Vue

Vue.js jest pierwszym frameworkiem, który pragnę przybliżyć. Pierwszą rzeczą, na którą trzeba zwrócić uwagę, jest **bardzo dobra dokumentacja projektu**. Poza bardzo dobrym opisem aplikacji tworzonej za pomocą tego frameworka, dokumentacja zawiera również poradnik, jak przenieść aplikacje napisane w poprzednich wersjach do wersji najnowszej. Nie jest to coś bardzo powszechnie spotykanego, zwłaszcza dla projektów, za którymi nie stoją duże korporacje.

Użytkownicy, którzy często korzystają z konkretnej strony internetowej mogą zapisać skrót do tej witryny na ekranie głównym urządzenia mobilnego. Ponieważ większość informacji jest przechowywana w pamięci podręcznej, aplikacja ładuje się natychmiast. **Dzięki działaniu aplikacji w tle najnowsze informacje są buforowane za każdym razem, gdy urządzenie jest podłączone do Internetu.** Możliwość pracy offline (bez połączenia z Internetem) jest jedną z głównych zalet progresywnych aplikacji internetowych.

PWA nie wymagają specjalnego projektowania ani programowania. Wystarczy **przekonwertowana zwykła strona internetowa,** do której należy dodać kilka funkcji natywnych, dopasować do wymagań mobile, przemyśleć jej wygląd. Chociaż wygląd aplikacji może nie odróżniać się od zwykłej [strony www](https://smartbees.pl/uslugi/strony-internetowe), jednak warto wziąć pod uwagę najnowsze standardy i zastosować jeden z dostępnych wzorców projektowych, by nikt nie miał wątpliwości, że strona jest zgodna z PWA.

Framework ten od początku był projektowany z myślą o stworzeniu strony podatnej na interakcje. Dzięki takiemu podejściu może on zostać potraktowany jako biblioteka i zostać zaimportowany na dosłownie każdą stronę istniejącego już projektu, dając tym samym nowe funkcje i możliwości

Then VueJS will handle all the frontend activities without reloading the page, which results in a perfect user experience.

* It offers Reactive **two-way data binding** like Angular. It also supports **Virtual DOM and Event Sourcing** like React.
* It is 100% community-driven. As a result, Vue.js is not driven by the need of one particular organization.

When to use Vue.js:

* Progressive App Development or modernizing a large enterprise application iteratively.
* When performance is very critical, and SEO is vital.
* Faster development velocity and lower barrier to entry is a key requirement.

When not to use Vue.js:

* Speed of development is the most important criteria.
* When a company wants to use the same framework for multiple platforms.
* Large Enterprise App Development.

Narzędzia te umożliwiają tworzenie dużych i skomplikowanych aplikacji typu Single Page Application

Vue.js z kolei pozwala na prosty import biblioteki i rozpoczęcie korzystania z frameworka. Nie trzeba używać plików .vue, które wspiera biblioteka. Framework nie wymusza pisania aplikacji w TypeScript. Wystarczy prosty import potrzebnych do solucji bibliotek i można zacząć tworzyć aplikację.

Vue.js świetnie działa na takich stronach. Można swobodnie kontrolować tylko jedną część drzewa DOM, tą którą chce się kontrolować, resztę pozostawiając bez zmian.

Ostatnim pytaniem, które w tej kwestii można zadać, jest, czy jest zalecany lub potrzebny kompilator, który bedzie porcjował odpowienie dane.

Przy Vue.js nie jest to wymagane, można zastosować kompilator ale przy tworzeniu aplikacji na pojedynczą stronę może się to okazać albo przerostem formy nad treścią albo usprawenieniem budowy Aplikacji. Tutaj odpowiedź jest zmienna w zależności od wielkości aplikacji. Taka możliwość wyboru podczas budowy jest niewąpliwą zaletą.

Ponownie, by odpowiedzieć na pytanie, jak framework poprawnie skaluje się wraz ze wzrostem rozmiarów aplikacji, trzeba odnieść się do kilku bardziej szczegółowych kwestii.

