StimulusJS

(with Rails Nested Attributes)

What is StimulusJS?

Stimulus is a JavaScript framework with modest ambitions. It doesn’t seek to take over your entire front-end—in fact, it’s not concerned with rendering HTML at all. Instead, it’s designed to augment your HTML with just enough behavior to make it shine.
— stimulusjs.org

Made by Basecamp (of Rails fame)

Goals of Stimulus

- Not a full front-end framework
- Add interactivity to existing code
- Easy to add
- Reusable/separation of concerns

Three Core Parts

- Controllers
  - Written in javascript
- Actions
  - **`data-action`** attribute with a value like **`controller-name#methodName`**
- Targets
  - **`data-target`** with values like **`controller-name.targetName`**

Let's start with our HTML

```html
<div>
  <input type="text">
  <button>Greet</button>
</div>
```

Add a Controller

```
// src/controllers/hello_controller.js
import { Controller } from "stimulus"

export default class extends Controller {
}
```

Hook it up

```
<div data-controller="hello">
  <input type="text">
  <button>Greet</button>
</div>
```

Can You Hear Me Now?

```
// src/controllers/hello_controller.js
import { Controller } from "stimulus"

export default class extends Controller {
  connect() {
    console.log("Hello, Stimulus!", this.element)
  }
}
```

In Console

```
Hello Stimulus, <div....
```

Let's Say Hi

```
// src/controllers/hello_controller.js
import { Controller } from "stimulus"

export default class extends Controller {
  greet() {
    console.log("Hello, Stimulus!", this.element)
  }
}
```

Hook Up Action

```
<div data-controller="hello">
  <input type="text">
  <button data-action="click->hello#greet">Greet</button>
</div>
```

Default Actions

Element	Default Event
a	click
button	click
form	submit
input	change
input type=submit	click
select	change
textarea	change

Let's Try that Again

Don't need the `click->`

```
<div data-controller="hello">
  <input type="text">
  <button data-action="hello#greet">Greet</button>
</div>
```

Let's Change Something

```html
<div data-controller="hello">
  <input data-target="hello.name" type="text">
  <button data-action="hello#greet">Greet</button>
</div>
```

Targets in the Controller

```javascript
// src/controllers/hello_controller.js
import { Controller } from "stimulus"

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

  greet() {
    const element = this.nameTarget
    const name = element.value
    console.log(`Hello, ${name}!`)
  }
}
```

Sprinkle your HTML with controller, target, and action attributes:

```html
<!--HTML from anywhere-->
<div data-controller="hello">
  <input data-target="hello.name" type="text">

  <button data-action="click->hello#greet">
    Greet
  </button>

  <span data-target="hello.output">
  </span>
</div>
```

Write a compatible controller and watch Stimulus bring it to life:

```javascript
// hello_controller.js
import { Controller } from "stimulus"

export default class extends Controller {
  static targets = [ "name", "output" ]

  greet() {
    this.outputTarget.textContent =
      `Hello, ${this.nameTarget.value}!`
  }
}
```

Nested Attributes

Rails feature where you can update **`has_one`** or **`has_many`** relationships via a parent resources **`create`** or **`update`** methods.

Modeling

```ruby
class Store < ApplicationRecord
  has_many :books

  accepts_nested_attributes_for :books, allow_destroy: true

  # :reject_if -> takes a proc
  # :limit -> how many can you update
  # :update_only -> for has_one relationships
end
```

