

# This slide is not here

**Let's talk about  
web apps**

**the new web  
is simple**

**The client makes a request**  
**The server returns HTML**  
**The client displays HTML**

**Except when its not  
simple...**

**The client makes a request  
The server returns data  
The client generates HTML**

**Client side apps give  
you power at the cost  
of complexity**

# Until...

**Hotwire: HTML over The Wire**  
**Noel Rappin (@noelrap)**  
**<https://www.joinroot.com>**  
**<http://noelrappin.com>**  
**#hotwire\_html\_over\_the\_wire**

# We are going to

- Discuss Hotwire's structure
- Use a bunch of Turbo features on a site
- Augment them with a little Stimulus
- See where we are on time

# Follow along...

**[https://github.com/noelrappin/  
rails\\_conf\\_2021\\_hotwire](https://github.com/noelrappin/rails_conf_2021_hotwire)**

**These slides are in the  
repo as PDF**

# And as Markdown

**The markdown has all  
the code changes**

**Each step has a  
different branch**

# Here's our sample app

NorthBy   Welcome, Awesome Reader!   Log out

Apr 21   Apr 22   Apr 23   Apr 24   Apr 25   Apr 26   Show All

Search concerts

**Favorite Concerts** ▾

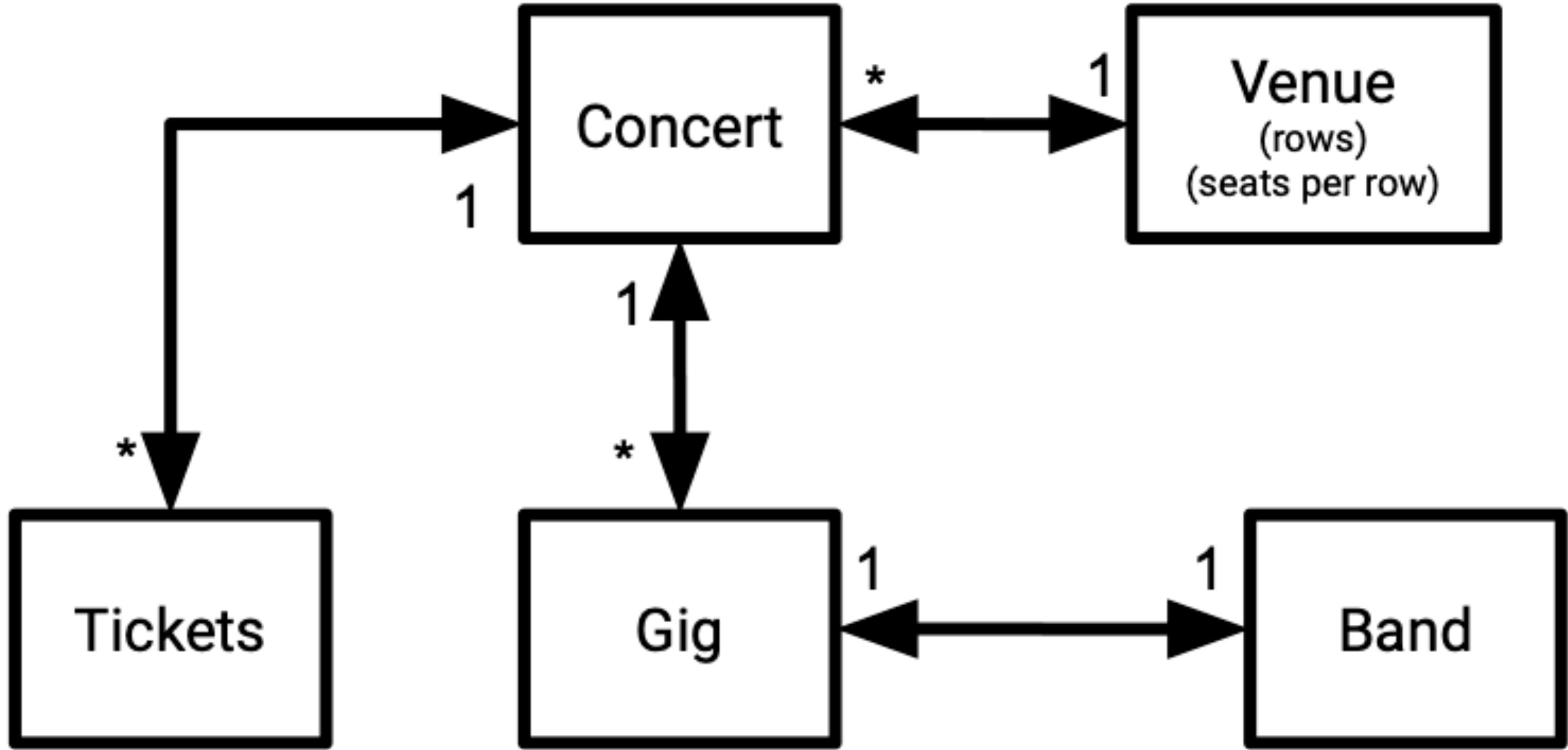
**Wednesday, April 21, 2021**

5:00 PM	<b>A Farewell to Arms</b> Deep Purple Arena Rock, Soft Rock, Hard Rock, and Alternative Rock <a href="#">Southern Massachusetts Institute</a>	92 Tickets Remaining	<a href="#">Make Favorite</a>	<a href="#">Edit</a>
9:00 PM	<b>Fear and Trembling</b> Dixie Chicks, King Crimson, King Crimson British Invasion, Hard Rock, Pop/Rock, Bubblegum Pop, Alternative Rock, and Singer/Songwriter <a href="#">Cormier College</a>	295 Tickets Remaining	<a href="#">Make Favorite</a>	<a href="#">Edit</a>

**Thursday, April 22, 2021**

1:00 PM	<b>Some Buried Caesar</b> Deep Purple, The Indigo Girls Arena Rock, Soft Rock, Hard Rock, Alternative Rock, and Metal <a href="#">Feil Academy</a>	117 Tickets Remaining	<a href="#">Make Favorite</a>	<a href="#">Edit</a>
