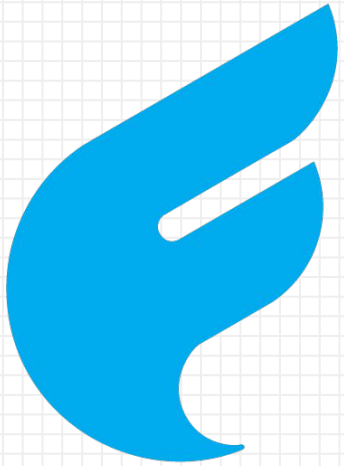
Component-Based JavaScript using Flight.js

HELLO!

I am Joel Byler

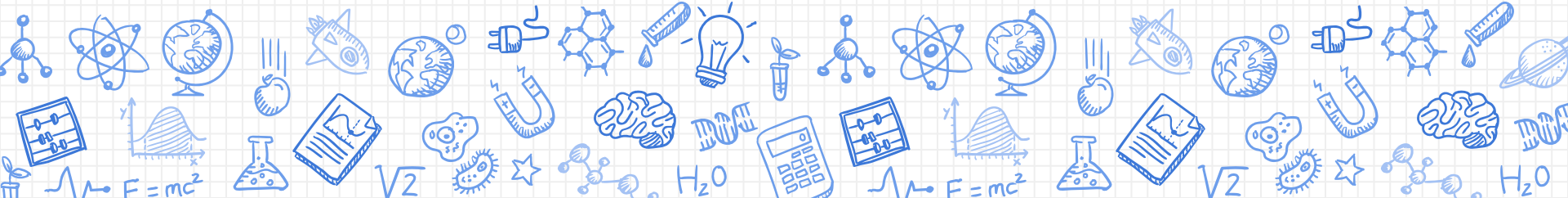
@joelbyler

Polyglot developer transitioning into a full-time Ruby on Rails developer at covermymeds®



Flight.js

"An event-driven web framework, from Twitter"



Just to clarify, this talk is **not** about

My story

Application

- started off as a simple CRUD app
- desire to improve UI
- multiple UX teams
- designs which are rapidly changing based on user feedback

My story

Team

- strong backend developers
- good with jQuery
- not strong front-end js frameworks
- small

```
<div class="example-form">
  <%= form_for(@example) do |f| %>\
    <%= f.label :name %>
    <%= f.text_field :name %>
    <br/>

    <%= f.label :email %>
    <%= f.text_field :email %>
    <br/>

    <%= f.label :gender %>
    <%= f.radio_button :gender, :male %> Male
    <%= f.radio_button :gender, :female %> Female
    <br />

    <%= f.check_box :agree %> <%= f.label :agree %>
    <br />

    <%= f.submit 'Save', class: 'btn' %>
    <%= link_to 'Cancel', examples_path %>
  <% end %>
```

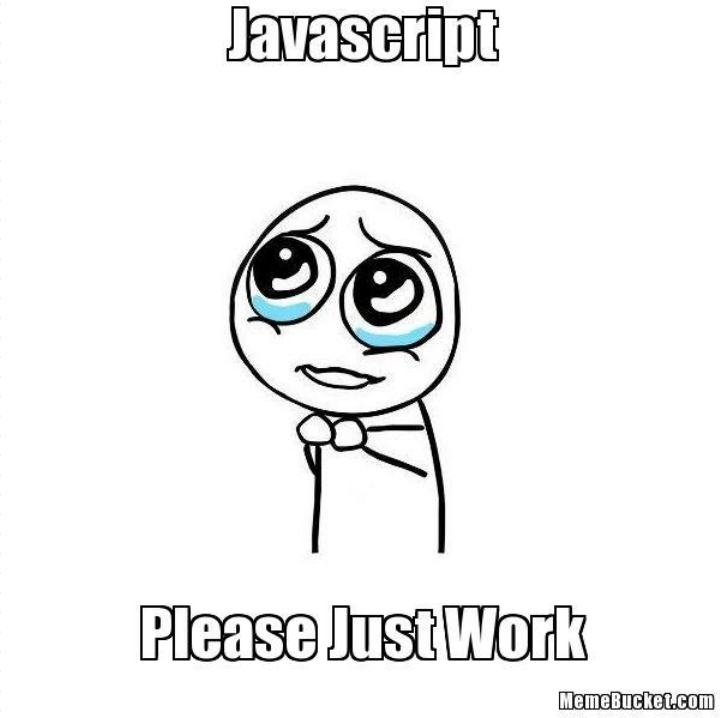
MS MVC Might look something like this

```
@using (Html.BeginForm("Update", "Example"))
{
  @Html.LabelFor(model => model.Name)
  @Html.EditorFor(model => model.Name)
  <br/>
  @Html.LabelFor(model => model.Email)
  @Html.EditorFor(model => model.Email)
  <br/>
  <input type="submit" value="Save" />
}
```

What we need

- a way to build UI components in js
- retrofit onto existing CRUD pages
- well supported by jasmine or other testing framework
- small learning curve

And not this for our client



https://files.gitter.im/FreeCodeCamp/FreeCodeCamp/aVtk/Javascript-535.png

Ok, no problem. How about ...

