ArrayPipeline

A new way to manipulate Ember Arrays

Introduction

- *Hello*, my name is **Andy**
- Follow me **@NivenHuH**
- Director at MochaLeaf
  - Software Engineer
  - Code Reviewer
  - Teacher
  - The “Business Guy”
- Software development since mid-90’s

Introduction

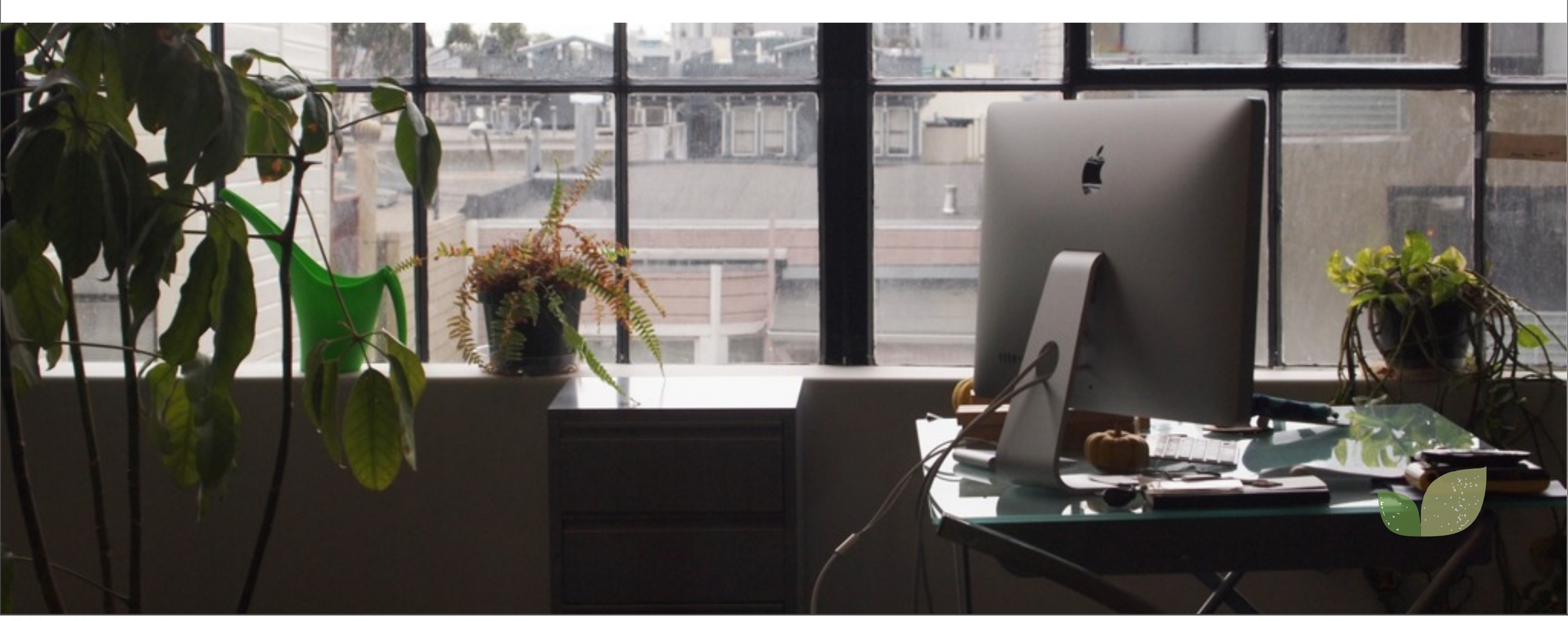
Consultancy

- Rich Web Applications
- Rich iOS / Android Apps

We're Hiring!

- Sr. JS Engineer
- UX Designer

Follow us on Twitter **@MochaLeaf**



Obligatory Picture of Cat

The Array Problem

The Array Problem

We want to do these things with arrays:

- Manipulate
- Filter
- Sort
- Introspect

We want to do these in any order, and we want to customize what we do.

The Ember “Native” Solution

Ember “Native” Solution

ArrayController / ArrayProxy

To wrap / maintain original content

itemController / objectAtContent

To manipulate objects in the array

SortableMixin

To sort objects in the array

Array.filter

To filter objects out of the array

Ember “Native” Solution

ArrayController / ArrayProxy

To wrap / maintain original content

- Unable to control order of operations
- Recalculate the whole, instead of just what's needed
- Complex when you need to do a number of operations

itemController / objectAtContent

To manipulate objects in the array

SortableMixin

To sort objects in the array

Array.filter

To filter objects out of the array

Ember “Native” Solution

ArrayController / ArrayProxy

To wrap / maintain original content

itemController / objectAtContent

To manipulate objects in the array

- applies to objects post sort (from content) [bug]

SortableMixin

To sort objects in the array

Array.filter

To filter objects out of the array

Ember “Native” Solution

ArrayController / ArrayProxy

To wrap / maintain original content

itemController / objectAtContent

To manipulate objects in the array

SortableMixin

To sort objects in the array

Array.filter

To filter objects out of the array

- Unable to utilize wrapped object properties (that were wrapped by objectAtContent/ itemController)

Ember “Native” Solution

ArrayController / ArrayProxy

To wrap / maintain original content

itemController / objectAtContent

To manipulate objects in the array

SortableMixin

To sort objects in the array

Array.filter

To filter objects out of the array

- Override arrangedContent to implement filtering
- If you want to filter then sort
  - implement additional ArrayProxy
  - override sort function to filter first

