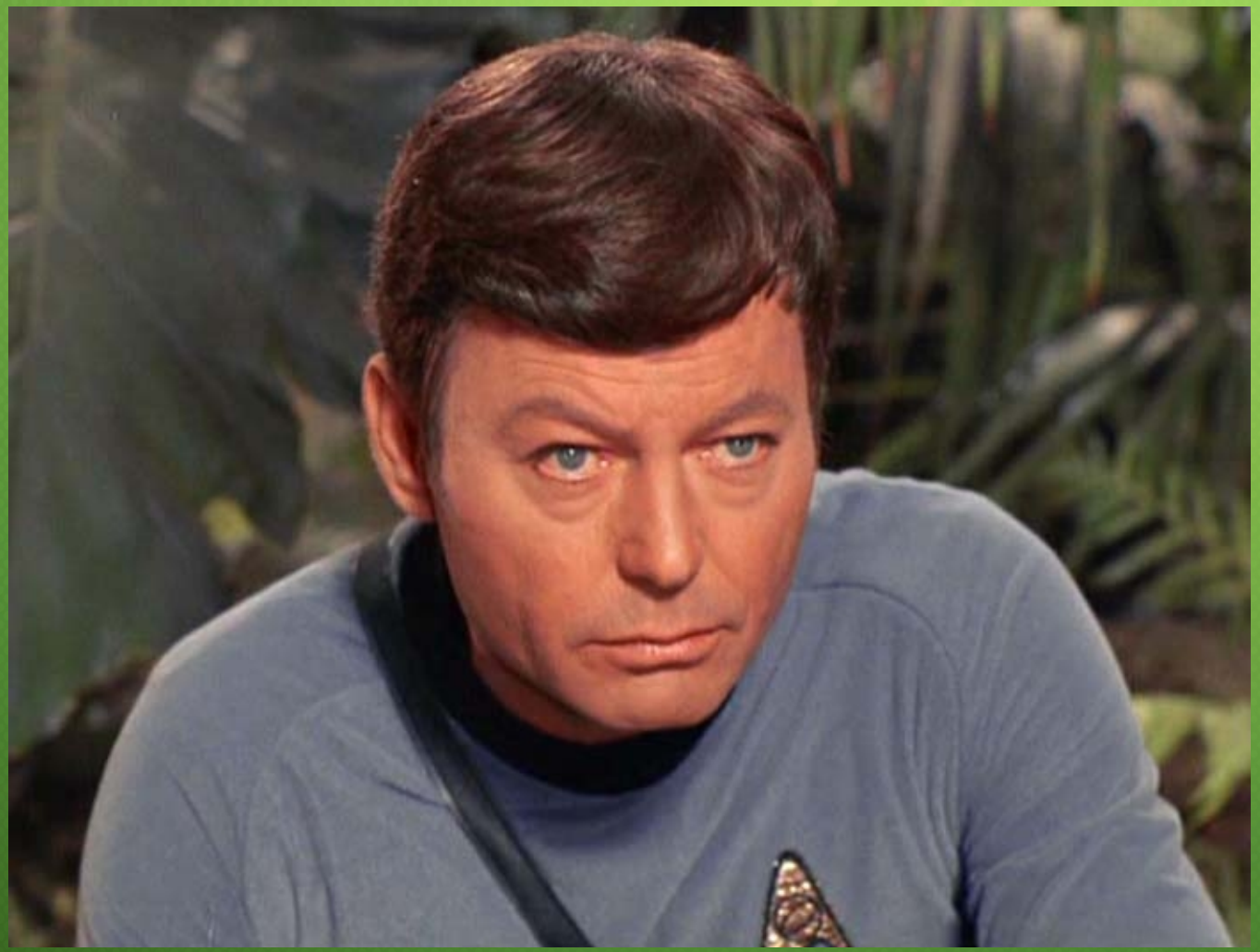
Rails 4

What's new?

Edge Rails Release Notes

http://edgeguides.rubyonrails.org/4_0_release_notes.html

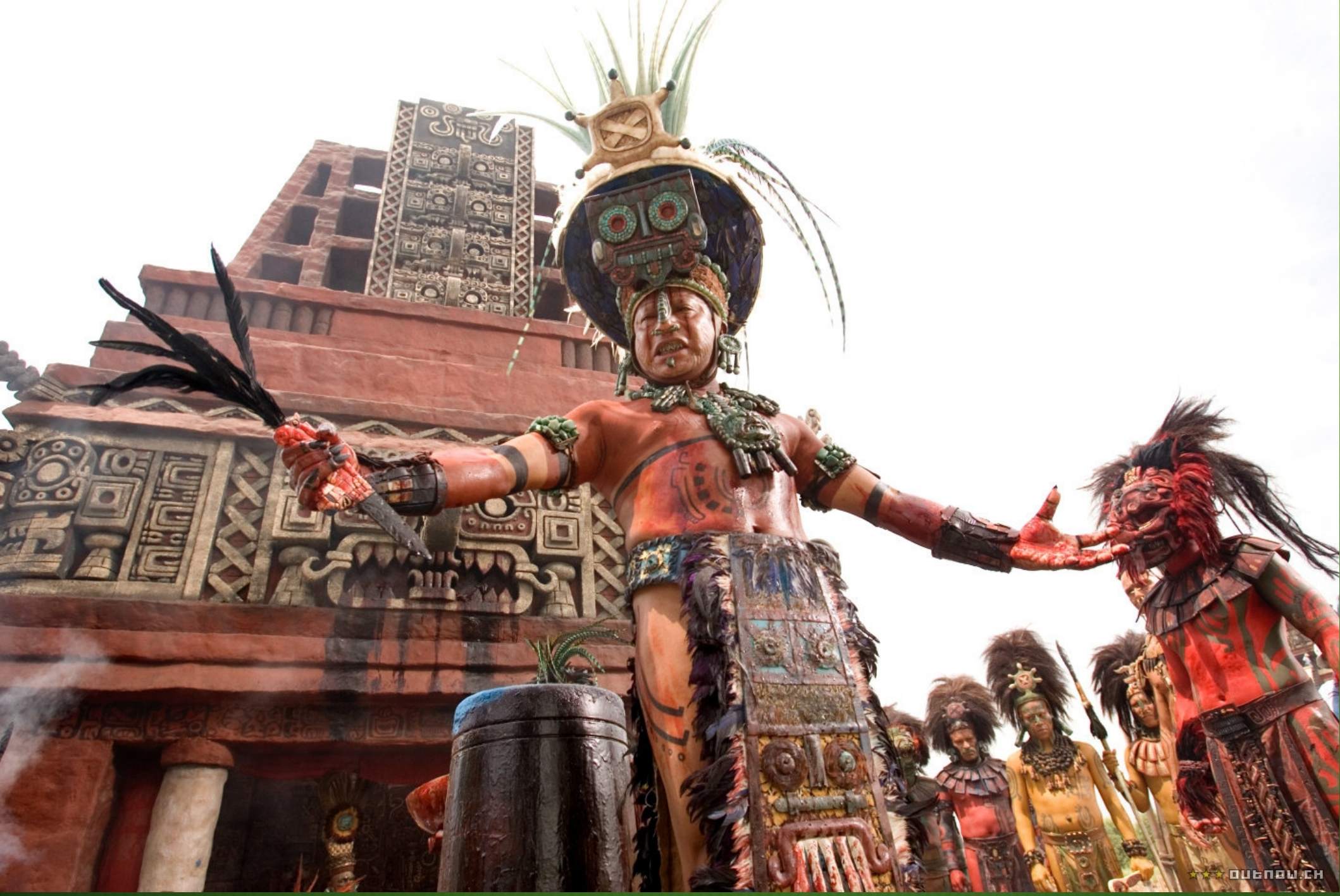
Ruby >= 1.9.3



Ruby 1.8 is dead, Jim.

