Rich JavaScript Applications – the Seven Frameworks (Throne of JS, 2012)

A week ago was the *Throne of JS* conference in Toronto, perhaps the most interesting and different conference I've been to for a while. Quoting its website:

Published Aug 1, 2012

> *It's no longer good enough to build web apps around full page loads and then "progressively enhance" them to behave more dynamically. Building apps which are fast, responsive and modern require you to completely rethink your approach.*

The premise was to take the seven top JavaScript frameworks/libraries for single-page and rich JavaScript applications — **AngularJS, Backbone, Batman, CanJS, Ember, Meteor, Knockout, Spine** — get the creators of all of them in one location, and compare the technologies head to head.*

Disclaimer: I was there to represent Knockout, so obviously I'm not neutral. In this post my focus is on what the creators said about the scope and philosophy of their technologies, and not so much on whether I agree or disagree.

** Yes, I know that's eight frameworks, not seven. This part was never fully explained...*

TL;DR Executive Summary

- For many web developers, it's now taken for granted that such **client-side frameworks are the way to build rich web apps**. If you're not using one, you're either not building an application, or you're just missing out.
- There's **lots of consensus** among the main frameworks about how to do it (Model-View-* architecture, declarative bindings, etc. —

to do it (Model-View-* architecture, declarative bindings, etc. details below), so to some extent you get similar benefits whichever you choose.
- Some major **philosophical differences** remain, especially the big split between frameworks and libraries. Your choice will deeply influence your architecture.
- The conference itself was stylish and upbeat, with a lot of socialising and conversations across different technology groups. I'd like to see more like this.

Technologies: Agreement and Disagreement

As each SPA technology was presented, some fairly clear patterns of similarity and difference emerged.

Agreement: Progressive enhancement isn't for building real apps.

All the technologies follow from the view that serious JavaScript applications require proper data models and ability to do client-side rendering, not just server rendering plus some Ajax and jQuery code.

Quote from Jeremy Ashkenas, the Backbone creator: "*At this point, saying 'single-page application' is like saying 'horseless carriage'*" (i.e., it's not even a novelty any more).

Agreement: Model-View-Whatever.

All the technologies made use of model-view separation. Some specifically talked about MVC, some about MVVM, and some specifically refused to define the third piece (just saying it's models, views, and some kind of application thing that makes them work together). The net result in each case was similar.

Agreement: Data binding is good.

All except Backbone and Spine have a built-in notion of declarative

data binding in their views (Backbone instead has a "bring your own view technology" design).

Agreement: IE 6 is dead already.

In a panel discussion, most framework creators said their IE support focus was limited to version 7+ (in fact, Ember and AngularJS only go for IE8, and Batman requires an ES5 shim to run on IE older than v9). This is the way of things to come: even jQuery 2 is set to drop support for IE older than v9.

The only stalwarts here appear to be Backbone and Knockout which support IE6+ (I don't know about Backbone's internals, but for KO this means transparently working around a lot of crazy edge-case IE6/7 rendering and eventing weirdnesses).

Agreement: Licensing and source control.

Every single one is MIT licensed and hosted on GitHub.

Disagreement: Libraries vs frameworks.

This is the biggest split right now. You could group them as follows:

Libraries	Frameworks
Backbone (9552)	Ember (3993)
Knockout (2357)	AngularJS (2925)
Spine (2017)	Batman (958)
CanJS (321)	Meteor (4172) — unusual, see later

Numbers in brackets are a point-in-time snapshot of the number of GitHub watchers, as a crude indicator of relative influence.

What does this mean?

- **Libraries** slot into your existing architecture and add specific functionality
- **Frameworks** give you an architecture (file structure, etc.) that you are meant to follow and, if you do, are intended to handle all common requirements

By far the most passionate advocate of the framework model is Ember, whose creator Yehuda Katz is formerly of the Rails and SproutCore projects (similar philosophy). His argument was that anything less is just not ambitious enough and isn't seriously advancing the state of the art. The counter-argument is that libraries are more focused, and hence can be easier to learn, adopt, customize, and help minimise project risk because your architecture isn't so deeply tied to a specific external project. Based on my conversations, I'd say the audience was split and supported both sides of this debate.

