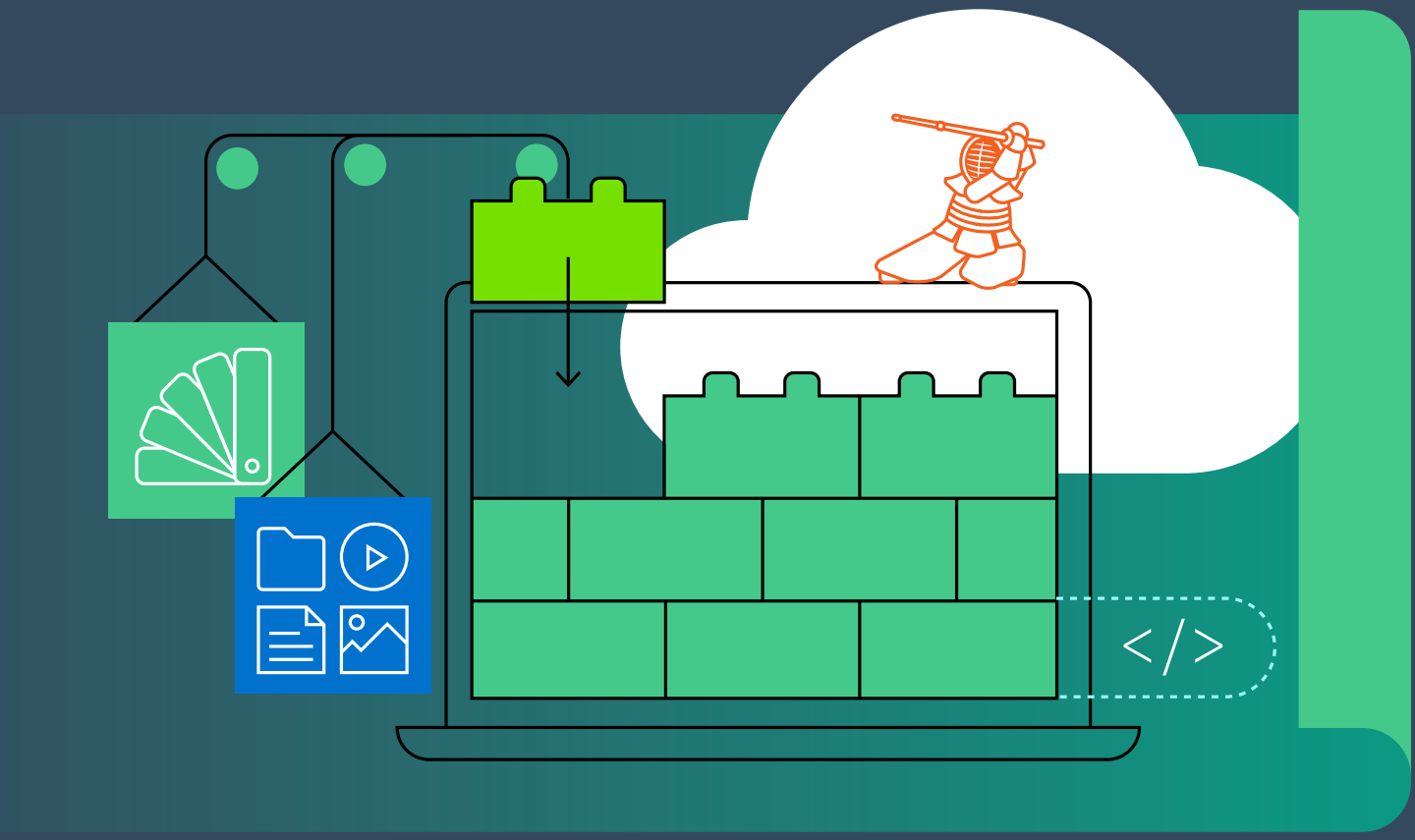
Progress



Planning a Vue Application

Brandon Satrom

WHITEPAPER

Table of Contents

Planning a non-trivial web application that utilizes Vue.js is something you may have done in the past, or are in the midst of doing now. This whitepaper documents a high-level process that you can utilize when planning a Vue.js application, covering project organization, collaboration considerations and tooling choices during development all the way through deployment and performance strategies. There are a lot of moving pieces involved in creating a real-world application in Vue.js, but this guide will help get you started.

Project Management

Before you write a single line of code, you need to decide how you're going to get things set up. This typically starts with project management, discussing and agreeing upon the approaches, tools and services you'll use to deliver your next application.

