PART 1/2

EMDASH

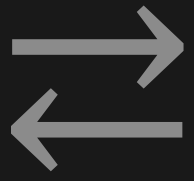
A DEPENDENTLY TYPED LOGICAL FRAMEWORK FOR COMPUTATIONAL SYNTHETIC CATEGORY THEORY

Based on the paper by Author

THE PROBLEM: A CHASM

INFORMAL MATHEMATICS

- Fluid & Intuitive
- Structural Reasoning



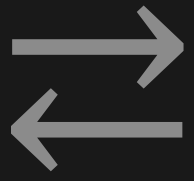
FORMAL SYSTEMS

- Rigid & Explicit
- Foundational Encoding

THE PROBLEM: A CHASM

INFORMAL MATHEMATICS

- Fluid & Intuitive
- Structural Reasoning
- Thrives on Abstraction



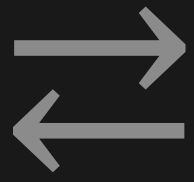
FORMAL SYSTEMS

- Rigid & Explicit
- Foundational Encoding

THE PROBLEM: A CHASM

INFORMAL MATHEMATICS

- Fluid & Intuitive
- Structural Reasoning
- Thrives on Abstraction



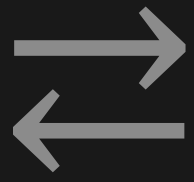
FORMAL SYSTEMS

- Rigid & Explicit
- Foundational Encoding
- Structure is a *proven property*, not an *intrinsic quality*

THE PROBLEM: A CHASM

INFORMAL MATHEMATICS

- Fluid & Intuitive
- Structural Reasoning
- Thrives on Abstraction



FORMAL SYSTEMS

- Rigid & Explicit
- Foundational Encoding
- Structure is a *proven property*, not an *intrinsic quality*

CAN WE BRIDGE THIS GAP?

A DIFFERENT VISION: FUNCTORIAL PROGRAMMING

> *"The core entities of a mathematical domain—such as categories and functors—are not merely encoded but are themselves the primitive building blocks of the formal language."*

Inspired by Kosta Dosen [1, 2]

A DIFFERENT VISION: FUNCTORIAL PROGRAMMING

> *"The core entities of a mathematical domain—such as categories and functors—are not merely encoded but are themselves the primitive building blocks of the formal language."*

Inspired by Kosta Dosen [1, 2]

Proofs of structural integrity should be **computations**, not separate objects.

INTRODUCING: EMDASH

A novel dependently typed logical framework that implements these functorial principles.

INTRODUCING: EMDASH

A novel dependently typed logical framework that implements these functorial principles.

- **Synthetic:** Categorical notions (`Cat`, `Obj`, `Functor`) are primitives.

INTRODUCING: EMDASH

A novel dependently typed logical framework that implements these functorial principles.

- **Synthetic:** Categorical notions (`Cat`, `Obj`, `Functor`) are primitives.
- **Computational:** Coherence laws are checked via definitional equality (βδι-reduction).

INTRODUCING: EMDASH

A novel dependently typed logical framework that implements these functorial principles.

- **Synthetic:** Categorical notions (`Cat`, `Obj`, `Functor`) are primitives.
- **Computational:** Coherence laws are checked via definitional equality (βδι-reduction).
- **Specification-Driven:** Implemented in TypeScript, formally specified in Lambdapi.

INTRODUCING: EMDASH

A novel dependently typed logical framework that implements these functorial principles.

- **Synthetic:** Categorical notions (`Cat`, `Obj`, `Functor`) are primitives.
- **Computational:** Coherence laws are checked via definitional equality (βδι-reduction).
- **Specification-Driven:** Implemented in TypeScript, formally specified in Lambdapi.
- **Practical:** The formal engine for **hotdocX**, an AI-assisted formalization platform.

CORE ARCHITECTURE

TYPE THEORY CORE

- λΠ-Calculus Modulo Theory
- HOAS for binders
- Bidirectional Type Checker
- Unification & Hole Solving

SYNTHETIC PRIMITIVES

- `Cat`, `Obj`, `Hom`
- `Functor`, `Transf`
- Yoneda (`hom_cov`)
- User-defined rewrite rules

CORE ARCHITECTURE

TYPE THEORY CORE

- λΠ-Calculus Modulo Theory
- HOAS for binders
- Bidirectional Type Checker
- Unification & Hole Solving

SYNTHETIC PRIMITIVES

- `Cat`, `Obj`, `Hom`
- `Functor`, `Transf`
- Yoneda (`hom_cov`)
- User-defined rewrite rules

The system is designed to be extensible, allowing users to add their own definitions, rewrite rules, and unification hints.