Vue.js z kolei nie narzuca takich praktyk. Jest to zdecydowanie jego wada jak i zaleta. Przy przeglądaniu dokumentacji można zaobserwować pewne techniki, ale z tej racji, że nie są narzucane, nie są też powszechnie stosowane. Zmianę wprowadza natomiast oficjalnie udostępniony CLI. Pozwala on na szybkie budowanie skalowalnych aplikacji z narzuconymi pewnymi technikami. Wśród deweloperów Vue.js panuje swoboda, każdy stosuje taką metodę, jaka mu odpowiada najbardziej.

Drugą ważną kwestią jest to, czy deweloperzy muszą w dużej mierze polegać na paczkach kodu nie wywodzących się od twórców frameworka oraz czy wystarczająco dużo funkcji i możliwości oferuje framework sam w sobie.

Przy Vue.js jest troszkę inaczej. Jest co prawda oficjalna paczka do routingu, jest paczka od twórców do zarządzania stanem aplikacji o nazwie Vuex. Jednak jeśli chodzi chociażby o walidacje formularzy, to jest to coś, czego nie ma w zaimplementowanym przez twórcę frameworku. Wtedy trzeba polegać na społeczności.

Kolejną, niezbędną do poruszenia kwestią, jest możliwość optymalizacji. To, czy łatwo zoptymalizować kod i czy łatwo go sprowadzić do stanu zadowalającego.

Vue.js sam optymalizuje częściowo kod, więc stosowanie rozwiązań osób trzecich nie jest konieczne.

Ostatnią sprawą jest łatwość zarządzania stanem aplikacji. Bardzo szybko przy rosnących aplikacjach pojawia się ten problem. Ważne wówczas są metody radzenia sobie z tym problemem, to czy jest jakiś wzorzec, biblioteka, czy nie ma nic.

Vue.js rozwiązuje ten problem za pomocą oficjalnej biblioteki Vuex, która ładnie się integruje z aplikacją. Nadal trzeba poznać metodę, jak to zrobić poprawnie, jakie praktyki są najlepsze przy tego typu zagadnieniach, ale jest to duże ułatwienie, ponadto oficjalne narzędzie sprawia że nie ma rozwiązań konkurencyjnych dla tego stanu aplikacji. A co z tym idzie, Internet jest pełen poradników i wskazówek

For the sake of speed, Vue.js utilizes virtual DOM: Think of this as a copy of an original DOM that figures out what elements to update, without rerendering the whole DOM. This approach makes page rendering pretty quick and improves application performance.

### **Reactive two-way data binding**

Another benefit in DOM manipulations is two-way data binding inherited by Vue from Angular.

In Vue, the bound data is updated reactively as are DOM objects, which is excellent for any application requiring real-time updates. Developer-wise, Vue’s reactivity will make data updating clearer and easier to complete.

### **Integration capabilities and flexibility**

An important aspect of any emerging technology is the ability to integrate with existing applications. With Vue.js it’s as easy as pie, because it relies only on JavaScript and doesn’t require any other tools to work.

Vue also allows you to write the templates as you wish: using HTML, JS, or JSX (JavaScript syntax extension). Between its components and lightweight nature, Vue can be used in nearly any project. And, we’re happy to report that switching from [React or Angular](https://www.altexsoft.com/blog/engineering/react-vs-angular-compared-which-one-suits-your-project-better/) won’t be a big problem, as Vue’s internal organization is basically a mix of the two.

### **Concise documentation**

We should give credit to Vue.js [documentation](https://vuejs.org/v2/guide/installation.html). Whether you are a newbie who’s going to learn the framework, or you’re a tech-savvy dude searching for an issue reference: Vue documentation’s got your back here. It’s well-structured and covers all the possible topics, precisely describing everything from installation to more in-depth things like reactivity and scaling of the app.

### **Reactivity complexity**

We have mentioned the two-way data binding implemented in Vue to manage DOM updates. While it’s a handy tool to keep components in sync, there is one issue concerning the way the reactivity system, as it’s called, works.

In plain language, the Vue.js app consists of components that a user can interact with. Each component has its watcher that rerenders the data each time a user triggers the component. The reactivity system rerenders only those chunks of data that were triggered. The thing is that it’s not that smart and often makes mistakes during data reading, so it requires data to be flattened.