Software Management Tools

To manage the development of any frontend application, consider the following tools to version code, store assets and oversee team member tasks.

SOFTWARE MANAGEMENT TOOL	EXAMPLES
Issues and feature tracker	GitHub, BitBucket, JIRA
Distributed version control system (DVCS)	Git, Mercurial
Cloud-based DVCS repository	GitHub, BitBucket
Document/asset storage	Internal Network, Google Docs, Basecamp, Office365
Team communication	Slack, HipChat, IRC, Google Hangouts
Task management	GitHub Issues, GitHub Project Boards, Trello, JIRA

No matter which tools you choose, it's essential that your team adopt and use the tools you select. Also, don't be afraid to monitor the use of these tools and improve your workflow if opportunities for improvement arise. New tools are released all the time and you may wish to adopt emerging tools that provide features that are missing in your current process. It's natural to adopt different tools as your team matures and your application grows.

Accessibility, i18n and Environments

Accessibility, i18n (internationalization) and targeting the correct execution environment for your app are essential parts of any development effort. More than just what to build, it's vital that you consider how you're going to build your app, what is its intended audience and how you're going to support it. Addressing these considerations at the start of your project will help you clearly articulate how you'll address key aspects of your app that are ancillary to the code itself, but essential for certain audiences (accessibility and i18n, for instance).

The following table summarizes some of these considerations and provides some helpful resources for addressing them for Vue.js applications.

APP CONSIDERATION	EXAMPLES	RESOURCES
Internationalization / globalization	UI and UX translations for multiple languages	https://github.com/kazupon/vue-i18n https://github.com/dkfbasel/vuex-i18n
SEO	Server-side rendering to enable search indexing	https://ssr.vuejs.org/en/
Cross-browser support	If your site must support IE10+ and all modern browsers (Edge, Chrome, Safari, etc.)	http://Babeljs.io
Accessibility	WAI-ARIA, WCAG	https://www.w3.org/WAI/intro/aria https://www.w3.org/WAI/intro/wcag
Offline-first		https://developers.google.com/web/fundamentals/primers/service-workers/
Progressive web apps (PWA)		https://github.com/vuejs-templates/pwa
Cross-platform native mobile app	NativeScript with Vue.js support	https://nativescript.org https://github.com/rigor789/nativescript-vue

The resources above are examples for consideration when deciding baseline standards and the types of support your application can offer. There are other approaches and new options cropping up all the time. What's more, if your app won't benefit from an offline-first or PWA approach, don't build one in. Always consider the goals and intended audience of your app.

Development Process Methodology

There are a number of different approaches to software development that have evolved over the last 50+ years. Waterfall, Agile, Scrum and Kanban are among the most notable.

Whichever methodology you select for your project, it's important to remain consistent and to ensure that you have buy-in and support from key stakeholders beyond your development team. This includes management, executives and project customers. Some methodologies—scrum, for example—require active engagement from non-engineering resources. Securing the support of these stakeholders is essential to a successful project.

Over the course of my career, I've found success with the more agile-leaning methodologies like Scrum and Kanban. For each company and in each case, however, I've been careful to adapt those processes where needed to fit within the culture and context of the team and the organization in which the team operates.

Tooling and Development

Tooling has grown in importance among web application developers in the last decade. As the complexity of web applications has grown, so too has the variety, scope and scale of the tools developers use to create these applications. In that context, package managers, module loaders and bundlers, linters, task runners and UI frameworks are the key building blocks for developing a robust Vue.js application.

Let's take a look at some of the current popular tooling options for Vue.js applications.

Package Managers

Package managers help you manage dependencies for your application, and ensure that these are available for every environment in which your app will run. For example, npm is often used to fetch dependencies for development, in addition to those needed for production.

Development dependencies are tools that you need during the creation of your app, but are not required in production. Examples include unit testing tools, code linters or transpilation libraries like TypeScript, which produces your production code assets at build-time and is not needed in production. Production dependencies are those dependencies that are required for your app to run in production, like Vue.js itself, CSS and UI libraries or utility tools like moment.js.

Here are a few tools to consider when choosing a package manager.

PACKAGE MANAGERS
npm
Yarn
jspm.io
Bower

Task Runners

JavaScript task runners enable you to automate many tasks that are common to complex web application development and deployment. Managing and performing these types of tasks are error-prone when left to humans. However, task runners make managing and performing these tasks simple and speeds up application development and deployment.

