# Introducing Orb 🕸

*Write WebAssembly with Elixir*

Orlando
ElixirConf® US

## Why WebAssembly?

Fast

Light

Sandboxed

Platform agnostic

Deterministic across architectures

Backwards-compatible W3C spec

## Every major platform

Browser

Server

Edge

Native

Spatial

# What is WebAssembly?

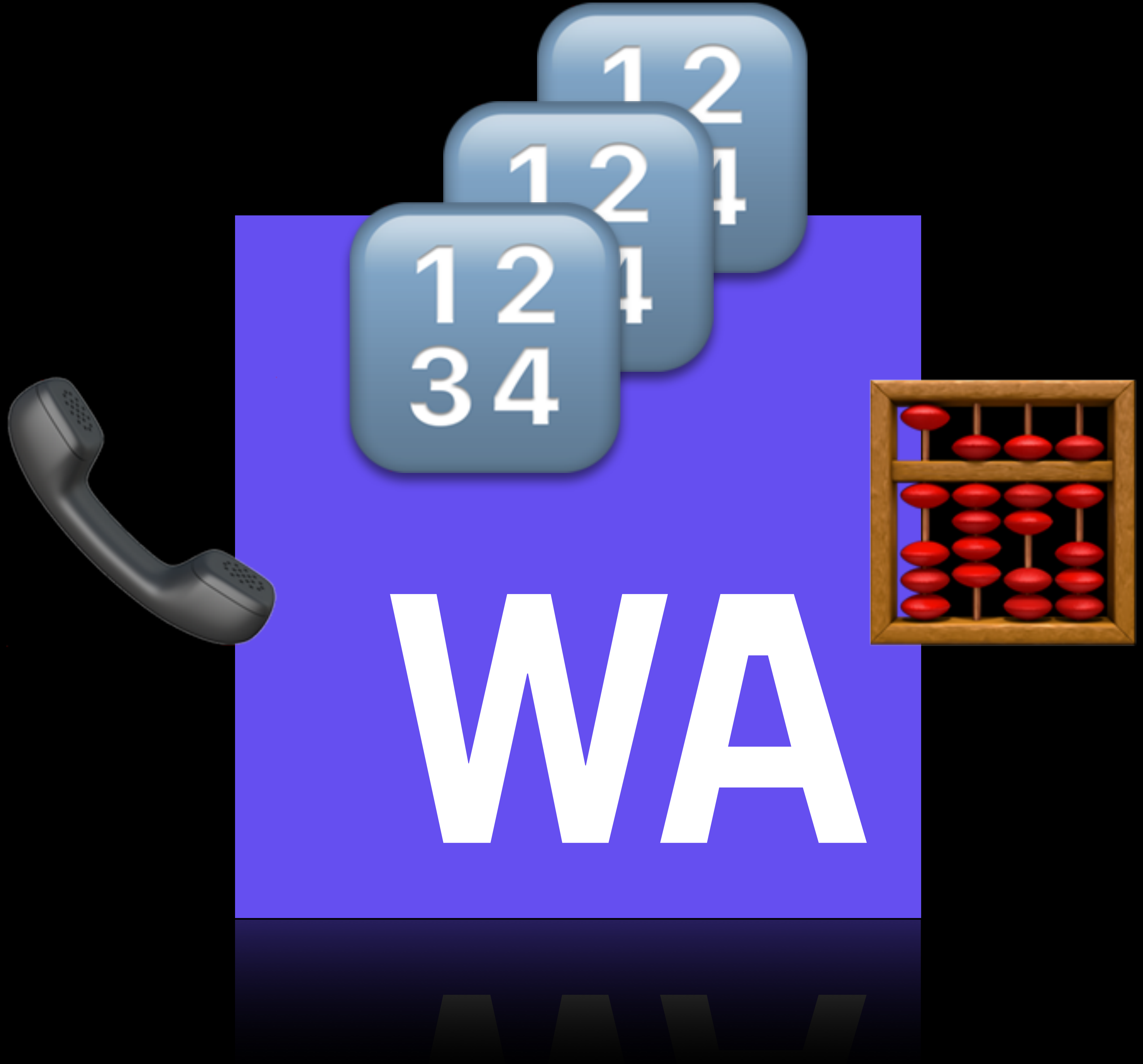
Everything is a number

Integer or float, 32 or 64-bit

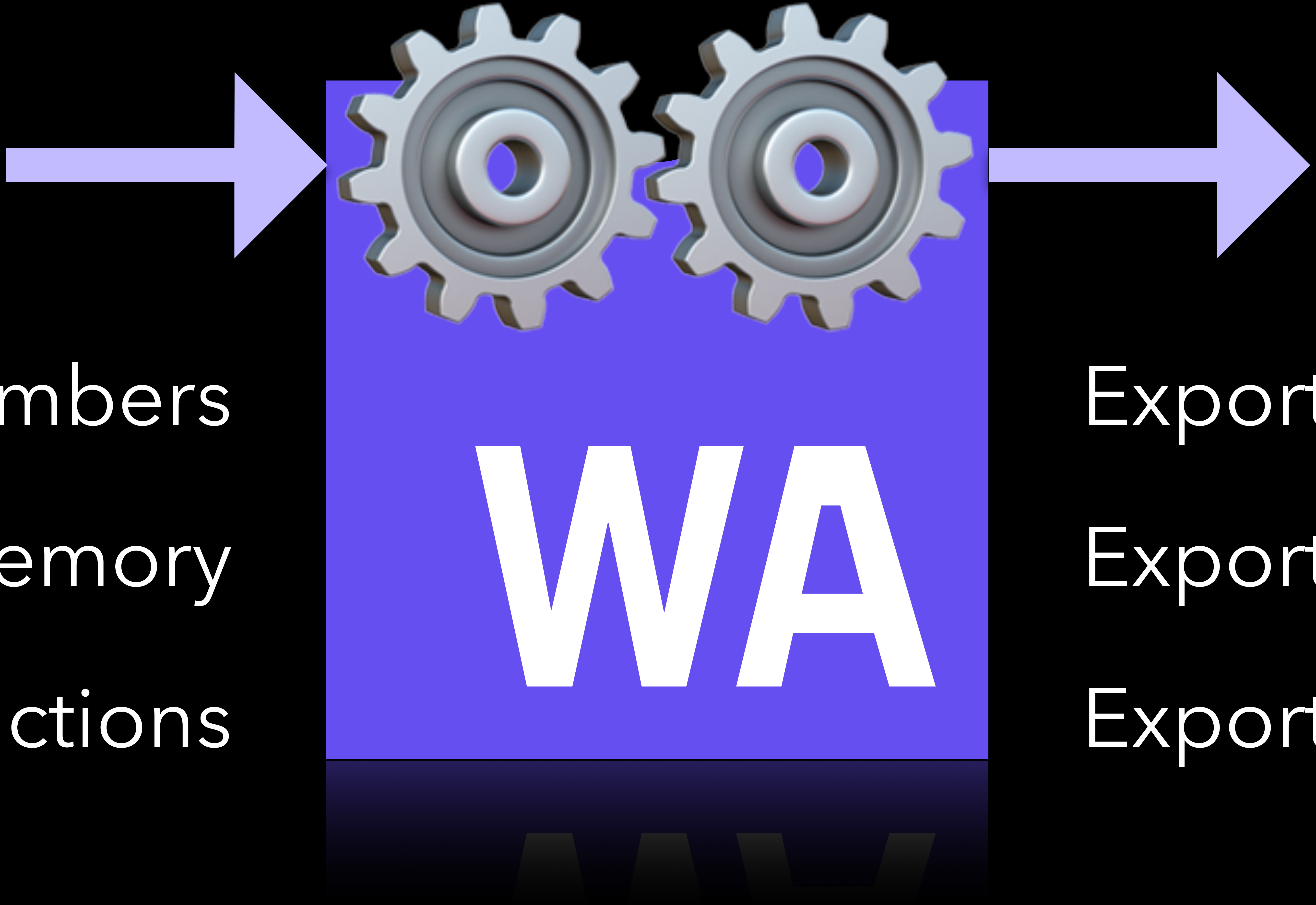
Array of memory

Mutable

No strings or data structures



## What is WebAssembly?



Import numbers

Import memory

Import functions

Export numbers

Export memory

Export functions

## Aims of Orb

*Feels like Elixir or Ruby*

*Built from the ground up for WebAssembly*

*Not tied to a particular runtime*

*Composable modules*

*Small output*

*Dynamically compile on-the-fly*

```
10 'This will draw 5 spheres
20 GOTO 160
50 IF VERT GOTO 100
60 CIRCLE (X,Y),R,C,,,.07
70 FOR I = 1 TO 5
80 CIRCLE (X,Y),R,C,,,I*.2:NEXT I