3:00 PM	<b>Edna O'Brien</b> The Beastie Boys New Wave <a href="#">Southern Nevada Institute</a>	301 Tickets Remaining	<a href="#">Make Favorite</a>	<a href="#">Edit</a>
5:00 PM	<b>The Road Less Traveled</b> <a href="#">The Road Less Traveled</a>	211 Tickets Remaining		

**The login is  
email: areader@example.com  
password: awesome**



# **It's not a complete app**

**We're going to  
augment this page  
with Hotwire**

**Hotwire is:  
Turbo and  
Stimulus**

# Server side rendering

- Regular HTML
- Navigation via turbo frames
- Form submit and turbo stream
- Client-side CSS switch
- Generic Stimulus controller
- Specific Stimulus controller

# Turbo Drive

# **Rebranding of Turbolinks**

**Automatically  
handles navigation in  
the background**

**Updates DOM without  
reloading page**

**Makes a page  
transition appear  
faster**

**But it applies to an  
entire page**

**What if we want to  
only refresh part of a  
page?**

# Let's add inline editing

# Turbo Frames

**We're going to make  
this change first, then  
explain it**

# **Branch workshop\_01**

/app/views/concerts/\_concert.html.erb

/app/views/concerts/\_form.html.erb

```
<%= turbo_frame_tag(dom_id(concert)) do %>
  THE WHOLE FILE
<% end %>
```

## app/controllers/concert\_controller.rb

```
def show
  if params[:inline]
    render(@concert)
  end
end
```

# app/controllers/concert\_controller.rb

```
def create
  @concert = Concert.new(concert_params)
  respond_to do |format|
    if @concert.save
      format.html { redirect_to @concert, notice: "" }
      format.json { render :show, status: :created, location: @concert }
    else
      format.html { render :new, status: :unprocessable_entity }
      format.json { render json: @concert.errors, status: :unprocessable_entity }
    end
  end
end
```

