13. CHOOSING A JAVASCRIPT MVC FRAMEWORK

Current frameworks

ANGULARJS by Google

Knockout.

METEOR

ember

AMPLIFYJS

canjs

BACKBONE.JS

Spine

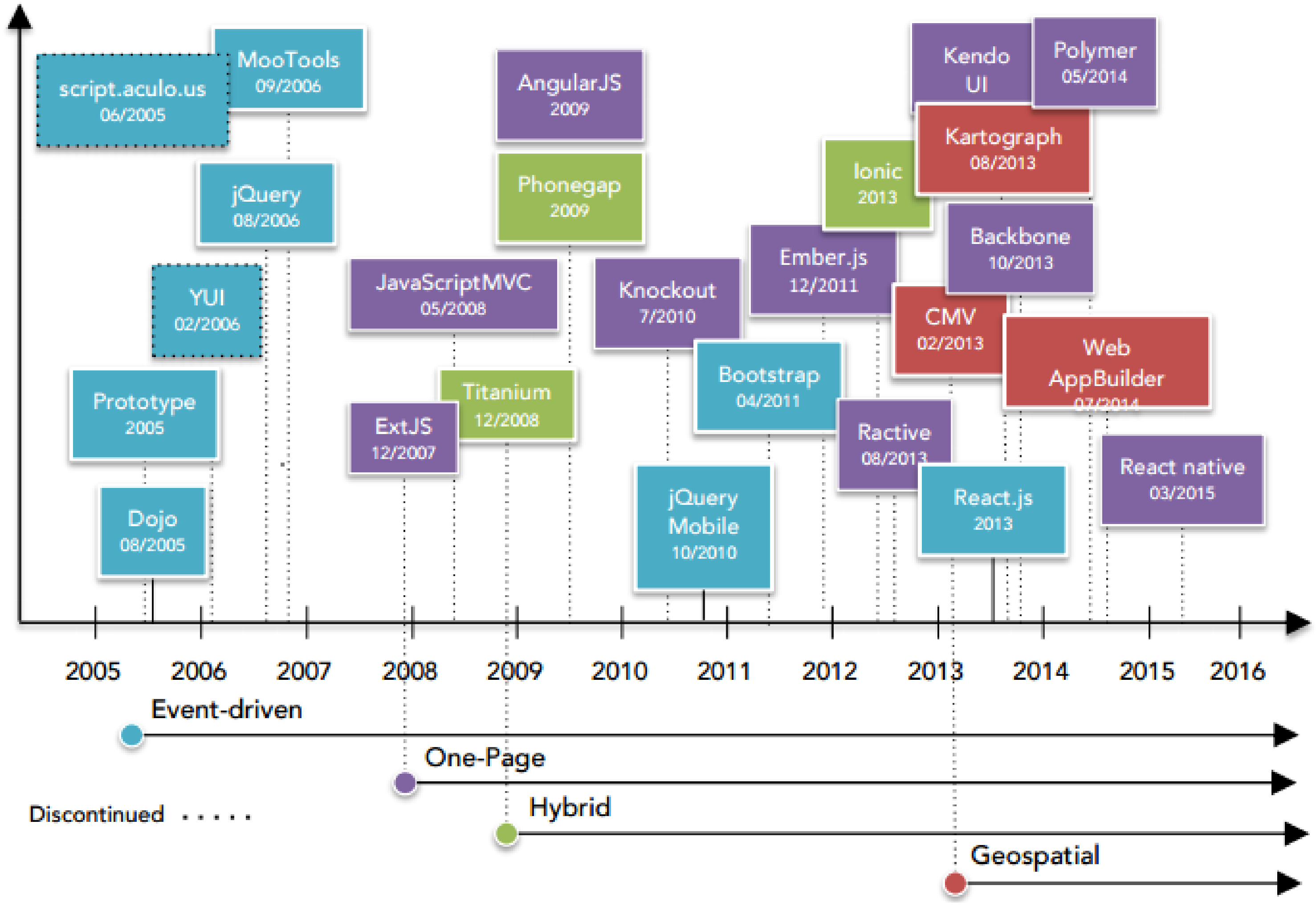
batman.js

http://todomvc.com/

JAVASCRIPT FRAMEWORKS

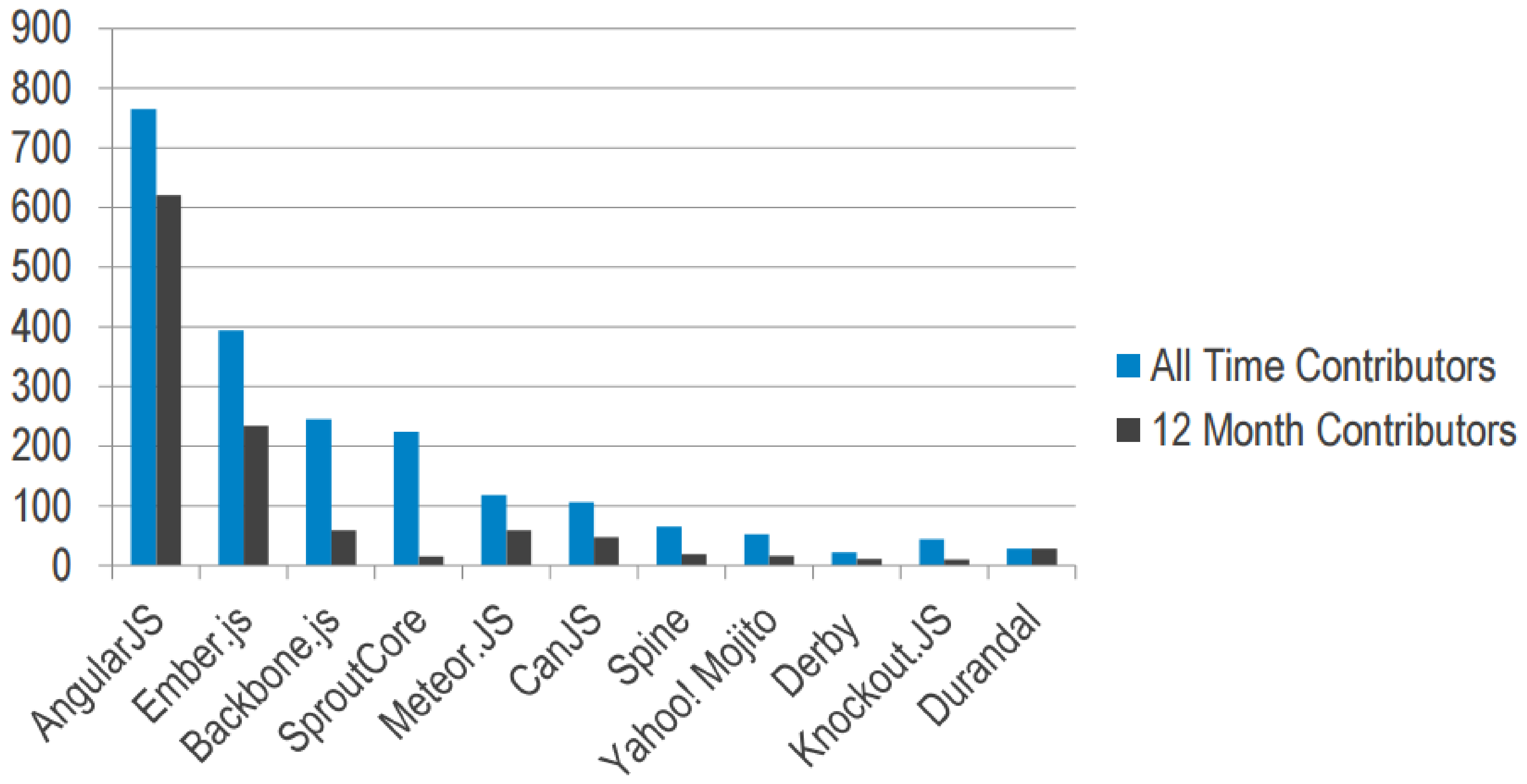
- Front-End MVC Frameworks
  - Angular, Backbone, Ember, Knockout Sproutcore, Spine, CanJS, Flight
- Full Stack Frameworks
  - Meteor, Mojito, Derby
- Single Page App frameworks
  - DurandalJS, batman.js
- Non-framework implementations
  - Vanilla JS , jQuery
- Real-time
  - Meteor, SocketStream, Firebase+AngularJS
- Node.js
  - Express+gcloud-node