Note that AngularJS is arguably somewhere in between library and framework: it doesn't require a particular layout of files at development time (library-like), but at runtime it provides an "app lifecycle" that you fit your code into (framework-like). I'm listing it as a framework because that's the terminology the AngularJS team prefers.

Disagreement: What's flexible, what's integrated.

Each technology has different levels of prescriptiveness:

	Views	URL routing	Data storage
AngularJS	Built-in DOM-based templates (mandatory)	Built-in (optional)	Built-in system (optional)
Backbone	Choose your own (most used handlebars.js, a string-based template library)	Built-in (optional)	Built-in (overridable)
Batman	Built-in DOM-based templates (mandatory)	Built-in (mandatory)	Built-in system (mandatory)
CanJS	Built-in string-based templates	Built in	Built in (optional)

	(mandatory)	(optional)	
Ember	Built-in string-based templates (mandatory)	Built-in (mandatory)	Built-in (overridable)
Knockout	Built-in DOM-based templates (optional, can do string-based too)	Choose your own (most use sammy.js or history.js)	Choose your own (e.g., knockout.mapping or just $.ajax)
Meteor	Built-in string-based templates (mandatory)	Built-in (mandatory?)	Built-in (Mongo, mandatory)
Spine	Choose your own string-based templates	Built-in (optional)	Built-in (optional?)

As expected, whenever a library leaves a decision open, they argue it is better overall to guarantee composablity with arbitrary 3rd-party libraries. And the obvious counter-argument is that integration can be more seamless if built-in. Again, based on my conversations, the audience was split and opinions went in all directions — usually based on how much other technology stack an individual was wedded to.

Quote from Tom Dale of Ember: "*We do a lot of magic, but it's good magic, which means it decomposes into sane primitives.*"

Disagreement: String-based vs DOM-based templates

(As shown in the above table.) For string-based templates, almost everyone used Handlebars.js as the template engine, which seems to dominate this space, though CanJS used EJS. Arguments in favour of string-based templates include "*it's faster*" (debatable) and "*theoretically, the server can render them too*" (also debatable, as that's only true if you can actually run all of your model code on the server, and nobody

can actually run all of your model code on the server, and nobody actually does that in practice).

DOM-based templates means doing control flow (each, if, etc.) purely via bindings in your actual markup and not relying on any external templating library. Argument include "*it's faster*" (debatable) and "*the code is easier to read and write, because there's no weird chasm between markup and templates, and it's obvious how CSS will interact with it*".

In my view, the strongest argument here came from the AngularJS guys who stated that in the near future, they expect DOM-based templating will be native in browsers, so we'll best prepare ourselves for the future by adopting it now. AngularJS is from Google, so they are already working on this with Chromium and standards bodies.

Disagreement: Levels of server-agnosticism

Batman and Meteor express explicit demands on the server: Batman is designed for Rails, and Meteor is its own server. Most others have a goal of being indifferent to what's on your server, but in practice the architecture, conventions, and some tooling in Ember leans towards Rails developers. Ember absolutely works on other server technologies too, though today it takes a little more manual setup.

The technologies — quick overview

Here's a rundown of the basic details of each technology covered:

Backbone

- **Who**: Jeremy Ashkenas and DocumentCloud
- **What**:
  - Model-View in JavaScript, MIT licensed
  - Most minimal of all the libraries — only one file, 800 lines of code!
  - Extremely tightly-scoped functionality — just provides REST-persistable models with simple routing and callbacks so you know when to render views (you supply your own view-rendering mechanism).
  - The best-known of them all, with the most production deployments on big-name sites (perhaps easy to adopt

deployments on big name sites (perhaps easy to adopt because it's so minimal)

