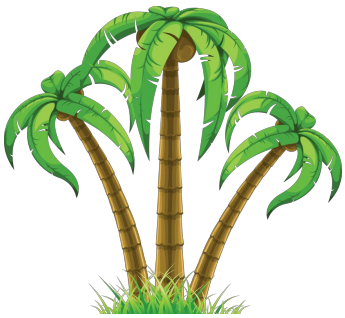
The jungle through Javascript frameworks.



Jonatan Karlsson
Web Programming
2013, BTH, Blekinge institute of technology
Advanced topic in Web development, PA1426
HT15
Karlskrona, Sweden
me@jonatankarlsson.se

Henrik Ölund
Web Programming 2013, BTH, Blekinge institute of technology
Advanced topic in Web development, PA1426
HT15
Karlskrona, Sweden
henke.olund@gmail.com

Abstract

In this article we have planned to dive into Javascripts world where new framework comes out “every day”. We will take the reader into a world where nothing are for granted and everything is a non-standard. In the current situation, there is a tremendous amount of Javascript frameworks[3] and that makes it difficult for a layman to choose the right framework, for the right task and this is something we will try figure out and explain to the reader.

Keywords: Javascript, Framework, MV*, Client-side, React, Mithril, Backbone.js, Ember.js

1. Introduction

1.1 Background

Today there are web technologies everywhere and these technologies are growing rapidly in the software industry. Technologies such as HTML, CSS and Javascript are used more than ever[1]. Almost everyone in europe has access to a web browser and are using it to view web sites[2] and as you probably know , the language of web browsers is Javascript.

To produce high quality javascript application it's very beneficial to use frameworks. Some might argue that is essential. They can help the structure the architecture and code to keep it modular and reusable. It helps the developers in the team to share the same common pattern on the application code. A framework can also help the team with providing useful features like "Templating", "Data-binding", "Routing" and "Security".

Choosing a framework might be exhausting, there are tons of them[3]. They are being updated, maintained and developed on every day. What should you even look for?

1.2 Intention

In this article we hope to enlighten the layman on which framework is the best to select for a client-based application that consume a REST-api.

In this context the phrase layman is a person working on a corporation and have gotten a new project where he should create a web based application that consumes a REST-api. The person in question have little experience regarding the javascript-frameworks since before but know the syntax of the language.

1.3 Method

First part

We will start with doing a selection of which four frameworks to investigate. The selection will be based on data from stats.js.org[3] and the selection of the frameworks we will choose from that site will be based on four questions;

Does the framework follow the MV*-pattern?

We chose this question criteria because we are only interested in frameworks that follows the MV* pattern. MV* is a acronym for different Modul View, insert something that a blogger finds good, design pattern. Examples of the something is "Presenter", "ViewModel" and the more common ending; "Controller"[4]. Essentially it's the same thing, but different words for solving problems with the architecture for the framework.

Is the framework popular on google?

This is very important while choosing a framework because it is the most common search engine in Sweden[5]. And therefore we used google trends to verified the selected frameworks popularity.

Have the framework risen in popularity since 2013?

The selection of frameworks should be popular. We can't go and investigate a framework that is less than, let's say, three weeks old. It doesn't make sense in a real world corporation to use a framework like that. In "every" young software there will be bugs, if not fully tested (which can never happen[6]), and this cannot be done in a short period of time. The framework will have "child diseases" and bugs in it. But if it is popular the possibility of the framework being tested in most situations and in depth is greater than a newly developed framework.

Does the framework have any corporation that backs them?

It is important that a framework is backed by a big corporation. Usually a big corporation has developers that are experienced in web development and therefore has knowledge of what is great and what isn't.

Second part

In the second part of the method we will collect data from each framework then try to get a understanding of each framework by asking ourselves these questions;

1. What philosophies have pushed this framework forward?
2. What kind of problem does this framework solve?
3. Which famous products has been created with this framework
4. Does the framework handle HTTP-request in a neat way?

