SINGLE-PAGE APPLICATION IS NOT THE ONLY WAY

Jiri Werner, PRIMEHAMMER

BACKEND VS. FRONTEND

- frontend (css, effects), backend (everything else)
- standard http request is too slow
- users require more interactivity and snappy response times
- frontend getting more complicated, more responsibility

SINGLE-PAGE APPLICATIONS

- render HTML dynamically on the client
- send AJAX request to the server
- fetch JSON payload
- re-render the DOM
- handle events
- manage state

BENEFITS OF SINGLE-PAGE APPLICATIONS

- complex and fast UI
- near-native experience
- high interactivity

COSTS

- increasingly complex
- dedicated team of front-end developers required
- need to maintain separate codebases
- keeping API up-to-date
- duplication of code (schema, validations)

RAILS & UNOBTRUSIVE JAVASCRIPT

- Rails is traditionally a backend framework
- its approach to handling AJAX operations is using Server-generated JavaScript Responses:
  - form is submitted via a XMLHttpRequest
  - server creates or updates a model object
  - server generates a JavaScript response that includes the updated HTML template for the model
  - client evaluates the JavaScript returned by the server, which then updates the DOM

BENEFITS OF SERVER-GENERATED JAVASCRIPT RESPONSE

- according to **David Heinemeier Hansson (DHH)**:
  - reuse templates without sacrificing performance
  - less computational power needed on the client
  - easy-to-follow execution flow

WHAT CAN GO WRONG WITH SERVER-GENERATED JS RESPONSE?

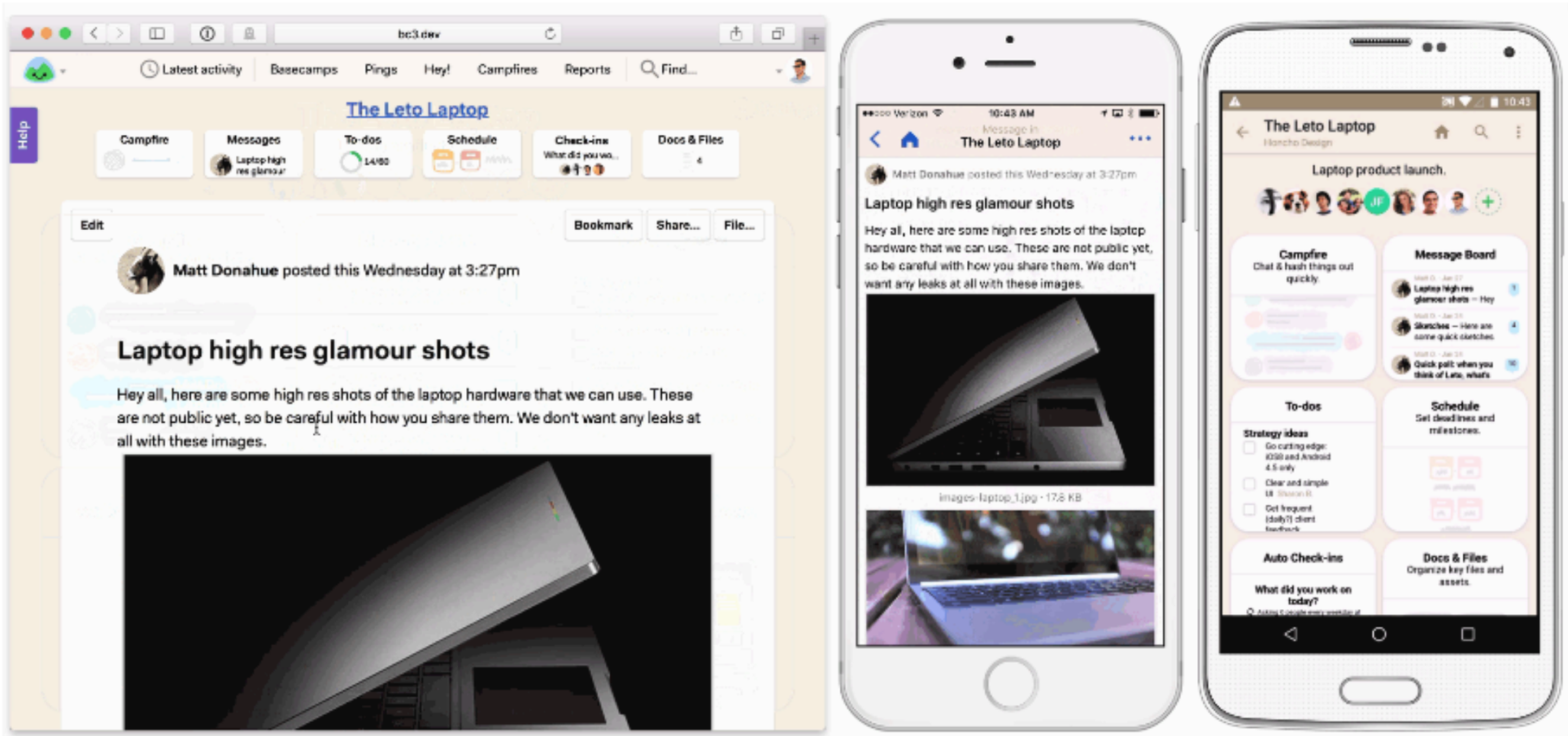
- can lead to unmaintainable code
- mixing JavaScript with ERB is painful
- unrelated snippets of JavaScript code spreading throughout the application
- event listeners need to be manually updated
- events not firing when parts of the DOM are replaced

