Project 6: Professional Frontend JavaScript Framework Evaluation

Eric Groom

1. Introduction
Over the last few years, JavaScript front-end frameworks have seen a massive jump in popularity. In most cases, they are used to build single-page web applications but can also be used for simple web pages and cross-platform mobile and desktop applications. They are most useful for data-driven applications with frequent page updates, as this code can be tedious to write and maintain without using a framework or library of some kind. This evaluation includes the three currently most popular frameworks: React, Vue and Angular. Each of them have their strengths and weaknesses which will be explored in this paper via analysis and the evaluation itself. Given more time, I would have also liked to include Ember and Aurelia.

Note: There is code in each of these frameworks I have written to support this analysis which can be found at https://github.com/ericgroom/csc876-p6.

2. Building of Criteria

2.1 Stakeholder Definition
A critical aspect of LSP evaluation is the definition of the stakeholder. The entire evaluation and creation of criteria is based on the priorities of said stakeholder. For this evaluation, the stakeholder will be a hypothetical CEO of a company where their software is their main product. The stakeholder would like to evaluate which framework they should use for future projects. Some of their priorities include: longevity of the framework, learning curve, community support and popularity.

2.2 Suitability Attribute Tree

- **1 Javascript_Frameworks**
  - **11 Community Support**
    - **111 Weekly NPM downloads**
    - **112 Github stars**
  - **12 Maintainership**
    - **121 Frequency of Updates**
    - **122 Number of Contributors**
    - **123 Age (years)**
    - **124 Corporate backing**
    - **125 Open source**
  - **13 Performance**
    - **131 Bundle size**
      - **1311 Hello world application (bundle)**
      - **1312 Todo application (bundle)**
    - **132 Rendering speed test (ms)**
  - **14 Developer Experience**
    - **141 Typescript support**
    - **142 Sensible webpack defaults**
      - **1421 Hot reloading**
      - **1422 Source Maps**
      - **1423 Minification and uglifying**
    - **143 Usable without build tools**
    - **144 Server side rendering framework available**
    - **145 Testability**
      - **1451 included test runner and assertion library**
      - **1452 Difficulty to mock library functions**
      - **1453 Ability to test asynchronous code**
    - **146 Customizability of build tools**
    - **147 Browser extension**
      - **1471 First-party extension exists**
      - **1472 Ability to inspect application state**
      - **1473 Ability to edit application state**
      - **1474 Ability to run arbitrary code on state**
    - **148 Code base size**
      - **1481 Number of source files**
        - **14811 Hello world application (files)**
        - **14812 Todo application vs hello world (files)**
      - **1482 Lines of code**
        - **14821 Hello world application (LOC)**
        - **14822 Todo application (LOC)**
  - **15 Difficulty to learn**
    - **151 As someone new to web development**
    - **152 As someone with a high mastery of modern javascript (es6)**
    - **153 As someone familiar with OOP and MVC**

2.3 Evaluation Criteria and Aggregators

Elementary criteria for project: Javascript_Frameworks

111		**Weekly NPM downloads**
Value	%	One way to measure the popularity of a framework is the number of downloads it has on Node Package Manager (NPM). This number is often a bit inflated due to being incremented whenever adding a new team member, deploying to production, etc.
0	0	
100000	70	
500000	95	
1000000	100	

112		**Github stars**
Value	%	Github is a git-repository host where the majority of open source projects host their projects. Github stars can be an indicator of popularity, but it can be skewed since not everyone who uses the framework stars their Github page.
0	0	
20000	80	
40000	90	
100000	100	

121		**Frequency of Updates**
Value	%	A framework should be frequently updated to maintain support and add new features. This input is measured by number of releases in 2018 since this is being recorded right at the end of 2018.
0	0	
10	60	
20	90	
30	100	

122		**Number of Contributors**
Value	%	Most open source projects allow outside developers to submit patches or pull requests to make additions or changes to the project. A high number of contributors means that a project likely doesn't depend on one sole contributor, there is a friendly community that welcomes additions, and that people want to improve the framework.
0	0	
1	10	
5	20	
100	70	
200	90	
300	100	

