mitrais

continuous commitment

JAVASCRIPT FRAMEWORK DECISION, ANALYSIS, RESOLUTION (DAR)

VER.1.0

Table of Contents

Table of Figures

1. Overview

JavaScript has gone through a renaissance over the past 5 years. Since the initial release of the NodeJS runtime environment in 2009, there has been an immense amount of attention put into developing full stack JavaScript applications.

The single threaded nature of JavaScript decreases the complexity of web development pipelines. From this, there have been many improvements to the availability of JavaScript modules and building a website from reusable parts is now the norm.

A Web Developer using modern JavaScript frameworks is able to build singular components, or use open source components, and assemble a webpage much quicker than in the past.

Mitrais has evaluated the 5 best JavaScript frameworks that are most advanced, feature rich, and popular amongst the developer community and make it possible to build complex and feature rich single page interactive web applications.

This document summarises Mitrais' evaluation highlighting the advantages and importantly the limitations of the solutions to assist Mitrais customers in choosing the best JavaScript framework platform for their needs.

2. Evaluated Platforms

	PLATFORM	VERSION
1.	AngularJS	1.5.8
2.	ReactJS	15.3
3.	Ember.js	2.7
4.	Aurelia.js	1.0.1
5.	VueJS	1.0.26

Table 1 – Evaluated Platforms

3. Other Platforms

Mitrais decided to only assess 5 of the leading JavaScript Frameworks. There are many vendors in this space and customers may wish to assess some of the other frameworks which were not evaluated from the list below. Mitrais regularly reviews and updates its findings to cater for changing customer demand and market conditions, and may conduct DARs on these other platforms in the near future :

- PolymerJS 1.4.0
- RiotJS 2.6.1
- InfernoJS 0.7.25
- CycleJS 6.0.0

4. Evaluation Criteria

Criteria used:

1. **Availability of Tools**
   Availability of Tools was analysed using https://www.npmjs.com/, the official site of Node Package Manager in conjunction with GitHub to see the amount of tools, packages, and implementations associated with each library.

2. **API Documentation**
   This is to evaluate completeness of API documentation and quick start guides.

3. **Product Support**
   Reference was made to the availability of blogs, tutorials, sponsored educational content, as well as the availability of mentoring from the core team (see Appendix 4).

4. **Easy to Adopt**
   This is to evaluate how quickly can a user become proficient enough to render basic views with the framework. Things that contribute to this are the simplicity of the documentation and quick start guides, and complexity of the actual framework in practice.

5. **Templating**
   Templating is an analysis of rendering reusable components in each framework.

6. **Ease of Development**
   Ease of Development refers to the complexity of managing a team utilizing each of these frameworks. Some frameworks are more oriented towards isolated components, while others require more connected architectures.

7. **Performance/ Page Rendering**
   Performance is analysed against the speed of loading tables of data and performing rapid updates. Appendix 5 provides some statistics.

8. **Uptake by Developers**
   Uptake refers to the popularity of the framework in the development community.

9. **Initial Learning Curve**
   Initial learning curve is subjective, but refers to the complexity of setting up your initial project scaffolding.

10. **Availability and Cost of Learning Materials**

11. **Future Features and Upward / Backward Compatibility**
    Although some libraries are fairly new and haven't had time to address this problem, there are libraries that have made updates that are not compatible with previous versions and this needs to be considered and addressed in development.

12. **Routing Capability**
    Routing is the process of navigating views on a webpage. Frameworks handle this process differently.

13. Supports Isomorphic JavaScript

Isomorphic JavaScript is the new term ascribed to websites that can be rendering server side OR client side. This is useful when the complexity of the User Experience increases.

14. Internationalisation

This is the process of planning and adaptation of websites for use in a variety of languages and cultures.

15. Form Binding and Validation

Form Binding and Validation is the process of making changes or displaying information as a result of the input of a form.

16. Size

The size is the amount of information that needs to be loaded for the framework to function.

17. Starter Kits

Starter Kits refers to initial scaffolding tools for each library. Typically, a library will have example applications that you can work off of that already have key features like hot-reloading and a simple web server to save you the trouble of starting from a blank folder.

