The **Excitement** of **Elixir**

The **Speaker**

Devin Torres
@devinus

The **Platform**

All the benefits of OTP, wrapped in a blueberry pancake

Fault-tolerant

Write once, run forever

Concurrent

The recent years have seen much work in Erlang/OTP on making the runtime concurrently parallel and we are reaping the benefits.

— Jesper Louis Andersen

Distributed

Computing tools of the future will have to be very good at distributed systems.

— Justin Sheehy, CTO at Basho


ERLANG
DEAL WITH IT
MakeAGIF.com

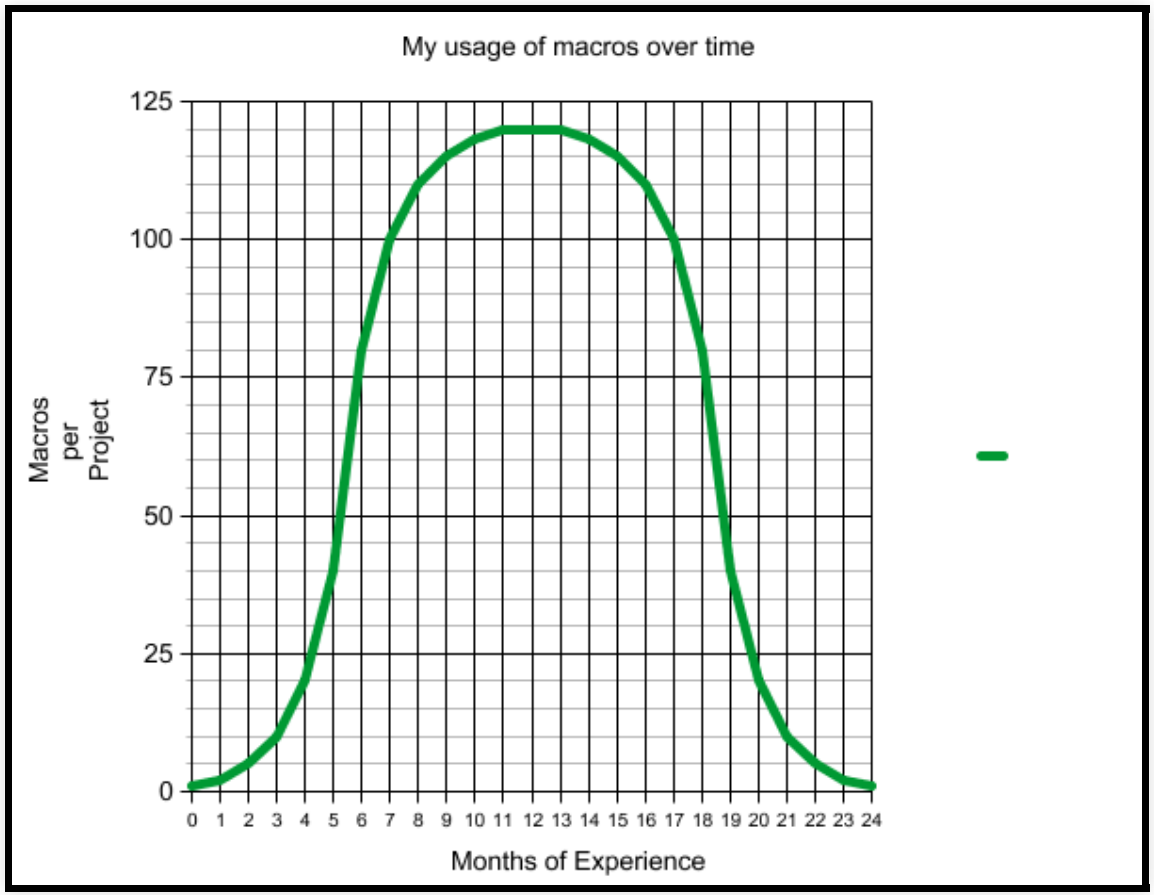
The **Language**

Balancing practicality and power

Macros

With great power, comes great responsibility

— Uncle Ben


My usage of macros over time
Macros per Project
125
100
75
50
25
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Months of Experience

