PATRICK KING

Full-stack web developer in Ruby on Rails + React

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

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

Software Developer *- VizworX Inc. - March 2013 to Present*

Vizworx goes here!

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

University goes here!

Student Researcher *- University of Calgary iGEM Team - May 2009 to August 2009*

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.

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.

Webpack Webpack is a little like Git, in that the API you use to drive it doesn't map extremely well to the concepts it thinks about, and also in that it is very powerful. I've personally only scratched the surface in terms of asset workflows that you can build with Webpack.

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, 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.

What I'm Good At

Postgres SQL IIS Python C# + .NET (Web and Windows desktop) WebGL + Three.js Team Foundation Version Control (TFS) ArcGIS (APIs + SDKs + server configuration) Browserify

What I've Played With

Java + Android Objective-C + iOS R MongoDB Go Ember.js

MSBuild Ampersand.js GLSL (with WebGL) Puppet Meteor.js Second Life (LSL) Drupal SQL Server (TSQL) C + C++ Django Kinect Hanami Blender

Exploring Canada's Energy Future ***(Github)*** - I lead development on this data visualization for the National Energy Board working at Vizworx.

Pipeline Incidents ***(Github)*** - A second data visualization for the NEB, which I also lead.

"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

Fluent in French and English - Avid winter cyclist - References available on request - ***This resume is valid HTML***