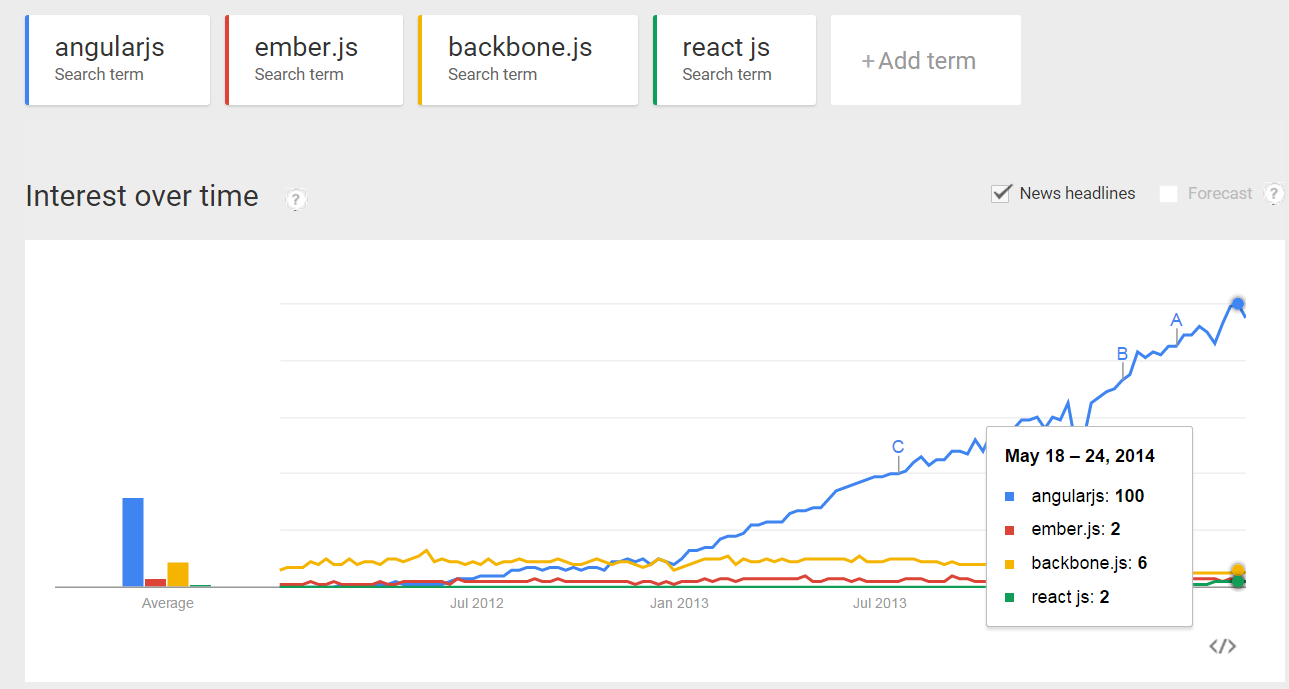
Javascript Framework Evaluation

We are evaluating a few javascript frameworks to support DCGS-N program. We will evaluate Angular JS, Backbone, ReactJS, KnockoutJS, EmberJS for ease of use, learning curve required and popularity. In addition to JS frameworks, we will evaluate a few JS UI libraries: bootstrap UI, Angular UI, Jquery UI.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Maturity | True MV\* | Popularity | Opinionated | Lightweight | Ease of learning | Good ecosystem |
| ReactJS |  |  |  |  | ✓ | ✓ |  |
| AngularJS | ✓ | ✓ | ✓ | ✓ |  |  |  |
| EmberJS | ✓ | ✓ |  | ✓ |  |  |  |
| Knockout | ✓ |  |  |  | ✓ | ✓ |  |
| Backbone | ✓ | ✓ |  |  |  |  |  |

