Finale Evaluierung

Einleitung

Um am Ende der Prototyping Phase eine sinnvolle Entscheidung zu treffen, brauchen wir gewisse Evaluations Kriterien, die wir auf alle 3 bzw. 4 Prototypen anwenden können. Im folgenden Abschnitt werden zunächst diese Kriterien beschrieben und im nachfolgenden ist eine Tabelle zu finden, wo diese Kriterien auf unsere Prototypen angewandt wurde.

Kriterien

1. **Lernaufwand:** Die Codebase des zukünftigen Projektes sollte für andere Angestellte gut leserlich und verständlich sein. Dabei geht es darum, dass ein anderer erfahrener Mitarbeiter, welcher nichts mit dem Projekt zu tun hat, mit relativer Leichtigkeit den Code versteht. Natürlich gibt es bei den unterschiedlichen Prototypen jeweils eine andere Codesyntax, allerdings wird hier nur die Lesbarkeit und die Einfachheit evaluiert. Der Lernaufwand um mit dem Framework umgehen zu können, sollte natürlich möglichst gering sein.
2. **Dokumentation:** Es sollte eine ausführliche deutsche oder mindestens englische Dokumentation online zur Verfügung stehen.
3. **Community:** Es muss eine stabile Verbreitung und gute Recherchemöglichkeit im Internet vorhanden sein.
4. **Performance:** Im grundlegenden soll die alte Swing-GUI durch eine neue, adaptive, schnelle und responsive UI ersetzt werden. Dabei sollte zum Beispiel eine Pageloading time von unter 2 Sekunden angestrebt werden. Des Weiteren spielt auch die Größe der Webseite (in MB) bei der Performance eine Rolle. Eine gute und schnelle UI sollte nicht mehr als 1 MB groß sein (am besten bei 500kB gzipped). Eine gute Hilfe für die Evaluierung der Performance bietet Google mit seinem Profiler Test.
5. **Binding:** Das Verbinden der Daten mit den GUI-Komponenten sollte vom Framework bereits geliefert werden. Änderungen im Daten-Model werden automatisch in der Oberfläche sichtbar und umgekehrt Änderungen in den Eingabe-Komponenten werden sofort im JSON-Model aktualisiert. Ein bidirektionales automatisches Binding ist einem Unidirektionalen vorzuziehen.
6. **Komponenten:** Es sollte eine Umfangreiche Komponenten-Bibliothek die möglichst auf das Framework optimiert ist zur Verfügung stehen. Hierbei ist insbesondere auch auf die Anzeigeperformance bei großen Datenmengen (Tabelle > 1000 Zeilen, Editoren mit > 250 Eingabeelementen). Tabellen müssen scrollbar, filterbar, sortierbar, Spaltenbreite einstellbar sein, spezielle Renderer verwenden können und möglichst Druckbar sein.
7. **Flexibility:** Im Laufe der Entwicklung kann es dazu kommen, dass wir möglicherweise third-party Plugins und Libraries einbinden möchten. Das zukünftige Projekt muss in der Lage sein, diese Extensions leicht und unkompliziert einbinden zu können.
8. **Build / Debugging Tools:** Gute Build und Debugging Tools führen am Ende zu einer schnelleren und fehlerfreieren Auslieferung der Software. Manager wie npm machen uns da die Arbeit leichter und sind dementsprechend bevorzugt. Des Weiteren sollte der Debugging-Prozess unkompliziert mit der IDE durchführbar sein. Bei Webanwendungen ist dieser nämlich häufig im Webbrowser und dadurch umständlicher.
9. **Testing Tools:** Das zukünftige Projekt soll natürlich auch testbar sein. Gute Testing frameworks, welche wir von Java gewöhnt sind, sind erstrebenswert.

