
DEVNATION
April 13-17, 2014 San Francisco, California

Brian Leathem @brianleathem

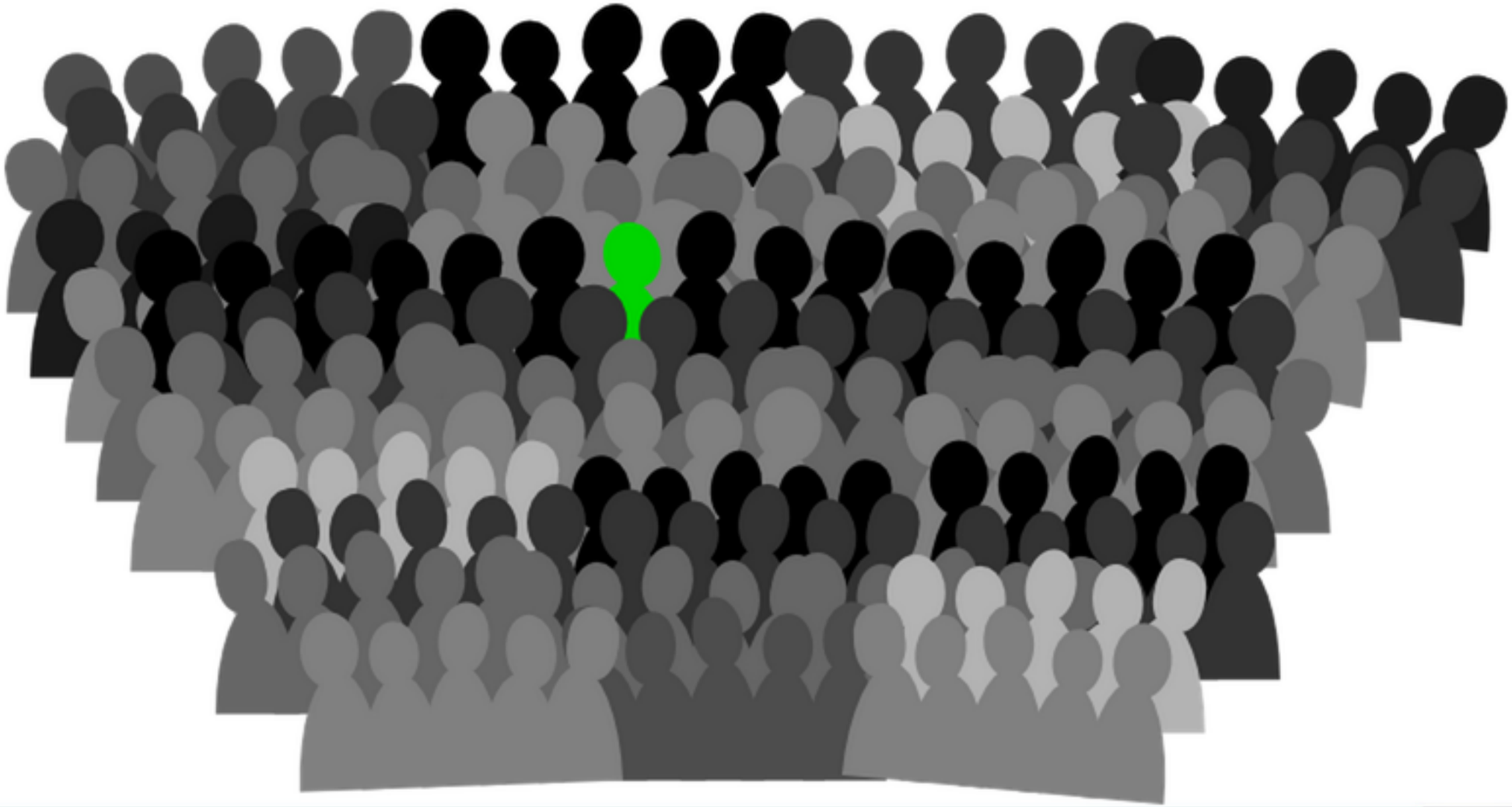
The plan

- jQuery Plugin review
- Widget factory advantages
- Use cases and examples

Who am I?