However, this is a known issue and it is [addressed](https://vuejs.org/v2/guide/reactivity.html) in Vue’s documentation, providing guidance on how to set up reactivity properly.

### **Lack of support for large-scale projects**

Vue.js is still a young framework. Its community and development team size is still incomparable with more mature Angular.

### **Risk of over flexibility**

We have mentioned flexibility, which is a controversial quality in terms of a big project. Giving your development team too many options might result in philosophically different programming approaches battling inside one team.

### **Limited resources**

While the ecosystem is pretty wide, and there are all the required tools to start developing with Vue, it’s still not as big as React or Angular. To be more precise, just compare the number of plugins available for React and Vue.js: The difference is in hundreds.

## React

React to najpopularniejszy framework spośród tych, które zostaną przedstawione.

best known for its [virtual DOM](https://reactjs.org/docs/faq-internals.html) feature.

**When not to use React:**

* The development team has little to no experience in JavaScript.
* If the speed of development is the most important criteria.
* For simpler, less interactive applications.

With React, we recommend [Express.js/Express](https://expressjs.com/) as a backend service. It is an [unopinionated](https://stackoverflow.com/questions/802050/what-is-opinionated-software" \l ":~:text=Opinionated%20software%20is%20built%20and,for%20which%20it%20was%20developed.)backend framework for Node.js and is also the underlying library for many other Node web frameworks.

The main reason to use React for your project is its ability to fetch fast-changing data, which also needs to be logged. It's an open-source technology that can be used with the MVC (Model-View-Controller) structure with React contributing to the View layer.

it can be used to build components that consist of mainly custom HTML elements. These components can be reused to develop user interfaces efficiently.

### What Makes React and Express Compatible?

* NodeJS and React are both Javascript Languages that can be executed both client-side as well as server-side.
* NodeJS uses the speed optimized V8 engine to address bulk requests that are handled through callback functions to ensure quantity and quality.
* Developers can run the React code directly in the NodeJS environment.
* The React DOM has components specially designed to work with NodeJS, which reduce lines of code, making server-side rendering comparably easy.

### Who should choose to build with React and Express?

* This pair can prove great for developers and development teams with extensive JavaScript experience, since both frameworks use JavaScript.
* React was initially designed to [handle large amounts of traffic](https://www.simform.com/best-frontend-frameworks/#section1) on the internet. Without a doubt, the combination of Express and React can cater to multiple simultaneous requests and help manage server load.
* If you're looking to develop an [SPA](https://en.wikipedia.org/wiki/Single-page_application)(Single Page Application), Express will be a perfect lightweight model for asynchronous data loading.
* Using a Node web framework for the backend increases the scalability of your project. So, if you're looking for your website to grow in the coming days, this is a perfect pair.

Dużym plusem Reacta jest fakt, że po dobrym zaznajomieniu się z frameworkiem istnieje możliwość rozpoczęcia pisania aplikacji na urządzenia działające na systemie Android oraz iOS bez znajomości środowisk Java, Objective C czy Swift.

Ostatnim pytaniem, które w tej kwestii można zadać, jest, czy jest zalecany lub potrzebny kompilator, który bedzie porcjował odpowienie dane.

Mpa: Dla Reacta też jest to możliwe. Nie jest to metoda, za pomocą której pisze się większość aplikacji w tym języku, ale jest to jak najbardziej możliwe rozpoczęcie budowy takiej aplikacji nie jest wcale skomplikowane.

Mpa: można używać w taki sposób, nawet sam zespół Facebooka czyli twórców z niego w taki sposób korzysta. W momencie kiedy będzie dużo małych aplikacji, możliwe że zaistnieje potrzeba wprowadzenia prekompilatora, który będzie porcjował dane, by nie pobierać za każdym razem wielkiej paczki JS z mnóstwem skryptów, które bedą użyte na innej stronie.

React nie wymaga żadnych prekompilatorów. Co prawda najlepiej działa w standardzie EcmaScript 6. Aby z tego standardu korzystać niezbędne jest środowisko deweloperskie, ponieważ przegladarki nie obsługują jeszcze tego standardu.

