Sencha

An Idera, Inc. Company

The Evolution of JavaScript Tooling

A MODERN DEVELOPER'S GUIDE

Alan R. Weiss

INTRODUCTION

JavaScript application source code has traditionally been hard to understand, due to code being spread across JavaScript, HTML, and CSS files, as well as events and data flowing through a number of non intuitive paths. Like all software, the JavaScript development environment includes bundlers, package managers, version control systems, and test tools. Each of these requires some learning curve.

Inconsistencies and incompatibilities between browsers have historically required various tweaks and special cases to be sprinkled around the code, and very often fixing a bug in one browser breaks something on another browser. As a result, development teams struggle to create and maintain high quality, large-scale applications while the demand for what they do soars, especially at the enterprise-application level where business impact has replaced “How many lines of code have you laid down?”

To deal with this complexity, the open source community as well as commercial companies have created various frameworks and libraries, but these frameworks and libraries have become ever more complicated as they add more and more features in an attempt to make it easier for the developer. Still, frameworks and libraries offer significant advantages to developers and can also organize and even reduce complexity.

This guide discusses some of the more popular frameworks and libraries that have been created to ease the burden of writing complex user interface (UI) code and how enterprise applications, especially data-intensive apps, can benefit from using these frameworks and UI components to deliver applications faster, with better quality, and yet stay within any development shop’s budget.

Complexity of modern web development

Andrew S. Tanenbaum, the inventor of Minix (a precursor to Linux often used to bring up new computer chips and systems), once said, “The nice thing about standards is that you have so many to choose from.” [1] Browsers followed a number of standards, but not all of them, and many just went their own way.

That’s where the trouble started—the so-called “Browser Wars.” How each browser displayed the data from these websites could be quite different. Browser incompatibilities still exist today, and one could say they are a little worse because the Web has gone mobile.

Developing in today’s world means being as compatible as possible with as many of the popular web browsers as possible, including mobile and tablet.

What about mobile?

Learning Android Java (Android) can be difficult if the developer hasn’t been brought up with Java. For Apple iOS, Objective C is a mashup of the C programming language and Smalltalk, which is different but not entirely alien to C++ developers. (After all, object-oriented concepts are similar.) But given the coming of (Apple) Swift and a new paradigm, “protocol-oriented programming,” Objective C has a questionable future.

In contrast, the JavaScript world, through techniques such as React-Native or Progressive Web Apps, allows for development of cross-platform apps that look like native apps and are performant. From a business perspective, an enterprise can gain a number of advantages by only using one tool set to build sophisticated web and mobile apps.

Constant Change Causes Consternation

The JavaScript world is particularly rich in how much functionality and how many packages are available. The number is staggering. The number of key technologies that help developers create applications faster is also large, but the rate of change in this field causes what’s called “JavaScript churn,” or just churn. For example, when Angular moved from version 1 to 2 (and again from 3 to 4), the incompatibilities required serious porting time. Until we embrace emerging Web Components standards,[2] not everything will interoperate with everything else.

One thing that can be said is that investing in old technologies not backed by standards can be career-limiting, thus the importance of ECMA and ECMAScript standards as well as adherence to more or less common design patterns (most programming is still, even to this day, maintenance of existing code rather than fresh new starts and architectures). Using commonly used design patterns like Model-View-Controller (MVC), Model-View-Viewmodel (MVVM), and Flux means that your code can be modified and maintained more easily than if you invent an entirely new paradigm.

Having large ecosystems and using popular, robust, well-supported tools is one strategy proven year after year to yield positive results for the company and the developer's career, and having industry-common or industry-standard libraries means that you can find teammates to help with the development and testing. Modern development methodologies practically demand the use of frameworks, reusable libraries, and well-designed APIs and components.

