Phoenix Live View

LiveView provides rich, real-time user experiences with server-rendered HTML.

— LiveView programming model is declarative
— LiveView will re-render the relevant parts of its HTML template and push it to the browser

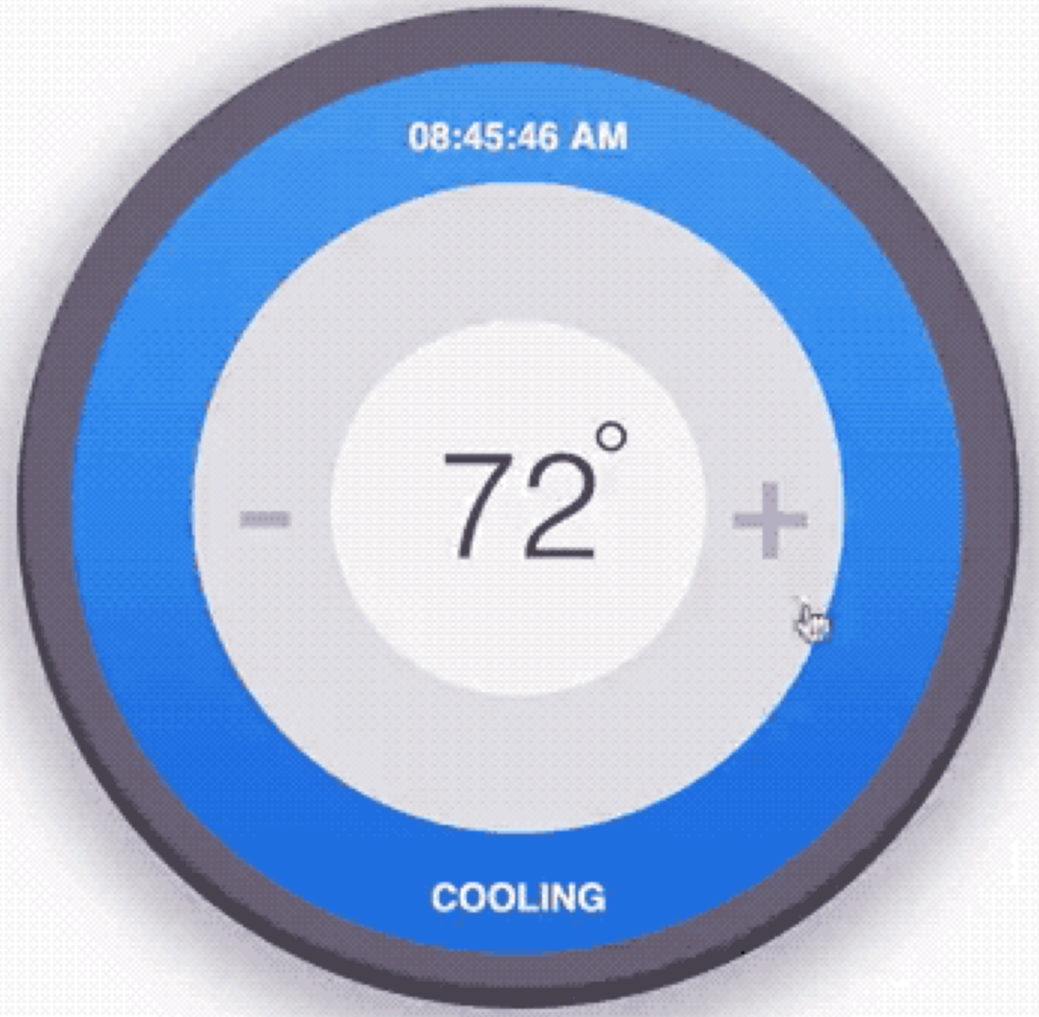
Interactive UIs

ThermostatView.render/1

```
def handle_event(“inc”, _, socket) do
  {:ok, update(socket, :val, &(&1 + 1)}
end
```

```
<div>
  <span><%= @val %>°</span>
  <a phx-click=“dec">-</a>
  <a phx-click="inc">+</a>
</div>
```