Ponownie, by odpowiedzieć na pytanie, jak framework poprawnie skaluje się wraz ze wzrostem rozmiarów aplikacji, trzeba odnieść się do kilku bardziej szczegółowych kwestii.

Podobnie wygląda to w przypadku Reacta. Jest kilka praktyk które wywodzą się ze społeczności korzystającej z tego frameworka, ale jest w kwestii programisty, by o nich poczytać, wybrać te, który mu najbardziej odpowiadają, i stosować się do nich.

Drugą ważną kwestią jest to, czy deweloperzy muszą w dużej mierze polegać na paczkach kodu nie wywodzących się od twórców frameworka oraz czy wystarczająco dużo funkcji i możliwości oferuje framework sam w sobie.

React wypada najgorzej w tej kwestii. Zbiór realizowanych przez niego funkcji jest bardzo mały, a dodatki takie jak Redux, który jest odpowiedzialny za stan aplikacji, albo React Router, są stworzone przez społeczność. Nieoficjalne zbiory mają dużą konkurencję, co sprawia, że często trudno podjąć decyzję, która paczka spełnia wymagania projektu.

Kolejną, niezbędną do poruszenia kwestią, jest możliwość optymalizacji. To, czy łatwo zoptymalizować kod i czy łatwo go sprowadzić do stanu zadowalającego.

 sam optymalizuje częściowo kod, więc stosowanie rozwiązań osób trzecich nie jest konieczne. Tak samo jest z Reactem. Podczas tworzenia wersji produkcyjnej aplikacja zostanie odpowiednio zmniejszona.

Ostatnią sprawą jest łatwość zarządzania stanem aplikacji. Bardzo szybko przy rosnących aplikacjach pojawia się ten problem. Ważne wówczas są metody radzenia sobie z tym problemem, to czy jest jakiś wzorzec, biblioteka, czy nie ma nic.

Zespół Facebooka tworząc Reacta nie zapewnił takiego rozwiązania. Środowisko osób trzecich stworzyło jednak dobre biblioteki, takie jak Redux czy Flux. Plusem ich jest to, że bardzo łatwo je zintegrować z aplikacją, ale wymagają dodatkowej nauki.

## Angular

## Angular and Express (Node.js) – The TypeScript-Friendly Pair

The use of NodeJS is not compulsory for your Angular application, but NodeJS makes Angular development easier due to several reasons.

* Node lets you set up a simple webserver to locally host your application in your system.
* Node Package Manager gives you the Angular CLI or ng CLI ( Angular command-line interface). Angular CLI is a valuable tool for scaffolding your app so that there’s no need to write boilerplates manually.
* Angular and Express both support TypeScript. As browsers can’t directly execute TypeScript, you need to transpile TypeScript code into JavaScript. You also need to bundle the JS files and CSS files along with the HTML doc in order to prepare your app for web hosting. Angular CLI assists in doing all these things in the background. By default, ng cli uses Webpack to bundle the application and is very helpful for beginners who are new to Angular development, as it hides such complexities.

Framework ten został stworzony z myślą o dużych aplikacjach. Widać to od samego początku, gdy CLI dostarczone przez zespół Google, oprócz nadzoru nad relacjami plików i sprawdzania składni, wymusza pracę w środowisku TypeScript i narzuca ramy projektu, takie jak nazewnictwo i katalogowanie plików. Aplikacje pisze się w oparciu o komponenty. Dużą zaletą jest to, że każdy komponent może mieć własne reguły CSS, które nie wychodzą poza dany komponent. Do kontroli przepływu danych są serwisy, które również są bardzo dobrze przemyślane. Największą zaletą są jednak typowane zmienne.