DECEASED

ArrangeableMixin

https://github.com/emberjs/ember.js/pull/1562

ArrangeableMixin
(SortableMixin + filtering)

- PR closed to not be merged
- Excessive recalculation on changes
- A little buggy
- Not flexible enough
- Was a personal lifesaver before rethinking the problem

The ArrayPipeline

It's Just A Series of Tubes

The ArrayPipeline

Goals

- Easy to understand order of operations
- Familiar interface (Unix pipes)
- Recalculate only what's needed, rest is cached
- Able to chain n-number of operations
- More modular
- Eliminate the need to override built in methods *

** It's not necessarily bad to override built in methods, but I prefer to utilize hooks*

The ArrayPipeline

Components

ArrayPipelineMixin

- Integrate into ArrayProxy ArrayController
- Use as is, no subclassing
- Specify content & plugins
- Utilize results array

The ArrayPipeline

Components

ArrayPipelineMixin

- Integrate into ArrayProxy ArrayController
- Use as is, no subclassing
- Specify content & plugins
- Utilize results array

PipePlugin

- Integrates into ArrayPipeline
- Specify properties array (for observation changes)
- Specify process function (for pipe processing)
  - Input: previous result set from last PipePlugin
  - Output: array for next PipePlugin

A Sample Problem

Surprise: It Involves Books

The Book Problem

- We have a list of books
- We want to select some of these books for another list
- Of the selected books, we want to filter out the book by the first letter of their name
- Finally, we want to sort our filtered results by name, then by year

Selecting a Book

```coffeescript
App.BooksController = Em.ArrayController.extend
  # Used for highlighting a book
  selectedBooks: []
```

Adding isSelected to Each Book

```coffeescript
# Used to add a 'isSelected' property to each Book object
BookProxy = Em.ObjectProxy.extend
  # We want the controller to be our BooksController
  controller: null
  isSelected: (->
    @get('controller.selectedBooks').indexOf( @get('content') ) != -1
  ).property('controller.selectedBooks.@each')
```

Wrapping Each Book

```coffeescript
# Wraps each book in our content array with a BookProxy
BookProxyPipe = Em.PipePlugin.extend
  # The process method is required for each PipePlugin.
  # It takes an inputArr of objects to operate on and is expected to return
  # an array of relevant objects post process.
  process: (inputArr) ->

    # This will return back an array of wrapped Books
    inputArr.map (item) ->

      # Each PipePlugin will have access to a controller instance
      # (Will default to the controller that the ArrayPipeline was mixed into)
      BookProxy.create( content: item, controller: @get('controller') )
```

Filtering Each Selected Book

```coffeescript
# A PipePlugin to filter out our content array for isSelected objects
App.SelectionFilter = Em.PipePlugin.extend
  # A list of properties that we will recompute on if changed
  properties: ['isSelected']

  # This process method is going to take our input array and filter out
  # for each property we have
  process: (inputArr) -> inputArr.filterProperty('isSelected', true)
```

Filtering Each Selected Book by First Letter

```coffeescript
# This is a custom filter method that will filter on the first letter of each book name
BookLetterFilterPipe = Em.PipePlugin.extend
  # A list of properties we want to filter on
  properties: ['name', 'controller.selectedLetter']

  # Process method to filter our books by name
  process: (inputArr) ->
    # We have no processing to do if a letter is not selected
    letter = @get('controller.selectedLetter')
    return inputArr if !letter?

    # This will return a filtered version of our inputArr if we do have a letter to use
    regex = new RegExp("^#{letter}", 'i')
    inputArr.filter (item) ->
      return false if (typeof item.get('name') != 'string')
      return item.match(regex)
```

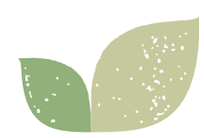
Sorting Each Selected & Filtered Book

```coffeescript
# A PipePlugin to sort our books by year then name
BookSortPipe = Em.PipePlugin.extend
  # A list of properties we will sort by
  properties: ['year', 'name']

  # Process method to sort our inputArr by each property
  process: (inputArr) ->
    # Sort our Input Array and return a sorted array
    inputArr.sort (obj1, obj2) ->
      result = 0

      # For each of our properties, we are going to compare items until we
      # can sort (or run out of properties)
      forEach( @get('properties'), (property) ->
        result = Em.compare(obj1, obj2) if result == 0
      )

      # Return our result code so the native sort function can do it's thing
      return result
```



Setting Up The Pipeline

```coffeescript
App.BooksController = Em.ArrayController.extend Ember.ArrayPipelineMixin,
  # You can mix strings with class definitions, the plugins will be auto-instantiated
  plugins: [BookProxyPipe, 'App.SelectionFilter', BookLetterFilterPipe, BookSortPipe]

  # Used for highlighting a book
  selectedBooks: []

  # Used for filtering by first letter
  selectedLetter: null
```

A Call To Action

A Call To Action

AKA: I need help!

- I’m a consultant, so free time is scarce :’(
- Please contribute @ GitHub:

  https://github.com/Mochaleaf/ArrayPipeline
- First revision will be posted **Sunday**
- Follow @**NivenHuH** for updates!

Did I Mention That We're Hiring?

Looking for...

- Senior Javascript Developer
- UX Designer

Perks

- Working on fun Ember projects
- Learning new stuff with our team
- Beer: in the fridge all the time :)

Thanks!