# Colbyn Wadman

I specialize in building custom tools and utilities — but I've tackled a wide range of problems beyond that

APPS TOOLS AUTOMATION DATA/AI PIPELINES SOLO BUILDER TECHNICALLY FLUENT WRITER

hello@colbyn.com colbyn.com 801-367-4487

## About Me

Self-taught systems builder with a background in functional programming and years spent crafting custom parsers, transforming ASTs, and working from first principles. My path has been unconventional, shaped by independent technical work that prioritized depth over conformity. I work best where off-the-shelf tools break down — especially in Rust, Swift, and domains requiring tight control.

## Core Competencies

- **Compilers & Tooling**
  - Domain-specific parsers, macro-based HTML compilers, prompt-driven content pipelines
  - Dataset generation from EPUBs with retry logic and schema validation
- **Systems & Rendering**
  - Rust and Swift CLI tools, TextKit2 engines, custom layout systems
  - UI prototyping, including a React-style renderer in Rust
- **Data & Reverse Engineering**
  - Modeling structured/unstructured data at scale
  - Dissecting obfuscated APIs (e.g. Google Maps payloads)

## Selected Projects

- **Subscript:** A Swift note-taking app with typesetting, freeform sketches and offline-compiled HTML exports.
- **WebCompiler:** Rust-based static site generator with modular transforms and LLM scaffolding. Replaces CMS workflows.
- **Dataset Compiler:** EPUB-to-custom dataset pipeline using LLMs, with deterministic logic and app specific code-gen.
- **Google Maps Decoder:** Reverse-engineered map payloads to build a structured lead-gen engine.

## Tooling Philosophy

I build tools I can reason about end-to-end. Whether it's a compiler or a templating DSL, I care about determinism, clarity, and making brittle problems tractable by reshaping them at the tooling layer.

## Fit

I'm drawn to roles where conventional tools fall short — places that need structural

thinkers and builders who prefer clarity over convenience.

# History

## 2015

### Galileo Processing Data Center Tech

Basic Networking | Datacenter Misc. | Communication

## 2016

### Uplynk/VDMS (Verizon Digital Media Services) Jr. Developer

Jr. developer in the QA team lead by Asiel Brumfield. I'm not entirely sure when I began working. I left to pursue a startup with my uncle in encrypted images.

Python | Q/A | video streaming

## 2018

### Secret Startup Experiment

Video | DRM

### SubSys/Compiler

**GitHub:**
https://github.com/SubSys/Compiler

A cross-language compiler pipeline that translated **Elm** to **Rust**, implemented in **Haskell**. Built to investigate whether a statically typed frontend could target systems-level output while preserving semantics and safety guarantees.

The project leveraged Haskell's type system not just to implement compiler stages, but to assert compile-time properties of the translation itself — from type mapping to memory safety behavior. The result was a proof-of-concept pipeline that tested the edges of language interoperability.

#### Key Features

- Elm → Rust transpilation with semantic fidelity.
- Typed IR modeled and verified in Haskell.
- Explores correctness as a first-class design constraint.

Compiler Engineering | Elm | Rust | Haskell | Type Systems | Interoperability | Open Source | Experimental

### colbyn/commands

**GitHub:**
https://github.com/colbyn/commands

A minimal DSL for CLI scripting — inspired by Bash, but redesigned around indentation and composability. This tool replaces brittle shell scripts with readable, structured code for automation-heavy workflows like **AWS deployments**.

The syntax mimics the clarity of Python, but compiles to raw shell — enabling devs to write clean scripts without giving up full control. Think `Make` meets `just`, with extras.

#### Key Features

- Indentation-sensitive syntax for nested blocks.

- Function embedding, aliasing, and multiline strings.
- Supports structured parsing, silent execution, and CLI grouping.

CLI Tools | Automation | DSL | Bash-Inspired | AWS | Open Source

# 2019

## Imager

imager.io | github.com/imager-io

**One of the best image optimization tools on the market.**

A modular, open-source image optimization platform designed to outperform commercial tools like kraken.io — with no SaaS dependencies. At its core: a **Rust-based encoder** guided by **machine-learned visual metrics**, wrapped in a clean CLI and consumable through **Node.js bindings**.