- Brian Leathem
- Software Engineer @ Red Hat
- RichFaces project lead (JSF)
- RichWidgets — http://richwidgets.io

Who are you?



Tapping into the jQuery ecosystem

jQuery

Consistent cross browser API for:

- DOM traversal
- HTML Manipulation
- Event handling
- Animation
- Ajax

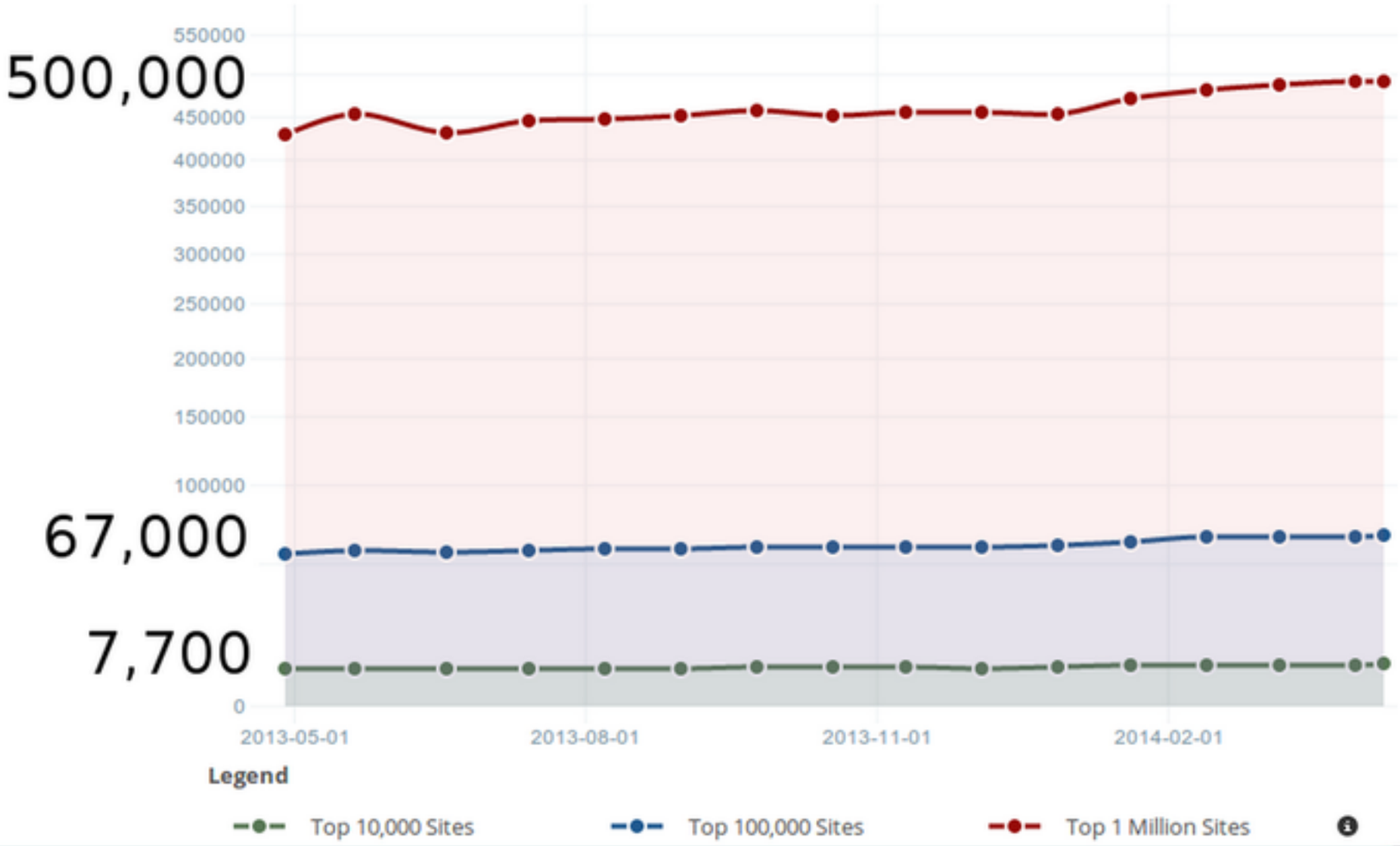
jQuery API

```
$('some-selector').method(parameters);
```

