SVGOMG

- Summary
- What we like?
- Possible Improvements
- Q & A with Jake Archibald
  - Why the web?
  - What worked really well during development?
  - If you could have any API to improve your app, what would it be?

{% Img src="image/T4FyVKpzu4WKF1kBNvXepbi08t52/JBaM0M3E1sG5xURP28m7.png", alt="Svgomg screenshot", width="800", height="1570" %}

Summary

SVGOMG: A beautiful, material, responsive frontend for SVGO.

What we like?

Built by our own Jake Archibald, SVGOMG is an almost perfect example of a fully responsive and capable tool written with web technologies. It features a beautiful Material Design look, ServiceWorker ensures that the app loads quickly and is available offline, and the transitions are smooth on mobile.

Possible Improvements

The only real nitpick we'd have to offer is that the initial UX is confusing due to the main UI missing. Other than that, job well done!

Q & A with Jake Archibald

Why the web?

Laziness. Total laziness. I'm not an expert in developing Windows native apps, I'm not an expert in OSX native apps, nor am I an expert in creating native apps for iOS, Android, Windows Phone or Linux. I can however do the web, and that one skill set let me build something *once* that worked on all those platforms.

What worked really well during development?

I'm really happy with the performance of it. I ensure the page renders before JS is available. In fact, it gets to first render with only 5k of HTML with some inlined CSS and SVG. The main scripts and CSS are all loaded in the background. This means the site appears to load in 1.5s even on 3G with an empty cache, and most of that is DNS and SSL.

The opening screen is really simple, so doing that in 5k wasn’t a challenge. It really bothers me that so many sites wait on JS for their first render, some even require their JS to make further requests before rendering. This pushes 3G render time towards 10s – as a mobile user I know I wouldn’t put up with that.

The main JS is 15k, but doesn’t include the parts that parse and minify the SVG, that’s loaded as an extra phase in the background. It’s great because interactivity lands really quickly, and the user doesn’t notice the extra loading. If the user manages to select an SVG before that script is available, the loading of that script appears to be part of the processing time.

I also used ServiceWorker to make the whole thing work offline. Working offline is a pretty cool feature, but I’m mostly doing it for performance. Subsequent visits to SVGOMG render almost instantly, whatever connection the user has. Given the variations in mobile connectivity, that’s really valuable!

If you could have any API to improve your app, what would it be?

I used Babel so I could make use of future JavaScript stuff. It'd be great to have some of that working natively in the platform. Specifically, async/await, arrow functions, argument defaults and destructuring.

I had to use a library to gzip the output to find out its gzipped size. Using a library for this is kinda annoying as that code is already in the browser for HTTP stuff, there's just no API to it. Ideally it should be some kind of transform stream so I can count the size of the output without having the whole thing in memory.

□□□ #2
□ □□□ □□□ 2026-10-31 22:20:01 UTC
□ □□□ □□□ 2026-10-31 22:20:03 UTC