Everything I built with Claude Artifacts this week

21st October 2024

I'm a huge fan of Claude's **Artifacts** feature, which lets you prompt Claude to create an interactive Single Page App (using HTML, CSS and JavaScript) and then view the result directly in the Claude interface, iterating on it further with the bot and then, if you like, copying out the resulting code.

I was digging around in my Claude activity export (I built a claude-to-sqlite tool to convert it to SQLite I could explore it in Datasette) and decided to see how much I'd used artifacts in the past week. It was more than I expected!

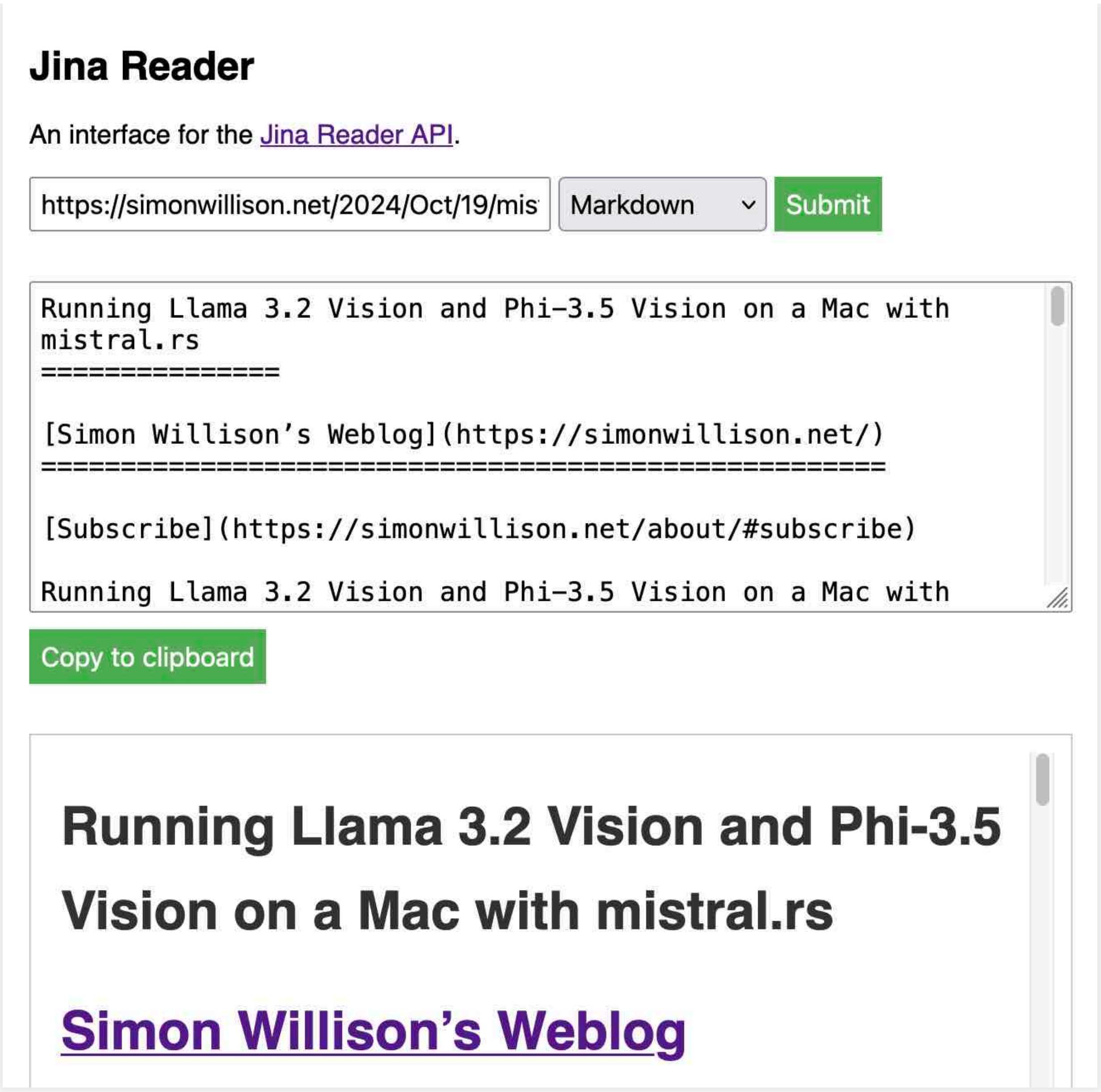
Being able to spin up a full interactive application—sometimes as an illustrative prototype, but often as something that directly solves a problem—is a remarkably useful tool.

Here's most of what I've used Claude Artifacts for in the past seven days. I've provided prompts or a full transcript for nearly all of them.

URL to Markdown with Jina Reader

I got frustrated at how hard it was to copy and paste the entire text of a web page into an LLM while using Mobile Safari. So I built a simple web UI that lets me enter a URL, calls the Jina Reader API to generate Markdown (which uses Puppeteer under the hood) and gives me that Markdown with a convenient "Copy" button.

Try it out: https://tools.simonwillison.net/jina-reader (Code)


Jina Reader
An interface for the Jina Reader API.
https://simonwillison.net/2024/Oct/19/mis
Markdown
Submit
Copy to clipboard

```
Running Llama 3.2 Vision and Phi-3.5 Vision on a Mac with
mistral.rs
===============

[Simon Willison’s Weblog](https://simonwillison.net/)
=====================================================

[Subscribe](https://simonwillison.net/about/#subscribe)

Running Llama 3.2 Vision and Phi-3.5 Vision on a Mac with
```

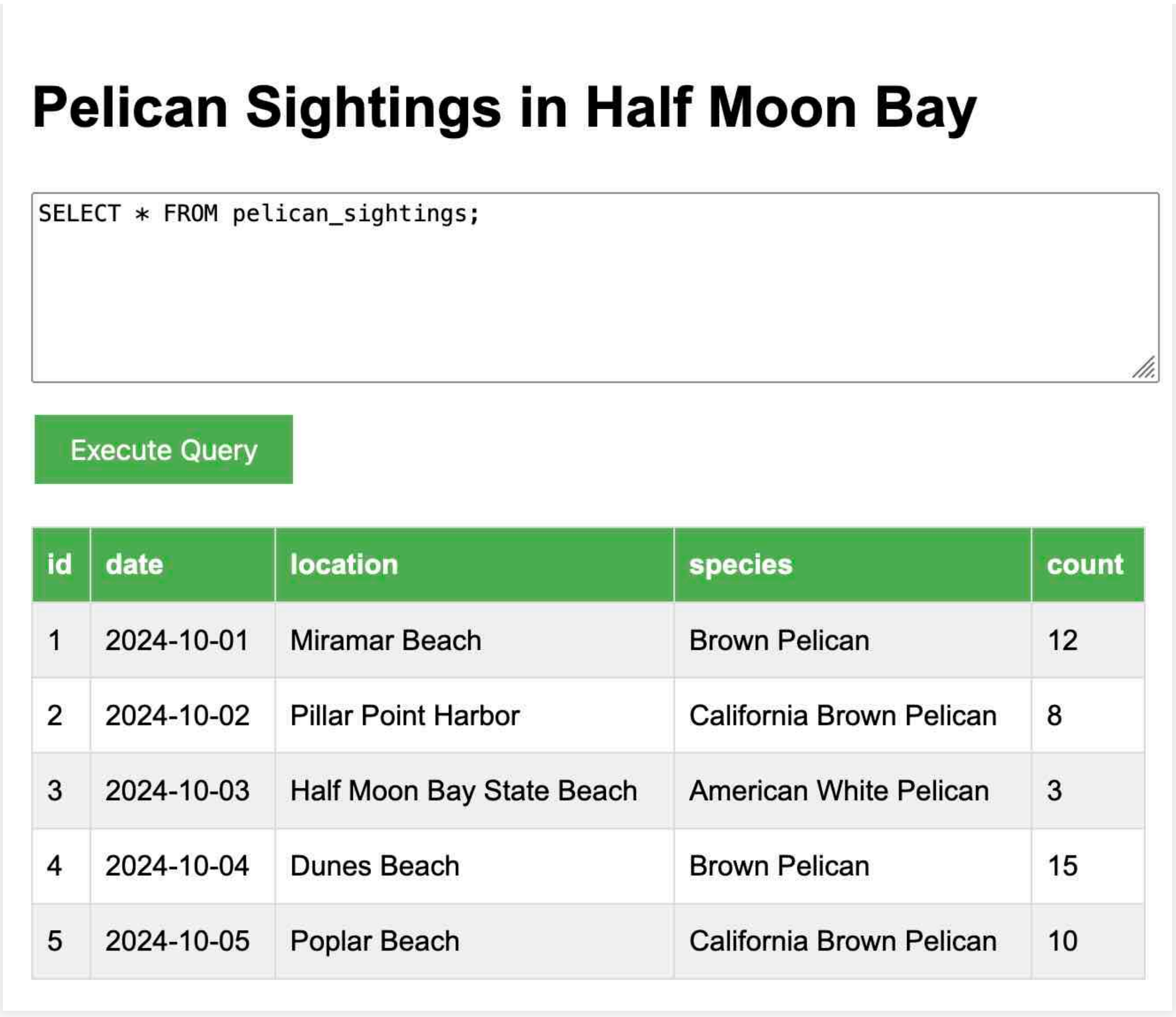
Running Llama 3.2 Vision and Phi-3.5 Vision on a Mac with mistral.rs