Do stworzenia aplikacji w Angularze potrzebny jest TypeScript. Przeglądarki nie rozumieją jego składni dlatego przed uruchomieniem aplikacji jej kod jest tłumaczony do kodu JavaScriptu. Operację tą dokonuje się przeważnie przed wrzuceniem solucji na stronę. Są metody by dokonywać operacji tłumaczenia kodu TS na JS u klienta. Byłby to natomiast ogromny krok wstecz jeśli chodzi o optymalizację kodu. Więc Angular nie został stworzony do aplikacji na serwisy MPA

Z pewnością Angular nie został stworzony do małych aplikacji na stronę internetową. Kodu jest za wiele, ma zbyt skomplikowaną strukturę i co z tym idzie, nie został stworzony do kontrolowania małych fragmentów pojedynczych stron.

Ostatnim pytaniem, które w tej kwestii można zadać, jest, czy jest zalecany lub potrzebny kompilator, który bedzie porcjował odpowienie dane.

W przypadku Angulara odpowiedź brzmi „tak”. Obecność kompilatora jest konieczna. Jest to niewątpliwa wada przy budowie małych aplikacji.

Ponownie, by odpowiedzieć na pytanie, jak framework poprawnie skaluje się wraz ze wzrostem rozmiarów aplikacji, trzeba odnieść się do kilku bardziej szczegółowych kwestii.

W przypadku Angulara odpowiedź na pytanie, czy da się w prosty sposób zarządzać aplikacją podczas jej budowania brzmi „tak”. Jest mnóstwo dobrych praktyk, począwszy od tego, jak pliki powinny się nazywać, aż do kwestii ich przechowywania. Angular CLI ([o którym więcej można przeczytać w tym wpisie](http://wordpress1802290.home.pl/autoinstalator/wordpressplus/blog/budowanie-aplikacji-angular4-przy-angularcli/)) udostępnia oprócz środowiska deweloperskiego zespół takich dobrych praktyk. Podczas tworzenia nowych komponentów, pliki, katalogi, komponenty i inne moduły są nazywane według określonego schematu

Drugą ważną kwestią jest to, czy deweloperzy muszą w dużej mierze polegać na paczkach kodu nie wywodzących się od twórców frameworka oraz czy wystarczająco dużo funkcji i możliwości oferuje framework sam w sobie.

Angular jest samowystarczalny, ma zaimplementowane mnóstwo rozwiązań, które na ogół przy tworzeniu aplikacji wystarczają. Plusem tego jest fakt, że dodatki są aktualizowane wraz z nową wersją frameworka.

Kolejną, niezbędną do poruszenia kwestią, jest możliwość optymalizacji. To, czy łatwo zoptymalizować kod i czy łatwo go sprowadzić do stanu zadowalającego.

Angular posiada bardzo dużo tego typu wzorców. Są one niezbędne do optymalizacji danego programu użytkowego. Zbudowana wersja deweloperska do stanu produkcyjnego bez takich procesów posiada bardzo duży rozmiar.

Ostatnią sprawą jest łatwość zarządzania stanem aplikacji. Bardzo szybko przy rosnących aplikacjach pojawia się ten problem. Ważne wówczas są metody radzenia sobie z tym problemem, to czy jest jakiś wzorzec, biblioteka, czy nie ma nic.

Przy tworzeniu aplikacji w Angularze do dyspozycji są serwisy, które przestają się sprawdzać przy tworzeniu bardzo dużych aplikacji. Wtedy warto sięgnąć po rozwiązanie społeczności o nazwie NgRx, które pozwala zarządzać stanem aplikacji za pomocą klasy Observable.

**Wydajność**

Jeśli chodzi o wydajność to React i Vue mają podobne czasy więc prędkość nie będzie tutaj raczej znaczącym czynnikiem

W React’cie jednak kiedy stan komponentu ulega zmianie to komponent jest renderowany ponownie wraz ze wszystkimi swoimi elementami potomnymi.

W Vue zależności komponentów są automatycznie śledzone podczas renderowania więc system dokładnie wie kiedy komponent rzeczywiście potrzebuje ponownego wyrenderowania podczas zmiany swojego stanu.

Dzięki takiemu rozwiązaniu to Vue przejmuje z rąk programisty kwestie optymalizacji, tak żeby zapewnić ją jak najlepszą a samemu programiście pozwala skupić się na tworzeniu.

**HTML & CSS**

W React wszystko jest po prostu JavaScriptem. Nie tylko HTML wyrażony przez JSX.