90 IF VERT THEN RETURN
100 CIRCLE (X,Y),R,C,,,1.3
110 CIRCLE (X,Y),R,C,,,1.9
120 CIRCLE (X,Y),R,C,,,3.6
130 CIRCLE (X,Y),R,C,,,9.8
140 IF VERT GOTO 60
150 RETURN
160 CLS:SCREEN 1:COLOR 0,1:KEY OFF:VERT=0
170 X=160:Y=100:C=1:R=50:GOSUB 50
180 X=30:Y=30:C=2:R=30:GOSUB 50
190 X=30:Y=169:GOSUB 50
200 X=289:Y=30:GOSUB 50
210 X=289:Y=169:GOSUB 50
220 LINE (30,30)-(289,169),1
230 LINE (30,169)-(289,30),1
240 LINE (30,169)-(289,30),1,B
250 Z$=INKEY$: IF Z$="" THEN 250
RUN
```

```elixir
defmodule WithinRange do
  use Orb

  defw validate(num: I32), I32, under?: I32, over?: I32 do
    under? = num < 1
    over? = num > 255

    not (under? or over?)
  end
end
```

## Math operators

```elixir
defmodule CalculateMean do
  use Orb

  I32.global(
    count: 0,
    tally: 0
  )

  defw insert(element: I32) do
    @count = @count + 1
    @tally = @tally + element
  end

  defw calculate_mean(), I32 do
    @tally / @count
  end
end
```

State

## Running Orb in Elixir

```
inst = Instance.run(CalculateMean)
Instance.call(inst, :insert, 4)
Instance.call(inst, :insert, 5)
Instance.call(inst, :insert, 6)
Instance.call(inst, :calculate_mean) # 5
```

Also see: wasmex

```
defw u32_to_hex_lower(value: I32, write_ptr: I32.U8.UnsafePointer), i: I32, digit: I32 do
  i = 8

  loop Digits do
    i = i - 1

    digit = I32.rem_u(value, 16)
    value = value / 16

    if digit > 9 do
      write_ptr[at!: i] = ?a + digit - 10
    else
      write_ptr[at!: i] = ?0 + digit
    end

    Digits.continue(if: i > 0)
  end
end
```

## Loops and control flow

```
defw u32_to_hex_lower(value: I32, write_
  i = 8


  loop Digits do
    i = i - 1
```

Loops and control flow

```
    else
      write_ptr[at!: i] = ?0 + digit
    end

    Digits.continue(if: i > 0)
  end
end
```

## Loops and control flow

```
value = value / 16

if digit > 9 do
  write_ptr[at!: i] = ?a + digit - 10
else
  write_ptr[at!: i] = ?0 + digit
end
```