JavaScript Frameworks & Libraries : Timeline



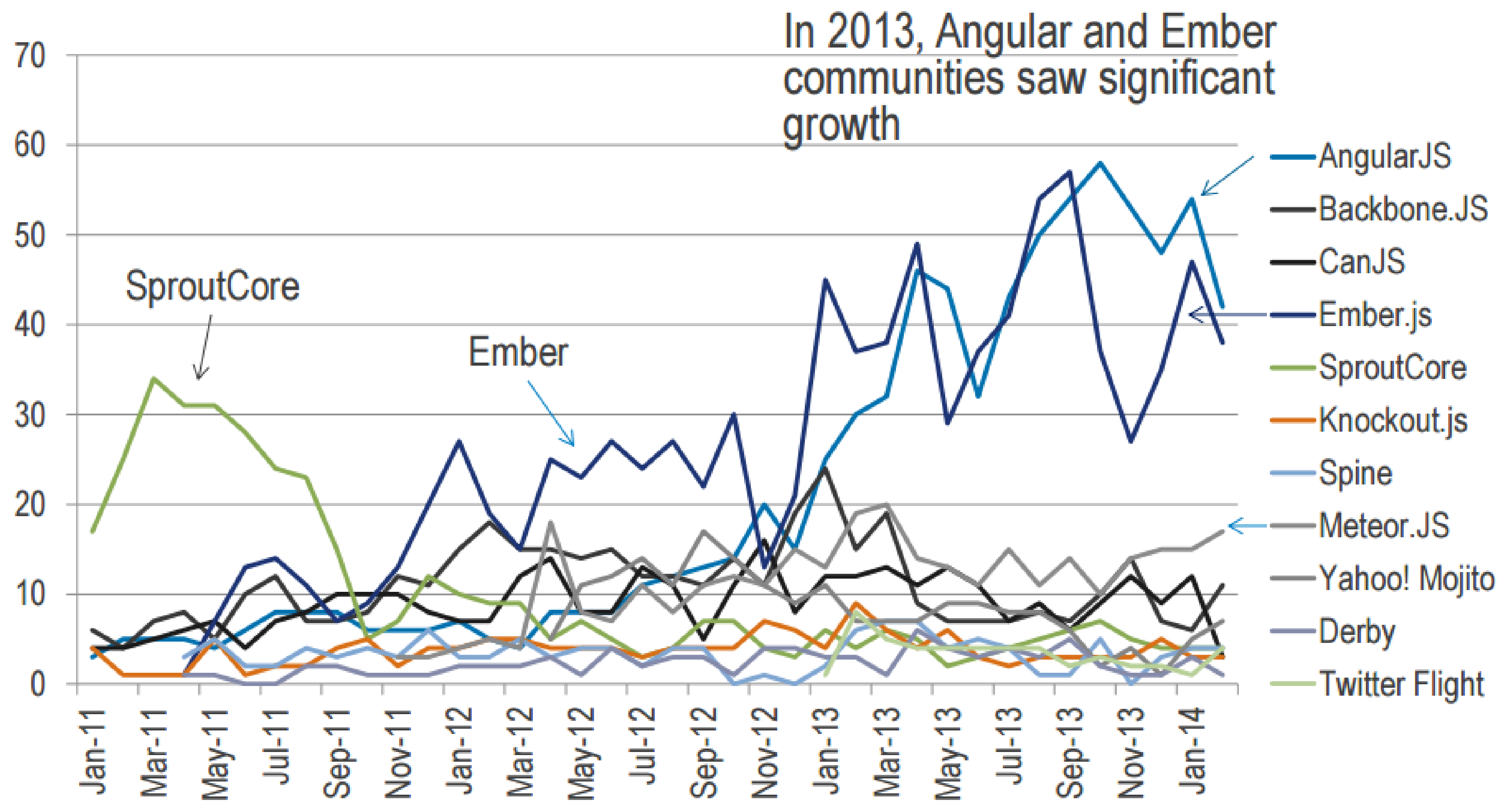
Source:blackduck 2014

Contributors



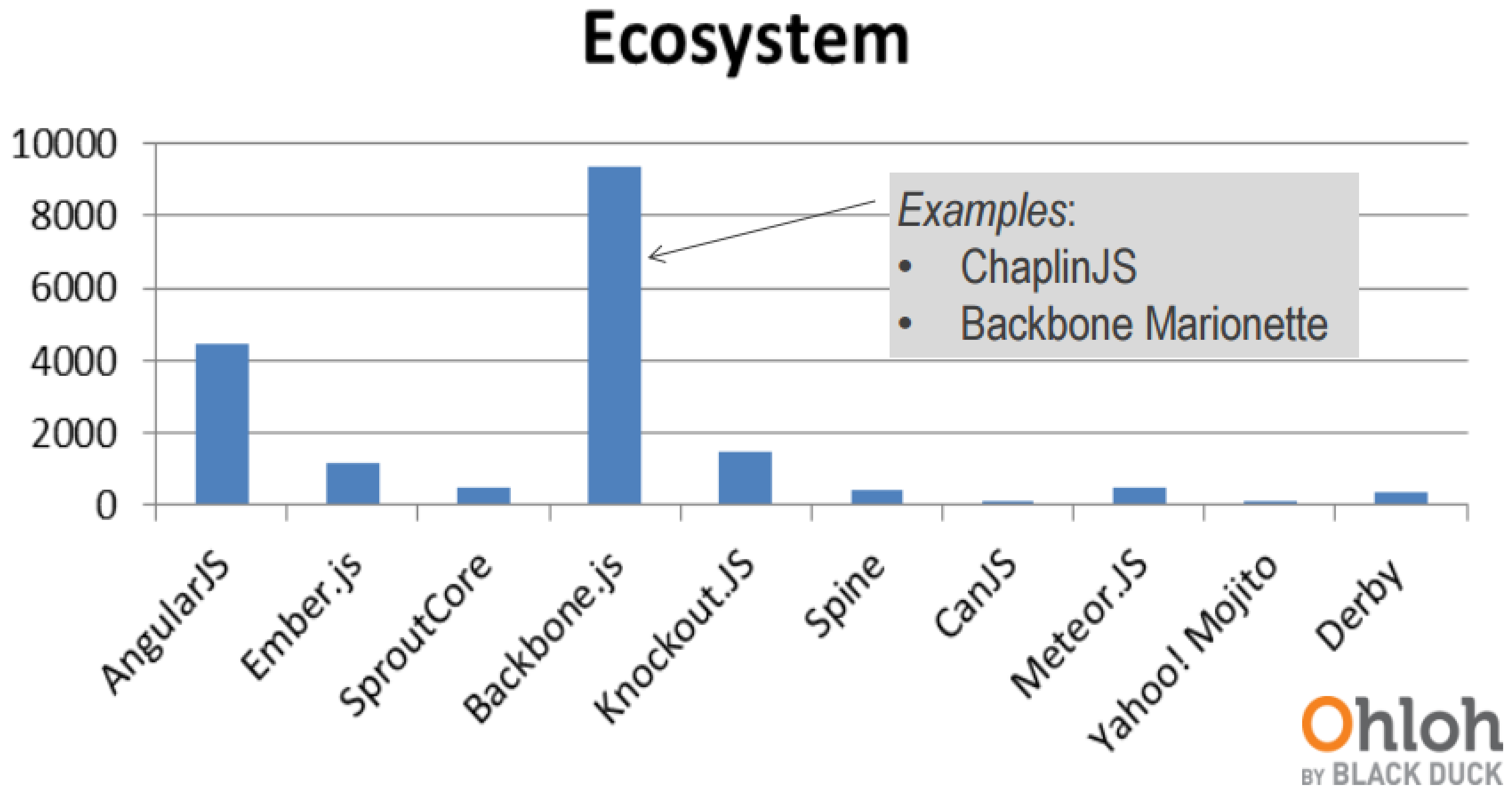
Source:blackduck 2014

Monthly Contributors Trend