Code that writes code

```elixir
defmodule MIME do
  @external_resource "mime.types"
  stream = File.stream!("mime.types")

  mapping = Enum.flat_map(stream, fn (line) ->
    if String.match?(line, ~r/^[#\n]/) do
      []
    else
      [type|exts] = String.split(String.strip(line))
      [{type, exts}]
    end
  end)

  for { type, exts } <- mapping, ext <- exts do
    def type(unquote(ext)) do
      unquote(type)
    end
  end
end
```

Protocols

```elixir
defmodule Nested do
  alias Nested.Access

  import Kernel, except: [put_in: 3]

  def get_in(thing, [key]) do
    Access.get(thing, key)
  end

  def get_in(thing, [key | rest]) do
    get_in(Access.get(thing, key), rest)
  end

  def put_in(thing, [key], value) do
    Access.put(thing, key, value)
  end

  def put_in(thing, [key | rest], value) do
    Access.put(thing, key, put_in(Access.get(thing, key), rest, value))
  end
end
```

defprotocol

```
defprotocol Nested.Access do
  def get(thing, key)
  def put(thing, key, value)
end
```

defimpl

```
defimpl Nested.Access, for: Map do
  def get(map, key) do
    Map.get(map, key)
  end

  def put(map, key, value) do
    Map.put(map, key, value)
  end
end
```

```
defimpl Nested.Access, for: HashDict do
  def get(map, key) do
    HashDict.get(map, key)
  end

  def put(map, key, value) do
    HashDict.put(map, key, value)
  end
end
```

Structs

The power of records, the flexibility of maps

```elixir
defmodule Character do
  @derive [Access]
  defstruct [name: nil, level: 1]
end

char = %Character{foo: "bar"} #=> CompileError

char = %Character{name: "Devinus Maximus"}

char[:name] #=> "Devinus Maximus"

defimpl Inspect, for: Character do
  def inspect(char, _opts) do
    "#{char.name} (#{char.level})"
  end
end

inspect(char) #=> "Devinus Maximus (1)"
inspect(char, structs: false)
#=> "%{__struct__: Character, level: 1, name: \"Devinus Maximus\"}"

char = Map.delete(char, :__struct__) # See also: Map.from_struct(char)
inspect(char) #=> %{level: 1, name: "Devinus Maximus"}
```

The **Standard Library**

Batteries included?

Nested Access

```elixir
party = %{group: [%Character{name: "Devin"}, %Character{name: "José"}]}

all = fn :get_and_update, data, next ->
  Enum.map(data, next) |> List.unzip() |> List.to_tuple()
end

# Level up!
get_and_update_in(party, [:group, all, :level], &{&1, &1 + 1}
#=> {[1, 1], %{group: [Devin (2), José (2)]}}
```

Agents

```elixir
defmodule World do
  @name __MODULE__

  def start_link do
    Agent.start_link(fn -> %{rooms: [], players: []} end, name: @name)
  end

  def add_player(player) do
    Agent.update(@name, &Map.put(&1, :players, [player | &1.players]))
  end

  def who do
    players = Agent.get(@name, &(&1.players))
    [Enum.map(players, &[inspect(&1), ?\n]), "Total: #{length(players)}"]
  end
end

World.start_link
World.add_player(%Character{name: "Devin"})
World.add_player(%Character{name: "Kurt"})
IO.puts World.who
# Kurt (1)
# Devin (1)
# Total: 2
```

Tasks

```
defmodule FantasyNames do
  def generate do
    Enum.map(1..5, fn _ -> request end)
  end

  defp request do
    IO.puts "Requesting fantasy name..."
    %{body: body} = HTTPoison.get("http://fantasy-name-generator.info/get
    body
  end
end
```

```elixir
defmodule FantasyNames do
  def generate do
    1..5 |> Enum.map(fn _ -> request end) |> Enum.map(&Task.await(&1))
  end

  defp request do
    Task.async(fn ->
      IO.puts "Requesting fantasy name..."
      %{body: body} = HTTPoison.get("http://fantasy-name-generator.info/g
      body
    end)
  end
end
```