<a phx-click=“inc”>+


08:45:46 AM
72°
COOLING

```elixir
defmodule DemoWeb.PageController do
  use DemoWeb, :controller

  def thermostat(conn, _) do
    live_render(conn, DemoWeb.ThermoStatView)
  end
end
```

```
defmodule DemoWeb.ThermostatView do
  use Phoenix.LiveView
  import Calendar.Strftime

  def render(assigns) do
    ~L"""
    <div class="thermostat">

      <div class="bar <%= @mode %>">
        <a phx-click="toggle-mode"><%= @mode %></a>
        <span><%= strftime!(@time, "%r") %></span>
      </div>
      <div class="controls">
        <span class="reading"><%= @val %></span>
        <button phx-click="dec" class="minus">-</button>
        <button phx-click="inc" class="plus">+</button>
      </div>
    </div>
    """
  end
  # ...
end
```

```elixir
defmodule DemoWeb.ThermostatView do
  # ...
  def mount(_session, socket) do
    if connected?(socket), do: Process.send_after(self(), :tick, 1000)
    {:ok, assign(socket, val: 72, mode: :cooling, time: :calendar.local_time())}
  end

  def handle_info(:tick, socket) do
    Process.send_after(self(), :tick, 1000)
    {:noreply, assign(socket, time: :calendar.local_time())}
  end

  def handle_event("inc", _, socket) do
    {:noreply, update(socket, :val, &(&1 + 1))}
  end

  def handle_event("dec", _, socket) do
    {:noreply, update(socket, :val, &(&1 - 1))}
  end

  def handle_event("toggle-mode", _, socket) do
    {:noreply,
     update(socket, :mode, fn
       :cooling -> :heating
       :heating -> :cooling
     end)}
  end
end
```

LiveEEx Template

```
<div class=“bar”>
 <%= strftime!(@time, "%r") %>
 <span><%= @val %>°</span>
  <a phx-click=“dec">-</a>
  <a phx-click="inc">+</a>
</div>
```