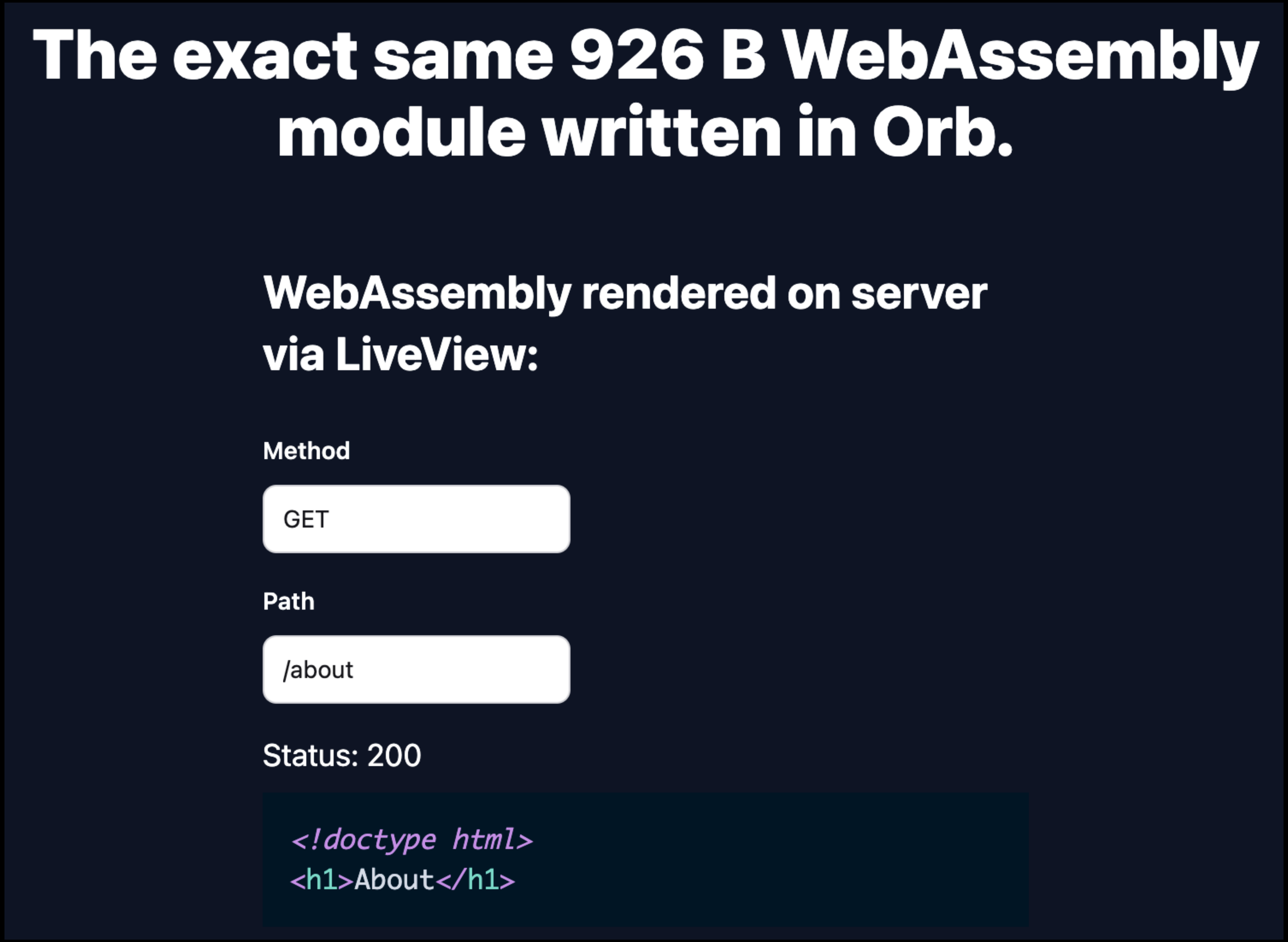
## Loops and control flow

```
defw u32_to_hex_lower(value: I32, write_ptr: I32.U8.UnsafePointer), i: I32, digit: I32 do
  i = 8

  loop Digits do
    i = i - 1

    digit = I32.rem_u(value, 16)
    value = value / 16

    if digit > 9 do
      write_ptr[at!: i] = ?a + digit - 10
    else
      write_ptr[at!: i] = ?0 + digit
    end

    Digits.continue(if: i > 0)
  end
end
```

## Loops and control flow

## HTTP response demo

components.guide/wasm/http-server



**The exact same 926 B WebAssembly module written in Orb.**

**WebAssembly rendered on server via LiveView:**

Status: 200

```
<!doctype html>
<h1>About</h1>
```

```
I32.String.match @path do
  ~S"/" ->
    200

  ~S"/about" ->
    200

  _ ->
    404
end
```

Strings

```
defw get_status(), I32 do
  I32.String.match @method do
    ~S"GET" ->
      I32.String.match @path do
        ~S"/" ->
          200

        ~S"/about" ->
          200

        _ ->
          404
      end

    _ ->
      405
  end
end
```