The toolchain included native bindings (`webp-dev-rs`, `ffmpeg-dev-rs`, `x264-dev`, `vmaf-sys`) and a Rust-to-JS bridge via `imager-io-js`. Benchmarks showed over **90% file size reduction** compared to popular SaaS optimizers — without perceptual quality loss.

### Key Features

- Rust-based encoder with ML-guided format selection (WebP, JPEG, x264).
- `imager-bench` benchmark suite beating kraken.io and jpegmini.
- Zero-dependency Node.js pipeline with native bindings for real-time processing.
- Full FFmpeg and VMAF support via custom Rust FFI crates.

Image Processing | Machine Learning | Rust | Node.js | WebP | FFmpeg | x264 | VMAF | Performance Optimization | Open Source

## colbyn/web-images-js

**GitHub:**
https://github.com/colbyn/web-images-js

A zero-dependency image pipeline for Node.js — built entirely in Rust and embedded directly in JS workflows. Offers full control over memory, binary size, and output quality without external tools or native runtime bindings.

### Key Features

- High-performance image loading and transformation in native Rust.
- Statically linked builds for reproducibility and safety.
- Optimized for deployment in serverless and JAMstack environments.

Rust | Node.js | Web Development | Performance Optimization | Image Processing | Dependency-Free | Open Source

## colbyn/subscript-old

**GitHub:**
https://github.com/colbyn/subscript-old

I renamed this project to `subscript-old` after I decide to call my note taking tools 'subscript' which seemed more fitting because it dealt with typesetting. **There's actually some pretty cool ideas in here. Abandoning this project is a major regret of my life.**

A data-driven, Rust-native frontend library designed for backend developers — Subscript rethinks web UI as infrastructure. Built for

those who want to author views, state, and styles entirely in Rust, with zero XML, full CSSOM control, and expressive compile-time macros.

Ideal for backend developers treating the frontend as just another client — including full inline CSS support (media queries, keyframes, pseudo elements) via a "selector-less functionalized CSS" model. Built-in routing, component messaging, and versioned view syntax enable deeply structured applications without frontend boilerplate.

### Feature Highlights

- Inline Rust macros for **media queries**, **keyframes**, and **pseudo classes**.
- **Pattern-matching-like URL parsing** using `parse_url!`, with totality checks and type-safe bindings.
- **Component messaging and subscriptions**, including typed broadcasting and routing by component type.
- Versioned view macros like `v1!` for ergonomic and incremental UI design.
- Zero-runtime-diffing model closer to Incremental DOM than virtual DOMs.

### Example Syntax

```
parse_url! {
  [] => {
    Page::Homepage
  },
  ["account", user_id: Uuid] => {
    Page::AccountUser {
      id: user_id
    }
  },
  _ => Page::NotFound
};
```

```
v1! {
  display: "flex";
  button !{
    event.click[] => {
      move || Msg::Increment
    };
    "Increment";
  }
}
```

### Tags

Rust · Frontend · WebAssembly · Macro Systems · Routing · UI Framework · CSSOM · Open Source

**Impact:** Inventive take on frontend architecture through Rust. Encourages architectural unification between client and server codebases, with precise control over styling and rendering.

## Milestone

**At the end of 2019 I raked up an impressive 1,323 contributions on GitHub** that placed me among the platform's most active developers.

To put this in perspective, data from a 2016–2017 GitHub ranking shows the 256th most active user recorded 1,322 contributions—my activity surpassed this benchmark, placing me among the platform's top contributors. While my work wasn't featured in such lists due to follower-based filtering, my focus remains on creating impactful projects like imager-io/imager (688 stars) and SuperSwiftMarkup/SuperSwiftMarkdownPrototype (58 stars). These contributions, driven by expertise in Rust and Swift, reflect my commitment to building tools that empower the developer community.

Today, my GitHub profile is a living portfolio of grit and expertise.