- **Why**:
  - It's so small, you can read and understand all of the source before you use it.
  - No impact on your server architecture or file layout. Can work in a small section of your page — doesn't need to control whole page.
  - Jeremy seems to exist in a kind of zen state of calm, reasonable opinions about everything. He was like the grown up, supervising all the arguing kids.
- **Where**: GitHub and own site
- **When**: In production for nearly 2 years now

Meteor

- **Who**: The Meteor development group, who just raised $11.2 Million so they can do this full-time
- **What**:
  - Crazy amazing framework from the future, barely reminiscent of anything you've ever seen (except perhaps Derby)
  - Bridges a server-side runtime (on Node+Mongo) with a client-side one, so your code appears to run on both, including the database. WebSockets syncs between all client(s) and server.
  - Does "live deployments" every time you edit your code – client-side runtimes are updated on the fly without losing their state
  - Makes more sense if you watch the video
  - Like everyone I spoke to at the event, I really want this to succeed — web development needs something this radical to move forwards
- **Why**: You've had enough of conventional web development and now want to live on the bleeding edge.
- **Where**: GitHub and own site
- **When**: It's still early days; I don't know if there are any production Meteor sites yet except built by the core team. They're totally serious about doing this, though.

Ember

- **Who**: Yehuda Katz (formerly of jQuery and Rails), the Ember team, and Yehuda's company Tilde
- **What**:
  - Everything you need to build an "ambitious web application", MIT license
  - Biggest framework of them all in both functionality and code size
  - Lots of thought has gone into how you can decompose your page into a hierarchy of controls, and how this ties in with a statemachine-powered hierarchical routing system
  - Very sophisticated data access library (Ember.Data) currently in development
  - Intended to control your whole page at runtime, so not suitable for use in small "islands of richness" on a wider page
  - Pretty heavily opinionated about files, URLs, etc., but everything is overridable if you know how
  - Design inspired by Rails and Cocoa
  - Tooling: They supply project templates for Rails (but you can use other server platforms if you write the code manually)
- **Why**: Common problems should have common solutions — Ember makes all the common solutions so you only have to think about what's unique to your own application
- **Where**: GitHub and own site
- **When**: Not yet at 1.0, but aiming for it soon. API will solidify then.

AngularJS

- **Who**: Developed by Google; used internally by them and MIT licensed.
- **What**:
  - Model-View-Whatever in JavaScript, MIT licensed
  - DOM-based templating with observability, declarative bindings, and an almost-MVVM code style (they say Model-View-Whatever)
  - Basic URL routing and data persistence built in
  - Tooling: they ship a Chrome debugger plugin that lets you explore your models while debugging, and a plugin for the

Jasmine testing framework.

- **Why**:
  - Conceptually, they say it's a polyfill between what browsers can do today and what they will do natively in a few years (declarative binding and observability), so we should start coding this way right now
  - No impact on your server architecture or file layout. Can work in a small section of your page — doesn't need to control whole page.
- **Where**: GitHub and own site
- **When**: In production now (has been at Google for a while)

Knockout

- **Who**: The Knockout team and community (currently three on the core team, including me)
- **What**:
  - Model-View-ViewModel (MVVM) in JavaScript, MIT licensed
  - Tightly focused on rich UIs: DOM-based templates with declarative bindings, and observable models with automatic dependency detection
  - Not opinionated about URL routing or data access — combines with arbitrary third-party libraries (e.g., Sammy.js for routing and plain ajax for storage)
  - Big focus on approachability, with extensive documentation and interactive examples
- **Why**:
  - Does one thing well (UI), right back to IE 6
  - No impact on your server architecture or file layout. Can work in a small section of your page — doesn't need to control whole page.
- **Where**: GitHub and own site
- **When**: In production for nearly 2 years now

Spine

- **Who**: Alex MacCaw
- **What**:
  - MVC in JavaScript, MIT license

    - Worked example originally written for an O'Reilly book grew into an actual OSS project
    - Is a kind of modified clone of Backbone (hence the name)
- **Why**: You like Backbone, but want a few things to be different.
- **Where**: GitHub and own site
- **When**: It's past v1.0.0 now