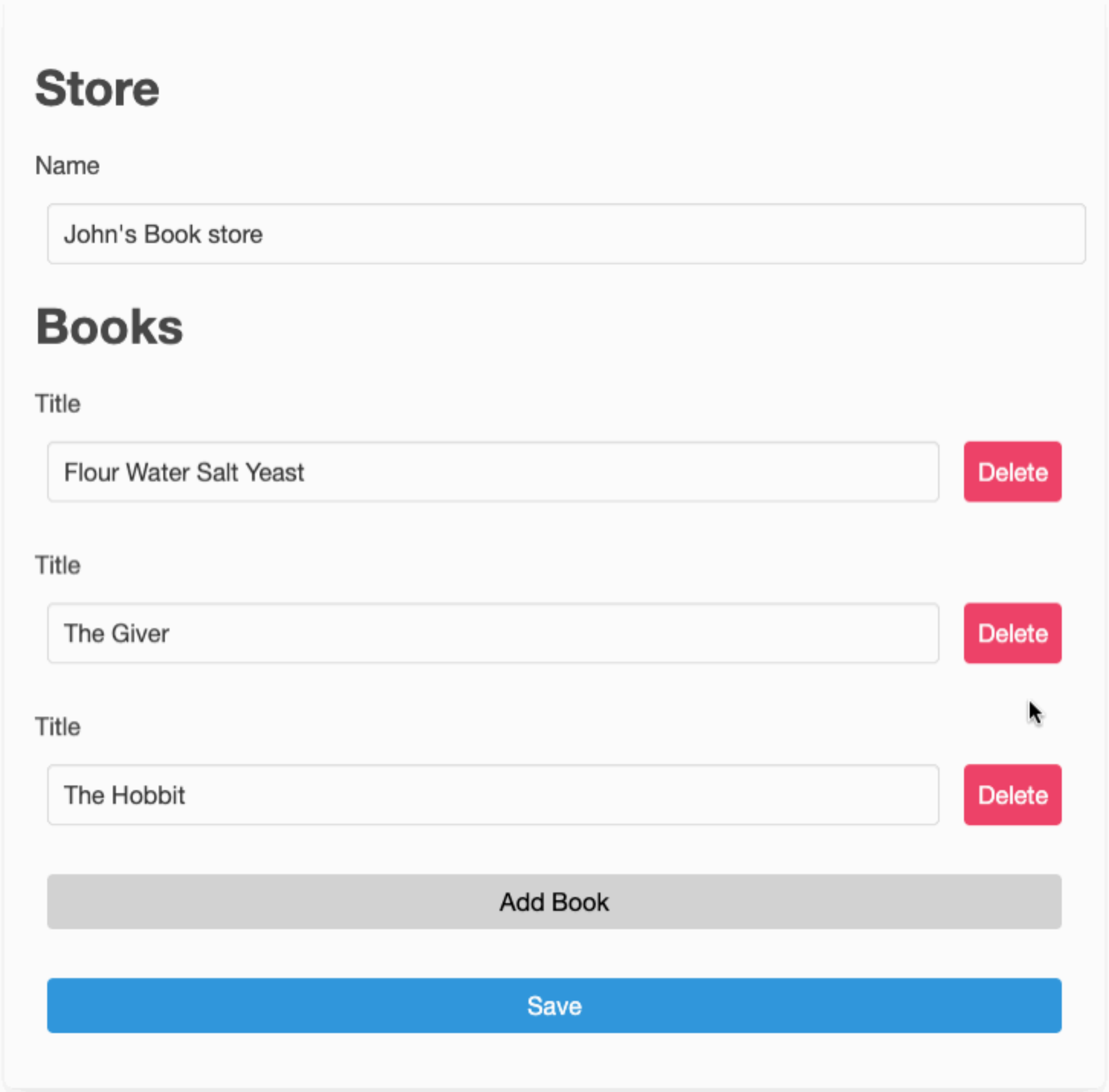
Params

```
{
  store: {
    name: "John's store",
    books_attributes: {
      "0" => { title: "Lord of the Rings" }, # new record
      "1" => { id: 4, title: "Dictionary" }, # update
      "2" => { id: 2, _destroy: 1 }, # destroy
    },
  },
}
```

Dynamic

So, let's say we want to have a page where we can add multiple books and change the store name.

Anything that involves adding multiple books means we need to need to generate inputs. So either we need to hide a large number of inputs and slowly reveal them, or we need to build them when the user requests it.



Store
Name
John's Book store
Books
Title
Flour Water Salt Yeast
Delete
Title
The Giver
Delete
Title
The Hobbit
Delete
Add Book
Save

Our Html

```
<div>
  <%= form_with model: @store do |form|%>
    <h1>Store</h1>
    <%= form.label :name %>
    <%= form.text_field :name %>
    <h1>Books</h1>
    <div>
      <%= form.fields_for :books do |book_form| %>
        <%= render partial: "book_fields", locals: { book_form: book_form } %>
      <% end %>
    </div>
    <div class="fields">
      <%= button_tag "Add Book", class: "new" %>
    </div>
    <div class="fields">
      <%= form.button "Save", type: :submit %>
    </div>
  <% end %>
</div>
```

Book Field

```erb
<div class="fieldset">
  <%= book_form.label :title %>
  <div class="fields">
    <%= book_form.text_field :title %>
    <%= button_tag "Delete", class: "delete" %>
    <%= book_form.hidden_field :_destroy, value: 0 %>
  </div>
</div>
```

Controller

```ruby
@store = Store.includes(:books).find(params[:id])

if @store.update(update_params)
  redirect_to @store
else
  render :edit
end
# ...
def update_params
  params.require(:store).permit(:id, :name, books_attributes: %i[id title _destroy])
end
```

Let's Destroy

```
<%= book_form.hidden_field :_destroy,
    value: 0,
    data: {
      target: "destroy.destroyInput",
    } %>
```

