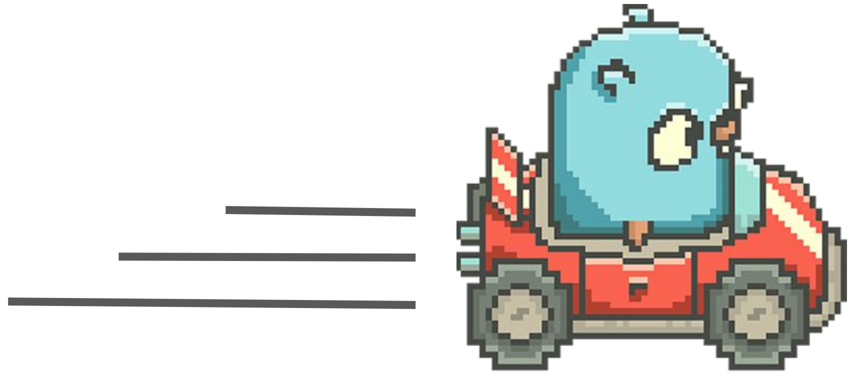


The Fast and the Furrious

Moving faster with Hotwire HTML-over-the-wire

bespinian

Cloud Native Citizens

What is Hotwire?

Hotwire

Two Frameworks

- Turbo - HTML-over-the-wire framework
- Stimulus - JavaScript Enhancement

Part of Basecamps “Majestic Monolith” Philosophy

Turbo

Turbo Drive	- Navigation and Magic
Turbo Frames	- Page Decomposition
Turbo Streams	- Live Page Updates
Turbo Native	- iOS & Android Integration