Ruby 2.0

Feb. 24th 2013



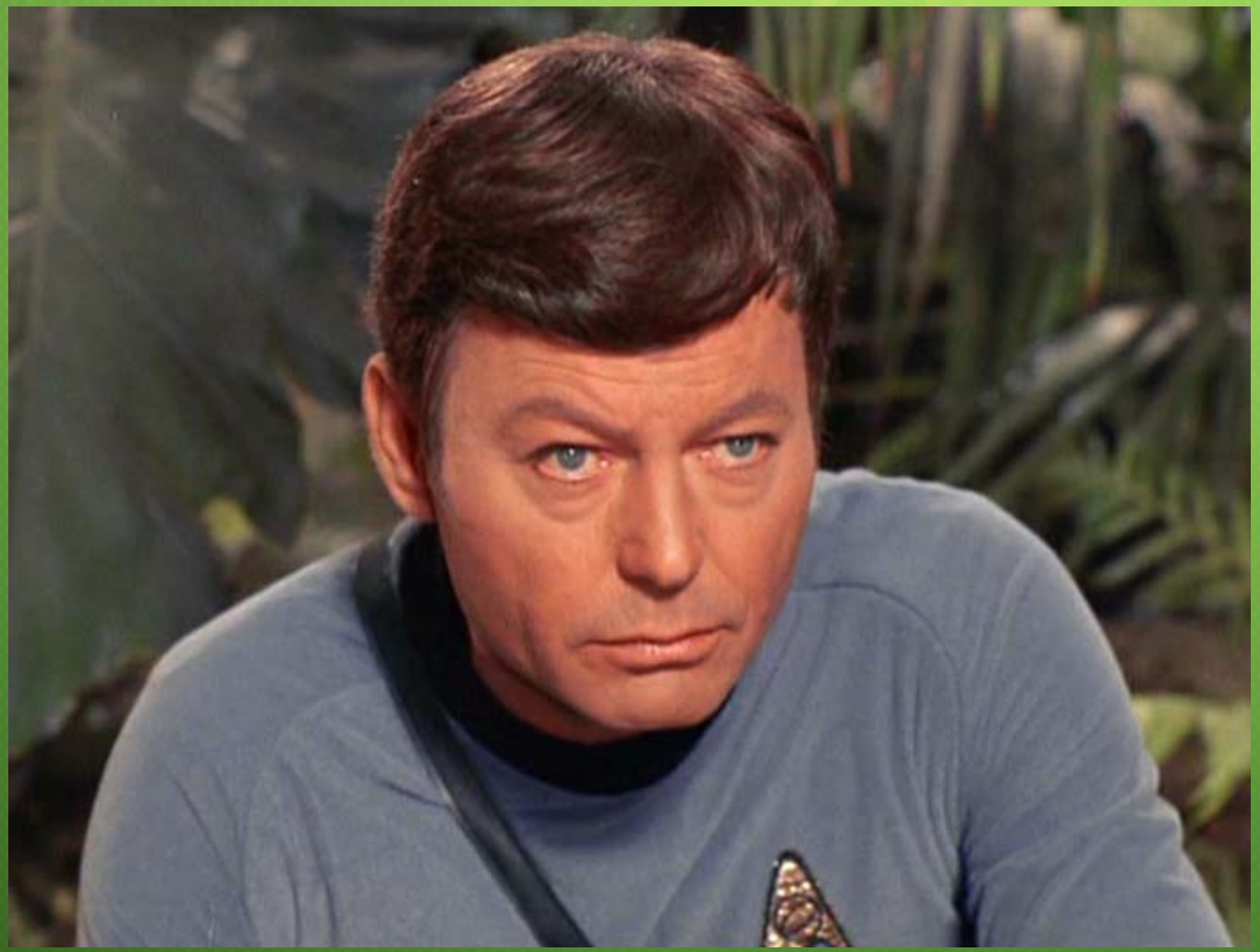
... unless the world ends

Strong Parameters

Mass assignment protection for ActionController with a whitelist approach

https://github.com/rails/strong_parameters

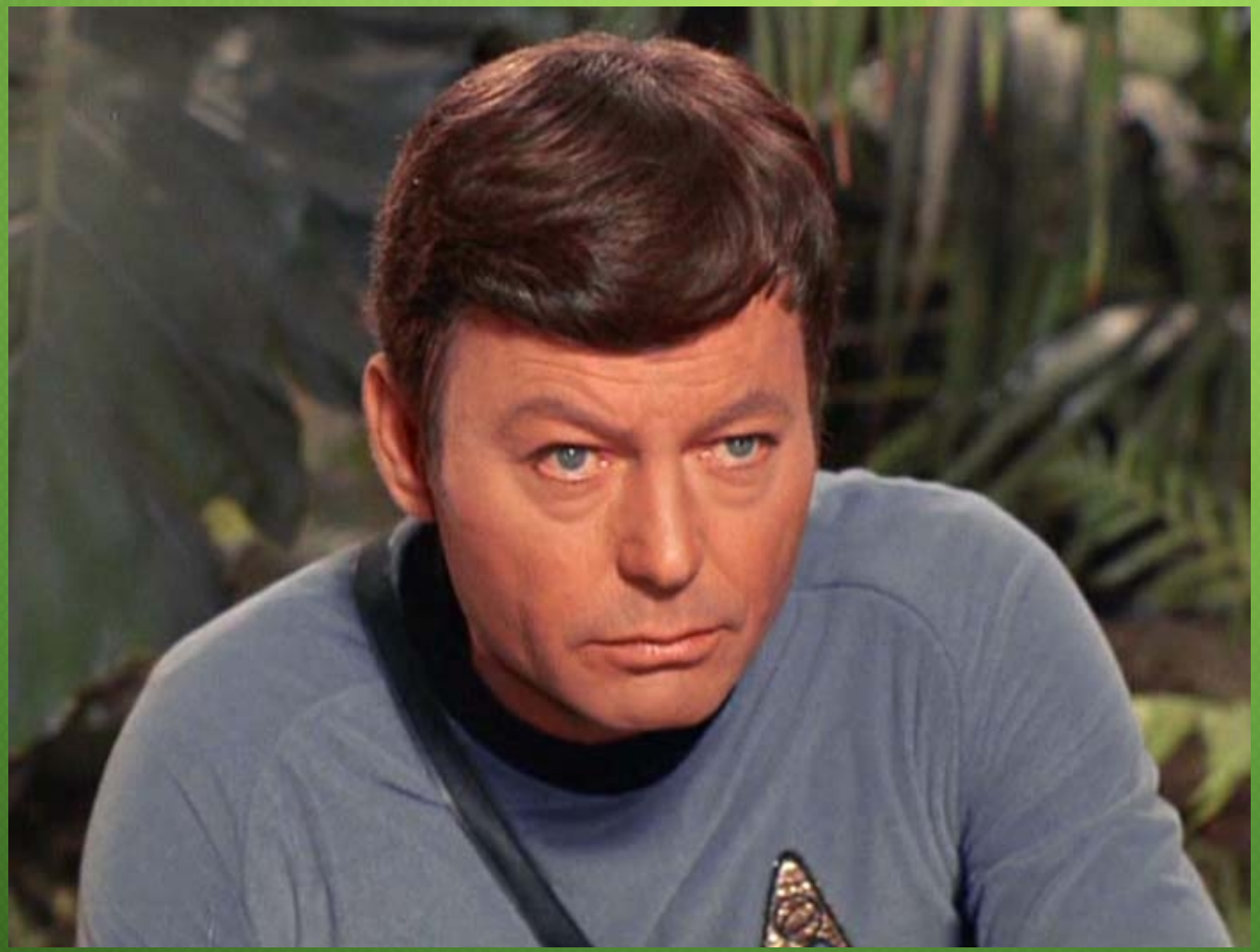
```ruby
class PeopleController < ActionController::Base
  # This will raise an ActiveModel::ForbiddenAttributes exception because it's
  # using mass assignment without an explicit permit step.
  def create
    Person.create(params[:person])
  end

  # This will pass with flying colors as long as there's a person key in the
  # parameters, otherwise it'll raise a ActionController::MissingParameter
  # exception, which will get caught by ActionController::Base and turned into
  # that 400 Bad Request reply.
  def update
    redirect_to current_account.people.find(params[:id]).tap { |person|
      person.update_attributes!(person_params)
    }
  end

  private
    # Using a private method to encapsulate the permissible parameters is just a
    # good pattern since you'll be able to reuse the same permit list between
    # create and update. Also, you can specialize this method with per-user
    # checking of permissible attributes.
    def person_params
      params.require(:person).permit(:name, :age)
    end
end
```