123		**Age (years)**
Value	%	It would be unwise to integrate a framework that is too young into the stakeholder's business; it should be adequately mature to have garnered consistent development and testing.
0	0	
1	40	
2	60	
3	95	
4	100	

124		**Corporate backing**
Value	%	Many open source frameworks are created and developed by a corporate company such as Facebook creating React. While this isn't strictly necessary, there is a confidence boost that can be obtained from knowing that a viable company develops tools essential to your business instead of just a few casual maintainers.
0	50	
1	100	

125		**Open source**
Value	%	A framework should be open source. Open source software allows for testing across a wide gamut of use cases and usually results in a better product. The only case where a non-open source framework should be considered is if it is developed in-house at the stakeholder's company. This however often leads to a waste of development effort unless it is strictly necessary.
0	40	
1	100	

1311		**Hello world application (bundle)**
Value	%	Size of bundle generated by webpack for basic hello world in each framework. This is to give insight into the base-size one can expect when using a framework. Values are in KB.
0	100	
10	95	
30	90	
50	80	
100	60	
500	0	

1312		Todo application (bundle)
Value	%	Size of bundle generated by webpack for basic todo application in each framework. This is to give insight into the how one can expect the bundle size of an application to increase as you add functionality. Values are in KB
0	100	
100	90	
200	70	
600	0	

132		Rendering speed test (ms)
Value	%	For this evaluation I created a short rendering speed test (viewable on my GitHub listed in project document). This test generates a large number of random numbers and updates the state one value at a time, in order to test the frameworks ability to batch updates. In order to test singular updates, there is also a function to sort the list of numbers generated and then update the displayed table, this is to test the framework's ability to track updates in a list. Results are an average of three runs in milliseconds.
0	100	
1000	100	
2000	70	
5000	0	

141		Typescript support
Value	%	Typescript is becoming more popular, and a framework should support Typescript. There are four possible outcomes: framework doesn't support Typescript or not well (0), framework does support typescript but not without additional configuration (1), framework supports Typescript but not enabled by default (3), framework's default option is to use Typescript.
0	20	
1	60	
2	90	
3	100	

1421		Hot reloading
Value	%	Hot reloading helps improve development productivity and isn't used in production, thus it should be included in almost every case.
0	0	
1	100	

1422		Source Maps
Value	%	The use of a build system means that source code mapping is lost after being built for production. This can make debugging very difficult as line numbers and file names will be obscured. A framework should enable source maps in production in order to make for easier debugging of production applications.
0	0	
1	100	

1423		Minification and uglifying
Value	%	Without minification and uglifying code, source code files can be much larger than needed, especially if the developers add external dependencies. Thus a framework should enable this feature in production to reduce bundle size and improve load times.
0	0	
1	100	

143		Usable without build tools
Value	%	Some frameworks can be used by simply including a script tag in your existing html. This means that the framework can be slowly integrated into older code instead of taking over the entire codebase. Additionally, if build tools are too much complexity for a project, they can be avoided.
0	50	
1	100	

144		Server side rendering framework available
Value	%	Server side rendering is becoming more important in JavaScript application development due to the realization that SEO is extremely difficult in a client-side rendered application and also yields the ability to cache pages. This can be difficult to implement and complexity can grow as your code does, thus a mature server-side rendering framework based on the framework in consideration should exist.
0	0	
1	100	

1451		included test runner and assertion library
Value	%	A framework should include a unit testing environment with proper configuration.
0	0	
1	100	

1452		Difficulty to mock library functions
Value	%	One problem that often arises with unit testing is the mocking of library functions, a framework should either provide pre-made mocks for library functions or allow for easy mocking.
0	0	
100	100	

0	0	asynchronous call has finished. A framework's code structure or testing utilities should allow for easy testing for asynchronous code since it is a common use-case.
100	100	

146		Customizability of build tools
Value	%	Occasionally, frameworks will abstract away build tools to reduce what the developer has to consider. However, occasionally, it will be necessary for the developer to customize the application's build settings. The framework should provide an easy way to integrate custom build configuration.
0	0	
100	100	