Sources

**Top 2017 GitHub Contributors:**
https://web.archive.org/web/20200415010317/https://gist.github.com/paulmillr/2657075/

**Colbyn's GitHub Profile (stats at the bottom):**
https://github.com/colbyn?tab=overview&from=2019-12-01&to=2019-12-31

# 2020

## Began College

Notable Comments

> *You are such a profound writer and thinker. It has been my privilege to be your instructor of record. One day I will say, I had him in my English class. I have such high hopes for you! Go conquer your world! You're awesome.*
> *—Dr. Jim Birrell*

LaTeX

## UVU CS Grader - Computer Science

In my first semester at UVU I took CS1400 by professor Bianca Ruiz, who then offered me a grading position at the end of the semester. I enjoyed this job and wish I stayed (I took trigonometry and calculus concurrently and was expecting subsequent semesters to be just as difficult).

The instructor said my code looked “beautiful” before I signed up as a grader.

Python Grading Teaching Communication

## The HTML Toolchain

**GitHub:**
https://github.com/subscript-publishing/subscript-html

A custom publishing system that reimagines LaTeX for the web. Designed to turn structured STEM notes into responsive, navigable documents — blending handwritten diagrams, inline math, and semantic HTML in one toolchain.

Unlike traditional LaTeX-to-PDF pipelines, this system outputs fully interactive HTML with live navigation, mobile-ready layout, and auto-generated metadata. Built for note-taking first, publishing second.

### Key Features

- Autogenerated tables of contents and section links.
- Hybrid support for typeset and hand-drawn inputs.
- Responsive layout tuned for cross-device academic reading.

Academic Publishing LaTeX-Inspired HTML JavaScript Responsive Design Note-Taking Open Source

## My Beautiful Math Notes

**school-notes-spring-2020:**
https://colbyn.github.io/school-notes-spring-2020/

A real-world testbed for Subscript’s HTML toolchain — compiling a semester of math coursework into an interactive, web-native format. Features richly formatted equations, diagrams, and dynamic TOC navigation across topics.

The project illustrates how digital notes can preserve mathematical rigor without compromising on usability or design.

### Key Features

- Clean math typesetting with inline and block notation.
- Linked headings and deep navigation for modular study.
- Designed for screen reading, not PDF output.

**Final Culmination (pretty slick):**
https://colbyn.github.io/all-school-notes/
Although looking back I prefer the old style:
https://colbyn.github.io/school-notes-fall-2022/

Mathematics · Academic Publishing · Subscript · HTML · JavaScript · Interactive Content · Open Source

## Subscript (iPad Edition + Authoring Tools)

**GitHub:**
https://github.com/subscript-publishing/subscript

**Content publishing VIA Web-Technologies!**

Supports freeform and typed markup content in one medium.

### Key Features

- Seamlessly intermix markup with hand drawn content VIA the Subscript Freeform Tools (iPad only).
- Redesigned macOS and iOS editor interfaces from the ground up.
- Native rendering pipeline tuned for complex typeset math, and freeform sketch inputs.

Academic Publishing · LaTeX-Inspired Markup Language · Drawing Tools · JavaScript · HTML · iPad · UI/UX Design · Swift · macOS · iOS · Open Source

# 2021

## AMI Uploader

**GitHub:**
https://github.com/colbyn/ami-uploader

A lightweight Rust CLI designed to automate the upload of **LinuxKit-generated AMIs** to AWS via S3. It streamlines the gap between image generation and cloud deployment — replacing manual S3 uploads and AMI registration with a single, scriptable command.

Built for reliability in CI pipelines, the tool supports alternate credential injection, customizable AMI naming, and metadata tagging. Designed for infrastructure engineers managing reproducible builds.

### Key Features

- One-command upload from local disk to registered AMI.
- Supports name overrides, alternate AWS keys, and region targeting.
- Integrates cleanly into LinuxKit or DevOps build pipelines.

CLI Tools · Rust · AWS · S3 · AMI · LinuxKit · DevOps · Open Source

## 2023

### punk-lang

**GitHub:** https://github.com/colbyn/punk-lang

A deliberately minimal markup language with LaTeX-inspired syntax and depth-sensitive highlighting — designed to test how little structure is needed to express complex educational content clearly.