`attr_accessible` **is dead, Jim.**

Rails 3's
mass assignment protection
has been moved to a gem

https://github.com/rails/protected_attributes



`attr_accessible` **is dead, Jim.**

Queue API

Rails.queue
for processing jobs in the background

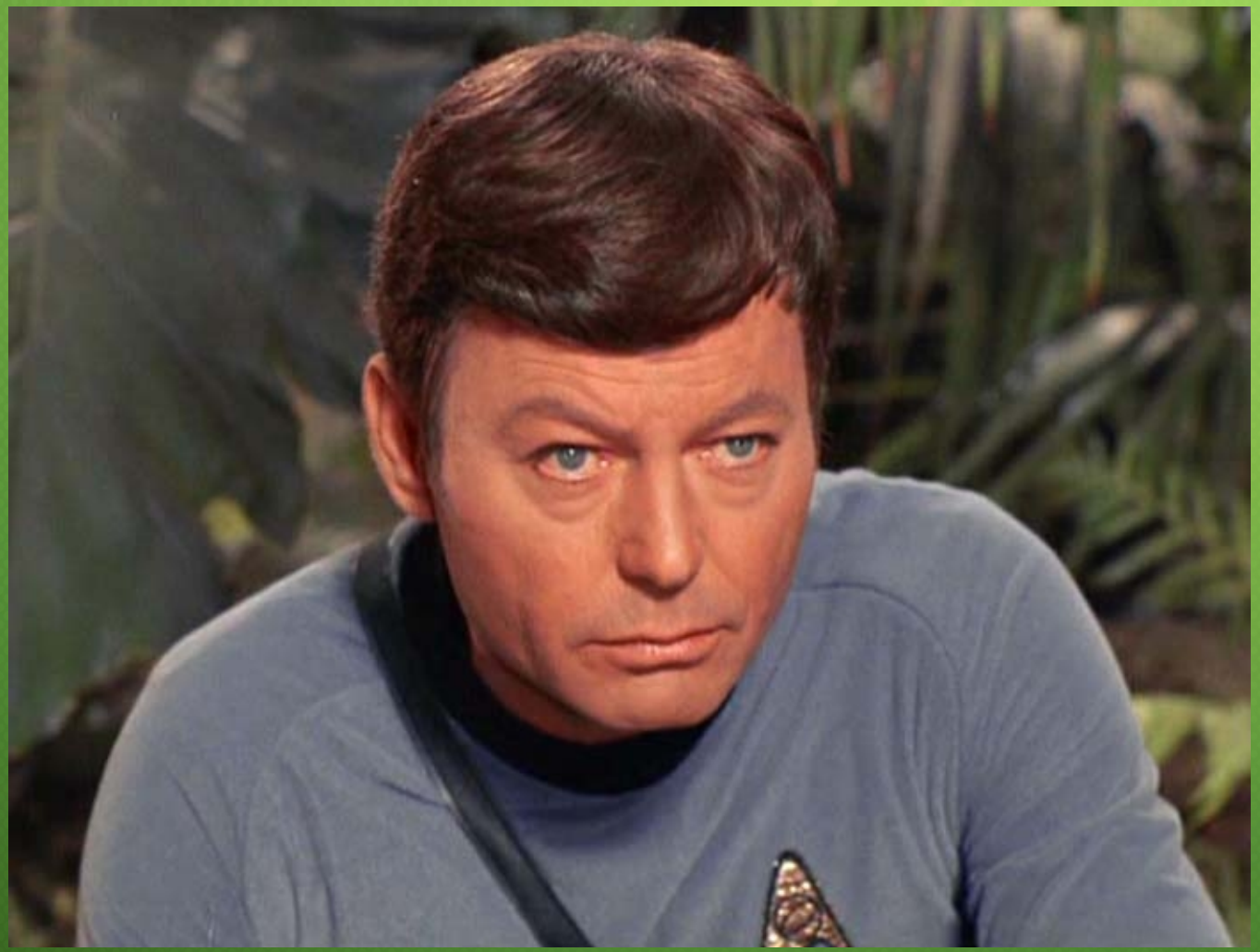
ActiveSupport::Queue

```ruby
# Imagine some arbitrary but expensive code...
class ExpensiveOperation
  # All we need to do is make our object respond to a run method.
  def run
    # ...
  end
end


# Configure your queue related settings in your environment files.
# config/application.rb
#
# Default Synchronous
config.queue = ActiveSupport::SynchronousQueue.new
# Default Threaded
config.queue = ActiveSupport::Queue.new
# Sidekick Queue
config.queue = Sidekiq::Client::Queue.new
# Resque Queue
Config.queue = Resque::Rails::Queue.new


# Just push an instance of your object to the queue.
Rails.queue.push ExpensiveOperation.new
```

This also enables async mailers that send your mails in the background

```
config.action_mailer.async = true
```

Or define queue and async per mailer.



