PATRICK KING

Full-stack web developer in Ruby on Rails + React + Node.js

patrick.f.king@gmail.com
(403) 922-4960

307-1422 Centre A ST NE
Calgary, AB
T2E 2Z9

OBJECTIVE

Further my breadth and depth of skills in server and client development, and expand into cloud and devops.

EXPERIENCE

VizworX Inc. - March 2013 to Present

Unreleased NEB Data Visualization - 2017 to 2018

React.js | Immutable.js | Redux.js | Webpack | IIS

Role: Consulting Developer - Challenges: While transitioning out of the company, I bootstrapped this new visualization and assisted with scoping and estimation before handing it off to the team. The main challenge was to offload and document all of the knowledge I had built up of the client and our projects. Keeping up with code review also became interesting, as I was no longer the author of any of code, and had less frequent contact with the team over time.

Pipeline Incidents *(NEB)* ***(Github)*** *2017*

React.js | Immutable.js | Redux.js | Webpack | IIS | Node.js

Role: Lead Developer - Challenges: I onboarded a new junior developer, while transitioning the team to a new React based stack, while delivering a very complex browser data visualization app under a very tight deadline. My organizational skills, time management, client interfacing and detail orientation were tested on this project! We delivered the app successfully and on time, after resorting to a limited amount of overtime. I lead the adoption of code review as a practice at Vizworx during this project.

Exploring Canada's Energy Future *(NEB)* ***(Github)*** *2016 - 2017*

Coffeescript | Browserify | D3.js | Node.js | IIS | Phantom.js

Role: Lead Developer - Challenges: I stepped into this project after it was already underway. As a pilot project for this client we had wide latitude in our choices of technology and approach, but we had to deliver this browser based visualization under tight deadlines and with limited access to the client's hosting environment. Challenges included building the app with tools not well suited to browser app development (D3 especially), integrating Node.js applications into an IIS hosting environment, and interfacing with an external design team and external client staff. The project teams moved fast but our partners were chiefly concerned with the design of the app, ensuring that essential features mentioned only in passing were discussed, captured, and scoped was also a challenge.

Ivrnet Central - 2016

Ruby on Rails | Rspec | Docker | Postgres SQL

Role: Developer - Challenges: This was a short engagement with another Calgary software company to support their development team; the main challenge was to drop into an unfamiliar Rails codebase and make worthwhile contributions during the six week span, while also learning the basics of Docker.

Geoviz - 2015 to 2016

React.js | Flux.js | Leaflet | Node.js | Objective-C + iOS | C# + WPF | ArcGIS Server + SDKs

Role: Developer - Challenges: Geoviz was a sprawling prototype project, with applications all communicating with each other on Windows hosts, iOS devices, servers, and in the browser. The essential challenge was keeping all of these platforms in sync as new features were added and communication protocols changed. Another notable challenge was implementing a React based mapping application at a time when adapters for Leaflet and Mapbox were at an early stage of development, or did not exist at all. Adapting DOM-state heavy and procedural tools like Leaflet to React's less stateful rendering approach was a challenge.

WeEmploy - 2014

Ruby on Rails | Postgres SQL | Rspec | Sunspot Solr + Lucene

Role: Developer - Challenges: The application concerned an online job board, and one notable challenge was designing a search system integrating Rails and Solr for matching candidates to jobs based on skills and other traits. The client brought their own design team with some ambitious ideas, which demanded solid client facing abilities and presented some interesting HTML/CSS implementation challenges.

Jobbsite - 2013 to 2015

Ruby on Rails | Postgres SQL | Rspec | Cucumber | Phantom.js | Selenium | Paperclip | Prawn | Ampersand.js

Role: Developer - Challenges: I began on Jobbsite as a junior developer working under a talented senior to build a custom time and order management system for a client, and ended as a leading developer on a larger team adapting the software to serve as a SAAS product. There were numerous challenges over the years, including onboarding additional junior devs, adding a rich client experience using a browser toolkit called Ampersand.js, the Rails 3 to 4 upgrade, a continuous need to optimize the speed of a sprawling test suite, balancing the needs of the client vs. the needs of the product, and implementing a reliable multi-tenant database system.