Task runners can be used to start a local development server, compile, minify/uglify assets, run test suites and more. In recent years, Webpack has become the de facto standard in the Vue.js community, though there are other solid options available.

TASK RUNNERS
webpack
npm
Grunt
Gulp
Tree.js

Linters and Enforcement

When working as a part of a team of engineers, a common goal is to ensure that each piece of code authored is written as if it were coded by a single person. The “common voice” idea extends from things like application structure and error-handling, all the way down to formatting and code styles.

There are three types of tools that aid in enforcing a consistent coding style within a team, and should be configured before coding begins.

TOOL	EXAMPLES
Code linters	ESLint, CSSLint, Standardjs
Code style checkers	ESLint, Standardjs
Code editor formatting/style	.editorconfig

The Vue.js CLI

Many developers using modern frontend frameworks can quickly become overwhelmed with all of the setup and configuration required to get a simple application up-and-running. What used to take minutes in the early JavaScript and jQuery days now seems to require hours to get package managers, linters, task runners and testing tools all working together. To combat this tooling fatigue, Vue.js offers a command-line utility that provides all the app set-up and configuration for you, so you can get to coding in minutes. It's an extensible tool that is perfect for most projects, provides a variety of templates for Webpack, PWA and basic usage and also scaffolds production-ready code from the first command.

UI Components

Building any non-trivial web application is going to require you to create UI components beyond what the browser itself has to offer. Textboxes, labels and dropdown lists will only get you so far.

When it comes to UI components, there are a lot of solid options, both open-source and commercial.

TOOL	DESCRIPTION
Progress® Kendo UI®	Popular commercial UI component library that includes data grids, charts, schedulers and many other common components. Supports common customizable themes and easily integrates into any Vue project. Get started quickly.
Vuetify	An open-source library containing many of the components needed to create applications that adhere to the Material Design specification.
Semantic-UI-Vue	Official Vue.js components for the Semantic UI framework. Designed to help create responsive layouts with "human-friendly HTML."
Bootstrap-Vue	Vue.js components for this popular CSS framework and grid system, often used for application layout.

Testing Methodologies

How you test, the tools you choose for testing and the ways you decide to implement tests are less important than the fact that you prioritize some form of testing in your application. It's likely that you'll want to test each module or unit of your code with unit tests. As you start to string code units together into a complete application, you'll want to consider functional end-to-end testing. The list below contains some popular unit and functional test tools for Vue.js applications.

TOOL	PURPOSE
vue-test-utils	Official test library for Vue.js. Provides methods for unit testing Vue components with support for Jest, Mocha, tape and AVA testing frameworks.
Jest	The Jest testing framework is a zero-configuration test framework designed for React, but it has become popular for testing Vue apps as well.
Karma	The Karma test runner is ideal for writing and running unit tests while developing the application. It can be an integral part of the project's development and continuous integration processes.
Navalia	Navalia is an end-to-end test-runner and browser automation framework with a simple API and support for complex user-interactions.

Codebase Distribution Strategies

The days of building web-based applications solely for the browser are well behind us. In today's day and age, it's possible to use web technologies to build desktop and fully native mobile applications. Modern language interpreters and transpilers like Babel and TypeScript make this possible by converting the JavaScript we create into an Abstract Syntax Tree, or AST. An AST is a series of commands that describe our code, but which is written at a higher level than our code itself. ASTs make our code portable, meaning that other programs can take these AST representations of our web code and output whatever code is needed for another platform or target.

For example, NativeScript (a popular cross-platform native mobile application framework) uses an AST to convert JavaScript and TypeScript code into native code that delivers a fully native application experience.

For your own application, you need to consider both your initial target and any future platforms on which you'll want to serve your app.

Browser Only

If your app will only run in a browser, then your strategy is simple: deploy to a single server environment and your code will be served to the browser like a traditional web app.

Server-Side Rendering

Server-side rendering provides huge performance and SEO gains over solely rendering Vue.js applications from the browser. Because Vue.js rendering in the DOM is separate from the core engine, it's possible to render views on the server and merely send HTML to the browser to represent the initial state of the application. Once the server has rendered these initial payloads, Vue picks up on the client-side, hydrating JavaScript and application logic when the app is ready. Server-side rendering is simple in Vue—simply follow the Complete SSR Guide from the Vue team, or use the Nuxt.js community project.