Streams

```elixir
defmodule Monsters do
  def sewer_rat do
    %Character{name: "Sewer Rat", level: 1}
  end

  def hoodlum do
    %Character{name: "Hoodlum", level: 5}
  end
end

neverending_supply_of_rats = Stream.repeatedly(&Monsters.sewer_rat/0)
neverending_supply_of_hoodlums = Stream.repeatedly(&Monsters.hoodlum/0)

rat_distribution = Stream.take(neverending_supply_of_rats, 4)
hoodlum_distribution = Stream.take(neverending_supply_of_hoodlums, 2)

baddies = Stream.cycle(Stream.concat(rat_distribution, hoodlum_distributi

World.add_room(%{name: "Sewer Junction", monsters: Enum.take(baddies, 2)}
#=> %{name: "Sewer Junction", monsters: [Sewer Rat (1), Sewer Rat (1)]}

World.add_room(%{name: "Gang Headquarters", monsters: Enum.take(baddies,
#=> %{name: "Gang Headquarters",
#     monsters: [Sewer Rat (1), Sewer Rat (1), Sewer Rat (1), Sewer Rat (
#                Hoodlum (5)]}
```

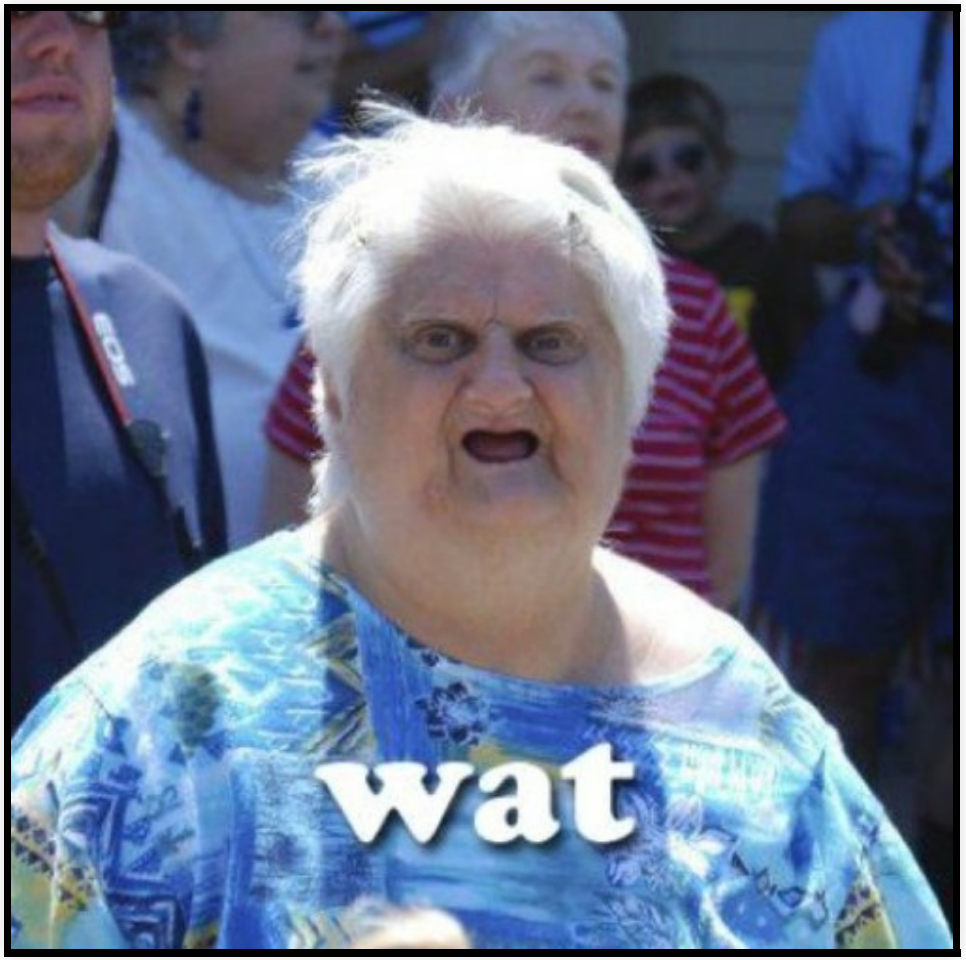
Bonus: Chunked `pmap`

```elixir
defmodule CrazyTown do
  def map(collection, chunk_size \\ 10, mapper) do
    collection
      |> Stream.chunk(chunk_size)
      |> Stream.flat_map(fn (chunk) ->
           Stream.map(chunk, &Task.async(fn -> mapper.(&1) end))
         end)
      |> Stream.map(&Task.await(&1))
  end
end

Enum.take(CrazyTown.map(1..100, &(&1 * 2)), 30)
#=> [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,
#    40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60]
```

