




Two Households, Both Alike In Dignity

A Not-So-Tragedy of Refactoring
Front-end APIs

Julia Nguyen - @fleurchild



Julia Nguyen @fleurchild

Founder @ifmeorg
Storyteller @mhprompt
Organizer @wscsf
Developer @indiegogo



A movie poster for William Shakespeare's Romeo + Juliet. The background is a close-up of a young man and woman kissing. The man has blonde hair and is looking down. The woman is on the right, her face partially visible. The background is a textured teal color with floral patterns. In the center, there is a decorative frame containing the title and director information. The title 'ROMEO + JULIET' is in large, bold, black letters. Above it, 'WILLIAM SHAKESPEARE'S' is in smaller black letters. Below it, 'FROM THE VISIONARY DIRECTOR OF MOULIN ROUGE' is in even smaller black letters. The frame is ornate with floral and scrollwork designs.

WILLIAM SHAKESPEARE'S
ROMEO + JULIET
FROM THE VISIONARY DIRECTOR OF MOULIN ROUGE

Prologue

- Backer Experience and Trust Team
- Tech stack
 - Back-end: Ruby on Rails
 - Front-end: Angular 1, TypeScript (recent)

**Indiegogo is more
than crowdfunding**

Perk = Product?

STORY [UPDATES](#) [COMMENTS](#) [BACKERS \(7\)](#)

PERKS

Short Summary

For my Spring 2016 BFA Exhibition, I presented a body of work titled "Menagerie". It consisted of over 130 5x7" color photographs. Now, I am doing a short run of t-shirts as a promotion for the show and as a fun thing for my art to be on!

What I Need & What You Get

The upfront printing of the shirts is close to 300 for 20+ shirts , so how we will do it is you will reserve your shirt on IndieGogo **for 16\$** You will also include your **shirt size**. If you want more than one, please pay the correct amount and include that size as well.

For now, the shirts will come in one color (white) with the designs shown above on the front and back.

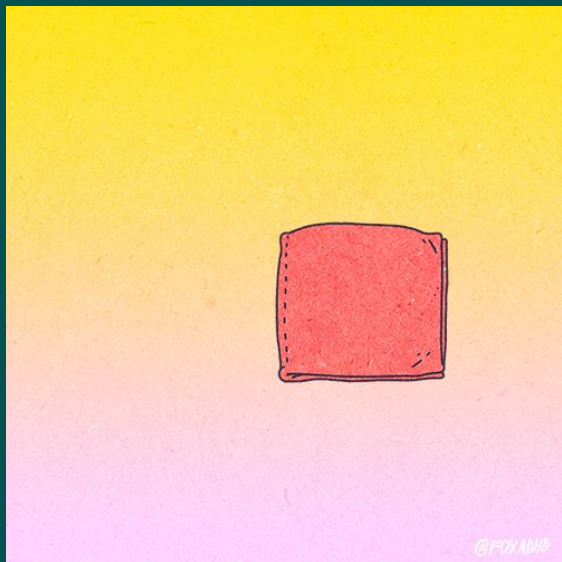
\$20 USD

Shirt + a Random Print

Not only do you get one of the shirts, but I will also include a 5"x7" print from the Menagerie Series. If you have seen the show and have a particular print in mind, please let me know personally.

6 claimed

ESTIMATED MAY 2016



Act I

C.R.E.A.M.

A close-up photograph of a bronze statue of Juliet, the character from Shakespeare's Romeo and Juliet. She is depicted with her hair flowing and a serene expression. The text "IN FAIR VERONA" is superimposed in white, bold, sans-serif capital letters across the center of the image. The background is a solid, light blue color. The entire image is framed by a dark teal background with white L-shaped corner brackets in the top-left and bottom-right corners.

IN FAIR VERONA



Two Households, Both Alike In Dignity

A Not-So-Tragedy of Refactoring Front-end APIs



admin
San Francisco, United States
[About](#)

