IBM Plex Sans

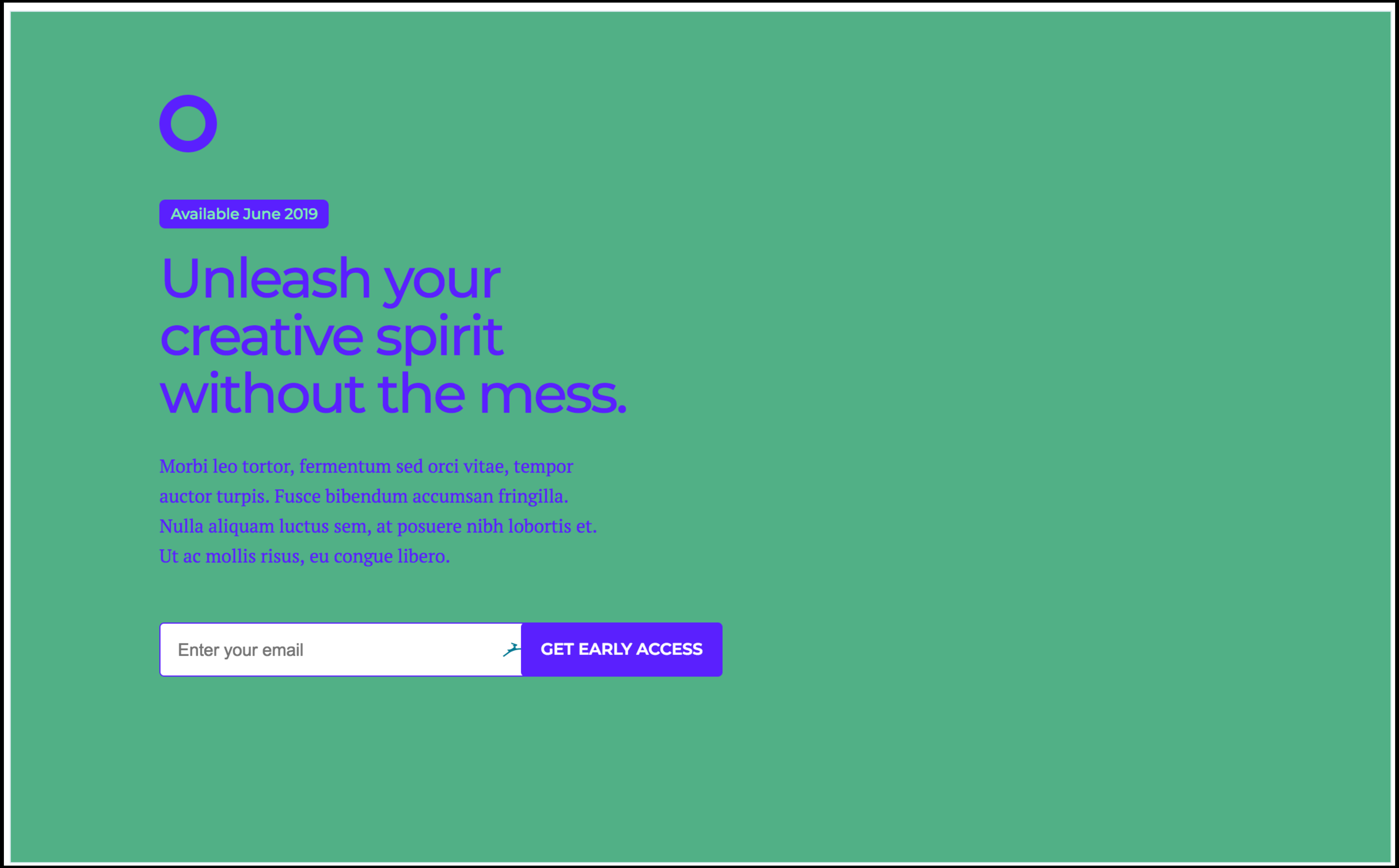
Light

12px

400px

Oh no! Conditionals.