Compiled

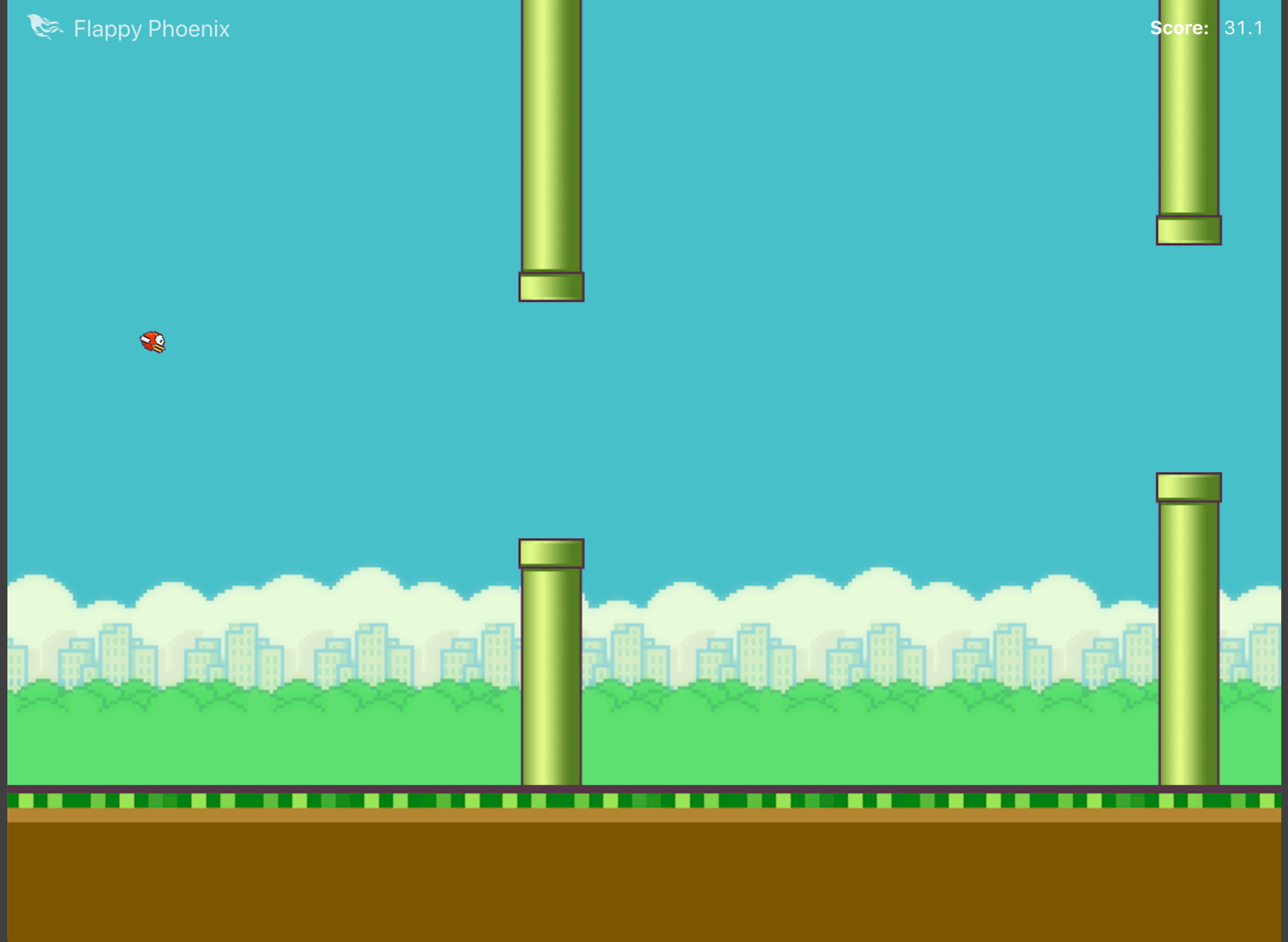
```
arg0 =
  if changed?(__changed__, :time) do
    strftime!(assigns.time, “%r”)
  end

arg1 =
  if changed?(__changed__, :val) do
    assigns.val
  end

%Phoenix.LiveView.Rendered{
  static: [
    "<div class=“bar”>\n,
    “\n<span>",
    "°</span><a phx-click=\”dec\”>…",
  ],
  dynamic: [arg0, arg1],
  fingerprint: 101666437924953863 29
}
```


09:57:43 AM
72°
COOLING

Demo time!

Official Examples

— Rainbow (don't do this at home)


Flappy Phoenix
Score: 31.1

Flappy Phoenix

How does it work?

- `Phoenix.LiveView.Channel` is built on top of `Phoenix.Channels`
- Works over a websocket
- Bidirectional communication
- `Client->Server` with special markup
- `Server->Client` with `.leex`, optimized diffing and `morphdom`

`Phoenix.LiveView.Engine`

— Performs diff tracking

— Tracks static and dynamic contents

— On the first render, LiveView sends the static contents and in future updates only the modified dynamic contents are resent.

LiveEEx Template

```
<div class=“bar”>
 <%= strftime!(@time, "%r") %>
 <span><%= @val %>°</span>
  <a phx-click=“dec">-</a>
  <a phx-click="inc">+</a>
</div>
```

Compiled

```
arg0 =
  if changed?(__changed__, :time) do
    strftime!(assigns.time, “%r”)
  end

arg1 =
  if changed?(__changed__, :val) do
    assigns.val
  end

%Phoenix.LiveView.Rendered{
  static: [
    "<div class=“bar”>\n,
    “\n<span>",
    "°</span><a phx-click=\”dec\”>…",
  ],
  dynamic: [arg0, arg1],
  fingerprint: 101666437924953863 29
}
```

`morphdom`

Lightweight module for morphing an existing DOM node tree to match a target DOM node tree. It's fast and works with the real DOM—no virtual DOM needed!

```
var morphdom = require('morphdom');

var el1 = document.createElement('div');
el1.className = 'foo';
el1.innerHTML = 'Hello John';

morphdom(el1, '<div class="bar">Hello Frank</div>');

expect(el1.className).to.equal('bar');
expect(el1.innerHTML).to.equal('Hello Frank');
```

phoenix_live_view.js

```
import LiveSocket from "live_view"
let liveSocket = new LiveSocket("/live")
liveSocket.connect()
```

Events

```
<button phx-click="dec" class="minus">-</button>
<button phx-click="inc" class="plus">+</button>
<input name="email" phx-focus="myfocus" phx-blur="myblur"/>
```

— Click, Key, Focus, Blur an Events supported

— When pushed, the value sent to the server will be the event's `key` (other methods are available as well, like `phx-value`).