Evaluierungstabelle

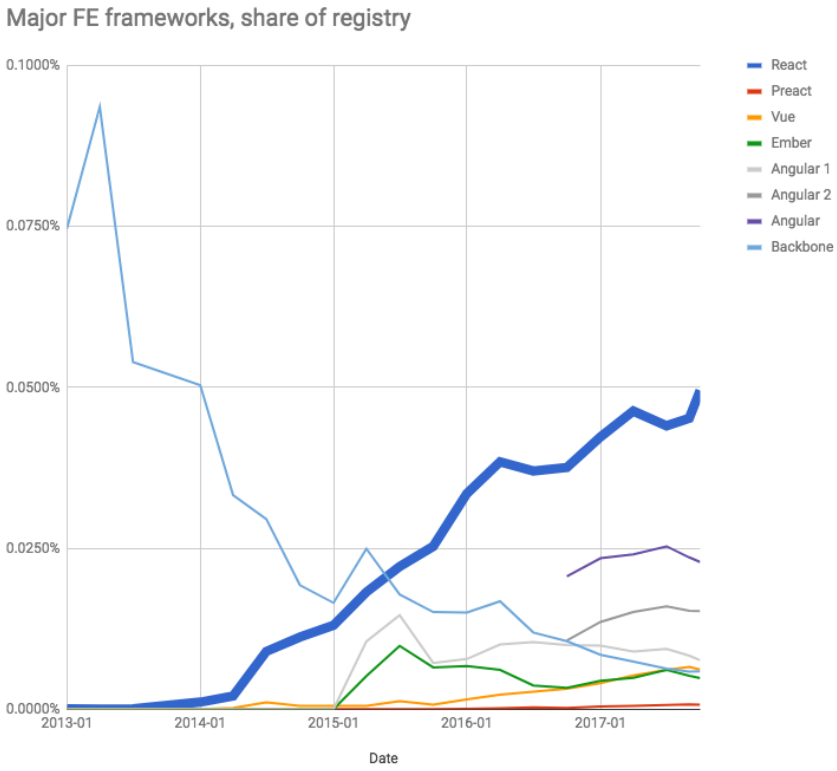
Kriterien	Vue	React	Angular	Bootstrap
Learnaufwand	Sehr gute Readability & learning curve ist relativ klein. Templates machen die Arbeit hier sehr angenehm. Lernaufwand sehr gering	Mittelmäßige Readability. JSX & unnötig kompliziertes conditional rendering. Lernaufwand gering. Mehr dazu hier.	Mäßige Readability. Vue und React haben hier den Vorteil, das template und logik alles in einer Datei gespeichert werden. Bei Angular benötigt man für die Darstellung einer Seite 4 unterschiedliche Dateien. Von Haus aus Typescript.	
Dokumentation	Offizielle Dokumentation nur in englisch. Community Dokumente auch in Deutsch verfügbar.	Offzielle Dokumentation in deutsch und englisch.	Offizielle Dokumentation ist in englisch, Japanisch, Chinesisch, Koreanisch verfügbar. Community Dokumente auch in Deutsch verfügbar.	
Community	138.000 Sterne auf GitHub. 144.000 Webseiten die im Moment Vue benutzen. Wachsende Online Community. Sehr kleiner Job-Market. Keine Unterstützung von Großunternehmen. 274 Contributer.	129.000 Sterne auf GitHub. 250.000 Webseiten die im Moment React benutzen. Wachsende Online Community. Extrem großer Job-Market. Unterstützung von Großunternehmen (Facebook). 1.296 Contributer.	47.000 Sterne auf GitHub. 32.000 Webseiten die im Moment Angular benutzten. Stagnierte Online Community. Extrem großer Job-Market. Unterstützung von Großunternehmen (Google). 924 Contributer.	
Performance	Sehr kleine Bundle-size und somit gute Performance. Eigene Komponentenperformance von der eingebundenen Komponente abhängig (zB Element UI). Gute Performance auch ohne Optimierungs Guidelines möglich. Code-Splitting durch Routing möglich.	Performance gut, solange man Optimierungs Guidelines beachtet. Code-splitting durch Routing & Lazy-Loading möglich. Mehr dazu hier.	Performance sehr gut. Mit Angular 7 ist jetzt die schnellste Version von Angular. Ist auch von ein gebundenen Komponente abhängig (zB Material). Gute Performance auch ohne Optimierungs Guidelines möglich. Code-Splitting durch Routing möglich.	
Binding	Two-way-binding. Einfach und verständlich.	One-way-binding. Mit callbacks etc. muss two-way selber realiziert werden: hier.	Two-way-binding. Einfach und verständlich.	
Komponenten	Kuratierte UI Frameworks vorhanden. Bieten alle umfangreiche Funktionalitäten. Spezielle Komponenten für große Tabellen ebenfalls vorhanden und gut dokumentiert.	Sehr viele freie UI Komponenten verfügbar. Spezialisierte Komponenten, zum Beispiel für Tabellen durch react-virtualized gegeben.	Komponenten können einfach eingebunden werden. Zahlreiche Komponenten mit umfangreichen Funktionalitäten vorhanden. Spezialisierte Komponenten ebenfalls vorhanden.	