1471		First-party extension exists
Value	%	A framework should have an official browser extension as they can yield a big boost to development productivity, and the maintainers of a framework can better maintain the extension than a third-party
0	50	
1	100	

1472		Ability to inspect application state
Value	%	A framework's browser extension should at a minimum allow the developer to inspect the state of the application in it's various components or divisions.
0	0	
1	100	

1473		Ability to edit application state
Value	%	In addition to being able to view an application's state, being able to edit an application's state directly and see changes made to the view can help immensely in debugging.
0	0	
1	100	

1474		Ability to run arbitrary code on state
Value	%	Often when writing new functions, it is helpful to see how the function would affect application state. Thus it is extremely useful to be able to run any JavaScript code in order to update application state.
0	0	
1	100	

14811		Hello world application (files)
Value	%	Number of source files in an initial project. This metric can be useful to determine the amount of mental overhead needed when understanding a new project.
0	50	
1	100	
5	95	
10	85	
20	70	
50	0	

14812		Todo application vs hello world (files)
Value	%	Number of files changed in order to implement a basic todo application from the framework's base hello world. This is a useful metric to determine how much logic is contained per file and how much overhead there is to adding new features.
0	0	
1	100	
5	100	
10	90	
20	70	
50	0	

14821		Hello world application (LOC)
Value	%	Number of lines of code in key files (excluding configuration) in framework's initial hello world. This is to give an idea of how much logic is required for a basic page.
0	0	
50	100	

14822		Todo application (LOC)
Value	%	Number of lines needed to implement a basic todo application in the framework. This is a useful metric to determine how much the code grows as features grow.
0	0	
50	100	
100	90	
200	85	
500	0	

151		As someone new to web development
Value	%	A framework should abstract away some of the more tedious aspects of JavaScript development to welcome newer developers. This is important to the stakeholder as it increases the candidate pool for developers.
0	0	
100	100	

152		As someone with a high mastery of modern javascript (es6)
Value	%	A framework should be easy to learn for developers that are already highly familiar with JavaScript. One reason that this wouldn't be true is if the framework uses a lot of "magic" and breaks conventions. This is important for the longevity of the framework and so that the JavaScript community continues to use the framework instead of moving to other more intuitive frameworks.
0	0	
100	100	

153		As someone familiar with OOP and MVC
Value	%	OOP and MVC are popular development patterns and a framework should try apply them where possible to make it easier to learn. These developers need not necessarily be familiar with web development, especially if they are just writing supporting code and not code interacting with the DOM.
0	0	
100	100	

2.4 Aggregator Structure

1	HC	**Javascript_Frameworks**
11	25%	Maintainership is most important to the stakeholder since a framework is usually the backbone of a codebase and they would like their products to last for many years. Performance is the next most important since users can be impatient and want a fast-loading and rendering application. Community support is also very important to the stakeholder as they would like to be able to easily hire new developers who have experience with the framework. Lastly, while still important, the least important aspects are Developer Experience and Difficulty to learn. None of these aspects can be absent in a framework while it still remains sufficient which is why a Hard Conjunctive aggregator is used.
12	30%	
13	25%	
14	10%	
15	10%	

11	HC	**Community Support**
111	60%	A framework should have both a high number of downloads and a high number of GitHub stars. It would be unusual for these values to not be strongly correlated.
112	40%	

12	SD-	**Maintainership**
121	30%	Maintainership is an important aspect to choosing a framework. They often form the backbone for an application and the stakeholder would like their product to last for many years. All of the inputs are only indicators of maintainership and thus a hard aggregator should not be used as the absence of one doesn't indicate a total failure.
122	30%	
123	20%	
124	10%	
125	10%	

13	HC-	**Performance**
131	80%	In real world scenarios, often bundle size is the more limiting factor in display time due to network speeds. Rendering speed is important, but usually less important than overall bundle size
132	20%	

14	SD-	**Developer Experience**
141	5%	Developer experience is important when choosing a framework in order to maintain productivity and developer happiness. In the end, these are not essential to the end products functionality and thus no single one of these criterion is absolutely essential.
142	15%	
143	5%	
144	10%	
145	20%	
146	25%	
147	10%	
148	10%	