Source:blackduck 2014

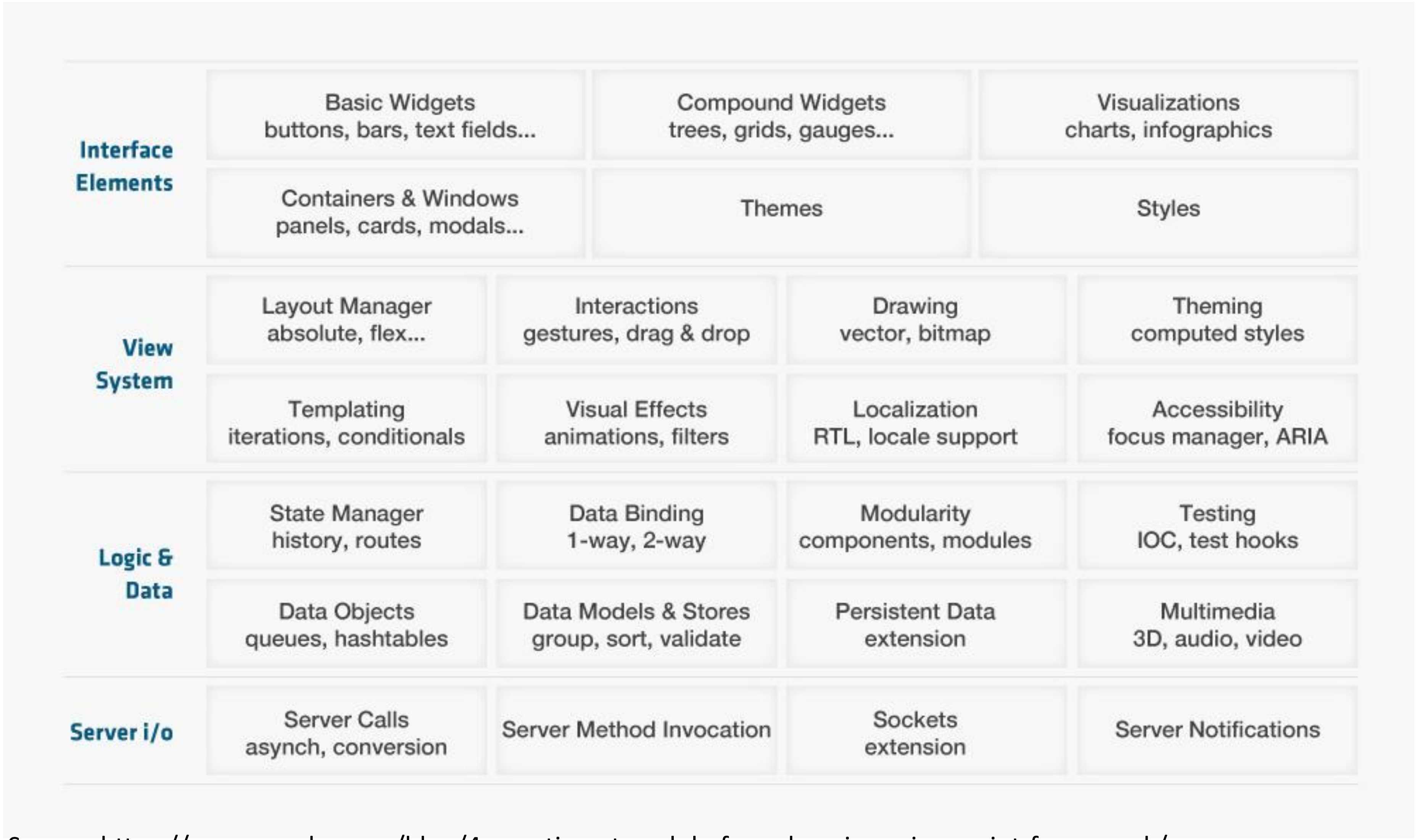
Ecosystem –Related projects



Blackduck data Insights

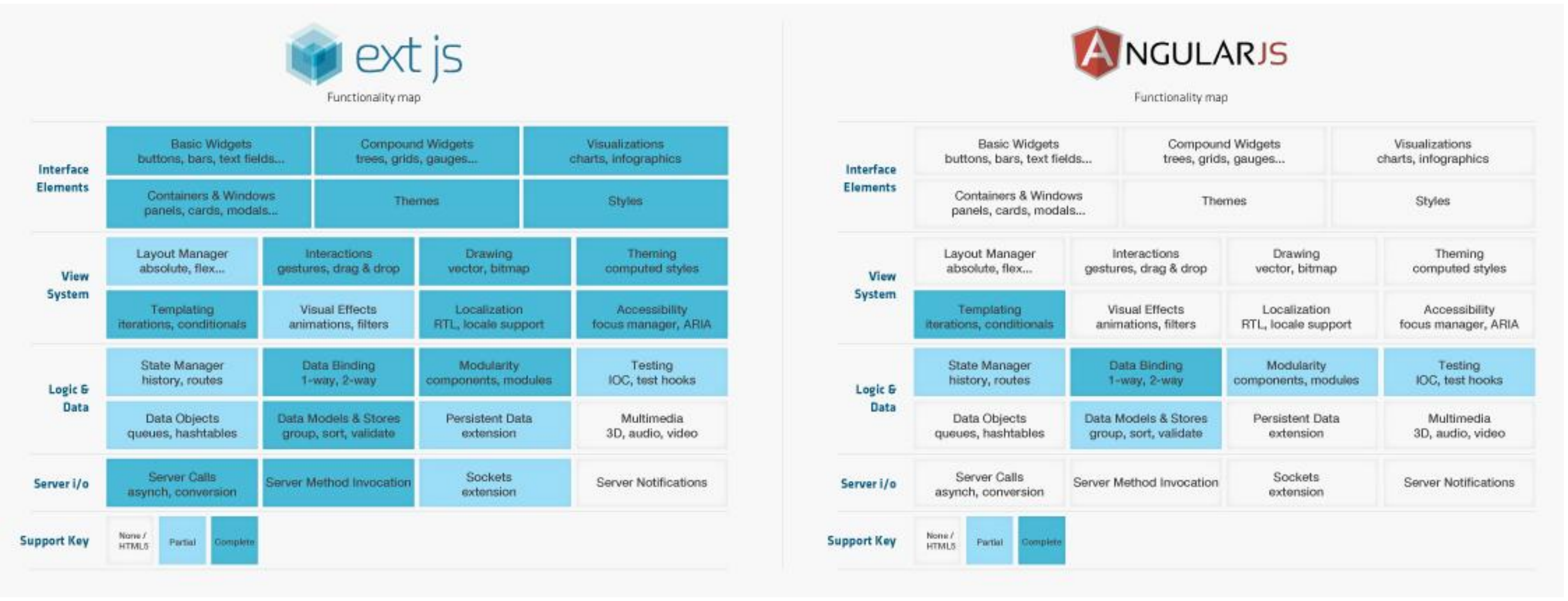
- **AngularJS** and **Ember** appear to be the fastest growing frameworks
- Lesser talked about full-stack Meteor is gaining momentum
- The strong ecosystem surrounding **Backbone** demonstrates both the popularity and commitment to this approach.