The old mailer is dead, Jim.

Russian Doll Caching

Nested fragment caching that automatically invalidates parent cache values

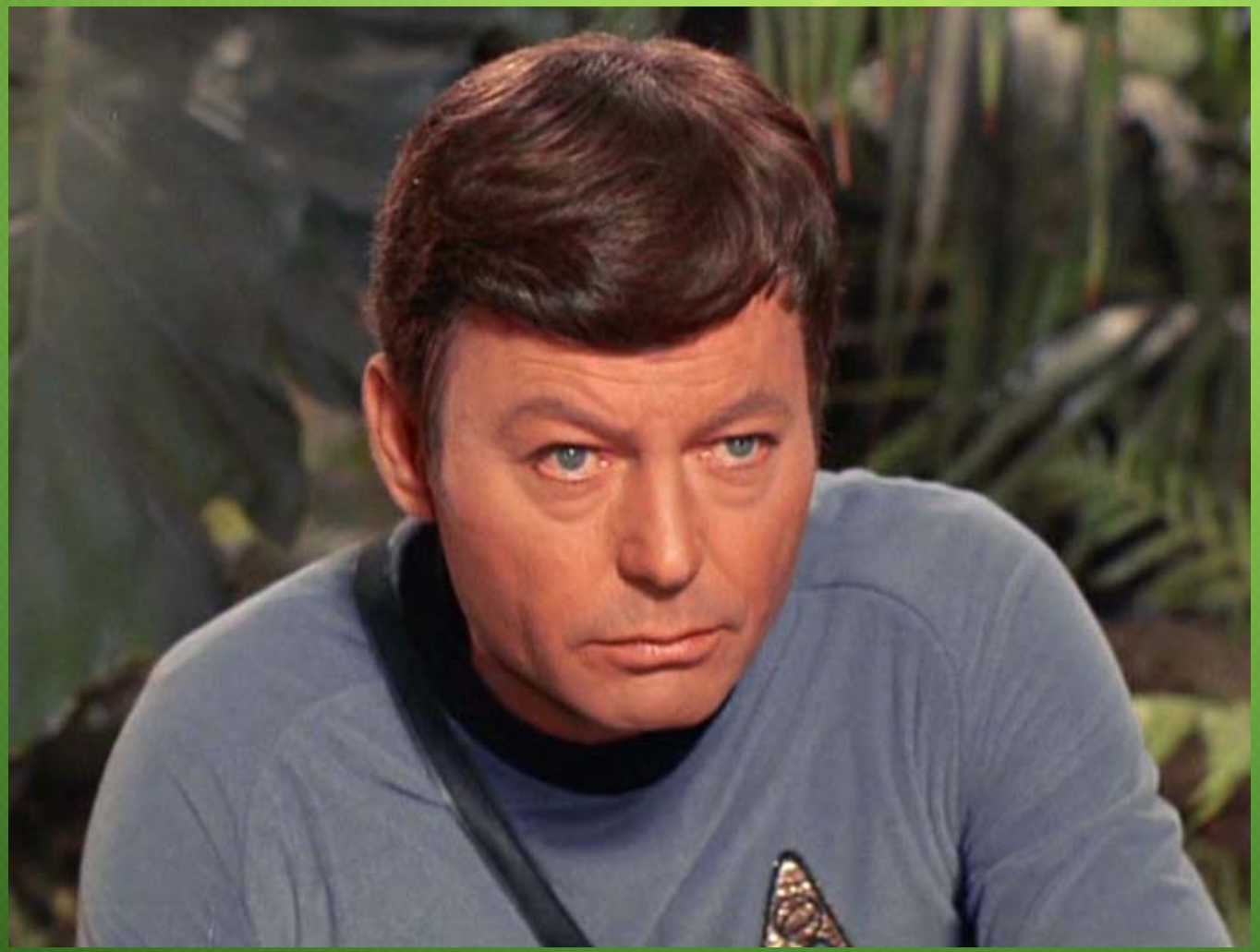
https://github.com/rails/cache_digests

```erb
# app/views/projects/show.html.erb
<% cache [ "v1", project ] do %>
  <h1>All documents</h1>
  <%= render project.documents %>

  <h1>All todolists</h1>
  <%= render project.todolists %>
<% end %>


# app/views/documents/_document.html.erb
<% cache [ "v1", document ] do %>
  My document: <%= document.name %>

  <%= render document.comments %>
<% end %>


# app/views/todolists/_todolist.html.erb
<% cache [ "v1", todolist ] do %>
  My todolist: <%= todolist.name %>

  <%= render document.comments %>
<% end %>


# app/views/comments/_comment.html.erb
<% cache [ "v1", comment ] do %>
  My comment: <%= comment.body %>
<% end %>
```

```erb
# app/views/projects/show.html.erb
<% cache project do %>
  <h1>All documents</h1>
  <%= render project.documents %>

  <h1>All todolists</h1>
  <%= render project.todolists %>
<% end %>


# app/views/documents/_document.html.erb
<% cache document do %>
  My document: <%= document.name %>

  <%= render document.comments %>
<% end %>


# app/views/todolists/_todolist.html.erb
<% cache todolist do %>
  My todolist: <%= todolist.name %>

  <%= render document.comments %>
<% end %>


# app/views/comments/_comment.html.erb
<% cache comment do %>
  My comment: <%= comment.body %>
<% end %>
```

Fragment caching as the default caching strategy

https://github.com/rails/actionpack-action_caching
https://github.com/rails/actionpack-page_caching



