subject: # bringing the modern web to the original ipad mini

date: jul 21, 2026

re: resurrecting ios 6 with a remote browser

I recently visited my parents and found the first generation iPad mini they bought me when I was ten years old. More than a decade later, it is still a beautiful piece of hardware. Small, light, sturdy, the kind of device Apple used to make before every screen became a laptop replacement.

Then I turned it on and remembered why I stopped using it: It was running iOS 9.3.5, which is brutal on the A5 chip. Every tap had weight, animations stuttered, opening an app felt like an eternity. Apple really killed this device with iOS 9.

Still, I felt like the hardware was too beautiful to leave in a drawer. I wanted to turn it into a smooth, distraction free reading device.

The problem was that reading is not really offline anymore. Even if the actual reading happens in iBooks, the surrounding workflow lives on the web: finding articles, checking references, downloading PDFs, opening links, asking an LLM a question when something does not make sense.

Getting there was a nostalgic trip through the legacy jailbreak toolchain. I jailbroke it with Carbon, used Legacy iOS Kit to drop it into kDFU mode, downgraded to iOS 8.4.1, set up OpenSSH, and used CoolBooter to dual boot into iOS 6.1.3.

The result was great. iOS 6 on the A5 runs butter smooth. I transferred a few books over the air with iFile and suddenly had exactly what I wanted: a small tablet with a great form factor, a still good screen, and the lovely skeuomorphic iBooks app.

Everything was perfect. Right up until the first time the reading workflow left the book.

the webkit wall

I installed modern root certificates, but that only gets you so far. Safari on iOS 6, or really WebKit, is missing too much: modern TLS support, CSS Grid, Flexbox, and large chunks of modern JavaScript.

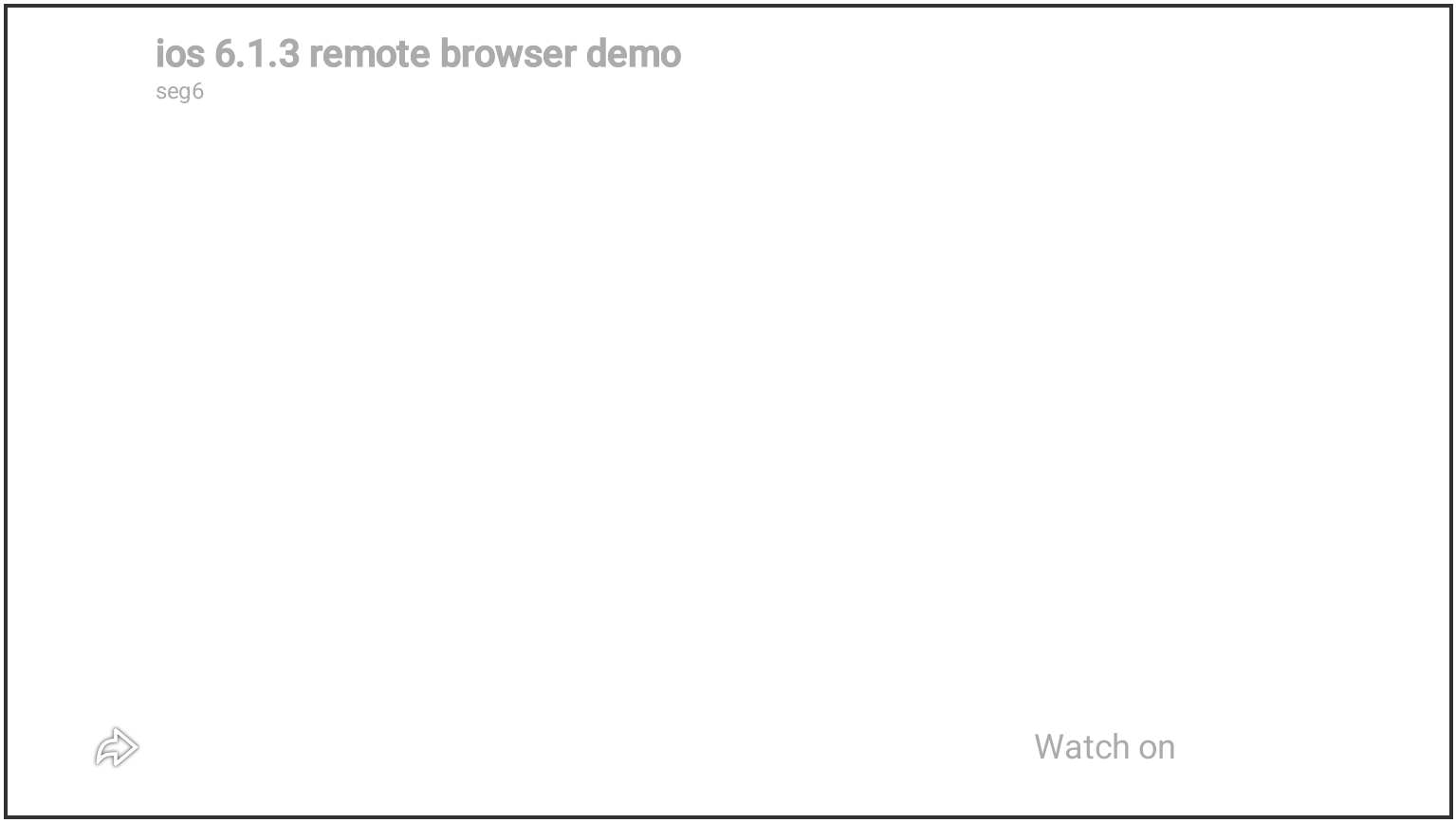
The modern web is structurally incompatible with a browser engine from 2012. And because this is iOS, every third party browser is stuck with the same system WebKit.

The obvious solutions all have problems:

- **VNC / Remote Desktop:** Bad latency, bad battery life, and a UI built for a mouse instead of touch.
- **Off-the-shelf remote browsers:** Mostly defunct, or clunky in the same way. I wanted something that felt like Safari, not a remote desktop session.

So the shape of the problem changed: if the iPad could not render the web, something else can render it for the iPad.

I needed a remote browser that felt native, ran at 30fps, respected a 512MB RAM limit, and had touch first controls. Since I was on vacation, it also felt like the perfect pet project to hand to an LLM and see how far it could get. Out of that experiment, Surf was born.



the architecture

Surf has two halves: a backend running on a VPS, and a native iOS 6 app written in Objective-C.

The backend runs a headful Chromium instance in Docker through Xvfb. Running Chromium headful under a virtual display avoids the bot detection penalties that headless browsers tend to trip. A Go server talks to Chromium through the Chrome DevTools Protocol and handles input, tabs, and navigation.

For the visual feed, I first tried CDP's `Page.screencastFrame`, but it was too slow for smooth scrolling. Instead, the backend uses `ffmpeg` to grab the X11 screen directly and encode it to H.264 on the fly.

The stream is sent over a WebSocket connection. I had plans to spend more time on the network layer, but the current implementation already hits the target frame rate, so I left that alone.

The native client is built with Theos. Every build gets pushed to the iPad over SSH and installed with `dpkg`.

wrapping up

The iPad is useful again. Surf has a unified omnibox, tabs, clipboard sharing with the host OS, downloads, uploads, and a bunch more little tweaks that make the experience *delightful*.

It is obviously ridiculous to browse the modern web this way. But it is also exactly the kind of ridiculous that makes old computers feel alive again.

If you'd like to tinker with it, the source code is available on GitHub: https://github.com/seg6/rbrowser.

thread: DISCUSSION open thread