Phoenix Framework

What is Elixir, and what is it to Phoenix?

- Functional
- Dynamically typed
- Extreme pattern matching potential
- Makes it easy to have fault tolerence and error recovery

When live meets default

- Phoenix's core is a PubSub model
- Makes it easy to implement channels to pass messages
- Phoenix LiveView takes the PubSub model to the extreme
    - Most UI changes are websocket messages
    - State is on server
    - Can serve millions of users on a single server instance

Lets go

index_live.ex

```elixir
defmodule SummerUniWeb.Live.IndexLive do
  use SummerUniWeb, :live_view
  alias SummerUni.Posts
  alias SummerUni.Posts.Post

  @impl true
  def mount(_params, _session, socket) do

    {:ok, socket
      |> assign_new(:posts, fn _ ->
        SummerUni.Posts.list_post()
      end)}
  end
end
```

index_live.html.heex

```
<%= if @live_action in [:new, :edit] do %>
  <.modal return_to={Routes.index_path(@socket, :index)}>
    <.live_component
      module={SummerUniWeb.PostLive.FormComponent}
      id={@post.id || :new}
      title={@page_title}
      action={@live_action}
      post={@post}
      return_to={Routes.index_path(@socket, :index)}
      user={@user}
    />
  </.modal>
<% end %>

<%= link("New twixt",
  to: Routes.index_path(@socket, :new),
  class: "mb-12 bg-blue-600 rounded text-white px-3 py-2 hover:bg-blue-300 self-end"
) %>

<div class="flex flex-col gap-8">
  <%= for post <- @posts do %>
    <.live_component module={SummerUniWeb.PostComponent} post={post} id={post.id} />
  <% end %>
</div>
```

Making things live

- Subscribe to a topic
- Broadcast evens on said topic
- recieve events in liveviews

lib/summeruni/posts.ex

```elixir
@topic "posts"

def subscribe do
    Phoenix.PubSub.subscribe(PubSub, @topic)
end

def broadcast(data, event) do
    Phoenix.PubSub.broadcast!(PubSub, @topic, {event, data})
    data
end
```