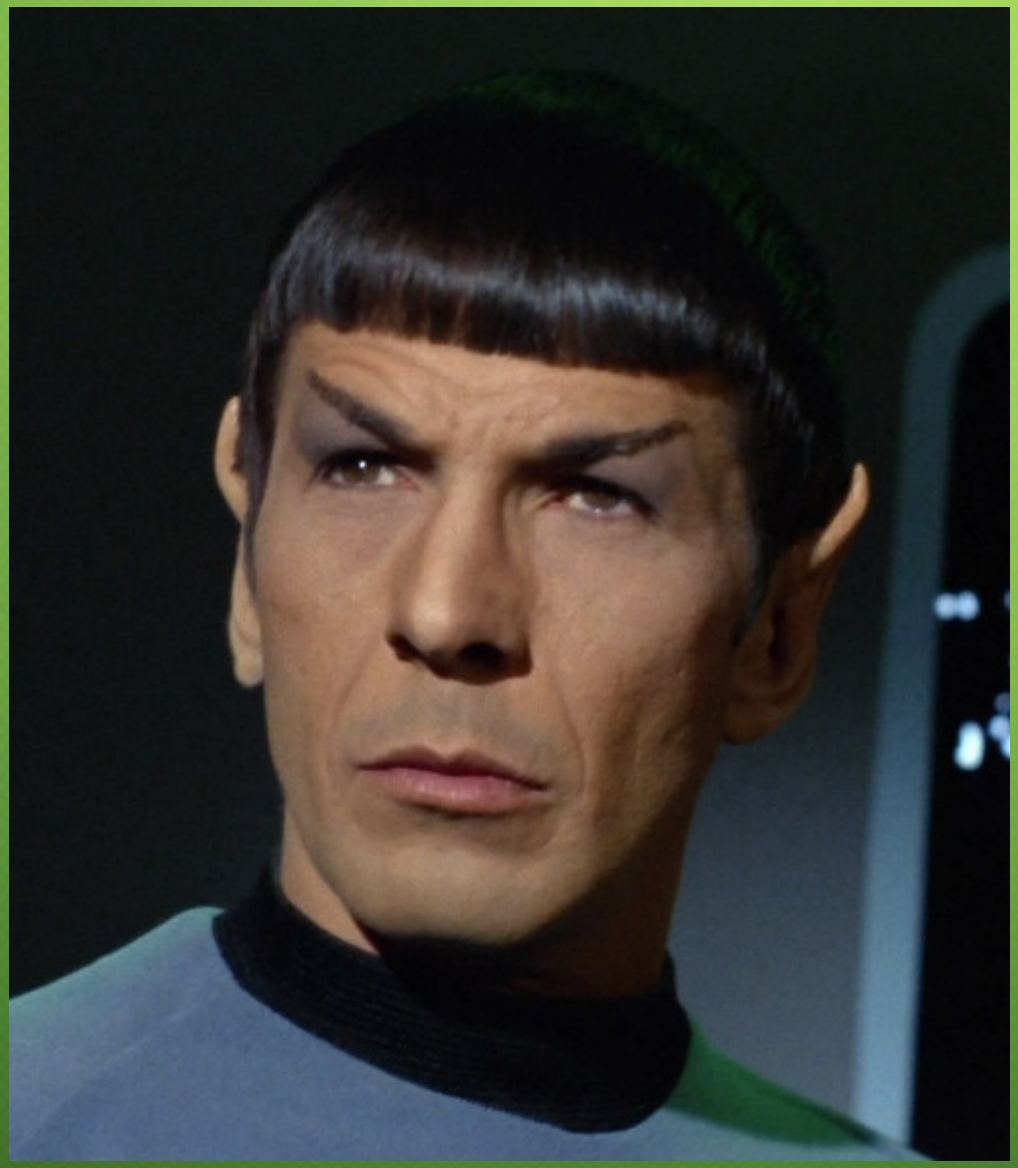
I'm getting bored, Jim.

Turbolinks

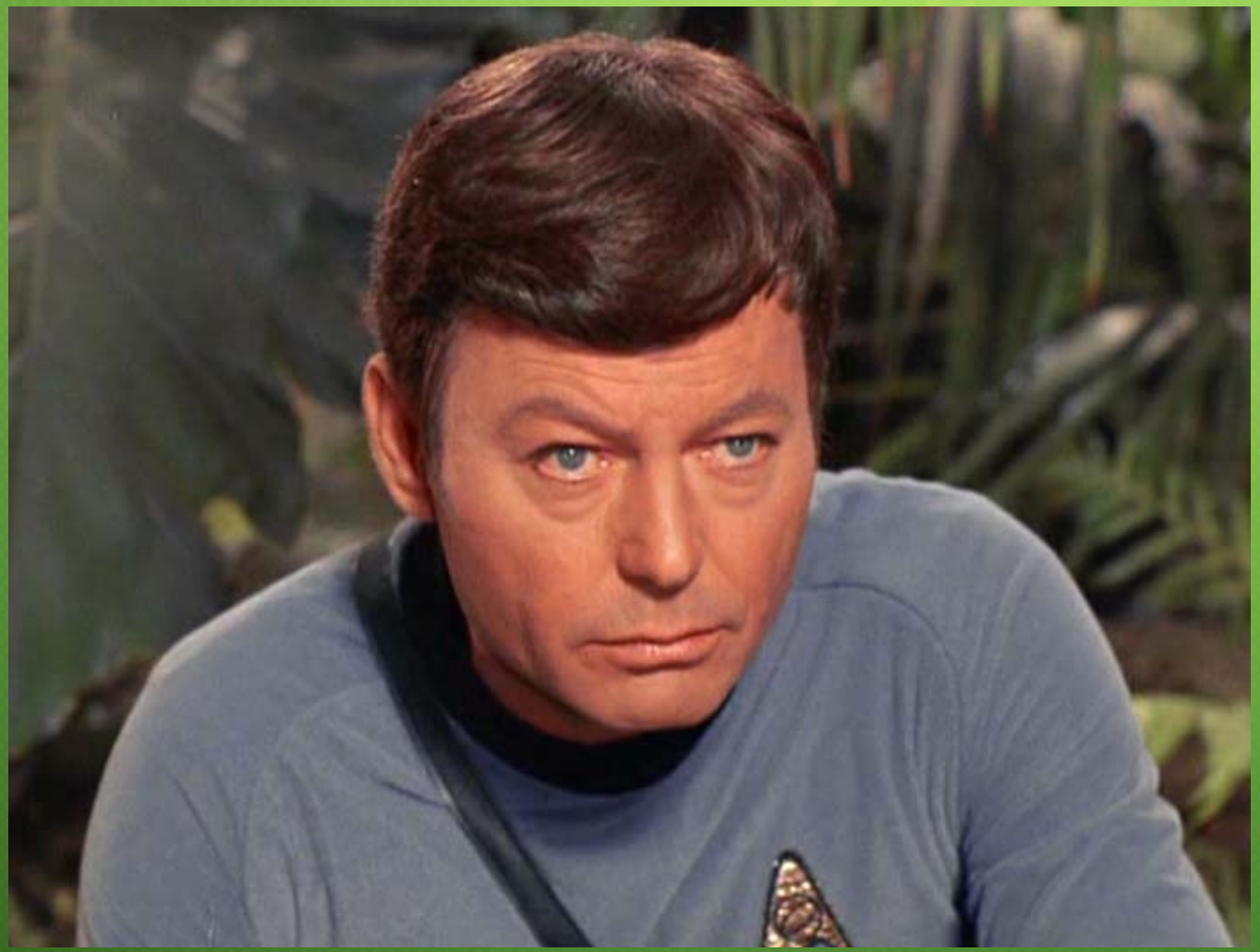
Speed up following links by using JavaScript to replace the page's body and title

https://github.com/rails/turbolinks

```coffeescript
# Rails 3:
$(document).ready ->
  alert "page has loaded"


# Rails 4:
$(document).bind "page:change" ->
  alert "page has loaded"

# Opt out:
<a href="/articles" data-no-turbolink>
  Articles
</a>
```



Fascinating, but is it faster?