Update our Button

```
<%= button_tag "Delete",
    class: "delete",
    data: {
      action: "destroy#deleteRow",
      index: book_form.index,
    } %>
```

```javascript
// controllers/destroy_controller.js

import { Controller } from "stimulus";

export default class extends Controller {
  static targets = ["destroyInput"];

  deleteRow() {
    event.preventDefault();

    let inputIndex = event.currentTarget.getAttribute("data-index");
    this.destroyInputTargets[inputIndex].value = 1;
  }
}
```

Hide and Seek

```
deleteRow() {
  event.preventDefault();

  event.currentTarget.closest(".fieldset").classList.toggle("hidden");
  let inputIndex = event.currentTarget.getAttribute("data-index");
  this.destroyInputTargets[inputIndex].value = 1;
}
```

Let's Modularize

Now we have a controller that destroys things and hides them.

But what if we want to hide something? Or maybe change a different value other than destroy?

Toggle Controller - HTML

Div

```
<div data-controller="toggle" data-toggle-class="hidden">
```

Button

```
<%= button_tag "Delete",
    class: "delete",
    data: {
      action: "toggle#toggle",
      index: book_form.index,
    } %>
```

Toggle Controller - JS

```
// controllers/toggle_controller.js

import { Controller } from "stimulus";

export default class extends Controller {
  static targets = ["content"];

  toggle(event) {
    event.preventDefault();
    const toggleIndex = event.currentTarget.getAttribute("data-index");
    this.contentTargets[toggleIndex].classList.toggle(this.data.get("class"));
  }
}
```

Value Controller - HTML

Div

```
<div data-controller="toggle value" data-value="1">
```

Button

```
<%= button_tag "Delete",
    class: "delete",
    data: {
      action: "toggle#toggle value#update",
    } %>
```

Value Controller - HTML

Book Field

```
<%= book_form.hidden_field :_destroy,
  value: 0,
  data: {
    target: "value.content",
  } %>
```

Value Controller

```
// controllers/value_controller.js

import { Controller } from "stimulus";

export default class extends Controller {
  static targets = ["content"];

  update(event) {
    event.preventDefault();
    const valueIndex = event.currentTarget.getAttribute("data-index");
    this.contentTargets[valueIndex].value = this.data.get("value");
  }
}
```

Let's Add Some Fields!

We need to produce this:

```
<div class="fieldset">
  <label for="store_books_attributes_1_title">Title</label>
  <div class="fields">
    <input type="text"
           value="Lord of the Rings"
           name="store[books_attributes][1][title]"
           id="store_books_attributes_1_title" />
    <button name="button" type="submit" class="delete">Delete</button>
      <input value="0"
             type="hidden"
             name="store[books_attributes][1][_destroy]"
             id="store_books_attributes_1__destroy" />
  </div>
</div>

<input type="hidden"
       value="25"
       name="store[books_attributes][1][id]"
       id="store_books_attributes_1_id" /></div>
```

How Rails Does Nested Attributes

```
<input ... store[books_attributes][1][title] />
```

How can we most easily do this?

Our Server Renders the Fields

```ruby
module FormFields
  class StoreBooksController < ApplicationController
    def new
      store = Store.new
      store.books.build

      helpers.form_for(store) do |form|
        form.fields_for :books do |book_form|
          render partial: "stores/book_fields", locals: { book_form: book_form }
        end
      end
    end
  end
end
```

If we do that, we get:

```
<input name="store[books_attributes][1][id]" id="store_books_attributes_0_id" />
...
<input name="store[books_attributes][1][id]" id="store_books_attributes_1_id" />
...
<input name="store[books_attributes][1][id]" id="store_books_attributes_0_id" />
...
<input name="store[books_attributes][1][id]" id="store_books_attributes_0_id" />
```

Rails to the Rescue

There is a **`child_index`** parameter to set the starting index.

```
form.fields_for :books,
  child_index: params[:index] do |book_form|
```

So what is our abstraction?

Per the stimulus docs, this is a "content loader". Or maybe for now, an indexed content loader.

