Elixir ❤ Resilience

Expectations

i("Jearvon Dharrie")

```
DataType
  Developer
Name
  Jearvon Dharrie
Description
  Works at Comcast
ElixirApps
  2
```

Resilient

(of a substance or object) able to recoil or spring back into shape after bending, stretching, or being compressed.

Elixir

Elixir is a dynamic, functional language designed for building scalable and maintainable applications.

Elixir **leverages the Erlang VM**, known for running **low-latency, distributed and fault-tolerant systems**, while also being successfully used in web development and the embedded software domain.

URL Shortener

Requirements

- Shortens a URL
- Expands a URL
- HTTP GET/POST
- Scalable and fault tolerant

Language

Interactive Development

`iex`

```
iex(1)> IO.puts("Hello QCon")
Hello QCon
:ok
iex(2)>
```

Mix

```
mix new urlz --sup
mix test
```

Umbrella Apps

Web Frameworks

- Phoenix
- Plug

Immutable

```
attendees = ["tammy", "mike", "tejesh"]
["tammy", "mike", "tejesh"]
new_attendees = ["jearvon" | attendees]
["jearvon", "tammy", "mike", "tejesh"]
attendess
["tammy", "mike", "tejesh"]
```

Erlang Interop

```
:observer.start()
:rand.uniform(100)
```

Compiler

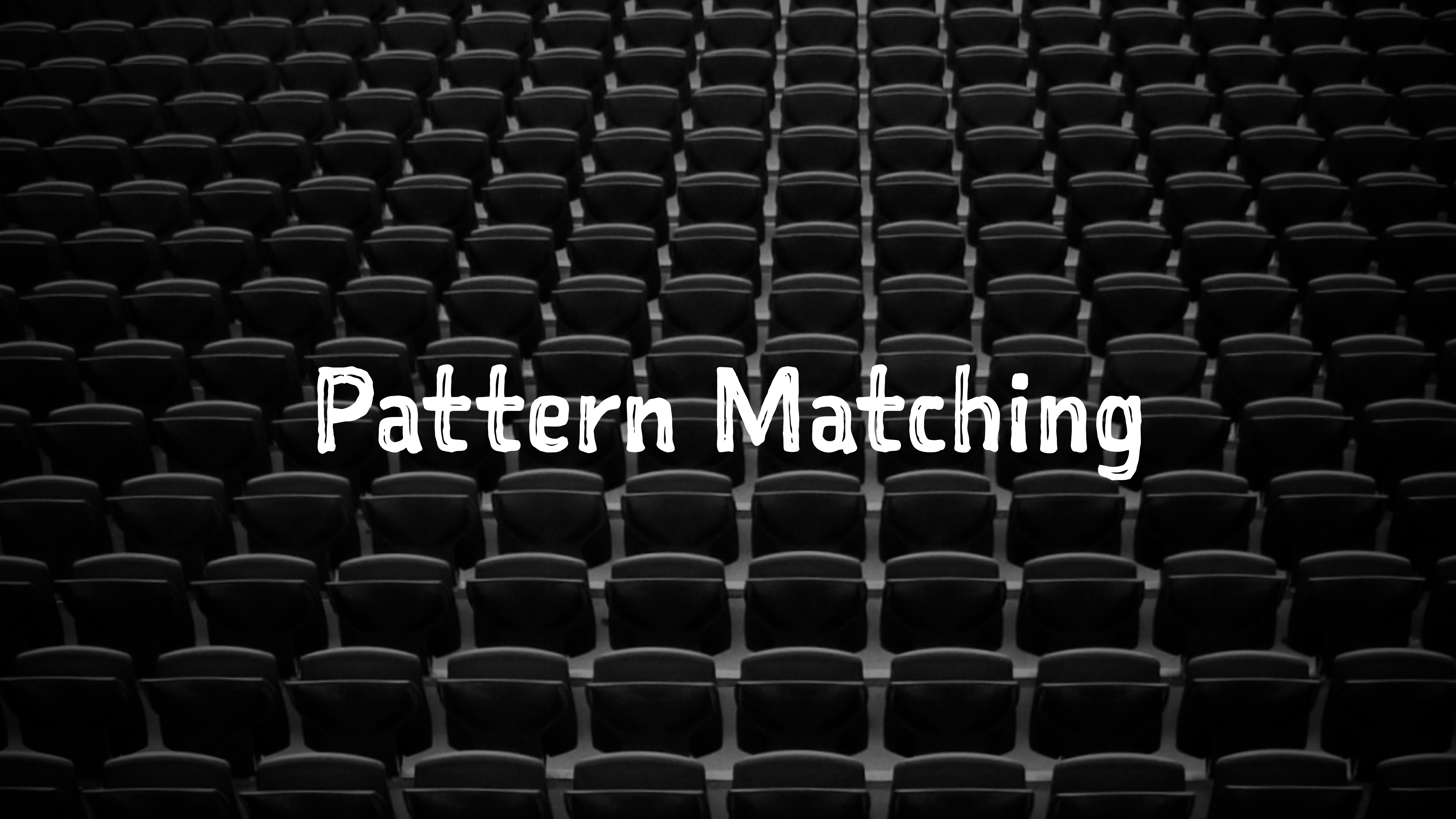
```
Compiling 1 file (.ex)
warning: variable "destination" is unused
  lib/urlz/router.ex:13
```