Angular?

would work great, but client would have to add a bunch of ng- whatever attributes all over their existing forms and web pages.

```
<div ng-controller="ExampleController">
  <%= form_for(@example) do |f| %>
    <%= f.label :name %>
    <%= f.text_field :name, "ng-model" => "example.name" %>
    <br/>

    <%= f.label :email %>
    <%= f.text_field :email, "ng-model" => "example.email" %>
    <br/>

    <%= f.label :gender %>
    <%= f.radio_button :gender, :male, "ng-model" => "example.gender" %> Male
    <%= f.radio_button :gender, :female, "ng-model" => "example.gender" %> Female
    <br />

    <%= f.check_box :agree, "ng-model" => "example.agree" %> <%= f.label :agree %>
    <br />

    <%= f.submit 'Save', class: 'btn', "ng-click" => "update(example)" %>
    <%= link_to 'Cancel', examples_path %>
  <% end %>
```

Ok, no problem. How about ...

Angular?

would work great, but client would have to add a bunch of ng-whatever attributes all over their existing forms and web pages.

Backbone?

could probably do the trick, but it's kind of big and way too complicated for this application and the team to maintain

http://www.troll.me/2011/09/23/xzibit-yo-dawg/yo-dawg-i-heard-you-like-mvc-so-i-put-a-backbone-js-in-your-ror-so-you-can-mvc-while-you-mvc/

Ok, no problem. How about ...

Angular?

would work great, but client would have to add a bunch of ng-whatever attributes all over their existing forms and web pages.

Backbone?

could probably do the trick, but it's kind of big and way too complicated for this application and the team to maintain.

React?

going the right direction but a larger learning curve and outside of the client's comfort zone.

http://blog.teamtreehouse.com/wp-content/uploads/2015/08/react-hype.jpg

So now what?

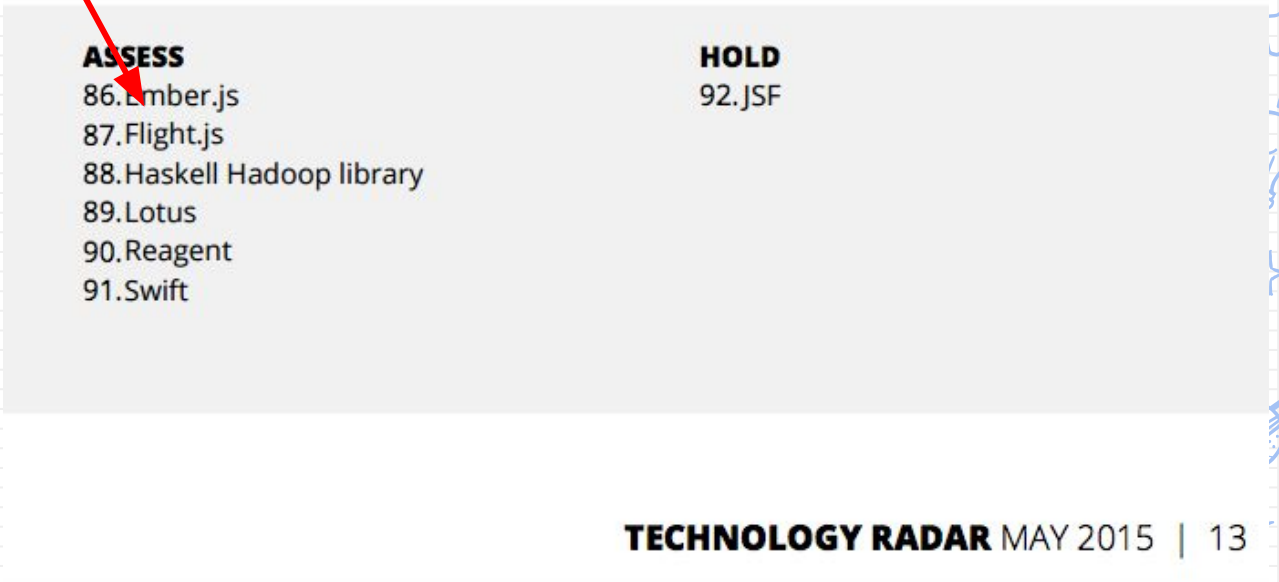
Oh, I remember I saw a mention of flight.js on ThoughtWorks' TechRadar

