

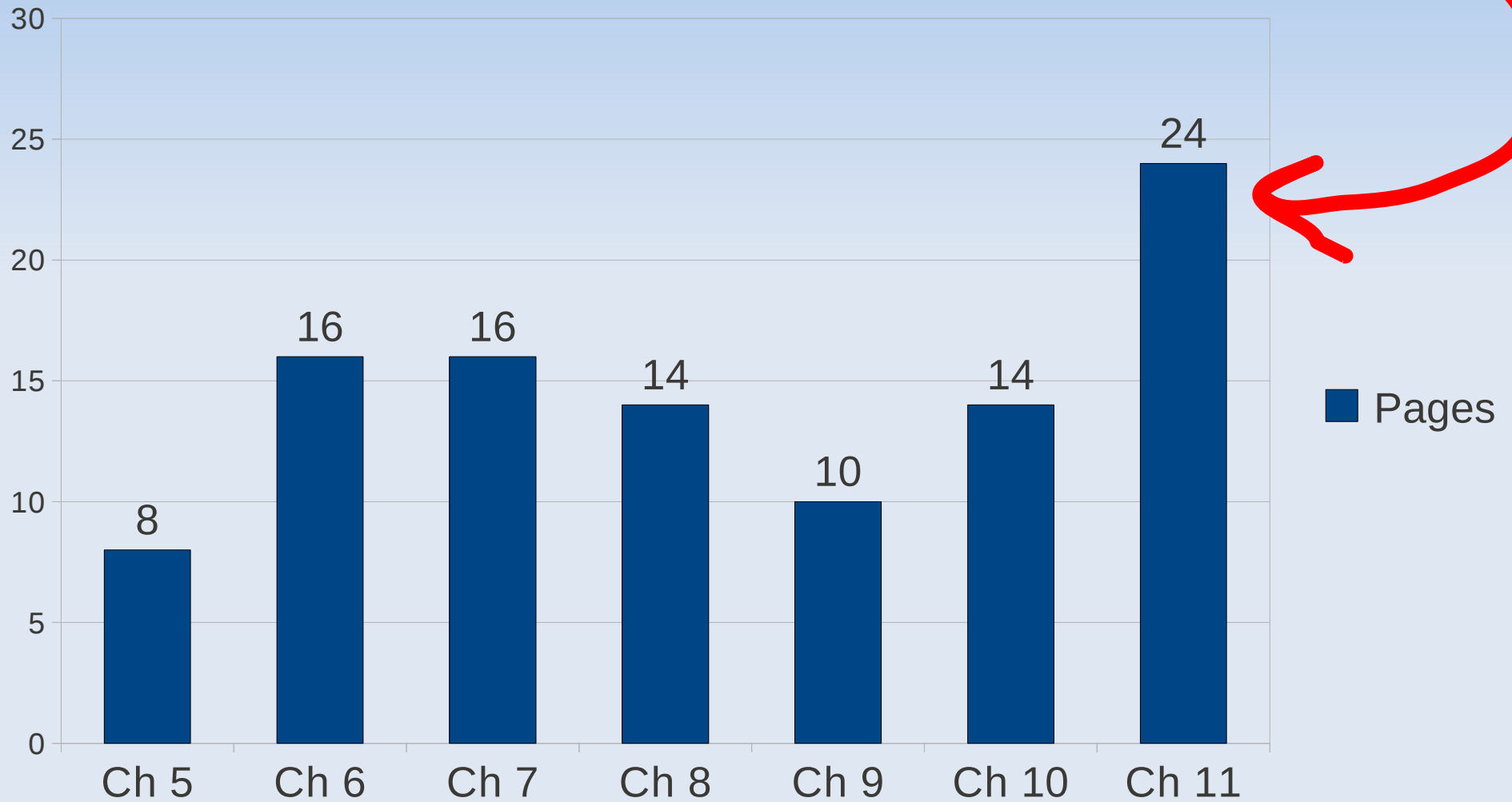
# **Agile Web Development with Rails, Chapter 11: Task F: Add a Dash of Ajax**

David Grayson

Based on the book by Sam Ruby.

Las Vegas Ruby Meetup, 2012-01-07

# This was a long chapter!



# Iteration F1: Moving the Cart

- Our goal:

The screenshot shows the Pragmatic Bookshelf website header with the logo and title 'PRAGMATIC BOOKSHELF'. A dark green sidebar on the left contains links for Home, Questions, News, and Contact. A white box in the top right corner, titled 'Your Cart', is circled in red. It displays '1x CoffeeScript \$36.00' and 'Total \$36.00' with a button labeled 'Empty cart' below it.