## Strings

## Color picker demo

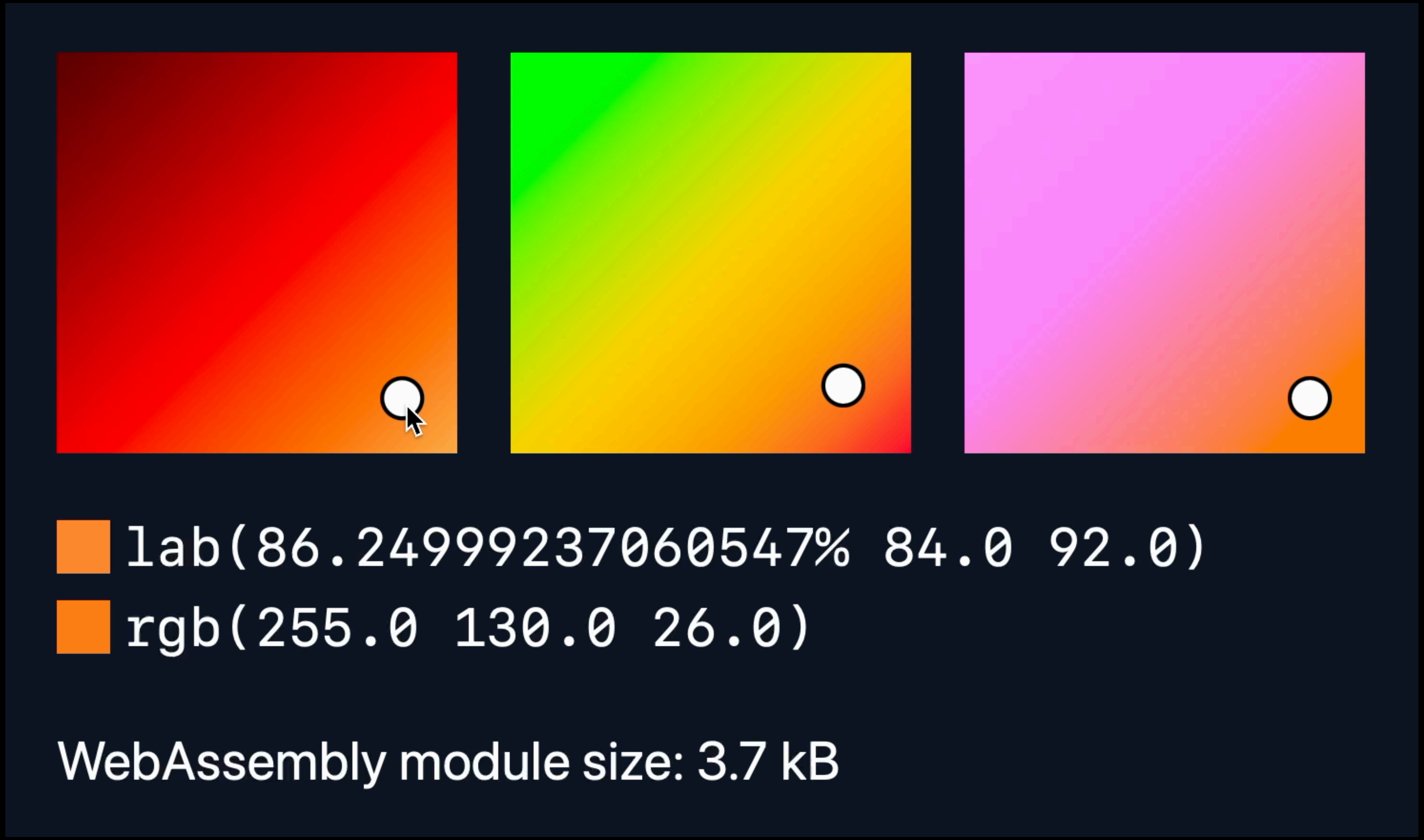
components.guide/wasm-demo/color

Color math in WebAssembly

SVG & HTML rendering in WebAssembly

Runs both on server and browser

JavaScript forwards events to WebAssembly instance



🟧 lab(86.24999237060547% 84.0 92.0)
🟧 rgb(255.0 130.0 26.0)

WebAssembly module size: 3.7 kB

```
defwp swatch_svg(component_id: I32), I32.String do
  build! do
    ~S(<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1 1" width=")
    @swatch_size
    ~S(" height=")
    @swatch_size
    ~S(" class="touch-none" data-action )
```

## HTML / SVG rendering

```
defwp swatch_svg(component_id: I32), I32.String do
  build! do
    ~S(<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1 1" width=")
    @swatch_size
    ~S(" height=")
    @swatch_size
    ~S(" class="touch-none" data-action )

    if I32.eq(component_id, @component_l),
      do: ~S{data-pointerdown="l_changed" data-pointerdown+pointermove="l_changed"}

    if I32.eq(component_id, @component_a),
      do: ~S{data-pointerdown="a_changed" data-pointerdown+pointermove="a_changed"}

    if I32.eq(component_id, @component_b),
      do: ~S{data-pointerdown="b_changed" data-pointerdown+pointermove="b_changed"}
```