When we have collected the data and interpreted it, we will write a personal paragraph about the findings based on the ultimate framework from our point of view. We will also use a benchmarking tool to compare the rendering speed where the framework will render a TODO-application.

2. Result

2.1 Which frameworks did we select?

We have chosen to research three popular frameworks and one framework that is a up-and-comer. The first one we selected was React due to being used in one of the most visited website on the world wide web www.facebook.com[7] and the popularity for React has risen steadily over the last year[8]. The second one we have chosen to investigate is Ember.js. Yahoo!, Linkedin, twitch, Netflix, Microsoft and Kickstarter are developed using Ember and the list goes on and on. There are a lot of big

corporations using the framework[9] and due to that the selection of the Ember framework was natural since it also follow the other criteria too. The third one is a favorite to one of our colleague. So this one was almost a must for us. The framework our colleague always talks about is Backbone.js. As any other framework selected for researching in this article this too follow the agreed upon criterias and it's also ranked quite high on
stats.js.org/ even though the popularity on google has gone down[8]. The last framework we have chosen to include is Mithril. It's not as popular as the others[8] but it has many strong points such as a great documentation[10], good code examples[11], and great performance[12].

2.2 Not included

There are many frameworks we could have chosen but it was out of scope for this paper. The only framework we thought about changing was Mithril with Angular. You have probably heard about Angular before since it is the most popular framework today[8]. It's backed and developed by Google and they use the framework[13] to develop their own sites. The amount of jobs where angular knowledge is a requirement is quite high[14]. We chose not to include the most popular framework because the amount of research papers and articles is very high[15] and we would rather use a something new and fresh that fits our criteria.

2.3 React 

What philosophies have pushed this framework forward?

"We built React to solve one problem: building large applications with data that changes over time."[16]

What kind of problem does this framework solve?

React is built for easy creation of the composable and dynamic user interface.[17] And because of this people likes to think of it as the V in the MVC.[16]

Which famous products has been created with this framework?

Facebook.com, instagram.com and imgur.com.[18]

Does the framework handle HTTP-request in a neat way?

No, React doesn't handle creations of HTTP-request in a neat way. In fact if you building a website in React you have to use an external dependency for making the HTTP-request. [19]

2.4 Ember.js 

What philosophies have pushed this framework forward?

The focus for Ember.js is ambitious web applications. The developers behind Ember.js always looks to take advantage of new browser and language features such as ES6 (future web standard foresight)[20]. Ember is a very opinionated framework that follows the "Conventions over configuration"[21][20]. Ember collected ideas from well known existing frameworks/libraries and implemented the things they found necessary[22]. Ember follows "stability without stagnation" which means that backwards compatibility is important and can be maintained while innovating new ideas and implementing them [23]. Ember provides very good defaults and cultivates shared abstractions but it also means that they are strongly opinionated[23].

What kind of problem does this framework solve?

Ember follow common idioms and design patterns so the developer can focus more on how to make the new app special, instead of reinventing the wheel[24].

Which famous products has been created with this framework?

Discourse, Vine, Live Nation, Twitch.tv, Nordstrom and even desktop applications such as Apple's Itunes Music[9].

Does the framework handle HTTP-request in a neat way?

Yes, there are two different ways to fetch data from a REST-API. Using plain JQuery or using an adapter. Ember comes with a standard adapter called "DS.RESTAdapter"[25][26].

2.5 Backbone.js 

What philosophies have pushed this framework forward?

Backbone is a light framework with few opinions. It is designed to help you organize your web application. It can be used to enhance your existing site or even create a completely new one, where the site is generated using client-side javascript[27]. It facilitates the separation between models and views.

What kind of problem does this framework solve?

Backbone provides a means of organizing your code into the MV* pattern.

Which famous products has been created with this framework?

Pinterest, Soundcloud, Trello, LinkedIn (mobile), BitTorrent.com and so on.

Does the framework handle HTTP-request in a neat way?

Yes, it is pretty neat. Backbone is pre-configured to sync with a RESTful API. You just have to create a Collection with the url of your resource endpoint and backbone handles it[28].