15	SD-	**Difficulty to learn**
151	15%	Difficulty to learn is important when choosing a framework as it can make it difficult to transfer developers between projects, reduce interest in the framework, and reduce the hiring pool. While a framework catering to newer web-developers is a nice goal, it is more important that it focuses on being accessible to existing developers.
152	65%	
153	20%	

131	A
1311	30%
1312	70%

Bundle size

Bundle size is very important for web applications. A larger bundle size results in more bytes to send to a client, a larger parse time and longer time to first paint. Average bundle size is hard to measure, since it will vary based on your applications needs: Some frameworks provide needed functionality out-of-the-box, while others will require adding extra dependencies and thus increasing bundle size.

142	HC
1421	20%
1422	40%
1423	40%

Sensible webpack defaults

This is not an extensive list of nice-to-have features but essential features. So, all of these features should be present in a framework's webpack configuration or alternative.

145	HC
1451	20%
1452	50%
1453	30%

Testability

Testability is an important metric when choosing a framework since it is often necessary for writing stable code.

147	SD+
1471	20%
1472	50%
1473	20%
1474	10%

Browser extension

A browser extension can help developers work faster and more easily. Not all of the below features are necessary, they are merely niceties.

148	HC-
1481	35%
1482	65%

Code base size

Codebase size is important as an increase in codebase size usually leads to a higher bundle size and a longer amount of time needed to onboard new developers. Number of source code files helps improve organization, but usually doesn't affect logic which makes number of lines of code more important, however neither should be unsatisfied.

1481	A
14811	70%
14812	30%

Number of source files

Hello world's can vary between frameworks, so the todo application should have a higher consideration.

1482	A
14821	70%
14822	30%

Lines of code

Hello world's can vary between frameworks, so the todo application should have a higher consideration.

3. Evaluation

3.1 Competitors

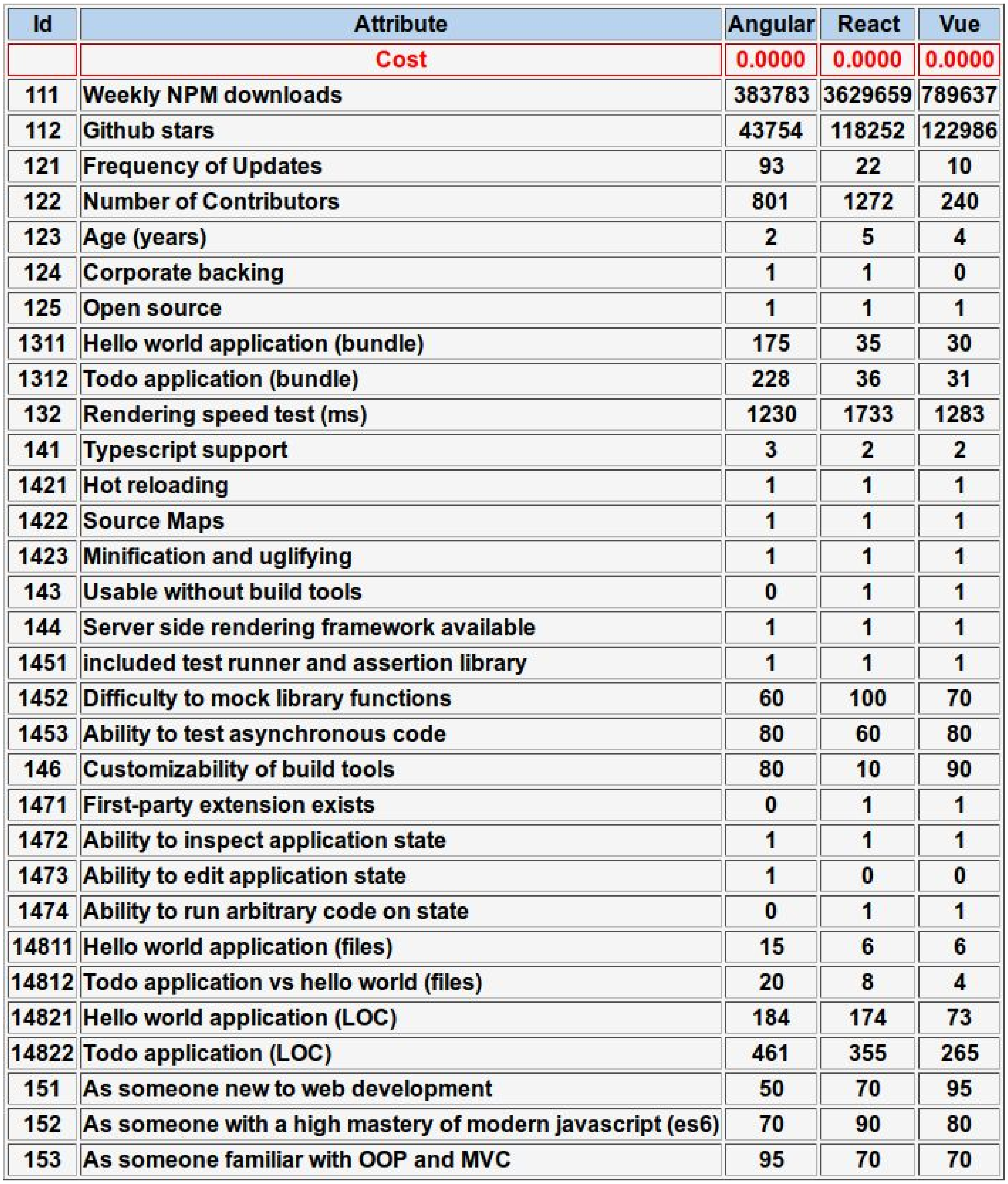
The subjects of this evaluation were the three most popular frameworks at the time of writing: React, Vue and Angular. The exact version numbers evaluated are: React: 16.7.0, Vue: 2.5.17, and Angular 7.1.0. The status of each framework can and will change in the future, but these are my professional opinions at this time.