Batman

- **Who**: the team at Shopify (an eCommerce platform company)
- **What**:
    - MVC in JavaScript, almost exclusively for Rails+CoffeeScript developers, MIT licensed
    - Most opinionated of them all. You *must* follow their conventions (e.g., for file layout and URLs) or, as they say in their presentation,"*go use another framework*"
    - Full-stack framework with pretty rich models, views, and controllers and routing. And observability mechanism of course.
    - DOM-based templating.
- **Why**: If you use Rails and CoffeeScript, you'll be right at home
- **Where**: GitHub and own site
- **When**: Currently at 0.9. Aiming for 1.0 in coming months.

CanJS

- **Who**: the team at Bitovi (a JavaScript consulting/training company)
- **What**:
    - MVC in JavaScript, MIT licensed
    - REST-persistable models, basic routing, string-based templating
    - Not widely known (I hadn't heard of it before last week), though is actually a reboot of the older JavaScriptMVC project
- **Why:** Aims to be the best of all worlds by delivering features similar to the above libraries while also being small
- **Where:** GitHub and own site
- **When:** Past 1.0 already

Summary

Summary

If you're trying to make sense of which of these is a good starting point for your project, I'd suggest two questions areas to consider:

- **Scope**. How much do you want a framework or library to do for you? Are you starting from blank and want a complete pre-prepared architecture to guide you from beginning to end? Or do you prefer to pick your own combinations of patterns and libraries? Either choice has value and is right for different projects and teams.
- **Design aesthetic**. Have you actually looked at code and tried building something small with each of your candidates? Do you *like doing it*? Don't choose based on descriptions or feature lists alone: they're relevant but limited. Ignoring your own subjective coding experience would be like picking a novel based on the number of chapters, or a spouse based on their resume/CV.

Despite the differences, I'd argue there is one killer feature all the above technologies share: the idea of Model/View separation. This is a classic design pattern that predates the web itself by about 20 years. If you're building even the most basic kind of web application UI, you'll almost certainly benefit from applying this on the client.

OLDER NEWER

WRITTEN BY

Steve Sanderson

A web developer with a particular interest in building rich JavaScript apps. I work at Microsoft on the ASP.NET team. Follow me on Twitter.

Recommend Share Sort by Best

Join the discussion…

Clark • 3 years ago
Various publications and articles about software and computer technology. On web-site pages You can find details about machines and devices, drivers and security software, browsers and optimization, utilities and organizers.
• Reply • Share ›

Tony Brown • 3 years ago
It was fun to read, amazing all the miss guided devs out there that are just jumping on the hipster bandwagon, don't get me wrong, I absolutely love working with Backbone , Node and Mongo and either Mocha or Jasmine to test but it's kinda funny how people are just dead up wrong in their thinking "Angular is the best, or Knockout rules" lol good read :D
• Reply • Share ›

singapore website design compa • 3 years ago
Hi, Neat post. There is a problem along with your site in internet explorer, may test this? IE still is the market chief and a large component to people will omit your great writing because of this problem.
• Reply • Share ›

design companies in singapore, • 3 years ago
Thanks for some other fantastic article. The place else may just anyone get that type of information in such an ideal way of writing? I have a presentation subsequent week, and I am on the look for such info.
• Reply • Share ›

Terry • 3 years ago
Check out Lava JS. It looks very cool! http://lava.codeplex.com
• Reply • Share ›

Nathaniel • 3 years ago
Hi there colleagues, its impressive post concerning tutoringand entirely explained, keep it up all the time.
• Reply • Share ›

Earl • 3 years ago
I'm wondering why SproutCore was mentioned in the article but not part of the comparison? Was it just that they didn't have anyone at the event? Or is SproutCore a different class of platform?
• Reply • Share ›

Dave Smith • 3 years ago
Terrific summation. Thanks for taking the time to do this!