Philippe Creux – April 2018 – VanRuby

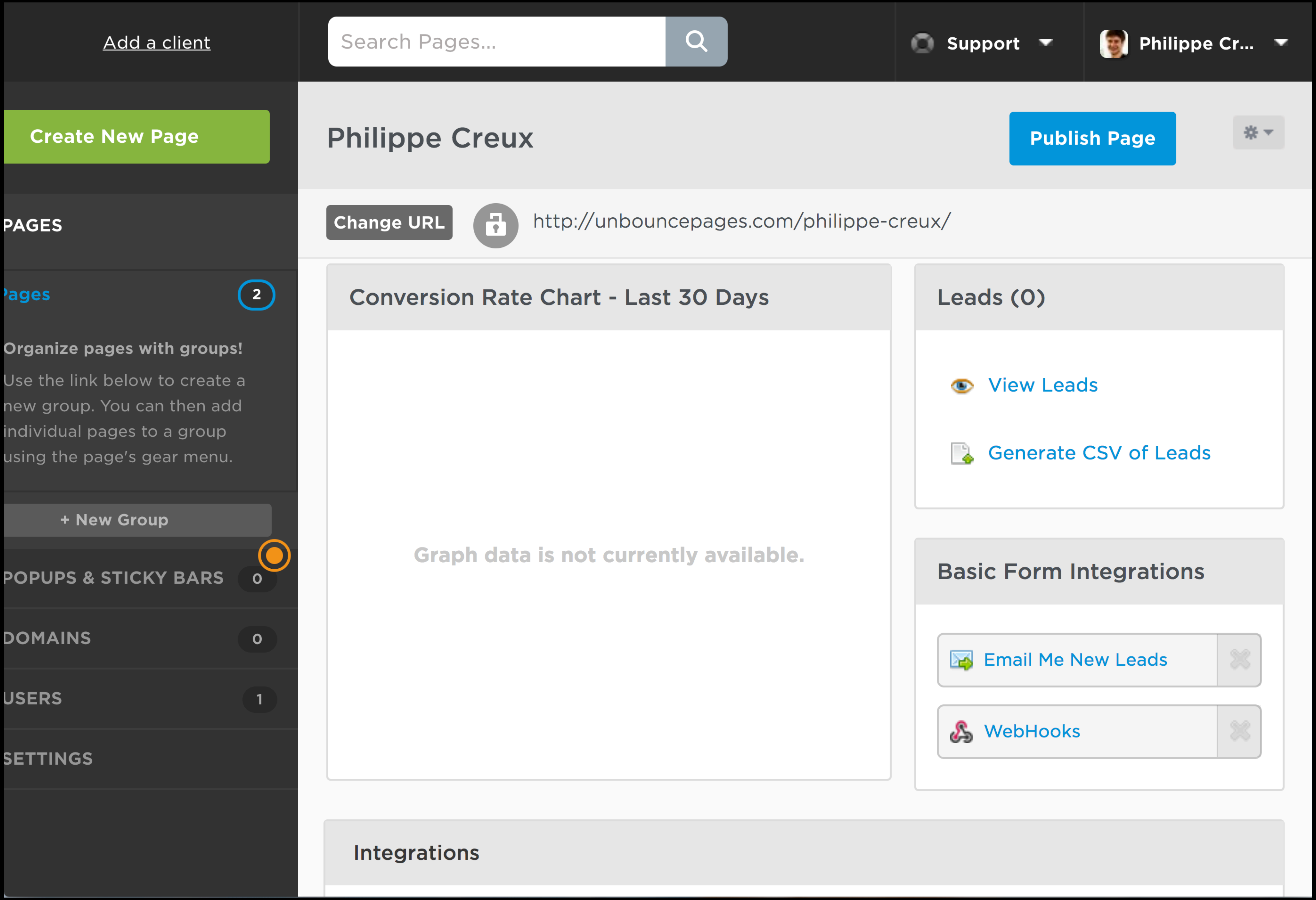


Available June 2019

Unleash your creative spirit without the mess.