2.6 Mithril

What philosophies have pushed this framework forward?

The philosophies behind Mithril is to create applications where the code is discoverable, readable, maintainable and to make the developer become a better developer. [29]

What kind of problem does this framework solve?

Mithril is a fully developed MVC-framework that solves the most common problems when developing a dynamic website.

Which famous products has been created with this framework?

Flarum.org, Guild wars 2 and lichess.org [30]

Does the framework handle HTTP-request in a neat way?

Yes, Mithril has a built in API for handling HTTP-request and they simply call it “m.request”.[31]

3. What defines a great framework?

According to us a great framework is easy to learn and get started with. It needs to be fast, in production, and it should not be dependent on other frameworks or libraries. Another important factor in a framework is the documentation, which is the entry point for every developer. In our opinion it should go straight to the point and keep it simple at first then continue to more advanced concepts. The documentation should also have loads of examples to make the implementations of your own projects or mockups easy and fun. When selecting a new framework it is important to be able to quickly create a prototype to make the project manager happy. Without a prototype the corporation will never shift into a new gear to make the developers grow and increase their competence. A great framework should help, but not get in the way of your development. It makes simple things easier to develop and at the same time allows the developer to produce complex things as well.

There is also another important aspect of a framework and it’s the handling of XMLHttpRequest. The handling of those are important since it’s very common in a client based application to use external REST-api:s to store and get the data. When developing a new application it is nice to not feel the need to include another

dependency to handle those request or feel the need to implement it yourself. This is what we mean when we talk in our survey about a neat way to handle HTTP-request.

4. Discussion

When a standard does not exist it is hard to really make a proper decision. Every framework out there has it's own values and ways to solve the problem they have and the mindset of the authors is different which makes it even harder to really know what makes a framework good or even great. That is why we have chosen to compare the frameworks against our ideal framework.

The frameworks we have researched have all its own way of making the world a better place. They have all their own charm, like React and their own markup JSX which make it easy for a developer with HTML-experience to understand and use it. Ember with their opinionated how-you-should-make-it style. Mithril with their extremely fast view rendering and loading time. Backbone with their great open source reading materials such as "Developing Backbone.js applications"[32].

The codebase of the frameworks can become quite large and complex which increase the downloading time and evaluation time for the browser. For example the source code for Ember is 95kB which is quite large compared to Mithrils' 12kB or even Backbones' with their extremely small size of 7,3kB[33]. But we should note that you might need dependencies when developing with Backbone. Either way, the point we want to make is that Ember comes with a seemingly great developing tool called Ember-cli. You can use the tool to auto-generate a base for the application, generate blueprints for adapters, controllers, models, routes and use it for building the application and testing it. This will increase the speed of which an Ember application can be developed. This is something we haven't found on any other of the selected frameworks. But as we noted before, the size of the other frameworks are significant smaller and that's maybe why there are no CLIs for the other frameworks.
React and Mithril has both a small API which further increase the support for skipping the CLI-program for them. The previous statement is quite true for Backbone too, it's a bit larger than the others but it is less than Embers.

There are things that are similar with some of these frameworks. React and Mithril, for example, seems to have the same idéa about what makes the rendering be as fast and efficient as possible. Their approach is to use a virtual DOM[34] comparison of two version of the newly calculated view and the previous calculated view and then only change and render the parts that is different of the real view in the current DOM. The normal rendering principle is to always reevaluate the view and then render it which is significantly slower than the virtual DOM comparison[12].

There are also bad sides with every framework we have researched. Like React that only handle the view-part of the MVC-pattern. That was something we found when we gained even more knowledge about React. Even though the framework only cover the view part it is not excluded in this paper. This is since it has the fundamental view of

the correct MV* pattern. In the way they have implemented their applications they do not need to think of anything else than the view part and calling a REST, SOAP-API.