Logger

```
=ERROR REPORT==== 20-Jun-2014::15:39:10 ===
** Generic server <0.250.0> terminating
** Last message in was {'EXIT',<0.249.0>,
                           {{badmatch,{error,etimedout}},
                            [{poolboy,new_worker,1,
                                 [{file,"src/poolboy.erl"},{line,237}]},
                             {poolboy,prepopulate,3,
                                 [{file,"src/poolboy.erl"},{line,258}]},
                             {poolboy,init,3,
                                 [{file,"src/poolboy.erl"},{line,122}]},
                             {gen_server,init_it,6,
                                 [{file,"gen_server.erl"},{line,306}]},
                             {proc_lib,init_p_do_apply,3,
                                 [{file,"proc_lib.erl"},{line,239}]}]}}
** When Server state == {state,
                        {<0.250.0>,poolboy_sup},
                        simple_one_for_one,
                        [{child,undefined,cberl_worker,
                          {cberl_worker,start_link,
                           [[{host,
                              "fantasy-name-generator.info:8091"},
                             {username,"devin"},
                             {bucketname,"mud"},
                             {transcoder,'Elixir.Couchie.Transcoder'}]]
},
                          temporary,5000,worker,
                          [cberl_worker]}],
                        {set,5,16,16,8,80,48,
                         {[],[],[],[],[],[],[],[],[],[],[],[],[],[],[],
[]},
```

uhm...



Task.Supervisor

```elixir
defmodule Foo do
  use Application

  def start(_type, _args) do
    import Supervisor.Spec, warn: false

    children = [
      supervisor(Task.Supervisor, [[name: :tasks_sup]]),
      worker(Task, [fn -> :timer.sleep(5000); raise "Kurt stinks" end])
    ]

    opts = [strategy: :one_for_one, name: Foo.Supervisor]
    Supervisor.start_link(children, opts)
  end
end
```

```
=ERROR REPORT==== 25-Jul-2014::15:54:02 ===
** Task <0.68.0> terminating
** Started from 'Elixir.Foo.Supervisor'
** When function  == #Fun
**      arguments == []
** Reason for termination ==
** {#{'__exception__' => true,
      '__struct__' => 'Elixir.RuntimeError',
      message => <<"Kurt stinks">>},
    [{'Elixir.Foo','-start/2-fun-0-',0,[{file,"lib/foo.ex"},{line,13}]},
     {'Elixir.Task.Supervised',do_apply,2,
                               [{file,"lib/task/supervised.ex"},{line,70
}]},
     {proc_lib,init_p_do_apply,3,[{file,"proc_lib.erl"},{line,239}]}]}
```

```
15:56:31.388 [error] ** Task #PID<0.133.0> terminating
** Started from Foo.Supervisor
** When function  == #Function<0.124279262/0 in Foo.start/2>
**      arguments == []
** Reason for termination ==
** {%RuntimeError{message: "Kurt stinks"},
 [{Foo, :"-start/2-fun-0-", 0, [file: 'lib/foo.ex', line: 13]},
  {Task.Supervised, :do_apply, 2, [file: 'lib/task/supervised.ex', line:
 70]},
  {:proc_lib, :init_p_do_apply, 3, [file: 'proc_lib.erl', line: 239]}]}
```