1. Andrew S. Tanenbaum, ***Computer Networks*, 2nd ed., p. 254**
2. https://www.webcomponents.org/specs

Popularity of Modern Frameworks and Libraries

Stack Overflow, an incredibly popular developers website used for questions and answers (#57 according to Alexa as of January 2019), tracks a great deal of data on the popularity of various technologies and has become a go-to source for developers. Their most recent survey continued to show the incredible popularity of both JavaScript and JavaScript libraries and frameworks:

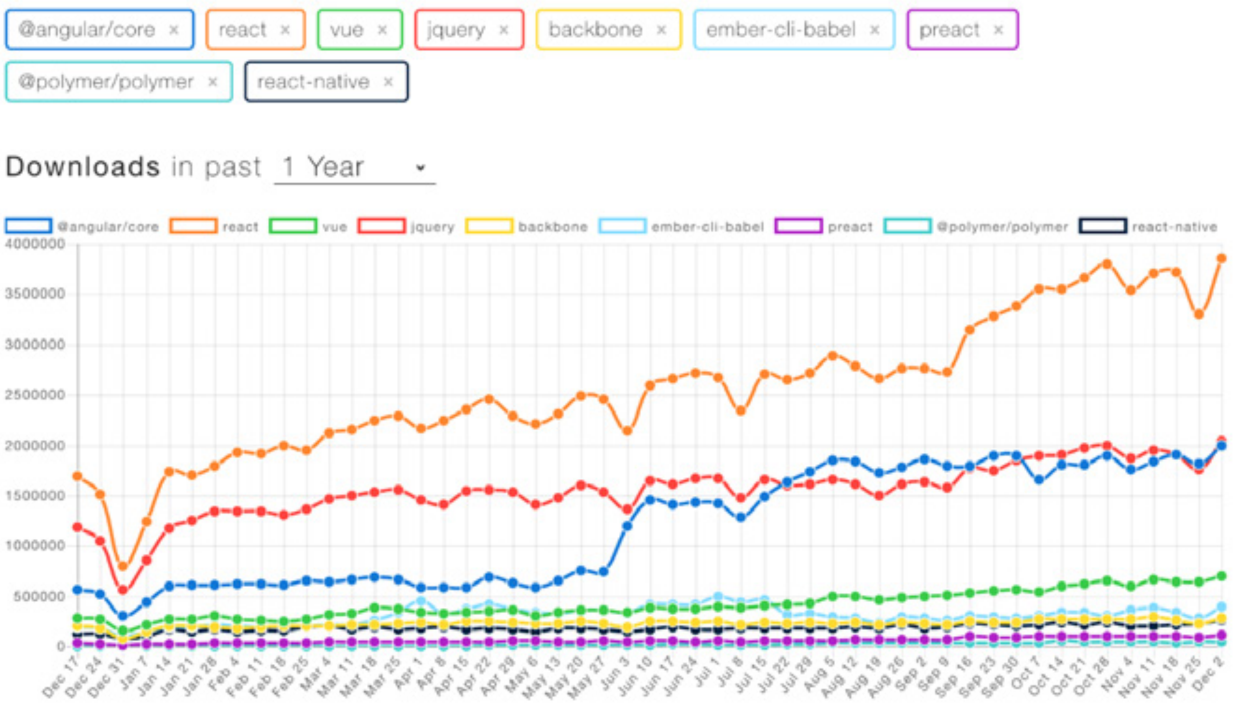


Figure 1: *NPM Downloads of Popular Frontend Libraries*

According to Stack Overflow, based on the type of tags assigned to questions, the top eight most discussed topics on the site are JavaScript, Java, C#, PHP, Android, Python, jQuery and HTML — not C, C++, or more exotic languages like Ocaml or Haskell. If you're building websites, you're very likely going to want to use technologies that are popular because the number of open source and commercial/supported products provides you with the ability to code and test more quickly, resulting in faster time to market.

What this means to developers is that the JavaScript world continues to lead all others in the number of developers, and while older technologies like jQuery are still popular, clearly React and Angular are important and continue growing. The newcomer, Vue, is also becoming more and more popular.

Selecting Angular, React, or Vue

Angular versus React versus Vue — so many open source tools. Add to that libraries like Backbone.js and a hundred others. How can developers update their knowledge of so many? Which one should they choose? *To some extent this decision is choosing text editors: it's a personal choice, it's fiercely defended, and in the end each might actually work for you.*

If your main concern is popularity so you don't get boxed into learning a complicated, rich programming environment only to see support wither away, then React is clearly "winning" as the long-term trend line shows. But popularity is only one attribute in a long shopping list of important decision factors.