Matrix showing different features required for enterprise application development



Interface Elements	Basic Widgets buttons, bars, text fields...	Compound Widgets trees, grids, gauges...	Visualizations charts, infographics	
	Containers & Windows panels, cards, modals...	Themes	Styles	
View System	Layout Manager absolute, flex...	Interactions gestures, drag & drop	Drawing vector, bitmap	Theming computed styles
	Templating iterations, conditionals	Visual Effects animations, filters	Localization RTL, locale support	Accessibility focus manager, ARIA
Logic & Data	State Manager history, routes	Data Binding 1-way, 2-way	Modularity components, modules	Testing IOC, test hooks
	Data Objects queues, hashtables	Data Models & Stores group, sort, validate	Persistent Data extension	Multimedia 3D, audio, video
Server i/o	Server Calls asynch, conversion	Server Method Invocation	Sockets extension	Server Notifications

Source: https://www.sencha.com/blog/4-questions-to-ask-before-choosing-a-javascript-framework/



ext js

Functionality map

Interface Elements	■ Basic Widgets buttons, bars, text fields...	■ Compound Widgets trees, grids, gauges...	■ Visualizations charts, infographics	
	■ Containers & Windows panels, cards, modals...	■ Themes	■ Styles	
View System	▨ Layout Manager absolute, flex...	■ Interactions gestures, drag & drop	■ Drawing vector, bitmap	■ Theming computed styles
	■ Templating iterations, conditionals	▨ Visual Effects animations, filters	■ Localization RTL, locale support	■ Accessibility focus manager, ARIA
Logic & Data	▨ State Manager history, routes	■ Data Binding 1-way, 2-way	■ Modularity components, modules	▨ Testing IOC, test hooks
	▨ Data Objects queues, hashtables	■ Data Models & Stores group, sort, validate	▨ Persistent Data extension	⬜ Multimedia 3D, audio, video
Server i/o	■ Server Calls asynch, conversion	■ Server Method Invocation	▨ Sockets extension	⬜ Server Notifications
Support Key	⬜ None / HTML5	▨ Partial	■ Complete	

ANGULARJS

Functionality map

Interface Elements	⬜ Basic Widgets buttons, bars, text fields...	⬜ Compound Widgets trees, grids, gauges...	⬜ Visualizations charts, infographics	
	⬜ Containers & Windows panels, cards, modals...	⬜ Themes	⬜ Styles	
View System	⬜ Layout Manager absolute, flex...	⬜ Interactions gestures, drag & drop	⬜ Drawing vector, bitmap	⬜ Theming computed styles
	■ Templating iterations, conditionals	⬜ Visual Effects animations, filters	⬜ Localization RTL, locale support	⬜ Accessibility focus manager, ARIA
Logic & Data	▨ State Manager history, routes	■ Data Binding 1-way, 2-way	▨ Modularity components, modules	▨ Testing IOC, test hooks
	⬜ Data Objects queues, hashtables	▨ Data Models & Stores group, sort, validate	⬜ Persistent Data extension	⬜ Multimedia 3D, audio, video
Server i/o	⬜ Server Calls asynch, conversion	⬜ Server Method Invocation	⬜ Sockets extension	⬜ Server Notifications
Support Key	⬜ None / HTML5	▨ Partial	■ Complete	

Source: https://www.sencha.com/blog/4-questions-to-ask-before-choosing-a-javascript-framework/

Criteria for Informed Choice (MV*)

- Common design goal
  - Completely transparent/explicit code – clarity with the cost of being more verbose, and
  - Convention over configuration – more power with less code, but in some cases can appear rather magical to the unfamiliar user

Criteria for Informed Choice (MV*)

- Who will be the future maintainer of the application? Will they be familiar with your chosen framework?
- What productivity features do you want? E.g. do you want to use 2-way data binding? Are you happy for the framework to automatically create objects in the background unless you want to customise them?
- Do you prefer code to be as transparent and explicit as possibly at the cost of it being more verbose? Or are you happy to conform with conventions in order to do more with less code?
- Are you planning on specialising in the framework and/or investing in training your team?

Criteria for Informed Choice (MV*)

Routing

single page applications need the ability to present several screens to the user without needing to completely reload the page each time.