18. Testing Tools

Testing tools are libraries/frameworks or other conveniences designed to make developing and testing your application easier.

19. Security

Security refers to any specific security concerns associated with each framework.

20. Build and Deploy Tools

Build and deploy tools are those tools utilized for bundling your pages and delivering this content to your server for users to access.

21. Debugging Tools

Debugging tools are tools to aide developers in testing, especially End to End testing.

22. Scalability

Scalability addresses the possibility of growing the web application indefinitely in size and scope.

23. Maintainability

Maintainability is the ease of working on the framework, on-boarding developers, and having a consistent layout so your product's architecture does not become convoluted.

5. Platform Evaluation

All criteria have been given a scope of 1 – 5, where 1 is lowest assessment.

CRITERIA	EMBER	REACT	AURELIA	ANGULAR	VUE
1. Effort / availability of tools to upgrade legacy STRUTS apps	3	3	3	3	3
2. API Documentation	3	3	3	3	3
3. Licensing (Cost)	5	4	5	5	5
4. Product Support (Updates, community, documentation)	5	4	5	4	3
5. Easy to adopt	3	4	4	3	5
6. Templating (clientside vs serverside)	3	5	3	5	4
7. Ease of development	2	4	5	2	5
8. Performance / Page rendering speed	2	4	3	3	4
9. Uptake by developers (i.e.: who/how many are using it, how many stack overflow questions.)	5	5	2	5	2
10. SEO indexing	3	3	3	5	3
11. Initial Learning curve	2	4	4	2	5
12. Availability and cost of learning materials	4	4	2	4	2
13. Future Features & upward / backward compatibility	4	5	4	4	2
14. Routing capability	3	3	3	3	3
15. Supports isomorphic JavaScript	2	5	2	5	2
16. Internationalisation	3	3	3	3	3
17. Form Binding and Validation	3	3	3	3	3
18. Size (LOC and files)	1	4	4	3	5
19. Starter Kits	3	5	2	5	2
20. Testing Tools	4	5	4	4	2
21. Security	3	3	3	3	3
22. Build & Deploy Tools	3	3	3	3	3
23. Debugging Tools	3	5	3	5	2
24. Scalability	4	5	3	5	2
25. Maintainability	4	5	4	5	3
TOTAL SCORE	**80**	**101**	**83**	**95**	**79**

Table 2 – Platform Evaluation

6. Platform Recommendation

6.1. Detailed Evaluation

6.1.1. Effort / availability of tools to upgrade legacy STRUTS apps

Although there are no standard ways to migrate from STRUTS to a JavaScript framework, the general consensus is that a JavaScript framework shall be much easier to work with. In transitioning, the code will be much more succinct in any language and will no longer require XML config. files and verbose form handlers.

There is not one framework that stands above the rest in this category.

6.1.2. API Documentation

The Documentation for a framework or library is an essential lifeline in development. While this is true, it's important not to take the length of the documentation as an indicator of its worth. Some documentation, like AngularJS's, is significantly longer than the others. This is also an indicator of the complexity of the framework.

Ether and Angular would be the more complicated frameworks to work with, while Aurelia and React are slightly shorter and easier to pick up.

Aurelia has some more advanced features and is a complete framework so is slightly longer.

React is a library for rendering Views that integrates with Model and Controller libraries and so it has much shorter documentation.

VueJS is the shortest and easiest to pick up, but also has the least features.

There is no clear winner in this category.

6.1.3. Licensing

There are no major concerns here. All frameworks have the MIT License with the exception of ReactJS which has a 3-Clause BSD license.

Google, Microsoft, Apple and many other large competitors to Facebook have gone on record saying they are not concerned with using ReactJS on their projects and it is safe to assume there are no serious concerns with licensing.

6.1.4. Product Support

Right now ReactJS has the largest community around it, with a plethora of tutorials, blogs and books being written about it.

Ether has a very strong community as well, and you can get official support from developers on their core team.