Morbi leo tortor, fermentum sed orci vitae, tempor auctor turpis. Fusce bibendum accumsan fringilla. Nulla aliquam luctus sem, at posuere nibh lobortis et. Ut ac mollis risus, eu congue libero.

At Unbounce, we enable anyone to create landing pages. This, is a landing page.


Add a client

Search Pages...

Support

Philippe Cr...

Create New Page

Philippe Creux

Publish Page

PAGES

ages 2

Organize pages with groups!

Use the link below to create a new group. You can then add individual pages to a group using the page's gear menu.

+ New Group

POPUPS & STICKY BARS 0

DOMAINS 0

USERS 1

SETTINGS

Change URL http://unbouncepages.com/philippe-creux/

Conversion Rate Chart - Last 30 Days

Graph data is not currently available.

Leads (0)

View Leads

Generate CSV of Leads

Basic Form Integrations

Email Me New Leads

WebHooks

Integrations

There are a lot of features that can be toggled on and off depending on your user type, your subscription, etc.

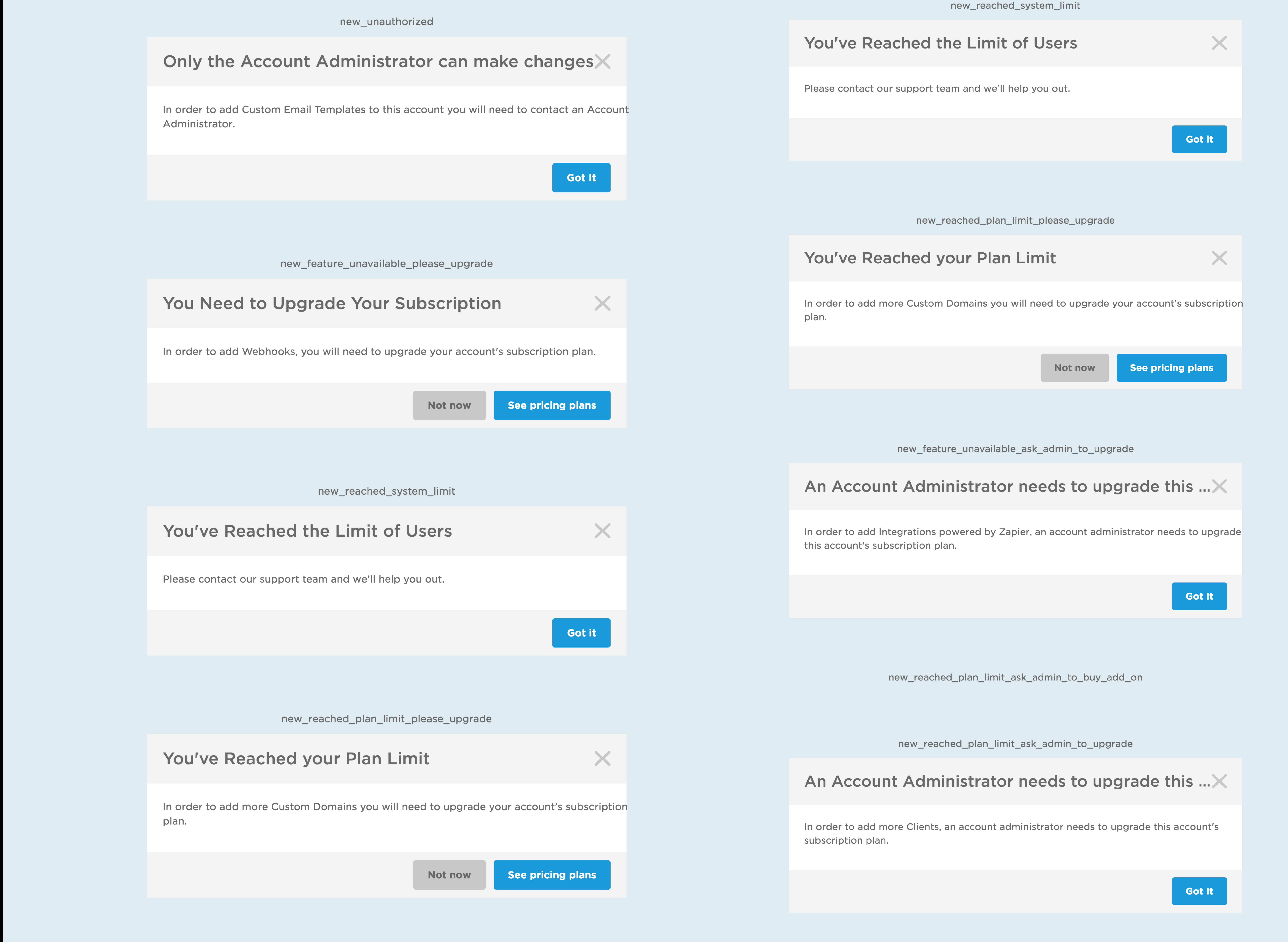
```
def webhook?
  feature_available? && user_authorized? && under_plan_limit?
end
```