Vue bierze klasyczne webowe technologie i buduje na nich.

Funkcja renderująca z JSX ma kilka zalet:

* Możesz wykorzystać pełnie możliwości JS do budowania swojego widoku. Wliczając w to tymczasowe zmienne, kontrolę przepływu danych i bezpośrednie odwoływanie się do elementów z poziomu JS.
* Wsparcie ze strony narzędzi a więc lintowanie, sprawdzanie typów, autouzupełnianie ze strony IDE itd w bardziej zaawansowany sposób niż w przypadku tego jakie dostarcza Vue.

W Vue również jest wsparcie dla funkcji renderujących i JSX ponieważ czasem potrzebujesz takich możliwości. Jednakże domyślnie oferowane są szablony pisane w HTML jako łatwiejsza alternatywa takiego rozwiązania.

Każdy poprawny HTML jest tównież poprawnym szablonem dla Vue i prowadzi to do kilku zalet:

* Dla wielu programistów którzy mieli styczność z HTML, system szablonów jest znacznie bardziej naturalny do czytania i pisania. Takie preferencje mogą wydawać się czymś subiektywnym ale sprawiają, że programista jest bardziej produktywny a to na pewno nie jest już subiektywne.
* Istniejące już aplikację oparte o HTML są znacznie łatwiejsze do migrowania na Vue.
* Sprawia to, że proces uczestniczenia w tworzeniu staje się łatwiejszy dla mniej doświadczonych programistów i projektantów.
* Możesz korzystać z nawet pre procesorów takich jak Pug ( formalnie znany jako Jade ) żeby tworzyć twoje szablony w Vue.

Niektórzy kłócą się nad kwestią że tutaj też trzeba nauczyć się specyficznego syntaksu ale moim zdaniem jest to co najwyżej powierzchowne. Jeśli ktoś korzysta z JSX to wie, że jest to syntaks nałożony na JS. Dla programistów którzy są obeznani z JS na pewno proces nauki będzie szybszy ale nie można powiedzieć, że szybki i super łatwy. W przypadku składni Vue w HTML koszt nauki jest relatywnie niski jeśli ktoś jest zaznajomiony z HTML. Niektóre zadania również mogą wymagać tak na prawdę większej ilości pracy jeśli wykorzystamy czysty JSX czy funkcje renderującą.

**CSS na poziomie komponentu**

Jeśli nie rozbijasz swoich komponentów na wiele plików to CSS w Reactcie jest osadzony w JS. Rodzi to nowy sposób na tworzenie CSS zorientowanego na komponent a proces ten różni się od tradycyjnego wytwarzania. Dodatkowo, pomimo że istnieje wsparcie dla procesu wyodrębnienia CSS do pojedynczego bundla w czasie budowanie to dalej często jest tak, że podczas uruchomienia bundle’a ten jest zaciągany osobnym wywołaniem żeby stylowanie działało prawidłowo. Więc ostatecznie wygląda to tak, że zyskujesz jeśli chodzi o dynamikę styli w oparciu o JS przy ich tworzeniu ale tracisz na wielkości bundla i koszcie w postaci dłuższego czasu wykonania.

Sporą różnicą tutaj jest to, że w Vue domyślnie wspieranym sposobem stylowania jest umieszczenie styli w tagu <style> kiedy tworzymy jedno-plikowe komponenty, Pozwala to na dostęp do pełnego CSS w tym samym pliku gdzie znajduje się reszta kody twojego komponentu.

I ostatecznie stylowanie w Vue w oparciu o wzór całego komponentu w jednym pliku sprawia, że jest to znacznie bardziej elastyczny proces. Przez vue-loader możesz korzystać z dowolnego preprocesora, post procesora i nawet integracji z modułami CSS – wszystko wewnątrz tagu <style>.

**Skalowanie**

Dla wielkich aplikacji zarówno React jak i Vue dostarczają system routingu.