Simon Willison’s Weblog

I wrote more about that project here.

SQLite in WASM demo

A Hacker News conversation about SQLite’s WASM build lead me to the @sqlite.org/sqlite-wasm package on NPM, and I decided to knock together a quick interactive demo.



Pelican Sightings in Half Moon Bay

```
SELECT * FROM pelican_sightings;
```

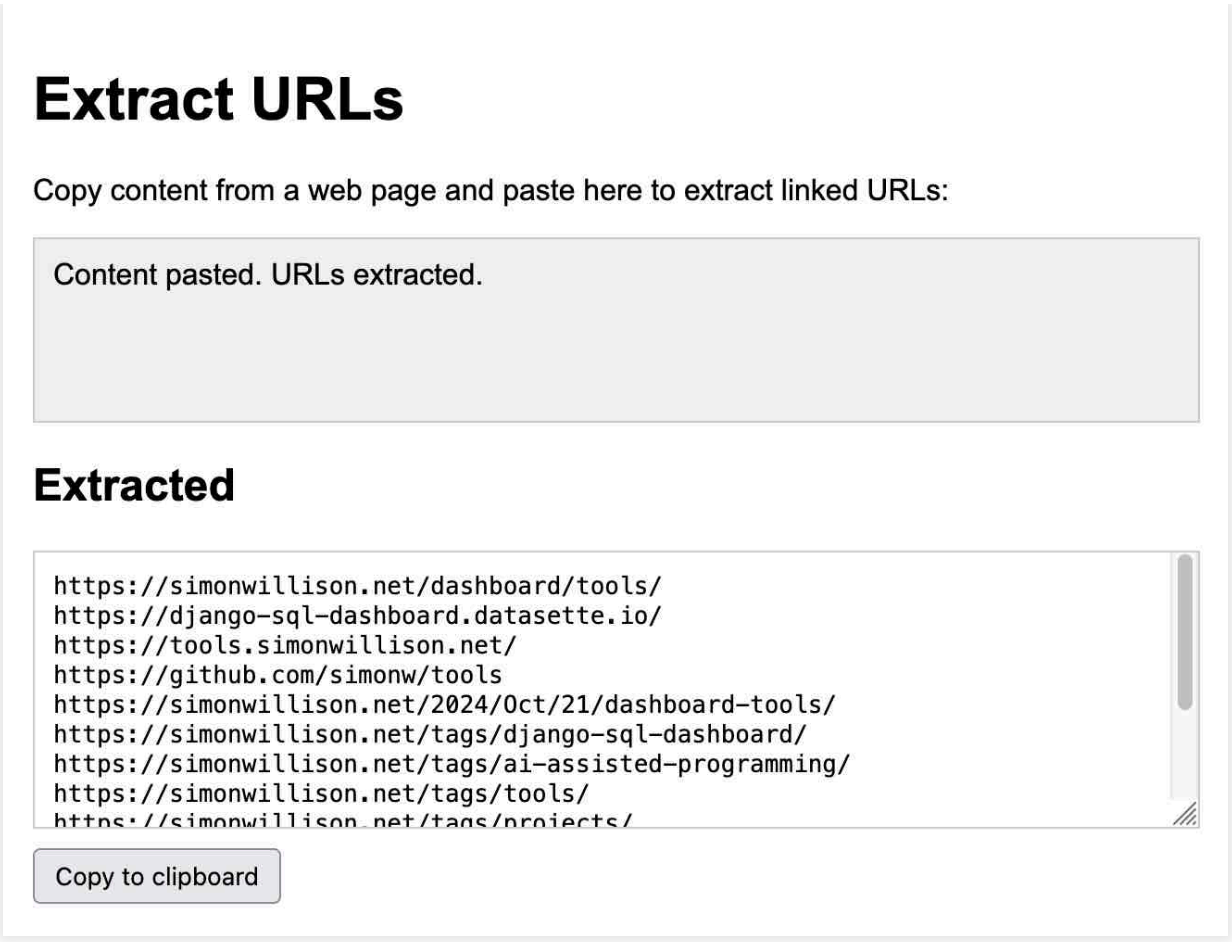
id	date	location	species	count
1	2024-10-01	Miramar Beach	Brown Pelican	12
2	2024-10-02	Pillar Point Harbor	California Brown Pelican	8
3	2024-10-03	Half Moon Bay State Beach	American White Pelican	3
4	2024-10-04	Dunes Beach	Brown Pelican	15
5	2024-10-05	Poplar Beach	California Brown Pelican	10

Try it out here: tools.simonwillison.net/sqlite-wasm

Code, Claude transcript

Extract URLs

I found myself wanting to extract all of the underlying URLs that were linked to from a chunk of text on a web page. I realized the fastest way to do that would be to spin up an artifact that could accept rich-text HTML pastes and use an HTML parser to extract those links.



Extract URLs

Copy content from a web page and paste here to extract linked URLs:

Content pasted. URLs extracted.