The **Tooling**

- IEx
- Mix
- ExUnit
- Hex
- exrm

IEx

```
iex(1)> h

• c/2       — compiles a file at the given path
• cd/1      — changes the current directory
• clear/0   — clears the screen
• flush/0   — flushes all messages sent to the shell
• h/0       — prints this help message
• h/1       — prints help for the given module, function or macro
• l/1       — loads the given module's beam code and purges the current
version
• ls/0      — lists the contents of the current directory
• ls/1      — lists the contents of the specified directory
• pwd/0     — prints the current working directory
• r/1       — recompiles and reloads the given module's source file
• respawn/0 — respawns the current shell
• s/1       — prints spec information
• t/1       — prints type information
• v/0       — prints the history of commands evaluated in the session
• v/1       — retrieves the nth value from the history
• import_file/1             — evaluates the given file in the shell's co
ntext
```

```elixir
defmodule Combat do
  def fight(attacker = %Character{level: attacker_level},
            victim   = %Character{level: victim_level}) do
    level_difference = (victim_level - attacker_level)

    # Attacker always has advantage
    if level_difference > 0 do
      "#{victim.name} #{outcome(level_difference)} #{attacker.name}."
    else
      "#{attacker.name} #{outcome(level_difference)} #{victim.name}."
    end
  end

  defp outcome(diff) when diff > 100 do
    "utterly destroys"
  end

  defp outcome(diff) when diff > 10 do
    "decimates"
  end

  defp outcome(_diff) do
    "narrowly defeats"
  end
end

devin = %Character{name: "Devin", level: 10}
jose  = %Character{name: "José",  level: 10_000_00}
Combat.fight(devin, jose)
#=> "José utterly destroys Devin."
```

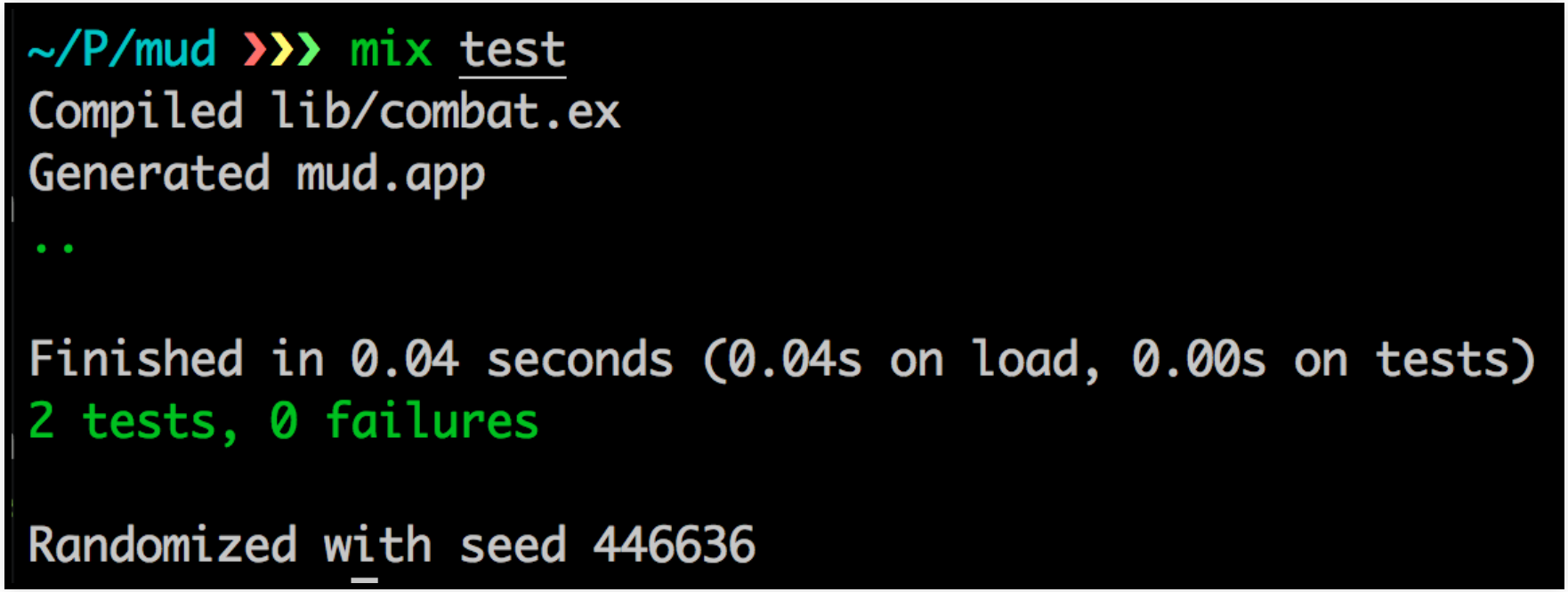
IEx.pry

```
defmodule Combat do
  require IEx

  def fight(attacker = %Character{level: attacker_level},
            victim   = %Character{level: victim_level}) do
    level_difference = (victim_level - attacker_level)

    IEx.pry

    # ...
  end

  # ...
end
```

```
iex(5)> Combat.fight(devin, jose)
Request to pry #PID<0.41.0> at iex:9. Allow? [Yn] Y

Interactive Elixir (0.14.3) - press Ctrl+C to exit (type h() ENTER for help)
pry(1)> level_difference
999990
pry(2)>
```

