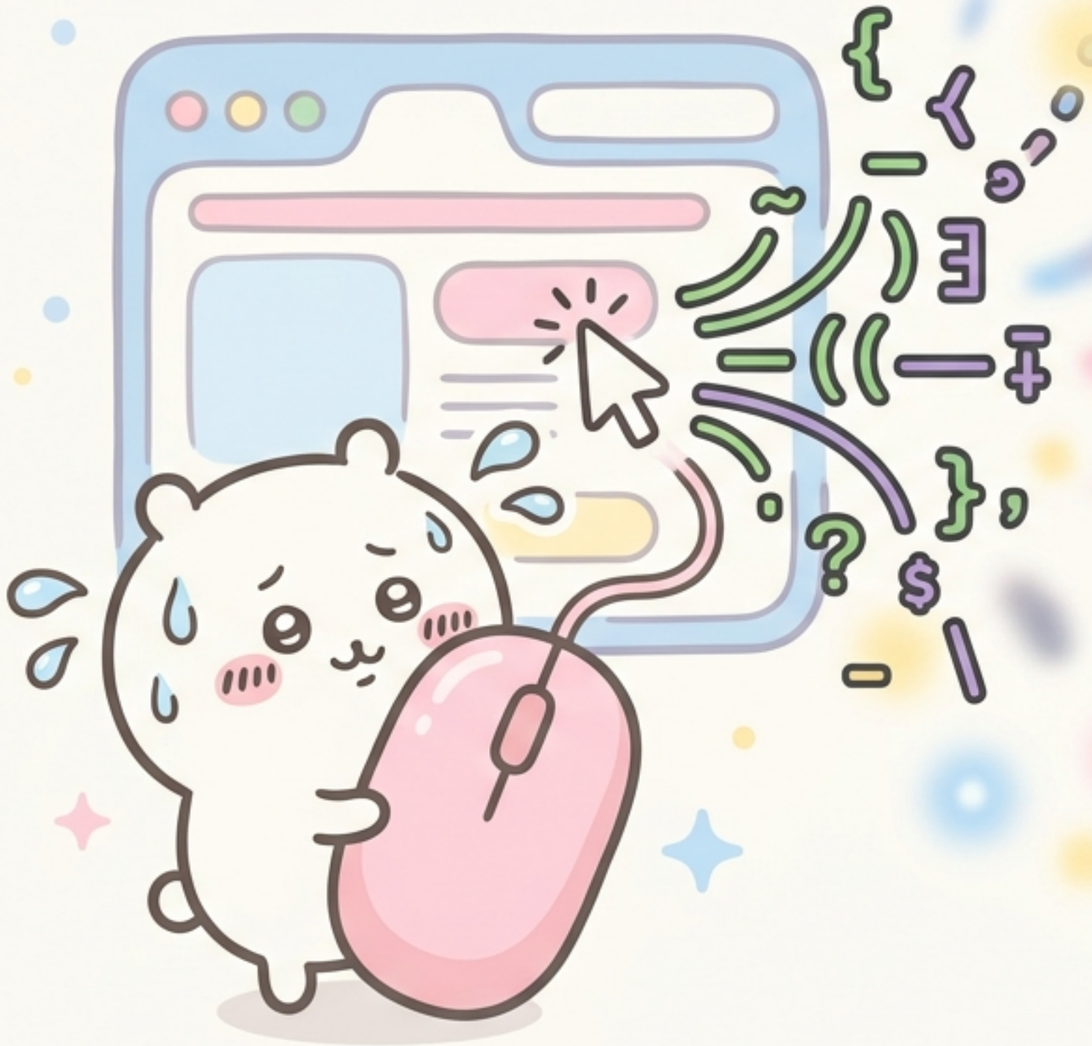


The Inspiration

Sparked by pretext—manipulating text layouts at 120fps without touching the DOM.