Each feature must meet multiple criteria to be available.

A feature might not be available for a lot of reasons.

We want to tell users why they can't enable a feature

So a simple condition statement is not sufficient.



new_unauthorized

Only the Account Administrator can make changes

In order to add Custom Email Templates to this account you will need to contact an Account Administrator.

new_feature_unavailable_please_upgrade

You Need to Upgrade Your Subscription

In order to add Webhooks, you will need to upgrade your account's subscription plan.

new_reached_system_limit

You've Reached the Limit of Users

Please contact our support team and we'll help you out.

new_reached_plan_limit_please_upgrade

You've Reached your Plan Limit

In order to add more Custom Domains you will need to upgrade your account's subscription plan.

new_reached_system_limit

You've Reached the Limit of Users

Please contact our support team and we'll help you out.

new_reached_plan_limit_please_upgrade

You've Reached your Plan Limit

In order to add more Custom Domains you will need to upgrade your account's subscription plan.

new_feature_unavailable_ask_admin_to_upgrade

An Account Administrator needs to upgrade this ...

In order to add Integrations powered by Zapier, an account administrator needs to upgrade this account's subscription plan.

new_reached_plan_limit_ask_admin_to_buy_add_on

new_reached_plan_limit_ask_admin_to_upgrade

An Account Administrator needs to upgrade this ...