— Bind context of evens via `phx-target`.

Forms

```
<button type="submit" phx-disable-with="Saving...">Save</button>
```

— Form inputs are set to `readonly` on submit

— Any HTML tag can be annotated, automatically restored

Loading State

- `"phx-connected"` - applied when the view has connected to the server
- `"phx-disconnected"` - applied when the view is not connected to the server
- `"phx-error"` - applied when an error occurs on the server.

Use Cases from Twitter

Phoenix Framework

The count is: 1

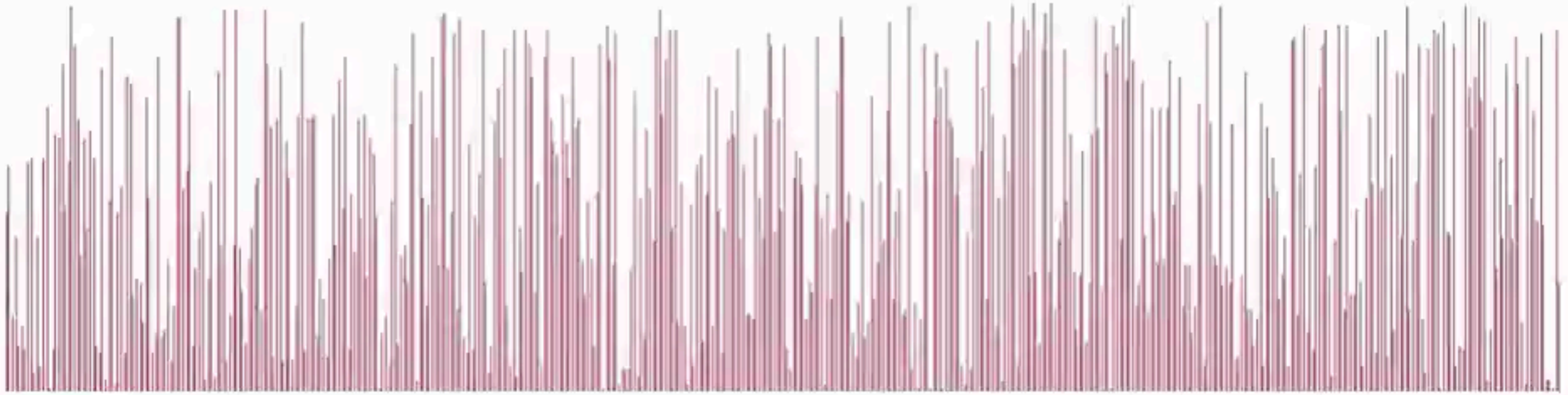
LvSample - Phoenix Framework

localhost:4000/realtime

133%

Phoenix Framework

Get Started



MetaTag

- Reports
- Team
- Billing
- Settings

alex@alexgaribay.c...
Sign Out

Reports

Edit Report

Cancel

1. Logistics | 2. Questions | 3. Respondents

REPORT NAME

Test

REPORT START TIME

6:52PM

REPORT CHANNEL

test-channel-rename

SCHEDULE

- [x] Monday
- [x] Tuesday
- [x] Wednesday
- [x] Thursday
- [x] Friday

Next

Phoenix Framework

Get Started

Inbox

From	Subject	
Sarah	Check out this new Elixir course	
John	re: LiveView is awesome	
Peter	Phoenix reaches 10 millions downloads on hex.pm	
Maria	CFP for ElixirConf closes in a few hours	

Welcome to WikiLiveView!

Here you can view all
recent events on wikipedia.com.