Flexibility	Sehr gute Flexibility. Externe Plugins können einfach mit npm installiert werden und mit einem einfachen import genutzt werden.	Sehr gute Flexibility. Externe Plugins können einfach mit npm installiert werden und mit einem einfachen import genutzt werden.	Sehr gute Flexibility. Externe Plugins können einfach mit npm installiert werden und mit einem einfachen import genutzt werden.	
Build / Debugging Tools	Skripte zum starten, testen & bauen durch npm gegeben. IDE Debugging durch zum Beispiel WebStorm.	Skripte zum starten, testen & bauen durch npm gegeben. IDE Debugging zum Beispiel in Visual Studio Code mit Hilfe von Plugins möglich.	Skripte zum starten, testen & bauen durch Angular CLI (ng). IDE Debugging durch zum Beispiel WebStorm.	
Testing Tools	Gute Testing Frameworks wie Vue Test Utils, Mocha und Jest Matchers sind hier vorhanden. Bieten Unit Testing.	Gute Testing Frameworks wie Enzyme, Mocha und Jest Matchers sind hier vorhanden. Bieten Unit Testing.	Gute default Testing Frameworks, wie Jasmine Javascript Test Framework und der Karma Test Runner. Bieten Unit Testing.	

Tech Trends

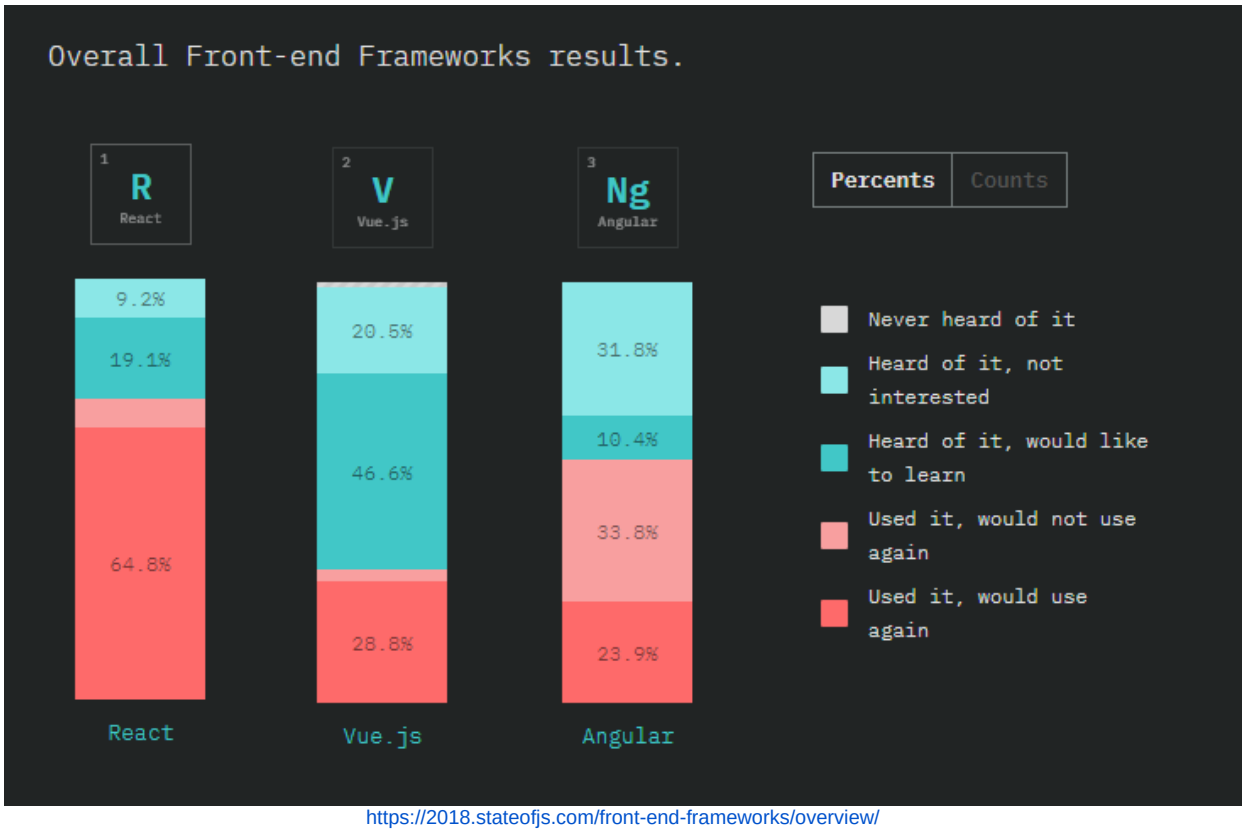
Im folgenden befinden sich Grafiken zum Job-Demand, Developer-Usage und weiteren Statistiken, welche eventuelle die Entscheidung beeinflussen.