THE KILLER FEATURE: FUNCTORIAL ELABORATION

THE TRADITIONAL WAY: DATA + PROOF

In Coq/Agda/Lean, you define a functor by providing:

1. **Data:** An object map `$F_0$` and a morphism map `$F_1$`.
2. **Proof:** A separate proof term `F_preserves_comp` demonstrating that `$F_1(g \circ f) = F_1(g) \circ F_1(f)$`.

THE KILLER FEATURE: FUNCTORIAL ELABORATION

THE TRADITIONAL WAY: DATA + PROOF

In Coq/Agda/Lean, you define a functor by providing:

1. **Data:** An object map `$F_0$` and a morphism map `$F_1$`.
2. **Proof:** A separate proof term `F_preserves_comp` demonstrating that `$F_1(g \circ f) = F_1(g) \circ F_1(f)$`.

The structure and its laws are separate entities.

THE EMDASH WAY: DEFINITIONAL CHECK

In Emdash, defining a functor uses a single primitive:

```
MkFunctorTerm(C, D, fmap0, fmap1)
```

The elaboration engine itself **definitionally verifies** the coherence laws during type checking.

THE EMDASH WAY: DEFINITIONAL CHECK

In Emdash, defining a functor uses a single primitive:

```
MkFunctorTerm(C, D, fmap0, fmap1)
```

The elaboration engine itself **definitionally verifies** the coherence laws during type checking.

Coherence is a check, not a proof.

HOW IT WORKS: THE `MKFUNCTORTERM` PRIMITIVE

1. **Check Components:** Verify types of `C`, `D`, `fmap0`, `fmap1`.

HOW IT WORKS: THE `MKFUNCTORTERM` PRIMITIVE

1. **Check Components:** Verify types of `C`, `D`, `fmap0`, `fmap1`.
2. **Stage the Law:** Create a hypothetical context with generic morphisms `f` and `g`.

HOW IT WORKS: THE `MKFUNCTORTERM` PRIMITIVE

1. **Check Components:** Verify types of `C`, `D`, `fmap0`, `fmap1`.
2. **Stage the Law:** Create a hypothetical context with generic morphisms `f` and `g`.
3. **Construct Terms:**
    - `LHS := fmap1 (g ∘ f)`
    - `RHS := (fmap1 g) ∘ (fmap1 f)`

HOW IT WORKS: THE `MKFUNCTORTERM` PRIMITIVE

1. **Check Components:** Verify types of `C`, `D`, `fmap0`, `fmap1`.
2. **Stage the Law:** Create a hypothetical context with generic morphisms `f` and `g`.
3. **Construct Terms:**
    - `LHS := fmap1 (g ∘ f)`
    - `RHS := (fmap1 g) ∘ (fmap1 f)`
4. **Compute & Compare:**
    - Normalize `LHS` and `RHS`.
    - Check if `normalize(LHS) ≡ normalize(RHS)`.

HOW IT WORKS: THE `MKFUNCTORTERM` PRIMITIVE

1. **Check Components:** Verify types of `C`, `D`, `fmap0`, `fmap1`.
2. **Stage the Law:** Create a hypothetical context with generic morphisms `f` and `g`.
3. **Construct Terms:**
    - `LHS := fmap1 (g ∘ f)`
    - `RHS := (fmap1 g) ∘ (fmap1 f)`
4. **Compute & Compare:**
    - Normalize `LHS` and `RHS`.
    - Check if `normalize(LHS) ≡ normalize(RHS)`.
5. **The Verdict:**
    - ✅ **Success:** Elaboration succeeds. The term is accepted.
    - ❌ **Failure:** Throw a `CoherenceError` with the non-equal normal forms.

BLUEPRINT FOR SYNTHESIS: THE LAMBDAPI SPECIFICATION

The implementation is not ad-hoc; it's guided by a formal, executable specification.

```
constant symbol Cat : TYPE;
injective symbol Obj : Cat → TYPE;
injective symbol Hom : Π [A : Cat] (X: Obj A) (Y: Obj A), TYPE;

// ...

constant symbol Functor_cat : Π(A : Cat), Π(B : Cat), Cat;

// Morphisms in the functor category ARE natural transformations
rule @Hom (Functor_cat _ _) $F $G ↪ Transf $F $G;

// Functoriality is a computational rule
rule compose_morph (@fapp1 _ _ $F _ _ $a) (@fapp1 _ _ $F _ _ $a')
  ↪ fapp1 $F (compose_morph $a $a');
```