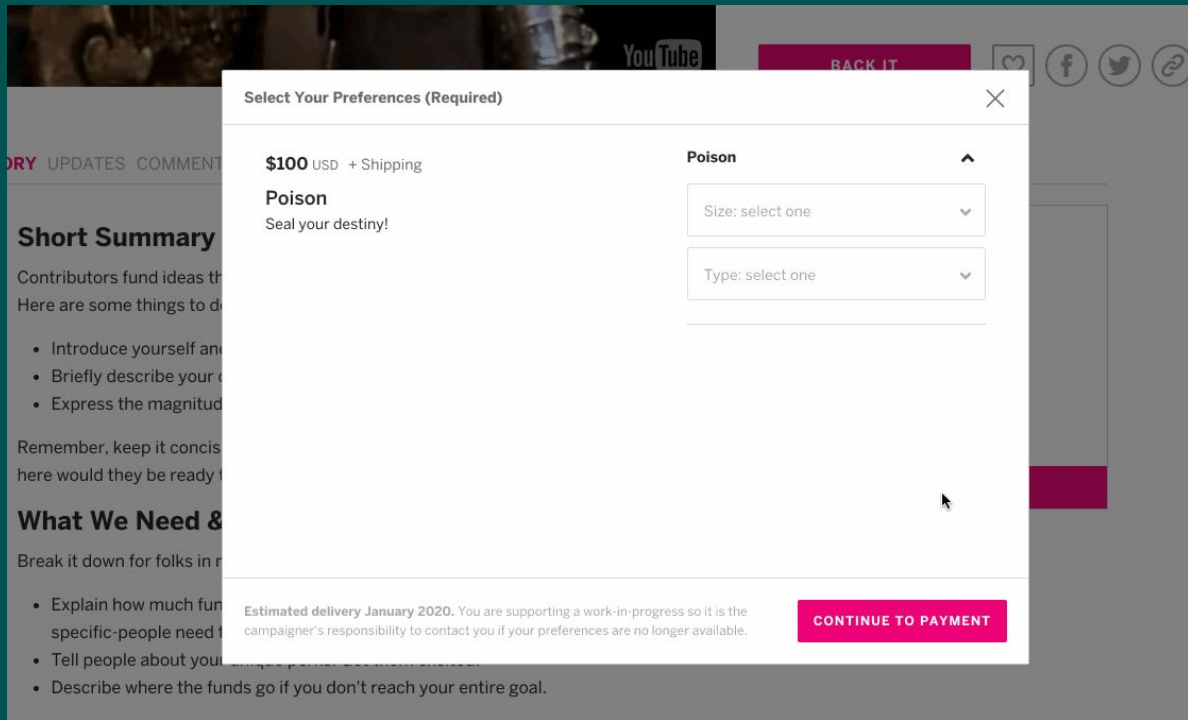
\$0 USD raised by 0 backers

0% of \$9,000 [flexible goal](#)

2 months left

BACK IT





Capulets: Items and Options



TORY [UPDATES](#) [COMMENTS](#) [BACKERS](#)

Our Story

(Introduce yourself...)

\$0 USD raised by 0 backers

0% of \$532 [flexible goal](#)

BACK IT



PERKS

\$32 USD

Watermelons

Yum!

1 claimed

Let us know if you think this campaign [contains prohibited content](#).

Montagues: Apple Pay





Act II

An Apple. A Pay.

Select Your Preferences (Required)



\$100 USD + Shipping

Poison

Seal your destiny!

Poison



Size: select one



Size: Fairy

Size: Banshee

Size: Beast

Estimated delivery **January 2020**. You are supporting a work-in-progress so it is the campaigner's responsibility to contact you if your preferences are no longer available.

SAVE

- Dagger

(3) Consolidate modals for Items and Options and Multiperk/Apple Pay and handle opening/hiding more consistently

- Consolidate UI
- Create polymorphic states on what to open/hide and which animations to display

(2) Cleanup shared data between PerkPrefsService and CampaignPerkSelection

- Data about whether a perk has items and options

(2) Refactor setPerk and updateCart in PerkPrefsServices

- Both modals need to consistently add perk to persistentCampaignCart in perkPrefsService
- Avoid modifying of variables on perkPrefs object outside of the service

(2) Refactor selectPerk and sendPerkSelectionEvent in CampaignPerkSelection

- Cleanup logic for selectPerk and selectPerkWithOptions
- Cleanup how items information is passed into sendPerkSelectionEvent

(2) Reduce and cleanup cross module dependencies

Too many modals!