I'm going to watch Meteor closely.

I'm thinking KnockOut could benefit a lot from Angular's simplistic expression language. E.g.: {{foo}} vs. data-bind="text: foo"
• Reply • Share ›

Bryan Garaventa • 3 years ago

Previously mentioned was accessibility, and this actually is very important, since it depends on the functional behaviors of interactive component design.

ARIA by itself will not fix this.

There is a Bootstrapping library available at
http://whatsock.com/bootstrap
That helps with this.

▲ | ▼ • Reply • Share ›

FORTRAN • 3 years ago

Magnificent site. Plenty of helpful info here. I'm sending it to several buddies ans additionally sharing in delicious. And obviously, thanks in your sweat!

▲ | ▼ • Reply • Share ›

Aman • 3 years ago

It seems that Mr. K has stolen the show here.

'Any thoughts about Dart-based frameworks? What about Dart Web Components?'

A quick search tells me that: we all were sleeping while the next big thing for Web Apps was taking place in the form of Dart.

@K Sir request you to please say more. Some of us don't have mental blocks to see the obvious next big thing.

Exciting! & Amazing!

▲ | ▼ • Reply • Share ›

K • 3 years ago

Any thoughts about Dart-based frameworks? What about Dart Web Components?

▲ | ▼ • Reply • Share ›

dekorasyon • 3 years ago

nice posts, thank you for sharing with us.

▲ | ▼ • Reply • Share ›

Aman • 3 years ago

Initially chose Knockoutjs but got stuck.
1) there is no multi-view management. You can't build Single Page Applications with it.
2) you need at least 10 other js libraries to use knockoutjs, as it hardly handles much.
We wasted 1-2 months figuring out what all needed to be used. This can be confirmed from other user's experiences. There is a tutorial on Pluralsight and a blog on SPA using ASP.NET MVC and knockoutjs. The author has had to spend lot of time evaluating all these libraries.

Emberjs is more verbous than Angularjs. .get() and .set() need to be used just to read and write properties. You have to explicitly define all viewmodel properties which is not required in Angularjs. (Please correct me if I am wrong)

Finally we got everything (almost) and more from Angularjs in just a 29k download. Believe me there are a ton of features in just 29k download.

The things we would never find time to use are available and so easy. Inbuilt

The things we would never find time to use are available and so easy. Inbuilt internationalization. There is a superb testing framework also.
We will finally be able to write tests for our App.

Amazing!

We are thankful to Angularjs team for doing such a great job.

▲ | ▼ • Reply • Share ›

Michael • 3 years ago

Different interesting publications and articles about Michel Telo. On site You can find life history and notes from MSM, photographs and discography, tours data and lyrics, video materials and social links.

▲ | ▼ • Reply • Share ›

Eddie • 3 years ago

Different wholesome notes and articles about commercial law. On web-site pages You can find data about medium business and small business, cash and currencies, internet business and profitable business, also companies and customers.

▲ | ▼ • Reply • Share ›

Stephanie • 3 years ago

Useful publications and articles about various fireplaces for everyone. On site You can find data about granite and marble, onychite and adarce, indoor scene and tabletops, veneer and stairs, also drawing rooms and antechambers.

▲ | ▼ • Reply • Share ›

Mary • 3 years ago

Different useful publications and notes about garden. On site pages You can find data about agricultural engineering and agricultural methods, orchard management and agrotechnology, also gardening tools and gardening equipment.

▲ | ▼ • Reply • Share ›

Sarah • 3 years ago

Various singular winged words for all. On site pages You can find winged words about music and english language, summer and beauty, politics and love, science and smile, friendship and life, dreams and winter.

▲ | ▼ • Reply • Share ›

Ida • 3 years ago

Big collection of various replies. On site pages You can find data about software and computers, art and culture, Internet and music, jurisprudence and english language, bdellotomy and fishery, foreign words and thesis defence.