BLUEPRINT FOR SYNTHESIS: THE LAMBDAPI SPECIFICATION

The implementation is not ad-hoc; it's guided by a formal, executable specification.

```
constant symbol Cat : TYPE;
injective symbol Obj : Cat → TYPE;
injective symbol Hom : Π [A : Cat] (X: Obj A) (Y: Obj A), TYPE;

// ...

constant symbol Functor_cat : Π(A : Cat), Π(B : Cat), Cat;

// Morphisms in the functor category ARE natural transformations
rule @Hom (Functor_cat _ _) $F $G ↪ Transf $F $G;

// Functoriality is a computational rule
rule compose_morph (@fapp1 _ _ $F _ _ $a) (@fapp1 _ _ $F _ _ $a')
  ↪ fapp1 $F (compose_morph $a $a');
```

This declarative style makes coherence laws part of the system's core computational behavior.

INTERACTIVE THEOREM PROVING

Emdash isn't just for defining; it's for proving.

A proof-in-progress is a term with unsolved goals, called **Holes** (`?h0`).

```
Goal ?g0: ⊢ Π (n : Nat). Nat
```

Users construct proofs by refining these holes using tactics.

EXAMPLE PROOF SESSION

Goal: Prove `id : Nat -> Nat`

```
// 1. State Goal
defineGlobal("id_nat_proof", Pi("n", Expl, Nat, _ => Nat), Hole("?g0"))
// > Goal ?g0: ⊢ Π (n : Nat). Nat

// 2. Introduce hypothesis
intro(Var("id_nat_proof"), "?g0", "n")
// > Hole ?g0 solved with `λ n. ?g1`
// > New Goal ?g1: n : Nat ⊢ Nat

// 3. Solve goal with exact term
exact(Var("id_nat_proof"), "?g1", Var("n"))
// > Hole ?g1 solved with `n`

// 4. Proof Complete!
// Final term: λ (n : Nat). n
```

VALIDATION & TESTING

Emdash is a working TypeScript implementation, verified by a comprehensive test suite.

- ✅ Inductive types (Nat, List, Vec) & dependent functions.
- ✅ Higher-order unification & pattern matching.
- ✅ **Crucially:** Tests confirm that `MkFunctorTerm` accepts valid functors and throws `CoherenceError` for invalid ones.
- ✅ Full interactive proof sessions are tested from start to finish.

VALIDATION & TESTING

Emdash is a working TypeScript implementation, verified by a comprehensive test suite.

- ✅ Inductive types (Nat, List, Vec) & dependent functions.
- ✅ Higher-order unification & pattern matching.
- ✅ **Crucially:** Tests confirm that `MkFunctorTerm` accepts valid functors and throws `CoherenceError` for invalid ones.
- ✅ Full interactive proof sessions are tested from start to finish.

The system behaves as specified.

THE GRAND VISION: HOTDOCX

Emdash is the formal engine for **hotdocX**, a web platform for AI-assisted formalization.

Goal: Transform mathematical documents into executable, verifiable, and interactive formal content.

THE GRAND VISION: HOTDOCX

Emdash is the formal engine for **hotdocX**, a web platform for AI-assisted formalization.

Goal: Transform mathematical documents into executable, verifiable, and interactive formal content.

Emdash's interactive proof mode and clear error reporting are designed for human-AI collaboration.

CONCLUSION

Emdash demonstrates a practical pathway for "functorial programming":

- A **synthetic** approach to category theory in a dependently typed framework.
- A novel **functorial elaboration** mechanism that makes coherence a definitional check.
- A robust implementation guided by a **formal specification**.
- A practical tool with interactive proving, serving as the kernel for the **hotdocX** platform.

FUTURE WORK

DEEPEN THE SYNTHESIS

- ω-Categories via `Total_cat`
- Limits, Adjunctions
- Univalence

ENHANCE THE FRAMEWORK

- Universe Hierarchy
- Performance Tuning
- Richer Tactic Language

HOTDOCX & AI

- AI-driven proof suggestions
- Document parsing (LaTeX)
- Visualization tools

THANK YOU

QUESTIONS?

Project Repository: github.com/hotdocx/emdash

hotdocX Platform: hotdocx.github.io

Continue: ./slides-part-2-hotdocx.html

PART 2/2

HOTDOCX & JSCOQ

AN INTERACTIVE, AI-AUGMENTED, AND MONETIZABLE PLATFORM FOR COQ

THE CHALLENGE: SHARING COQ DEVELOPMENTS

Coq is a cornerstone of formal methods, but sharing its rich, interactive proofs is hard.