We found that with Ember it takes too long time to learn, because their documentation is very extensive and it constantly changes. And with Backbones own documentation is nothing to consider. It takes to long time to process and gives information about irrelevant stuff. But regardless of the irrelevant stuff in Backbone the community is great which is a big plus if you are interested to learn more from other people than the “official”, main, developers.

Without writing a line of code with Mithril, it seems way too good. In our experience the hype of Mithril might be to good to be true. Mithril is, at the moment of this entry, only one year old which is too young to be fully developed and tested in every situation. But the statistic of being the fastest framework on rendering a TODO-example it seems great. The documentation is good, it is even larger than the source code itself. It is straight to the point and with a lot of code examples which enhance the reading experience. And the copyright holder of the framework has a great blog where he talks about real life situations and how he solved it.

When you see API for Mithril it seems like the developers of the framework really thought it through. It is exactly a new developer wants it: easy, and small.

4. Conclusion

While following our ideal world we saw quickly that Backbone and React was not the framework we were looking for. They are both considered good but they lacked key features. In Backbone it seems a bit too complicated to create a neat application from scratch and the documentation was a pain. Ember might be a good alternative for the intention, though it falls on the amount of time it takes to really get comfortable with their API and the way they want to structure an application. Then there are only one viable candidate left; Mithril. Mithril has everything we are looking for in a framework. Even though it is new, it does everything right. It has a fast rendering speed and a simple and small API which makes the learning curve of the framework fast and easy.

5. Summary

There are a lot of frameworks available and it has become a jungle. This thesis aims to guide you through a few of the recent frameworks selected from the literature study. The frameworks were selected by asking *“relevant”* questions seen from a perfect world specified by the authors. The authors discuss positives and the negatives for each of the frameworks. Then made an informed decision of which framework to choose.

6. Future works

We would like to explore even more frameworks since there are many frameworks that fit the criteria. We only touched the uppermost layer and we really want to dive even further down. We would also love to write example applications with all the selected frameworks to get a hands on experience for each framework.

7. Abbreviations

REST-API:

> "REST (REpresentational State Transfer) is an architectural style, and an approach to communications that is often used in the development of Web services"[35]

XMLHttpRequest:

> "XMLHttpRequest is an API that provides client functionality for transferring data between a client and a server. It provides an easy way to retrieve data from a URL without having to do a full page refresh. This enables a Web page to update just a part of the page without disrupting what the user is doing. XMLHttpRequest is used heavily in AJAX programming."[36]

AJAX:

> "Asynchronous JavaScript + XML, while not a technology in itself, is a term coined in 2005 by Jesse James Garrett, that describes a "new" approach to using a number of existing technologies together, including: HTML or XHTML, Cascading Style Sheets, JavaScript, The Document Object Model, XML, XSLT, and most importantly the XMLHttpRequest object."[37]

SOAP-API:

> "SOAP, originally an acronym for Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission."[38]

8. References

[1] Making Sense Out of a Jungle of JavaScript Frameworks
http://link.springer.com/chapter/10.1007%2F978-3-642-39259-7_28
(Accessed 2015-10-18)

[2] Number of Internet Users
http://www.internetlivestats.com/internet-users/
(Accessed 2015-10-18)

[3] JS.ORG | STATS
http://stats.js.org/
(Accessed 2015-10-18)

[4] MVVM, MVC and MV* design pattern
http://leolanese.com/blog/?p=1377
(Accessed 2015-10-18)

[5] 2013 Search Engine Market Share By Country
http://returnonnow.com/internet-marketing-resources/2013-search-engine-market-share-by-country/
(Accessed 2015-10-18)

[6] Foundations of software testing
Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black
ISBN: 978-1-4080-4405-6

[7] The top 500 sites on the web.
http://www.alexa.com/topsites
(Accessed 2015-10-18)

[8] Google Trends
http://www.google.com/trends/explore#q=%22react%20js%22%2C%20%22Ember%20js%22%2C%20%22backbone%20js%22%2C%20%22Angular%20js%22%2C%20%22Mithril%20js%22&date=1%2F2e013%2034m&cmpt=q&tz=Etc%2FGMT-2
(Accessed 2015-10-18)