Extracted

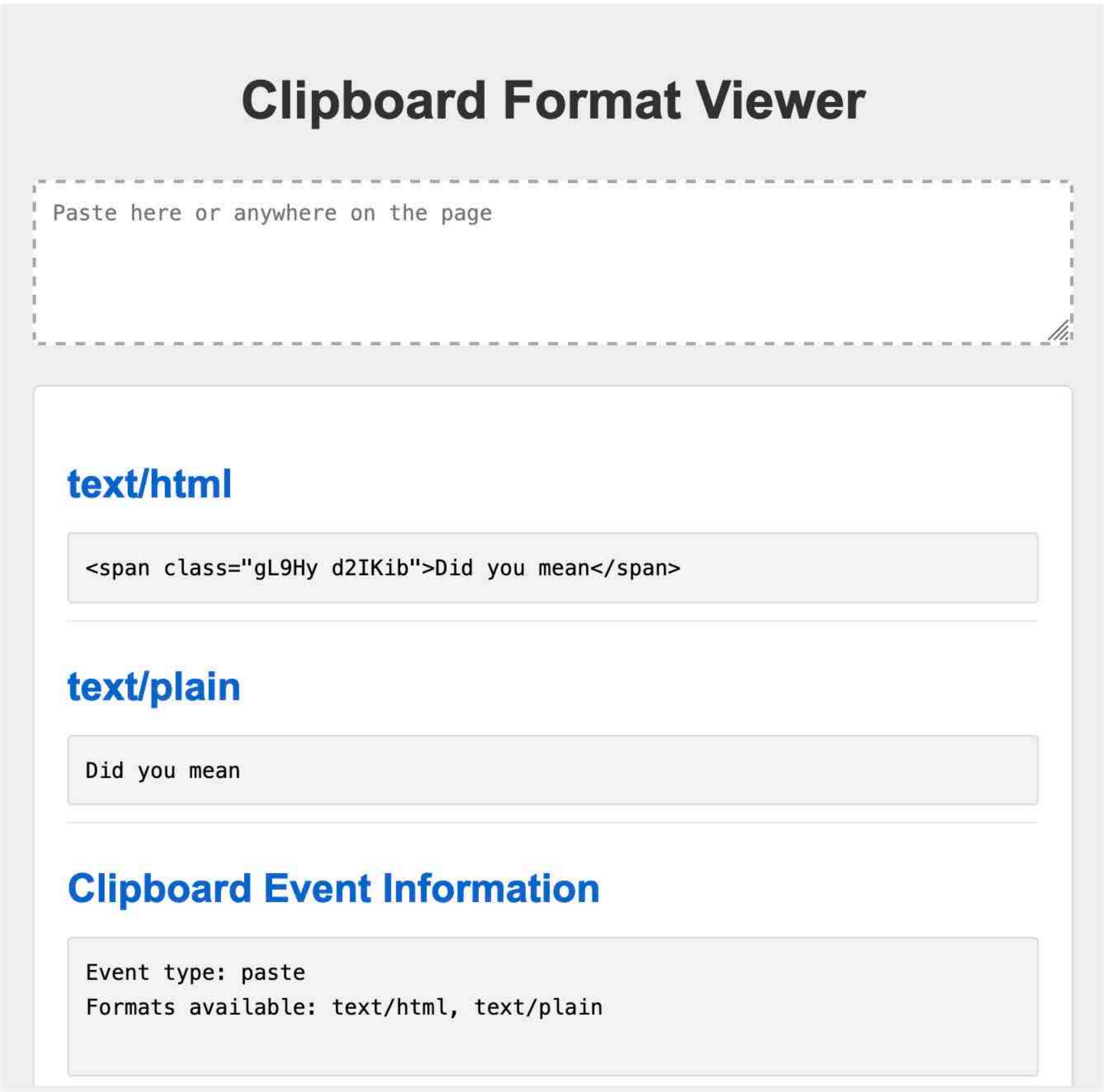
```
https://simonwillison.net/dashboard/tools/
https://django-sql-dashboard.datasette.io/
https://tools.simonwillison.net/
https://github.com/simonw/tools
https://simonwillison.net/2024/Oct/21/dashboard-tools/
https://simonwillison.net/tags/django-sql-dashboard/
https://simonwillison.net/tags/ai-assisted-programming/
https://simonwillison.net/tags/tools/
https://simonwillison.net/tags/projects/
```

https://tools.simonwillison.net/extract-urls

Code, Claude transcript

Clipboard viewer

Messing around with a tool that lets you paste in rich text reminded me that the browser clipboard API is a fascinating thing. I decided to build a quick debugging tool that would let me copy and paste different types of content (plain text, rich text, files, images etc) and see what information was available to me in the browser.


Clipboard Format Viewer
Paste here or anywhere on the page
text/html

```
<span class="gL9Hy d2IKib">Did you mean</span>
```

```
Did you mean
```

Clipboard Event Information

```
Event type: paste
Formats available: text/html, text/plain
```

https://tools.simonwillison.net/clipboard-viewer

Code, Claude transcript

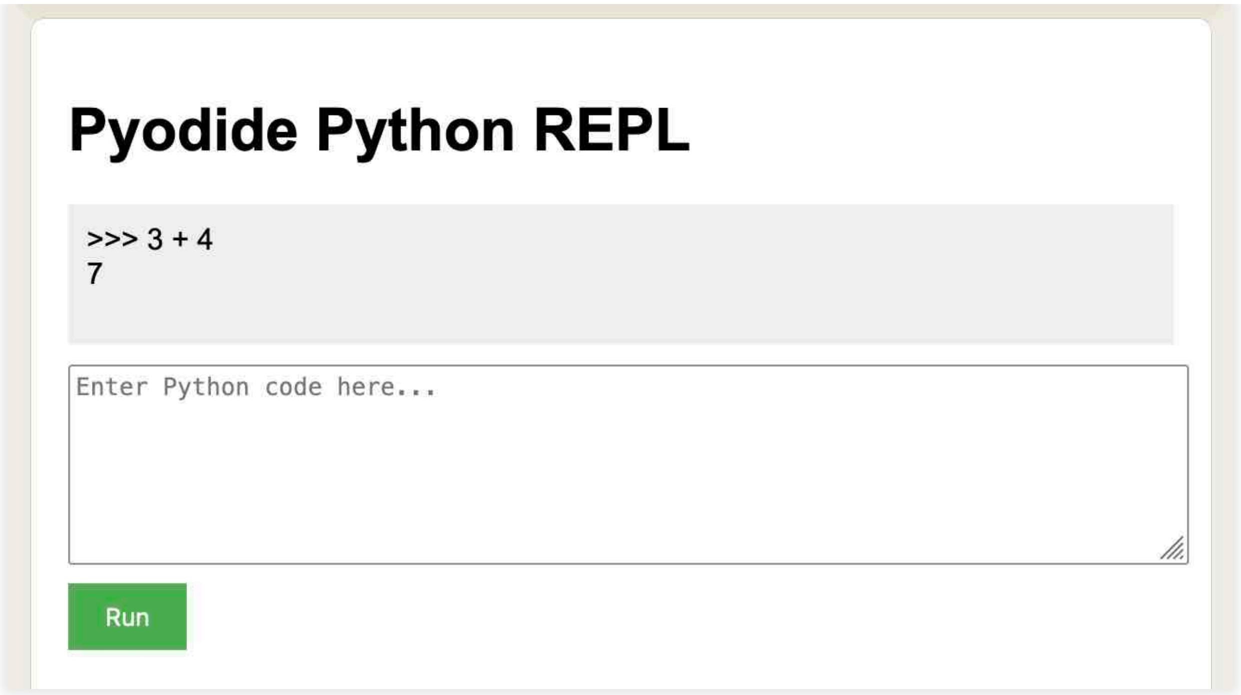
Pyodide REPL

I didn't put a lot of effort into this one. While poking around with Claude Artifacts in the browser DevTools I spotted this CSP header:

```
content-security-policy: default-src https://www.claudeusercontent.com; script-src 'unsafe-eval' 'unsafe-inline' https://www.claudeusercontent.com https://cdnjs.cloudflare.com https://cdn.jsdelivr.net/pyodide/; connect-src https://cdn.jsdelivr.net/pyodide/; worker-src https://www.claudeusercontent.com blob:; style-src 'unsafe-inline' https://www.claudeusercontent.com https://cdnjs.cloudflare.com https://fonts.googleapis.com; img-src blob: data: https://www.claudeusercontent.com; font-src data: https://www.claudeusercontent.com; object-src 'none'; base-uri https://www.claudeusercontent.com; form-action https://www.claudeusercontent.com; frame-ancestors https://www.claudeusercontent.com https://claude.ai https://preview.claude.ai https://claude.site https://feedback.anthropic.com; upgrade-insecure-requests; block-all-mixed-content
```

The `https://cdn.jsdelivr.net/pyodide/` in there caught my eye, because it suggested that the Anthropic development team had deliberately set it up so Pyodide—Python compiled to WebAssembly—could be loaded in an artifact.

