# Devansh Jethmalani

I make softwares and things that help making them.

Mobile: +91 9152035616 Website: devanshj.me Email: jethmalani.devansh@gmail.com
GitHub: github.com/devanshj X: x.com/devanshj

## Summary

I've been coding since I was 13 years old. It's been 12 years now. In these years I've explored various branches of software development across platforms. However, front-end remains my core competency, which is why I'm applying for this role.
Apart from making softwares for users, I've also made "things" (tools, libraries, articles, etc) for other software developers. If you're a technical person reading this, I have a few references for a quick assessment. Check them out and you might just skip reading the rest of my resume ;)

1. I'm a Pro TypeScripter. See my experimental PR "Feature: Self-Type-Checking Types" for TypeScript and my library @sthir/predicate.
2. I've a deep understanding of React's programming model. Read my article called "Hooks: React's do-notation" which was on the front-page of HackerNews.
3. Watch the case study of my project "SBK Panch Elections" which has two technical deep dives: "React without React" and "Modals with Native UX".

## Key Skills

JavaScript, TypeScript, ReactJS, CSS, HTML, NodeJS

## Technologies I've worked with (non-exhaustive)

**Languages:** JavaScript, TypeScript, Java, Kotlin, PHP, SQL, Python, C++, C, HTML, CSS, SASS.
**Runtimes, Platforms, Databases**: Browser, NodeJS, Android, React Native, ElectronJS, Terminal, Windows.h, PostgreSQL, MySQL, MongoDB, Firebase, Heroku, Dokku, AWS Lightsail.
**Frameworks, Libraries, Design systems:** ReactJS, NextJS, AngularJS (v1), RxJS, MobX, jQuery, JavaFX, ExpressJS, Heroku's Oclif, Jest, TailwindCSS, Palantir's Blueprint, Uber's BaseWeb, Shopify's Polaris, Semantic UI, Bootstrap.
**Tools:** VS Code, Figma, Adobe Illustrator, Adobe Photoshop, Nomnoml, Excalidraw.

## Experience

**Hobbyist, Open Source Contributor, Freelancer • 2013 - Present**
See my projects and contributions below

**Front-End Developer, IV Consultancy Services Pvt Ltd • Mar 2024 - Present**
I was primarily on the reports team. I worked closely with the consultants team to understand customer's requirements and implement them. I've implemented 50+ custom reports which directly impacted the customer's satisfaction with the product. I also single-handedly implemented an internal WYSIWYG tool through which consultants themselves can tweak reports without writing any code hence not needing a developer intervention. This resulted in bringing down the turnaround time to make a change from days to minutes. Throughout my time here I've used technologies like React, TypeScript, JavaScript, HTML, CSS, jQuery, MobX & MobX-Keystone, RxJS, Lexical, Vite. I have also gone beyond my scope and worked a bit on the backend with technologies like Java, Spring, and MSSQL.

**Front-End Intern, Groflex • Jan 2024 - Feb 2024**
Helped develop a business ERP SaaS using technologies like React, Redux, React-Router, AgGrid,Bulma and SASS

**Front-End Intern, Servify • Dec 2019 - Jan 2020**
Helped develop an internal finance tracking tool using technologies like React, Redux, Redux-Thunk, React-Router, Axios and SASS.

## Featured Projects

Over the past 12 years, I've built more than 30 projects. Here are the most relevant front-end focused ones...

**@devanshj/rex, Personal Project, 2022**
A library that lets you write Functional Reactive Programming (FRP) using React Hooks. Technologies used: React, TypeScript. Source code: github.com/devanshj/rex.

**InquireScript, Course Work, 2020**
A web and cli app that takes a program written in JavaScript and converts into a survey-like form. Technologies used: React, TypeScript, Firebase, Uber's Base Web, Heroku's Oclif, Create React App. Source code: github.com/devanshj/InquireScript.

**SBK Panch Obituary Creator, Community Project, 2018**
A web app to create obituaries. Technologies used: AngularJS (v1), Bootstrap 4, jQuery, NodeJS, ExpressJS, Firebase Hosting, Gulp, Adobe Illustrator

**Happy Table, Personal Project, 2017**
A data table component based on AngularJS (v1) and Semantic UI. Technologies used: AngularJS (v1), Semantic UI, jQuery. Source code: github.com/devanshj/happy-table

## Featured Open Source Contributions

I've contributed over 80 Pull Requests in my open-source journey. Here are some notable ones I'd like to highlight...

**Zustand**
I was involved with Zustand (5M+ weekly npm downloads and 50K+ stars on github) to improve the developer experience by inferring the types instead of having to explicitly declare it. This was challenging as the api involved mutations which are almost impossible to encode on the type-level. Yet I found a way to do it, which is outlined in the "Proposal: Encoding store mutations type-level" issue and implemented in the "breaking(types): Add higher kinded mutator types" PR. The whole effort including deprecations, migration path, documentation, etc took 2-3 months and over 30 PRs. You can find all of them here.

**XState and @cassionzen/useStateMachine**
Like Zustand, XState (1M+ weekly npm downloads and 25K+ stars on github) too didn't have a great developer experience as it didn't have much type inference and type-safety. And just like Zustand, typing XState was also nearly impossible because one would require kind of running the whole statechart on type-level. Which is exactly what my library TXState did with almost 2500 lines of type-level TypeScript. I could never really complete it or merge it with XState but it helped XState authors gain confidence and they wrote a type generator for which I helped them a bit. XState was gracious enough to pay me for this "TypeScript Consultancy". While I was building TXState I came across another library called "useStateMachine" which had the same issues. So I ported TXState to it in the "Porting txstate" PR. I also rewrote it while I was at it with better types on the implementation level.

**TypeScript**
While I was working with TXState I developed a pattern called "Self-Type-Checking Types" and I decided to take a stab and make it seamlessly integrate into TypeScript itself. You can find more about it in the "Feature: Self-Type-Checking Types" PR. I also came across a shortcoming in TypeScript while I was working with TXState which I attempted to resolve in the "feat: dependent contextual inference" PR.

## Education

**Bachelor of Engineering** in Computers, Thadomal Shahani Engineering College (Batch 2017-2021), University of Mumbai.