Pipe Operator

```
|>
```

```elixir
conn
|> put_resp_content_type("text/plain")
|> send_resp(200, "Hello World!")
```

```
send_resp(put_resp_content_type(conn, "text/plain"), 200, "Hello QConNY")
```



Pattern Matching

```elixir
case File.read("qconny.txt") do
  {:ok, result} ->
    result
  {:error, :enoent} ->
    result
  end
end
```

```
def handle_call({:shorten, url}, state) do
  IO.puts("I will shorten a #{url}")
end

def handle_call({:expand, url}, state) do
  IO.puts("I will expand a #{url}")
end
```

Behaviors

Behaviors

```
defmodule Parser do
  @callback parse(String.t) :: any
  @callback extensions() :: [String.t]
end
```

Behaviors

```
defmodule YAMLParser do
  @behaviour Parser

  def parse(str), do: # ... parse YAML
  def extensions, do: ["yml"]
end
```

Protocols

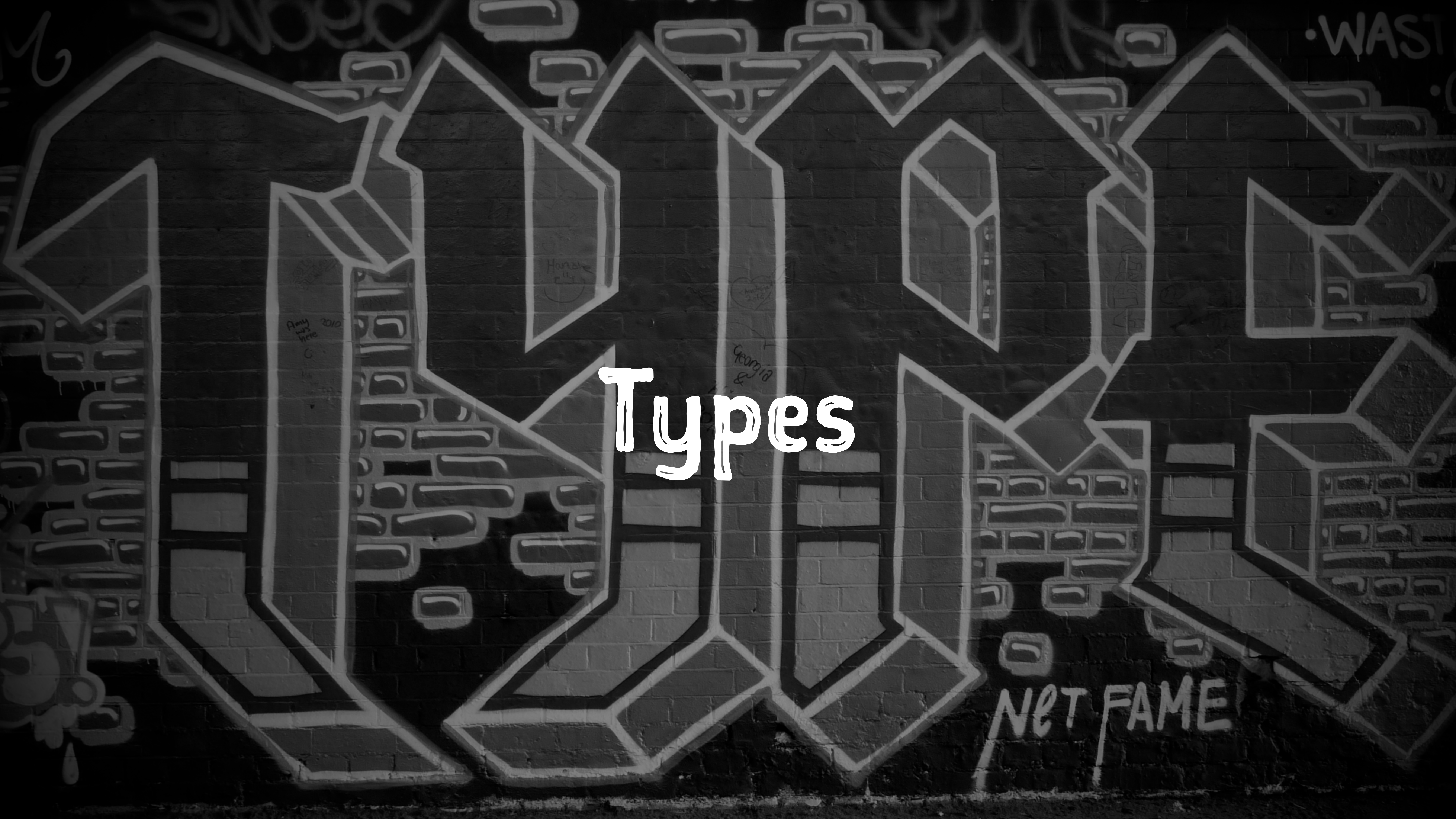
Protocols

```
defimpl String.Chars, for: Urlz.Url do
  def to_string(%Urlz.Url{destination: destination, slug: slug} = url) do
    "slug: #{slug} destination: #{destination}"
  end
end
```

Protocols

```