React is currently the most popular of the frameworks. It was created by and is actively developed by Facebook. Some of the key features of React are that it features a intuitive api for those familiar with modern JavaScript (ES6), it's virtual DOM for improved rendering performance, and the fact that it is the most popular means it gets a lot of community contributors and development of new ideas to improve the framework. React did have a licensing issue about one year ago, and although that has since been resolved, this is a concern to those who are choosing React for long-term projects.

Vue currently has the least market share of the three frameworks evaluated but has a strong community behind it. The framework is extremely easy for those who aren't as familiar with web development and includes some utilities for common tasks such as binding view state to application state that other frameworks don't have by default, or aren't quite as simple. Vue is mainly developed by Evan You and has recently been gaining popularity, especially in China, due to it being simpler than React for many common tasks.

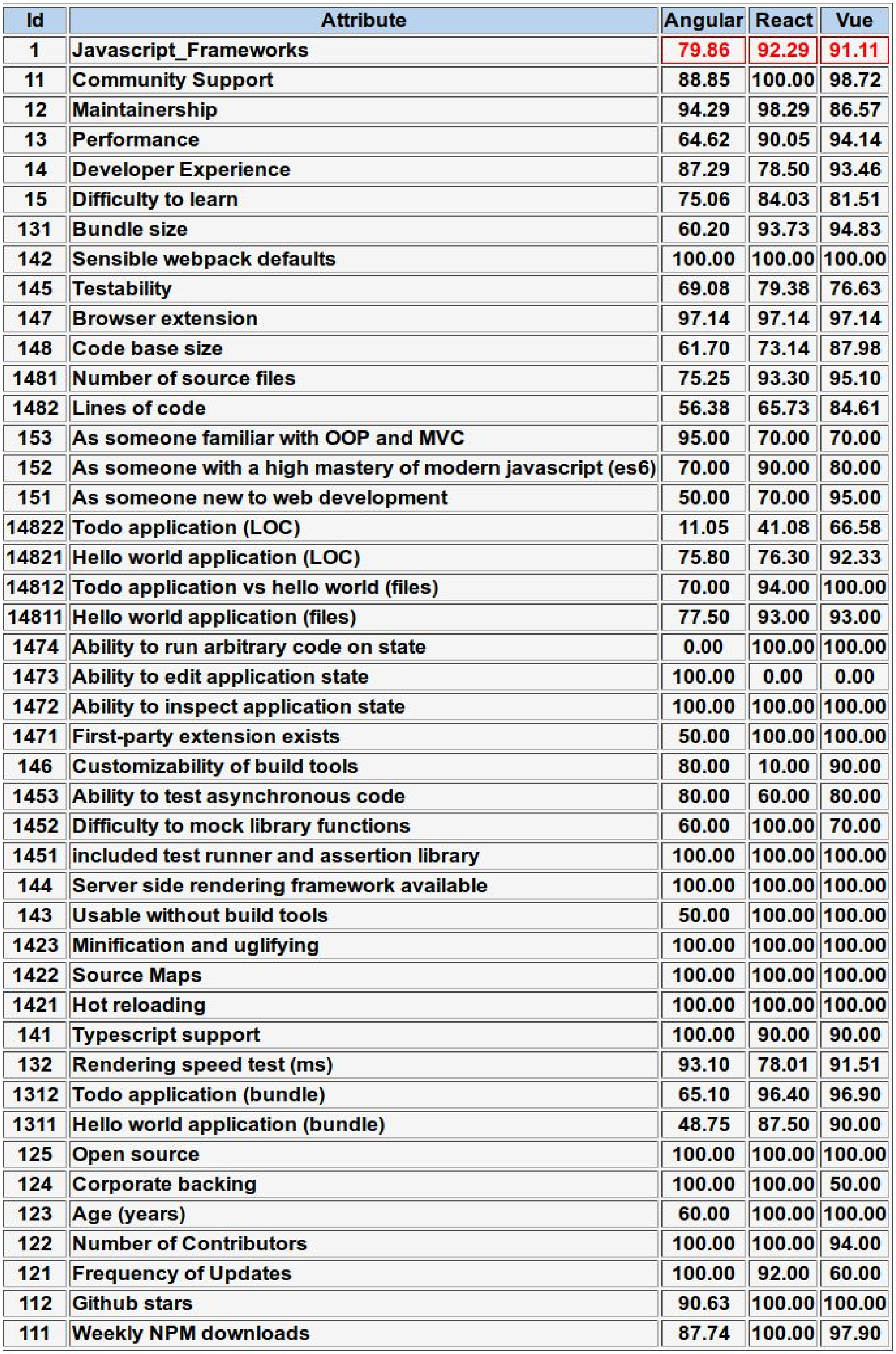
Angular is mainly used in enterprise applications. Currently, it is the least popular of the three frameworks evaluated as far as community support goes; although, there may still be more jobs available than Vue. Angular makes heavy use of Typescript, Dependency Injection, and Reactive programming, which makes it feel very familiar to C# developers and more traditional OOP languages. Angular used to be the most-used framework, but lost the majority of its popularity when