



Component-driven Rails with ViewComponents



Kirill Platonov

Building Shopify Apps with Rails & ViewComponents since 2021

Created open-source projects:

- **Polaris ViewComponents**
- Hotwire Livereload
- Shopify Hotwire Sample
- Shopify GraphQL

kirillplatonov.com

Polaris ViewComponents

polarisviewcomponents.org

POLARIS VIEWCOMPONENTS

Previews

Filter previews by name...

> Action List

> Autocomplete

> Avatar

> Badge

> Banner

> Bleed

> Box

> Button

> Button Group

> Callout Card

> Caption

> Card

> Checkbox

> Choice List

> Collapsible

> Contextual Save Bar

> Data Table

> Description List

> Display Text

> Divider

PreviewHTML

</>🔗🔄🔗

< 3/4 inch Leather pet collar Paid

DuplicatePromote ▼Save<>

Perfect for any pet

Credit card

Credit card information

SourceNotesParams

1<%= polaris_page(
2title: "3/4 inch Leather pet collar",
3subtitle: "Perfect for any pet",
4compact_title: true,
5back_url: "/products",
6next_url: "/product/2",
7secondary_actions: [
8{ content: "Duplicate", url: "/duplicate" },
9]

Goals for this talk:

- Overview of ViewComponents
- Help to decide if you need them
- Share practical tips

Why components in Rails?

Increase development speed 🚀

Code

```
<%= polaris_page(title: "My page with form") do |page| %>
  <% page.with_primary_action(disabled: true) { "Action" } %>

  <%= polaris_card do %>
    <%= polaris_form_layout do |form_layout| %>
      <% form_layout.with_item do %>
        <%= polaris_text_field(label: "Store name") %>
      <% end %>
      <% form_layout.with_item do %>
        <%= polaris_text_field(label: "Account email") %>
      <% end %>
      <% form_layout.with_item do %>
        <%= polaris_button(submit: true, primary: true) { "Submit" } %>
      <% end %>
    <% end %>
  <% end %>
<% end %>
```

Result

My page with form

Action

Store name

Account email

Submit

Avoid bloated views

Hello Tailwind 🙌

```
<form>
  <div class="space-y-12">
    <div class="border-b border-gray-900/10 pb-12">
      <h2 class="text-base/7 font-semibold text-gray-900">Profile</h2>
      <p class="mt-1 text-sm/6 text-gray-600">This information will be displayed publicly so
be careful what you share.</p>

      <div class="mt-10 grid grid-cols-1 gap-x-6 gap-y-8 sm:grid-cols-6">
        <div class="sm:col-span-4">
          <label for="username" class="block text-sm/6 font-medium text-gray-
900">Username</label>
          <div class="mt-2">
            <div class="flex items-center rounded-md bg-white pl-3 outline-1 -outline-offset-1
outline-gray-300 focus-within:outline-2 focus-within:-outline-offset-2 focus-within:outline-
indigo-600">
              <div class="shrink-0 text-base text-gray-500 select-none sm:text-
sm/6">workcation.com/</div>
              <input type="text" name="username" id="username" class="block min-w-0 grow py-
1.5 pr-3 pl-1 text-base text-gray-900 placeholder:text-gray-400 focus:outline-none sm:text-
sm/6" placeholder="janesmith">
            </div>
          </div>
        </div>

        <div class="col-span-full">
          <label for="about" class="block text-sm/6 font-medium text-gray-900">About</label>
          <div class="mt-2">
            <textarea name="about" id="about" rows="3" class="block w-full rounded-md bg-white
px-3 py-1.5 text-base text-gray-900 outline-1 -outline-offset-1 outline-gray-300
placeholder:text-gray-400 focus:outline-2 focus:-outline-offset-2 focus:outline-indigo-600
sm:text-sm/6"></textarea>
          </div>
          <p class="mt-3 text-sm/6 text-gray-600">Write a few sentences about yourself.</p>
        </div>
      </div>
    </div>

    <div class="mt-6 flex items-center justify-end gap-x-6">
      <button type="button" class="text-sm/6 font-semibold text-gray-900">Cancel</button>
      <button type="submit" class="rounded-md bg-indigo-600 px-3 py-2 text-sm font-semibold
text-white shadow-sm hover:bg-indigo-500 focus-visible:outline-2 focus-visible:outline-offset-2
focus-visible:outline-indigo-600">Save</button>
    </div>
  </div>
</form>
```