Flight?

super small footprint

really simple to teach

testable



ASSESS

86. Ember.js
87. Flight.js
88. Haskell Hadoop library
89. Lotus
90. Reagent
91. Swift

HOLD

92. JSF

From ThoughtWorks TechRadar, May 2015

In the crowded space of JavaScript frameworks, we want to highlight Flight.js (flightjs.github.io) as a **lightweight** framework to build components. Flight gets by without much magic when adding behavior to DOM nodes. Its LANGUAGES & FRAMEWORKS event-driven and component-based nature **promotes writing decoupled code**. This makes testing individual components comparatively easy. … We do like that it uses functional mixins for behavior, like **composition instead of inheritance**.

Flight.js ?

- Released v1.0 on January 30, 2013
- Latest release from Oct 6, 2015 v1.5.1
- v2.0 is under active development
- MIT License
- Uses jQuery, and prefers to use RequireJS (AMD)
- Plays well with other js libraries / frameworks
- Not MV* (not MVC or MVVM or whatever)

Demo

Want to see a demo?

Success

- able to make more complex UI changes
- quickly respond to user feedback
- reduced tech debt
- testable
- easy to learn

http://images.sodahead.com/polls/002550747/4953642218_success_kid_answer_1_xlarge.jpeg

What we will be looking at

- Components
- Attaching to DOM
- Events
- Mixins
- Advice
- Testing!

Flight – Component

- js object that has a reference to a DOM node
- can manipulate the DOM within its node
- listen to or trigger an event on its DOM node
- components are completely decoupled

The engineers at twitter, built flight.js to be a library that allows you to build small components that can be part of a larger system.

Flight – Component

basic building blocks of flight.js

- Attributes
- Initializer
- Functions (for behavior)

Flight – Component – Attributes

attributes are used by both components and mixins to declare values which can be shared across the functions within them.

can also be used to declare overridable default attributes or attributes which are required to be passed into a new component's attachTo() call (discussed later).

can also hold state of the component

Flight – Component – Attributes

```
var Widget = flight.component(function() {
  this.attributes({
    text: 'Lorem ipsum',
    menuItem: '.menuItem',
    items: []
  });
  ...
});
```

Flight – Component – Initializer

As the name implies, this function is called to initialize a new component. This is where we can create our event listeners and setup our component for user interaction.

Flight – Component – Initializer

```
var Widget = flight.component(function() {
  ...
  this.after('initialize', function() {
    this.on('click', this.activate);
  });
  ...
});
```

Flight – Component – Behavior

functions declare the behavior of this component.

because the component should actually do something interesting

Flight – Component – Behavior

```
var Widget = flight.component(function() {
  ...
  this.activate = function() {
    this.select('saveButton').show();
  };
  ...
});
```

Flight – Component – Attaching to the DOM

jQuery style selectors

each DOM element that matches this selector will initialize its own instance of the component.

does NOT return a reference to any instances of the component

Flight – Component – Attaching to the DOM

```
var Widget = flight.component(function() {
  ...
});

Widget.attachTo('.widget');
```

Flight – Component – Attaching to the DOM with overrides

```
var Widget = flight.component(function() {
  this.attributes({
    title: 'Lorem ipsum',
  });
});

Widget.attachTo('.widget', { title: 'CodeMash!' });
```

Flight – Component – Attaching to the DOM with overrides

```
var Widget = flight.component(function() {
  this.attributes({
    title: null, // <== req'd in attachTo if null
  });
});

Widget.attachTo('.widget', { title: 'CodeMash!' });
```

Put it all together

```
var Widget = flight.component(function() {
  this.attributes({
    menuItem: '.menuItem',
    saveButton: '#save'
  });
  this.activate = function() {
    this.select('saveButton').show();
  };
  this.after('initialize', function() {
    this.on('click', this.activate);
  });
});
Widget.attachTo('.widget');
```

Flight – Events

attached to a component using the ‘on’ function.

components may also unsubscribe to events using the ‘off’ function

attributes can be used to attach event handlers to DOM elements which are contained within the attached node

Flight – Events (continued)

they're just regular javascript events

this makes it very easy to integrate into existing / legacy applications !