- Is your application transferring enough data on each full page load that making it a single-page app would bring significant benefits? E.g. more responsiveness in terms of latency or reduced data transfer costs.
- Does the framework have built-in support for routing?
- If you decide to use a single-page architecture, do your users need the ability to share links to sub-screens within the interface? If so look for a framework that routes based on the URL.
- Does the framework support routing as an add-on (e.g. Angular)?

Criteria for Informed Choice (MV*)

Templating

Some frameworks have been built to use a specific templating technology exclusively, others allow a free choice. Tighter integration with a specific framework is possibly about enabling more interesting features e.g. 2-way-data binding and custom components, but this comes at the cost of choice.

- Do you prefer to use templates or embed HTML markup within your JavaScript code?
- Does the framework under consideration force you to use a specific templating solution or are you free to use any of your choice? If the former is true, are there enough advantages that it's worth putting up with it?

Criteria for Informed Choice (MV*)

Dependencies

Some frameworks have dependencies on third party libraries, which pose conflict with any libraries a user wishes to use.

- Does the framework depend on any 3rd party libraries (e.g. jQuery)? If so, does this present a conflict with any libraries you want to use (e.g. Zepto) or force you to use specific versions?

Criteria for Informed Choice (MV*)

Module Loaders

Asynchronous Module Definition (AMD) comes from wanting a module format that was enables us to implicitly load dependencies asynchronously, some times we are required to manually load.

- If you are using an AMD module loader, such as RequireJS, is the framework compatible with it?

- Ember does not like the AMD approach to module load
- Angular has its own built-in solution for modularity and dependency resolution, but can still be used with an AMD style loader if desired.

Criteria for Informed Choice (MV*)

Testability

How easy is it to test the code your produce within the framework?

- Do you need to do a lot of complicated set-up to isolate the unit you want to test?
- Is it obvious how to get at and test the features of the unit under test? Or do you need to learn a new API?
- Are you locked into using certain test frameworks? Or at least, if there is a recommended framework, what are the costs/obstacles likely to be if you try to use an alternative? E.g. choosing to test an Ember app with Mocha instead of QUnit.
- Is it easy to write end-to-end system tests (using Karma or similar)?

Criteria for Informed Choice (MV*)

Remote API Integration

For basic AJAX calls most of the frameworks either provide an HTTP client service or you can choose your own, e.g. jQuery.getJSON or similar

- Does the framework provide its own HTTP client service or do you have to choose your own?
- Does the framework support a model abstraction with REST integration? If so, what schema does it expect the JSON replies to use?
- Do you have control over the design of the external API or will you have to write adaptors? If the latter, how does the framework support custom API adapters?
- Will multiple users be able to edit your application data? If so, how and when should shared data get reloaded from the server? Will it be cached by the framework or explicitly by your own code? When will the cache be invalidated and resynced with the server?

Criteria for Informed Choice (MV*)

Documentation

Effective and Good Documentation with developer support

- Is the available documentation up to date?
- For frameworks with unstable APIs (e.g. Ember-Data at the time of writing), is the documentation clearly linked to software release version?
- Is it easy to retrieve documentation for previous releases? The framework may advance quickly, but it may be difficult for you to upgrade if you've written a sizeable portion of your application already.
- Does the reference material contain relevant examples for both typical and advanced use-cases?
- Is there a good mix of material e.g. articles, tutorials, videos, developer guides, API references?
- Is there a healthy community discussing / supporting / writing about it?

Criteria for Informed Choice (MV*)

