0) show the pdf on the screen and have them walk me through what happens.

1) refresher on routes, controllers, actions. Note this is result of be rake routes.
talk about diff btw new/create and edit/update -- turn to partner to discuss (notice one is a GET)

2) refresher on params --prewrite this on the board as well

what params does Rails get from your request?
Quiz 1: GET www.website.com/dogs/8?favorite=pizza
params: { id: 8, favorite: pizza}

Quiz 2: POST with this form:

```
<form action="/dogs" method="post">
  <input type="text" name="dog[name]" value="charlie">
  <input type="text" name="dog[age]" value="young">
</form>
```

```
params: { dog: {
            name: 'charlie',
            age: 'young'
          }
        }
```

recap: where do params come from?
-path
-query string
-body of request if there is one

3) show the controller. quick talk about **redirect**
-redirect is a NEW request, and is always a GET.
-why redirect? so URL will be accurate (if you just render :show from :index, url is wrong)
-example: put a redirect_to "http://www.google.com" in index and try to get there.
-network tab. status code 302 is a redirect. response header location is where we will end up.
-NEW request. tells your client to make a NEW request.
-cannot have both render and redirect and see the error raised

4) intro to views.
  -yesterday we returned json in response from controller. today, html! through views
  -view = template.
  -handles the presentation logic of the info we give it.
  -we give it info thru instance variables from the controller, which gets interpolated thru ERB

5) ERB
  -your HTML response gets interpreted by browser;
  -erb is interpreted BEFORE THAT; on the server side, before it is put in html
  -lets see it in action.

6)
  INDEX.HTML.ERB
  -change index to render :index instead of json. will look for index.html.erb in users folder in view.
  -make index.html.erb in our views / users folder.
  -use keyboard shortcuts. dash + tab for non print, equals + tab for print. show:

```
<h1><%= 1 + 2 + 3 %></h1>
<% ['ella', 'luna', 'dumbo'].each do |dog| %>
  <p><%= dog %></p>
<% end %>
```

  -rightclick, view page source.
  -it's just plain html! browser doesn't know we did math / iteration.
  -erb was interpreted on the server side BEFORE the html was sent back.
  -now, write the actual view.

```
<h1> all the users! </h1>
<ul>
  <% @users.each do |user| %>
    <li>
      <%= user.name %>
      from
      <%= user.country.name %>
    </li>
  <% end %>
</ul>
```

-add in a link to the user show page around each name:

```
<a href="<%= user_url(user) %>">
  <%= user.name %>
</a>
```

TAKE A BREAK

NEW.HTML.ERB
-- pop quiz on new v. create
-- show route for new -- should be users/new. run on localhost.
  -make action, make a dummy @user = User.new for now (say we will come back to it)
  -make new.html.erb

```
<form action="<%= users_url %>" method="POST">

  <label for="user_name"> Name</label>
  <input type="text" name="user[name]" value="<%= @user.name %>" id="user_name">

  <label>
    Country ID
    <input type="text" name="user[country_id]" value="<%= @user.country_id %>">
  </label>

  <input type="submit" value="create user!">
</form>
```

-- talk about nesting params. this is so name + country_id will be nested under user.
-- make a few? bramble, sennacy.
-- fail a name validation on purpose to get bounced out; show server logs
-- change controller action to render :new again

-- add at top:

```
  <% @user.errors.full_messages.each do |msg| %>
    <%= msg %>
  <% end %>
```

EDIT.HTML.ERB
-- copy and paste everything from new
-- this doesn't feel very DRY...
-- render a partial instead!

--first let's make the partial.

_form.html.erb
--copy and paste everything over from 'new' but change @user to user.
-- why local variables? if we typo, it'll yell at us.
(whereas if we typo ivar, or forget to set it in controller, @ivar will set it to nil)

-- other things that will be wrong:
-- action (create should be users_url, update should be user_url(user))
-- button text (should be 'create!' or 'update!')

--add at top of _form:

```
<% if user.persisted? %>   (persisted checks if this record exists in db)
  <% url = user_url(user.id) %>   (ask class what the update url is)
  <% button_text = "Update user!" %>
<% else %>
  <% url = users_url %>       (ask class what the new url is)
  <% button_text = "Create user!" %>
<% end %>
```

-- now, in form, swap out your new local variabls url and button text
in the action and the submit.

```
<form action="<%= url %>" method="POST">
<input type="submit" value="<%= button_text %>">
```

-- AND, remember that for update requests we have to do hidden method. so, first line of partial form:

```
<% if user.persisted? %>
  <input type="hidden" name="_method" value="PUT">
<% end %>
```

-- render the partial:

edit.html.erb:

```
<h1>Edit your user!</h1>
  <%= render "form", user: @user %>
```

new.html.erb:

```
make a new user!
<%= render "form", user: @user %>
```

last thing if still alive: add delete button to each user in index.
-add in a button to delete users on each
-point out: forms are default post or get. hidden is how we trick it.

```
<form action="<%= user_url(user) %>" method="POST">
  <input type="hidden" name="_method" value="DELETE">
  <input type="submit"  value="Destroy this user">
</form>
```

-- update controller to redirect to users_url upon destroy.
-- pop quiz on what redirect does.

8) link_to and button_to : DON'T. you get these next week.

9) daily quiz!