Agile Software Engineering Lab, University of Calgary (Dr. Frank Maurer) - May 2011 to February 2013

NSERC Surfnet Research Network Page *- 2013*

Ruby on Rails | Postgres SQL | Paperclip

Role: Lead Developer - Challenges: My last task with Dr. Maurer's lab was to implement a website to document the work of the Surfnet research network, where network members could log in and upload descriptions and papers detailing their projects. This was my first Ruby on Rails project. Effectively, I attempted to learn Rails and implement a simple CMS on my own in two months, and in retrospect this goal was entirely too ambitious. Not knowing this at the time, I made a serious attempt to complete it. I learned a lot while building a substantial portion of the site, before handing it off to another intern.

MRI Table Kinect - 2012

C# + WPF | Kinect | Objective-C + iOS

Role: Developer - Challenges: Building off of MSE-API and working for a grad student in Dr. Maurer's lab, I built this prototype which synchronizes the display on a touch tabletop with an iPad by tracking the device with a Kinect. The main challenge was the short timeframe, the project came together in one week. The work resulted in a **publication in MediCHI'13**. (Unfortunately, I did not get to go to the conference in Paris to present the work.)

Multi-Surface Environment API - 2012

C# + WPF | Kinect | Node.js | Objective-C + iOS

Role: Developer - Challenges: As one developer in a team of six, we built a framework for cross device communication incorporating Kinect skeletal recognition and gestures on mobile devices. The main challenges included keeping codebases for different platforms in sync, and coordinating

work between a medium size team of students and interns who had many other demands on their time. In and around work on this project, I started a drive within the lab to move our source control from an in-house TFS Version Control server to Github.

eGrid - 2011

C# + WPF ArcGIS C# SDK

Role: Developer - Challenges: eGrid was a prototype power grid management application, for tracking work crews and power equipment issues / outages citywide. The main challenge was diving into an existing codebase while learning C#, touch interface APIs, and GIS basics. eGrid ran on an early generation touch tabletop with early Microsoft touch software, both of which came with unique issues to work around.

University of Calgary iGEM Team - May to August 2009

Synthetic Biology Interactive (Second Life Installation)

LSL (Second Life)

Role: Team Leader - Challenges: I lead a team of three other students to build a virtual installation in Second life, including two biomedical sciences students with no software development experience, mentoring and guiding these students in the basics of coding and 3D graphics was a key challenge. An interesting technical problem was working within the limited environment provided by Second Life's integrated Linden Scripting Language: each script is limited to 64kb of memory for compiled code and stack / heap memory. But as a garbage collected language LSL includes no way to manage memory directly. Adding a line of code to a script, or adding one too many elements to a list, could result in an out of memory error. Splitting functionality across multiple scripts allows access to more memory, but requires the addition of an IPC-like communication system over the game's chat channels.

EDUCATION

BHSc - *Bioinformatics (Honours) - University of Calgary - 2012*

Honours Thesis: *An Algorithm for Chromatin Immunoprecipitation Sequencing Analysis.* Python R

SKILLS + TECH

What I'm Best At

React.js The low-to-no-state and continuous re-rendering approach that React encourages for building complex UIs was eye opening when I first encountered it, and now it feels wrong to build UIs in any other way. React is one of those rare tools which changes the way you think about development.

Node.js I began working with Node after several solid years with Rails, and the ease with which you can build a simple working server in Node impressed me. Promises (and asnyc/await) were good to learn too, but Node will never be my first choice for server development; Javascript has too many flaws as a language (as compared to say, Ruby) and Rails makes a lot of good choices for you by default.

Ruby on Rails RSpec Cucumber The convention based layout of every Rails app is both its best and worst feature. Portable conventions are a huge aid to understanding a new app, I've been productive contributing to a substantial Rails app on day one after getting to the code. But Rails doesn't set you up very well to scale an app's size and complexity, it naturally encourages monolithic apps. Careful use of Ruby's excellent object orientation features is the way out.

