Satyrn: A Markdown-based JavaScript Sandbox

a project from Rebooting the Web of Trust VIII

https://github.com/WebOfTrustInfo/satyrn

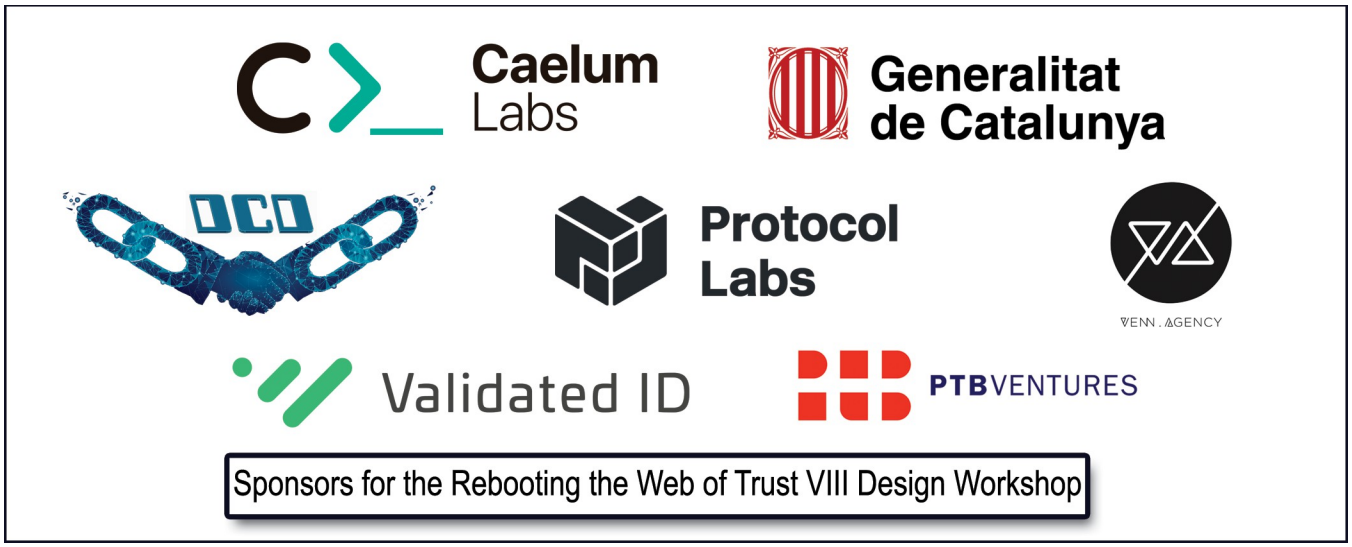
by Joe Andrieu (joe@legreq.com), Eric Welton (eric@korsimoro.com),
Will Abramson (will.abramson@napier.ac.uk), and Ganesh Annan (gannan@digitalbazaar.com)

INTRODUCTION

We set out to create a JavaScript native interpretation of Jupyter https://jupyter.org/, a notebook for both static narrative and interactive code.

From Joe Andrieu:

> *First, Jupyter is awesome. I first learned about it in Jimmy Song's Programming Bitcoin course, which uses Jupyter to great effect with his open-source curriculum at https://github.com/jimmysong/pb-exercises*
>
> *However, as I went through the class, I had significant technical issues with installing and setting up Jupyter and I could see how awesome this would be as a framework for general programming education for folks using JavaScript. Yes, there are some tools, like JSFiddle https://jsfiddle.net/ that let you interactively play with JavaScript content, but nothing that combined the static parts of curriculum, like examples, illustrations, and explanations, with interactive parts where you could actually try some code out. It's true that Python is a beauty for working with large numbers like those found in cryptography (and hence cryptocurrency), while JavaScript requires extra work. BUT! There are still more JavaScript programmers than Python, and I could only see opportunity in making it easier for the average web developer (full stack as*



well as front-end types) to learn about bitcoin, cryptocurrencies, and the underlying cryptographic primitives.

I first set out to create such a tool with a fellow alumnus of Jimmy's course, but that didn't produce fruit.