I got Claude to spin up a very quick demo to prove that this worked:



```
>>> 3 + 4
7
```

https://claude.site/artifacts/a3f85567-0afc-4854-b3d3-3746dd1a37f2

I’ve not bothered to extract this one to my own `tools.simonwillison.net` site yet because it’s purely a proof of concept that Pyodide can load correctly in that environment.

Photo Camera Settings Simulator

I was out on a photo walk and got curious about whether or not JavaScript could provide a simulation of camera settings. I didn’t get very far with this one (prompting on my phone while walking along the beach)—the result was buggy and unimpressive and I quickly lost interest. It did expose me to the Fabric.js library for manipulating canvas elements though.

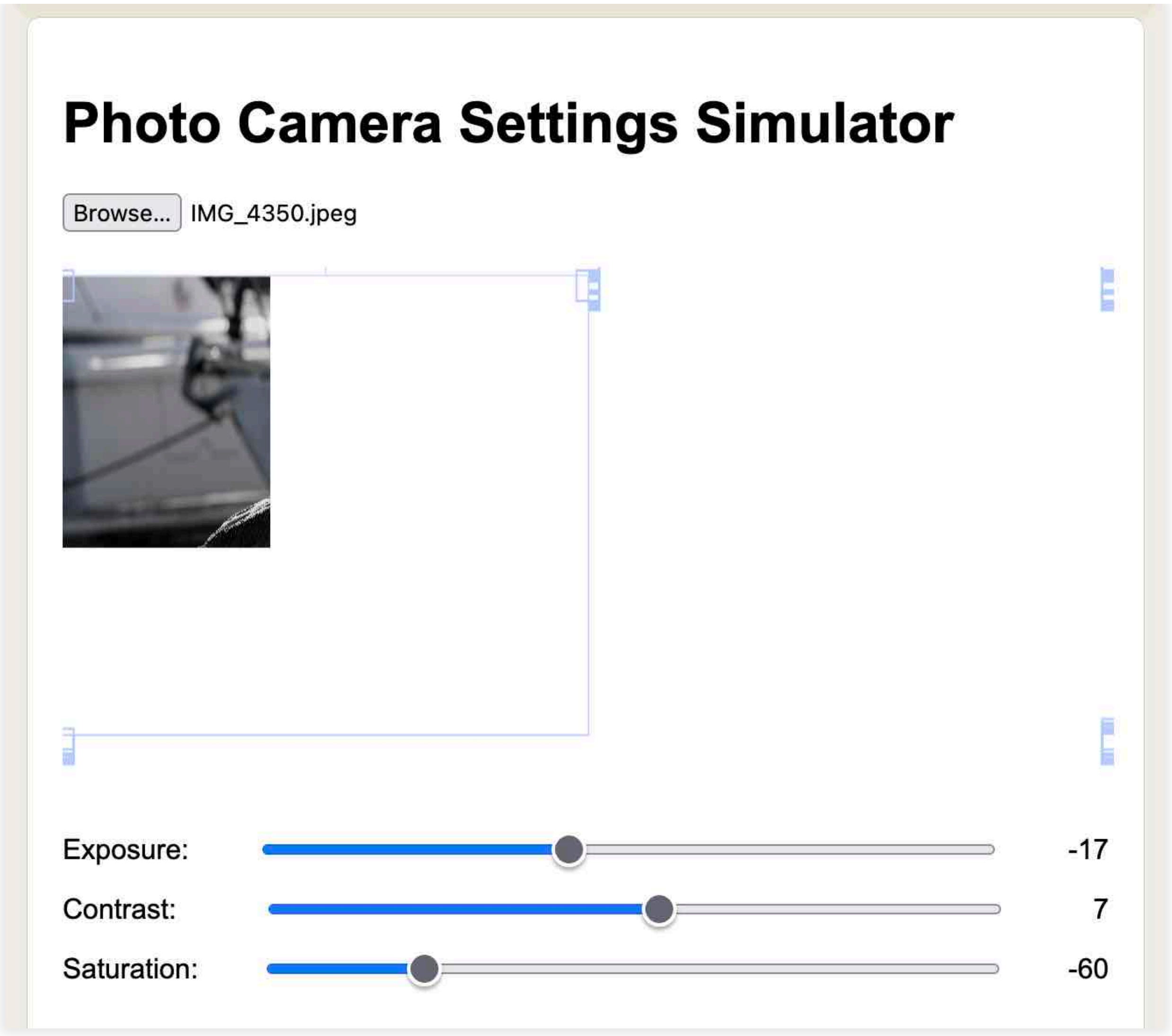


Photo Camera Settings Simulator

Browse... IMG_4350.jpeg

Exposure: -17

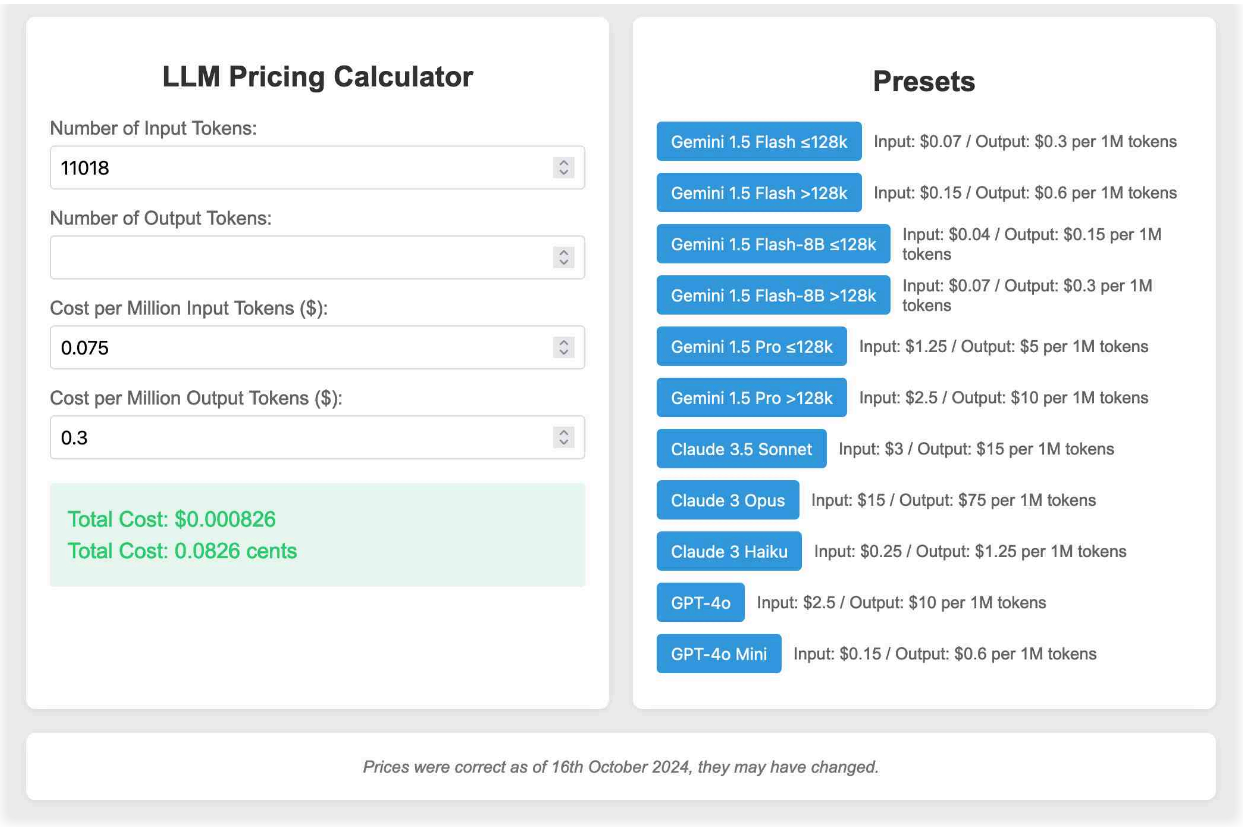
Contrast: 7

Saturation: -60

https://claude.site/artifacts/e645c231-8c13-4374-bb7d-271c8dd73825

LLM pricing calculator

This one I *did* finish. I built this pricing calculator as part of my experiments with Video scraping using Google Gemini, because I didn't trust my own calculations for how inexpensive Gemini was! Here are detailed notes on how I built that.