CURRENT STATE

- Static PDFs (Flattened)
- Code Repositories

THE COMMUNITY'S DESIRE

- Dynamic & Accessible Platforms

THE CHALLENGE: SHARING COQ DEVELOPMENTS

Coq is a cornerstone of formal methods, but sharing its rich, interactive proofs is hard.

CURRENT STATE

- Static PDFs (Flattened)
- Code Repositories
- High barrier to entry for non-experts

THE COMMUNITY'S DESIRE

- Dynamic & Accessible Platforms

THE CHALLENGE: SHARING COQ DEVELOPMENTS

Coq is a cornerstone of formal methods, but sharing its rich, interactive proofs is hard.

CURRENT STATE

- Static PDFs (Flattened)
- Code Repositories
- High barrier to entry for non-experts

THE COMMUNITY'S DESIRE

- Dynamic & Accessible Platforms
- Initiatives like "Coq Exchange" & "Coq Platform Docs" showed the need.

THE CHALLENGE: SHARING COQ DEVELOPMENTS

Coq is a cornerstone of formal methods, but sharing its rich, interactive proofs is hard.

CURRENT STATE

- Static PDFs (Flattened)
- Code Repositories
- High barrier to entry for non-experts

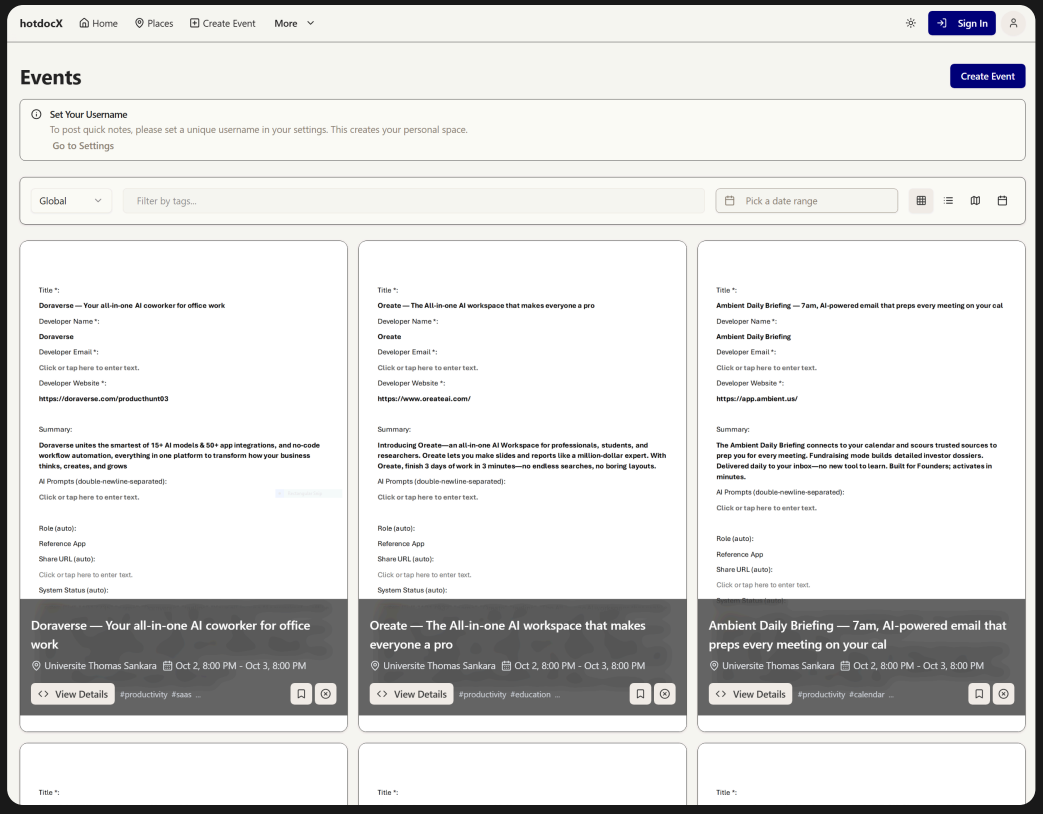
THE COMMUNITY'S DESIRE

- Dynamic & Accessible Platforms
- Initiatives like "Coq Exchange" & "Coq Platform Docs" showed the need.

THE CORE PROBLEM: A MISSING PLATFORM THAT IS BOTH INTERACTIVE AND SUSTAINABLE.

