Server Side Rendering FTW

Adrian Kosmaczewski

2023-02-10

I am, I have been, and forever will be a big advocate of server-side rendering. I think it is an essential way to build dynamic web content. I believe in this adamantly, feverishly, strongly, and relentlessly.

JavaScript is not made for rendering HTML. The fact that it can modify the DOM does not mean you *have* to use it for that. Just don't. HTTP requests are meant to return HTML. Web browsers render HTML just fine. Do that.

And I'm not the only one[1] who thinks like that. We have been lied by a full industry claiming that using JavaScript on the client (and the server! FFS!) to request JSON data, and then to manipulate the DOM accordingly, was the best possible way to create web applications.

What. The. Actual. Fuck.

The creators of those "SPA frameworks" actually bullied people[2] who dared speak the truth of the utter insanity of using such systems. Talk about mafia tactics.

The results of SPAs are abysmal. The web has crawled to a halt, crippling the user experience of everything we use on the web.

The drawbacks for end-users are many:

- Broken back button functionality.
- Nonworking stop buttons.
- Really slow updates when state change.
- Inconsistent page states.
- Bookmarking and sharing URLs with others.

And to add insult to injury, a total lack of proper error handling and feedback when network connections die. Which hey! Happens more often that we would like to acknowledge in 2023.

Even smaller apps, that would be best served with server-side rendering, fall into the trap of frontend evangelists trying to sell their shit.

Ironically, the "IT Crowd"-like solution for this kafkaesque situation consists of… reloadinng the whole page once again. So much for a "single page application," dammit. Remember when your navigator would do that for you every time you navigated from one page to the other? Yeah.

[1] https://infrequently.org/2023/02/the-market-for-lemons/
[2] https://fediverse.zachleat.com/@zachleat/109830047951867907

And even for developers, the tooling is maddeningly slow. JavaScript is a bad choice for this. I can rebuild this website from scratch using Hugo[3] in less than a second. On the other hand, frontend applications take ages to build, using half-assed tools like gulp, npm, or whatever the build tool *du jour*.

Oh! But you can share (usually a few small, quite insignificant data classes) JavaScript between both client and server. Wohooooooo!

We're a brain-dead industry. I'm fucking tired.

Making web apps? Mustache templates[4] and similar technologies are the way to go. They have to reclaim their due place as the proper mechanism to render HTML on the server. Using programming languages like Go, Crystal, C#, or Rust, and so many others[5], means you can have blazingly fast performance on the server, while letting browsers do what they do best: display HTML and CSS.

And yes, *if you must*, sprinkle some JavaScript here and there if it *really* helps. Not just for the sake of it; but when it adds real value to the end user experience.

For how long are we going to have to cope with the shitty web experience of 2023?

Update, 2023-02-17: More[6] about the subject of stopping with this SPA madness.

[3] https://gohugo.io/
[4] https://mustache.github.io/
[5] /blog/fortune-apps/
[6] https://dev.to/oxharris/rethinking-the-modern-web-5cn1