Example methods:

- `addClass / removeClass / css`
- `html / text`
- `on / off`
- `animate`

jQuery Ubiquity



jQuery Plugins

- Extend jQuery with custom functionality
- Easy way to share code

```
<html>
  <p>Some text in a document</p>
</html>

...

<script>
  $('p').myplugin();
</script>
```

A simple plugin

```
(function( $ ) {
 $.fn.myplugin = function() {
  return this.each(function() {
   $(this)
    .addClass("some-class")
    .css("color", "red");
   });
 };
})( jQuery );
```

Use cases

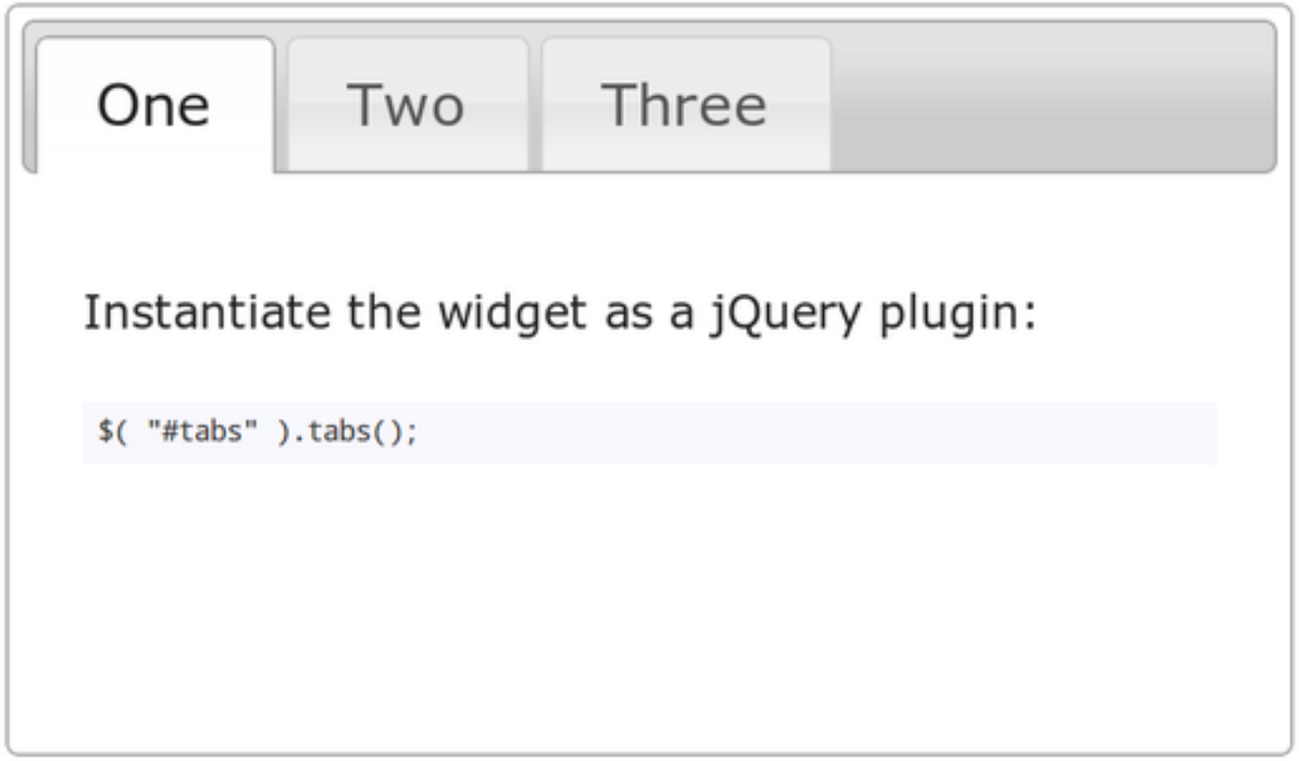
Well suited to use cases where plugin execution *results in some intended side-effect:*