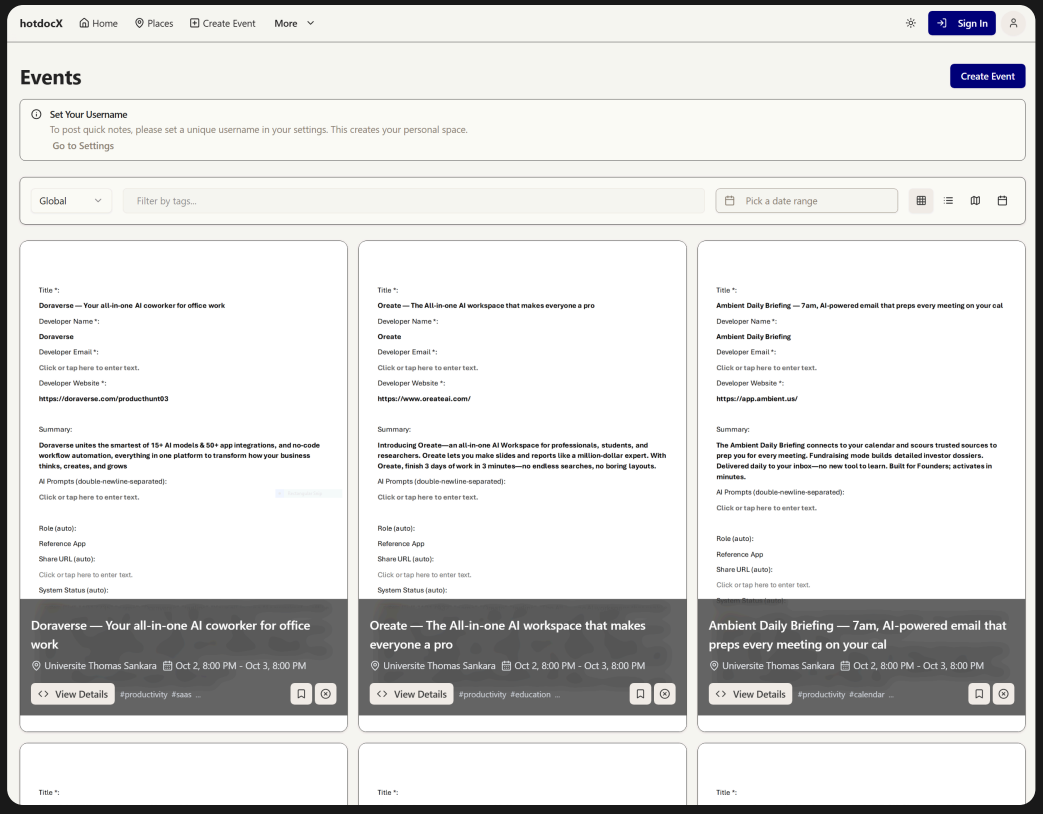
OUR SOLUTION: THE HOTDOCX PLATFORM

An AI-powered social marketplace that transforms documents into live, interactive web applications.



OUR SOLUTION: THE HOTDOCX PLATFORM

An AI-powered social marketplace that transforms documents into live, interactive web applications.

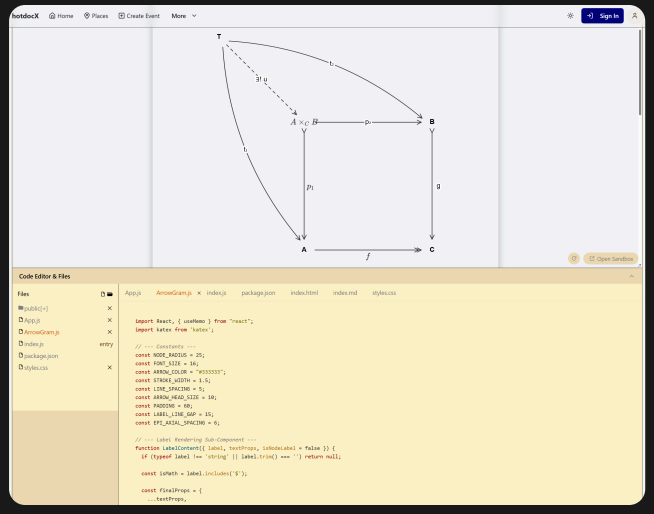


We're moving from "papers-with-code" to **"papers-as-apps"**.