HOSTED ON GIGALIXIR | VIEW CODE | BUILT BY FELIX KLEMENT

Show 5 last recent events

Toyota ZZ engine
edit by 2601:583:C080:56A6:9C4F:A14D:B18A:B2F6

View

User:LauraHale/Motherhood in Francoist Spain
edit by LauraHale

View

Tetrahydroisoquinoline
edit by ChemNerd

corrected her mother's name

View

Tetrahydroisoquinoline
edit by ChemNerd

→Tetrahydroisoquinolines

View

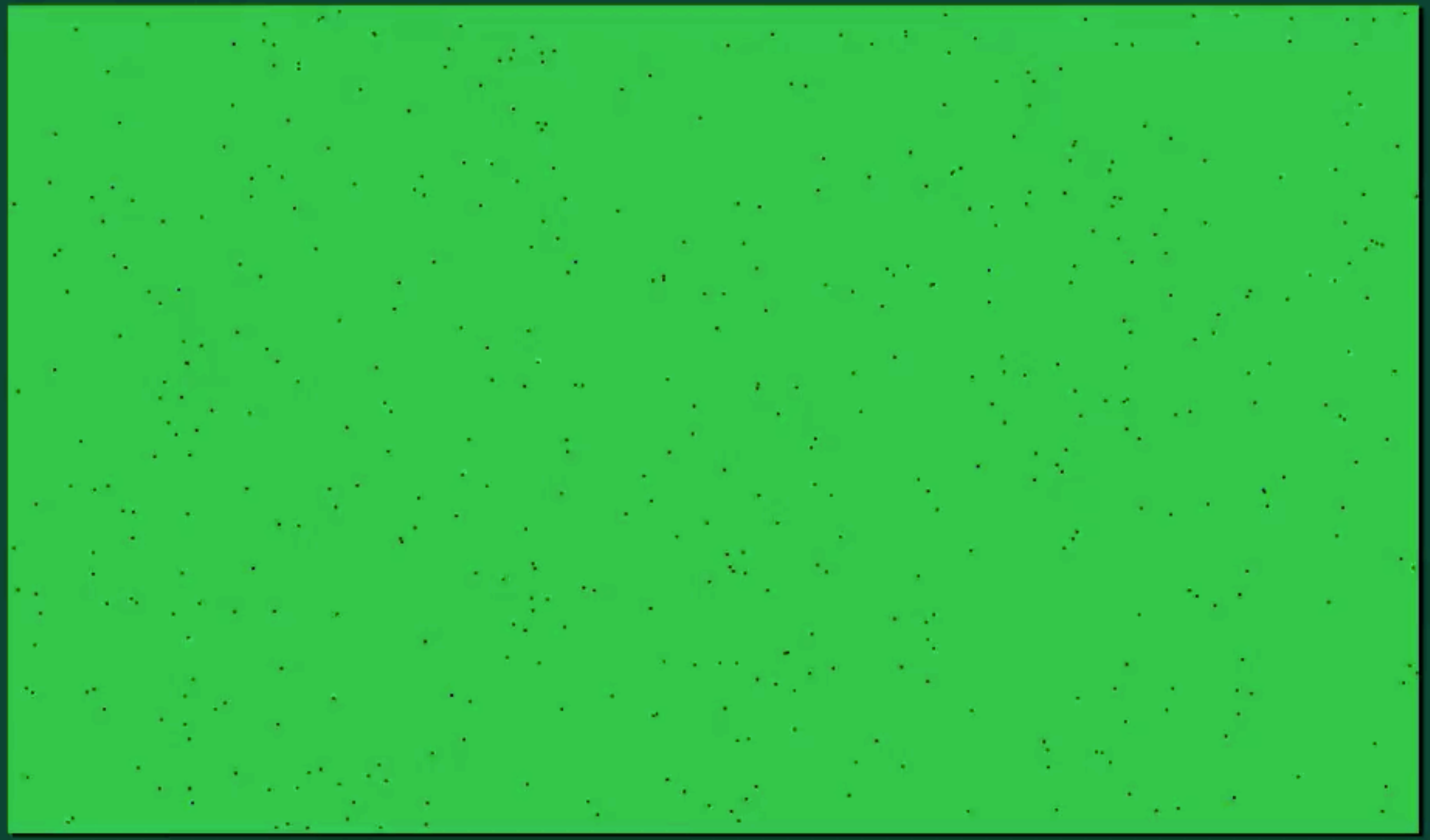
File:Ana María O'Neill.jpg
edit by Level C