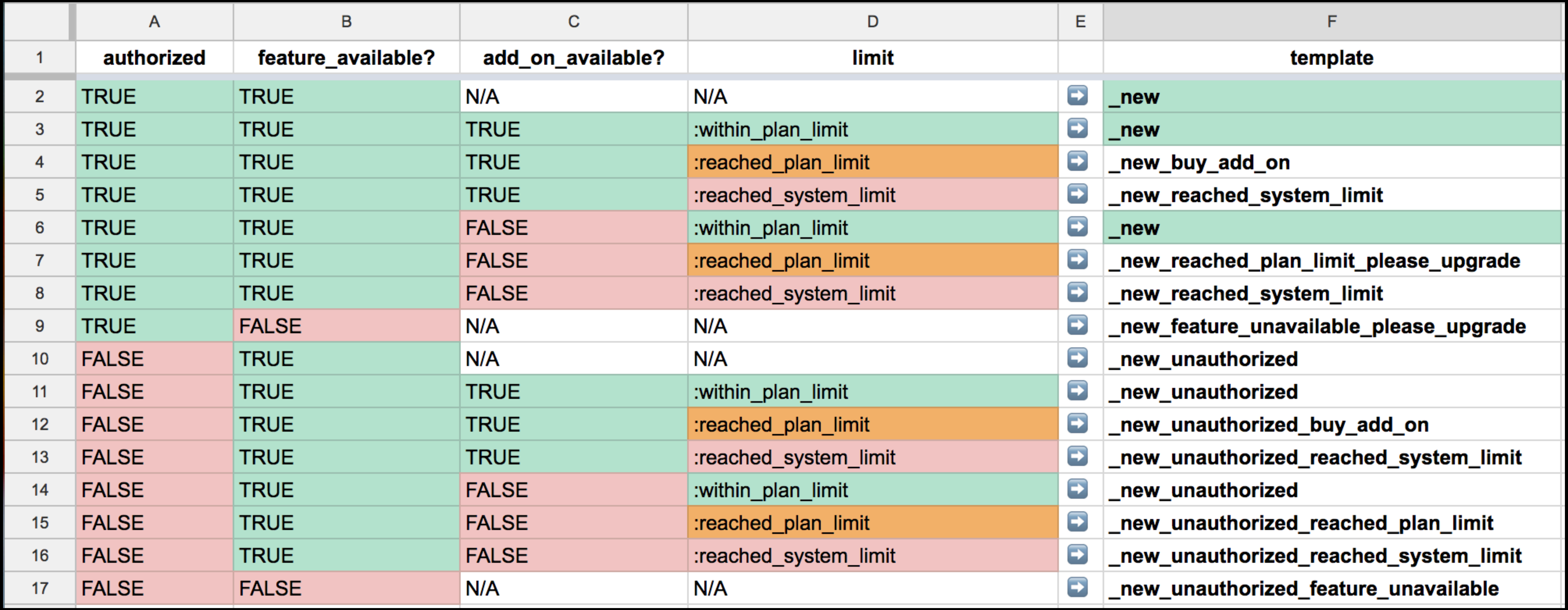
In order to add more Clients, an account administrator needs to upgrade this account's subscription plan.

```ruby
# returns the partial to render:
#   '_new', '_new_reached_plan_limit', '_new_buy_add_on', 'new_reached_system_limit',...
def template(
  authorized,        # boolean - authorized to use the feature (admin vs view-only user)
  feature_available, # boolean - feature available on your plan
  add_on_available,  # boolean or nil - add-on available to purchase
  limit              # one of: :within_plan_limit, :reached_plan_limit, :reached_system_limit, nil (N/A)
  )

  if authorized && feature_available && limit == :within_plan_limit
    return '_new'
  end

  if authorized && add_on_available && limit == :reached_plan_limit
    return '_new_buy_add_on'
  end

  # ...
end
```

The method determining the message to render could be quite tedious to write… and read.

```ruby
def template(authorized, feature_available, add_on_available, limit)
  if authorized
    if feature_available
      if add_on_available
        case limit
        when :within_plan_limit
          '_new'
        when :reached_plan_limit
          '_new_buy_add_on'
        when :reached_system_limit
          '_new_reached_system_limit'
        when nil
          '_new'
        end
      else # !add_on_available
        case limit
        when :within_plan_limit?
          '_new'
        when :reached_plan_limit?
          '_new_reached_plan_limit_please_upgrade'
        when :reached_system_limit?
          '_new_reached_system_limit'
        else
          '_new'
        end
      end
    else # !feature_available
      '_new_feature_unavailable_please_upgrade'
    end
  else #!authorized
    if feature_available
      if add_on_available
        case limit
        when :within_plan_limit?
          '_new_unauthorized'
        when :reached_plan_limit?
          '_new_unauthorized_buy_add_on'
        when :reached_system_limit?
          '_new_unauthorized_reached_system_limit'
        else
          '_new_unauthorized'
        end
      else # !add_on_available
        case limit
        when :within_plan_limit?
          '_new_unauthorized'
        when :reached_plan_limit?
          '_new_unauthorized_buy_add_on'
        when :reached_system_limit?
          '_new_unauthorized_reached_system_limit'
        else
          '_new_unauthorized'
        end
      end
    else # !feature_available
      '_new_unauthorized_feature_unavailable'
    end
  end
end
```

… really tedious.