LLM Pricing Calculator

Number of Input Tokens: 11018

Number of Output Tokens:

Cost per Million Input Tokens ($): 0.075

Cost per Million Output Tokens ($): 0.3

Total Cost: $0.000826
Total Cost: 0.0826 cents

Presets

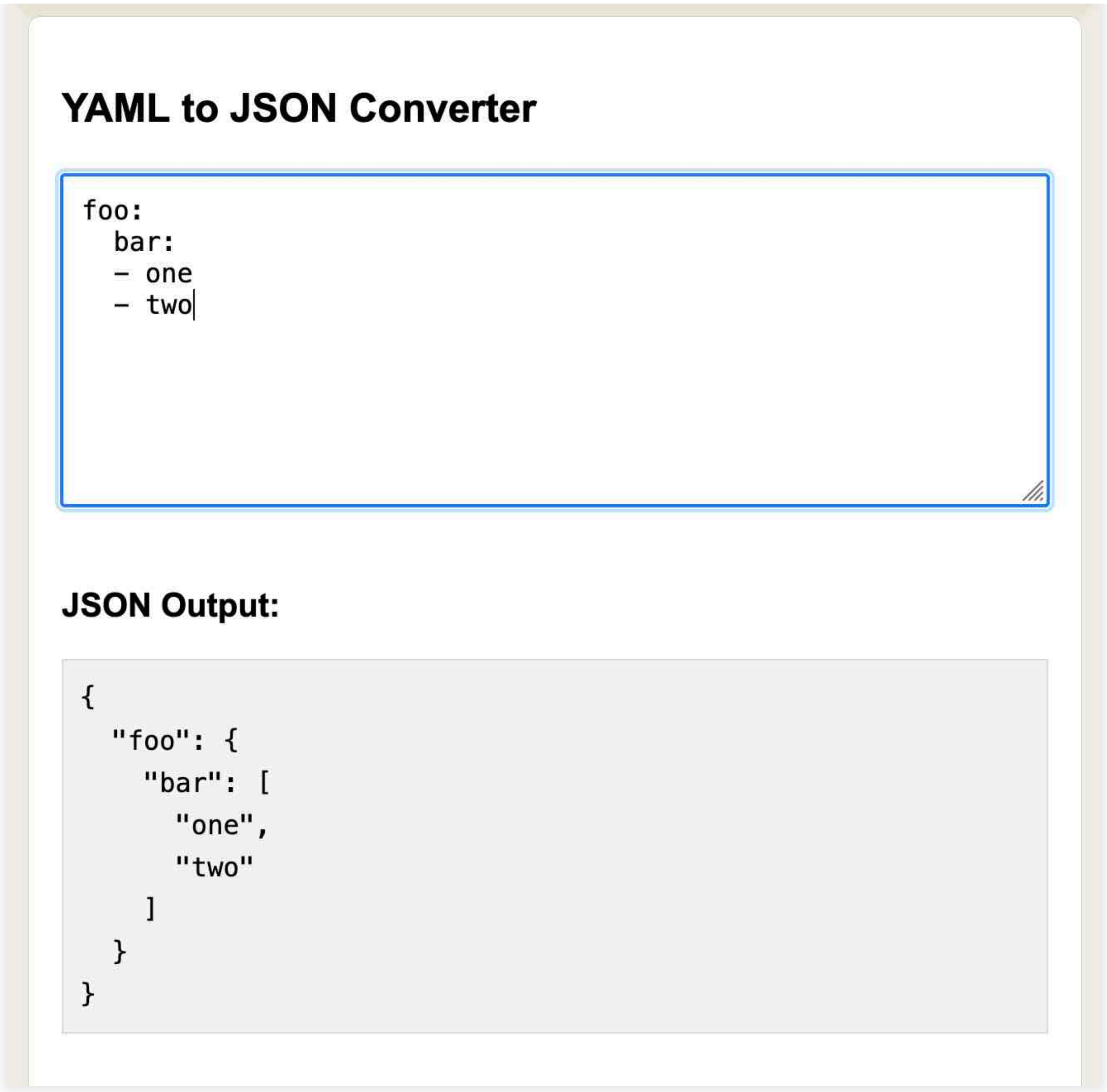
Gemini 1.5 Flash ≤128k	Input: $0.07 / Output: $0.3 per 1M tokens
Gemini 1.5 Flash >128k	Input: $0.15 / Output: $0.6 per 1M tokens
Gemini 1.5 Flash-8B ≤128k	Input: $0.04 / Output: $0.15 per 1M tokens
Gemini 1.5 Flash-8B >128k	Input: $0.07 / Output: $0.3 per 1M tokens
Gemini 1.5 Pro ≤128k	Input: $1.25 / Output: $5 per 1M tokens
Gemini 1.5 Pro >128k	Input: $2.5 / Output: $10 per 1M tokens
Claude 3.5 Sonnet	Input: $3 / Output: $15 per 1M tokens
Claude 3 Opus	Input: $15 / Output: $75 per 1M tokens
Claude 3 Haiku	Input: $0.25 / Output: $1.25 per 1M tokens
GPT-4o	Input: $2.5 / Output: $10 per 1M tokens
GPT-4o Mini	Input: $0.15 / Output: $0.6 per 1M tokens

Prices were correct as of 16th October 2024, they may have changed.

https://tools.simonwillison.net/llm-prices

YAML to JSON converter

I wanted to remind myself how certain aspects of YAML syntax worked, so I span up a quick YAML to JSON converter tool that shows the equivalent JSON live as you type YAML.



YAML to JSON Converter

```
foo:
  bar:
  - one
  - two
```

JSON Output:

```
{
  "foo": {
    "bar": [
      "one",
      "two"
    ]
  }
}
```