View

Phoenix LiveView Ant Farm

Rendering 500 concurrent ants

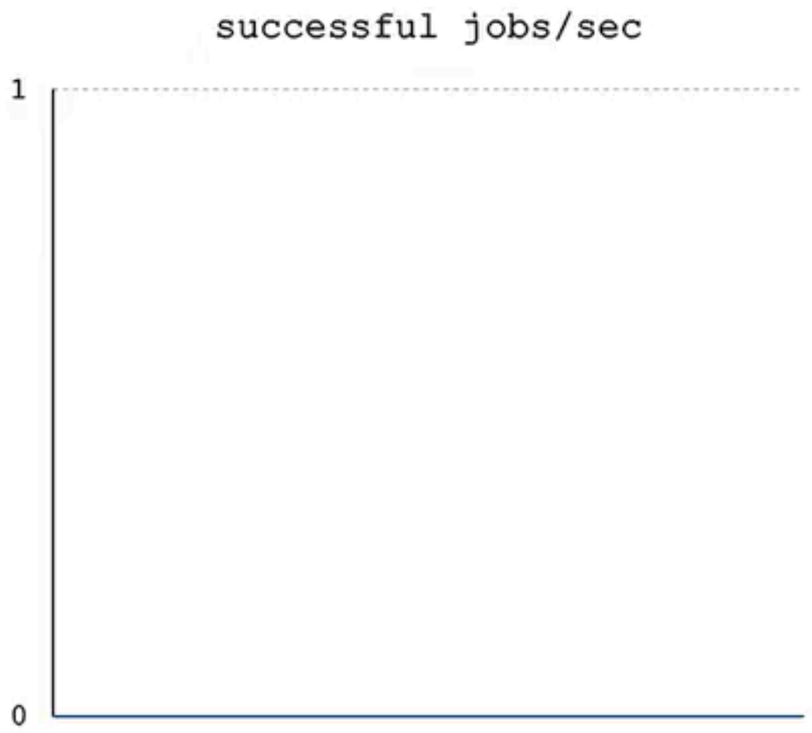
PLEASE DO NOT TAP ON GLASS

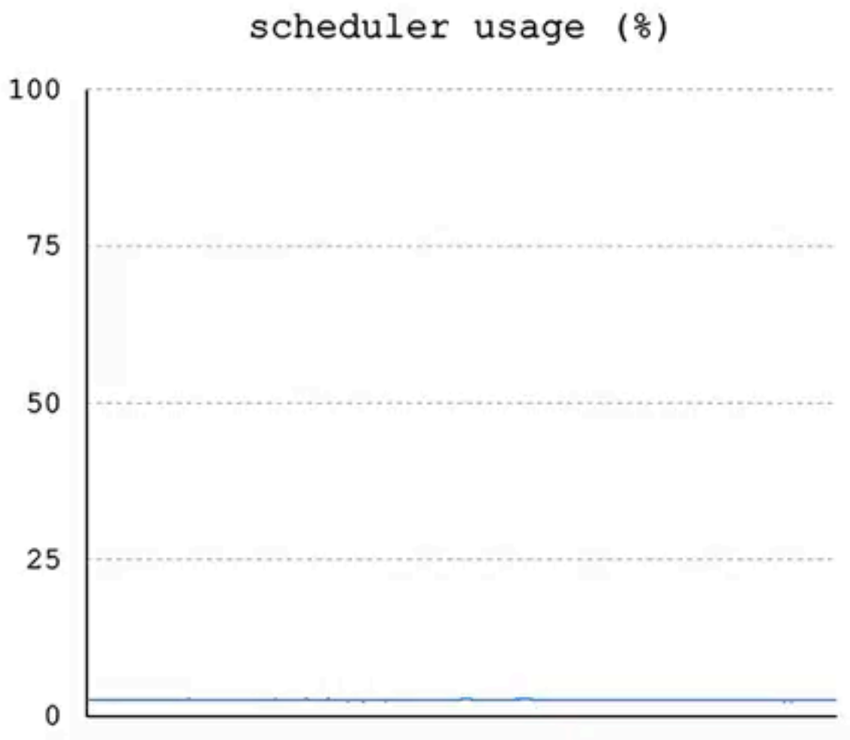


Crafted with ♥ by bigardone | Check out the source code

jobs	failures %	schedulers
0	0	1

job count	jobs/sec	scheduler usage	memory (MB)
0	0	3%	37





ObserverLive

Made by zorbash | source

HOME | NETWORK | SYSTEM | ETS | MNESIA | APP

Interval - 1000ms +

Erlang/OTP 20 [erts-9.3.3.3] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:10] [hipe] [kernel-poll:false]

System	Count / Limit	System Switch	Status	Memory Info	Size
Proc Count	311/262144	Smp Support	true	Allocated Mem	72.625 MB (100%)
Port Count	14/65536	Multi Scheduling	enabled	Used Mem	45.6895 MB (62.91%)
Atom Count	21523/1048576	Logical Processors	12	Unused Mem	26.9355 MB (37.09%)

Mem Type	Size	Mem Type	Size	IO/GC	Interval 1000ms
Total	47.52 MB	Binary	792.5469 KB	IO Output	21.3277 MB
Process	14.3183 MB	Code	15.3517 MB	IO Input	23.9757 MB
Atom	646.9072 KB	Reductions	185602	GC Count	48977
ETS	47.52 MB	RunQueue / ErrorLoggerQueue	1/0	GC Words Reclaimed	594411501

Schedulers

1 : 1.2331%	7 : 0.0981%	13 : 0.0%	19 : 0.0%
2 : 0.6359%	8 : 0.1053%	14 : 0.0%	20 : 0.0%
3 : 0.1022%	9 : 0.0996%	15 : 0.0%	21 : 0.0%
4 : 0.0949%	10 : 0.0965%	16 : 0.0%	22 : 0.0%