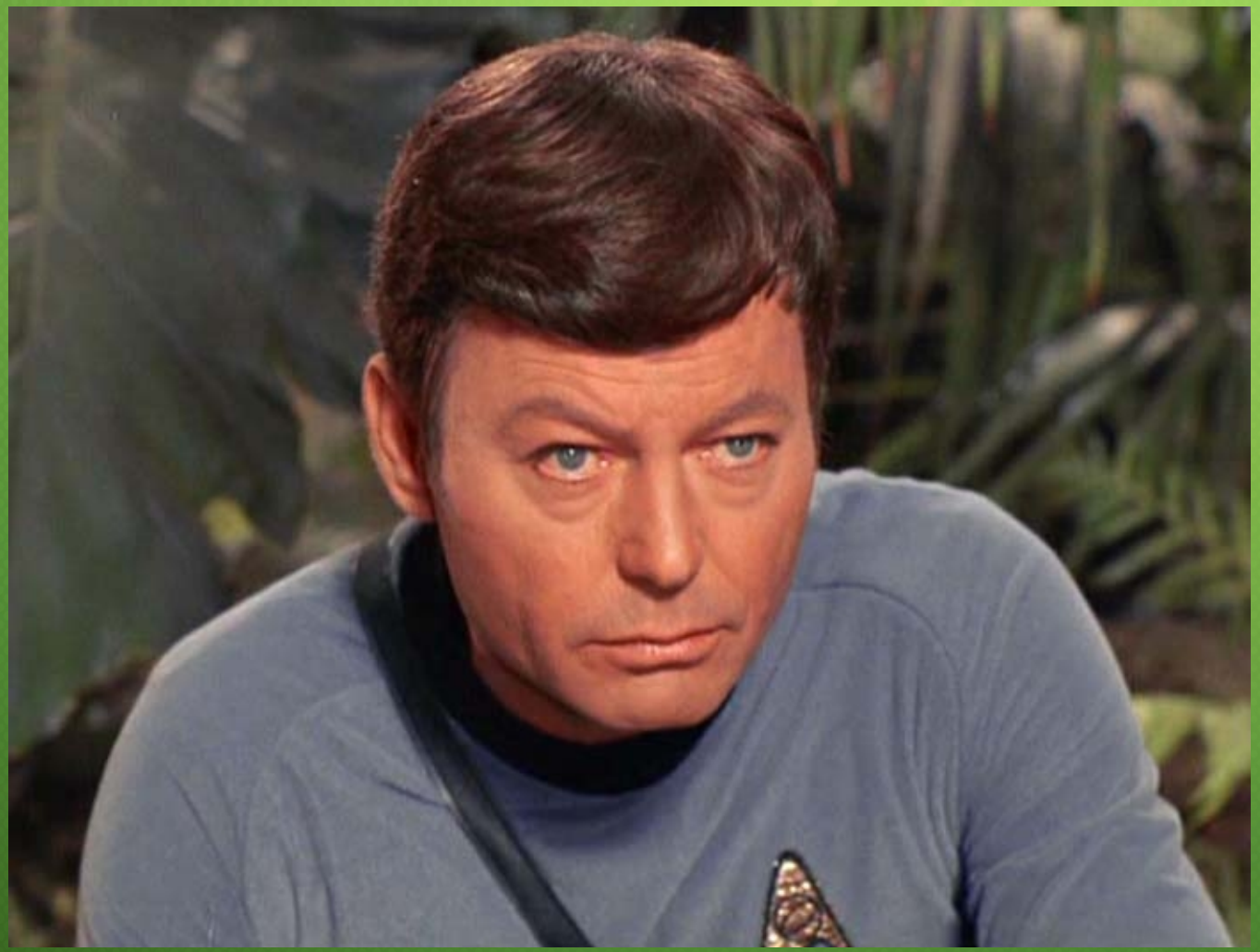
Your experience may vary, Spock.

Live Streaming

Stream arbitrary data at arbitrary intervals

ActionController::Live
http://tenderlovemaking.com/2012/07/30/is-it-live.html

```ruby
# Imagine (again) some arbitrary but expensive code...
class ExpensiveOperation
  def self.run
    # I said IMAGINE!
    sleep 1
    "The devil likes metal!\n"
  end
end


# Now take a controller and include the magic.
class SomeController < ActionController::Base
  include ActionController::Live

  def index
    666.times {
      response.stream.write ExpensiveOperation.run
    }
    response.stream.close
  end
end
```

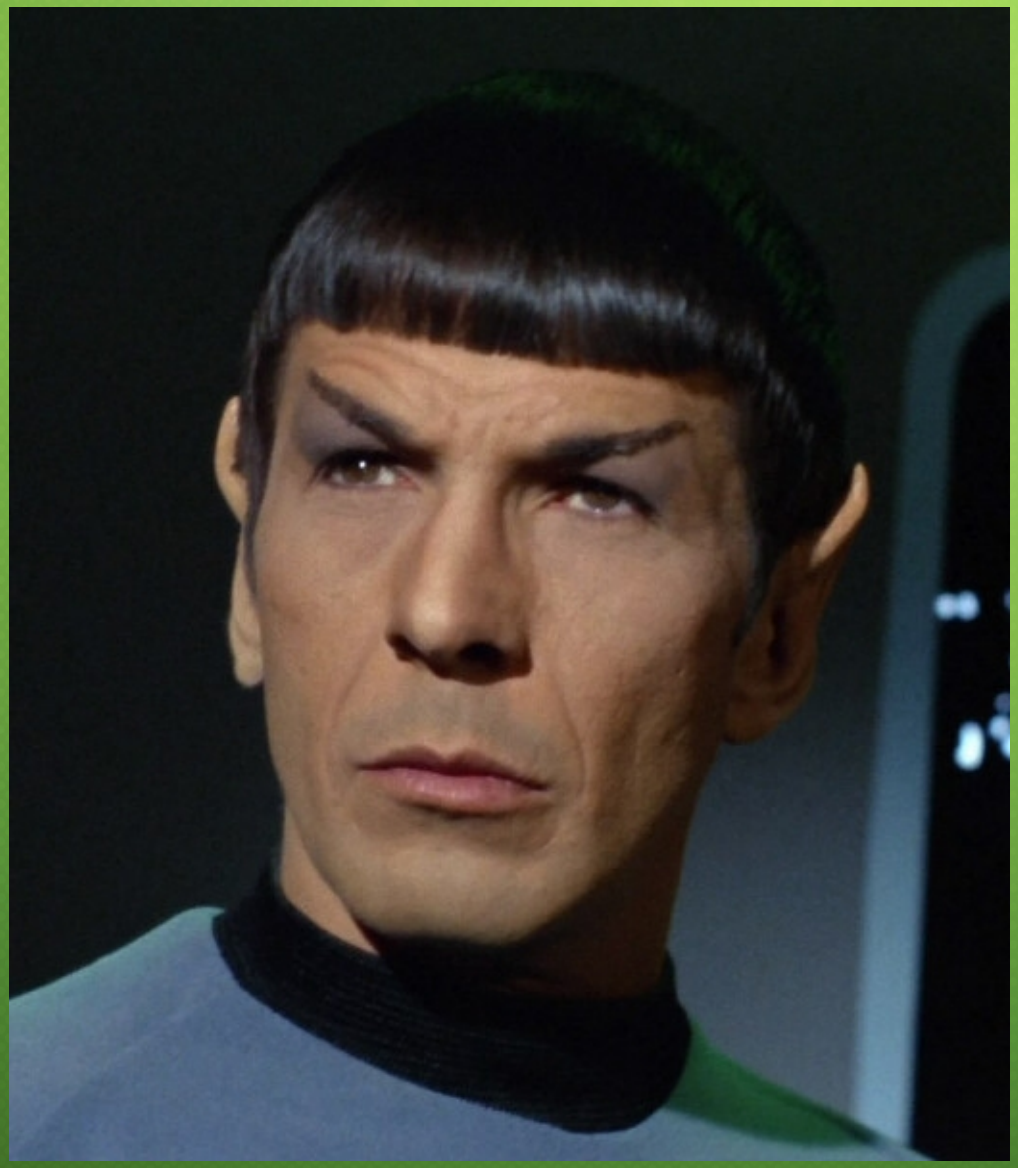


I'm so excited about this, Jim.