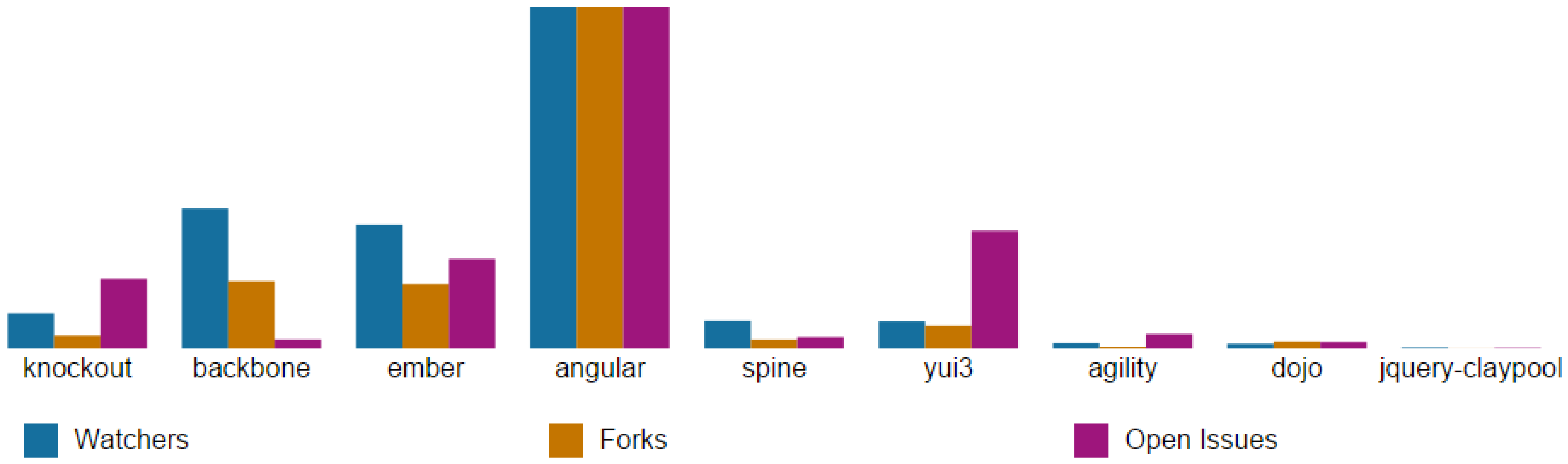
Browser Compatibility

MV* Frameworks tend to be forward-looking. Compatible with modern browsers and HTTP1.1 and HTTP 2.0

- What are the minimum browser versions supported by the framework and how to they compare with the versions you need for your project?
- Will you be able to gracefully degrade certain features in order to support less capable browsers?

Choice of JavaScript MVC Framework

snapshot on 26/01/2016



Source: http://js-toolbox.org/category/frameworks

When do you need a Javascript MV* Framework

- For building single-page applications for simple or complex ui or to reduce number for http requests required for new views.
- Opinionated way to build a application framework.

Criteria for selecting a framework

Source: Addy Osmani

- What is the framework really capable of?
- Has the framework been proved in production?
- Is the framework mature?
- Is the framework flexible or opinionated?
- Have you really played with e framework?
- Does the framework have a comprehensive set of documentation?
- What is the total size of the framework, factoring in minification, gzipping and any modular building that it supports?
- Have you reviewed the community around the framework?

https://www.smashingmagazine.com/2012/07/journey-through-the-javascript-mvc-jungle/

CHOOSE RIGHT FRAMEWORK: WHEN TO USE WHAT

https://www.smashingmagazine.com/2012/07/journey-through-the-javascript-mvc-jungle/

- Minimalistic solution for separation of concerns
- Support for persistence layer, RESTfl sync, models, views (with controllers), event-driven communication, templating and routing
- It should be imperative, allowing one to update the view when a model changes
- Architectural definitions are left to developer
- Building simple to complex applications

Use Backbone.js

- Tries to tackle desktop-level application development for the web.
- Opinionated, modular, support a variation of MVC, avoid the need to wire everything in an application, support persistence, computed properties and have auto-updating (live) templates.
- Support proper state management rather than the manual routing solution
- Templating and scaffolding tools
- Extensive documentation

Use Ember.js

- Lightweight Framework which supports live-binding templates, routing, integration with major libraries (like Jquery and Dojo) and is optimized for performance.
- Supports MVC
- Supports Complex Application frameworks

Use CanJS

- Excellent base for building large scale applications
- Support mature widget infrastructure, modules which support lazy-loading and can be asynchronous, simple integration with CDNs, wide array of widget modules for graphics, charting and grids
- Support for Internationalization (i18n, i10n)
- Support for complex architectures using OOP, MVC

Use Dojo

- Benefits YUI extension infrastructure.
- MVC based with routing, view transitions
- Support for Complex architectures using widgets/components
- Scaffolding tools (yuiproject)

Use YUI

- A framework that values asynchronous interfaces and lack any dependencies
- Opinionated but flexible on how to build applications
- Provides bare-bones for MVC, events, routing while still being tiny.
- Optimized for use with CoffeeScript
- Comprehensive documentation

Use Spine

- A framework that makes it easy to build complex dynamic UIs with a clean underlying data model and declarative bindings.
- Automatic update my UI on model changes using two-way data bindings and support for dependency tracking of model data
- Ability to integrating with any existing frameworks
- Built-in templating and extensibility

Use KnockoutJS

- Just build websites, with out great deal of code involved
- Abstract away browser differences so focus on I
- Easy binding of events, interaction with remote services
- Extensibility and huge plugin community

Use JQuery

- A declarative framework that uses VIEW to derive behaviour
- Focus on achieving through HTML extensibility, components that specify application intentions.
- Test-driven development, URL management through routing, separation of concerns through a variation of MVC.
- Provides a HTML compiler for creatin your own DSL in HTML.
- Supports Web Components and Object.Observe
- Scaffolding tools available

Use AngularJS