	A	B	C	D	E	F
1	**authorized**	**feature_available?**	**add_on_available?**	**limit**		**template**
2	TRUE	TRUE	N/A	N/A	➡	**_new**
3	TRUE	TRUE	TRUE	:within_plan_limit	➡	**_new**
4	TRUE	TRUE	TRUE	:reached_plan_limit	➡	**_new_buy_add_on**
5	TRUE	TRUE	TRUE	:reached_system_limit	➡	**_new_reached_system_limit**
6	TRUE	TRUE	FALSE	:within_plan_limit	➡	**_new**
7	TRUE	TRUE	FALSE	:reached_plan_limit	➡	**_new_reached_plan_limit_please_upgrade**
8	TRUE	TRUE	FALSE	:reached_system_limit	➡	**_new_reached_system_limit**
9	TRUE	FALSE	N/A	N/A	➡	**_new_feature_unavailable_please_upgrade**
10	FALSE	TRUE	N/A	N/A	➡	**_new_unauthorized**
11	FALSE	TRUE	TRUE	:within_plan_limit	➡	**_new_unauthorized**
12	FALSE	TRUE	TRUE	:reached_plan_limit	➡	**_new_unauthorized_buy_add_on**
13	FALSE	TRUE	TRUE	:reached_system_limit	➡	**_new_unauthorized_reached_system_limit**
14	FALSE	TRUE	FALSE	:within_plan_limit	➡	**_new_unauthorized**
15	FALSE	TRUE	FALSE	:reached_plan_limit	➡	**_new_unauthorized_reached_plan_limit**
16	FALSE	TRUE	FALSE	:reached_system_limit	➡	**_new_unauthorized_reached_system_limit**
17	FALSE	FALSE	N/A	N/A	➡	**_new_unauthorized_feature_unavailable**

Ok, let's get back to the spec… That's so much easier to comprehend than nested conditionals…

The good news is that it’s pretty straightforward to do in Ruby. :)

```
RULES = {
  # authorized, feature_avail?, add_on_avail?, limit
  [ true,       true,           nil,           nil]                      => '_new',
  [ true,       true,           true,          :within_plan_limit]     => '_new',
  [ true,       true,           true,          :reached_plan_limit]    => '_new_buy_add_on',
  [ true,       true,           true,          :reached_system_limit]  => '_new_reached_system_limit',
  [ true,       true,           false,         :within_plan_limit]     => '_new',
  [ true,       true,           false,         :reached_plan_limit]    => '_new_reached_plan_limit_please_upgrade',
  [ true,       true,           false,         :reached_system_limit]  => '_new_reached_system_limit',
  [ true,       false,          nil,           nil]                      => '_new_feature_unavailable_please_upgrade',

  [ false,      false,          nil,           nil]                      => '_new_unauthorized_feature_unavailable',
  [ false,      true,           nil,           nil]                      => '_new_unauthorized',
  [ false,      true,           true,          :within_plan_limit]     => '_new_unauthorized',
  [ false,      true,           true,          :reached_plan_limit]    => '_new_unauthorized_buy_add_on',
  [ false,      true,           true,          :reached_system_limit]  => '_new_unauthorized_reached_system_limit',
  [ false,      true,           false,         :within_plan_limit]     => '_new_unauthorized',
  [ false,      true,           false,         :reached_plan_limit]    => '_new_unauthorized_reached_plan_limit',
  [ false,      true,           false,         :reached_system_limit]  => '_new_unauthorized_reached_system_limit',
}

def template(authorized, feature_available, add_on_available, limit)
  RULES.fetch([authorized, feature_available, add_on_available, limit])
end
```

TADA!