HTTP
PATCH

Instead of using PUT
Rails defaults to PATCH
for partial object updates

http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.6



I'm still testing Turbolinks ...

Routing Concerns

Concerns split your models into separate modules

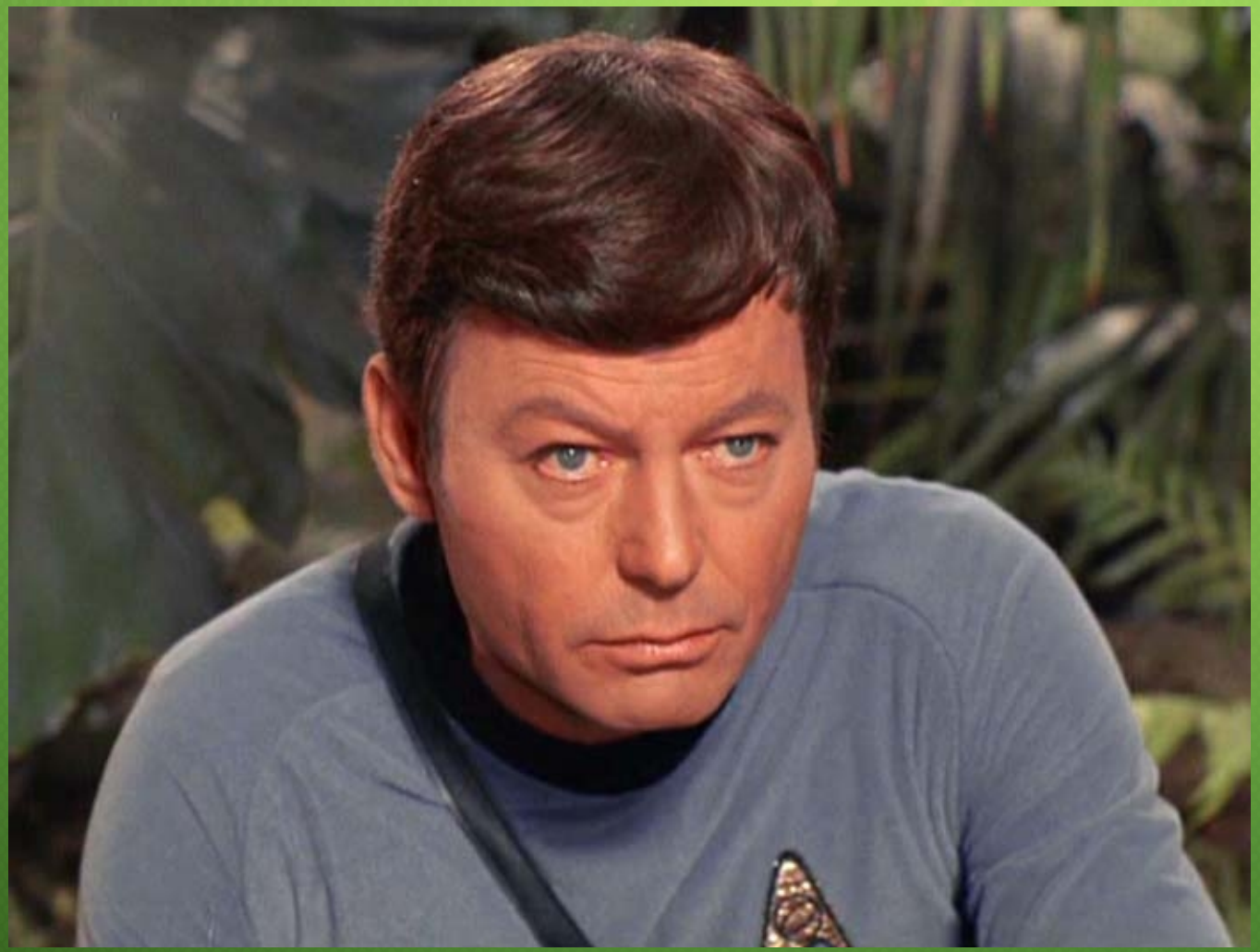
ActiveSupport::Concern

(Existing best practice for Rails)

Routing concerns abstract common routing resource concerns

https://github.com/rails/routing_concerns

```ruby
# Rails 3:
Foo::Application.routes.draw do
  resources :messages  { resources :comments }
  resources :uploads   { resources :comments }
  resources :documents { resources :comments }
end


# Rails 4:
Foo::Application.routes.draw do
  concern :commentable do
    resources :comments
  end

  resources :messages, :uploads, :documents, concerns: :commentable
end
```



DHH loves this stuff, Jim.