Gemfile	CI: Run tests in headless Chome	2 months ago
Gemfile.lock	CI: Run tests in headless Chome	2 months ago
LICENSE	Turbolinks 5.1.0	8 months ago
README.md	Add a note about when to build before running tests	2 months ago
config.ru	Run bin/rackup and visit localhost:9292/fixtures to develop locally a...	3 years ago
package.json	Turbolinks 5.2.0	29 days ago

README.md

Turbolinks

Turbolinks® makes navigating your web application faster. Get the performance benefits of a single-page application without the added complexity of a client-side JavaScript framework. Use HTML to render your views on the server side and link to pages as usual. When you follow a link, Turbolinks automatically fetches the page, swaps in its `<body>`, and merges its `<head>`, all without incurring the cost of a full page load.



WITH TURBOLINKS, CLICKING A LINK...

WHAT IS STIMULUS?

- Stimulus is a minimalist JavaScript framework from Basecamp (small teams, pragmatic approach)
- does not attempt to compete with full featured front-end frameworks (React/Angular)
- adds data-attributes to your server generated HTML (controllers, targets, actions)
- designed to observe the rendered HTML and connect elements to JavaScript objects automatically
- state is managed in the DOM

A modest JavaScript framework for the HTML you already have.

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

```
<!--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:

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

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

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

enter a name | Greet

```
$ rails new todolist --webpack

$ yarn add stimulus
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Todolist</title>
    <%= csrf_meta_tags %>

    <%= stylesheet_link_tag    'application', media: 'all', 'data-turbolinks-track': 'reload' %>
    <%= javascript_include_tag 'application', 'data-turbolinks-track': 'reload' %>
    <%= javascript_pack_tag 'application', 'data-turbolinks-track': 'reload' %>
  </head>

```

```

// app/javascript/packs/application.js
import { Application } from "stimulus";
import { definitionsFromContext } from "stimulus/webpack-helpers";

const application = Application.start();
const context = require.context("./controllers", true, /\.js$/);
application.load(definitionsFromContext(context));

```

If your controller file is named...	its identifier will be...
clipboard_controller.js	clipboard
date_picker_controller.js	date-picker
users/list_item_controller.js	users--list-item
local-time-controller.js	local-time

- [x] A New Hope
- [x] The Empire Strikes Back
- [x] Return of the Jedi
- [x] The Phantom Menace
- [x] Attack of the Clones
- [x] Revenge of the Sith
- [x] The Force Awakens
- [ ] The Last Jedi
- [ ] Rogue One

Solo | Add

2 active tasks left.