they changed from version 1 to 2, which dramatically changed the framework. Although the maintainers have most likely learned from their mistakes, this is something the stakeholder should keep in mind since longevity is one of their primary concerns.

3.2 Attribute Values



Id	Attribute	Angular	React	Vue
	Cost	0.0000	0.0000	0.0000
111	Weekly NPM downloads	383783	3629659	789637
112	Github stars	43754	118252	122986
121	Frequency of Updates	93	22	10
122	Number of Contributors	801	1272	240
123	Age (years)	2	5	4
124	Corporate backing	1	1	0
125	Open source	1	1	1
1311	Hello world application (bundle)	175	35	30
1312	Todo application (bundle)	228	36	31
132	Rendering speed test (ms)	1230	1733	1283
141	Typescript support	3	2	2
1421	Hot reloading	1	1	1
1422	Source Maps	1	1	1
1423	Minification and uglifying	1	1	1
143	Usable without build tools	0	1	1
144	Server side rendering framework available	1	1	1
1451	included test runner and assertion library	1	1	1
1452	Difficulty to mock library functions	60	100	70
1453	Ability to test asynchronous code	80	60	80
146	Customizability of build tools	80	10	90
1471	First-party extension exists	0	1	1
1472	Ability to inspect application state	1	1	1
1473	Ability to edit application state	1	0	0
1474	Ability to run arbitrary code on state	0	1	1
14811	Hello world application (files)	15	6	6
14812	Todo application vs hello world (files)	20	8	4
14821	Hello world application (LOC)	184	174	73
14822	Todo application (LOC)	461	355	265
151	As someone new to web development	50	70	95
152	As someone with a high mastery of modern javascript (es6)	70	90	80
153	As someone familiar with OOP and MVC	95	70	70