%Urlz.Url{slug: "1234", destination: "http://google.com"}
|> to_string()

"slug: 1234 destination: http://google.com"
```



Types

```
type :: any() # the top type, the set of all terms
 | none() # the bottom type, contains no terms
 | atom()
 | map() # any map
 | pid()
 | port()
 | reference()
 | struct() # any struct
 | tuple() # tuple of any size
```

Custom Types

```elixir
@type url :: %Urlz.Url{destination: String.t, slug: String.t}
```

Typespecs

```
Enum.reduce(%{}, &(Map.put(&2, &1, hash(data[&1]))))
```

Typespecs

```elixir
@spec add(number, number) :: number
def add(x, x) do
  x + x
end
```

Dialyzer

dialyxir

```
defp deps do
  [{:dialyxir, "~> 0.5", only: [:dev], runtime: false}]
end
```

dialyxir

```
mix dialyzer
```

```
lib/urlz/shortener.ex:50: Invalid type specification for function 'Elixir.Urlz.Shortener':expand/2.
The success typing is (_,binary()) -> #{'__struct__':='Elixir.Urlz.Url', 'slug':=_, 'destination':=_}
```

Property Tests

```
defp deps do
  [{:eqc_ex, "~> 1.4"}]
end
```

```
mix eqc.install --mini
```

