hotdocX & jsCoq — A Platform for Interactive, AI-Augmented, and Monetizable Coq Experiences

The hotdocX Authors

Abstract

The dissemination of and interaction with Coq developments remains a significant challenge, often confined to static documents or code repositories that lack dynamism. We present an experience report on integrating Coq into hotdocX, a novel web platform that transforms documents into interactive, AI-augmented applications. By leveraging a `jsCoq` template within the hotdocX ecosystem, we create a new paradigm for "live" formal artifacts. This enables not only interactive educational tutorials and "papers-with-proofs" but also a creator economy for formal methods content. Furthermore, we explore the profound synergy between the browser-native jsCoq and Emdash, hotdocX's own dependently-typed logical framework. This opens a tangible pathway towards a browser-editable, extensible Coq-light kernel, presenting new opportunities for research in tactic development, meta-theory, and education.

1. Introduction

The Coq proof assistant is a cornerstone of formal verification and computer-aided proof. However, the fruits of this labor—the formalizations themselves—are often difficult to share and engage with beyond the community of expert users. They typically exist as code repositories or are flattened into static PDFs, losing the rich, interactive nature of the original proof development. Past initiatives like Yves Bertot's "Coq Exchange" [11] and the "Coq Platform Docs" [12] have highlighted the community's desire for more dynamic and accessible platforms for sharing Coq content.

This paper reports on our work to address this challenge by interfacing Coq with **hotdocX** [3], an AI-powered social events marketplace designed to transform documents into interactive web applications. hotdocX uses an in-browser IDE (Sandpack) to render "events" which are live, runnable applications generated from user-provided documents (PDFs, LaTeX, text) and AI prompts. By creating a `jsCoq` template for this platform, we enable any Coq script to become a first-class, interactive, and shareable hotdocX event.

Our contribution is threefold:

1. An **experience report** on the practical aspects of interfacing with `jsCoq` in a modern web sandbox environment, including technical hurdles and their solutions.
2. A demonstration of a new **application platform** for Coq that supports novel use cases in education and research dissemination, including a creator economy via virtual currency.
3. A **vision for future work** where the browser-based nature of `jsCoq` and hotdocX's native logical framework, Emdash [4], converge towards a new kind of interactive, browser-native proof assistant kernel.

2. The hotdocX Platform

hotdocX is a serverless web application built on Convex and React. Its core concept is the **AI Template**. A user selects a template, provides source documents (e.g., from a native upload, SharePoint, GitHub, or arXiv), and adds a natural language prompt. The hotdocX backend, powered by Google's Gemini AI, orchestrates a content generation pipeline to produce a set of files for an interactive Sandpack application.

The template system is highly flexible. For example, the `arrowgram-paged` template [7, 10] can take a description of a commutative diagram, generate a JSON specification for it, and render it as an SVG within a paginated, two-column PDF-style document complete with KaTeX-rendered mathematics and Mermaid conceptual diagrams. The `emdash` template [8, 9] allows users to interactively experiment with a novel, dependently-typed functorial programming language directly in the browser. This architecture provides a rich context for embedding formal methods tooling.

3. Interfacing with Coq: An Experience Report

Our goal was to create a `jsCoq` template to make Coq a first-class citizen of the hotdocX ecosystem. The implementation, found in `convex/lib/templates/templates_jsCoq.ts`, uses the jsCoq CDN to load the environment and then fetches a user-provided Coq script (e.g., `/public/index.v`) to initialize the session. The result is a live, interactive Coq proof session, embedded within a hotdocX event [6].

A significant challenge arose during this integration. We discovered that versions of `jsCoq` after `0.15.1` (specifically, the move to ES Modules in `0.16.0`) introduced a "Cannot use

'import.meta' outside a module" error within the Sandpack environment. Consequently, our current implementation is pinned to `jscoq@0.15.1`. While this works, it highlights a practical friction point for developers seeking to embed `jsCoq` in modern sandboxed web applications. Overcoming this would be a valuable contribution, enabling the broader community to more easily build tools on the latest `jsCoq` versions.