Exploding the Static UI with Physics-Based Particles

The Hypothesis

Transforming a pure vanilla JS terminal portfolio so every visible character becomes a canvas particle on tab switch.

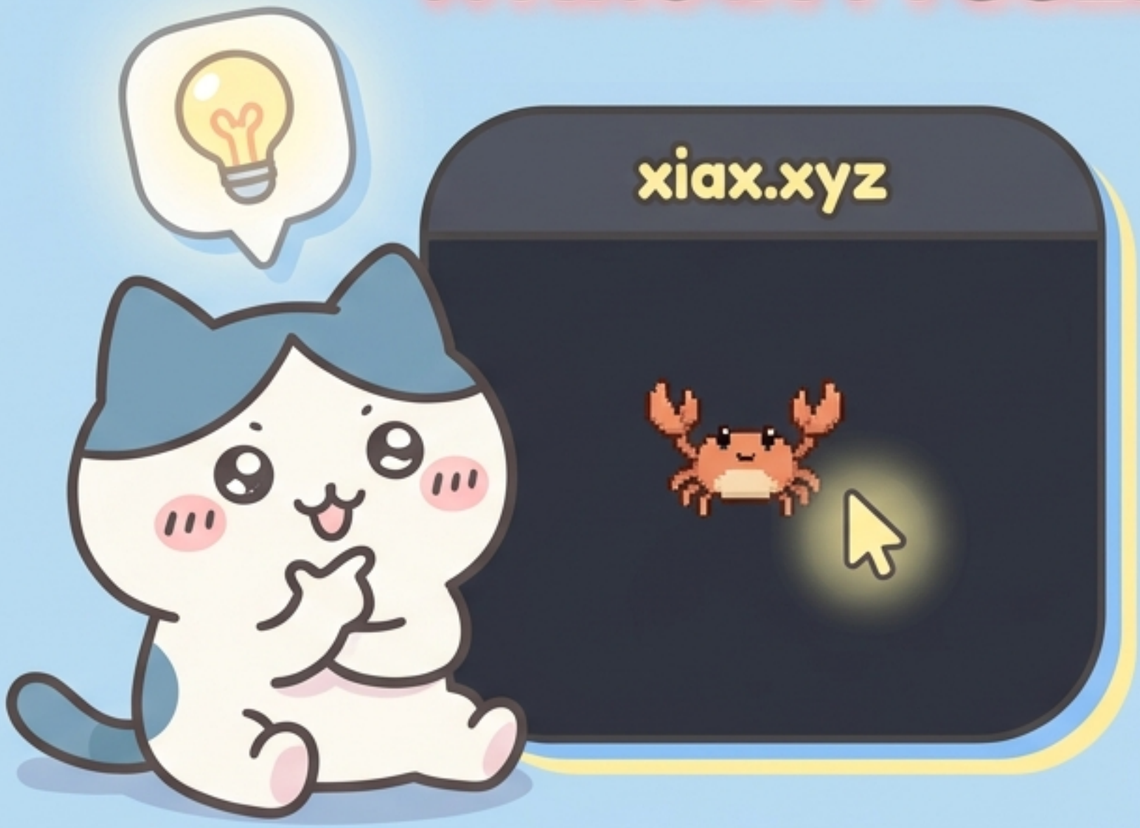
The Mechanics

Force: Inverse-square blast radius.

Environment: Gravity, air resistance, rotation, and fade.

Resolution: Terminal typewriter seamlessly prints new content after the blast clears.

Extracting 5,000+ DOM Nodes Without Freezing the Browser



DOM Text

↓

getBoundingClientRect() per node

Fallback to per-character Range measurement only for rare wrapped text.