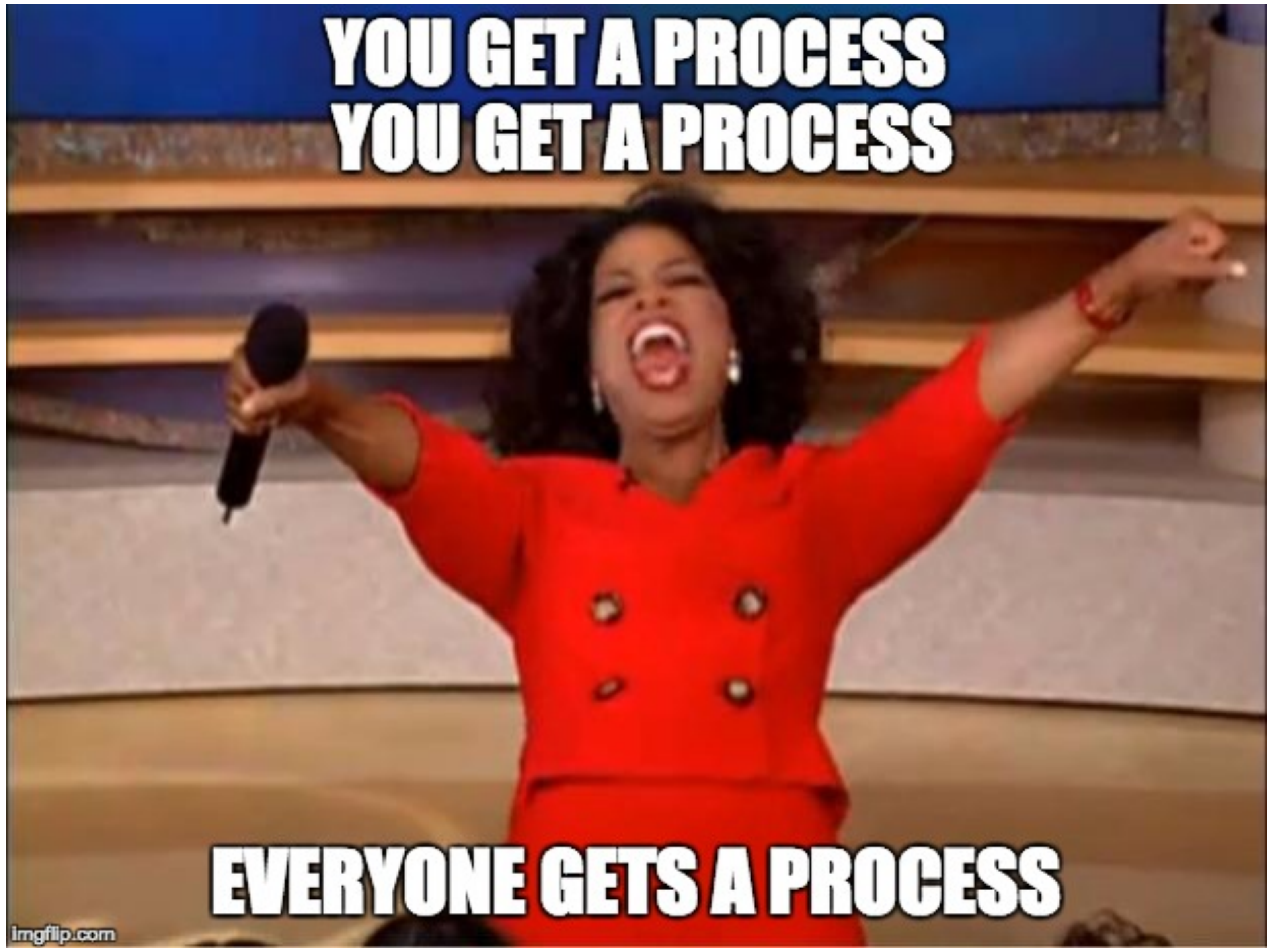
Controlled Randomness

```
@tag numtests: 100000
property "shortens all strings" do
  forall s <- string() do
    string_length =
      s
      |> Urlz.Shortener.ex_shorten
      |> String.length

    ensure string_length > 0
   end
end
```