Major Front-end frameworks



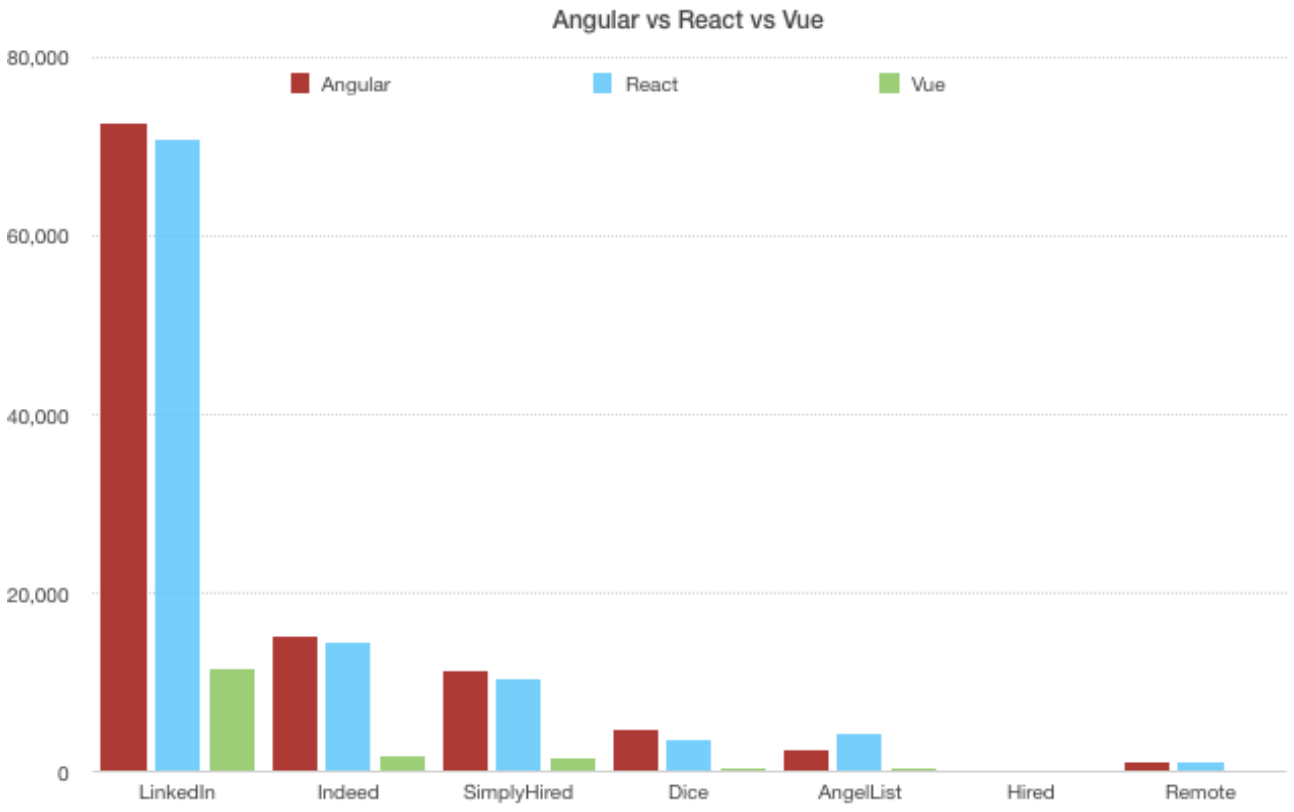
https://blog.npmjs.org/

stateofJS Umfrage

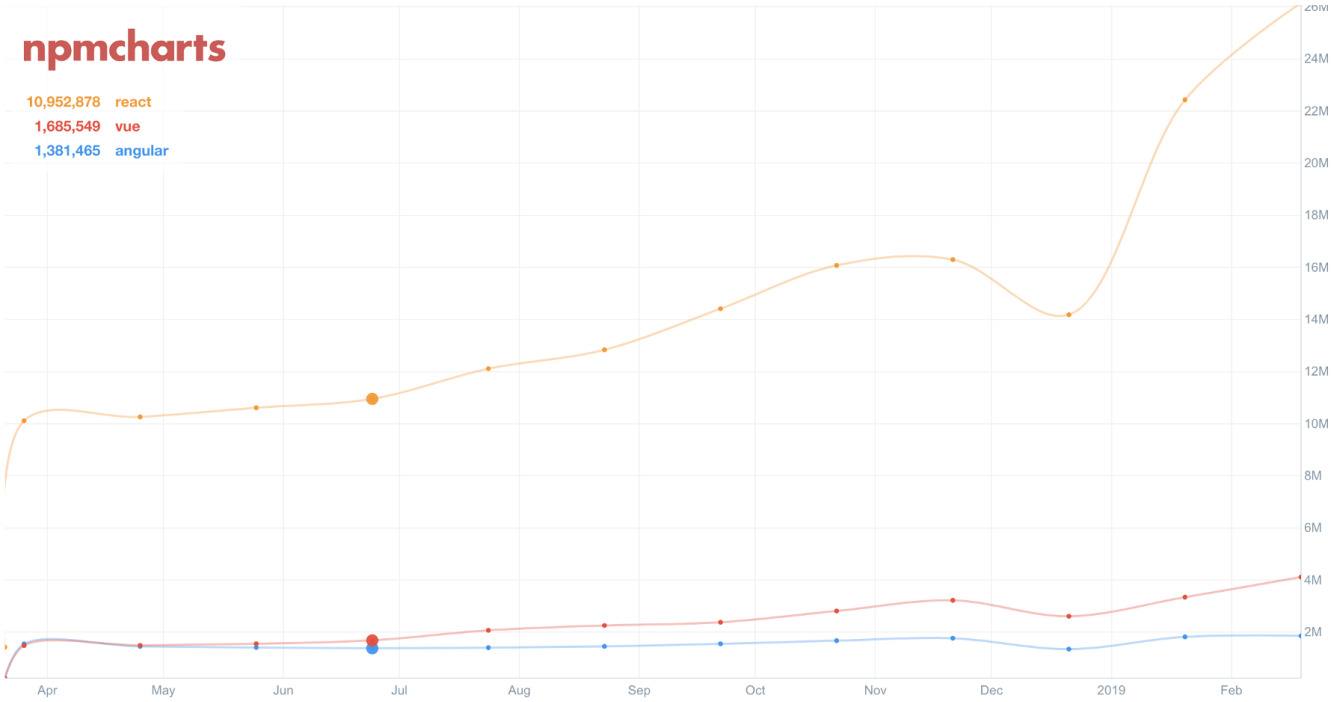


https://2018.stateofjs.com/front-end-frameworks/overview/

Job-Market

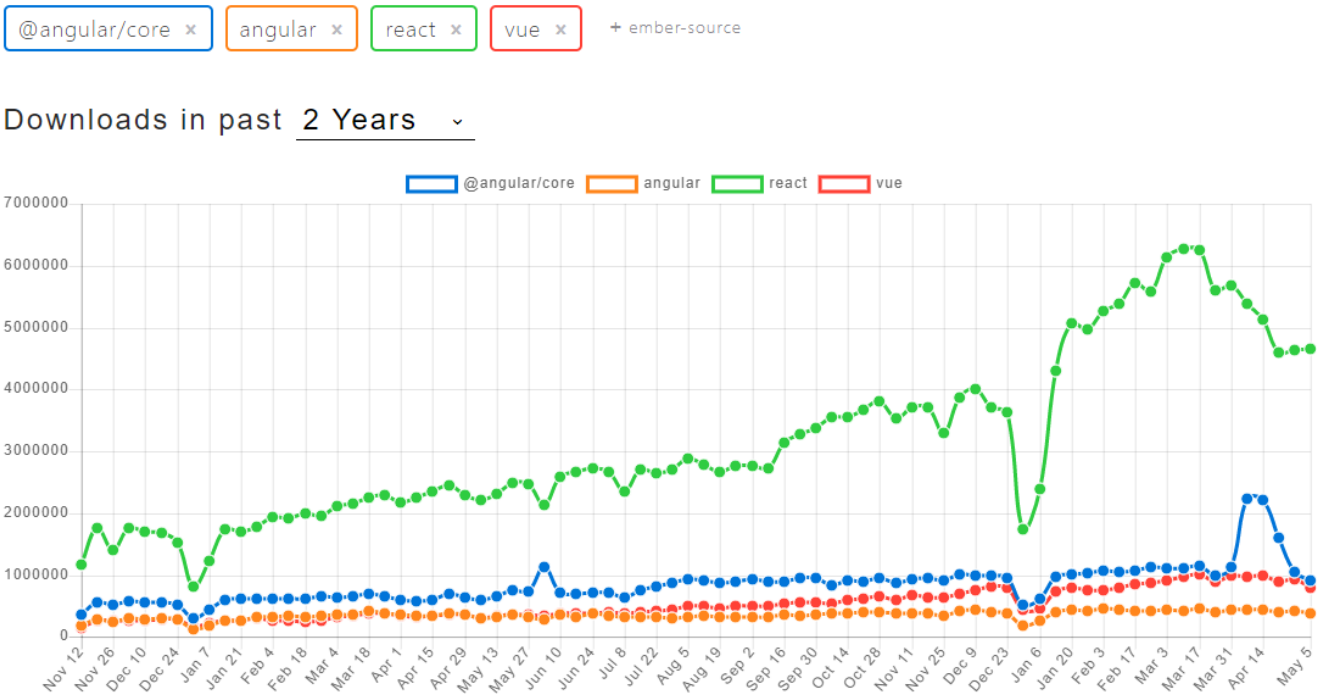


npm Charts



https://npmcharts.com/

npm Trends



https://www.npmtrends.com/

Google Trends

https://trends.google.com/trends/

Fazit

Die drei großen Frameworks bringen die gewünschten Core-Funktionalitäten mit, unterstützen Type-Script und JUnit-Testing. Dennoch haben sich in den Implementierungen der Prototypen schon Unterschiede gezeigt, die wir für entscheidend halten.

- Tendenz Vue aus Entwicklersicht am besten, jedoch noch nicht ausgereift. Third-party Libraries springen erst jetzt auf noch nicht ausgereift. Vue 3 angekündigt Mögliche Änderungen?
- Tendenz React Ausgereift, große Community & Verbreitung. Unterstützung großer Third-party Libraries (**PrimeReact**). Stabiler Trend.
- Tendenz Angular Hohe Komplexität & großer Lernaufwand. Sehr viele Updatecyclen (alle 6 Monate) Ähnlich Oracle Java. In vielen Trends abfallend (siehe Tech Trends). Wird in der Community schon als veraltet bezeichnet.
- Tendenz Bootstrap Keine echte Alternative, da nur Front-end Styling Aufgaben gelöst werden (Kein Binding). Starke Abhängigkeit zum .NET Framework bei Indiva. Müsste man mit einer Binding-Bibliothek wie React oder Angular kombinieren.

Wir plädieren somit für die Kombination React + PrimeReact.