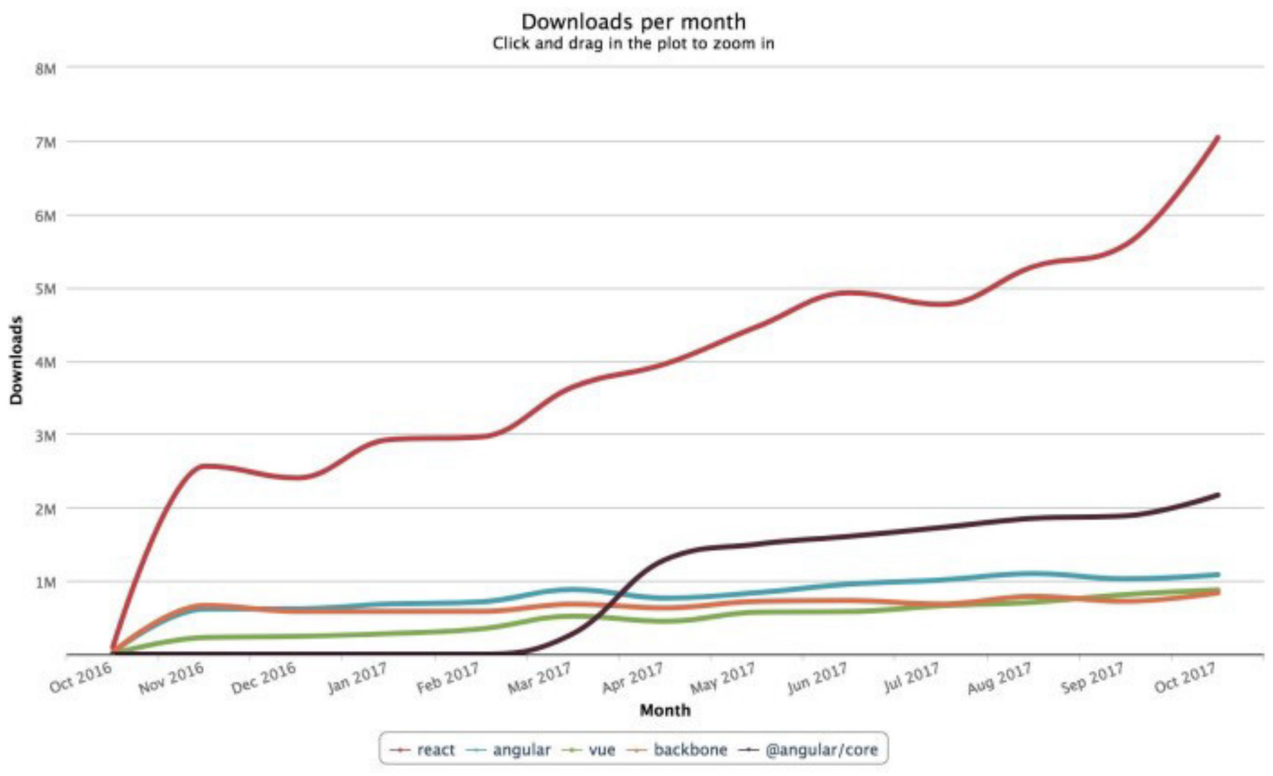


Figure 2: *Long-term trend lines of various popular frameworks and libraries*

Let’s examine each of these in an easy-to-digest format.

React

Best for: Websites that have a lot of dynamic data and desire very high performance.

React is the most popular JavaScript library. Developed and open sourced by Facebook in 2013, it quickly became important for the development of huge web applications that involve dynamic handling of data. React’s big strength comes when developers have to create complex applications and want to modularize them so that pieces can be reused. React, by its very nature, has performance advantages (see the guide “When to Choose ExtReact” for more information on how React uses Virtual DOMs) and allows developers to write in JSX. React is especially suited for real-time data handling. Instagram and WhatsApp, two extremely popular Web applications, use React.

PROS	CONS
Easy to learn. React has relatively simple syntax and a lot of HTML. By contrast, Angular requires a steep learning curve and learning TypeScript (strong typing). React works with HTML, CSS3, and other familiar, easy-to-learn tools.	**React continues to evolve**. Developers need to keep learning and keep on top of React’s changes.
React is a JavaScript library, not a framework. This means that it provides a declarative method of defining UI components. It can also be easily integrated with other libraries.	**JSX can be tough**. React actively makes use of JSX, allowing a mixture of HTML with JavaScript. Although JSX can protect the code from hacking injections, it does involve complexity and a learning curve.
Separate data and presentation layers. React provides separation of data and presentation layers and supports MVC, MVVM, and Flux design patterns.	**React is not a full framework**. Integrating this library into a developer’s tool chain (for example, an MVC-supporting framework) needs to be done by the developer.
DOM binding. Developers do not have to bind DOM elements to functionality. React handles this by splitting the binding across multiple areas of the code. React also includes the concept of a Virtual DOM, and only items that have changed get updated and displayed to the user	**Documentation**. One of the most frequent complaints against React is the lack of official documentation.