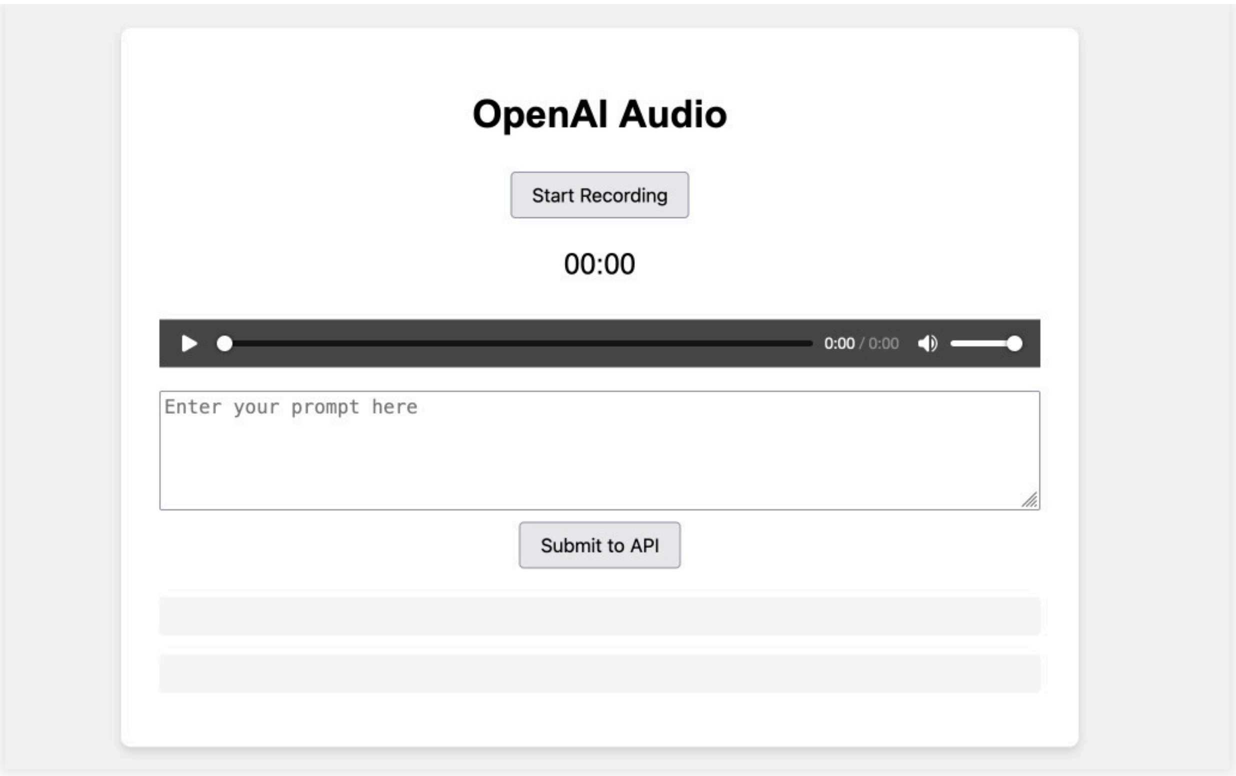
https://claude.site/artifacts/ffeb439c-fc95-428a-9224-434f5f968d51

Claude transcript

OpenAI Audio

This is my most interesting artifact of the week. I was exploring OpenAI's new Audio APIs and decided to see if I could get Claude to build we a web page that could request access to my microphone, record a snippet of audio, then base64 encoded that and send it to the OpenAI API.

Here are the full details on how I built this tool.



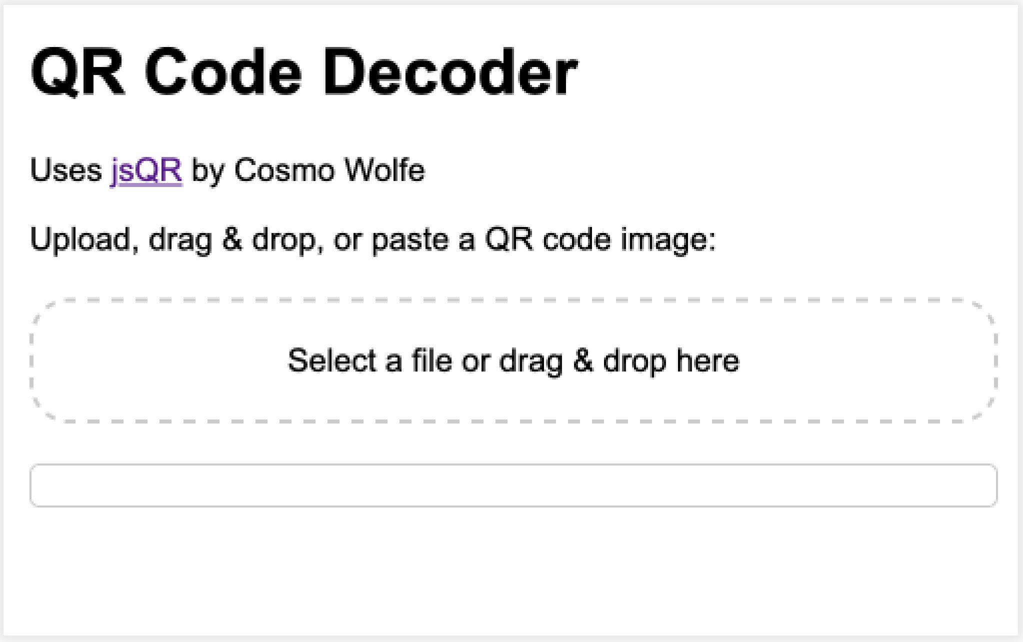
https://tools.simonwillison.net/openai-audio

Claude Artifacts can't make API requests to external hosts directly, but it can still spin up enough of a working version that it's easy to take that, move it to different hosting and finish getting it working.

I wrote more about this API pattern in Building a tool showing how Gemini Pro can return bounding boxes for objects in images.

QR Code Decoder

I was in a meeting earlier this week where one of the participants shared a slide with a QR code (for joining a live survey tool). I didn't have my phone with me, so I needed a way to turn that QR code into a regular URL.


QR Code Decoder
Uses jsQR by Cosmo Wolfe
Upload, drag & drop, or paste a QR code image:
Select a file or drag & drop here

https://tools.simonwillison.net/qr

Knocking up this QR decoder in Claude Artifacts took just a few seconds:

> `Build an artifact (no react) that lets me paste in a QR code and displays the decoded information, with a hyperlink if necessary`

[…]

> have a file open box that also lets you drag and drop and add a onpaste handler to the page that catches pasted images as well

Full conversation here.

Image Converter and Page Downloader

Another very quick prototype. On Hacker News someone demonstrated a neat idea for a tool that let you drop photos onto a page and it would bake them into the page as base64 URLs such that you could "save as HTML" and get a self-contained page with a gallery.

I suggested they could add a feature that generated a "Download link" with the new page baked in—useful on mobile phones that don't let you "Save as HTML"—and got Claude to knock up a quick prototype:

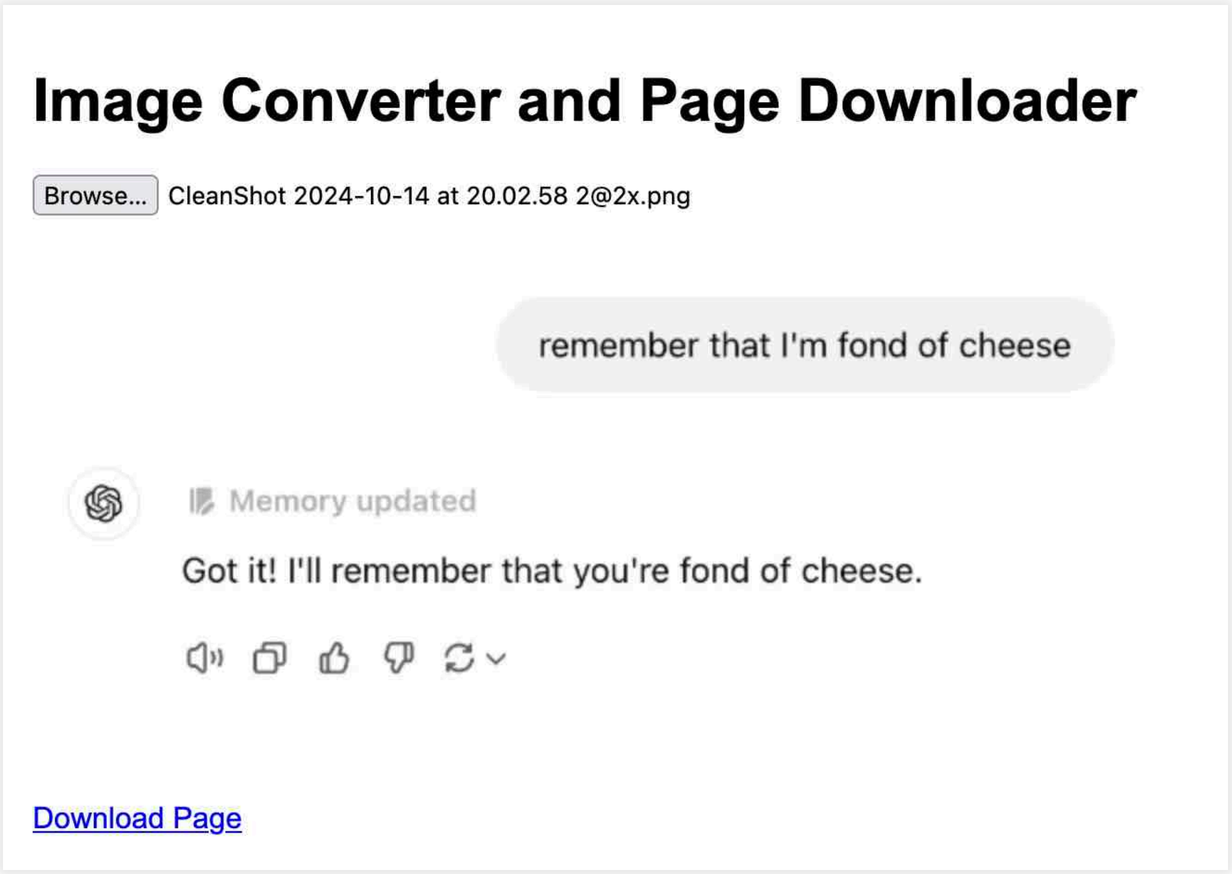


Image Converter and Page Downloader

Browse... CleanShot 2024-10-14 at 20.02.58 2@2x.png

remember that I'm fond of cheese

Memory updated

Got it! I'll remember that you're fond of cheese.

Download Page

In this case I shared the code in a Gist and then used the new-to-me `https://gistpreview.github.io/?GIST_ID_GOES_HERE` trick to render the result:

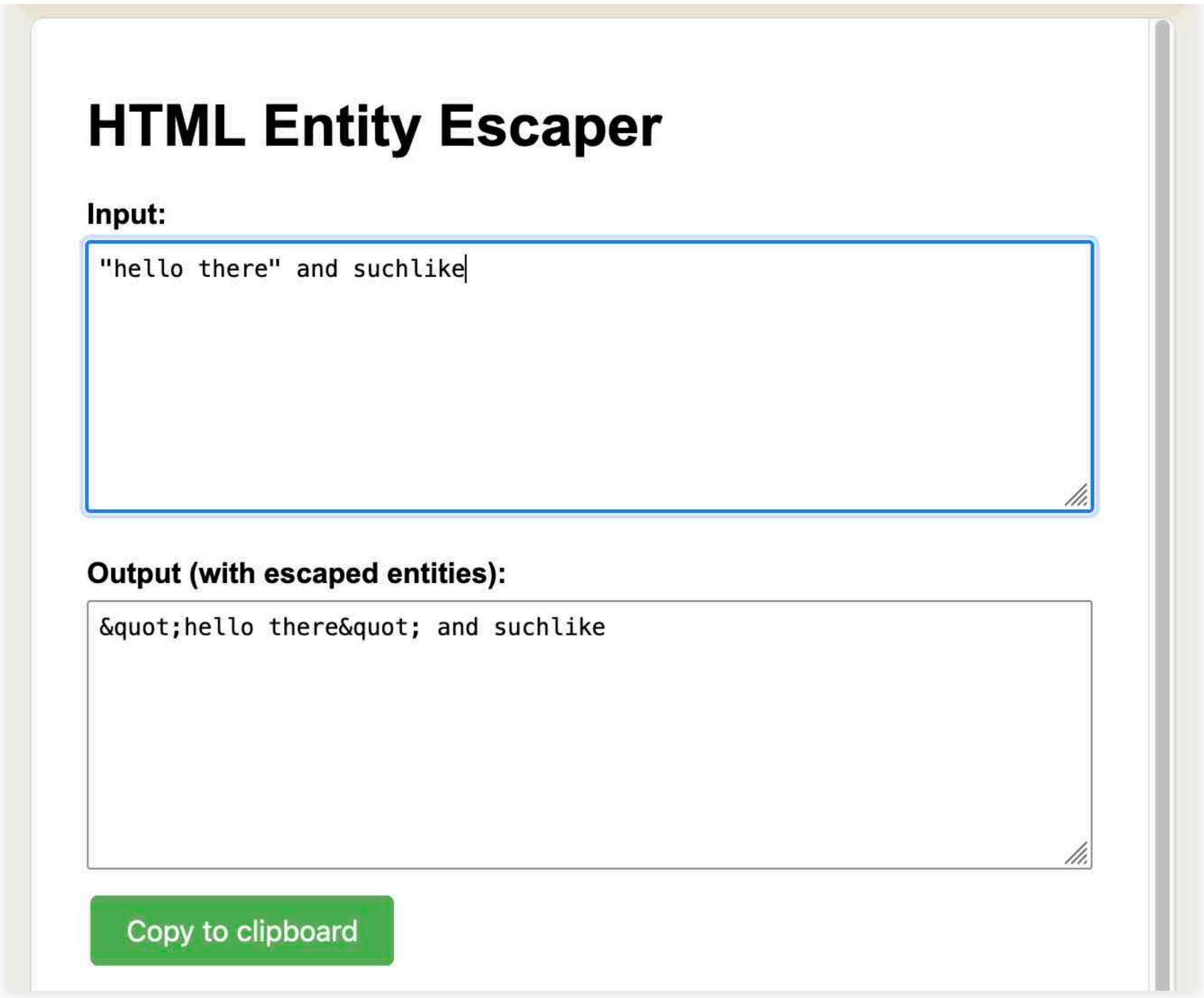
https://gistpreview.github.io/?14a2c3ef508839f26377707dbf5dd329

gistpreview turns out to be a really quick way to turn a LLM-generated demo into a page people can view.

Code, Claude transcript

HTML Entity Escaper

Another example of on-demand software: I needed to escape the HTML entities in a chunk of text on my phone, so I got Claude to build me a tool for that:



```
"hello there" and suchlike
```

```
&quot;hello there&quot; and suchlike
```

https://claude.site/artifacts/46897436-e06e-4ccc-b8f4-3df90c47f9bc

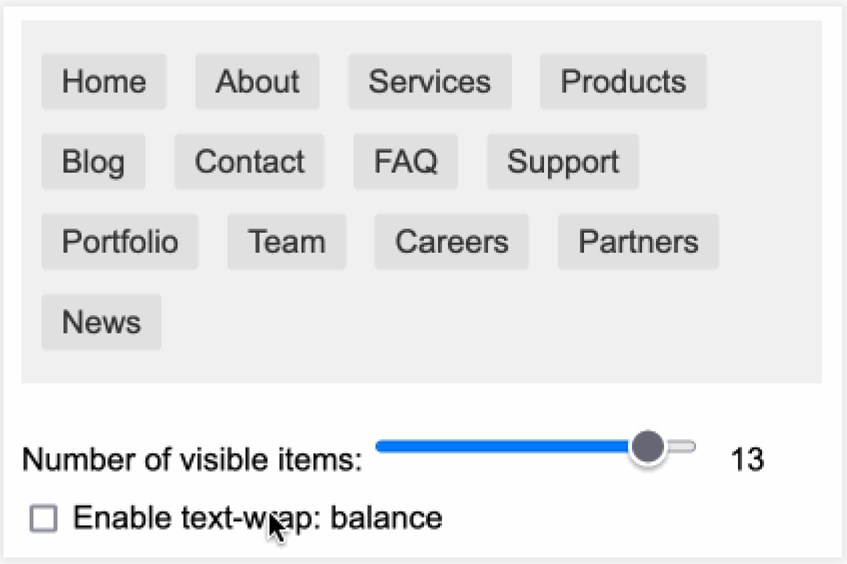
Here’s the prompt I used:

> Build an artifact (no react) where I can paste text into a textarea and it will return that text with all HTML entities - single and double quotes and less than greater than ampersand - correctly escaped. The output should be in a textarea accompanied by a "Copy to clipboard" button which changes text to "Copied!" for 1.5s after you click it. Make it mobile friendly

Claude transcript

text-wrap-balance-nav

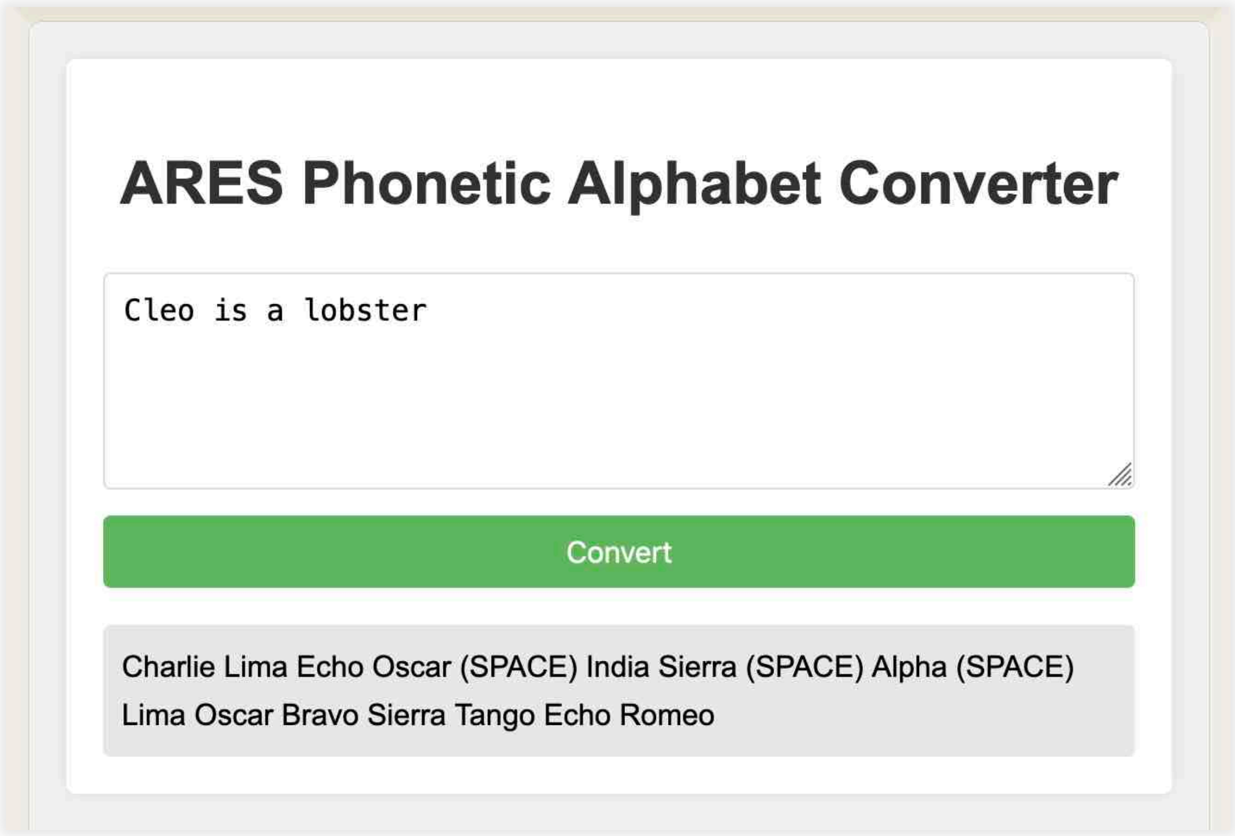
Inspired by Terence Eden I decided to do a quick experiment with the `text-wrap: balance` CSS property. I got Claude to build me an example nav bar with a slider and a checkbox. I wrote about that here.



https://tools.simonwillison.net/text-wrap-balance-nav

ARES Phonetic Alphabet Converter

I was volunteering as a HAM radio communications operator for the Half Moon Bay Pumpkin Run and got nervous that I'd mess up using the phonetic alphabet—so I had Claude build me this tool:



Charlie Lima Echo Oscar (SPACE) India Sierra (SPACE) Alpha (SPACE) Lima Oscar Bravo Sierra Tango Echo Romeo

https://claude.site/artifacts/aaadab20-968a-4291-8ce9-6435f6d53f4c

Claude transcript here. Amusingly it built it in Python first, then switched to JavaScript after I reminded it that I wanted "an interactive web app".

This is so useful, and so much fun!

As you can see, I'm a *heavy* user of this feature—I just described 14 projects produced in a single week. I've been using artifacts since they were released on 20th June (alongside the excellent Claude 3.5 Sonnet, still my daily-driver LLM) and I'm now at a point where I fire up a new interactive artifact several times a day.

I'm using artifacts for idle curiosity, rapid prototyping, library research and to spin up tools that solve immediate problems.

Most of these tools took less than five minutes to build. A few of the more involved ones took longer than that, but even the OpenAI Audio one took 11:55am to 12:07pm for the first version and 12:18pm to 12:27pm for the second iteration—so 21 minutes total.

Take a look at my claude-artifacts tag for even more examples, including SVG to JPG/PNG, Markdown and Math Live Renderer and Image resize and quality comparison.

I also have a dashboard of every post that links to my tools.simonwillison.net site, and the underlying simonw/tools GitHub repo includes more unlisted tools, most of which link to their Claude conversation transcripts in their commit history.

I'm beginning to get a little frustrated at their limitations—in particular the way artifacts are unable to make API calls, submit forms or even link out to other pages. I'll probably end up spinning up my own tiny artifacts alternative based on everything I've learned about them so far.

If you're *not* using artifacts, I hope I've given you a sense of why they're one of my current favourite LLM-based tools.

Posted 21st October 2024 at 2:32 pm · Follow me on Mastodon or Twitter or subscribe to my newsletter

More recent articles

- Project: VERDAD - tracking misinformation in radio broadcasts using Gemini 1.5 - 7th November 2024

- Claude 3.5 Haiku - 4th November 2024
- ~~Weeknotes~~ Monthnotes for October - 30th October 2024

Part of series **How I use LLMs and ChatGPT**

19. Building a tool showing how Gemini Pro can return bounding boxes for objects in images - Aug. 26, 2024, 4:55 a.m.
20. Notes on using LLMs for code - Sept. 20, 2024, 3:10 a.m.
21. Video scraping: extracting JSON data from a 35 second screen capture for less than 1/10th of a cent - Oct. 17, 2024, 12:32 p.m.
22. **Everything I built with Claude Artifacts this week** - Oct. 21, 2024, 2:32 p.m.
23. Run a prompt to generate and execute jq programs using llm-jq - Oct. 27, 2024, 4:26 a.m.
24. You can now run prompts against images, audio and video in your terminal using LLM - Oct. 29, 2024, 3:09 p.m.

javascript 686 projects 434 tools 21 ai 887 generative-ai 748 llms 745 ai-assisted-programming 79 anthropic 92 claude 97 claude-artifacts 20 claude-3-5-sonnet 34

Next: Initial explorations of Anthropic's new Computer Use capability

Previous: Running Llama 3.2 Vision and Phi-3.5 Vision on a Mac with mistral.rs