▲ | ▼ • Reply • Share ›

EvanZ • 3 years ago

I have to say, web development these days seems like the Wild West to an outsider. It's hard to know as a novice which of these libraries (or frameworks!) to choose, and once you think you have a handle on it, something new and shinier comes along. I guess that's good for the industry as a whole in the long run that competition should make the tools that much better.

Anyway, thanks for putting this information out there.

▲ | ▼ • Reply • Share ›

Tactiles • 3 years ago

Nice! Its very easy to know about architect on this site.Very useful information here.Thanks

• Reply • Share ›

Diego Martelli • 3 years ago

Great article. Thank you.
There are a lot of library/framework and few time to try and understand each one.
Some times we works in one man or all senior front end team, sometimes we works with js junior (super skilled java or c# developer may be JavaScript junior too). So for me Another great point to analyse is: what about the learning curve for junior and senior developer?

• Reply • Share ›

J A Fernandez • 3 years ago

I have been developing a small mobile app in AngularJS and have been quite satisfied with it. We usually write lots of server side Java code, but in this instance wanted a purely client-side app (except for a small JSON feed for the data to be displayed) running out of plain Apache.

My question is: Given the completely different issues on browser capabilities in the mobile/tablet segments, how does this affect the usability these frameworks?

There is no IE6 issue but lots of other like processor speed and browser fragmentation even in the "smart-phone" segment. Any thoughts of which ones of these frameworks are designed with mobile as one of their targets?

• Reply • Share ›

Ezekiel Victor • 3 years ago

I have to say I'm completely with Wil Moore III. It seems like the "enterprise" frameworks attempt to create entirely new languages -- presumably for people who don't understand JavaScript. An example of this and a testament to learning the language is the perceived issue of "zombie events" in Backbone.js. That is a clear lack of understanding of the DOM, event binding, etc. In many examples of zombie events, you see the person writing the code making obvious mistakes creating statically referenced objects instead of non-static clearly showing a lack of understanding for the language.

At this time I'm a believer in Backbone.js because it embraces what JS developers already know -- events -- and suggests a level of organization that we're already familiar with, models and views (quite literally; some other frameworks use the M, V, and C terms so loosely it's frustrating). I'm intrigued by the observables and bindings of KO and AngularJS, yet I'm still not convinced that will scale in terms of performance and maintainability.

• Reply • Share ›

jeff • 3 years ago

All you Sencha-philes: Sencha is not free.

• Reply • Share ›

Neha Khanna • 3 years ago

I have gone through 4 frameworks Sandy, Backbone, KnockoutJS and PureMVC. I liked Sandy for lightweight and quick development where as i prefer KnockoutJS for some big

developments especially when you are migrating some flex applications to HTML5.

Reply • Share ›

Dawesi • 3 years ago

Can't be taken seriously without Sencha frameworks in the comparison.

Pity, would have been great to see a framework that supports everything from IE6 to the latest kick all butts on all browsers!

Reply • Share ›

Maarten • 3 years ago

Why are there so few people aware of Wakanda?
It's a complete stack from backend to frontend based on javascript, html5, css.
noSql database included with awesome modelling.
http://www.wakanda.org/

Reply • Share ›

dan • 3 years ago

Currently at work we build websites for clients, an aggregated mobile site and have started to build native apps.

We are currently looking at rebuilding our mobile website as well as another web application that will have to be mobile friendly. At the current moment in time we use jQuery for the sites that we use.

Seeing as we are building a new web app and a new aggregated m-site I thought it would be a good time to review frameworks both for web and mobile and see if we need to change from jQuery.

So the frameworks I have looked into are

jQueryMobile Sencha with Sencha Touch JQtouch JQmobi

Backbone Knockout

My inital thoughts are to stay with jQuery as the knowledge is already here and dont want just to change for the sake of it but at the same time if there is another Framework we should be using I want to look into it

see more

Reply • Share ›

Jerry • 3 years ago

A) re: SEO and "one-off" apps; when javascript controls the client, the html file become a container of rich metadata, that upon docload, is rejected for the User interface. Think if every keyword, and ONLY keywords were contained within the body tag of your document. This would greatly enhance your ratings.