Reusable components. React provides the ability to reuse code components of any level anytime. This saves a lot of time during development and makes it easier for developers to handle upgrades. Any change made to one component has no effect on others.	
Unidirectional data flow. React utilizes downward data binding. This helps to make sure that any changes made to child structure does not end up affecting their parents. This makes the code more stable, and all the developer needs to do to change an object is modify its state and apply updates. React enforces top-down data flow.	
Interoperate with other components. React can be used as the View component of Angular or other framework, because it does not tie you to a specific technology stack.	
Search Engine Friendly. React can run on the server, and the virtual DOM will be returned and rendered to the browser as a regular web page.	
Supports Functional Programming. React can use the Redux state management library.	

Angular

Best for: Developers that don't want to hassle with integrating libraries into other frameworks.

Angular, or in the past AngularJS, is older than React and is the second most popular in our analysis. It was created and is maintained by Google and was developed to address the need for a **complete framework** that could deal with single-page applications. A single-page application (SPA) is a web application or web site that updates information and data by dynamically rewriting the current page rather than loading entire new pages from a server. Therefore, the application looks more like a desktop PC application but also works well on mobile devices if designed properly. All code is loaded with a single page load, or only partial pages are loaded dynamically when the page is updated. Angular can have a relatively long learning curve as it is really a complete development framework, but Angular itself contains no UI components. Sencha fills that void with pretested, consistently developed UI elements with their ExtAngular product (read our free guide, "When to Choose ExtAngular" to learn more).

PROS	CONS
Single-Page Applications (SPAs). Angular was designed for this very scenario.	**Complex syntax**. Angular can be complicated, and doesn't fit every programmer's style of writing code.
Progressive Web Applications (PWAs). If you want to build apps that look like native applications across mobile, tablet, or desktop, Angular has this.	**Steeper Learning Curve**. Angular can be difficult to learn, though the documentation and online help available to some extent helps mitigate that.
Full-scale framework. The developer doesn't need to integrate Angular into their own MVC framework and tools. Also supports Flux design pattern.	**Difficult backward compatibility**. Moving Angular apps to more recent versions can take awhile as porting can be difficult due to version changes.
Documentation. Unlike many web technologies, Angular has extensive documentation with a lot of YouTube videos available. This helps shorten the learning curve.	**High JavaScript churn**. Because Google and the open source world is constantly improving Angular, the developer has to keep up with the new versions or risk developing apps that will require significant porting in the future if they have to be changed.
Universal TransferState API and DOM. With this new design, code can be shared between the server and the client, which can improve performance.	
Build optimizer. Like a good optimizing compiler, this removes all unnecessary runtime code, reducing application size and improving performance.	
Router hooks. Router cycles can now be tracked from the start of running guards till the activation has been completed.	
Two-way data binding and MVVM (Model-View-View-Model). Bidirectional binding minimizes the risk of errors and enables a singular behavior of the app. For MVVM design patterns, it has the capability of allowing developers to work separately on the same app section with the same set of data.	

Google backs Angular. Just as Facebook uses React, Google uses and is continuing to develop Angular. It is also used by Google Adwords, so long-term support is likely.	

Vue

Best for: Developers with more simple applications and a lower learning curve.

The new kid on the block and gaining in popularity at an extraordinary pace is Vue. Between November 2016 and October 2017, one estimate (npm downloads) shows Vue growing at a ridiculously high rate of 13,933.4%, but the absolute download numbers show React at over 7 million downloads in October 2017 versus Vue's nearly 900,000, with Angular in the middle at over 2 million downloads from npm. Interestingly, all JavaScript frameworks and libraries continue to grow at torrid paces. The source of this data is npm-stat: https://npmcharts.com/

Vue strikes a middle ground between full-blown framework and "just a library." Noncore functions such as routing, state management, build toolchains, and the CLI are external, but they are all officially maintained, well documented, and designed to work together. However, you don't have to use them all. Angular enforces a certain structure on how code is organized; Vue does not.