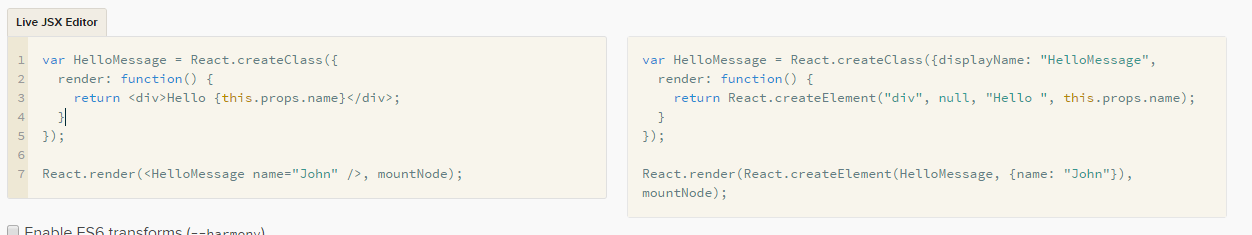
# Javascript Frameworks

## ReactJS

Overall it’s a new light weight View “framework” and gaining traction.

**Pros:**

1. Un-opinionated “framework”
2. Fast render time (60 fps). Focus only on rendering and event handing.
3. Use a JSX Transformer/compiler to compile ReactJS syntax to native JS and HTML. In the pic below. Left side is the JSX syntax. Right hand is the translation into Javascript/HTML



1. Creation of reusable UI components is simple.
2. React and Knockout are similar.
3. Simple learning curve.
4. BSD license

**Cons:**

1. Fairly new. Facebook release it in 2013. Not many best practices available to follow.
2. One 1 or 2 books available.
3. Provide only the View of the MV\*. Developers have to roll their own data/model layer.
4. Not truly MVC framework.
5. No “React UI” library as in Angular UI or JQuery UI.
6. React features a one-way data binding model. Ember or AngularJS uses a two-way data binding model.
7. No built-in routing.
8. Does not support externalizing HTML templating when creating reuse code.

## Backbone

**Pros:**

1. Un-opinionated framework.
2. Mature framework.
3. Support routing.
4. Developers have full control of the DOM.
5. Short learning curve.
6. Creation of reusable controls via extension of Backbone.View.extend.
7. MIT license

**Cons:**

1. No true automatic 2-way binding. Must write code to read/write from/to DOM.
2. Developers have full control of the DOM!
3. Developers have the freedom to pollute the View with “model” code which is difficult to do with Angular.
4. Must roll-your-own dependency injection.
5. No “Backbone UI” library.

## Angular JS

**Pros:**

1. Opinionated framework. Full blown end-to-end framework.
2. Has templating, routing and controllers just as Ember JS.
3. “hottest” framework at the moment. Highest adoption rate at the moment.
4. Forces best practice on developers via dependency injection, true separation of concern between View, Controller and Services.
5. Promote unit testing.
6. It has good ecosystem supporting testing (Karma & Protractor), UI libraries (both commercial and open source), books, best practice.
7. Promote reusable controls via directives.
8. V2.0 is in near future and support TypeScript.
9. MIT license

**Cons:**

1. Opinionated framework
2. Slightly steep initial learning curve.

## Ember JS

**Pros:**

1. Opinionated “framework”
2. MIT license
3. Has templating, routing and controllers just as Angular JS.
4. Prefer Convention over configuration.
5. More MVVM than MVC. Controller is the VM.

**Cons:**

1. Depends on Jquery and Handlebar.
2. Need to learn Handler templating syntax.

## Knockout JS

**Pros:**

1. Opinionated “framework”
2. Mature light-weight and MVVM binding library.
3. Simple learning curve
4. Provides 2-way binding between the View and Model.
5. Good support for templating for reuse HTML control.
6. MIT license

**Cons:**

1. Does not dictate how you write the model layer.
2. No “Knockout JS” UI library. Commercial UI libraries supporting KO such as JQWidget and Telericks.
3. KO should be used in conjunction with other libraries to create full blown SPA.