This integration immediately creates powerful new workflows:

- **Education:** An instructor can create a hotdocX event containing a lecture (as a PDF), a set of Coq exercises (`exercises.v`), and an AI prompt like "Create a tutorial from the lecture and load the exercises into the jsCoq environment." Students can then interact with the tutorial and solve exercises entirely in the browser. Using hotdocX's creator economy features, the instructor could even place the solutions behind a "pay-to-view" paywall using virtual coins.
- **Research:** A researcher can publish a paper on arXiv and simultaneously create a hotdocX event that links to it. The event could contain a `jsCoq` session pre-loaded with the key definitions and lemmas from the paper, allowing readers to experiment with the formalization directly. This moves beyond "papers-with-code" to "papers-as-apps".

4. Future Work: Towards a Browser-Native Coq-light Kernel

The most exciting prospect arises from the synergy between `jsCoq` and **Emdash** [4], the native logical framework of hotdocX. Emdash is a dependently typed language inspired by Kosta Dosen's functorial programming [1, 2], implemented in TypeScript and formally specified in a Lambdapi dialect [5]. It features a bidirectional type checker, unification-based hole solving, and a novel "functorial elaboration" mechanism that verifies coherence laws definitionally.

Both Emdash and `jsCoq` are designed to run in the browser. Emdash's kernel is written in TypeScript, making it natively inspectable and extensible in a JavaScript environment. This presents a unique opportunity for convergence. We propose a research program focused on creating a hybrid **Emdash/Coq-light kernel**.

The path involves systematically augmenting the Emdash kernel, which is already a λΠ-calculus modulo theory, with the constants, reduction rules, and inductive types of the Calculus of Inductive Constructions (CIC). Because Emdash's definitional equality is already extensible via user-supplied rules, this process can be done incrementally.

The implications of a browser-native, hackable Coq-light kernel are significant:

- **Education:** It would provide an unparalleled tool for teaching the theory of CIC. Students could inspect, modify, and experiment with the kernel's rewrite rules and term structures directly in the browser, without a complex local build environment.
- **Research on Coq:** It would create a rapid prototyping environment for new language or tactic features. A researcher could implement a new tactic or a modification to the conversion algorithm in TypeScript and immediately test it in a live Coq-like environment, dramatically shortening the feedback loop.
- **AI-Assisted Formalization:** An AI agent could more easily interact with a native JavaScript/TypeScript kernel than by parsing the string output of a compiled binary. This could lead to more sophisticated AI-driven tools for proof synthesis and tactic generation.

5. Conclusion

By interfacing `jsCoq` with the `hotdocx` platform, we have developed a novel, practical, and powerful new way to share, teach, and interact with Coq formalizations. Our experience provides a concrete use case that highlights both the promise and the practical hurdles of embedding `jsCoq` in modern web applications. Looking forward, the co-location of `jsCoq` and the `emdash` logical framework within the same browser-native ecosystem offers a compelling roadmap towards a new generation of interactive, extensible, and accessible theorem proving tools.

References

[1] Dosen, K., & Petrić, Z. (1999). *Cut-Elimination in Categories*.
[2] Dosen, K., & Petrić, Z. (2004). *Proof-Theoretical Coherence*. KCL Publications.
[3] hotdocX Project. https://hotdocx.github.io
[4] hotdocX Team. (2024). *Emdash: A Dependently Typed Logical Framework*. https://github.com/hotdocx/emdash
[5] 1337777. (2024). *Emdash Specification in Lambdapi*. https://github.com/1337777/cartier/blob/master/cartierSolution18.lp
[6] hotdocX Example. (2024). *jsCoq Interactive Session*. https://hotdocx.github.io/#/hdx/25191CHRI43000
[7] hotdocX Example. (2024). *Arrowgram AI Template*. https://hotdocx.github.io/#/hdx/25188CHRI26000
[8] hotdocX Example. (2024). *Emdash Re-formattable Example*.

https://hotdocx.github.io/#/hdx/25188CHRI25004
[9] hotdocX Example. (2024). *Emdash Experiment-able Example*. https://hotdocx.github.io/#/hdx/25188CHRI27000
[10] Arrowgram Project. https://github.com/hotdocx/arrowgram
[11] Bertot, Y. (2019). *Coq Exchange*. https://project.inria.fr/coqexchange/news/
[12] Lamiaux, T. (2025). *Coq Platform Docs*. https://coq.inria.fr/docs/platform-docs