion, and modification. HTTP REST with plans to use Server Side Events in the future.
- Implement a context menu to support filesystem and provide standardized file and directory control.
- Implement autosave functionality to prevent data loss through efficient debounced updates.
- Project environment file to store client state for repeated use and personalization.

Final Design

Below is Localstar running from a Deno local executable in Firefox. The local filesystem is connected and shown on the left.





Starboard + Dependencies: 12.9 MB

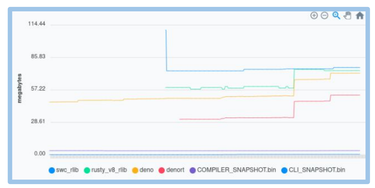
Localstar + Dependencies: 290kb

Testing and Validation

- 37 unit tests across 9 test suites covered the most critical sources of error in the client code. All pass in final build
- Compatibility testing was done to test cross-platform support and the self-contained nature of Localstar using a Windows Sandbox environment. Localstar successfully ran, without additional resources under these parameters.
- Google Lighthouse was used for accessibility testing, yielding 100/100 for all applicable automated tests
- More tests are needed but time constraints of project restricted their implementation. Future testing is planned to cover browser support, and end to end testing.

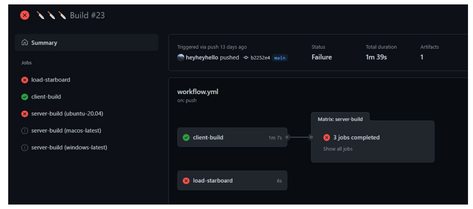
Validating size expectatons - our chosen platform grew over the course of our project. Now 65MB binary for Linux. 38MB for Windows.

This is still smaller than `miniconda` Python installation



Discussion

We still believe using Starboard was the correct choice to achieve the goals we set out to. However, it also led to complications with Python cell fragility and concerns over lack of package support.

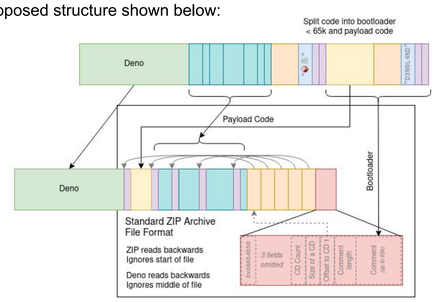


GitHub Actions as seen in the image above, our CI tool of choice, was far more problematic than we would have liked. Many development hours were spent supporting this pipeline that could have been spent elsewhere.

Recommendations

New Features

- Client file system improvements, specifically relocation of files
- We recommend future work to be done to implement local filesystem access in Python/Pyodide.
- Project file feature was not implemented due to time constraints but would help achieve the low-setup principal of this project.
- For performance, we recommend offloading some code to worker threads or servers as opposed to in browser execution.
- Current embed layout could be ZIP archive, allowing easy edits, proposed structure shown below:



Implementation Changes

- Switching from REST to WebSockets or Server Sent Events would allow server to announce local filesystem changes and improve synchronization between client and local filesystem.
- We recommend further research be done on running Pyodide in a web worker, or another method from isolating Python execution from Starboard, addressing the problems in the discussion
- As mentioned in testing, we recommend the testing suite be improved and built upon, which would further our principal goals of having a maintainable, and accessible codebase for our target audience

Conclusion

- Localstar provides a notebook platform foundation with improvements over the current Jupyter-based notebook experience.
- Future-oriented design while remaining simple, hackable, performant, and easy to use.
- We are excited for the future of the project and the social changes which will come with increased use of notebooks in education.