



WYNNE'S
coding pros

Essential Javascript Frameworks to Be a Happy, Well-Paid Coder

Hi, Wynne Pirini here. I founded Wynne's Coding Pros to share everything I have learned about the *process* of learning to code.

I never started out as a coder. Originally I was a Chemical Engineer. Later, I became a project manager. And then entrepreneur.

After making 100's of websites, I decided I should learn to code also.

Javascript Frameworks can dramatically speed up the development process. But which one you decide to use can also lock you into a particular development strategy.

So in this report, I've laid out all the factors that you need to consider, before you proceeding down one path, or another.

Be sure to also visit Wynne's Coding Pros (wynnescodingpros.com) – where I use my combined experience as an engineer, project manager, and entrepreneur – to steer you in the right direction in your coding career.

Wynne Pirini

INTRODUCTION

In this report, we look at the most important frameworks to focus upon, in terms of happiness, long-term interest, and career opportunities.

We break this discussion down into three main areas:

Part 1 - Frontend Development

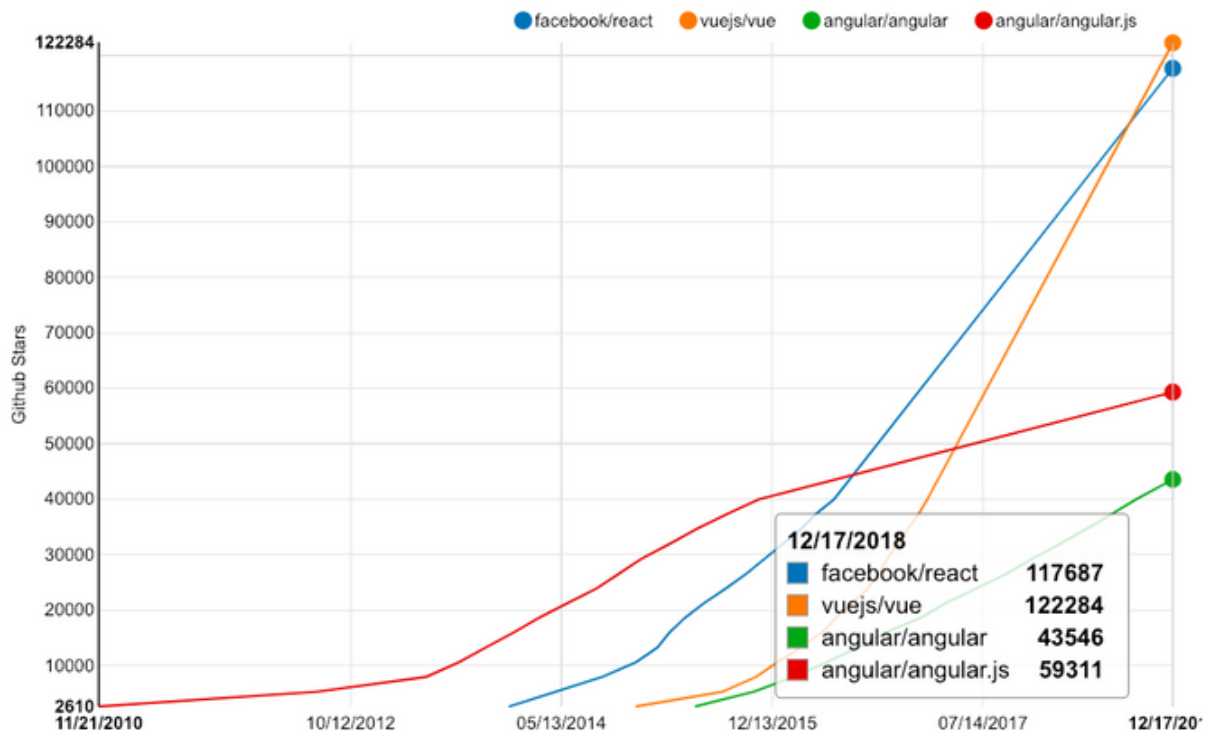
Part 2 - Data Layer

Part 3 - Backend Development

PART 1 - FRONTEND DEVELOPMENT

With the rise of tools like Bit, which let you share and reuse components to build new applications, getting them to work together, and synching across projects, component-based frameworks are taking over.