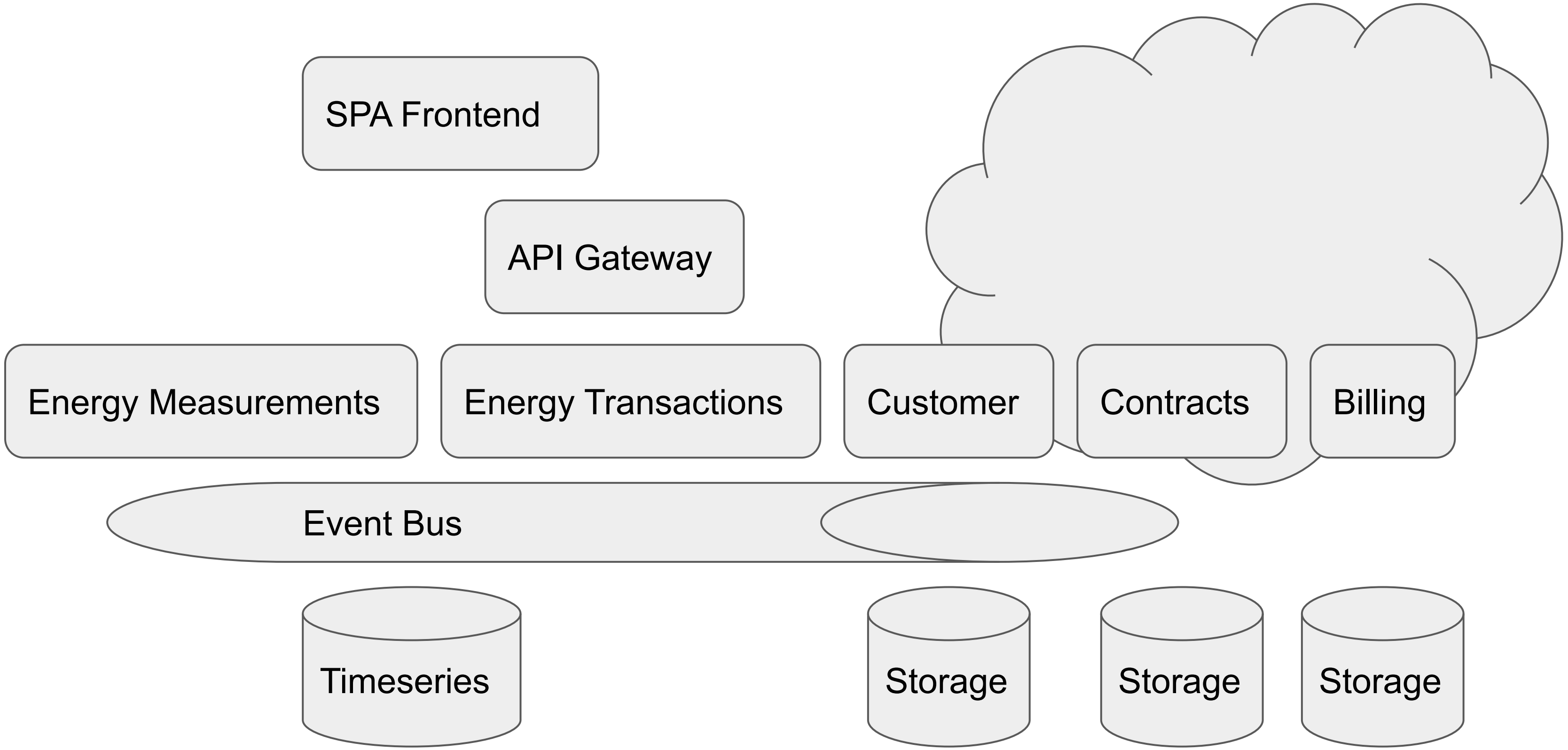
Stimulus

Controller	- Javascript Base Magic
Targets	- Automagic Element Selection

(Intermezzo)

Why are we here?

What We Might Build



Constraints

Small Team

No Frontend Developer

Timeline & Exact Features Unclear

Possible Base for Other Projects

Limited Re-use of Existing Components

Constraints Breed Creativity

Monolith

HTML-over-the-wire

Clear Domain Separation

Hexagonal Architecture

Do the absolute minimum

How do you use Turbo?

Turbo Frames

```
<turbo-frame id=”gopher”>
    <magic>
</turbo-frame>
```

HTML Tag

Requires an `id`

Recognized by Turbo Drive

Content Dynamically Changed in DOM

Turbo Frames - Annotations

`src` - replace frame content with response

`target` - display response in a specific frame

`loading` - load content lazily or greedily

Turbo Frames

```
<turbo-frame id=”a” src=”/hello>

</turbo-frame>
```

src - Call URL and replace frame content with response

```
HTTP/1.1 200 OK
Content-Type: text/html;
charset=UTF-8
Date: Sat, 03 Sep 2022
11:20:41 GMT


<p>Hello!</p>
```

```
<turbo-frame id=”a” src=”/hello>

  <p>Hello!</p>

</turbo-frame>
```

Turbo Frames

```
<a href=/hello data-turbo-target=a>hello</a>

<turbo-frame id="gopher">

</turbo-frame>
```

`data-turbe-target` - display response in a specific frame

```
<a href=/hello data-turbo-target=a>hello</a>

<turbo-frame id="a" src="/hello>

  <p>Hello!</p>

</turbo-frame>
```

Turbo Streams

`target` - display response in a specific frame

`action` - how to influence target

action can be one of: **`update`**, **`replace`**, **`append`**, **`prepend`**, **`remove`**, **`before`** or **`after`**

Turbo Streams

Streams can affect any **`turbo-frame`**

Streams can target multiple frames

Use custom MIME type **`text/vnd.turbo-stream.html`**

Can be used with Websocket or SSE

Turbo Streams

```
<turbo-stream target=”gopher” action=”update”>

    <template>

        <p>Hello!</p>

    </template>

</turbo-frame>
```

```
<turbo-frame id=”gopher”>

  <p>Hello!</p>

</turbo-frame>
```

Turbo Drive - Navigation

`data-turbo` - disable/enable turbo magic on an element

`data-turbo-method` - use HTTP verb other than the default GET

`data-turbo-action` - move browser history forward or replace current history

Turbo Drive - Events

`turbo:*` - event hooks into framework

```
document.addEventListener('turbo:before-fetch-request', async (event) => {
    event.preventDefault();
    const token = await getSessionToken(window.app);
    event.detail.fetchOptions.headers['Authorization'] = `Bearer ${token}`;
    event.detail.resume();
})
```

How do you use Stimulus?

Stimulus

Lightweight way to add some Javascript to document elements

Annotate element with Stimulus and controller instance is automatically attached when the element is loaded

Stimulus

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

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

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

```
<div data-controller="hello">
 <input data-hello-target="name" type="text">

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

 <span data-hello-target="output"></span>

</div>
```

Takeaways

Personal Takeaways

Extremely nice to work with, development feels very productive

Very HTTP, everything is resource based

Acts like a static HTML page in the best case

Fast development cycle is more productive

Go Faster

Only one running component simplifies everything

- Necessary resources on deployment
- Local development
- Mental Load

Downsides

Thin Client - Lots of Communication

No data storage in the frontend

Rethink how web frontend works

After the MVP

Possible Evolutions

Break Monolith into Microservices => Frontend will stay mostly the same

Add RESTful API => (Mostly) Same Endpoints, just different content based on MIME type

Go Nowhere => If things are “Swiss-scale”, a monolith might be enough.

Thanks!

bespinian.io

Links & Special Thanks:

Basecamp: https://basecamp.com

Hotwire: https://basecamp.com

Gogohotwire: https://github.com/cldmstr/gogohotwire

Gopher Images Courtesy of and Copyright by Erick Zelaya:

https://github.com/jmzelaya/gopher-kart