Enforce design systems

Share *views* code across apps

My case

Before

```
Untitled-1

<div class="Polaris-ButtonGroup">
  <div class="Polaris-ButtonGroup__Item">
    <button class="Polaris-Button Polaris-Button--pressable
Polaris-Button--variantSecondary Polaris-Button--sizeMedium
Polaris-Button--textAlignCenter" type="button">
      <span class="Polaris-Text--root Polaris-Text--bodySm
Polaris-Text--medium">Cancel</span>
    </button>
  </div>
  <div class="Polaris-ButtonGroup__Item">
    <button class="Polaris-Button Polaris-Button--pressable
Polaris-Button--variantPrimary Polaris-Button--sizeMedium Polaris-
Button--textAlignCenter" type="button">
      <span class="Polaris-Text--root Polaris-Text--bodySm
Polaris-Text--medium">Save</span>
    </button>
  </div>
</div>
```

After

```
Untitled-1

<%= polaris_button_group do |group| %>
  <%= group.with_button { "Cancel" } %>
  <%= group.with_button(primary: true) { "Save" } %>
<% end %>
```

How ViewComponent looks

Definition

```
Untitled-1

# app/components/greeting_component.rb
class GreetingComponent < ViewComponent::Base
  def initialize(name:)
    @name = name
  end
end
```

```
Untitled-1

# app/components/greeting_component.html.erb
<div>Hello, <%= @name %>!</div>
```

Usage

```
Untitled-1

<!-- usage -->
<%= render GreetingComponent.new(name: "Alice") %>
```


Wrappers

Definition

```
class ExampleComponent < ViewComponent::Base
  def initialize(title:)
    @title = title
  end
end
```

```
<span title="<%= @title %>"><%= content %></span>
```

Usage

```
<%= render(ExampleComponent.new(title: "my title")) do %>
  Hello, World!
<% end %>
```

```
<%= render(ExampleComponent.new(..).with_content("Hello, World!")) %>
```


Slots API ✨

Definition

```
Untitled-1
# blog_component.rb
class BlogComponent < ViewComponent::Base
  renders_one :header
  renders_many :posts
end
```

```
Untitled-1
<%=# blog_component.html.erb %>
<h1><%= header %></h1>

<% posts.each do |post| %>
  <%= post %>
<% end %>
```

Usage

```
Untitled-1
<%=# index.html.erb %>
<%= render BlogComponent.new do |component| %>
  <% component.with_header do %>
    <%= link_to "My blog", root_path %>
  <% end %>

  <% BlogPost.all.each do |blog_post| %>
    <% component.with_post do %>
      <%= link_to blog_post.name, blog_post.url %>
    <% end %>
  <% end %>
<% end %>
```

Component slots

Definition

```
Untitled-1

# blog_component.rb
class BlogComponent < ViewComponent::Base
  # Since `HeaderComponent` is nested inside of this component, we
  # have to
  # reference it as a string instead of a class name.
  renders_one :header, "HeaderComponent"

  # `PostComponent` is defined in another file, so we can refer to it
  # by class name.
  renders_many :posts, PostComponent

  class HeaderComponent < ViewComponent::Base
    attr_reader :classes

    def initialize(classes:)
      @classes = classes
    end

    def call
      content_tag :h1, content, {class: classes}
    end
  end
end
```

Usage

```
Untitled-1

<%=# index.html.erb %>
<%= render BlogComponent.new do |component| %>
  <% component.with_header(classes: "") do %>
    <%= link_to "My Site", root_path %>
  <% end %>

  <% component.with_post(title: "My blog post") do %>
    Really interesting stuff.
  <% end %>

  <% component.with_post(title: "Another post!") do %>
    Blog every day.
  <% end %>
<% end %>
```

Testing

```
require "test_helper"

class ExampleComponentTest < ViewComponent::TestCase
  def test_render_component
    render_inline(ExampleComponent.new(title: "my title")) { "Hello, World!" }

    assert_component_rendered

    assert_selector("span[title='my title']", text: "Hello, World!")
    # or, to just assert against the text:
    assert_text("Hello, World!")
  end
end
```

System tests

```
require "application_system_test_case"

class ButtonComponentSystemTest < ApplicationSystemTestCase
  def with_preview(path)
    visit "/rails/view_components/#{path}"
    wait_for_javascript
  end

  def test_disable_with_actions
    with_preview("button_component/disable_with_actions")

    assert_no_selector ".Polaris-Button--loading"

    # Disable
    click_on "Disable"
    assert_selector ".Polaris-Button--loading > .Polaris-Button__Content > .Polaris-Button__Spinner"

    # Enable
    click_on "Enable"
    assert_no_selector ".Polaris-Button--loading"
  end
end
```

More features on
viewcomponent.org

What can you build with components?