Developer Interest

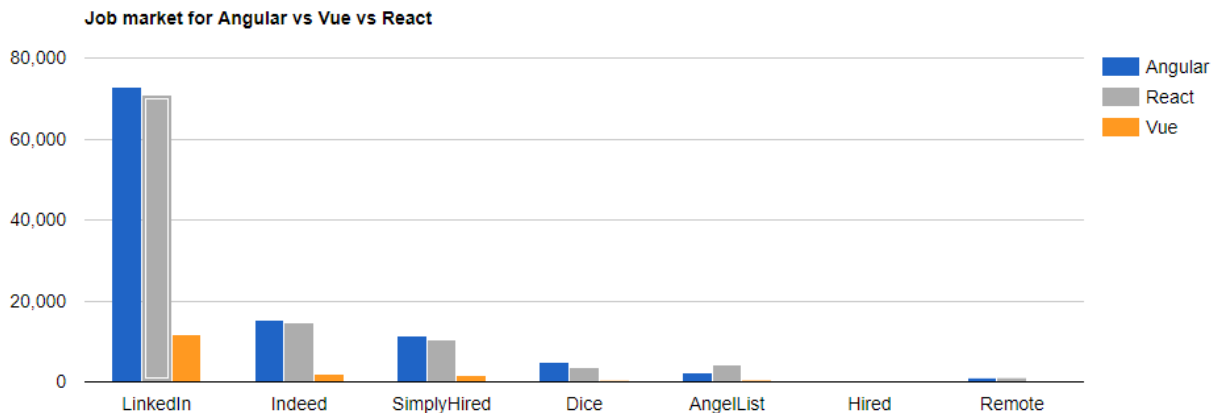


Wynne's Recommended Javascript Frameworks

Using Github as a guide to developer interest, Vue has 122,000 stars, React has over 117,000 stars, Angular.js has 59,000 stars.

Job Openings

But, when you look at job openings, it's a different story...



Source - Andrei Neagoie

We analyzed the number of open positions worldwide for frontend Javascript developers. Data comes from Indeed.com, with a sample size of over 60,000 job openings.

You can see, Angular and React developers are 'neck and neck', in terms of demand, with Vue coming in a distant 3rd place.

Learning Curve

Angular has a very steep learning curve because it's a complete MVVM (Model-View-View-Model) framework.

React is not a complete framework. It's concerned with the frontend. Hence, learning React is much faster than Angular.

Vue is even faster to learn than React. Also, transitioning to Vue from Angular or React is relatively easy. But Vue's flexibility is a double-edged sword - it's easy to write poor code, which can get you into trouble later when debugging and testing.

Pros and Cons of Each Framework

React

Advantages:

- Relatively easy to learn, thanks to its simple design, use of JSX (similar to HTML) for templating, and detailed documentation
- Developers focus on writing modern JavaScript and are less concerned with writing framework-specific code
- Extremely fast, courtesy of React's Virtual DOM implementation and various rendering optimizations
- Great support for server-side rendering, making it a powerful framework for content-focused applications
- First-class Progressive Web App (PWA) support, thanks to the `create-react-app` application generator
- Data-binding is one-way, meaning less unexpected problems
- Redux, the most popular framework for managing application state in React, is relatively straight forward easy to learn
- React implements Functional Programming (FP) concepts, creating easy-to-test and highly reusable code
- Applications can be made type-safe with either Microsoft's TypeScript or Facebook's Flow - both featuring native support for JSX
- Migrating between versions is fairly easy, with Facebook providing "codemods" to automate much of the process.
- Skills learned in React can be applied to React Native development.

Disadvantages

- React is moving away from class-based components, which may be a challenge for developers more comfortable with Object Oriented Programming (OOP)
- Mixing templating with logic (JSX) can be confusing at first

Angular

Advantages

- Angular works seamlessly with Typescript (one of the fastest growing Javascript variants)
- Angular-language-service — allows intelligence and autocomplete inside of component external HTML template files
- New features like a generation of Angular based npm libraries from CLI, generation, and development of WebComponents based on Angular
- One-way data binding that enables singular behavior for the app which minimizes risks from errors
- MVVM (Model-View-ViewModel) that allows developers to work separately on the same app section using the same set of data
- Dependency injection of the features related to the components with modules and modularity in general
- Structure and architecture is specifically created for project scalability

Disadvantages

- Variety of different structures(Injectables, Components, Pipes, Modules, etc.) makes Angular a steep learning curve. React and Vue.js are much easier to learn by comparison because they only deal with “Component”
- Relatively slower performance, than React or Vue

Vue

Advantages

- Empowered HTML - Vue.js has many characteristics in common Angular and this can help to optimize HTML blocks handling when used in different components
- Detailed documentation - Vue.js has great documentation, which helps with learning. You only need a basic understanding of HTML and Javascript to get started with this framework.
- Adaptability - Switching from other frameworks to Vue.js is relatively easy because of the similarity with Angular and React.
- Integration - Vue.js can be used for a wide variety of uses, from single-page applications to highly complex user interfaces. Smaller parts can be easily

integrated into existing infrastructure with no negative effect on the overall system.

- Scaling - You can develop large, reusable templates
- Size and speed - Vue apps are typically about 80KB in size. This helps with performance and flexibility compared to other frameworks.

Disadvantages

- Market Size - Vue.js has a small market share compared with React or Angular. Which means there are fewer people sharing information to improve the framework.
- Too much flexibility? - Sometimes, Vue.js can have issues with integrating into large projects where there is a lack of experience about how certain components will work together

Our Recommendation

Although Vue is a great framework, there aren't enough jobs at the moment. Maybe this will change in the next few years. But, we can only make projections based on the current, observable trends.

