Mozilla x Hack-Nation

Global AI Hackathon | Feb 7-8, 2026

in collaboration with MIT Sloan AI Club & MIT Club of Northern California

Bring Your Own AI to Every Website - The Universal Web Agent

Sponsored Track by Mozilla

---

1. Goals and Motivation

As AI improves, the hardest problems are shifting from generation to execution: what an AI can see, what it can do, and under what context or permission.

Those decisions have long converged in the browser. It's where people already are, where identity and live context exist, and where real actions take place across sites and sessions.

But today, developers don't actually have primitives for working with those capabilities directly. And users have no way to bring their own AI to the web — they're stuck with whatever model a website embeds, re-explaining preferences everywhere they go.

We've been building something we call the Web Agent API: a browser-level SDK that exposes graduated capability tiers, from tool calling and text to real page context and interaction.

This hackathon is an invitation to explore it with us, pressure-test the boundaries, and help shape what responsible execution in the browser should look like.

This isn't a finished answer — it can't be. But it is a concrete way into the problem space.

2. Hero Features – What You Could Build

The SDK exposes two related APIs that can be used independently or together.

Core AI & Tooling API

- Connect to LLMs
- Call MCP servers (local or remote)
- Produce structured or free-form outputs

This layer has no browser authority by default.

Browser Capability API

- Access live page and tab context
- Perform browser interactions
- Coordinate multiple agents with distinct roles

Higher tiers increase capability — and responsibility. You choose how far up the stack to go.

Capability Tiers:

- **Tier 1:** LLM access + MCP tool calling (text and structured outputs)
- **Tier 2:** Browser context and page interaction (read, navigate, act)
- **Tier 3:** Coordinated workflows (multiple agents with distinct roles)

Your challenge: build something that couldn't be built before — a workflow, interaction, or experience that depends on real browser context and makes clear decisions about what an AI can see, what it can do, and how permission is handled.

3. Example Use Cases

Pick one and go deep.

Visual Search & Action "What keyboard is this?" → identify → search → filter by preferences → rank → purchase

Voice-Native Navigation "Find the refund policy and summarize it." No clicking required.

Cross-Site Workflows "Find flights, check my calendar, draft an email." One intent, multiple sites.

Memory-Aware Browsing "Is this similar to what I bought last year?" Your AI, your history.

Preference-First Experiences Budget constraints, accessibility needs, brand preferences — context that travels with you.

4. How to Think About This

Think of AI as a browser capability, not a website feature.

In this world, the browser is not just a renderer — it's a coordinator. It manages permissions, routes tools, maintains memory, and mediates between the user and intelligent systems. Websites provide domain-specific tools without managing inference. Context flows on user terms.

We're moving responsibility up the stack. Once AI starts acting, it runs into the browser. The browser is already where identity, context, and action converge.

This is about who owns the moment AI acts.

As browser-level capabilities increase, permission becomes the core design challenge. We're especially interested in how you reason about:

- Permissions granted to an agent vs. a specific task
- How long permissions should last
- Which actions require explicit user confirmation
- What should be read-only vs. mutable

There is no single correct model. We're learning from the tradeoffs you surface.

Demo videos for inspiration:

- Video 01
- Video 02

Github inspiration:

- https://github.com/r/any-llm-ts
- https://github.com/r/Harbor

5. Evaluation Criteria

Projects will be evaluated on:

Clarity of Execution Boundaries Is it clear what the AI can see, do, and decide? Are the start and end of execution explicit?

Thoughtful Use of Browser Context Is browser context used intentionally, not just because it's available? Does the project demonstrate why the browser matters for this workflow?

Permission Design Are permissions scoped, time-bounded, or contextual? Does the project grapple with agent vs. task-level permission tradeoffs?

Legibility & User Control Can a user understand what's happening and why? Are actions inspectable, interruptible, or confirmable?

Judgment & Restraint Does the design show care around what shouldn't be automated? Are limitations or tradeoffs acknowledged rather than hidden?

Bonus consideration for projects that surface new questions, edge cases, or failure modes.

6. Why It Matters

This isn't about building another chatbot.

It's about what happens when intelligence becomes a browser primitive — portable, user-controlled, permission-mediated.

For accessibility, this could be transformative: voice-first browsing, intent-based navigation, automatic simplification and summarization become defaults, not add-ons.

For productivity, it changes everything: fewer clicks, fewer tabs, fewer manual workflows. Instead of learning interfaces, users express goals.

For developers, it removes unnecessary barriers: the plumbing becomes platform infrastructure, not application code.

Just as browsers once standardized how we access information, this challenge asks: what would it mean to standardize how we act on it?

Not as a chatbot. Not as a plugin. But as a foundational capability of the web.