Społeczność która zebrała się wokół Reacta btła bardzo innowacyjna jeśli chodzi o zarządzanie stanem – Redux /  
Flux. Taki wzorzec zarządzania stanem może zostać bardzo łatwo zaaplikowany do Vue, nawet sam Redux bez żadnych konkretnych zmian. W zasadzie w Vue został zaimplementowany właśnie taki wzorzec w postaci Vuex

Inną dość ważną różnicą pomiędzy tymi dodatkowymi bibliotekami dla zarządzania stanem i routingu jest fakt, że w przypadku Vue są one aktualizowane wraz z biblioteką core’ową. React natomiast pozostawia te kwestie społeczności tworząc bardziej zdefragmentowany system. Będąc jednak bardziej popularnym ekosyetem Reacta jest znacznie bardziej rozbudowany niż w Vue.

Vue oferuje generator projektu w CLI co sprawia, że wystartowanie z nowym projektem staje się trywialnie łatwe, uwzględniając przy tym, że masz do wyboru Browserify, Webpacka czy nawet opcje nie wybierania żadnego tego typu systemu. React również ma podobne rozwiązanie w postaci create-react-app które niestety ma pewne ograniczenia:

* nie pozwala na żadne zmiany ustawień podczas tworzenia projektu
* oferuje tylko jeden szablon zakładając, że będzie tworzył Single Page Application podczas gdy Vue dostarcza możliwość wielu zdefiniowanych dla użytkownika szablonów
* nie może generować projektów z przygotowanych przez użytkownika szablonów co jest użyteczne zwłaszcza w bardziej korporacyjnym środowisku gdzie istnieją pewne wcześniej  
  zdefiniowane konwencje

React ma dość stromą krzywą uczenia się. Żeby zacząć musisz coś wiedzieć o JSX i ES2015 z racji, że większość przykładów w React korzysta z klas. Będziesz musiał nauczyć się też czegoś o jakimś systemie budowania z racji, że o ile technicznie rzecz biorąc możesz korzystać z Babel Standalone to kompilowania swoich zmian na żywo i deweloperki to nie jest to absolutnie zalecane podejście w przypadku produkcji.

Vue ma znacznie niższy próg wejścia. Niektórzy twierdzą, że porównywalny z jQuery. Żeby zacząć wystarczy wkleić odpowiedni link i nie trzeba się już niczym przejmować – można zacząć tworzyć.

**Natywne renderowanie**

React Native pozwala Ci na pisanie aplikacji na iOS i Androida które będą się tam natywnie renderowały jakbyś napisał je w Swifcie czy Kotlinie. Przy tym wykorzystywana jest dokładnie taka  
sama składnia i konwencje jak w przypadku Reacta.

W tym temacie zespół odpowiedzialny za Vue.js oficjalnie rozmawia i wspiera projekt Weex. W tej chwili Weex jest w fazie silnego rozwoju i oczywiście nie może jeszcze konkurować z dużo lepiej zaprawionym w bojach, przetestowanym i z większym community, Reactem Native.

#### **ANGULAR.JS**

**Złożoność**

Vue jest znacznie prostsze niż AngularJS, zarówno w kwestii API jak i zaprojektowania. Nauczenie się wystarczającej ilości rzeczy, żeby napisać nietrywialną aplikację zazwyczaj zajmuje mniej niż dzień w przypadku Vue podczas gdy nie można powiedzieć czegoś podobnego o AngularJS

**Elastyczność i modułowość**

AngularJS dosyć mocno określa jak twoja aplikacja powinna wyglądać jeśli chodzi o jej strukturę podczas gdy Vue jest bardziej elastycznym i modułowym rozwiązaniem. Podczas gdy czyni to z niego rozwiązanie lepiej dostosowujące się do większej ilości projektów to jego projektanci postanowili również, że czasem lepiej jest podjąć decyzję za programistę żeby ten mógł skupić się na pracy ale w każdej chwili mógł to zmienić.

To dlatego postawienie projektu z szablonu webpacka zajmuje w Vue kilka chwil ale daje również możliwość późniejszego skonfigurowania sobie takich rzeczy jak ładowanie modułów na bieżąco, lintowanie, wyodrębnianie CSS i wiele więcej.