- Text manipulation
- Animation
- Layout

A Stateful Widget

- The active tab is the widget state
- Use the javascript API to cycle the active tab

Cycle Tabs



Stateful plugins

- Track & query state
- Mutate options
- Plugin interactions
- Reverse DOM changes
- Notify event listeners

Widgets

Widgets with OOP

```
( function( $ ) {
  function MyWidget( element ) {
    this.init( element );
  }
  MyWidget.prototype.init = ...
  MyWidget.prototype.destroy = ...
  MyWidget.prototype.addListener = ...
  MyWidget.prototype.removeListener = ...
  MyWidget.prototype.setOption = ...

  ...
```

OOP Plugins

```
( function( $ ) {
  ...

  $.fn.mywidget = function() {
    return this.each( function() {
      $.data( this, "mywidget", new MyWidget( this ) );
    });
  };
})( jQuery );
```

Widget API with OOP

Interact with the widget via an API:

```
$(document).ready( function() {
  var widget = $( "#foo" ).mywidget().data( "mywidget" );
  widget.doSomething();
});
```

OOP → Boilerplate!

Encapsulate the boilerplate code in a common prototype function:

```
( function( $ ) {
  MyWidget.prototype = ...
  $.extend(MyWidget.prototype, myPrototype);
})(jQuery);
```

OOP Advantages

- Object instance
- API for interactions
- D.R.Y. w/ a Shared prototype

OOP Dis-advantages

- Custom prototype maintenance
- Non-standard API
    - events
    - mutable options

jQuery UI
Widget Factory

jQuery UI widget factory

- ✔ Track & query state
- ✔ Mutate options
- ✔ Widget interactions
- ✔ Reverse DOM changes
- ✔ Notify event listeners

Consistent API

Stopwatch example

```
jQuery('#timer').stopwatch();
```

window.setInterval

> Calls a function or executes a code snippet repeatedly, with a fixed time delay between each call to that function.

```
window.setInterval( callback, interval );
```

Stopwatch example

Stopwatch example

Widget Factory Deep Dive

```
(function($) {
  $.widget("rich.stopwatch", {
    /** prototype object **/
  });
})(jQuery);
```

Anatomy of a widget

```
( function( $ ) {
  $.widget( "rich.stopwatch", {
    options: { ... }
    doSomething: function() {
      // public methods have no "_" prefix
    },
    _helper: function() {
      // private methods have an "_" prefix
    }
  });
})( jQuery );
```

Default Options

```
( function( $ ) {
  $.widget( "rich.stopwatch", {

    options: {
      autostart: true,
      direction: 'decreasing',
      increment: 100 // in milliseconds
    },
    ...
  });
})( jQuery );
```

Widget API

Invoking the public methods:

```
$('#timer').stopwatch('doSomething', 'optionalParam');

// or

var widget = $('#timer').data('richStopwatch');
widget.doSomething();
```

Invoking Methods

Private methods *are* accessible:

```
var widget = $('#timer').data('richStopwatch');
widget._helper();
```

Accessible private methods helps with widget extension

Lifecycle and callbacks

- _create / _destroy
- disable / enable
- _setOption
- _trigger

Widget Initialisation

Optionally override the `_create` method

- Apply DOM changes
- Register event listeners
- *create* event fired implicitly

Create

```
_create: function() {
  this.digitsElement = this.element.find('.digits');
  this.startTime = this.digits();
  this.digits(this.startTime);
  if (this.options.showProgressBar) {
    this._addProgressBar();
  }
  if (this.options.autostart) {
    this._createInterval();
  }
},
```

Widget Clean Up

- Leave the DOM as you found it
- Remove any event handlers
- Remove any timers/intervals

Destroy!

```
_destroy: function() {
  this._removeInterval();
  if (this.progressBar) {
    this.progressBar.parents('.progress').first().remove();
  }
  this.digitsElement.text(this.startTime);
  this._trigger('destroy', undefined, this._dumpUi());
}
```

Enable / Disable

```
$.widget = {  // the jQuery UI widget object
  ...
  enable: function() {
    return this._setOptions({ disabled: false });
  },

  disable: function() {
    return this._setOptions({ disabled: true });
  },

}
```