↓

Monospace Math

↓

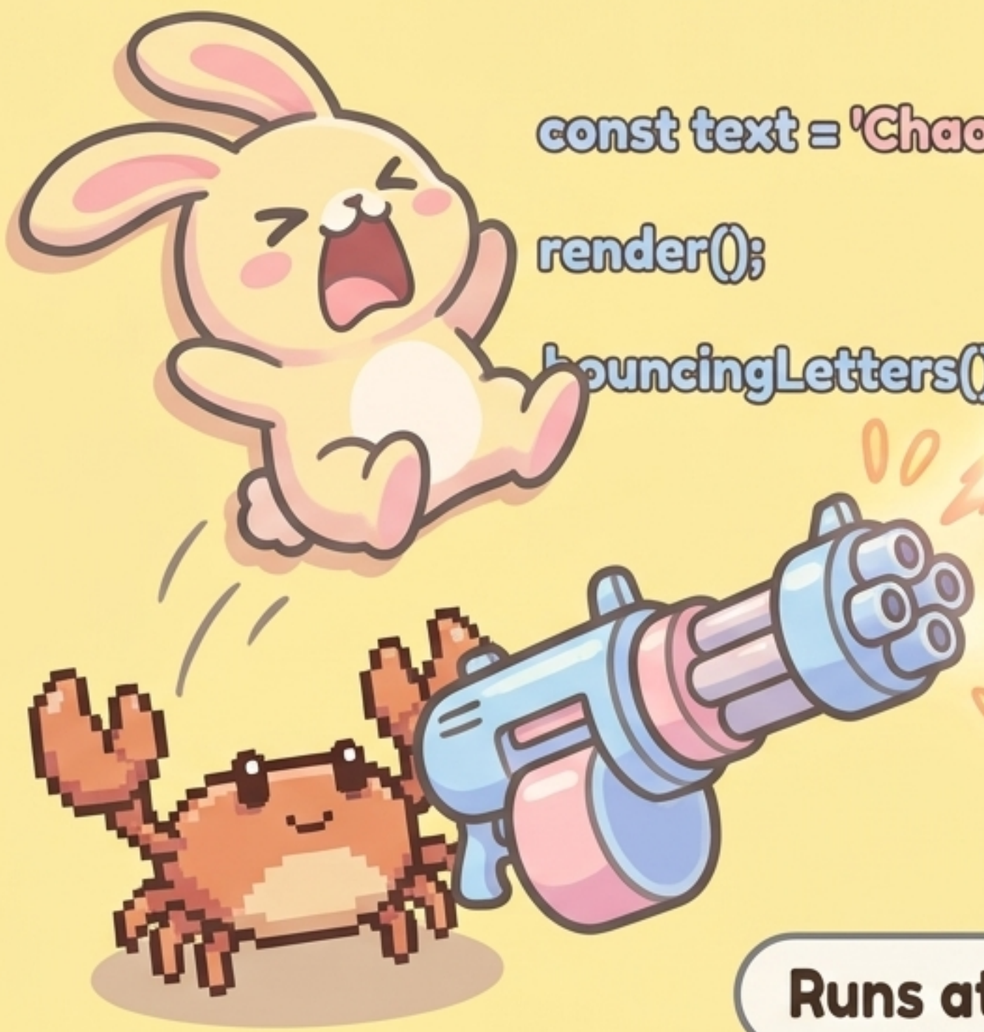
Canvas Rendering

Summoning Clawd:

- Trigger: Double-click anywhere.
- Action: Cursor hides. Clawd appears and follows movements with a smooth, calculated lag.

Result: DOM reads reduced from 5,000+ to ~200. Imperceptible and lightning fast.

Bringing Chaotic Spring Physics to 60fps Canvas Text



Ramping Fire:
Hold click to ramp fire rate from 6/sec up to 40/sec.

