If you've ever built a full stack application, I'm certain you know one thing. Environment variables are hard, like way harder than they should be, making sure you have the right ones in the right place, making sure that your package is building correct locally and on your server, your build systems, your CI, and everywhere else. It's hard to know if you even have the environment variables, much less that they're correct, format it as expected, and accessed correctly across your codebase. The goal of this new package is to solve that, because we've had this problem a ton too. In Create T3 app, we built a system to make it easier to validate your environment variables, but we realized that was too good for just Create T3 app. And people outside of our ecosystem, even outside of next as a whole, should be able to have confidence when they use their environment variables. What we've built is, and I should also be clear, when I say we, I almost entirely mean Julius, I had nothing to do with this beyond nudging him and giving him a domain. So yeah, Julius killed on this project. I'm really excited to show you all how cool and powerful T3 environment is. Here is T3nv, framework agnostic validation for type safe environment variables. I love the little red squiggly line under type safe, really emphasizes the point. Nice, simple site, easy to install, just add T3OSS environment core, and if you're using next, you can add the next JS package as well. And here's an example of how you use it. You create an ENV file where you create your ENV. If you have a client prefix like the next JS, you can have next public. You can specify here. These are the client only environment variables, and this is the prefix that they're all expected to have, and everything else is assumed to be a server environment. And as we scroll here, you see it's pretty easy to use this. You know, it's import ENV from this case, they have a path alias for the tilde slash, and now you have your environment variables guaranteed to be validated. And if you use the next package, I'm assuming it just wraps things automatically. I'm not sure if it's actually smart enough to do that. Here's the further config. Unlike the core package, runtime ENV is strict by default, meaning you have to, you structure all the keys manually as due to how next bundles, environment variables, and only explicitly access variables are included in the bundle. My only concern here is that I don't know if this solution will validate on build, but I am certain that julias was smart enough for that to be the case. There's only one way to be sure. Let's test it. People are saying it validates on build. Well, let's go make a new project with it regardless. Yeah, you should add the important next config. That's what I was thinking. We're actually going to do something you might not expect. We're going to create next app at latest with it. We'll do an ENV test. I'm going to use the app directory because it's the only way to know how to code in next anymore. Let's go install this package. The example uses PNPM, but I was lazy. I just used NPM here. I'm going to copy paste NPM install all kinds of so much slower than PNPM. I'm way too spoiled. Now that we have the package, let's go configure it. Examples are pretty simple, but if we go here, we can... you know, link, but ENV. So let's change these up a bit. We'll do server only, z dot string dot min. Well, say it has to be five characters, and we'll have next public client to z string dot emoji. What's it? There's a runtime environment. So you have to put them all in here as well. Is that correct? Ah, that's actually clever because next is stupid. That is an annoying quirk with how bundling is working with it. I'm happy it's down to just this one quirk, but I'm hopeful that if we we show this to the next team, they can help us figure something out. Otherwise, we won't have them on the edge. This will work for node bundles, but it won't work for edge. Does that sounds like a bug? And I can get them to fix that. This is an example of one of the weird things we have to do to get around next bundling patterns, but I'm committed to getting the team to fix that. So in the future, we might not even need this, but for now, it's the one remaining wart. Yeah. So by the size of it, this is optional if you're not using edge. Good to hear. Anyways, let's actually use it. So this is a server component example, because it's using the app router, which means it's pretty easy to see things that are server only because by default, everything is server only. So if in here, I was to const thing equals that silly. And here is server only as to console dot log only on server thing. And we're to go run this npm run dev. Yeah, as well as concerned about it's not checking there, but this isn't going to be valid. And if we go to walk close to 1000, y'all will see that it's going to error because there's an invalid environment variable. We don't have this server only or the client only thing. However, it didn't error during build. And I personally would really like it to error during build. So how do we fix that by going to the next config JS? To a reason, rename that as next config, yes. Nope. I'm not smart enough to solve this problem. Just mjess it. That's an easy enough solution. Again, this is why I this create t3 app and I need to go back because it makes all of these decisions correctly. Can't somebody say this can be ts now? I'm just going to next config ts. Wrong file. Nice. Oh, I know. I thought it was I thought it could be ts. Why do I think that? God damn it. Dot mjess it is nice. Cool. And we'll be sure to make the docs nice and clear. So stupid people like myself don't have to run into all these problems. Make sure that your environment variable stuff is dot mjess both the next config as well as the file where you create these things because next config doesn't support TypeScript yet. Anyways, what we can do, well, more importantly, when we try and build, we'll now see it fails because the environment variables aren't here, which is really convenient, especially when you're onboarding new developers to your team, who might not be familiar with the environment and might not have everything set up correctly. Now, they get a very clear and obvious error, but these environment variables are missing. So let's go out and really quick new file dot e and v dot local. So that's what's supposed to be, I think I said, five plus characters. So nope, is not that long. And the other one's supposed to be an emoji. So we will also make this one. Nope. So that's run this again. And we'll see now we're getting different errors. Server only string must contain at least five characters and next public client is an invalid emoji. Very useful getting real errors in here. Like as soon as you run the project the first time, you don't even get to the point where it's running and then you run into a weird error halfway through, you would just immediately get told how to handle this. So now let's fix it and actually use these things. So longer string and this one and emoji. And now export defaults. Yep. Thank you. Again, just use create three apps. Don't don't do my stupid decision here. But now this works. And if it works to build, then it will also work in the build. And here, I'm not rendering anything, but we can see here only on server longer string. And if I want to use that on client, since that's a server component, actually, I'll just show you guys in the console here, that doesn't log because it's a server component that only runs on server. And if you were to try to access this on client, you would get an error. And if I just put this here, it will go to client, but it will never log because again, server components. So the only way to leak your environment variables with this is to return them in the markup or as a prop that gets sent to the client. Pretty hard to do. Really nice overall. I've been impressed. And obviously, if we do our ENV.next public client too, then we get our emoji. Pretty dope, right? I think this is huge. You know, and already go check out ENV.t3.g. You can also check us out on GitHub. And I will be sure to put that link in the description. So please check us out. Give us a star. This package is awesome. And it's so cool taking the best lessons we have learned from create.t3 app and bring those to the rest of the JavaScript ecosystem. There are a few projects that wouldn't benefit for something like t3 environment. And I highly recommend checking it out. If you want to learn more about our philosophy with create.t3 app, I'll pin a video in the corner here where I talk more about that. Thank you guys as always. Peace, nerds.