[9] WHO'S USING EMBER.JS
http://emberjs.com/ember-users/
(Accessed 2015-10-18)

[10] Mithril articles
http://lhorie.github.io/mithril-blog/
(Accessed 2015-10-18)

[11] Mithril API reference
http://lhorie.github.io/mithril/mithril.html

(Accessed 2015-10-18)

[12] Todo Application Benchmark
http://matt-esch.github.io/mercury-perf/
(Accessed 2015-10-18)

[13] Made with Angular
https://www.madewithangular.com/
(Accessed 2015-10-18)

[14] Angular careers on Stackoverflow
http://careers.stackoverflow.com/jobs?searchTerm=angular
(Accessed 2015-10-18)

[15] BTH Summon
http://bth.summon.serialssolutions.com/?q=angularjs#!/search?ho=t&l=en&q=angularjs
(Accessed 2015-10-18)

[16] Why React?
https://facebook.github.io/react/docs/why-react.html
(Accessed 2015-10-18)

[17] Why did we build React?
https://facebook.github.io/react/blog/2013/06/05/why-react.html
(Accessed 2015-10-18)

[18] Sites using React
https://github.com/facebook/react/wiki/Sites-Using-React
(Accessed 2015-10-18)

[19] 5 Practical Examples For Learning The React Framework
http://tutorialzine.com/2014/07/5-practical-examples-for-learning-facebooks-react-framework/
(Accessed 2015-10-18)

[20] Ember.js: Philosophy and design
https://en.wikipedia.org/wiki/Ember.js#Philosophy_and_design
(Accessed 2015-10-18)

[21] Convention over configuration
https://en.wikipedia.org/wiki/Convention_over_configuration
(Accessed 2015-10-18)

[22] Is Angular.js or Ember.js the better choice for JavaScript frameworks?
https://www.quora.com/Is-Angular-js-or-Ember-js-the-better-choice-for-JavaScript-frameworks

(Accessed 2015-10-18)

[23] Ember.js
https://en.wikipedia.org/wiki/Ember.js
(Accessed 2015-10-18)

[24] Building Web Apps with Ember.js
Jesse Cravens, Thomas Q Brady
ISBN:978-1-4493-7087-9

[25] Connecting To An HTTP Server
http://guides.emberjs.com/v1.10.0/models/connecting-to-an-http-server/
(Accessed 2015-10-18)

[26] Pushing Records Into The Store
http://guides.emberjs.com/v2.1.0/models/pushing-records-into-the-store/
(Accessed 2015-10-18)

[27] Backbone Book
http://nicholasjohnson.com/backbone-book/
(Accessed 2015-10-18)

[28] Backbone.js API-Integration
http://backbonejs.org/#API-integration
(Accessed 2015-10-18)

[29] Mithril getting started
http://mithril.js.org/getting-started.html
(Accessed 2015-10-18)

[30] Who Uses Mithril
https://github.com/lhorie/mithril.js/wiki/Who-Uses-Mithril
(Accessed 2015-10-18)

[31] Mithril API reference m.request
https://lhorie.github.io/mithril/mithril.request.html
(Accessed 2015-10-18)

[32] Developing Backbone.js Applications
http://addyosmani.github.io/backbone-fundamentals/
(Accessed 2015-10-18)

[33] Comparison of JavaScript frameworks
https://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks
(Accessed 2015-10-18)

[34] Document Object Model
https://en.wikipedia.org/wiki/Document_Object_Model
(Accessed 2015-10-18)

[35] What is REST (representational state transfer)?
http://searchsoa.techtarget.com/definition/REST
(Accessed 2015-11-05)

[36] XMLHTTPRequest
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest
(Accessed 2015-11-05)

[37] Ajax
https://developer.mozilla.org/en-US/docs/AJAX
(Accessed 2015-11-05)

[37] SOAP protocol
https://en.wikipedia.org/wiki/SOAP
(Accessed 2015-11-05)