Mutable Options

The widget Factory provides the `option` method for getting/setting the initialisation options

```
$('#timer').stopwatch('option', 'showProgressBar', false);
```

Hook in via the `_setOption` method if required

Mutable Options

```
_setOption: function (key, value) {
  var widget = this;
  if (this.options.key === value) {
    return;
  }
  switch (key) {
    ...
  }
  this._super(key, value);
},
```

Mutable Options

```
case 'disabled':
  if (value === true) widget._disable();
  else widget._enable();
  break;
case 'showProgressBar':
  if (value === true && !widget.progressBar) {
    widget._addProgressBar();
  } else if (value === false && widget.progressBar) {
    widget._removeProgressBar();
  }
  break;
```

Events & Callbacks

- Events allow for object extension
- Callbacks provided as either:
    1. plugin options
    2. event listeners

Callbacks as options

Option name matches the event name:

```
var options = {
  ...

  create: function(event, ui) {
    ui.element.css('background-color', ready);
    ui.element.css('color', 'white');
  }
```

Callbacks as event listeners

jQuery event is prefixed with the event name

```
$('#timer').on('stopwatchcreate'), function(event, ui) {
  ui.element.css('background-color', ready);
  ui.element.css('color', 'white');
});
```

Triggering events

Triggering events couldn't be any easier:

```
this._trigger('eventname', event, ui);
```

Function Parameters:

- `event` usually propagates an observed event
- `ui` is an object that shares widget states

The UI parameter

Consolidate `ui` object creation in a method

```
_dumpUi: function() {
  return {
    element: this.element,
    digits: this.digits()
  };
},
```

A note on styling

Adopting the jQuery UI widget factory *does not* tie you to the jQuery UI themes.

Examples in this demo were styled with:

- Bootstrap (http://getbootstrap.com)
- less.js (http://lesscss.org/)

Testing and Docs

Testing

- Jasmine - BDD for javascript
- Jasmine-jQuery - API for handling DOMfixtures in your tests
- Karma - test runner

Jasmine Example

```
describe("The 'toBe' matcher compares with ===", function() {
  it("and has a positive case ", function() {
      expect(true).toBe(true);
    });
  it("and can have a negative case", function() {
      expect(false).not.toBe(true);
    });
});
```

More on testing

Tomorrow at DevNation:

> Mobile web development: workflow and best practices - Lukas Fryc (Room 208)

Testing in richwidgets:

> github.com/richwidgets/richwidgets/blob/master/TESTS.md

API docs

YUIDoc can be used to provide generated API documentation

```
options: {
  /**
   * Whether to start the stopwatch on plugin execution
   *
   * @property autostart
   * @type Boolean
   * @default true
   */
  autostart: true,
```

Stopwatch API docs

> /assets/api/classes/stopwatch.html

Stopwatch example

```
/**
 * A stopwatch widget
 *
 * @module Output
 * @class chart
 */
(function ($) {

  $.widget('rich.stopwatch', {

    options: {
      autostart: true,
      direction: 'decreasing',
      disabled: false,
      showProgressBar: true,
      increment: 100 // in milliseconds
    },

    _create: function() {
      this.digitsElement = this.element.find('.digits');
```

Credits

- http://www.flickr.com/photos/jdhancock/5845280258/
- http://leslycorazon.wikispaces.com/file/detail/head-silhouette-with-question-mark.png/319199232
- http://openclipart.org/detail/crowd-by-kattekrab
- http://brand.jquery.org/logos/
- http://www.flickr.com/photos/26782864@N00/3297205226/
- http://commons.wikimedia.org/wiki/File:Thunderbird_assembly_line.jpg
- http://openclipart.org/detail/171954/cycle-color-by-karthikeyan-171954
- http://openclipart.org/detail/90451/keep-tidy-outside-01-by-anonymous
- http://openclipart.org/detail/181653/new-years-party-by-liftarn-181653

Learn More on jQuery UI

- https://api.jqueryui.com/jquery.widget/
- http://ajpiano.com/widgetfactory
- https://learn.jquery.com/jquery-ui/widget-factory/how-to-use-the-widget-factory/

- http://richwidgets.io