To give some comparison, ReactJS has over 48,000 stars on GitHub, Ember has 16,000, Aurelia has 7,500, Vue has 26,000, Angular has 15,000.

Clearly ReactJS has the largest community, which plays into product support in a number of ways. Firstly, developers typically rely on stack overflow and blogs for

tutorials on implementation, and if the larger the development community, the easier it is to find a solution to a unique problem.

Angular's community is largely divided now between Angular 1 and 2, and so you will not get the same level of support on either of them as you would on a whole framework.

Ember.js has a very devout developer community and has terrific resources available for developers.

VueJS is very simplistic and so there is less demand for community support.

Lastly, Aurelia is very new and has a lot of excitement around it and official support from the development team, but is still in its infancy.

In this category, React is the clear winner.

6.1.5. Easy to Adopt

Ease of adoption depends on two factors:
a/ the design of the language, and
b/ the complexity of its features.

VueJS is clearly the easiest to learn, and a proficient JavaScript developer can pick up on its core features in an afternoon. In this sense it's easy to adopt, but it also has much fewer features than any of the other platforms.

ReactJS and Aurelia are also very clear and intuitive. Aurelia is a complete framework and has some cutting edge JavaScript features and so requires more training to pick up. ReactJS integrates with several other libraries such as Redux and React-Router, but is a very intuitive framework and scales in complexity with your education.

Ember and Angular are the most complicated to learn. Ember is reminiscent of Ruby on Rails and some familiarity with this concept is helpful. Likewise, Angular is a very complex framework that requires a great deal of training to become proficient.

ReactJS and Aurelia tie for this category, with VueJS being an honourable mention for smaller projects where development speed is paramount and there is minimal complex logic.

6.1.6. Templating

There are some benefits to each framework, and they all have unique approaches.

Ember, Angular and Aurelia has similar approaches with HTML templates.

VueJS is more declarative but is not a distinct departure from this technique.

ReactJS uses a syntax known as JSX to embed XML style syntax in JavaScript.

All templates are adequate and depend on the preference of the developer.

6.1.7. Ease of Development

The easiest framework to develop with is VueJS, but almost the most limiting. AngularJS is the most difficult to learn, with Ember.js being a close second.

ReactJS is very intuitive to learn and due to the concept of "unidirectional data flow", it's very explicit in how you develop your application.

Aurelia is a very unique framework but has a lot of distinct benefits, and requires a deeper understanding of modern JavaScript syntax to be effective.

An advanced developer in any language will be able to develop quickly, but ReactJS and Aurelia have strong benefits in building reusable components and using modular development.

6.1.8. Performance

React and Vue are slightly more performant than Angular 2.0 and Aurelia, and much more performant than Angular 1.0.

Ember is by far the slowest.

One benefit to using ReactJS is you can port your code to a framework like InfernoJS. Inferno is a library that uses the same syntax as ReactJS, but performs about twice as fast.

In this category React and Vue are the clear winners. Performance of any modern JavaScript framework will far outshine Struts though, and so Ember should not be disqualified on this principle.

6.1.9. Uptake by Developers

In stack overflow questions alone, ReactJS has 22,000, AngularJS has 195,000, VueJS has 0, Ember has 20,000, and Aurelia has 12,000.

Stack Overflow questions aren't always a measure of the support in the community. It can mean the framework is more confusing. For example, AngularJS has 72,000 of their questions unanswered.
In terms of downloads using node package manager, ReactJS is 500,000 a week. The second most is Angular with 118,000 and the rest are below 70,000.

6.1.10. SEO Indexing

There are some concerns with SEO in JavaScript applications. Dynamically generated content was not typically found by googlebot, as it failed to crawl the information fetched by the XMLHttpRequest API.

Companies like Prerender, BromBone, SEO.js are services that would integrate with your application to improve SEO.

AngularJS has some advantages in SEO due to its close relationship with Google, but SEO can be handled by any framework with some creativity.

6.1.11. Initial Learning Curve

Angular and Ember have the sharpest learning curves.