Flight – Events – listen for an event

```
this.on('click', {
  menuItem: this.selectMenuItem,
  saveButton: this.saveAll
});
this.on('some_event', this.someCoolStuff);
this.on(document, 'msgs_recd', this.loadMessages);
```

Flight – Events – trigger for an event

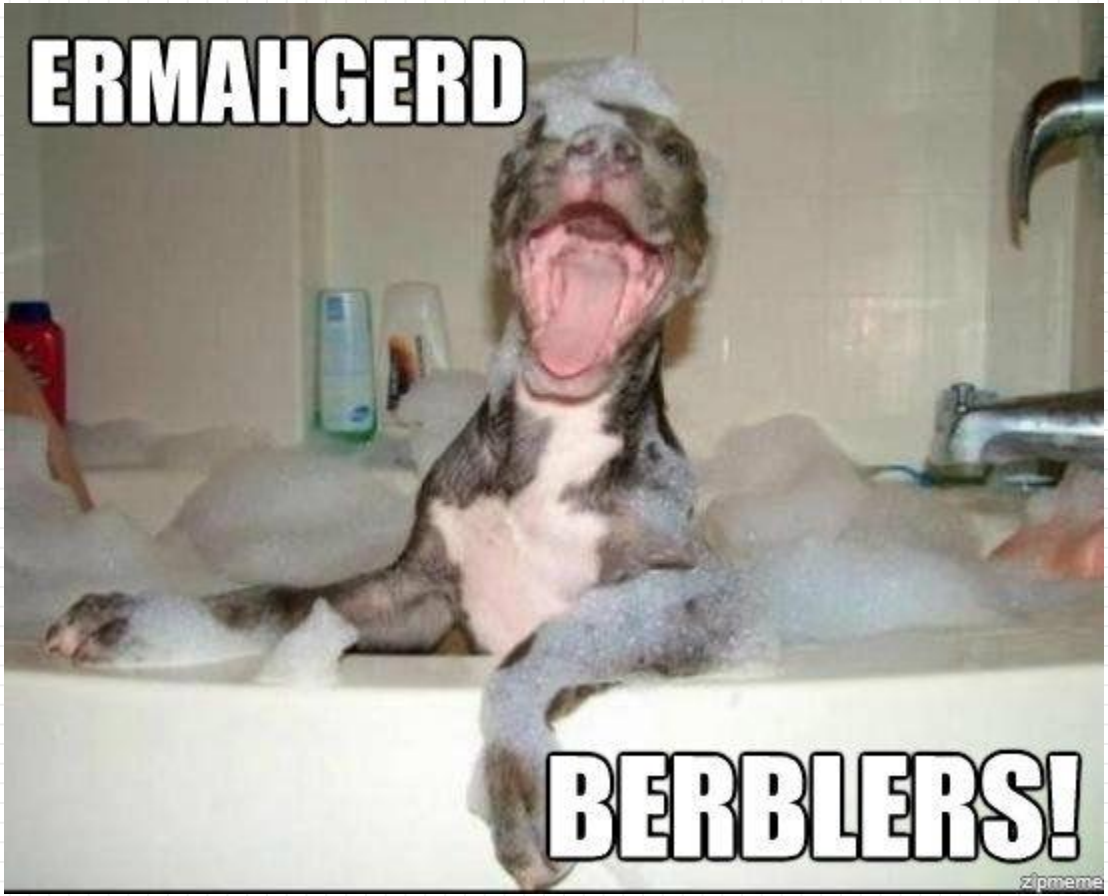
```
this.trigger('some_event');
this.trigger(document,
             'msgs_recd',
             { messages: response.messages });
```

Flight – Events – stop listening for an event

```
this.off('some_event');
```

Flight Events

```
var Widget = flight.component(function() {
  this.attributes({
    menuItem: '.menuItem',
    saveButton: '#save'
  });
  this.selectMenuItem = function(e) { ... };
  this.saveAll = function(e) { ... };
  this.after('initialize', function() {
    this.on('click', {
      menuItem: this.selectMenuItem,
      saveButton: this.saveAll
    });
  });
});
```