5 : 0.1015%	11 : 0.0943%	17 : 0.0%	23 : 0.0%
6 : 0.1024%	12 : 0.0984%	18 : 0.0%	24 : 0.0263%

No	PID	Memory	Name or Initial Call	Reductions	MsqQueue	Current Function
1	#PID<0.1194.0>	1.7204 MB	{:proc_lib, :init_p, 5}	29884965	0	{:recon_lib, :"-proc_attrs/1-lc$^0/1-0-", 3}
2	#PID<0.33.0>	1.0644 MB	application_controller	265592	0	{:gen_server, :loop, 7}
3	#PID<0.188.0>	811.9141 KB	Elixir.Phoenix.CodeReloader.Server	10648909	0	{:gen_server, :loop, 7}
4	#PID<0.53.0>	587.9141 KB	{:group, :server, 3}	9615	0	{:group, :more_data, 5}
5	#PID<0.1409.0>	449.4141 KB	{:proc_lib, :init_p, 5}	21446	0	{:cowboy_websocket, :loop, 3}
6	#PID<0.1190.0>	449.4141 KB	{:proc_lib, :init_p, 5}	65707	0	{:cowboy_websocket, :loop, 3}

(Coming soon)

— Use the Erlang Term Format to send messages to the client

— Include latency simulator, which allows you to simulate how your application behaves on increased latency

Sources

- Phoenix LiveView: Interactive, Real-Time Apps
- LiveView Project at ElixirConf 2018
- phoenix_live_view.ex
- phoenix_live_view/engine.ex
- phoenix_live_view.js

Sources

— Pablo Brudnick on Twitter

— Alex Garibay on Twitter

— Mike Binns on Twitter

— Plataformatec on Twitter

— Jørgen O. Erichsen on Twitter

— piacere on Twitter

— Ricardo García Vega on Twitter