Trying Cloudflare Pages: Best Server Tech Since cgi-bin

being sold short by subpar documentation

Posted on October 23, 2022 | *Taras Glek*

Cloudflare Pages 😍

I got into Cloudflare pages (https://developers.cloudflare.com/pages/) because it lets you host static websites with all the latest optimizations conveniently and for free. Setting up a static website with a custom domain is easy, and you get the benefits of automatic SSL, CDN, and latest version of HTTP3.

This is how this blog is hosted and I have a few other projects like this. I love the fact that I can get a super-fast static website up and running in a few minutes and I don't have to worry about infrastructure details.

Recently, I needed to add a serverside component to one of my projects in order to generate twitter cards (https://developer.twitter.com/en/docs/twitter-for-websites/cards/overview/abouts-cards). This required a database to dump pictures + metadata into (I love supabase (https://supabase.com/) for that). I also needed server-side functionality to serve the twitter card metadata + user screenshots.

Cloudflare Functions: Holy Documentation Batman! 😱

I started reading about Cloudflare functions (https://developers.cloudflare.com/pages/platform/functions/). It's cool that you can extend your static website with server-side JavaScript code. You just make a `functions` subdir and drop code into it. I managed to get hello-world function to work using the wrangler (https://developers.cloudflare.com/workers/wrangler/get-started/) tool. The setup was somewhat finicky, but not too bad.

Then I immediately hit a deadend in that the docs are terrible at explaining the provided API. The docs read more like incomplete corporate documentation wiki that your coworker is gonna guide you through than a public product (but I wasn't given a coworker or paid for this!). In provided samples there were no import statements to follow to delve into the library source code. Apparently Functions are some bastard relative of Workers. There was also some concept of corporate-sounding middleware. I was a really confused about how to get started beyond the simple hello world.

As a shortcut, I used GPT-3 to generate a basic typescript function for me. This let me look at TS type definitions and get a better idea of what's available so I could get developing.

Cloudflare Functions: OMG, these are actually good! 😍

Turns out middleware is a function that lets you proxy/rewrite static assets being served (including overwriting request/response headers). This made it easy to write a function that inserts custom markup in my static react site(pointing at preview images that users store in db). This combined nicely with fact that Pages serves same SPA for every http path instead of a 404. I'll probably rewrite this as a function that serve a simple twitter-bot-oriented-HTML, but was cool to finally try a proxy API that's pleasure to use.

I also wrote a simple function to serve preview images from supabase PostgREST API (https://supabase.com/docs/guides/api). I'll eventually redo this to use Cloudflare R2.

I defined the url + rewriting mapping for both functions using their clever filesystem-naming scheme.

After testing everything with wangler, serverside deployed and worked on first try! I'm so impressed!

Sample Code 🥊

I would like to be able to use it for my next set of projects, here are some sample code snippets to help me and others:

`functions/screenshot/[[path]].ts` resolves to `mydomain/screenshot/.*`

```
export const onRequest: PagesFunction = function onRequest(ctx) {
  let response = `Hello, world! ${ctx.request.url} ${ctx.request.method} ${JSON.stringify(ctx.request.headers)}`
  return new Response(response);
}
```

`functions/shared/_middleware.ts` rewrites content shared at `mydomain/shared/.*`

```
let handler:PagesPluginFunction = async (ctx) => {
  try {
    const response = await ctx.next(ctx.request, {headers: ctx.request.headers});
    let responseText = await response.text();
    responseText = "Hello, world! "+responseText.length;
    // Cloudflare gets upset when specify 304 and include a body, even if it's ""
    return new Response(response.status == 200 ? responseText : undefined, response);
  } catch (thrown) {
    return new Response(`Error serving ${ctx.request.url}: ${thrown}`, {
      status: 500,
      statusText: "Internal Server Error",
    });
  }
}

export const onRequest = [
  handler,
  // can chain more handlers here
];
```

Conclusion 🤗

In addition to awesome static hosting on Cloudflare Pages, one can add dynamic functionality dropping a few TypeScript files into a `functions`, then git `commit` + `push`. This is magnitudes simpler than any prior server stack I've tried. The free tier is generous, after the free tier the price seems lower than AWS.

I absolutely love the experience of zero-effort git-ops style static websites with serverless functions. Cloudflare Pages + Functions are now my favourite solution for deploying hobby projects, by far. I'm looking forward to writing more backend functionality in this style.

Last time serverside was this fun to write was when Apache served static pages and you could write cgi-bin scripts in any language of your choice, but we didn't have git or npm back then. I'm glad we have this now.

But, dear Cloudflare get your documentation together. Maybe you can just copy MDN (https://developer.mozilla.org/en-US/)? I got some stuff to work, but I still have questions as to how you intend me to package libraries into my functions, etc. Would love examples of every single API call you offer. Maybe you could let us in the kinds of hardcore cleverness your middleware enables?.

HN Comments (https://news.ycombinator.com/item?id=33307950)

Tags: #webdev (https://taras.glek.net//tags/webdev/) #cloudflare (https://taras.glek.net//tags/cloudflare/) #supabase (https://taras.glek.net//tags/supabase/) #serverless (https://taras.glek.net//tags/serverless/)

← **PREVIOUS POST (HTTPS://TARAS.GLEK.NET/POST/CURIOUS-CASE-OF-MAINTAINING-SUFFICIENT-FREE-SPACE-WITH-ZFS/)**