Primitives (aka lego bricks)

Buttons

```
<%= polaris_button(primary: true) { "Save theme" } %>
```

Form fields

```
<%= polaris_text_field(name: :store_name, value: 'Jaded Pixel') %>
```

Text wrappers

```
<%= polaris_text(variant: :bodyMd, as: :p, font_weight: :bold) do %>  
  Sales this year  
<% end %>
```

Card wrappers

```
<%= polaris_card(title: "Online store dashboard") do %>  
  <p>View a summary of your online store's performance.</p>  
<% end %>
```

Icons

```
<%= polaris_icon(name: "PlusCircleIcon") %>
```

Tooltips

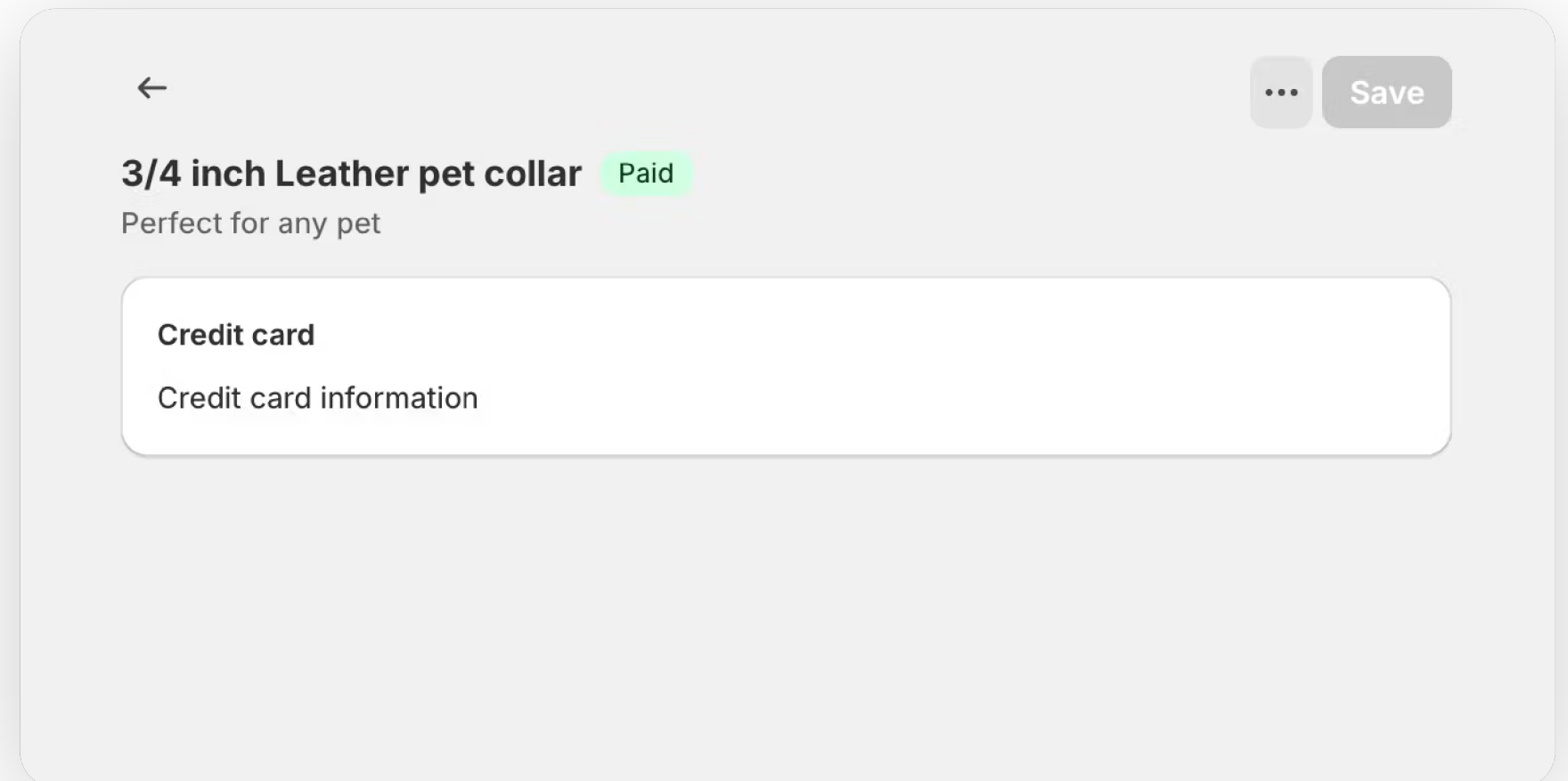
```
<%= polaris_tooltip(text: "I'm a tooltip") do %>  
  Hover over me  
<% end %>
```