Javascript (ES 2015+) ESLint Modern Javascript has grown into a more pleasant and powerful language. Despite all the new and useful features I use daily (modules, arrow functions, deconstruction assignments, classes), there are still numerous new features I haven't needed to tap into yet. I list JS together with a linting tool, because there are enough serious gotchas in the language that using a linter takes on the same importance as compiler checks in other languages. Some go further, using tools like Flow, React's proptypes, or languages like Typescript for stronger static checks.

Coffeescript Rails introduced me to Coffeescript, and I was instantly sold on its clean syntax, indentation based scopes, and effort to reduce/eliminate "the bad parts" of Javascript. Less code is better code! Modern JS is the better choice for professional work, but I still enjoy writing my personal projects in Coffeescript.

Immutable.js Immutable lets you build efficient, un-changeable data structures in Javascript. I had long heard the virtues of using immutable structures to drive apps but expected it to be challenging to use them, and I was pleasantly surprised just how flexible this library turned out to be. Immutable is easy to recommend for React projects of any complexity, and now I wish Javascript had a native ordered map type like the one Immutable provides.

Redux + Mobx + Flux I lump these three tools together because they all address the same problem in similar ways: managing state for React apps. Creating a centralized store for an application's state and enforcing that it only change through actions submitted to the store is a great way to bring order to the chaos of an interactive UI, but it can result in components that are not very modular if used carelessly.

Git (CLI + Git Flow) Git is a wonderful tool. I've always preferred the CLI over graphical tools, sooner or later your repository will get into a state that the GUI tool can't handle. At that point your only option is to sort it out at the console, so I reason that you might as well learn "the real interface" from the start. A personal favourite utility: gitk, a cross-platform graphical git history viewer. It's a little quirky, but blazing fast, supports most options that `git log` takes for searching/filtering commits, and works identically on Mac/Win/Linux.

D3.js I began working with the D3 charting library after becoming familiar with React, and I would describe it as a middle point between React and jQuery. Like jQuery it targets parts of the DOM with selectors, and like React it permits you to map from data structures to DOM elements and hides some of the complexity of updating the DOM. D3 is fine for the occasional chart with a few simple interactions, but it has most of jQuery's problems when building complex applications.

Agile practices Daily stand-ups, one week sprints, short weekly sprint planning meetings with JIRA tickets, extensive unit-testing, chat-ops, and the occasional pair programming session have been my way of life as a professional developer. I drove adoption of code review at VizworX on a project I lead, and was pleased to see colleagues on other projects adopt it too. Continuous integration is a great idea, but there is a maintenance cost to keep it online, my impression is that it only becomes worthwhile for teams larger than three or four developers.

jQuery **Promises + Bluebird.js**

What I'm Good At

Webpack · Postgres SQL · IIS · Phantom.js · Python · C# + .NET (Web and Windows desktop) · WebGL + Three.js · Team Foundation Version Control (TFS) · Leaflet · Browserify · Heroku · ArcGIS (APIs + SDKs + server configuration) · Prawn · French (Written + Spoken)

What I've Played With

Java + Android · Objective-C + iOS · Docker · Sunspot Solr + Lucene · R · MongoDB · Go · Ember.js · MSBuild · Ampersand.js · GLSL (with WebGL) · Puppet · Meteor.js · LSL (Second Life) · Drupal · SQL Server (TSQL) · C + C++ · Django · Kinect · Blender · Selenium

OTHER WORKS

Accessible Visualizations: A Case Study **(slides)** - I presented work on making the Energy Futures visualization accessible, as part of the our design partner's **Data Empowerment Speaker Series**.

"Star Trek: Armada" Gallery - A shrine for an old computer game I enjoyed. Coffeescript Node.js Three.js WebGL

Neoderelict - A simple prototype browser game. Coffeescript Three.js WebGL Blender

References available on request