**And that works.  
Really.**

# How?

# Custom HTML Tag

```
<turbo-frame id="concert_16">  
</turbo-frame>
```

# **Inside a turbo frame:**

**All Navigation stays  
within the frame**

**Unless otherwise  
specified**

**The returned HTML is  
parsed and looks for a  
Turbo Frame  
matching the DOM ID**

**If there is no  
matching frame, the  
frame goes blank**

**This means we can reuse the edit  
page inline, even though the edit  
page has more stuff**

**It also means we could have sent  
the whole page in response to the  
form submit, and it would have  
worked, but had worse  
performance**

# Two fun tricks

```
<turbo-frame id="concert_16" src="/concerts/16" loading="lazy">
  LOADING DISPLAY
</turbo-frame>
```

**src means "load from  
that URL after the  
page loads"**

**lazy means "wait to  
load until the element  
is visible"**

**Possible to cache the  
page structure and  
have it filled in**

**One problem: Our  
links broke**

# **Branch workshop\_02**

# Making the link work

app/views/concerts/\_concert.html.erb

```
<%= link_to(concert.name, concert, "data-turbo-frame": "_top") %>
```

# Making the favorite button work

app/views/favorites/\_list.html.erb

```
<%= turbo_frame_tag("favorite-concerts") do %>
  REST OF FILE
<% end %>
```

## app/views/concerts/\_concert.html.erb

```
<% if current_user.favorite(concert) %>
<%= button_to(
  "Remove Favorite",
  favorite_path(id: current_user.favorite(concert)),
  method: "delete",
  form: {data: {"turbo-frame": "favorite-concerts"}},
  class: SimpleForm.button_class)%>
<% else %>
<%= button_to("Make Favorite",
  favorites_path(concert_id: concert.id),
  method: "post",
  form: {data: {"turbo-frame": "favorite-concerts"}},
  class: SimpleForm.button_class) %>
<% end %>
```

# app/controllers/favorites\_controller

```
class FavoritesController < ApplicationController
  def index
  end

  def create
    Favorite.create(user: current_user, concert_id: params[:concert_id])
    render(partial: "favorites/list", locals: {user: current_user})
  end

  def destroy
    @favorite = Favorite.find(params[:id])
    @favorite.destroy
    render(partial: "favorites/list", locals: {user: current_user})
  end

  private
  def favorite_params
    params.require(:concert_id)
  end
end
```

**We don't have to  
change\_favorite...**

# This also works

**Almost -- the body of  
the page doesn't  
change**

# This is a job for...

# Turbo Streams

**Turbo Streams allows  
to change multiple  
page sections in a few  
different ways in one  
payload**

```
<turbo-stream action="ACTION" target="TARGET">
  <template>
    OUR HTML GOES HERE
  </template>
</turbo-stream>
```

**Turbo Streams is  
different from Turbo  
Frames**

**Multiple Turbo Stream payloads  
can be in the same response**

**Turbo Frames can only affect one DOM ID**

**Turbo Streams allows for different  
HTML actions (append, prepend,  
remove, replace, update)**

**Turbo Frames can only update**

**Turbo Streams match any DOM  
element ID**

**Turbo Frames only matches other Turbo Frames**

**Turbo Streams happen  
automatically on form submit**

**Turbo Frame happens on all navigation (except forms  
that return 200)**

**Turbo Stream processing does NOT  
happen on normal GET requests**

**Turbo Streams are  
meant to be  
enhancements**

**Ideally, they allow  
you to reuse existing  
templates**

**A Turbo Stream  
submit is a Rails  
format**

# **Branch workshop\_03**

# app/controllers/favorites\_controller.rb

```
class FavoritesController < ApplicationController
  def index
    end

  def create
    @favorite = Favorite.create(
      user: current_user,
      concert_id: params[:concert_id]
    )
  end

  def destroy
    @favorite = Favorite.find(params[:id])
    @favorite.destroy
  end

  private
  def favorite_params
    params.require(:concert_id)
  end
end
```