```ruby
class TasksController < ApplicationController
  def index
    @tasks = Task.all
  end

  def create
    @task = Task.create(task_params)
    render @task
  end

  def toggle
    @task = Task.find(params[:id])
    @task.update completed: !@task.completed
  end

  private

  def task_params
    params.require(:task).permit(:title)
  end
end
```

```ruby
<%# app/views/tasks/index.html.erb %>
<div data-controller="task-list"
     data-task-list-count="<%= Task.active.count %>">
  <div data-target="task-list.tasks" >
       <%= render @tasks %>
     </div>
     <br />
     <%= render "form", task: Task.new %>
     <div data-target="task-list.counter"></div>
</div>
```

```ruby
<%# app/views/tasks/_task.html.erb %>
<div id="task_<%= dom_id(task) %>"
  class="todo-item <%= "completed" if task.completed %>">

  <%= check_box_tag("completed", task.id, task.completed,
        data: { remote: true,
                method: 'put',
                url: url_for(action: 'toggle', id: task.id),
                action: "click->task-list#toggle",
              }) %>
  <%= task.title %>
</div>
```

```ruby
<%# app/views/tasks/_form.html.erb  %>
<%= form_with(model: task, html: { data:
      { type: "html",
        action: "ajax:success->task-list#addTask" }}) do |form| %>

  <%= form.text_field :title, id: :task_title,
        data: { target: "task-list.newTask" }  %>
  <%= form.submit "Add" %>
<% end %>
```

```javascript
import { Controller} from "stimulus";

export default class extends Controller {
  static targets = ["tasks", "newTask", "counter"];

  connect() {
    this.refreshCount();
  }

  addTask(event) {
    const [data, status, xhr] = event.detail;
    this.tasksTarget.insertAdjacentHTML('beforeEnd', xhr.response);
    this.increaseCount();
    this.newTaskTarget.value = "";
    this.newTaskTarget.focus();
  }

  toggle(event) {
    const element = event.target;
    element.parentElement.classList.toggle('completed');
    element.checked ? this.decreaseCount() : this.increaseCount();
  }

  get count() {
    return parseInt(this.data.get("count"));
  }

  set count(value) {
    this.data.set("count", value);
  }

  increaseCount() {
    this.count++;
    this.refreshCount();
  }

  decreaseCount() {
    this.count--;
    this.refreshCount();
  }

  refreshCount() {
    this.counterTarget.textContent = `${this.count} active tasks left
  }
}
```

```
import { Controller} from "stimulus";

export default class extends Controller {
  static targets = ["tasks", "newTask", "counter"];

  connect() {
    this.refreshCount();
  }

  addTask(event) {
    const [data, status, xhr] = event.detail;
    this.tasksTarget.insertAdjacentHTML('beforeEnd', xhr.response);
    this.increaseCount();
    this.newTaskTarget.value = "";
    this.newTaskTarget.focus();
  }

  toggle(event) {
    const element = event.target;
    element.parentElement.classList.toggle('completed');
    element.checked ? this.decreaseCount() : this.increaseCount();
  }

  get count() {
    return parseInt(this.data.get("count"));
  }
```

```
20      element.checked ? this.decreaseCount() : this.increaseCount();
21    }
22
23    get count() {
24      return parseInt(this.data.get("count"));
25    }
26
27    set count(value) {
28      this.data.set("count", value);
29    }
30
31    increaseCount() {
32      this.count++;
33      this.refreshCount();
34    }
35
36    decreaseCount() {
37      this.count--;
38      this.refreshCount();
39    }
40
41    refreshCount() {
42      this.counterTarget.textContent = `${this.count} active tasks left.`
43    }
44  }
```

```
<%# app/views/tasks/_task.html.erb %>
<div id="task_<%= dom_id(task) %>"
  class="todo-item <%= "completed" if task.completed %>">

  <%= check_box_tag("completed", task.id, task.completed,
        data: { remote: true,
                method: 'put',
                url: url_for(action: 'toggle', id: task.id),
                action: "click->task-list#toggle",
              }) %>
  <%= task.title %>
</div>
```

```
<%# app/views/tasks/_form.html.erb  %>
<%= form_with(model: task, html: { data:
      { type: "html",
        action: "ajax:success->task-list#addTask" }}) do |form| %>

  <%= form.text_field :title, id: :task_title,
        data: { target: "task-list.newTask" }  %>
  <%= form.submit "Add" %>
<% end %>
```

SUMMARY

- frontend development is getting increasingly complex
- pros & cons of developing an SPA need to be considered
- Turbolinks offer a reasonably performant alternative
- Stimulus is a minimalist framework working with server-side generated HTML and SJR and designed to help giving some structure to your JavaScript code

THANK YOU

ANY QUESTIONS?