CORE CONCEPT: AI TEMPLATES

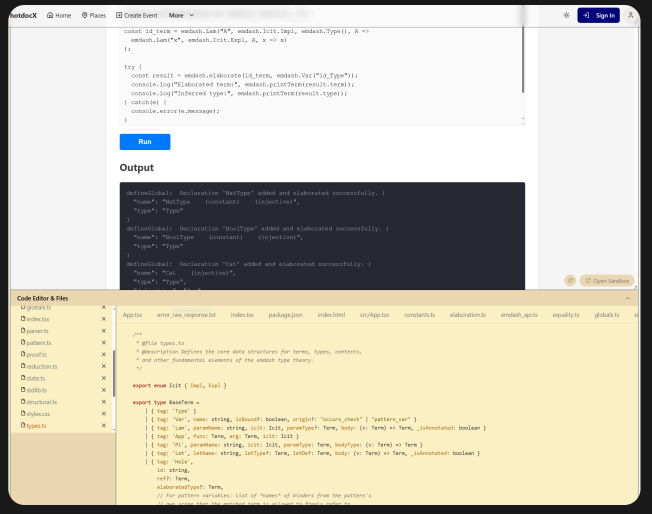
A user provides a document (PDF, LaTeX, etc.) and a prompt. Our AI pipeline uses a template to generate a runnable application.

Arrowgram Template



Renders commutative diagrams from natural language.

Emdash Template

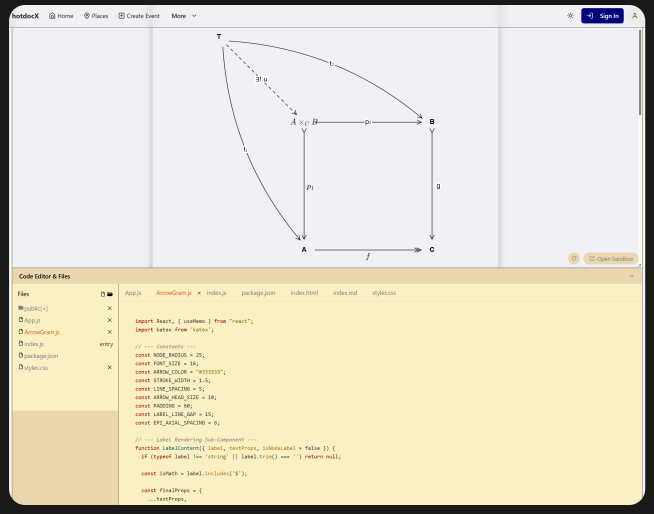


Provides a live environment for our logical framework.

CORE CONCEPT: AI TEMPLATES

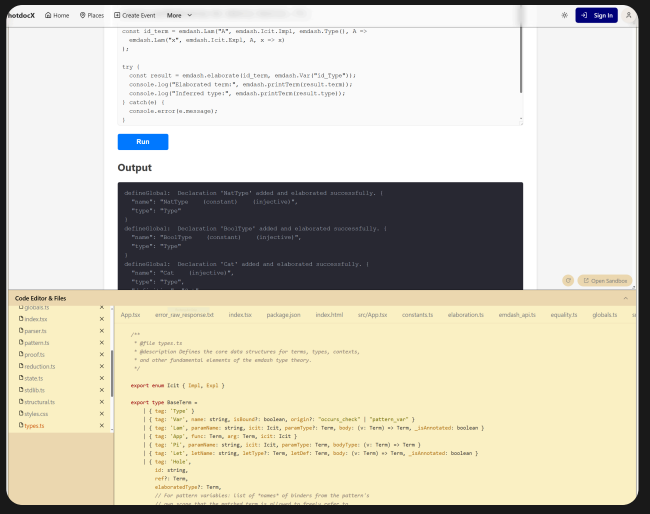
A user provides a document (PDF, LaTeX, etc.) and a prompt. Our AI pipeline uses a template to generate a runnable application.

Arrowgram Template



Renders commutative diagrams from natural language.

Emdash Template



Provides a live environment for our logical framework.

The platform is flexible and context-aware. Now, let's add Coq to the mix.

MAKING COQ A FIRST-CLASS CITIZEN

We created a **`jsCoq` template** for the hotdocX ecosystem.

