LESS MORE

JIM KANG: RÉSUMÉ

To view or more or less information, use the slider above.

I'm a software developer in Cambridge, Massachusetts. I've been a software developer since the year 2000. You can reach me at jimkang@fastmail.com.

My current strengths are making full stack web apps and finding ways to iterate quickly. I am working on a future strength: designing interactive explanations.

This is an overview of my work. If you just want to go straight to my publicly available code, here are my Github repos and my NPM packages.

WHAT I'M LOOKING FOR

I WANT TO BUILD PRODUCTS THAT

- Help people understand ideas and situations.
- Get people to imagine possibilities.
- Put power in the hands of users instead of taking it from them. (e.g. Pull instead of push, aid humans instead of replacing them.)
- Get users to think instead of just consume. This can be done without putting onerous burdens on them.

I WANT TO BUILD TECHNOLOGY THAT

- Is honest and predictable - does what it says it will do.
- Does as little as possible and takes advantage of what already works.

  This may mean:

  - Using rsync and shell scripts to deploy instead of wrapping things in Docker images, as cool as Docker is
  - Rendering image files with headless Chrome (which does elusively excellent kerning) instead of building a purer program from scratch
  - Finding and understanding the bit of code within a framework that has the "magic" feature that you want and just using that instead of rewriting everything to use that framework

- Art
- Visual explanations
- Procedural generation
- Product design

PROJECTS

Personal projects are important to me. They are a great way to get a sense of my interests and abilities, and unlike some of my commercial work, I can freely talk about all of it.

A few highlights:

- An interactive explanation of quadtrees.
  I built this because I was using a quadtree for another app and realized I didn't really understand quadtrees.

  It was well-received! Non-programmer-, non-mathematics-types have told me they understood quadtrees after trying it.

- @godtributes is my most popular bot. It had 27,000 followers on Twitter.
  It is a merciless generalization of the "BLOOD FOR THE BLOOD GOD!" meme.

  It was an interactive Twitter bot that, despite its limited modes of expression, inspired a lot of followers to fill in the blanks with their imaginations. Here's a nice review on I Love E-Poetry that discusses its poetic appeal.

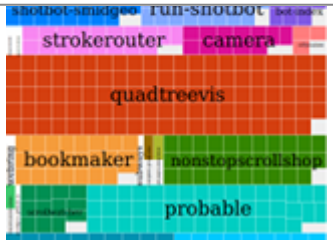
  Here is its source.

- annoy-node is Node bindings for Annoy, a popular Approximate Nearest Neighbors implementation in C++.

  There are a lot of machine learning models that figure out what is similar to what else. The way they express what they've figured out is in n-dimensional vectors representing things like words or songs.

  This Node module lets you use — in JavaScript — what those ML models have produced.

LESS MORE



If you want a **complete** view of my projects, look at Observatory. It organizes 300+ projects from a variety of perspectives.

JOBS

WEB AND MOBILE ERA

This part of my career started around 2008 and continues today.

SPOTIFY

I currently work at Spotify.

In 2016-2018, I worked on a team that built music recommenders for users (e.g. the This Is playlists — greatest hits playlists for artists that stay fresh by updating daily based on listening patterns) — and Time Capsule). Now, I work on a team that is building voice experiences.

Some things I've worked on at Spotify:

- An image generator that composes and renders thousands of playlist covers each day based on playlist contents by taking advantage of headless Chrome
- Internal apps (web apps, Chrome extensions, Electron apps) for evaluating and adjusting algorithmically-generated content
- Interaction and content prototypes
- Voice experience prototypes built in Alexa Skills and the browser (using the Web Speech API and DialogFlow)
- External promotional web sites
- Gathering data from users via surveys

My work involves full-stack web engineering, navigating a unique internal infrastructure. It also involves investigating both user needs and technical possibilities.

There is quite a bit of building things purely to see how users will react (in in-person user tests and A/B tests). I've learned much about the value and costs of gathering information

PAYPAL

I worked at PayPal on the Shop, a system built in Node.

The Shop site served coupons from various PayPal partners to hundreds of thousands of people per day. The front end was a single-page app built in AngularJS. The back end was a cluster of NodeJS servers that talked to PayPal services to manage the coupon and user information.

We used TDD, pair programming, and GitFlow heavily. We prioritized the mobile web experience, which I've taken to heart ever since.

NPR

I worked at NPR on a station management app and API.
It was called Composer, and it was written in Node and Backbone.
Over 200 NPR member stations used it to add program schedules to their web sites and keep track of what they played so they could pay for it.

I added this app's first automated tests (in order to safely do a big refactor to add OAuth to the API) and learned quite a bit about TDD as a result. Our team used a Kanban process and an unusually high level of design-development integration.

MODO LABS

The company built mobile apps for big institutions, mostly universities. It also wrapped up customers' existing data sources to make them easily mobile-consumable.

I worked on various universities' iOS apps here.

VOTER ACTIVATION NETWORK

Their business was centered around a massive data warehousing web application.

I worked on their ASP .NET and iOS apps.

OBAMA FOR AMERICA

I spent a month as a "Data Fellow" during the 2008 presidential campaign.

Indiana went blue, which it had not done since 1964.

application of technology can make a big impact when applied to the right context.

WINDOWS ERA

This era lasted from about right after college (1999) to 2007.

GN RESOUND

GN Resound is a hearing aid company. Hearing aids are small digital sound processors. Audiologists can calibrate and gather data from them by connecting them to a computer.

I worked on their hearing device adjustment software, which at the time, was a C++ COM/ATL-based Windows desktop application.

INSTALLSHIELD

This is where I learned to write code that worked in real-world situations when run by thousands (millions in a few cases) of real users.

I worked on IDEs that let developers author "setups," programs that installed software onto computers.
The IDEs were Windows applications written in C++.

KEYWORDS

There are a lot of busy people out there that just need to see if a particular word is in a resume. For them:

- JavaScript
- Node.js
- Test-driven development
- Modular architecture
- LevelDB
- CSS, Flexbox, animations
- Browserify
- Objective-C
- Git, GitFlow
- Tape, Mocha

LESS MORE

- C++
- SQL
- HTML (v1-5)
- Linux server adminstration
- Docker
- Puppeteer
- BigQuery
- AWS Lambda
- Alexa Skills Kit
- DialogFlow
- APIs
- Design
- At least a 1X engineer, maybe even 1.1X