The screenshot shows the Pragmatic Bookshelf website header with the logo and title 'PRAGMATIC BOOKSHELF'. A dark green sidebar on the left now contains the 'Your Cart' information, including '1x CoffeeScript \$36.00', 'Total \$36.00', and an 'Empty cart' button. Below the cart are links for Home, Questions, and News. The main content area is titled 'Your Pragmatic Catalog' and features a product listing for 'CoffeeScript' with a description and a book cover image. A red arrow points from the top-right cart in the previous screenshot to the top-left cart in this one.

# Iteration F1: Moving the Cart

- app/views/carts/show.html.erb:

```
<%= render @cart %>
```

- app/views/layouts/application.html.erb:

```
<!DOCTYPE html>
<html>
...
<body class="<%= controller.controller_name %>">
  <div id="banner">...</div>
</div>
  <div id="columns">
    <div id="side">
      <div id="cart"><%= render @cart %></div>
      ...
    </div>
    <div id="main"><%= yield %></div>
  </div>
</body></html>
```

# Iteration F1: Moving the Cart

- "render @cart" renders app/views/carts/\_cart.html.erb:

```
<div class="cart_title">Your Cart</div>
<table>
  <%= render cart.line_items %>
  <tr class="total_line">
    <td colspan="2">Total</td>
    <td class="total_cell">
      <%= number_to_currency cart.total_price %>
    </td>
  </tr>
</table>

<%= button_to 'Empty cart', cart, method: :delete,
  confirm: 'Are you sure?' %>
```

# Iteration F1: Moving the Cart

- Also need to set `@cart` in the `StoreController#index`:

```
class StoreController < ApplicationController
  def index
    @products = Product.order(:title)
    @cart = current_cart
  end
end
```

- Also need to adjust the stylesheet so the cart looks good in both contexts.

# Iteration F1: Moving the Cart

- `app/controllers/line_items_controller.rb`:

```
def create
  @cart = current_cart
  product = Product.find(params[:product_id])
  @line_item = @cart.add_product(product.id)
  respond_to do |format|
    if @line_item.save
      format.html { redirect_to store_url }
      format.json { render json: @line_item,
        status: :created, location: @line_item }
    else
      format.html { render action: "new" }
      format.json { render json: @line_item.errors,
        status: :unprocessable_entity }
    end
  end
end
```

# Iteration F1: Moving the Cart

- Done with Iteration F1!



The screenshot shows the Pragmatic Bookshelf website. The top navigation bar is green with the Pragmatic Bookshelf logo on the left and the text "PRAGMATIC BOOKSHELF" in green on the right. Below the navigation bar, the page is split into two main sections. On the left is a dark green sidebar containing a "Your Cart" section with a table showing "1x CoffeeScript \$36.00" and a "Total \$36.00". Below the cart is a button labeled "Empty cart" and a list of links: "Home", "Questions", "News", and "Contact". On the right is the "Your Pragmatic Catalog" section, which features a product listing for "CoffeeScript". The listing includes a small image of the book cover, the title "CoffeeScript", a description: "CoffeeScript is JavaScript done right. It provides all of JavaScript's functionality wrapped in a cleaner, more succinct syntax. In the first book on this exciting new language, CoffeeScript guru Trevor Burnham shows you how to hold onto all the power and flexibility of JavaScript while writing clearer, cleaner, and safer code.", the price "\$36.00", and an "Add to Cart" button. Below the CoffeeScript listing, the top of another listing for "Programming Ruby 1.9" is visible.

**Pragmatic Bookshelf**

## PRAGMATIC BOOKSHELF

### Your Cart

1x CoffeeScript	\$36.00
<b>Total</b>	<b>\$36.00</b>

[Empty cart](#)

[Home](#)  
[Questions](#)  
[News](#)  
[Contact](#)

### Your Pragmatic Catalog

---

 **CoffeeScript**

CoffeeScript is JavaScript done right. It provides all of JavaScript's functionality wrapped in a cleaner, more succinct syntax. In the first book on this exciting new language, CoffeeScript guru Trevor Burnham shows you how to hold onto all the power and flexibility of JavaScript while writing clearer, cleaner, and safer code.