B) I'm finding that the closer to 3GL any library gets, the greater the support. Browsers will one day (hopefully W3C compiant) be 3GL runtime engines. They already are if you can code javascript. Think about how much an impact xBASE had on the world. Javascript needs to be reduced to 3GL just like C and C++ were reduced to 3GL like dBase, Clipper, Foxpro and Paradox.

C) Glad to hear there are many others out there thinking along these lines.

Reply • Share ›

Shankar • 3 years ago

Very nice post. Got fantastic idea about different library/framwork.

Thanks

• Reply • Share ›

Angel. L • 3 years ago

What do you think about Qatrix (http://qatrix.com)? It`s also small and high performance.

• Reply • Share ›

Kevin M. • 3 years ago

Kudos for this summary and its obvious evenhandedness. A couple of nits:
1. It would even more meaningful to weight the followers/watchers of a product by the amount of time the product has been available (as soft as that is..)
2. Agree 100% with the point about KO's awesome docs & tuts; you could note that AngularJS also sets a high bar for the docs and tuts. Not so for the others; the developers focus seems so much to be on dev'g the product that communicating about its benefits and use falls entirely to the reader. How dumb is that?

Anyway, thanks again for the clarity, SS.

• Reply • Share ›

Dan • 3 years ago

I'd like to hear more about why Progressive Enhancement is dead. Seems like having a version that has no JavaScript and one that has rich interactivity are both possible.

Also, if a browser didn't support HTML5 video wouldn't you still want to try to fall back to Flash? Or if a browser didn't support drag and drop, you still support the upload button?

• Reply • Share ›

Pan • 3 years ago

Hi，can we translate this article to Chinese on our website www.ituring.com.cn ?

Ituring focus on IT and science books and information, and we use markdown on this website. Of course, this article will open to all visitors freely, and there will be a link back if you need.

• Reply • Share ›

Wil Moore III • 3 years ago

morale => moral (but you knew what I meant)

• Reply • Share ›

Wil Moore III • 3 years ago

In my opinion, the morale of the story is:

(a) go read the Backbone source and learn object attribute binding, routing, and see what clean/testable JavaScript looks like.
(b) take what you've learned and write your app on top of light-weight single-purpose components (without or without Backbone and definitely without a monolithic framework).

But hey, if one of those mono-frameworks is an 80%+ fit for your scenario, go for it.

• Reply • Share ›

Yakup İpek • 3 years ago

I found meteor.js have amazing features but one thing was looking horrible when i see that client is able to write to db. After check this question on stackoverflow http://stackoverflow.com/quest.... I understood that it is unfortunality not ready for production :).

How can we develop a serious app without restrict user permissions to db on client side ???

By the way it is again a great post Steve .

Please keep posting

• Reply • Share ›

eMBee • 3 years ago

nice overview. one question i am wondering about though is how updates are detected as explained here http://stackoverflow.com/quest...
it explains that angularjs does a simple dirty-checking whereas knockout or backbone use change listeners.

how do the others fare here?
was this topic talked about?

greetings, eMBee.

• Reply • Share ›

yauritux • 3 years ago

Hi bro, nice article. Initially, i hate javascript because i don't like and i don't trust any codes that running on client side. Thats why i was using some java frameworks such as zkoss/ZK, GWT, etc to do the javascript jobs. But now, in my current company i have to use many javascript codes. thats why i have to go in deep with the javascript and i've found many articles and tutorials about javascript stuffs. I've just known that javascript also had many good OOP concepts, frameworks and design patterns too. so, i thought it's not pretty hard for a guy that already familiar with java like me to learn it. that makes me very interested to learning this language now. So, thanks bro to give us such a nice article. Btw, i'm from Indonesia and currently living and working in Malaysia. Two thumbs up for u :-). btw, i'll add your blog in my blog as a buddy. I hope you don't mind with that :-)

• Reply • Share ›

Alan • 3 years ago

Thanks for being the first read on the Internet able to clearly explain frameworks and libraries without making my head explode. A fresh read.