Vue is the easiest to on-board, and Aurelia and React are similar in their complexity to understand. They both have very reasonable on boarding and scale linearly in complexity. You could use them very simplistically, or you can begin to address some more of their advances features.

6.1.12. Availability of Learning Materials

React and Angular have the most available materials, with Ember being close behind.

Aurelia is still new so there are not as many available materials but there is a very dedicated and available development team.

VueJS has the least materials.

6.1.13. Future Features

Angular is the biggest culprit here with their drastic departure between Angular 1.0 and 2.0. They created an upload adapter that helps with the process, but it is not a trivial process.

Aurelia is newer and so some settling in syntax could be expected.

ReactJS has been very consistent since its inception, although there have been advancements in its routing and state handling. It is a very modular framework and so advancements are typically presented as new options for developers.

6.1.14. Routing Capability

Each framework has their own approach to routing. None are drastic departures from the norm, and so no preference can be given to one framework's routing capabilities.

6.1.15. Isomorphic JavaScript

ReactJS, AngularJS and Ember can be built as Isomorphic applications.

Currently Vue can be implemented with jsdom as a server side library, but this is not a popular option.

Aurelia is designed to be client side, and this is one of its strengths.

The benefits of client side development are particularly for design implementations, and so Aurelia and VueJS has found some popularity in developer/designer hybrids.

6.1.16. Internationalisation

All frameworks have a library for i18n. There's no cause for concern about any of the frameworks regarding this criteria.

6.1.17. Form Binding and Validation

Data binding can be handled multiple ways. Typically, as long as the developer understands REST, any framework from jQuery up will be adequate for handling this. There are a significant number of open source libraries on GitHub for form generation. Forms and data binding are essential to each framework.

6.1.18. Size

VueJS is the smallest by far at only 22kb.

Ember is the largest at 435kb.

React + Redux (a state handling library) is 156kb.

Aurelia comes in at 255kb.

Angular 2 is 698kb.

React, Vue and Aurelia are by far the most compact.

Ember and Angular's size can make them clunky on older browsers and mobile devices if they are not developed as server side applications.

6.1.19. Starter Kits

Yeoman is a popular framework for scaffolding web applications.

There are extensive generators for React and Angular.

Aurelia, Vue and Ember are sparser, but there are still some available.

Adequate resources are available for all, but React and Angular have significantly more options. ReactJS actually has a visual editor for building project scaffolds and components known as structor.

6.1.20. Testing Tools

ReactJS typically utilizes Jest for testing, a tool designed specifically for testing React.

Aurelia also has its own testing library, which is a set of helpers to simplify the process.

Angular, Ember and Vue do not have specialized testing frameworks, but there is a plethora of tools available for testing.

All frameworks have well documented suggestions on incorporating testing frameworks.

6.1.21. Security

Security is more about how the developer treats the sensitive information and is not a direct component of the front end rendering. All frameworks are equally viable in designing a secure application.

6.1.22. Build and Deploy Tools

Ether and Aurelia have Command Line Interfaces for building and deploying applications which is a nice convenience.
React has a suite of DevTools for development and state management.
Angular has a tool as well-known as ng-inspector for browser based debugging.

VueJS has a set of chrome extensions for development called vue-devtools.

As for deployment, no framework is especially designed to integrate with a deployment tool.

6.1.23. Debugging Tools

Each of these frameworks has their own browser extension for debugging.

6.1.24. Scalability

ReactJS, AngularJS, Ember.js are all currently used on production web applications.

Ember is used at Twitch, Yahoo, zendesk amongst countless other sites.

ReactJS is used at Facebook, Instagram, Salesforce, and other major web applications.

AngularJS was used to build Netflix, Google's greentea, upwork, and JetBlue.

Aurelia and Vue are smaller and have less production applications backing them, but are being commonly employed on smaller projects. Aurelia is gaining traction and is expected to be used on some larger projects in the near future.

6.1.25. Maintainability

A great deal of maintainability depends on the developers themselves. Aurelia, ReactJS, Ether and AngularJS have very specific design for development. AngularJS and Ember are more complicated frameworks and require more careful planning and organization.