Act III

Starcrossed Perks

```

declare interface PerkFactoryPerk
{
  id: number;
  label: string;
  amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  estimated_delivery_date: Date;
  non_tax_deductible_amount:
  number;

  use_non_tax_deductible_amount:
  boolean;
  shipping_address_required:
  boolean;
  perk_image_public_id: string;
  shipping: any;
  retail_amount: number;
  sold_out: boolean;
  perk_item_links: PerkItemLinks[];
}

```

*Note: Used by
Multiperk/Apple Pay*

```

export interface Perk {
  id: number;
  label: string;
  amount: number;
  retail_amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  estimated_delivery_date: Date;
  non_tax_deductible_amount:
  number;

  use_non_tax_deductible_amount:
  boolean;
  shipping_required: boolean;
  perk_image_public_id: string;
  shipping_fees: any;
}

```

*Note: Used by Items and
Options through PerkBuilder*

```

export declare interface
PerkJSON {
  amount: number;
  campaign_slug: string;
  id: number;
  label: string;
  shipping_address_required:
  boolean;
  shipping?: {fees: Fees};
  shipping_required: boolean;
  shipping_fees: Fees;
  items: PerkItem[];
}

```

*Note: Used by Apple Pay
Service
Temporarily added
shipping_required and
shipping_fees to get Apple
Pay to work for Items and
Options*

All of the perks!



Act IV

A Wedding

Primary Goal

```
declare interface PerkFactoryPerk {
  id: number;
  label: string;
  amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  estimated_delivery_date: Date;
  non_tax_deductible_amount:
  number;

  use_non_tax_deductible_amount:
  boolean;
  shipping_address_required:
  boolean;
  perk_image_public_id: string;
  shipping: any;
  retail_amount: number;
  sold_out: boolean;
  perk_item_links: PerkItemLink[];
}
```

*Note: Used by
MultiPerk/Apple Pay*

```
export interface Perk {
  id: number;
  label: string;
  amount: number;
  retail_amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  estimated_delivery_date: Date;
  non_tax_deductible_amount:
  number;

  use_non_tax_deductible_amount:
  boolean;
  shipping_required: boolean;
  perk_image_public_id: string;
  shipping_fees: any;
}
```

*Note: Used by Items and
Options through PerkBuilder*

```
export declare interface
PerkJSON {
  amount: number;
  campaign_slug: string;
  id: number;
  label: string;
  shipping_address_required:
  boolean;
  shipping?: {fees: Fees};
  shipping_required: boolean;
  shipping_fees: Fees;
  items: PerkItem[];
}
```

*Note: Used by Apple Pay
Service
Temporarily added
shipping_required and
shipping_fees to get Apple
Pay to work for Items and
Options*



```
export declare interface Perk {
  id: number;
  label: string;
  amount: number;
  retail_amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  campaign_slug: string;
  estimated_delivery_date: Date;
  non_tax_deductible_amount: number;
  use_non_tax_deductible_amount: boolean;
  shipping_address_required: boolean;
  perk_image_public_id: string;
  sold_out: boolean;
  shipping_fees: Fees;
  perk_item_links: PerkItemLink[];
}
```



Server-side Cleanup

Serializer

Picks certain attributes from model
Serializers can be exposed from the
controller

```
module Api
  class Resource < SimpleDelegator
    include ActiveModel::Serialization
  end
end
```

Resource (???)

Contains logic for *dealing* with attributes, so you don't have duplicate logic between serializers

SimpleDelegator

A Ruby class that implements the decorator pattern.

Decorator Pattern

A design pattern that allows behaviour to be added to a single object without affecting other objects of the same class
Is an example of separation of concerns!

Two Perk Serializers

`private_api/perk_serializer.rb`

- Regular perk
- Querying methods found in the serializer
- Exposed in `campaign_perks_controller.rb`

`private_api/commerce/perk_serializer.rb`

- Perk with items and options
- Querying methods found in `/lib/commerce/resources/perk.rb`
- Exposed in `perk_items_controller.rb`