**\$36.00** [Add to Cart](#)

---

 **Programming Ruby 1.9**



# Iteration F2: Creating an Ajax cart

- Ajaxify the button simply by saying `remote:true` in `app/views/store/index.html.erb`:

```
<h1>Your Pragmatic Catalog</h1>
<% @products.each do |product| %>
  <div class="entry">
    <%= image_tag(product.image_url) %>
    <h3><%= product.title %></h3>
    <%= sanitize(product.description) %>
    <div class="price_line">
      <span class="price"><%=
number_to_currency(product.price) %></span>
      <%= button_to 'Add to Cart',
        line_items_path(product_id: product),
          remote: true %>
    </div>
  </div>
</div>
<% end %>
```

# Iteration F2: Creating an Ajax Cart

- Prevent the create action from redirecting in `line_items_controller.rb`:

```
def create
  @cart = current_cart
  product = Product.find(params[:product_id])
  @line_item = @cart.add_product(product.id)

  respond_to do |format|
    if @line_item.save
      format.html { redirect_to store_url }
      format.js
      format.json { render json: @line_item,
        status: :created, location: @line_item }
    else
      format.html { render action: "new" }
      format.json { render json: @line_item.errors,
        status: :unprocessable_entity }
    end
  end
end
end
```

# Iteration F2: Creating an Ajax Cart

- Instead of redirecting, Rails renders `app/views/line_items/create.js.erb`:

```
$('#cart').html("<%= j render @cart %>");
```

# Iteration F3: Highlighting Changes

- First, assign an instance variable in `line_items_controller.rb`:

```
def create
  @cart = current_cart
  product = Product.find(params[:product_id])
  @line_item = @cart.add_product(product.id)

  respond_to do |format|
    if @line_item.save
      format.html { redirect_to store_url }
      format.js { @current_item = @line_item }
      format.json { render json: @line_item,
        status: :created, location: @line_item }
    else
      format.html { render action: "new" }
      format.json { render json: @line_item.errors,
        status: :unprocessable_entity }
    end
  end
end
end
```

# Iteration F3: Highlighting Changes

- Modify `app/views/line_items/_line_item.html.erb` to use it:

```
<% if line_item == @current_item %>
<tr id="current_item">
<% else %>
<tr>
<% end %>
  <td><%= line_item.quantity %>&times;</td>
  <td><%= line_item.product.title %></td>
  <td class="item_price">
    <%= number_to_currency(line_item.total_price) %>
  </td>
</tr>
```

# Iteration F3: Highlighting Changes

- Modify `app/views/line_items/create.js.erb` to highlight the new item:

```
$('#cart').html("<%=j render @cart %>");
```

```
$('#current_item').  
  css({'background-color': '#8f8'}).  
  animate({'background-color': '#141'}, 1000);
```

# Iteration F4: Hiding an Empty Cart

- app/views/layouts/application.html.erb:

```
<div id="cart"  
  <% if @cart.line_items.empty? %>  
    style="display: none"  
  <% end %>  
  >  
  <%= render(@cart) %>  
</div>
```

```
<%= hidden_div_if(@cart.line_items.empty?, id: 'cart') do %>  
  <%= render @cart %>  
<% end %>
```

```
#cart{:style=>('display:none;' if @cart.line_items.empty?)}  
  = render @cart
```

# Iteration F4: Hiding an Empty Cart

- Also need to get rid of the "Your cart is currently empty." notice we generated when emptying the cart.



# Iteration F4: Hiding an Empty Cart

- app/views/line\_items/create.js.erb:

```
if ($('#cart tr').length == 1)
  $('#cart').show('blind', 1000);

$('#cart').html("<%=j render @cart %>");

$('#current_item').
  css({'background-color': '#8f8'}).
  animate({'background-color': '#141'}, 1000);
```

# Iteration F5: Making Images Clickable

- `app/assets/javascripts/store.js.coffee:`

```
$ ->  
  $(' .store .entry > img').click ->  
    $(this).parent().find(':submit').click()
```

# Testing Ajax Changes

- First, fix the tests that were breaking.
- Next, test the new Ajax stuff we added.

# Testing Ajax Changes

- test/functional/line\_items\_controller\_test.rb:

```
test "should create line_item via ajax" do
  assert_difference('LineItem.count') do
    xhr :post, :create, product_id: products(:ruby).id
  end

  assert_response :success
  assert_select_jquery :html, '#cart' do
    assert_select 'tr#current_item td', /Programming Ruby 1.9/
  end
end
```

```
$('#cart').html("<%=j render @cart %>");
```

app/views/line\_items/create.js.erb

# Testing Ajax Changes

- test/functional/store\_controller\_test.rb:

```
test "markup needed for store.js.coffee is in place" do
  get :index
  assert_select '.store .entry > img', 3
  assert_select '.entry input[type=submit]', 3
end
```

# The End