## **app/views/favorites/create.turbo\_stream.erb**

```
<%= turbo_stream.append("favorite-concerts-list", @favorite) %>
<%= turbo_stream.replace(dom_id(@favorite.concert)) do %>
  <%= render(@favorite.concert, user: @favorite.user) %>
<% end %>
```

## **app/views/favorites/destroy.turbo\_stream.erb**

```
<%= turbo_stream.remove(dom_id(@favorite)) %>
<%= turbo_stream.replace(dom_id(@favorite.concert)) do %>
  <%= render(@favorite.concert, user: @favorite.user) %>
<% end %>
```

**Okay, that gets us  
where we started...**

# Let's add animation

**You can do this  
without Turbo, but it's  
harder**

# **Branch workshop\_04**

**This uses the  
animate.css library  
for animation effects**

## app/views/favorites/create.turbo\_stream.erb

```
<%= turbo_stream.append("favorite-concerts-list") do %>
  <%= render(@favorite, animate_in: true) %>
<% end %>
<%= turbo_stream.replace(dom_id(@favorite.concert)) do %>
  <%= render(@favorite.concert, user: @favorite.user) %>
<% end %>
```

## app/views/favorites/\_favorite.html.erb

```
<%= concert = favorite.concert %>
<%= animate_in ||= false %>
<article class="my-6 animate__animated
              <%= animate_in ? "animate__slideInRight" : "" %>"
              id=<%= dom_id(favorite) %>""
              data-animate-out="animate__slideOutRight">
# AND SO ON
</article>
```

**Animation out is  
trickier, because  
Turbo removes the  
item, we need to  
capture that**

# Event hooks

## app/packs/entrypoints/application.js

```
document.addEventListener("turbo:before-stream-render", (event) => {
  if (event.target.action === "remove") {
    const targetFrame = document.getElementById(event.target.target)
    if (targetFrame.dataset.animateOut) {
      event.preventDefault()
      const elementBeingAnimated = targetFrame
      elementBeingAnimated.classList.add(targetFrame.dataset.animateOut)
      elementBeingAnimated.addEventListener("animationend", () => {
        targetFrame.remove()
      })
    }
  }
})
```

**Turbo Streams are  
especially good over  
ActionCable**

# **Branch workshop\_05**

# Uses Redis

# If you have docker

```
docker run --rm -it -p 6379:6379  
redis:latest
```

# Otherwise

## config/cable.yml

```
development:  
  adapter: async
```

**Note: all the partials  
have been scrubbed to  
not use current\_user**

## app/views/schedules/show.html.erb

```
<%= turbo_stream_from(@user, :favorites) %>
```

# app/models/favorite.rb

```
after_create_commit -> do
  Turbo::StreamsChannel.broadcast_stream_to(
    user, :favorites,
    content: ApplicationController.render(
      :turbo_stream,
      partial: "favorites/create",
      locals: {favorite: self}
    )
  )
end

after_destroy_commit -> do
  Turbo::StreamsChannel.broadcast_stream_to(
    user, :favorites,
    content: ApplicationController.render(
      :turbo_stream,
      partial: "favorites/destroy",
      locals: {favorite: self}
    )
  )
end
```

**There are a lot of  
simpler shortcuts here**

## **app/views/favorites/\_create.turbo\_stream.erb**

```
<%= turbo_stream.append("favorite-concerts-list") do %>
  <%= render(favorite, animate_in: true) %>
<% end %>
<%= turbo_stream.replace(dom_id(favorite.concert)) do %>
  <%= render(favorite.concert, user: favorite.user) %>
<% end %>
```

## **app/views/favorites/\_delete.turbo\_stream.erb**

```
<%= turbo_stream.remove(dom_id(favorite)) %>
<%= turbo_stream.replace(dom_id(favorite.concert)) do %>
  <%= render(favorite.concert, user: favorite.user) %>
<% end %>
```