According to noted developer John Hannah:

"React and Vue are rather similar, although there are some key differences.... This makes sense as Evan You, the developer of Vue, used React as one of his inspirations. They are much more like one another than they are to, say, Angular." From the Vue documentation, we see that both React and Vue:

- Utilize a virtual DOM for performance reasons
- Provide reactive and composable view components
- Maintain focus in the core library, with concerns such as routing and global state management handled by companion libraries

Source: John Hannah, https://jsreport.io/how-is-react-different-from-vue/

Like React, Vue emphasizes performance but, being smaller, is easier to learn initially.

PROS	CONS
Easy to learn. Uses templates rather than JSX. These templates are extensions of HTML, not JavaScript.	**Much smaller ecosystem**. Both React and Angular, being more mature, have much larger ecosystems than Vue.
Vue, like React, is a JavaScript library, not a framework. This means that it provides a declarative method of defining UI components. It can also be easily integrated with other libraries.	**Less JavaScript churn**. Vue is a library, and the developers seek to keep all of the companion libraries up to date.
Simpler styling. Styling is via style tags or CSS, with the default being simpler style tags.	
State management, routing, etc. This is handled by external (but coherently maintained) external libraries. Does not enforce a rigid code organization/design.	
Supports many design patterns. This should really be phrased as "doesn't enforce a design pattern on you," but Vue can support MVC or Flux.	

Developer-Focused Quick Look Table:

Programming is not only a challenging profession, it's an intensely personal, language-based pursuit that requires intelligent people—people almost always under enormous schedule and resource pressures. The selection of which JavaScript tools to use is, as we said, intensely personal and often reflects how developers think about problems. It is, in short, a people-oriented business and usually a "team sport" (hence the need for tools such as Assembla to manage source code across many projects and many people).

This quick-look table may be more useful to you than a technical analysis because it breaks down these three technologies into how they affect a person's ability to actually get the job done and provides a side-by-side rating and ranking. These ratings and rankings are a matter of opinion and might change based on a developer's familiarity, coding style, type of design patterns they use, the availability of others around them who can provide help and guidance, and quite likely a hundred other factors.

ATTRIBUTE	REACT	ANGULAR	VUE
Popularity	1	2	3
Backers	Facebook	Google	Laravel and Alibaba
Library (L) or Framework (F)?	L	F	L
Object-oriented coding style	✓	✓	✓
Imperative/functional process coding style	✓	✓	✓
If you need guidance, structure, and a helping hand		✓	
Learning curve	Moderate	High	Low
Coding speed (how fast can you deploy?)	Medium	Slower	Fastest
Need flexibility	✓		
Big ecosystem, lots of packages	1st	2nd	3rd
Love JavaScript, only want JavaScript	✓		
Building large applications	✓	✓	
Availability of developers	✓	✓	
Single-page application		✓	
Highest performance	1st	3rd	2nd
Quality and quantity of documentation	2nd	1st	3rd

Supports legacy JS code?	No, needs rewriting in JSX	Yes, but needs conversion to TypeScript	Yes, but may need conversion to Vue.js syntax

Source: Davison Pro, https://medium.com/@davisonpro/react-js-vs-angular-7a7bed92b5f6 and others

Building on the Shoulders of Giants: Why reinventing the wheel slows you down

Developing sophisticated frameworks, reusable libraries, and especially UI components widgets requires valuable time, and time is money. The burden of searching for and cobbling together different Free and Open Source (FOSS) tools can be expensive and time-consuming and can result in disappointment as some tools might be incompatible or require additional shim layers.