Organizational components

Code

```
<%= polaris_page(  
  title: "3/4 inch Leather pet collar",  
  subtitle: "Perfect for any pet",  
  next_url: "/product/2",  
  secondary_actions: [  
    { content: "Duplicate", url: "/duplicate" },  
  ]  
) do |page| %>  
  <% page.with_primary_action(disabled: true) { "Save" } %>  
  
  <%= polaris_card do %>  
    <p>...</p>  
  <% end %>  
<% end %>
```

Result



← ... Save

3/4 inch Leather pet collar Paid

Perfect for any pet

Credit card

Credit card information

Organizational components

Code

```
<%= polaris_layout do |layout| %>
  <% layout.with_section do %>
    <%= polaris_card(title: "Order details", sectioned: true) do %>
      <p>
        Use to follow a normal section with a secondary section to
        create...
      </p>
    <% end %>
  <% end %>

  <% layout.with_section(secondary: true) do %>
    <%= polaris_card(title: "Tags", sectioned: true) do %>
      <p>Add tags to your order.</p>
    <% end %>
  <% end %>
<% end %>
```

Result

Order details

Use to follow a normal section with a secondary section to create a 2/3 + 1/3 layout on detail pages (such as individual product or order pages). Can also be used on any page that needs to structure a lot of content. This layout stacks the columns on small screens.

Tags

Add tags to your order.

Form builders

Code

```
<%= form_with(model: product, builder: Polaris::FormBuilder) do |form| %>
  <%= polaris_form_layout do |form_layout| %>
    <%= form_layout.with_item do %>
      <%= form.errors_summary(within: :page) %>
    <% end %>

    <%= form_layout.with_item do %>
      <%= form.polaris_text_field(:title) %>
    <% end %>|

    <% form_layout.with_item do %>
      <%= form.polaris_dropzone :file_one, multiple: false %>
    <% end %>

    <% form_layout.with_item do %>
      <%= form.polaris_select(:status,
        options: { "Active" => "active", "Draft" => "draft" },
      ) %>
    <% end %>

    <% form_layout.with_item do %>
      <%= polaris_button(submit: true) { "Submit" } %>
    <% end %>
  <% end %>
<% end %>
```

Result

⚠ There is 1 error with this product:

- Title can't be blank

Title

ⓘ Title can't be blank

↑

Add file

or drop files to upload

Status

Active

Submit

Form builders

Errors banner

```
def errors_summary(within: :container)
  return if object.blank?
  return unless object.errors.any?

  title = I18n.t(
    "polaris.form_builder.errors_summary",
    count: object.errors.count,
    model: object.class.model_name.human.downcase
  )

  render Polaris::BannerComponent.new(
    title: title,
    status: :critical,
    within: within,
    data: {errors_summary: true}
  ) do |banner|
    [
      render(Polaris::ListComponent.new) do |list|
        object.errors.full_messages.each do |error|
          list.with_item { error.html_safe }
        end
      end,
      (template.capture { yield(banner) } if block_given?)
    ].compact.join.html_safe
  end
end
```

Form fields

```
def polaris_text_field(method, **options, &block)
  options[:error] ||= error_for(method)
  if options[:error_hidden] && options[:error]
    options[:error] = !!options[:error]
  end
  render Polaris::TextFieldComponent.new(form: self, attribute:
method, **options), &block
end

def polaris_select(method, **options, &block)
  options[:error] ||= error_for(method)
  if options[:error_hidden] && options[:error]
    options[:error] = !!options[:error]
  end
  value = object&.public_send(method)
  if value.present?
    options[:selected] = value
  end
  render Polaris::SelectComponent.new(form: self, attribute:
method, **options, &block)
end
```

