Spice Program Open Source 2015 learnings

Please see this Futurice Spice Program blog post to learn the context for this list.

Learned even more about practical problems in measuring time.

Direct uploads from browser to AWS S3.

Basics of Polymer.

React + Redux (lots of it).

Basic info about Google Cloud Services.

Practical usage of Google Cloud Storage (the Google's S3 equivalent).

Heroku Pipelines.

Improved & more modular Angular application structure formation.

Webpack, JSON Web Token implementation on the backend, and ES6.

Actually learning how Promises should be used in JavaScript.

Some knowledge about Dokku, the host-it-yourself-Heroku.

Got practical experience and more understanding about Node.js, npm, and bacon.js.

How the Azure Web App continuous deployment works.

A bit on how Travis CI operates under the hood.

How to make Joomla templates.

Learned about Let's Encrypt on Apache with vhosts.

Learned more about Web MIDI API and RxJS. Also more about ES6 and setting up a compact development environment for that.

Learned basics of RxJS.

Learned a lot about creating a reusable library with JS/browserify.

Learned some Cycle.js.

Learned a little about Node.js testing.

Learned the EMV chip card specification on very detailed level by implementing large part of the specification. I'm using the EMV knowledge on daily basis in my current client project.

Improved my Clojure language skills.

Learned to work with the Go programming language.

Wrote an OSS tool that I later then used in a customer project.

I learned better ways to use Dagger 2 in Android and trial and evaluate other app architectural patterns based around this.

Learned Redux first in a hobby project, then transferred the information to a work project.

Learned to program in Clojure and then used that knowledge of Clojure in a customer project.

Learned Emacs Lisp and tweaking my Emacs so that I'm more efficient when using it.

I learned Android development and now I get to do it as my day job. I have also learned a lot about the internals of Node.js which helps me when creating new backends or debugging existing systems.

Learned to explain software product development to a non-programmer audience.

Hopping on to a big project (desktop software, Wordpress) and getting up to speed to implement something quickly.

Starting stuff from scratch (desktop environment plugin, web service in Haskell).

Getting users and interacting with the community of users (bug reports, language translation contributions, code contributions, feature requests, praise).

In general new languages like Haskell and Vala.

While I haven't probably used the skills I've learned in my work, I've definitely used the tools I've written to be more productive.

Gained more experience with RxJS and got to know Cycle.js and Angular.js a bit.

Learned how to use npm scripts as build tools and that Brunch.io was difficult to set up when using React or Cycle.js.

Learned that HTML tables are impossible to style the same on different devices.

How to deploy a Clojure application to Heroku.

Learned that OpenShift is not as smooth to operate as Heroku, mostly lacking in documentation and the UX of their console.

Coding for Pebble watch.

New features in ECMAScript 6.

Learned more Node.js.

Really used TDD for the first time ever :)

Learned about IoT and problems with physical ports and objects.

The basics of TypeScript.

Basic Clojure, including formatting and deploying to Heroku.

Learned bits of React.

Learned enough of the JSPM ecosystem to know I wouldn't recommend it to be used in our current customer project.

Contributing to open-source projects in general.

I haven't made many contributions - but those I have made I have been able to use directly in my project work at Futurice, as well as gain deeper knowledge of the open source tools we use daily.

For me it's been mostly about publishing stuff. I've rarely gone through the process of making a library or feature for others to use, so besides learning new technologies I've also had to taught myself some releasing mindset (finishing touches, deployment, documentation, testing).

Learned how to best use the tools to work in a collaborative way with people anywhere in the world. E.g. Pull requests, issues, etc.

Basics of ExpressJs framework.

Learned about Node.js development.

Learned how to leverage Heroku -> faster development in customer projects -> customer has saved money.

As I had time to explore better deployment processes and products, I was able to bring those into the next project.

I learned thing about architecture, publish-subscribe products and usage (fast communication between complex systems using pubsub channels instead of ESB or similar beast).

Deep details of several Node.js packages used in the prod-level app built for a customer. Overall architecture ideas in the Node.js land, learned about new tools and packages and what kind of ways there are to build stuff.

Contributing OSS code to an existing project forces me to ensure I know what I'm doing and I understand the component in question so that I can discuss about the contribution I made properly with the authors, even one-line fix often requires learning about a whole lot of things around it.

Handling cookies in Scala/Play2.

Learned to use Compojure API.

Basics of ClojureScript.

Learned to create fingering for baroque keyboard music.

Learned the basics of Web Audio API.

Learned to deploy and configure a web backend server to an empty Digital Ocean server with build hooks and monitoring.

Learned several things about ES6.

React/Flux, Postgres, Clojure, and general backend stuff. I've used everything except Clojure in recent projects.

Made a small app that showcases a reactive iOS architecture to new hires and other interested people.

Studied some API design and backend development to build a simple backend for that app. Set up a blog that explains concepts from the app to iOS developers.

How a video streaming protocol work, about distributed computing on Apache Spark, and to explain programming concepts to high school kids.

Learned Apple's Swift, developed an iOS app, improved my SBT knowledge, improved my Scala.

Learned lots more about Android, Gradle, and RxJava (and reactive programming in general).