Example of duplication: `shipping_fees` method

Base Perk Serializer

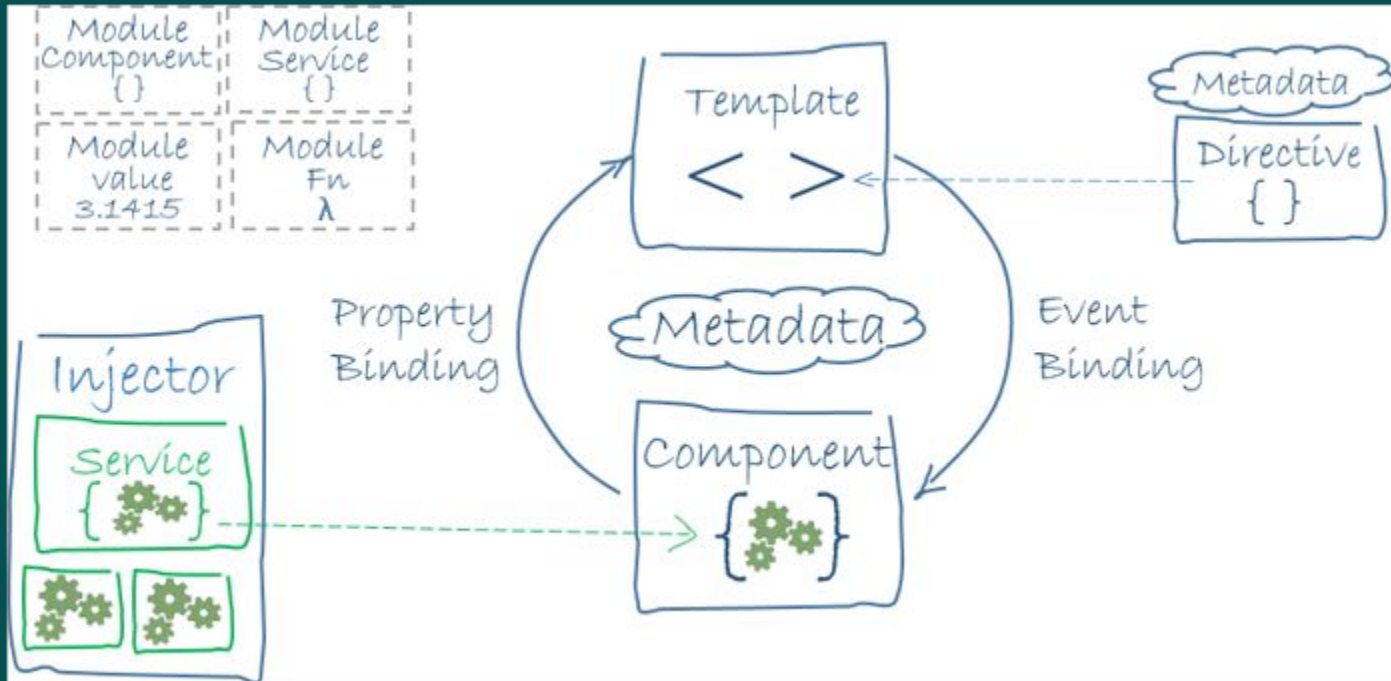
- Create **BasePerkSerializer** to be the parent serializer for perks
- Place shared attributes
- Initialize resource (SimpleDelegator) in that serializer, don't need a separate controller

```
1 module PrivateApi
2   class BasePerkSerializer < ActiveRecord::Serializer
3     attributes :id, :amount, :label, :description, :estimated_delivery_date, :featured, :non_tax_deductible_amount, :perk_image_public_id,
4
5     def initialize(object, options = {})
6       super(::Commerce::Resources::Perk.new(object), options)
7     end
8   end
9 end
```



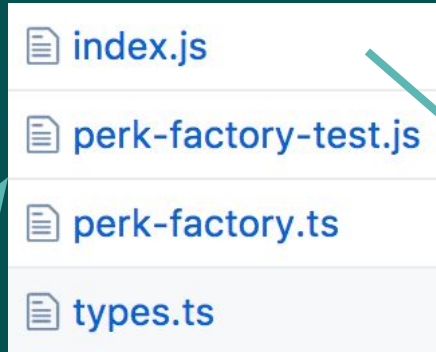
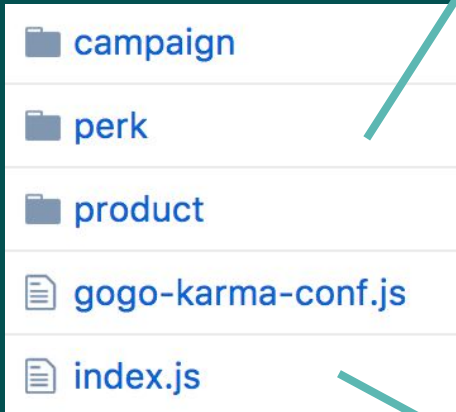
Client-side Cleanup

Angular Architecture



js/client

A place for services that make API calls



```
import perkFactory from './perk-factory.ts';
import 'angular';

angular.module('perk.perkFactory', ['utils'])
  .factory('perkFactory', perkFactory);
```

```
import '../ancillary';
import './perk';
import './product';
import './campaign';
```



One Type to Rule Them All

- Create **Perk** typing and related typings like **PerkItem** in **js/client/perks/types.ts**
- Get rid of **PerkJSON** and **PerkFactoryPerk**, replace with **Perk**