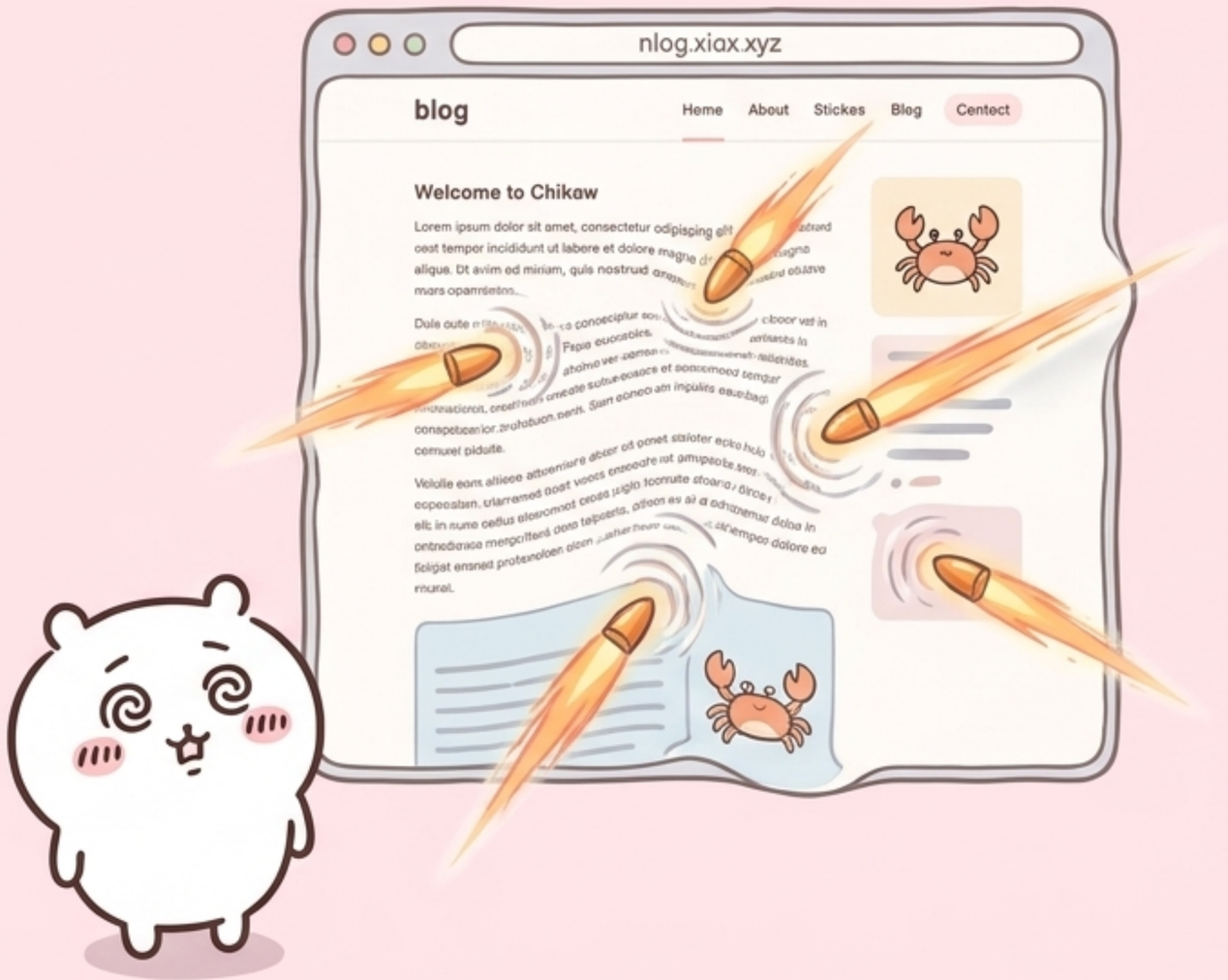
Impact Velocity:
Bullets hit characters, transferring immediate velocity impulse.

Bounce Logistics:
Bullets hit walls and bounce back with 55% energy retention.

Spring & Settle:
Release mouse, and characters spring back to home positions using damping physics.

Runs at 60fps with zero layout reflow. Pure canvas magic.