Angular has a much steeper learning curve than React. And, while Angular is growing, React appears to have a more promising future, both in terms of job openings and developer support.

If you're choosing a path between these 3 frontend frameworks, React is your best bet at the moment.

PART 2 - DATA LAYER

With the introduction of Node.JS in 2009, Javascript became popular as a serious programming language.

It wasn't long before robust frontend frameworks emerged - a few of which are mentioned in this report.

This began a shift in the way apps were developed. The traditional MVC framework split into two parts - a server, which handled MC (models and controller) and a front-end client which handled the view.

Then Facebook ran into a problem. At the time, it had the biggest web app in the world. And, people couldn't see the correct number of notifications unless they refreshed their page. Which isn't ideal from a usability point of view.

To deal with this, they developed the frontend framework known as React (2013), which also came with a 'pattern' known as the Flux architecture. Flux helps display current data, such as notifications, without users having to refresh their browsers.

Inspired by Flux, Dan Abramov collaborated with Andrew Clark (the developer of Flux) to create Redux, an open-source JavaScript library for managing application state. It's commonly used with frameworks such as React and Angular for building user interfaces.

Redux stores all dynamic information of an app in a single Javascript object. Whenever an app needs data, it queries the server, and updates the Javascript object, then displays that data to users. By keeping this data in one place, the info is always correct.

The problem with this approach is it takes A LOT of extra coding. And, the learning curve for new developers is fairly steep.

Facebook realized they had a complexity problem. To simplify things, they developed a new technology called GraphQL.

GraphQL achieves a lot of the things we currently do with Redux, but with much less work. But, on its own, doesn't replace all of the functionality of Redux.

That's where a project called "Apollo" comes in - which, in their own words, is, "A single versatile query system to replace a patchwork of legacy APIs, with all the devtools and cloud services you need to run GraphQL at scale"

Our Recommendation

In time, most React apps will use Apollo. And, lot's of enterprises are transitioning their apps over. So, learn Apollo GraphQL first because the next generation of apps will be built using this approach.

But, be prepared to learn Redux if an existing project uses this library.

PART 3 - BACKEND DEVELOPMENT

Node.JS

Node.js is an open-source server environment, which includes everything you need to execute a server-side program written in Javascript. So, the first place to start, if you want to learn backend development with Javascript, is Node.js.

But Node.js only provides the ability to communicate via HTTP. This is significantly more fundamental than traditional web servers. For example, there is no baked-in way to process a JSON body or other form body. There is no baked-in CORS support, nor is there a concept of multi-stage processing (middleware). Trying to teach someone how to construct their own HTTP headers and request body parsers can be daunting, especially in a tutorial.

That's why backend development frameworks grew popular. They build upon the core Node.js HTTP capabilities, providing common things like route handling, body parsing (through add-ons), etc. Basically, the way most people expect a web server to work since Node doesn't do this.

Using Express.js removes a major learning hurdle for beginners. They can focus on working with JavaScript on the server, instead of learning dense 'nuts and bolts' theory of how a web server works.

Once you become a 'gun' with Node.js, you can do away with Express.js and build your own lean solution - which will significantly speed up the performance of your apps. But, in the beginning, your goal should be quick wins and coder productivity.

Now, the question is, which framework should you use?

There are a number of great options. The more popular ones are:

- Express.js (GitHub stars: 41,000)
- Meteor.js (GitHub stars: 40,500)
- Koa.js (GitHub stars: 24,000)
- Sails.js (GitHub stars: 20,000)

Express.js

When you hear people referring to "MEAN stack developers", they mean MongoDB, Express.js, Angular, and Node.js.

Express.js is the defacto choice for backend frameworks. There are a ton of tutorials, and most Node.js developers are also experienced with Express.js.

Meteor.js

Meteor is a full-stack platform (frontend and backend) for developing apps.

It supports GraphQL and Apollo, out of the box.

Koa.js

Koa was created by the same team that created Express.js.

It uses some really cool ES6 methods that haven't even made it to some browsers yet, allowing you to work without callbacks (doing away with "callback hell"), while also providing much better error handling.

Koa does not bundle any middleware within its core, allowing you to use only the parts you actually need. Which leads to improved performance, compared with Express.js.

Sails.js

Sails.js is best suited to creating data-heavy enterprise-scale apps. Examples include chat rooms, real-time toolbars, and multiplayer games.

Our Recommendation

While is Koa.js awesome, Express.js has the largest community of active contributors, as well as clear, in-depth documentation.

Frameworks that utilize a middleware approach do a better job of managing system resources. Express.js is one of them, which makes development and testing process more transparent. Moreover, as Express does not limit the choice of additional tools, integration of push notifications, payment gateways, in-app purchases, social media, and analytics, is relatively straight forward.

Once again, quick wins and coder productivity is the goal here. For most developers, the most appropriate choice is Express.js.

SUMMARY

To summarize... for frontend development, React.js is your best bet. For data layer, Apollo GraphQL is the future. And, for backend development, use Express.js on top of Node.js.