Settle on Attributes, Remove Duplication

- **PerkFactory** is sort of misleading, only calculated shipping fees
 - Converted between **shipping.fees** and **shipping_fees**
- Refactor **PerkFactory**, eliminate **shipping.fees** and use **shipping_fees**
- Convert **PerkFactory** to TypeScript
 - **perk-factory.js** was 342 lines
 - **Perk-factory.ts** is 182 lines

A Good Fake Death!

```
declare interface PerkFactoryPerk {
  id: number;
  label: string;
  amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  estimated_delivery_date: Date;
  non_tax_deductible_amount:
  number;

  use_non_tax_deductible_amount:
  boolean;
  shipping_address_required:
  boolean;
  perk_image_public_id: string;
  shipping: any;
  retail_amount: number;
  sold_out: boolean;
  perk_item_links: PerkItemLink[];
}
```

*Note: Used by
MultiPerk/Apple Pay*

```
export interface Perk {
  id: number;
  label: string;
  amount: number;
  retail_amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  estimated_delivery_date: Date;
  non_tax_deductible_amount:
  number;

  use_non_tax_deductible_amount:
  boolean;
  shipping_required: boolean;
  perk_image_public_id: string;
  shipping_fees: any;
}
```

*Note: Used by Items and
Options through PerkBuilder*

```
export declare interface
PerkJSON {
  amount: number;
  campaign_slug: string;
  id: number;
  label: string;
  shipping_address_required:
  boolean;
  shipping?: {fees: Fees};
  shipping_required: boolean;
  shipping_fees: Fees;
  items: PerkItem[];
}
```

*Note: Used by Apple Pay
Service
Temporarily added
shipping_required and
shipping_fees to get Apple
Pay to work for Items and
Options*



```
export declare interface Perk {
  id: number;
  label: string;
  amount: number;
  retail_amount: number;
  description: string;
  items: PerkItem[];
  secret: boolean;
  featured: boolean;
  campaign_slug: string;
  estimated_delivery_date: Date;
  non_tax_deductible_amount: number;
  use_non_tax_deductible_amount: boolean;
  shipping_address_required: boolean;
  perk_image_public_id: string;
  sold_out: boolean;
  shipping_fees: Fees;
  perk_item_links: PerkItemLink[];
}
```



Refactoring Front-end APIs in Summary

- Start from the server-side and move to the client-side, you will uncover more
- Use serializers! You don't *usually* need all of the data!
- Consolidate serializers and remove duplication through the decorator pattern
- Model attribute names should be consistent between the server-side and client-side
- Consolidate services that make the same API calls, make them importable modules!

Act V

Postmortem



Technical Debt

Extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution

Tackling debt “as you go” reduces debt and prevents debt from accruing

Can't Refactor Everything, But You Can Document It

(3) Consolidate modals for Items and Options and Multiperk/Apple Pay and handle opening/hiding more consistently

- Consolidate UI
- Create polymorphic states on what to open/hide and which animations to display

(2) Cleanup shared data between PerkPrefsService and CampaignPerkSelection

- Data about whether a perk has items and options

(2) Refactor setPerk and updateCart in PerkPrefsServices

- Both modals need to consistently add perk to persistentCampaignCart in perkPrefsService
- Avoid modifying of variables on perkPrefs object outside of the service

(2) Refactor selectPerk and sendPerkSelectionEvent in CampaignPerkSelection

- Cleanup logic for selectPerk and selectPerkWithOptions
- Cleanup how items information is passed into sendPerkSelectionEvent

(2) Reduce and cleanup cross module dependencies

Tackling Technical Debt in Summary

- Better to investigate technical debt at the beginning of a project than discover it later
- Account for technical debt in sprint planning, integrate it in the pull request process (it's not just a backlog issue)
- Get your PM involved in the process, even if they are non-technical
- Ask lots of questions from developers who have worked with the codebase for longer (people skills people!)
- Tackle your refactoring in bite-sized chunks, easier to undo if you mess up

Useful Resources

[Sandi Metz' Rules for Developers](#)

[Refactoring.com by Martin Fowler](#)

["Don't reset --hard: Strategies for Tackling Large Refactors" by Siena Aguayo](#)

["7 Design Patterns to Refactor MVC Components in Rails" by Viktoria Kotsurenko](#)

["JavaScript Factory Functions vs Constructor Functions vs Classes" by Eric Elliott](#)

Could tackling technical debt have saved Romeo + Juliet?

If the Capulets and Montagues got their shit together, quite possibly

But maybe the ~*drama*~ made their relationship?



**Let's tackle technical debt like
we're Mercutio and it's 1996**

@fleurchild