If a developer uses the intent they provide for app development, they can build very modular and maintainable applications.

VueJS is the least maintainable as the application scales due to its very liberal and forgiving design.

ReactJS and Aurelia are very specific about how to build and organize your applications. Aurelia is slightly simpler in its design for individual components, but they both provide a very logical and obvious structure to development.

6.2. Summary

While all these libraries have their advantages and developers that will champion them as the "best" framework, right now ReactJS seems like the clear winner.

AngularJS has gone through some drastic changes lately and the community dwindled in its wake.

VueJS is an excellent framework for designers and prototyping applications, but does not scale with complex applications due to its simplistic design. Also, it has a smaller community and requires more autonomy from the developer. There is less structure and this can be dangerous to organizing a development team.

Aurelia is very exciting and promise to be a major framework in the near future, but it is still in its infancy as a project and while it is production ready, it does not have a large community of developers yet. They do provide official community support and so it is an attractive option for a cutting edge development team.

Ether is a very well established framework. It is larger and slower than all the others, but has a very dedicated community.

ReactJS shines above in its smaller size, well developed state handling libraries, and Native integration. ReactJS developers can reuse a significant amount of their capabilities programming in React-Native, a library with similar syntax used to make mobile applications.

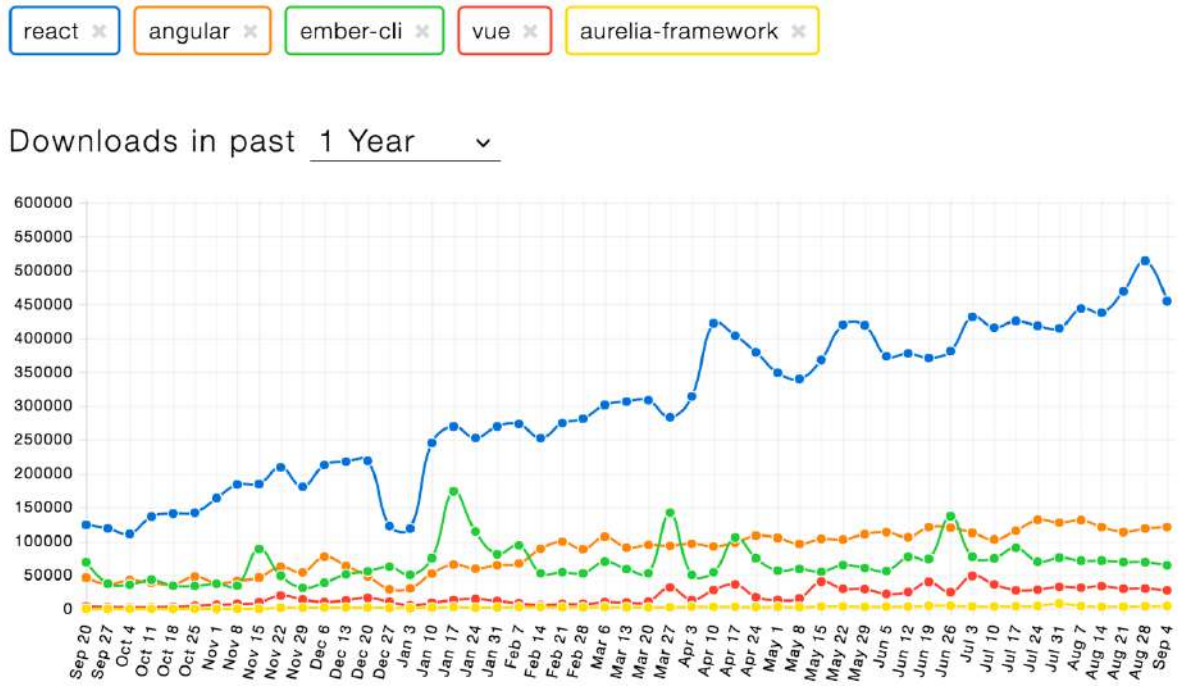
7. Conclusion

The Recommendation would be to utilize React + Redux, and eventually consider implementing React Native for mobile/tablet development.