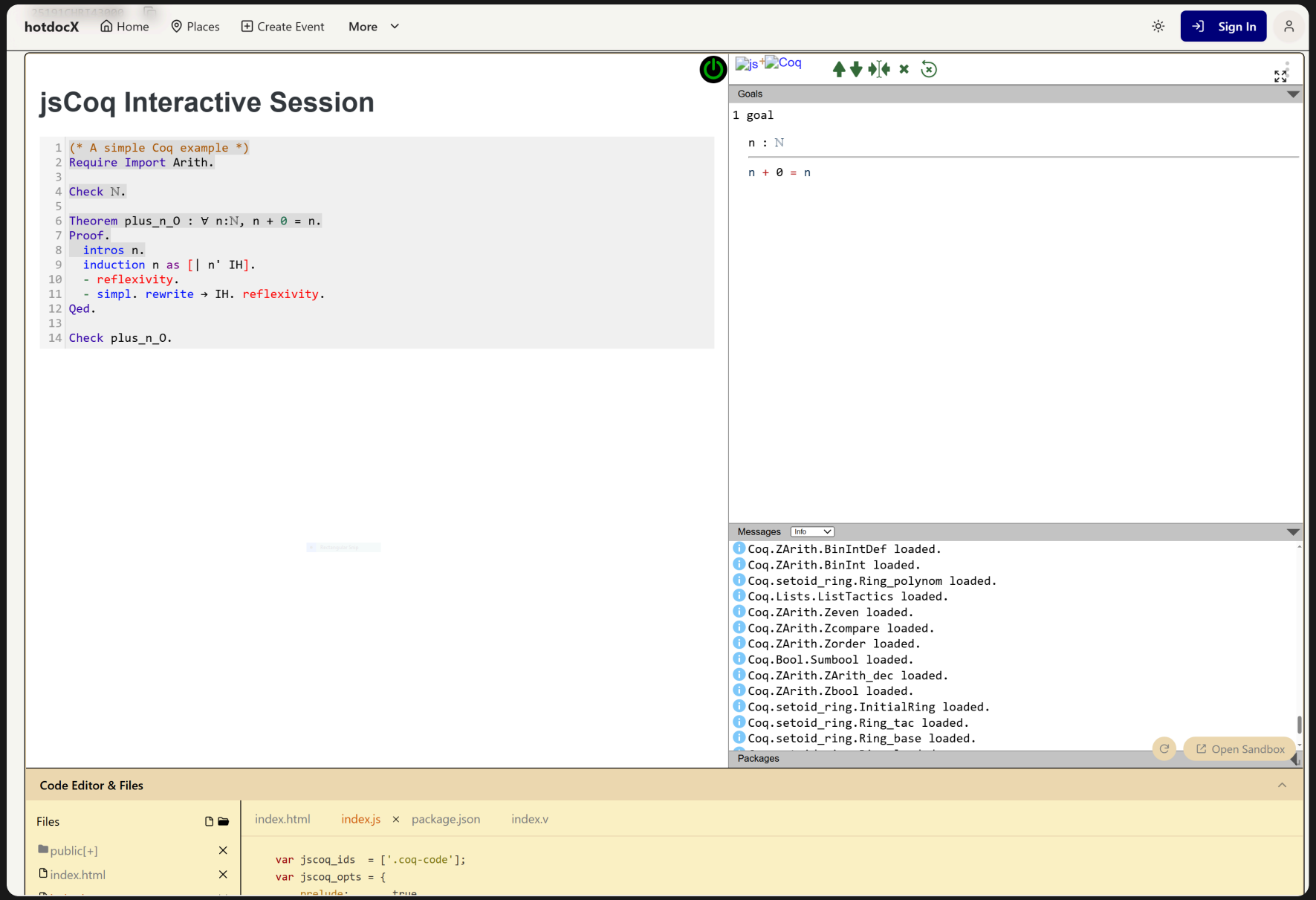
Any Coq script can become a first-class, interactive, and shareable hotdocX event.

MAKING COQ A FIRST-CLASS CITIZEN

We created a **`jsCoq` template** for the hotdocX ecosystem.

Any Coq script can become a first-class, interactive, and shareable hotdocX event.

LIVE DEMO 1: INTERACTIVE COQ SESSION



An interactive jsCoq session embedded within a hotdocX event.

UNLOCKING POWERFUL NEW WORKFLOWS

EDUCATION

- Create tutorials from lectures + Coq files.
- Students solve exercises in-browser.

RESEARCH

- Publish an arXiv paper and a companion hotdocX event.
- Readers can experiment with lemmas directly.

UNLOCKING POWERFUL NEW WORKFLOWS

EDUCATION

- Create tutorials from lectures + Coq files.
- Students solve exercises in-browser.
- **Monetize solutions** behind a pay-to-view wall.

RESEARCH

- Publish an arXiv paper and a companion hotdocX event.
- Readers can experiment with lemmas directly.

UNLOCKING POWERFUL NEW WORKFLOWS

EDUCATION

- Create tutorials from lectures + Coq files.
- Students solve exercises in-browser.
- **Monetize solutions** behind a pay-to-view wall.

RESEARCH

- Publish an arXiv paper and a companion hotdocX event.
- Readers can experiment with lemmas directly.
- Bridge the gap between paper and formalization.