**Podstawianie danych**

AngularJS wykorzystuje bindowanie w dwie strony pomiędzy zakresami gdy Vue zmusza to jednostronnego przepływu danych pomiędzy komponentami. Sprawia to, że przepływ danych staje się znacznie prostszy do zrozumienia w bardziej skomplikowanych aplikacjach.

**Dyrektywy vs komponenty**

Vue ma również bardziej widoczną granicę pomiędzy dyrektywami a komponentami. Dyrektywy oznaczają jakieś określone manipulacje na DOM a komponenty są natomiast zamkniętym w sobie DOMem z całą swoją funkcjonalnością. W AngularJS dyrektywy robią wszystko a komponenty są specjalną formą dyrektyw.

**Wydajność**

Vue pod względem wydajności bije AngularJS na głowę. Wszystko za sprawą tego, że jest łatwiejsze do optymalizacji i znacznie inteligentniej przeprowadza sprawdzanie i ponowne renderowanie zmienionego DOM. AngularJS jest wolny z racji na fakt posiadania wielu obserwatorów które muszą zostać ponownie wykonane gdy jakieś dane ulegną zmianie.

Angular >2.0

**TypeScript**

Angular najzwyczajniej wymaga TypeScripta, dostarczając wszystkie przykłady i materiały do nauki w TS

Jednakże nie każdy chce korzystać z TS. W wielu przypadkach mniejszych aplikacji, wprowadzanie systemu typów może skutkować lekkim przygnieceniem produktywności. W takich przypadkach lepiej skorzystać z Vue z racji, że Angular bez TS może być dosyć sporym wyzwaniem.

**Wydajność**

Obydwa frameworki są bardzo szybkie z pewnymi, niewielkimi zmianami w tabelkach podczas benchmarków co raczej nie powinno wpływać na osąd.

**Rozmiar**

Obecna wersja Angular z kompliacją AOT i tree-shakingiem może spowodować, że rozmiar znacząco spadnie. Ale większy projekt w Vue jest i tak 2x mniejszy od tego wykonanego w Angularze nawet pomimo faktu wykorzystania AOT i projektu wygenerowanego przy użyciu CLI.

**Elastyczność**

Vue znacznie mniej Ci narzuca niż Angular, oferując szereg wsparcia dla różnych systemów budowania bez narzucania Ci tego jak powinna wyglądać struktura aplikacji.

**Krzywa uczenia się**

Wszystko czego potrzebujesz żeby zacząć z Vue jest znajomość HTML i ES5 chociaż oczywiście w tych czasach powinna to już być ES2015+. Z tak podstawowymi zdolnościami możesz zbudować naprawdę bardziej zaawansowaną aplikację w mniej niż dzień czytając oficjalny poradnik / dokumentację.

Krzywa ta w przypadku Angulara wygląda dużo ostrzej. API jest ogromne i zanim użytkownik rzeczywiście się z nim w jakiś sposób oswoi mija trochę czasu. Oczywiście jest też trochę więcej konceptów i praktyk które programista musi poznać, żeby rzeczywiście być produktywnym. Oczywiście taka złożoność Angulara wynika z faktu, że został zaprojektowany z myślą o tworzeniu wielkich aplikacji ale sprawia to, że jest trudniejszy dla niezbyt doświadczonych programistów którzy chcą zacząć z nim swoją przygodę.

**React więc daje tobie:**

* szybkość znalezienia pracy
* zwykle większy zarobek
* korzystanie z największej społeczności
* niewysoki próg wejścia w środowisko
* przyjemność z pisania kodów.

**Angular jest najlepszy w:**

* nauce najlepszej jakości pisania kodu
* wymaga dużej wiedzy, ale daje spory rozwój jako programista - developer
* tworzeniu dużych projektów
* najbardziej logiczna struktura Frameworka
* duża społeczność kodujących.

**Vue.js jest:**

* najszybszy do nauki
* ma najniższy próg wejścia
* świetny do tworzenia mniejszych projektów (idealny do tworzenia "lekkich" aplikacji webowych)
* alternatywą dla Angulara, kiedy potrzeba tylko podstawowych funkcjonalności
* przyjemny w pisaniu kodów.