Mobile and Desktop

If you're considering targeting mobile devices or desktop computers with your app, here are some tools to consider for leveraging your Vue.js codebase on non-browser platforms

TOOL	PURPOSE
NativeScript + Vue.js	Community-driven NativeScript plugin that enables you to build cross-platform iOS and android apps using Vue.
Weex	Cross-platform UI framework that uses Vue as its JavaScript runtime.
Electron	Build cross platform desktop apps with JavaScript, HTML and CSS.

Progressive Web Apps (PWAs)

Progressive Web Apps use modern web capabilities to deliver app-like user experiences, particularly on mobile devices. They evolved from pages in browser tabs to immersive, top-level apps, maintaining the web's low friction at every moment. Some of the key characteristics of PWAs include:

- Progressive - Work for every user, regardless of browser choice because they're built with progressive enhancement from the start.
- Responsive - Fit any form factor—desktop, mobile, tablet or whatever is next.

- Connectivity independent - Enhanced with service workers to work offline or on low-quality networks.
- App-like - Use the app shell model to provide app-style navigation and interactions.
- Fresh - Always up to date, thanks to the service worker update process.
- Safe - Served via Transport Level Security to prevent snooping and ensure content hasn't been tampered with.
- Discoverable - Are identifiable as "applications" thanks to W3C manifests and service worker registration scope, helping search engines find them.
- Re-engageable - Make re-engagement easy through features like push notifications.
- Installable - Enable users to "keep" apps they find most useful on their home screen without the hassle of an app store.
- Shareable - Easily share via URL and not require complex installation.

Vue has a lot of features that make creating PWAs easy, as well as a CLI template that scaffolds a full-featured PWA boilerplate project with a single command.

Define Your Deployment Strategy

When you're ready to start moving your application closer to test, staging or production environments, you need a plan to regularly move your code between environments. A continuous integration (CI) server is the ideal solution for managing your deployments, regardless of whether you intend to deploy to a live environment with every push.

Setting up for CI also improves your approach to local development, especially when you think about your CI approach from the start. Ideally, anything you do during CI, you should first do locally to ensure that other developers on the team can easily replicate your setup, and that your CI system is properly configured to obtain dependencies, run tests and the like.

For Vue.js applications, I recommend considering one of the following CI tools:

- Travis CI
- Jenkins
- Codeship

JavaScript Error Monitoring

A JavaScript error monitoring tool should be used to capture runtime errors that occur in your staging and production environments. Typically, you won't use this tool in development once you get it configured.

This is one of the most commonly skipped steps in the creation of web apps, but should not be overlooked. A quality tool, like one of the tools below, will save you countless hours tracking down production issues that are difficult to replicate in a local environment.

1. Track:js
2. Sentry
3. Raygun

Style Guide, Architecture and State Management

Defining a style for your team and project is essential, as is deciding the architectural styles you'll implement.

Style Guide

As a mature application framework, Vue.js is very opinionated about the guidance it provides and the patterns you're encouraged to use. Before starting your Vue application, consider a careful review of the Vue.js Style Guide for some pointers, recommendations and common patterns to consider. Doing so will help you scale your app as it becomes more mature and well-trafficked.

State Management

State management is offered for non-trivial Vue.js via the Vuex extension. For developers more familiar with Redux via React apps, this library is also supported, though not as deeply integrated with the framework.

TOOL	PURPOSE
Vuex	Centralized state management patterns and library for Vue.js apps. Integrates with the official Vue dev tools.
Redux	Robust state container that lives apart from components and can help manage complex application state. Commonly used with React, but is platform-agnostic and can be used with Vue apps, as well.

Backend API

When building your web application, you'll want to make sure you consider data storage and access from the start. Even if you're working with an existing data repository, I highly recommend wrapping that store in an API and taking an API-first approach to your development project.

API-first development means that you document, build and test your API first, which results in a relatively stable API before you write any dependent application code. However, this doesn't mean frontend development has to wait. During API construction, frontend developers can build prototypes as early consumers of the API and provide valuable feedback to API developers.

The strongest argument in favor of API-first development is to reduce the chances that API bugs or weaknesses end up propagating into or are amplified by the data later. As much as possible, you don't want your frontend to have to bend over backwards to work around or mask deficiencies in your API later. Having a documented and solid API before a line of production code is written can go a long way to saving you headaches in the future.

So, build your API first. Document it, test it and be ready to evolve it as you build against it.

A few key details to remember when taking an API-first approach is that security and authentication need to be embedded in the API, and that data environments should be kept separate. When developers are coding against the API, they should be working with development data, never live production resources.