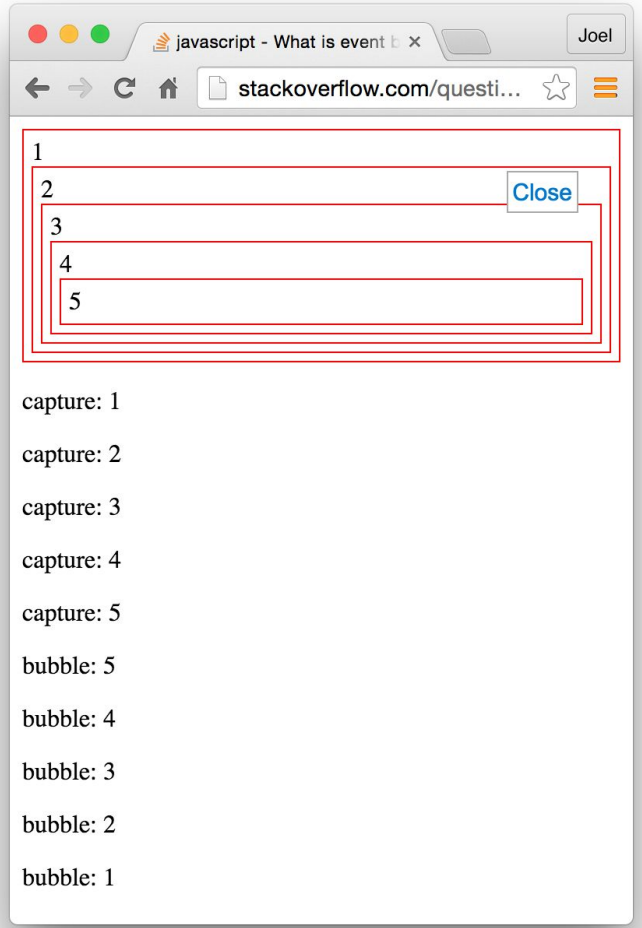


https://s-media-cache-ak0.pinimg.com/736x/74/a4/bd/74a4bdd87df2f65cce9adbfd3bba40df.jpg

Event bubbling

first, capture drills down from the outside in, until it meets the node that the event was triggered on

then bubble ripples outward from the node that the event was triggered on



http://stackoverflow.com/questions/4616694/what-is-event-bubbling-and-capturing

Flight – Mixins

mixins allow common functions to be reused to compose more interesting components

mixins can be composed of other mixins

Flight – Mixins

```
function withDropdown() {
  this.openDropdown = function() { ... };
  this.selectItem = function() { ... };
}

var Widget = flight.component(widget, withDropdown);

Widget.attachTo('.widget');
```

Flight – Advice

a predefined mixin which defines before, after, and around functions to hook new behavior to perform setup or teardown for your components

similar Aspect Oriented Programming? (AOP)

Flight – Advice

```
this.before('refreshSummary', function() {
    ...
});
this.after('refreshSummary', function() {
    ...
});
```

Flight – Advice (plus one that we've already looked at)

```
this.before('refreshSummary', function() {
  ...
});
this.after('refreshSummary', function() {
  ...
});
this.after('initialize', function() {
  this.on('click', this.activate);
});
```

Flight – Advice (continued)

alternatively, around will execute both before and after

note: you can not do BOTH

```
this.around('refreshSummary', function() {
  ...
});
```

Flight - Testing

flight components can be tested using jasmine

there is also a jasmine extension, named jasmine-flight

and for those who use mocha, there is mocha-flight

Flight – Testing (or just use standard jasmine)

```
describe("ExampleComponent", function() {
  beforeEach(function() {
    loadFixtures("example.html");
  });
  it("include name in json summary", function() {
    $('#example_name').val('joe');
    ExampleComponent.attachTo('.example-form');
    expect($(".example-summary").html()).toMatch('joe');
  });
});
```

Tips

if you need to stop event propagation:

```
event.stopPropagation()
```

try to keep ajax out of your components (hard to test), trigger and listen to events to get this type of functionality

for selectors, prefer classes over ids

Tips (continued)

debugging

DEBUG.enable(); DEBUG.events.logAll();

use care in naming your events

don't use something like 'dataReceived'

covermymeds®

be sure to stop by the booth!

Columbus

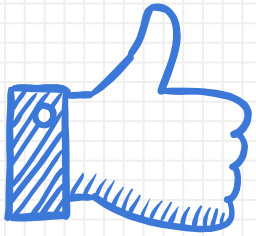
Cleveland

References (continued)

- Angus Croll on Twitter Flight and Mixins
  https://www.youtube.com/watch?v=ZDpB12LwFGE
- Jason Harwig, Altamira TC
  https://www.youtube.com/watch?v=F0h01zebb_E
  https://github.com/jharwig/flight-presentation
- Dan Webb at JSConfUS 2013
  https://www.youtube.com/watch?v=4WRmFp8jZjc

References

- Thoughworks Tech Radar (May 2015 edition)
  http://thght.works/10i0wzc
- The Twitter Flight Edge (Safari Books Online)
  http://bit.ly/1XXLoBQ
- Great data sets for dev examples / experiments
  https://github.com/fivethirtyeight/data



THANKS!

Any questions?

You can find me at

@joelbyler

joelbyler@gmail.com