Complex DSLs

Code

```
<% data = [
  { product: "Emerald Silk Gown", price: "$875.00", sku: 124689, qty: 140, sales:
"$122,500.00" }, # ...
] %>

<%= polaris_card(sectioned: false) do %>
  <%= polaris_data_table(data, totals_in_header: true) do |table| %>
    <% table.with_column("Product", total: "Totals") do |row| %>
      <%= row[:product] %>
    <% end %>
    <% table.with_column("Price", numeric: true) do |row| %>
      <%= row[:price] %>
    <% end %>
    <% table.with_column("SKU Number", numeric: true) do |row| %>
      <%= row[:sku] %>
    <% end %>
    <% table.with_column("Net quantity", numeric: true, total: 255) do |row| %>
      <%= row[:qty] %>
    <% end %>
    <% table.with_column("Net sales", numeric: true, total: "$2000") do |row| %>
      <%= row[:sales] %>
    <% end %>
  <% end %>
<% end %>
```

Result

Product	Price	SKU Number	Net quantity	Net sales
Totals			255	\$155,830.00
Emerald Silk Gown	\$875.00	124689	140	\$122,500.00
Mauve Cashmere Scarf	\$230.00	124533	83	\$19,090.00
Navy Merino Wool Blazer with khaki chinos and yellow belt	\$445.00	124518	32	\$14,240.00

Complex DSLs

Component classes

```
class Polaris::DataTableComponent < Polaris::Component
  renders_many :columns, ->(title, **system_arguments, &block) do
    DataTable::ColumnComponent.new(title, **system_arguments, &block)
  end

  def initialize(data, **system_arguments)
    @data = data
    # ...
  end

  def render_cell(**arguments, &block)
    render(DataTable::CellComponent.new(**arguments), &block)
  end
end
```

```
class Polaris::DataTable::ColumnComponent < Polaris::Component
  attr_reader :title

  def initialize(title, **system_arguments, &block)
    @title = title
  end

  def call(row)
    @block.call(row)
  end
end
```

View

```
<table class="Polaris-DataTable__Table">
  <thead>
    <tr>
      <% columns.each_with_index do |column, index| %>
        <%= render_cell(tag: "th", scope: "col",
**column.system_arguments) do %>
          <%= column.title %>
        <% end %>
      <% end %>
    </tr>
  </thead>
  <tbody>
    <% @data.each do |row| %>
      <%= render Polaris::BaseComponent.new(**row_arguments(row)) do %>
        <% columns.each_with_index do |column, index| %>
          <%= render_cell(tag: "td", **column.system_arguments) do %>
            <%= column.call(row) %>
          <% end %>
        <% end %>
      <% end %>
    <% end %>
  </tbody>
</table>
```

Practical tips

Consider the stage of business

Helpers as aliases

Before

```
<%= render(Polaris::ButtonComponent.new) { "Button name" } %>
```

After

```
<%= polaris_button { "Add product" } %>
```

```
module Polaris
  module ViewHelper
    POLARIS_HELPERS = {
      button: "Polaris::ButtonComponent",
      # ...
    }
    POLARIS_HELPERS.each do |name, component|
      define_method :"polaris_#{name}" do |*args, **kwargs, &block|
        render component.constantize.new(*args, **kwargs), &block
      end
    end
  end
end
```


Use Rails engines

Use Lookbook

lookbook.build

The image displays the Lookbook web application interface, which is used for documenting and previewing UI components. The interface is divided into several sections:

- Code Editor (Left):** Shows the source code for a button component. The code includes comments and a `render` method that uses `ButtonComponent.new` to create a button with the text "Click me".
- Component List (Middle-Left):** A sidebar showing a tree view of components. The "Button" component is selected, and its sub-components (Playground, Themes, Header, Simple list, Feedback, Alert, Warning, Success, Danger, Blank Slate, Navigation, Navbar, Phlex Example) are listed.
- Preview (Middle-Right):** A large area showing the rendered output of the selected component. It displays three button styles: "Primary (default)" (green), "Secondary" (gray), and "Danger" (red). Each button has a corresponding label: "A primary action", "A less important action", and "A dangerous action".
- Warning Alert (Bottom-Right):** A detailed view of a "Warning alert" component. It shows a yellow warning box with the text "This is a warning". Below the preview, there is a "Warning alert" section with a description: "Used to notify users about some potentially hard-to-undo consequences to an action." and a note: "Don't use this for form validation errors! Use the form-specific messaging component instead."

The interface also includes a top navigation bar with the "LOOKBOOK DEMO" logo, a search bar, and a theme selector (light/dark). The bottom of the interface features a tabbed view for "Source", "Notes", "Params", and "Assets".

Take inspiration from open-source:

- [Primer ViewComponents](#)
- [GOV.UK Rails Components](#)
- [Polaris ViewComponents](#)
- [Avo Admin](#)
- [view component-contrib](#)

Beware, it's addictive 😄💧

The end. Questions?



kirillplatonov.com/talks/viewcomponents