## app/views/controllers/favorites\_controller.rb

```
def create
  @favorite = Favorite.create(
    user: current_user,
    concert_id: params[:concert_id]
  )
  head(:ok)
end

def destroy
  @favorite = Favorite.find(params[:id])
  @favorite.destroy
  head(:ok)
end
```

**You can verify this  
with two browsers  
side-by-side**

**Lets make the  
favorites block show  
hide based on that  
button**

**I'm going to show this,  
and then explain it**

# **Branch workshop\_06**

## app/views/favorites/list.html.erb

```
<span class="<%= SimpleForm.button_class %>" blue-hover ml-6"
  data-controller="flip"
  data-flip-status-value="true"
  data-flip-true-class="rotate-90"
  data-flip-false-class="-rotate-90"
  data-action="click->flip#toggle click->fade#fade">
<%= image_pack_tag(
    "chevron-right.svg",
    width: 25,
    height: 25,
    class: "inline transform transition-transform duration-1000",
    "data-flip-target": "elementToChange" ) %>
</span>
```

**The key bits here are  
the data attributes**

**A controller is where  
Stimulus code is  
executed**

**Controllers are  
attached to DOM  
elements with the  
data-controller  
attribute**

**Which matches a file,  
in this case  
`flip_controller.js`**

# app/packs/controllers/flip\_controller.js

```
import { Controller } from "stimulus"

export default class FlipController extends Controller {
  static classes = ["true", "false"]
  static targets = ["elementToChange"]
  static values = { status: Boolean }

  toggle() {
    this.statusValue = !this.statusValue
  }

  statusValueChanged() {
    this.elementToChangeTarget.classList.toggle(
      this.trueClass,
      this.statusValue
    )
    this.elementToChangeTarget.classList.toggle(
      this.falseClass,
      !this.statusValue
    )
  }
}
```

**Targets are declared as data-  
controllername-target=targetname  
as in data-flip-  
target=elementToChange**

# Targets are declared in the controller

- elementToChangeTarget
- elementToChangeTargets
- hasElementToChangeTarget

# Values and classes

**data-flip-status-value=true**  
**data-flip-true-class=rotate-90**  
**data-flip-false-class=-rotate-90**

- `this.statusValue`
- `this.statusValueChanged()` (called on startup)
- `this.trueClass`
- `this.hasTrueClass`

# Actions

`data-action="click->flip#toggle"`  
`action -> controller # method`

**So the action calls the toggle  
method  
which changes the value  
which calls the change hook**

**Change hooks get  
called whenever the  
property or  
underlying DOM  
attribute are changed.**

## app/views/favorites/list.html.erb

```
<section class="my-4"
  id="favorite-section"
  data-controller="fade"
  data-fade-status-value="true"
  data-fade-clean-value="true"
  data-fade-fade-in-class="fadeInDown"
  data-fade-fade-out-class="fadeOutUp">

<div id="favorite-concerts-list"
  data-fade-target="elementToChange"
  data-action="animationend->fade#animationend
  animationstart->fade#animationstart">
```

## app/packs/controllers/fade\_controller.js

```
import { Controller } from "stimulus"

export default class FadeController extends Controller {
  static classes = ["fadeIn", "fadeOut"]
  static targets = ["elementToChange"]
  static values = { status: Boolean, clean: Boolean }

  fade() {
    this.statusValue = !this.statusValue
    this.cleanValue = false
  }

  animationend() {
    this.toggleClass("max-h-0", !this.statusValue)
  }

  animationstart() {
    this.toggleClass("max-h-0", false)
  }
}
```

## app/packs/controllers/fade\_controller.js

```
statusValueChanged() {
  if (this.cleanValue) {
    return
  }
  this.toggleClass("animate__animated", true)
  this.toggleClass(`animate__${this.fadeInClass}`, this.statusValue)
  this.toggleClass(`animate__${this.fadeOutClass}`, !this.statusValue)
}

toggleClass(cssClass, state) {
  this.elementToChangeTarget.classList.toggle(cssClass, state)
}
}
```