When Canvas Extraction Fails on the Complex Next.js DOM



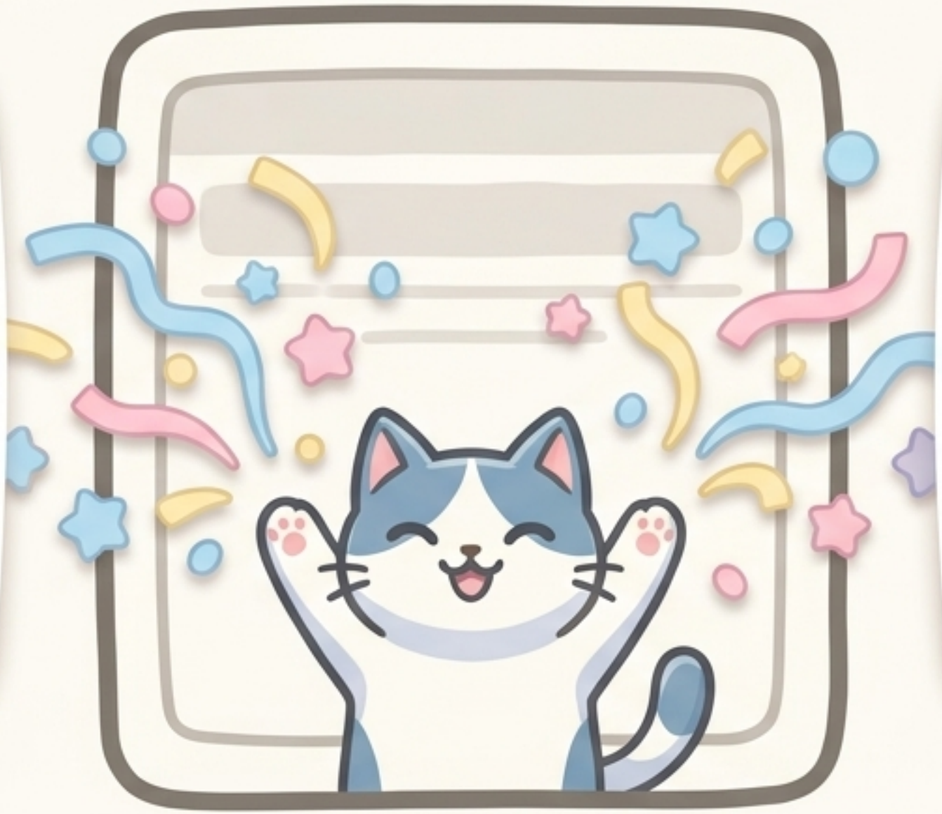
Feature	Terminal (xiax.xyz)	Blog (nlog.xiax.xyz)
Environment	Vanilla JS, single scrollable div	Next.js, sticky headers, variable fonts
Rendering	Pure Canvas overlay	Transparent Canvas (bullets only)
Clawd Element	Drawn directly on canvas	Fixed-position <img> (CSS animations)
Impact Logic	Canvas coordinate updates	CSS Transforms pushing actual DOM nodes

Key Insight: Don’t fight the DOM. Let the browser do the work. The layout stays correct, links remain clickable, and bullet impacts create physical ripples in the actual DOM.

Shipping 7 Effects and 2,000 Lines of Code in One Session

The Interactive Arsenal

- **Terminal:** Tab Explosions, Typewriter loading, Clawd Machine Gun.
- **Blog:** Post Card, Tag Cloud, and Code Block Explosions.
- **Easter Eggs:** Konami Code Mode for permanent Clawd. Scroll to the end of any post for a Confetti Burst.



The Claude Code Pipeline

Built entirely in one afternoon through a 5-step process:

1. Research target APIs (pretext).
2. Architect Canvas/DOM handoffs.
3. Implement inverse-square physics engines.
4. Iterate touch events & CSS specificity.
5. Port to Next.js complex DOM.

Web development can still be incredibly fun.