• Reply • Share ›

Nick Husher • 3 years ago

I'm curious what you thought of Eric Ferraiuolo's "Advocatus Diaboli" presentation.

https://speakerdeck.com/u/eric...

• Reply • Share ›

Brad Williams • 3 years ago

FWIW, we recently transliterated our app from Ember.js to AngularJS, and except for moving some code out of the Ember.js router state machine out into the controllers, there is almost no difference in usage and patterns. Sure there are "philosophical differences"

is almost no difference in usage and patterns. Sure there are "philosophical differences such as how the framework has you template in and around HTML elements, but in practice it's a distinction without a difference for the developer who needs to get user stories to done-done.

Reply • Share ›

Gersart • 3 years ago

Thank you for the post. Very clear and more useful than many with the breakdowns of info. I especially like the suggestion to try the various systems before committing to one.

As for those who post "you forgot this one" or "what about blah", they can't play if they don't participate. I think the prelude as to why and how was very consise, even if your favorite wasn't included. This is far more useful than a "umpteen JavaScript blah-de-blahs that you should know about" blog post...

Reply • Share ›

Ashish Jindal • 3 years ago

Agree with Diego Ferreiro.
Y can't find Mojito in this? :)

Because it takes away .Net from the server side? ;)

On a serious note... I think Mojito is definitely a noteworthy and serious competitor to any and all of these frameworks, specially when you start implementing the web-server stack for any application, and that too you need a great framework for developing across devices.

What do you think?

Reply • Share ›

Tom Geiger • 3 years ago

Hi Steven... GREAT POST, I've been enjoying all the great information your have been publishing, especially KnockoutJS. .. just an FYI on this post.. your AngularJS links are broken...
Tom G

Reply • Share ›

darren hurst • 3 years ago

I agree but check this out
https://github.com/Geekdad/BDS...

It brings a framework like approach to the backbone.js library.

Reply • Share ›

ara.t.howard • 3 years ago

"It's no longer good enough to build web apps around full page loads and then "progressively enhance" them to behave more dynamically. Building apps which are fast, responsive and modern require you to completely rethink your approach."

... unless you happen to be github or 37signals, in which case you can easily build apps and progressively enhance to be fast and responsive ....

+1 on the 'completely rethink' bit though. it turns out that's the actual key to making an app fast and responsive - not the framework chosen.

Reply • Share ›

Michael Lang • 3 years ago

I've been working on a library the past year, jquery-auto-async. I posted about it recently. Check it out.

http://candordeveloper.com/201...

The article uses Microsoft MVC for the backend implementation, but any backend would work with the client side code.

• Reply • Share ›

Load more comments

ALSO ON **STEVE SANDERSON'S BLOG** WHAT'S THIS?

Scaffolding Actions and Unit Tests with MvcScaffolding

20 comments • 2 months ago

Avat **Nathan** — So good ! Thanks for your hard work.

Deleporter: Cross-Process Code Injection for ASP.NET

16 comments • 2 months ago

Avat **Martin Schinz** — Hey Steven, love the project, are you still active on it? Github looks very quiet, so I was wondering if …

Knockout v2.3.0 released; v3.0.0 beta available

52 comments • 2 months ago

Avat **Godard** — cant keep up with all these releases! Keep em coming though.

Knockout 2.2.0 released

32 comments • 2 months ago

Avat **david** — Thanks , I've just been searching for info about this subject for ages and yours is the greatest I have came upon …

Subscribe Add Disqus to your site Add Disqus Add Privacy

READ NEXT

Node.js development with WebMatrix 2 + Express (Part 3)

This is the **third in a three-part series** of videos about building a mobile web app with Node.js, Express, and WebMatrix:

Published **Jul 11, 2012**