What is Blitz.js?

Written by Seth Corker on Benevolent Bytes

What is Blitz.js?

Blitz.js is a new framework that's built on Next.js. It's positioned as a Rails-like framework, it's monolithic and focused on developer productivity while using the modern JavaScript tech you're used to. Although new, I'm watching Blitz.js as I believe it has great potential to refresh JS fullstack development, it looks like a great way to dive in without the hassle of decision fatigue and complicated configuration.

Why does Blitz.js exist?

Figure 1: illustration of woman in lotus pose hovering above yoga mat

Web development has evolved a lot over the past decade. There are more libraries and frameworks for JavaScript than ever before but there's been a trend to opt for smaller libraries, decoupling and microservices. These all have benefits but one of the tradeoffs is productivity. Blitz.js is a reaction to this, it want's to bring back the simplicity of the web with modern tooling. It takes some of the powerful tools you use today and packages them up into a nice and easy to work with bundle.

Simplicity

I enjoy the modern web. There's many ways to get things done the way you need to. You can choose the right libraries for your particular project and requirements. One critique of this however, is decision fatigue. With so many options, what do you choose? When starting a new project you will constantly make decisions about each and every library, tool and weight their strengths and weaknesses.

1. Should you make a SPA or use SSR?
2. Will you use REST or GraphQL?
3. How will you manage your state? Redux? MobX?
4. What view layer do you want to use, React, Vue or maybe Svelte?
5. How will the project be built? Webpack? Rollup?
6. Which features of JS do I want to use and which babel plugins do I need to add?

These decisions are before you've written a line of actual application code, it's all setup. It's no wonder beginners can get overwhelmed and web veterans have can get disenchanted with the direction modern web development is going. Tools like Create React App are a reaction to this just like Blitz.js. The benefit is simplicity while maintaining the right to pick and choose.

Small companies

A lot of existing libraries are created for large companies that have different problems to smaller companies. Sometimes it doesn't make sense to adopt certain technologies because they just don't solve the problems small companies are trying to solve. Technologies like GraphQL and frameworks like Relay or Apollo fit together to solve problems which you might only run into at large scale. Blitz.js looks to be your-go to tool until you need to scale. Even then, it's built on other technologies that are proven at scale so the leap might not even be that great. It's a foundation that grows with your needs when you need to grow.

Why am I excited?

Figure 2: illustration of person in jetpack

The *good* old days of web development

I started my career as a web developer redesigning and maintaining websites. During the early days I did everything by hand. The HTML was handcrafted, so was the CSS. What little JS which was added was mainly for a sticky header or some mobile optimizations. Once I had a version which was ready to deploy, I'd connect to the server over FTP and copy the files over. Where they simpler times? Probably. Was I more productive than I am today, not really. It may have been simpler but there was a lot of intensive processes around the code. Adding a header and footer to every page was manual, the changes to would require a massive find and replace across every HTML file.

I never knew about source control so manual backups had to take place after every change. Over time, I evolved my process and new tools came out to make it easier to achieve. Copy and pasting header HTML was replaced with templating and a build step. I traded some additional complexity for developer productivity. Over time I integrated Gulp and Bower to ease minification, browser compatibility compilation of SCSS. Why am I telling you this? Everything in programming is a tradeoff and you need to find the right tradeoffs for the things you're building. I evolved over time as a developer and my tooling evolved too. Blitz.js looks like a way to bring back the simplicity you remember having with modern tools and the benefits that come with them.

My first fullstack experience

My first foray into fullstack development was Ruby on Rails. The reason I chose it despite not knowing Ruby at the time was developer productivity. I needed to create something I had never done before, I needed a new approach because my cobbled together web tools only took me so far and I'd never worked with databases, an API or CRUD outside of the classroom. Despite the odds stacked against me, I managed to learn and be productive with Rails. It was batteries included and I owe it a lot. It was flexible enough to allow you to get stuff done and opinionated enough that made it easy to figure out the *right* way to do it.

This was a big contrast to my next job where React powered the frontend, the backend was a RESTful API with no ORM. There were benefits but there were also times I missed the simplicity of Rails. I thought of going back for side projects but I'm too invested in the JS ecosystem, that's where I'm most productive and I don't want to leave it behind. Blitz.js might be the best of both worlds. A different take on Rails for JS. It's Rails but with React built in.

Why does Blitz.js have a future?

The JS ecosystem is vast and powerful, there is great tooling and libraries available to suit almost any need but the challenge comes in choosing these tools, configuring them correctly and combining them while being productive. Blitz.js does this work for you, the tools exist and they've been configured for you. What I think gives Blitz.js a good future is that it is built on what already exists. It leverages other ecosystems to thrive.

Figure 3: illustration of person holding a laptop while being lifted up by three arrows

Next.js

Next.js is a powerful framework in its own right. By leveraging this, Blitz.js can build on top of this solid foundation and get TypeScript support, routing, code splitting and more for free. As Next.js evolves, Blitz.js can too.

Prisma 2

Blitz.js piggybacks off the work done by Prisma. Rails had a great ORM which I liked a lot, Prisma is step above that which allows for more flexible data modelling and is setup to work well with TypeScript.

CLI

My favorite feature of Rails is the CLI scaffolding. As a beginner to fullstack development, Rails made it easy to generate everything you needed to get an entire working MVC setup. With a single command a model would be created along with a controller and all the CRUD views you desired. The CLI is what brings everything Blitz.js has to offer, together in one easy to use place.

Community evolution

Blitz.js has a manifesto which defines one of the most important tenants, “Community over Code”. This is a simple idea but it’s powerful, it sets the stage for a constructive community that learns from other communities rather than competing with them. Part of this commitment includes some transparent development practices, there already exists an RFC for the Blitz App architecture. This means you can have a say in how Blitz.js evolves and what choices should be made.

Is it ready for production?

No. Blitz.js still lacks maturity. It’s early days so expect the APIs to change a lot. The documentation is limited but if you’re brave though, Blitz.js utilizes existing tech so much that you’ll probably be able to make something production ready with some extra time and effort. Nevertheless, I’m excited to see Blitz.js grow and evolve - I hope it can be the Rails for JS developers.

Resources If you’re interested in finding out more, take a look at some of the official places down below.

- Blitz.js Repo on GitHub
- User Guide

---

illustrations provided by ManyPixels

Figure 4: person in parachute steering