I tried again at a BTCR hackathon, where I fell in love with Visual Studio's interactive sandbox, but couldn't push through the monolith that is VS Code to figure out how to use that sandbox mode as a generic tool for files other than the single .md provided by Microsoft. However, that experience established what became the foundation for Satyrn.js, namely using markdown for curriculum with a simple addition of an interactive editor and code execution for JavaScript code blocks.

The idea was to combine the file format of markdown—which is popular and well known to developers using github—with a simple custom rendering of JavaScript code blocks to add the interactivity we wanted.

Since there exist common tools for all of these components, this turned out to be just about the perfect scope of work for the three-day workshop in Barcelona. We used Electron to create a JavaScript native app, showdown.js to convert from markdown to html, and the Ace editor for, well, editing. Combined with the built-in support for node.js in Electron, we had a proof of concept that got the basics right using standard markdown and standard JavaScript.

Of course, there were gaps and a few bugs. And there still are (see the section on next steps for a discussion); however, the basics work. It's cross-platform and open source and should work on every platform you can run node.js on. We haven't tried it as a mobile app (definitely out of scope), but it works on Linux, MacOs, and Windows.

Now we can start developing JavaScript curriculum in markdown and see how best to advance the platform.

NEXT STEPS

New Repo

We published the initial version at https://github.com/WebOfTrustInfo/satyrn and will leave that for posterity, along with this write-up of our work (in the RWOT8 final papers directory).

We are currently transitioning to a new repo at https://github.com/satyrnjs/satyrn so the Rebooting the Web of Trust work can continue focused on the workshops while Satyrn moves on to a more traditional open source lifecycle.

Bugs and Issues

Always worth investigating and resolving.

Secure JavaScript

The biggest problem with the current app is that it has a bit of an existential quandary between being an editor

and a browser. As a an editor—like MS Word—it is document-centric. However, like a browser, it needs to be able to navigate across a variety of "chapters" in a set of curriculum, so that any given markdown file presents just one component of a larger body of education. It's also designed to work with existing node modules, so that the curriculum can @require a node-module that is, in theory, distributed with the markdown files. This is sort of the point: Satyrn is a tool where you can learn to use any node.js module. (In fact, it's much better for that than it is for browser-side frameworks, like React or Angular, which really want an HTML DOM, which we do not support.) Finally, the use of node.js gives interactive content full access to the node.js API, including the ability to read and delete files, even run arbitrary executables.

In other words, we are making it easy for people to download untrusted content, then run it with higher privileges than they might be used to for JavaScript. (The browser bends over backwards to keep js in a valid sandbox.)

We'd like to fix that. We are looking at a couple of options and we like Secure EcmaScript from the folks at Agoric, but there are a bunch of feature-related decisions we need to figure out, which should be based on a clear understanding of the use cases we want to support. That is the "big rock" for the next major version.

At the same time, this is a tool for developers. Yes, perhaps for developers who are learning—and as such maybe not so aware of the security implications—but we respect that both educators and students may want and/or need to run code that by most measures is a security risk; opening files and connecting over the network are standards operations with node.js which, deservedly, are often restricted in other contexts. We are working to provide better security without breaking the fundamental flexibility of the platform.

In the meantime, we respect our users as adults who can make informed decisions about running code from unknown sources. As a general rule, don't. If you can't trust the source of a Satyrn tutorial, don't run it. If you don't understand the security implication of a particular feature, don't run it.

What we do ensure is that no JavaScript code in a markdown file is ever run automatically. Every single line of executed JavaScript must be triggered by a Satyrn user. *You* have control over which code runs in your Satyrn. Use that power wisely.

Features

As we build out some curricula, we will identify features—and have identified some already—that we'd like to add.

A few already under development:

- Support for LaTex for math formulas.
- Using a forked process with IPC rather than running node.js interactively; this will give us better control over display output withour requiring parsing.
- Tying long term processes to their code block for output management, so users can stand up things like servers in one block, then query them in another, with separate console.log outputs.

BTCR Hackathon

Our next big focus is developing some curriculum, especially for BTCR, starting at the 2019 BTCR Hackathon in August and likely continuing into the next Rebooting the Web of Trust event in September.