3.3 Evaluation Results



Id	Attribute	Angular	React	Vue
1	Javascript_Frameworks	79.86	92.29	91.11
11	Community Support	88.85	100.00	98.72
12	Maintainership	94.29	98.29	86.57
13	Performance	64.62	90.05	94.14
14	Developer Experience	87.29	78.50	93.46
15	Difficulty to learn	75.06	84.03	81.51
131	Bundle size	60.20	93.73	94.83
142	Sensible webpack defaults	100.00	100.00	100.00
145	Testability	69.08	79.38	76.63
147	Browser extension	97.14	97.14	97.14
148	Code base size	61.70	73.14	87.98
1481	Number of source files	75.25	93.30	95.10
1482	Lines of code	56.38	65.73	84.61
153	As someone familiar with OOP and MVC	95.00	70.00	70.00
152	As someone with a high mastery of modern javascript (es6)	70.00	90.00	80.00
151	As someone new to web development	50.00	70.00	95.00
14822	Todo application (LOC)	11.05	41.08	66.58
14821	Hello world application (LOC)	75.80	76.30	92.33
14812	Todo application vs hello world (files)	70.00	94.00	100.00
14811	Hello world application (files)	77.50	93.00	93.00
1474	Ability to run arbitrary code on state	0.00	100.00	100.00
1473	Ability to edit application state	100.00	0.00	0.00
1472	Ability to inspect application state	100.00	100.00	100.00
1471	First-party extension exists	50.00	100.00	100.00
146	Customizability of build tools	80.00	10.00	90.00
1453	Ability to test asynchronous code	80.00	60.00	80.00
1452	Difficulty to mock library functions	60.00	100.00	70.00
1451	included test runner and assertion library	100.00	100.00	100.00
144	Server side rendering framework available	100.00	100.00	100.00
143	Usable without build tools	50.00	100.00	100.00
1423	Minification and uglifying	100.00	100.00	100.00
1422	Source Maps	100.00	100.00	100.00
1421	Hot reloading	100.00	100.00	100.00
141	Typescript support	100.00	90.00	90.00
132	Rendering speed test (ms)	93.10	78.01	91.51
1312	Todo application (bundle)	65.10	96.40	96.90
1311	Hello world application (bundle)	48.75	87.50	90.00
125	Open source	100.00	100.00	100.00
124	Corporate backing	100.00	100.00	50.00
123	Age (years)	60.00	100.00	100.00
122	Number of Contributors	100.00	100.00	94.00
121	Frequency of Updates	100.00	92.00	60.00
112	Github stars	90.63	100.00	100.00
111	Weekly NPM downloads	87.74	100.00	97.90

3.4 Executive Summary

Based on the results of the evaluation, I would recommend either React or Vue. If I had to choose one, it would be React due to its current popularity and active development at Facebook. Angular no longer seems that attractive due to its large bundle size, lessening popularity, and difficulty to learn. Vue could easily win over React in the future, so if the stakeholder ultimately decides that they prefer an easy to learn framework that is easy to integrate into existing applications, Vue could be a good option. Despite Vue's competitiveness, it's hard not to choose React right now: There is excellent tooling, a huge community and pool of candidates, and proposed changes to the framework that will help it even further.

4. Conclusions

Overall the results are about what I expected. I did however expect there to be a larger gap between React and Vue initially, but after seeing the results and thinking about the two frameworks, they are very similar at least when compared to other frameworks. I would have liked to include more frameworks in the evaluation but ran out of time due to having to develop a few applications in each framework in order to gain an accurate insight while balancing finals week.