Gmail **Anibal Santos <anibalsantosgo@gmail.com>**

[JSR] – Exploring JavaScript peripherals (The Adult Phase)

1 message

Zell Liew <zell@zellwk.com> Mon, Mar 11, 2019 at 12:21 PM
To: Anibal <anibalsantosgo@gmail.com>

Hey Anibal,

You're in the Adult Phase if you know enough JavaScript to be dangerous. You can build almost anything you want. You're confident with your code. It's mostly clean; it follows many best practices. It's not perfect, but it's good enough.

Now, it's time to leave the nest and look for something new, something related that brings you closer to where your goal.

You have a few options here. You can:

1. You can learn a frontend framework (like Angular or React).
2. You can learn Node to build a backend.
3. You can dive even deeper into JavaScript.

Let's talk about these three options in more detail.

Learning frontend frameworks

When you use a framework, you're locked into its philosophy. Change is costly. So, before you learn a framework, ask yourself if you need it first.

Not everyone needs a framework. Sometimes, vanilla is the way to go.

Consider reading this article to help you decide whether you need one.

Let's say you want to use a framework instead of going vanilla.

You need to decide on what framework to learn next. That's a hard choice. Many people get stuck here.

There are many articles out there comparing different frameworks. I suggest you spend an hour (max two) to read through these articles, then choose one that feels right for your situation. Here's one that may be helpful.

It doesn't matter which framework you choose. It doesn't matter if you choose the wrong one. Keep going and learn that framework deeply first.

I say this because frameworks are similar to each other. They do the same things (routing, dom manipulations, controlling state, etc). Once you have a grasp of what to look out for in the framework you chose, you'll have an easier time learning another framework later.

If you can, construct a small project for hands-on practice as you learn it; you'll see things that you wouldn't see if you just read the documentations.

Other libraries

Aside from frameworks, there are many libraries built on top of JavaScript. A few examples are:

1. GSAP – GSAP is a ultra performance animation library that works all the way back to IE6. If you want to build complex, cool animations, be sure to check out GSAP.
2. D3 – D3 is a library for manipulating documents based on data. If you want to learn to visualize huge amounts of data, be sure to check out D3. You'd also be interested in Data Sketches – a place where Shirley Wu and Nadieh Bremer showcases awesome visualizations they've built.
3. Lodash – Lodash is a utility library that makes it easier for you to work with JavaScript. It's great if you want to learn to manipulate data.
4. Webpack – Webpack is the most popular library for bundling your JavaScript assets right now. If you master it, you can concatenate and minify your JavaScript with a one-step build process. It

also allows you to use ES6 imports in your frontend code, which lets you create a better code architecture.
5. Gulp – is a popular task runner. In addition to bundling your JavaScript files like webpack does, you can create amazing workflows that save you tonnes of time. I even built a static site generator with Gulp for my website at zellwk.com. Here's ten free chapters from my Gulp book to help you get started.

Learn to build a backend

If you want to build a backend, I highly recommend you learn Node; it's JavaScript on the server. Node is a dedicated backend language, comparable to the giants like Ruby and Python. Many huge companies, like Netflix, PayPal and LinkedIn, use it.

The benefit to you learning Node is twofold.

First, you already know JavaScript. You'll be able to pick up Node faster than other backend languages (assuming you don't already know them).

Second, if you know Node, you can take advantage of it to use many frontend tools, like Webpack and Gulp, that are built on Node. You can also use npm as a package manager for your frontend libraries.

Besides learning Node, you need two more things – a server framework and a database language.

A server.

A server framework helps you spin up your backend quickly without having to type lots of code. You can learn to build a server manually without a framework if you want, but I don't think there's a need to.

The most popular server framework for Node is Express.js. I highly recommend learning it.

A database.

Think of databases like the hard drives on your computer. When you create a server, you need something like a hard drive to organize and store information.

There are many database languages, mostly split between Sequel based (SQL) and no-SQL databases. The most popular SQL based database is Postgres while the most popular no-SQL database is MongoDB.

If you're starting out, I highly recommend MongoDB because it's syntax is similar to JavaScript. It's easier for you to learn.

Here are some resources that'll help you spin up your first server quickly:

1. Building a Simple CRUD Application with Express and MongoDB
2. Wes Bos's Learn Node course (paid).

Diving further into JavaScript

Even though you're in the Adult Phase, there's still a lot you can learn about JavaScript. Feel free to choose any of these to dive into:

1. Object Oriented Programming

In the Teenage phase, you only learned the basics to OOP. If you want to dive further, you might want to check out these articles:

1. Subclassing vs the Mixin Pattern
2. How to use `bind` in JavaScript

You're pretty much done with OOP once you understand the above two concepts.

2. Functional Programming

If you want to dive further into FP, I suggest reading Professor Frisby's mostly adequate guide to Functional Programming.

In it, you'll learn about partial application, currying, monads, monoids, functors, endofunctors and many more FP terms.

3. The latest JavaScript improvements

JavaScript as a language has improved tremendously over the years. The latest stable version of JavaScript that’s usable across all browsers is EcmaScript 6 (ES6 or ES2015).

If you want to, you can learn about future versions (like ES7 and ES8) of JavaScript before they become stable in browsers.

If you want to use these future versions today, you need to use a complier called Babel. If you need Babel, you’d want to either learn Webpack or Gulp as well.

4. Test Driven Development

Test driven development (TDD) is a software development process that emphasizes writing tests as you code.

It helps you improve your JavaScript skills by forcing you to write testable code. In doing so, your code becomes terser, purer and more maintainable.

To get started with TDD, I suggest reading this book by Christian Johansen.

So many options, which to choose?

Keep a lookout for the next email. I’ll share a strategy with you.

Stay awesome,
Zell