The justification for this is its popularity, large open source community and consistent development of external resources. ReactJS is the most popular framework for a reason, and the fact that you can use your development team to build both mobile and web applications that use the same API makes it an extremely attractive choice.

Appendices

Appendix 1: *A chart comparing the amount of node package manager install for each of the 5 libraries/frameworks over the last year.*



Appendix 2: *A chart of each of the frameworks current popularity and open issues on Github. Open Issues represents concerns that the open source community has presented but have not yet been fixed, or recommendations for improvements.*

	STARS	OPEN ISSUES	OPEN BUGS
Aurelia	7649	32	47
ReactJS	48,882	540	659
Angular 2.0	52,033	689	863
Ember	16,798	186	273
Vue	26,841	35	42

Appendix 3: *Size of Frameworks:*

	ANGULAR 2.0	EMBER	REACT + REDUX	VUE	AURELIA
Size	566k	435k	139k	26k	255k

Appendix 4: *Frameworks Core Teams. Training means that the core team is available to provide training to your team, remotely or onsite. Uses framework means that the developers of the framework work on projects with the framework:*

	ANGULAR 2.0	AURELIA	EMBER	REACTJS	VUEJS
Members	20+	16+	12+	7+	1
Uses Framework	No	Yes	Yes	Initially	Yes
Training	No	Yes	Yes	No	Yes

Appendix 5: *Performance in Milliseconds*

	ANGULAR 2.0	AURELIA	EMBER	REACTJS	VUEJS
Creating 1000 rows	192.38	220.08	747.01	184.48	259.77
Remove Row	133.52	98.35	92.95	64.89	70.61
Update Every 10th Row in large table	11.87	11.71	37.67	18.78	15.76
Append Rows to large table	679.67	700.63	1211.5	326.41	743.44

Appendix 6: *Example View Rendering*

Vue:

```
<div id="app">
  {{ message }}
</div>
```

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue.js!'
  }
})
```

Angular 2.0:

```
  <body>
    <hello-app></hello-app>
  </body>
```

```
(function() {

  var HelloApp =
    ng.core
    .Component({
      selector: 'hello-app',
```

```
      template: '<h1>Hello World!</h1>'
    })
    .Class({
      constructor: function() {}
    });


document.addEventListener('DOMContentLoaded',
function() {

ng.platformBrowserDynamic.bootstrap(HelloApp);
  });


})();
```

ReactJS:

```
<div id="example"></div>
```

```
ReactDOM.render(
      <h1>Hello World!</h1>,
      document.getElementById('example')
    );
```

Aurelia:

```
<!-- Index.html -->

<body aurelia-app />
```

```
//app.js

Export class App{
    Message = 'Hello World';
}
```

```
//app.html

<template>
  <h1> ${message} </h1>
</template>
```

Ember (note the imports of jQuery, handlebars, and Ember):

```
<body>
  <script src="//code.jquery.com/jquery-
1.11.1.min.js"></script>
  <script
src="//builds.handlebarsjs.com.s3.amazonaws.com/handleb
ars-v2.0.0.js"></script>
  <script
src="//builds.emberjs.com/tags/v1.9.0/ember.js"></script>
</body>
```

```
Ember.TEMPLATES['index'] =
Ember.Handlebars.compile('<h1>He
llo World</h1>');

App = Ember.Application.create();
```

Copyright

Copyright © 2016 Mitrais

Disclaimer

Any and all information in this document has been compiled and provided for information purposes only. The information provided herein may include information compiled from a variety of third parties. Mitrais will not be liable for any loss, damage, cost or expense incurred in relation to or arising by reason of any person relying on the information in this document or any link to any website provided herein, whether or not caused by negligence on Mitrais' part. While Mitrais endeavours to provide the information up to date and correct, Mitrais make no representations or warranties of any kind, express or implied, about the accuracy, reliability, completeness or currency of the information or its usefulness in achieving any purpose. Readers of this document are responsible for assessing its relevance and verifying the accuracy of the content. In any event, the information provided herein should not be construed as providing advice whether legal or otherwise.