**Let's sort our  
favorites**

# **Branch workshop\_07**

# app/views/favorites/list.html.erb

```
<div id="favorite-concerts-list"
  data-controller="sort"
  data-fade-target="elementToChange"
  data-action="animationend->fade#animationend animationstart->fade#animationstart">
```

## app/views/favorites/\_favorite.html.erb

```
<article class="my-6 animate__animated
          <%= animate_in ? "animate__slideInRight" : "" %>"  
  id="<%= dom_id(favorite) %>"  
  data-animate-out="animate__slideOutRight"  
  data-sort-target="sortElement"  
  data-sort-value="<%= favorite.sort_date %>">
```

# app/packs/controllers/sort\_controller.js

```
import { Controller } from "stimulus"

export default class SortController extends Controller {
  static targets = ["sortElement"]

  initialize() {
    const target = this.element
    const observer = new MutationObserver((mutations) => {
      observer.disconnect()
      Promise.resolve().then(start)
      this.sortTargets()
    })
    function start() {
      observer.observe(target, { childList: true, subtree: true })
    }
    start()
  }
}
```

```
sortTargets() {
  if (this.targetsAlreadySorted()) {
    return
  }
  this.sortElementTargets
    .sort((a, b) => {
      return this.sortValue(a) - this.sortValue(b)
    })
    .forEach((element) => this.element.append(element))
}

targetsAlreadySorted() {
  let [first, ...rest] = this.sortElementTargets
  for (const next of rest) {
    if (this.sortValue(first) > this.sortValue(next)) {
      return false
    }
    first = next
  }
  return true
}

sortValue(element) {
  if ("sortValue" in element.dataset) {
    return parseInt(element.dataset.sortValue, 10)
  } else {
    return parseInt(element.children[0].dataset.sortValue, 10)
  }
}
```

**Now lets sort the  
concerts**

**This one is a little  
much..**

## app/controllers/concerts\_controller.rb

```
def update
  respond_to do |format|
    if @concert.update(concert_params)
      format.turbo_stream {}
      format.html { render(@concert, locals: {user: current_user}) }
      format.json { render :show, status: :ok, location: @concert }
    else
      format.html { render :edit, status: :unprocessable_entity }
      format.json { render json: @concert.errors, status: :unprocessable_entity }
    end
  end
end
```

## app/views/concerts/update.turbo\_stream.erb

```
<%= turbo_stream.replace(dom_id(@concert)) do %>
  <%= render(@concert, user: current_user) %>
<% end %>
```

## app/views/concerts/\_concert.erb

```
<%= turbo_frame_tag(  
  dom_id(concert),  
  class: "concert",  
  "data-sort-target": "sortElement",  
  "data-sort-value": concert.start_time.to_i,  
  "data-#{concert.start_time.strftime("%Y-%m-%d")}" : true) do %>
```

## app/views/schedules/show.html.erb

```
<section data-controller="sort">
  <% @concerts.sort_by(&:start_time).each do |concert| %>
    <%= render concert, user: @user %>
  <% end %>
</section>
```

## app/views/schedules/show.html.erb

```
<style>
<% @schedule.schedule_days.each do |schedule_day| %>
  <% today = "data-#{schedule_day.day.by_example("2006-01-02")}" %>
  .concert[<%= today %>]:first-child::before,
  .concert:not( [<%= today %>] ) + [ <%= today %> ]::before
  {
    content: "<%= schedule_day.day.by_example("Monday, January 2, 2006") %>";
    font-size: 1.875rem;
    line-height: 2.25rem;
    font-weight: 700;
  }
<% end %>
</style>
```

**This works... kind of**

# **Fin**

# Modern Front-End Development for Rails

Hotwire, Stimulus, Turbo, and React

# Noel Rappin (@noelrap)

**<http://pragprog.com/book/nrclient>**  
**<http://pragprog.com/book/tailwind>**  
**(50% with ModernWeb2021)**  
**<https://buttondown.email/noelrap>**

# Modern CSS with Tailwind

Flexible Styling Without the Fuss