OUR STARTING POINT

The following notes are from our initial draft, with a few comments interjected. We started out with a reasonabely scoped ambition, got the skeleton working in Barcelona, then took a couple months to round out the app and publish a Minimal Viable Product.

Requirements

- File format is markdown
- In the app, the markdown is unmodifiable, but rendered
- If viewed outside the app, files will be sensibly renderable (it is compatible markdown)
- Custom code section(s) render as an “interactive” editor
- Code in editor sections is executable, with console output
- Focus on teaching/testing node.js modules
- JavaScript execution context
- Standalone executable app
- Vanilla JavaScript
- Creators and users may install additional node modules, which will be installed in the file’s directory under node_modules (as typical with node.js apps). Installation is outside the app. However, such modules may be require()'d within a file. Node modules that conflict with modules already in use by Satyrn must be avoided.

Nice to have features

- Teacher mode
- REPL
- Module info to help users avoid conflicts

Out of scope

- Browser libraries (Vue, React, Angular, etc.)
- Native bindings
- Non-javascript languages
- External editor
- Minimal external dependencies

NOTE: The only real shift from this list is the acceptance of "Teacher mode" in the sense of allowing users to edit the markdown. Initially, Joe didn't want this--the driver was the student not the teacher--but it immediately became apparent that *we* wanted to use an editor during development, and that since we had already committed

to integrating the ACE editor, it was an easy lift.

Of course, this opens a slew of feature-creeping feature requirements, like how we manage window sizes and integration between the interactively edited JavaScript and the initial markdown. "Easy lift" is a red flag for smart developers and project managers everywhere. What seams "easy" often has hidden complexity and support costs.

So, we don't have the most awesome markdown editor out there, but at least if you're developing curriculum, you can use the tool directly to edit without requiring yet another app for editing markdown.

Electron Resources

https://github.com/sindresorhus/awesome-electron

Components

- Electron
- Electron template https://github.com/sindresorhus/awesome-electron#boilerplates
- Markdown renderer https://github.com/showdownjs/showdown
- Inline Editor Ace
- Console (output)
- Editor control buttons

Additional Credits

Lead Author: Joe Andrieu (joe@legreq.com)

Authors: Eric Welton (eric@korsimoro.com), Will Abramson (will.abramson@napier.ac.uk), and Ganesh Annan (gannan@digitalbazaar.com)

Sample APA Citation:

Andrieu, J., Welton, E., Abramson, W., and Annan, G. (2019). Satyrn: A Markdown-based JavaScript Sandbox . *Rebooting the Web of Trust VIII*. Retrieved from https://github.com/WebOfTrustInfo/rwot8-barcelona/blob/master/final-documents/satyrn.pdf.

About Rebooting the Web of Trust

This paper was produced as part of the Rebooting the Web of Trust VIII design workshop. On March 1st to 3rd,

2019, over 80 tech visionaries came together in Barcelona, Spain to talk about the future of decentralized trust on the internet with the goal of writing at least 5 white papers and specs. This is one of them.

RWOT Board of Directors: Christopher Allen, Joe Andrieu, Kim Hamilton Duffy

Silver Sponsors: Caelum Labs, Digital Contract Design, Generalitat de Catalunya, Protocol Labs, Venn Agency

Additional Sponsors: Validated ID, PTB Ventures

Community Sponsors: Blockchain Commons, Digital Bazaar, In Turn Information Management Consulting, Learning Machine, Legendary Requirements

Workshop Credits: Christopher Allen (Founder), Joe Andrieu (Producer and Facilitator), Shannon Appelcline (Editor-in-chief), and Carlotta Cataldi (Graphical Recorder)

Thanks to our other contributors and sponsors!

What's Next?

The design workshop and this paper are just starting points for Rebooting the Web of Trust. If you have any comments, thoughts, or expansions on this paper, please post them to our GitHub issues page:

https://github.com/WebOfTrustInfo/rwot8/issues

The ninth Rebooting the Web of Trust design workshop is scheduled for September 3rd-6th 2019 in Prague, The Czech Republic. If you'd like to be involved or would like to help sponsor the event, email:

rwot-leadership@googlegroups.com