WHY IS THIS DIFFERENT? A SUSTAINABLE BUSINESS MODEL

Previous platforms were often excellent academic projects but lacked a model for long-term sustainability.

WHY IS THIS DIFFERENT? A SUSTAINABLE BUSINESS MODEL

Previous platforms were often excellent academic projects but lacked a model for long-term sustainability.

hotdocX is built from the ground up on a **creator economy**.

WHY IS THIS DIFFERENT? A SUSTAINABLE BUSINESS MODEL

Previous platforms were often excellent academic projects but lacked a model for long-term sustainability.

hotdocX is built from the ground up on a **creator economy**.

- **Monetize Your Content:** Creators define a "Purchase Menu" for events (e.g., view source, remix rights, access solutions).
- **Virtual Currency:** Users purchase "Coins" via Stripe to access premium content.
- **Subscription Tiers:** Free, Premium, and VIP tiers unlock platform features like content mirroring and advanced API access.

WHY IS THIS DIFFERENT? A SUSTAINABLE BUSINESS MODEL

Previous platforms were often excellent academic projects but lacked a model for long-term sustainability.

hotdocX is built from the ground up on a **creator economy**.

- **Monetize Your Content:** Creators define a "Purchase Menu" for events (e.g., view source, remix rights, access solutions).
- **Virtual Currency:** Users purchase "Coins" via Stripe to access premium content.
- **Subscription Tiers:** Free, Premium, and VIP tiers unlock platform features like content mirroring and advanced API access.

THIS ISN'T JUST A TOOL; IT'S A SELF-SUSTAINING ECOSYSTEM.

THE GRAND VISION: A BROWSER-NATIVE COQ KERNEL

The co-location of `jsCoq` and our native framework, `Emdash`, opens a unique research opportunity.

A HYBRID EMDASH/COQ-LIGHT KERNEL

We can systematically augment the Emdash (λΠ-calculus) kernel with the features of CIC, all within a native TypeScript environment.

THE GRAND VISION: A BROWSER-NATIVE COQ KERNEL

The co-location of `jsCoq` and our native framework, `Emdash`, opens a unique research opportunity.

A HYBRID EMDASH/COQ-LIGHT KERNEL

We can systematically augment the Emdash (λΠ-calculus) kernel with the features of CIC, all within a native TypeScript environment.

EDUCATION

A "hackable" kernel for teaching CIC without a complex build setup.

RESEARCH

Rapidly prototype new tactics and language features in TypeScript.

AI INTEGRATION

A native JS kernel is easier for AI agents to interact with than a compiled binary.

JOIN US IN BUILDING THE FUTURE OF COQ

We have a working platform, a clear vision, and a sustainable model (dynamic fee percentage on transactions, applied to new authors referred by white-labelled multi-tenant publishers-hosted instances of hotdocx; i.e. `professor1.github.io/hotdocx` with `publisher2.github.io/hotdocx`).

JOIN US IN BUILDING THE FUTURE OF COQ

We have a working platform, a clear vision, and a sustainable model (dynamic fee percentage on transactions, applied to new authors referred by white-labelled multi-tenant publishers-hosted instances of hotdocx; i.e. `professor1.github.io/hotdocx` with `publisher2.github.io/hotdocx`).

To move from a developer preview to a robust product for the community, we need your support.

JOIN US IN BUILDING THE FUTURE OF COQ

We have a working platform, a clear vision, and a sustainable model (dynamic fee percentage on transactions, applied to new authors referred by white-labelled multi-tenant publishers-hosted instances of hotdocx; i.e. `professor1.github.io/hotdocx` with `publisher2.github.io/hotdocx`).

To move from a developer preview to a robust product for the community, we need your support.

BECOME A SPONSOR

JOIN US IN BUILDING THE FUTURE OF COQ

We have a working platform, a clear vision, and a sustainable model (dynamic fee percentage on transactions, applied to new authors referred by white-labelled multi-tenant publishers-hosted instances of hotdocx; i.e. `professor1.github.io/hotdocx` with `publisher2.github.io/hotdocx`).

To move from a developer preview to a robust product for the community, we need your support.

BECOME A SPONSOR

Your contribution, whether individual or institutional, directly funds development, infrastructure, and community growth.

GITHUB.COM/SPONSORS/HOTDOCX

Sponsoring provides premium and VIP tiers on the hotdocX platform.

THANK YOU

QUESTIONS & DISCUSSION

Platform: hotdocx.github.io

Sponsor: github.com/sponsors/hotdocx