The language focuses on parseability and pedagogical clarity: every visual highlight is tied to syntactic depth, making nested logic or derivations easier to follow for learners. Built as a sketchpad for better markup ergonomics.

#### Key Features

- Clean, expressive syntax for STEM writing and teaching.
- Depth-aware syntax highlighting for conceptual layering.
- Structured output with minimal rules — optimized for parsing.

Language Design · Educational Content · LaTeX-Inspired · Syntax Highlighting · Markup Language · Experimental · Open Source

### Subscript Freeform Note-Taking App (iOS/macOS)

A handwritten, vector-based note-taking app built from scratch to prioritize **human expression, long-term readability, and semantic structure**—a radical departure from PDF-constrained or AI-assisted systems. Originally released on the App Store, I later withdrew it to re-architect the data model and rethink structural primitives.

The app uses a custom model space for device-independent rendering, handwritten stroke capture with velocity-aware smoothing (via a Swift port of `perfect-freehand`), and a semantically driven outline system (H1–H6) for navigation and TOC generation.

Subscript reflects a broader philosophy: **separation of content and presentation**, inspired by LaTeX, combined with the authenticity of freeform input—a medium resistant to AI mimicry.

▶ Intro: My Note-Taking App & Why It Matters in the Age of Bots (YouTube)

Swift · Native iOS/macOS · Unsafe Rust (FFI) · Vector Graphics · Digital Ink · Document Compiler

## 2024

### Parser & Tree Visualization Toolkit

A cohesive suite of language tooling libraries for **Swift** and **Rust**, focused on parser combinator frameworks and human-friendly visualization of abstract syntax trees (ASTs) and nested structures. Designed to support debugging, inspection, and language toolchain development in functional and systems programming contexts.

Core components include `MonadoParser`[†], a monadic parser combinator framework for Swift; `pretty-tree-rs`[‡], a minimal Rust library for rendering cleanly formatted hierarchical data; and `SwiftPrettyTree`[§], a Swift-native port for readable tree inspection in IDEs or CLI workflows.

### Key Features

- Composable, lossless parsers with precise position tracking (`MonadoParser`).
- Compact, dependency-free tree renderers for ASTs and nested structures.
- Interoperable debugging tools for language tooling, REPLs, and test harnesses.

Parser Combinators | Swift | Rust | Functional Programming | AST | Debugging Tools | Compiler Engineering | Open Source

† github.com/colbyn/MonadoParser

‡ github.com/colbyn/pretty-tree-rs

§ github.com/colbyn/SwiftPrettyTree

## 3in1Spanish Your go-to Spanish Dictionary, Phrasebook & Flash Cards

A Bilingual Spanish Dictionary, Phrasebook & Flash Card App.

Required a very sophisticated dataset generator that I called the compiler generator. See my YouTube Video for details with commentary.

**'How I autogenerate massive (dictionary) datasets with ChatGPT/LLMs and why this matters':** https://youtu.be/nofJLw51xSk?si=WrOwCT7WA6_VTBrO

Prompt Engineering | Data Pipelines | AI Centric Pipelines | Dataset Generation

# 2025

## SuperSwiftMarkdownPrototype

**GitHub:** https://github.com/SuperSwiftMarkup/SuperSwiftMarkdownPrototype

A Swift-native markdown editor prototype for iOS and macOS — built to test high-performance rendering and editing using **TextKit 2**. Targeted GitHub-Flavored Markdown support with a focus on modern editing ergonomics.

The prototype explored interaction models like multi-cursor editing, block-level selection, and native table editing — all backed by a custom layout engine to push the limits of TextKit's rendering fidelity.

### Key Features

- Multi-cursor + inline block selection for rapid editing workflows.
- Native table editing with contextual cursor handling.
- Built with TextKit 2 for speed and layout precision.

Swift | Markdown Rendering | iOS | macOS | TextKit 2 | Editing Tools | Open Source | Prototype