Performance Strategies

From a performance standpoint, it's worth investigating how to get the most out of your Vue.js application early on in the development process. Let's investigate some ways to ensure that your app runs well once you get it live.

Polyfills & Browser Support

Modern frameworks like Vue.js owe some of their popularity to the fact that they allow you to use cutting edge JavaScript language features (usually referred to as ES6, ES7, etc. or ECMAScript 2016, 2017) without having to worry too much about browser support. This has certainly sped up language feature adoption and enabled the TC-39 committee (which oversees the EcmaScript standard) to move fast when it comes to shipping new features for the language.

That said, when targeting modern JavaScript language features and browser capabilities, you want to be sure to only load polyfills or additional code when needed by a browser, and not for every user of your application. Using the tools below, you can ensure that your app visitors on modern browsers get the fastest experience and native use of modern features.

TOOL	DESCRIPTION
Babel-preset-env	An npm extension for Babel that enables you to specify the browsers and versions you want to support and ensures that Babel transpiles your source to the code required by the browsers you support.
Polyfill.io	A utility library that loads polyfills at runtime when a user visits your site or app. Polyfills are only loaded when needed by the browser, meaning that modern browsers won't be hit with the network cost of downloading code they don't need.

Bundling

Bundling code enables us to remove unused or "dead" code and minify our builds before we deploy, as well as shrink the overhead of that first set of JavaScript resources. Bundling tools also include capabilities to rename variables, functions and properties in order to obtain the smallest payload size possible for your servers to transfer over the network.

Tree-Shaking

Tree-shaking enables you to remove unused imports from your codebase, thus reducing the size of your bundle and the final assets shipped down to the browser. As long as you are using ES2015 module syntax in your Vue.js application (i.e. import and export), Webpack 2 and above, along with a minifier like UglifyJS, Rollup or Babel, Webpack will take care of tree-shaking on your behalf.

Lazy-Loading

Lazy-loading is an approach that loads dependent modules only when you need them. For instance, an About component on a homepage would only be fetched when the page is accessed by a user. This keeps initial application payloads small and speeds up the loading experience of your app. For Vue.js applications, lazy-loading is possible by creating async components and using code-splitting with Webpack. Vue's Router documentation provides full details on how to implement lazy-loading for your app.

Conclusion

Vue is a popular new technology that is growing rapidly for use in web applications. To make these applications even better, there is a rich ecosystem of tools and libraries that support it—including, of course, the Kendo UI library of Vue components.

The strategies we've presented here for tooling, testing, performance, style and distribution will help you get started on the right path toward success. Once you have mastered the basics and have developed and deployed efficient and effective web apps, there is a rich and diverse set of tools, libraries and methodologies for you to explore. But at least you will be doing that with a solid foundation in what works.

About the Author

Brandon Satrom

Brandon is the founder of Carrot Pants Press, a maker education and publishing company, the founder and CEO of Tangible Labs and an avid tinkerer.

About Kendo UI—Our Complete JavaScript UI Component Library

Kendo UI helps you build better Vue web apps faster. Kendo UI is the leading UI component library for web applications and supports all popular framework technologies. In addition to Vue, Kendo UI also supports jQuery, Angular and React. This library includes a wide array of components ranging from complex data grids, charts and schedulers to basic elements like buttons, sliders and dials. Kendo UI supports a selection of popular themes out of the box and also includes an easy-to-use theme editor that enables you to quickly match any custom web theme. Take a look for yourself with a free trial of Kendo UI!

Get started

About Progress

Progress (NASDAQ: PRGS) offers the leading platform for developing and deploying mission-critical business applications. Progress empowers enterprises and ISVs to build and deliver cognitive-first applications that harness big data to derive business insights and competitive advantage. Progress offers leading technologies for easily building powerful user interfaces across any type of device, a reliable, scalable and secure backend platform to deploy modern applications, leading data connectivity to all sources, and award-winning predictive analytics that brings the power of machine learning to any organization. Over 1,700 independent software vendors, 100,000 enterprise customers and 2 million developers rely on Progress to power their applications. Learn about Progress at www.progress.com or +1-800-477-6473.

Worldwide Headquarters

Progress, 14 Oak Park, Bedford, MA 01730 USA
Tel: +1 781 280-4000 Fax: +1 781 280-4095
On the Web at: www.progress.com
Find us on facebook.com/progresssw twitter.com/progresssw youtube.com/progresssw
For regional international office locations and contact information, please go to www.progress.com/worldwide

Rev 18/10 | 171002-0040 / RITM0029821

Progress