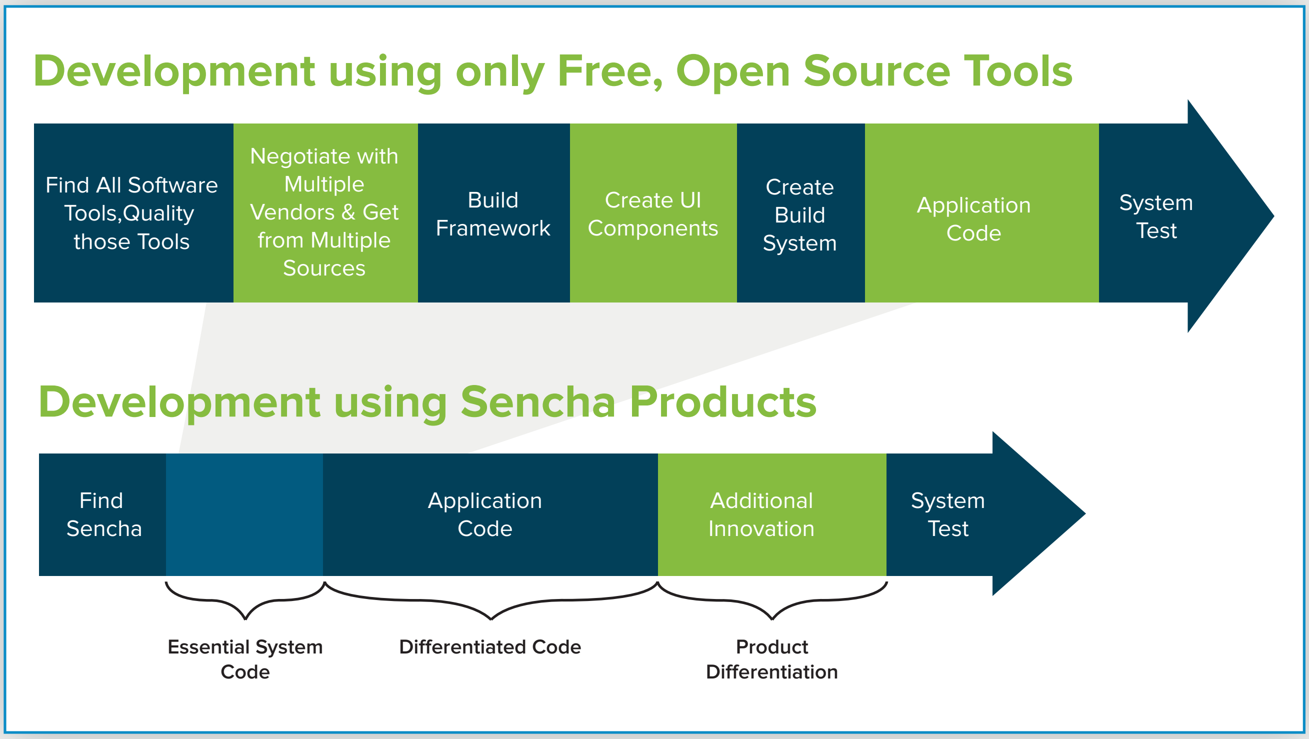
Today, key performance indicators have changed. The relentless drumbeat for CIOs and heads of development is not for lines of code, the number of function points delivered, and other engineering metrics. They have given way to business objectives, and because that is how CIOs and increasingly CTOs are being measured, so too are developers being measured by their impact on the business. These business objectives now include profit and cost reductions from increased efficiencies; they often also include time to market (TTM, or time to DevOps deploy a working solution), lower barriers to entry (LBE, starting with open tooling and open source or using free trial downloads to prototype), and total cost of ownership (TCO; how much will this technology cost me over the short term and long haul?).

As a result, developers are becoming more strategic in what matters for them—becoming well aware of and aligning themselves to their employer's business strategies. But more than that, the bar has been raised: now mobile, phone, tablet, desktop PC, even set-top boxes and smart televisions are potential targets for deployment.

Fortunately, solutions exist: adopting a good framework or set of libraries can save the developer time, money, effort, heartache and provide bright futures. For example, adopting React, Angular, or Vue as the basis of one's development efforts offers a multitude of benefits, and these benefits can be selected for based on requirements and circumstances.

Using robust, well-tested, cross-platform UI widgets such as Sencha Ext JS, ExtReact, or ExtAngular can save many person-months of finding what you need to use, coding and testing time. Sencha ensures all of these elements work with the vast majority of web browsers, and having your web application work on Android and iOS platforms with no more effort can save you from having to learn difficult and complicated languages. Just use JavaScript.

To preserve developer choice, it is quite possible to mix and match UI components from open source and Sencha commercial products. However, increased productivity comes from finding all you need from one vendor, reducing a developer's search time.



Visit www.sencha.com to learn more, access free 30-day trial downloads or a limited commercial license Community Edition.

Summary

START APPLICATION DEVELOPMENT EARLIER	TUNED FOR EFFICIENCY OF DEV	DO BUSINESS IN ONE STOP
Focus software development at the application layers, freeing up resources to enable product differentiation.	Software and documentation is readily available, pre tested, pre qualified, pre integrated, and tuned for code performance efficiency.	Sencha stands behind all of its products, providing one source for all Technical Support – a lifeline when your app is on the line!