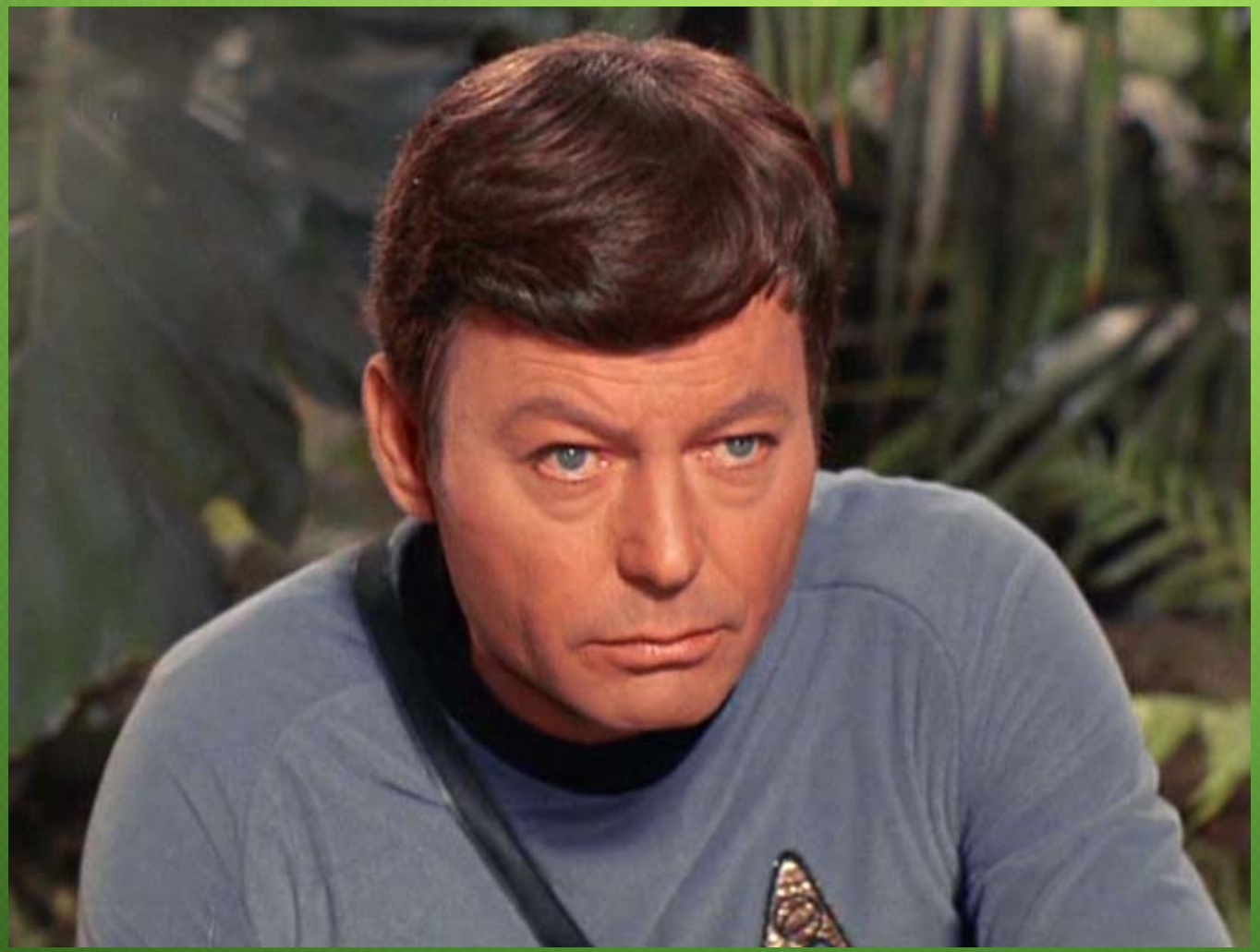
ActiveRecord::
Relation

```
# Rails 3:
> Post.all.class
Array


# Rails 4:
> Post.all.class
ActiveRecord::Relation

> Post.all.to_a.class
Array

> Post.scoped
DEPRECATION WARNING: Use Post.all instead

> Post.none
#<ActiveRecord::Relation []>

> Post.first
SELECT "posts".* FROM "posts" ORDER BY "posts"."id" ASC LIMIT 1

> Post.includes(:comments).where("comments.created_at > ?", 2.days.ago)
DEPRECATION WARNING

> Post.includes(:comments).where("comments.created_at > ?",
2.days.ago).references(:comments)
...
```

Old and deprecated

- `vendor/plugins`
- `scoped` (use `all` instead)
- Rails 2 finder syntax (`:first` …)
- dynamic finders (`find_by_`…)
- eager evaluated scopes
- ActiveResource
- …

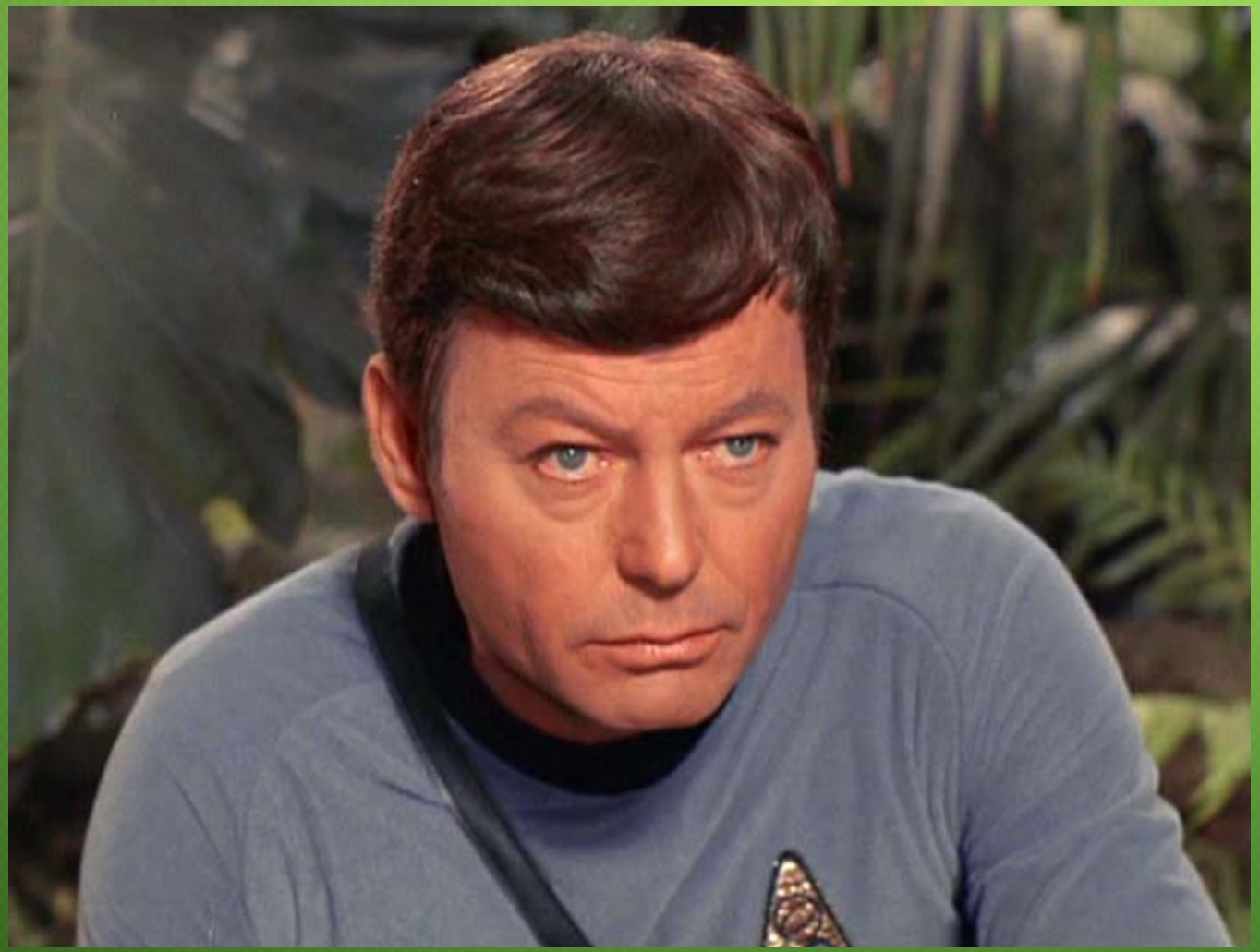
http://edgeguides.rubyonrails.org/4_0_release_notes.html



They are all dead, Jim.

- Ruby >= 1.9.3
- Strong Parameters
- Queue API
- Russian Doll Caching
- Turbolinks
- Live Streaming
- HTTP PATCH
- Routing Concerns
- ActiveRecord::Relation
- Old and deprecated

http://blog.wyeworks.com/2012/11/13/rails-4-compilation-links/



Rails 3 is dead, Jim.

But Rails 4 looks like a hot one.

Dominik Bamberger
SUSE Linux

bamboo@suse.com

twitter.com/bamb00zzle
github.com/b4mboo