IndexedContent - HTML

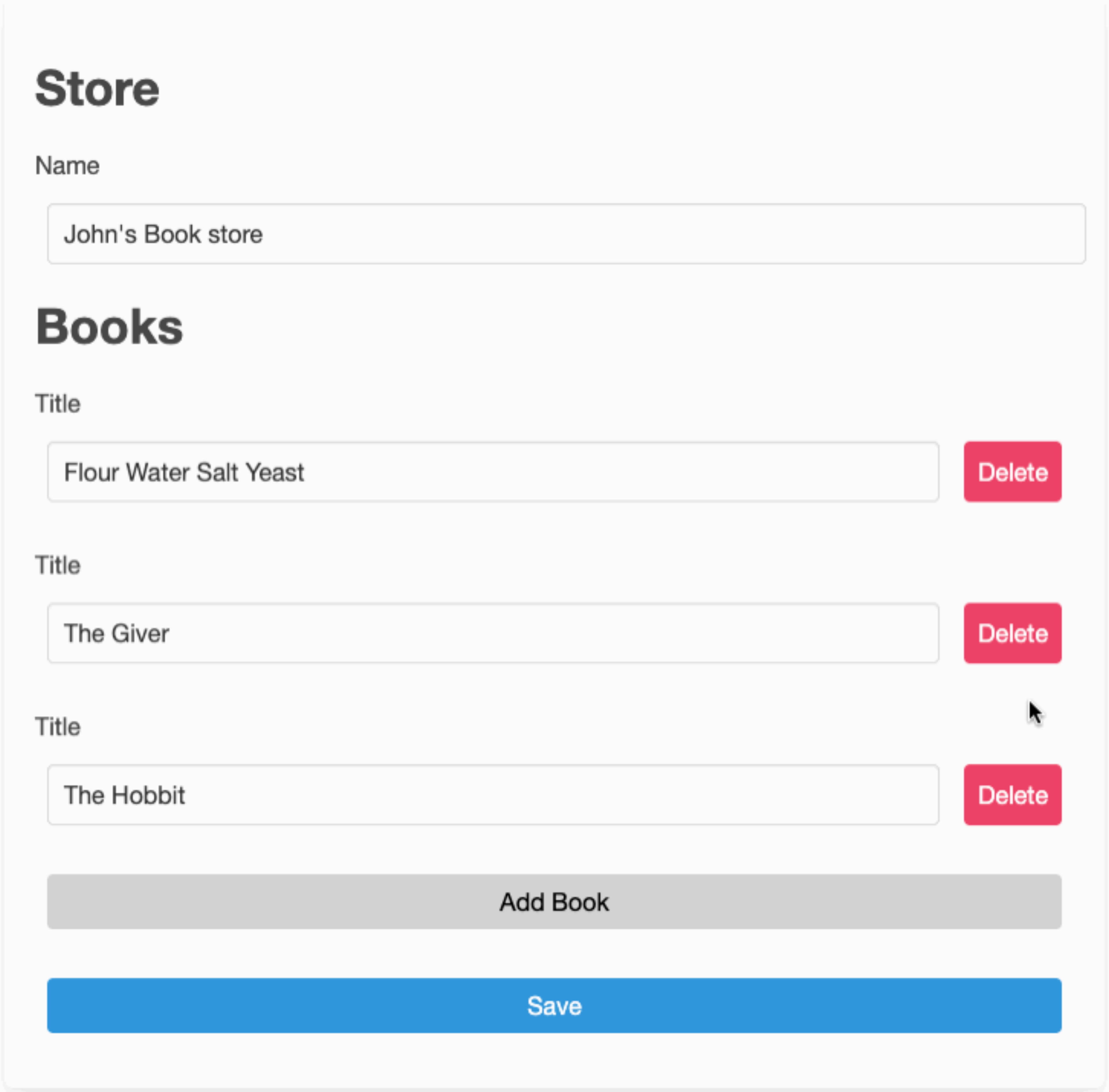
```
<div data-controller="toggle value indexed-content-loader"
     data-indexed-content-loader-next-index=<%= @store.books.length %>
     data-indexed-content-loader-insert-location="beforeend"
     data-indexed-content-loader-url="/form_fields/store_books/new"
     ...>

<%= button_tag "Add Book",
    class: "new",
    data: { action: "indexed-content-loader#insert" } %>
```

IndexedContent - JS

```
// src/controllers/indexed_content_loader_controller.js
import { Controller } from "stimulus";

export default class extends Controller {
  static targets = ["container"];

  connect() {
    this.nextIndex = this.data.get("next-index");
  }
```

```
insert() {
  event.preventDefault();

  let controller = this;
  let queryString = `?index=${this.nextIndex++}`;
}
```

```
  let url = this.data.get("url") + queryString;
  fetch(url)
    .then((response) => response.text())
    .then((html) => {
      controller.containerTarget.insertAdjacentHTML(
        controller.data.get("insert-location"),
        html
      );
    });
}
```



Store

Name

John's Book store

Books

Title

Flour Water Salt Yeast

Title

The Giver

Title

The Hobbit

Links

StimlusJS
Example App
Sneak Peeks Into Hey! Technology

Questions?