Doctests

```elixir
defmodule Combat do
  @doc """
  Engages two characters in mortal combat.

  > Excellent.

  iex> devin = %Character{name: "Devin", level: 10}
  iex> jose  = %Character{name: "José",  level: 10_000_00}
  iex> Combat.fight(devin, jose)
  "José utterly destroys Devin."
  """
  def fight(attacker = %Character{level: attacker_level},
            victim   = %Character{level: victim_level}) do
    # ...
  end

  # ...
end

defmodule CombatTest do
  use ExUnit.Case

  doctest Combat
end
```


~/P/mud ››› mix test
Compiled lib/combat.ex
Generated mud.app
..
Finished in 0.04 seconds (0.04s on load, 0.00s on tests)
2 tests, 0 failures
Randomized with seed 446636

Hex

```elixir
defmodule Execjs.Mixfile do
  use Mix.Project

  def project do
    [app: :execjs,
     version: "0.1.0",
     description: "Run JavaScript code from Elixir",
     package: package]
  end

  defp package do
    [files: ~w(lib priv README.md UNLICENSE VERSION),
     contributors: ["Devin Torres"],
     licenses: ["Unlicense"],
     links: [{"GitHub", "https://github.com/devinus/execjs"}]]
  end
end
```

```
$ mix hex.publish
Publishing execjs v0.1.0
  Dependencies:
  Excluded dependencies (not part of the Hex package):
  Included files:
    lib/execjs.ex
    lib/execjs/escape.ex
    lib/execjs/runtime.ex
    lib/execjs/runtimes.ex
    priv/jsc_runner.js.eex
    priv/node_runner.js.eex
    priv/rhino_runner.js.eex
    priv/spidermonkey_runner.js.eex
    README.md
    UNLICENSE
    VERSION
Proceed? [Yn] Y
Published execjs v0.1.0
```

The **Ecosystem**

Plug

```elixir
defmodule Plug.Head do
  @behaviour Plug

  def init([]) do
    []
  end

  def call(conn, []) do
    if conn.method == "HEAD" do
      %{conn | method: "GET"}
    else
      conn
    end
  end
end
```

Ecto

```
defmodule PlayerRanking do
  import Ecto.Query

  def newbies do
    query = from p in Player,
          where: p.level < 10,
         select: p
    Repo.all(query)
  end
end
```

Honorable mention

- Phoenix
- Calliope
- Hound
- Jazz

The **Dream**

- Breakpoint capable debugger
- QuickCheck for Elixir
- Lazy parsing framework

We're hiring!

ClutchAnalytics