## HTML / SVG rendering

```elixir
def root(conn, %{"module" => "color"}) do
  instance = Instance.run(LabSwatch, color_imports())
  initial_html = Instance.call_reading_string(instance, :to_html)
  wasm_size = byte_size(LabSwatch.to_wasm())

  render(conn, :color,
    initial_html: initial_html,
    page_title: "WebAssembly Lab Color Picker using Orb",
    wasm_size: wasm_size
  )
end
```

## Server rendering HTML in Phoenix

```
<wasm-simple-html class="block">
  <source src="/wasm/module/color_lab_swatch.wasm" type="application/wasm" />
  <%= raw(@initial_html) %>
</wasm-simple-html>

function update() {
  freeAll?.apply();
  const html = memoryIO.readString(toHTML());
  el.innerHTML = html;
}

el.addEventListener("click", (event) => {
  const action = event.target.dataset.action;
  if (typeof action === "string") {
    instance.exports[action]?.apply();
    update();
  }
});
```

## Client rendering HTML with JavaScript

# Compose modules

```elixir
defmodule BumpAllocator do
  use Orb

  I32.global(
    bump_offset: 0xFF,
    bump_mark: 0
  )

  Memory.pages(1)

  defwi bump_alloc(size: I32), I32.UnsafePointer,
    ptr: I32.UnsafePointer do
    ptr = @bump_offset
    @bump_offset = @bump_offset + size
    ptr
  end
end
```

# Compose modules

```elixir
defmodule BumpAllocator do
  use Orb

  I32.global(
    bump_offset: 0xFF,
    bump_mark: 0
  )

  Memory.pages(1)

  defwi bump_alloc(size: I32), I32.UnsafePointer,
    ptr: I32.UnsafePointer do
    ptr = @bump_offset
    @bump_offset = @bump_offset + size
    ptr
  end
end
```

```elixir
defmodule MyModule do
  use Orb

  Memory.pages(1)
  BumpAllocator.include()

  defw example(), ptr: I32.UnsafePointer do
    ptr = BumpAllocator.bump_alloc(42)
    # Do something with ptr
  end
end
```

# Compose modules

```elixir
defmodule BumpAllocator do
  use Orb

  I32.global(
    bump_offset: 0xFF,
    bump_mark: 0
  )

  Memory.pages(1)

  defwi bump_alloc(size: I32), I32.UnsafePointer,
    ptr: I32.UnsafePointer do
    ptr = @bump_offset
    @bump_offset = @bump_offset + size
    ptr
  end
end
```

```elixir
defmodule MyModule do
  use Orb
  use BumpAllocator

  defw example(), ptr: I32.UnsafePointer do
    ptr = bump_alloc(42)
    # Do something with ptr
  end
end
```

## Compose modules

```
use Orb
use BumpAllocator
use StringBuilder
use URLEncoded
```

hex

# SilverOrb: std lib

```
{:silver_orb, "~> 0.0.3"}
```

Bump allocator

Number formatters

String builder

URL encoding

ASCII & UTF8 utilities

Iterables

```
use Orb
use SilverOrb.BumpAllocator
use SilverOrb.StringBuilder
use SilverOrb.URLEncoded
```

# State machines

```elixir
defmodule Loader do
  use Orb
  use StateMachine

  I32.export_enum([:idle, :loading, :loaded, :failed])

  defw(get_current(), I32, do: @state)

  on(load(@idle), target: @loading)
  on(success(@loading), target: @loaded)
  on(failure(@loading), target: @failed)
end
```

```
defw can_edit?(post_id: I32, author_id: I32), I32 do
  inline do
    case CurrentUser.get() do
      %{type: :viewer, id: user_id} ->
        wasm do
          author_id === user_id
        end

      %{type: :admin} ->
        1
    end
  end
end
```

Dynamically compile on-the-fly

The Elixir ecosystem at compile time, the WebAssembly ecosystem at runtime modules

hex.pm

The Elixir ecosystem at compile time, the WebAssembly ecosystem at runtime

community

macros

modules portable hex.pm

safe

The Elixir ecosystem at compile time, the WebAssembly ecosystem at runtime

fast

community predictable macros

github.com/RoyalIcing/Orb

github.com/RoyalIcing/Orb

github.com/RoyalIcing/Orb

github.com/RoyalIcing/Orb

github.com/RoyalIcing/Orb

**github.com/RoyalIcing/Orb**

twitter.com/@royalicing

hachyderm.io/@royalicing

@royalicing.bsky.social

*Thank you.*