BEAM

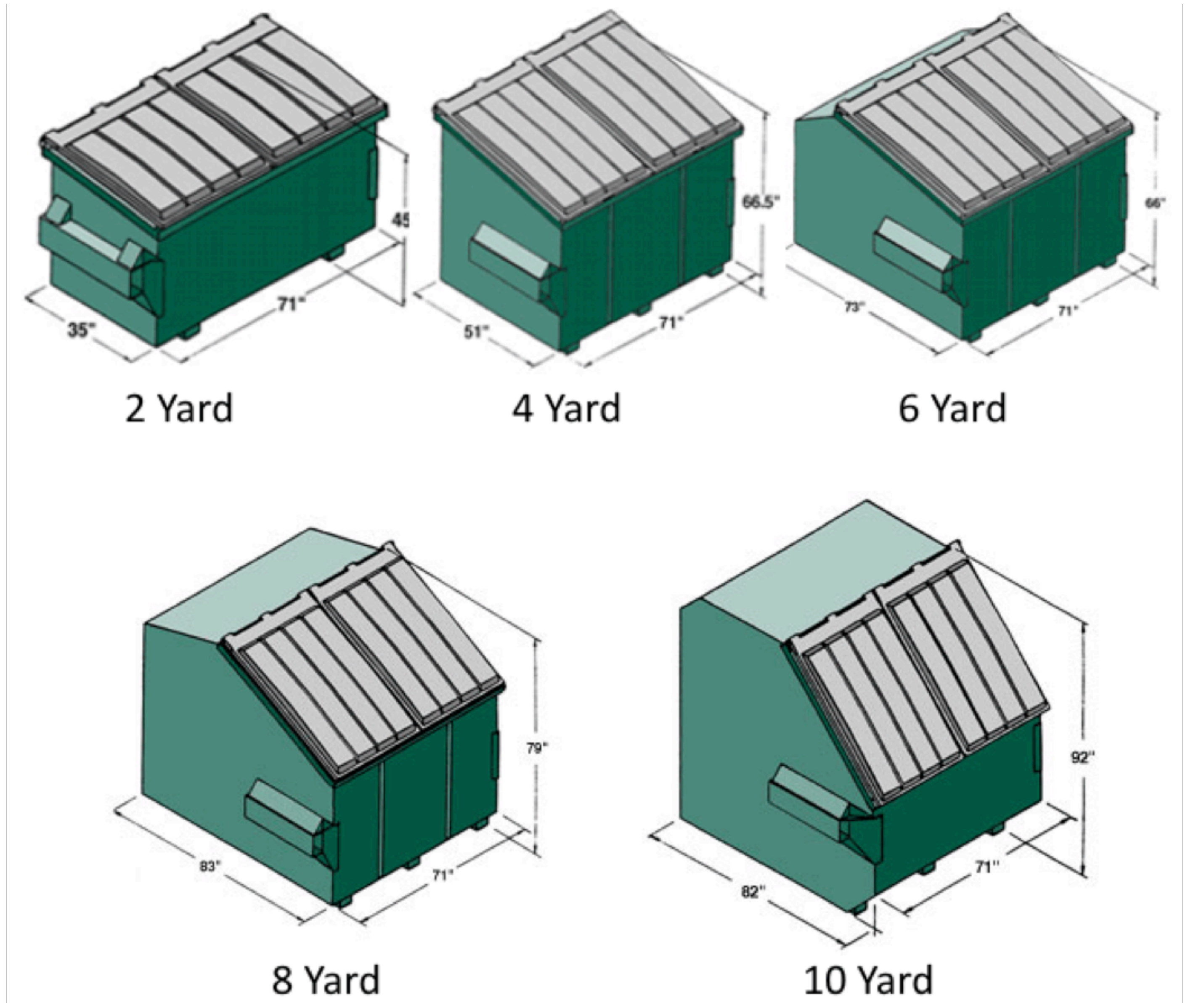
Processes


YOU GET A PROCESS
YOU GET A PROCESS
EVERYONE GETS A PROCESS
imgflip.com

Amtrak TRAIN INFORMATION
ClubAcela Behind Stairway One
Amtrak
Gate 5
Track Departs
5 11:36 AM
No. Name
143 REGIONAL
Status
ALL ABOARD
To
WILMINGTON
BALTIMORE
BWI AIRPORT
NEW CARROLLTON
WASHINGTON

Scheduler

Garbage Collection



Process

```
parent = self()
spawn(fn() -> send(parent, "Hello QconNY") end)
flush()
"Hello QConNY"
:ok
```

Process State

```elixir
defp loop(state) do
  receive do
    {:get, key, caller} ->
      send(caller, Map.get(map, key))
      loop(map)
    {:put, key, value} ->
      loop(Map.put(map, key, value))
  end
end
```

Agents

```
Agent.start_link(fn() -> %{} end)
```

Tasks

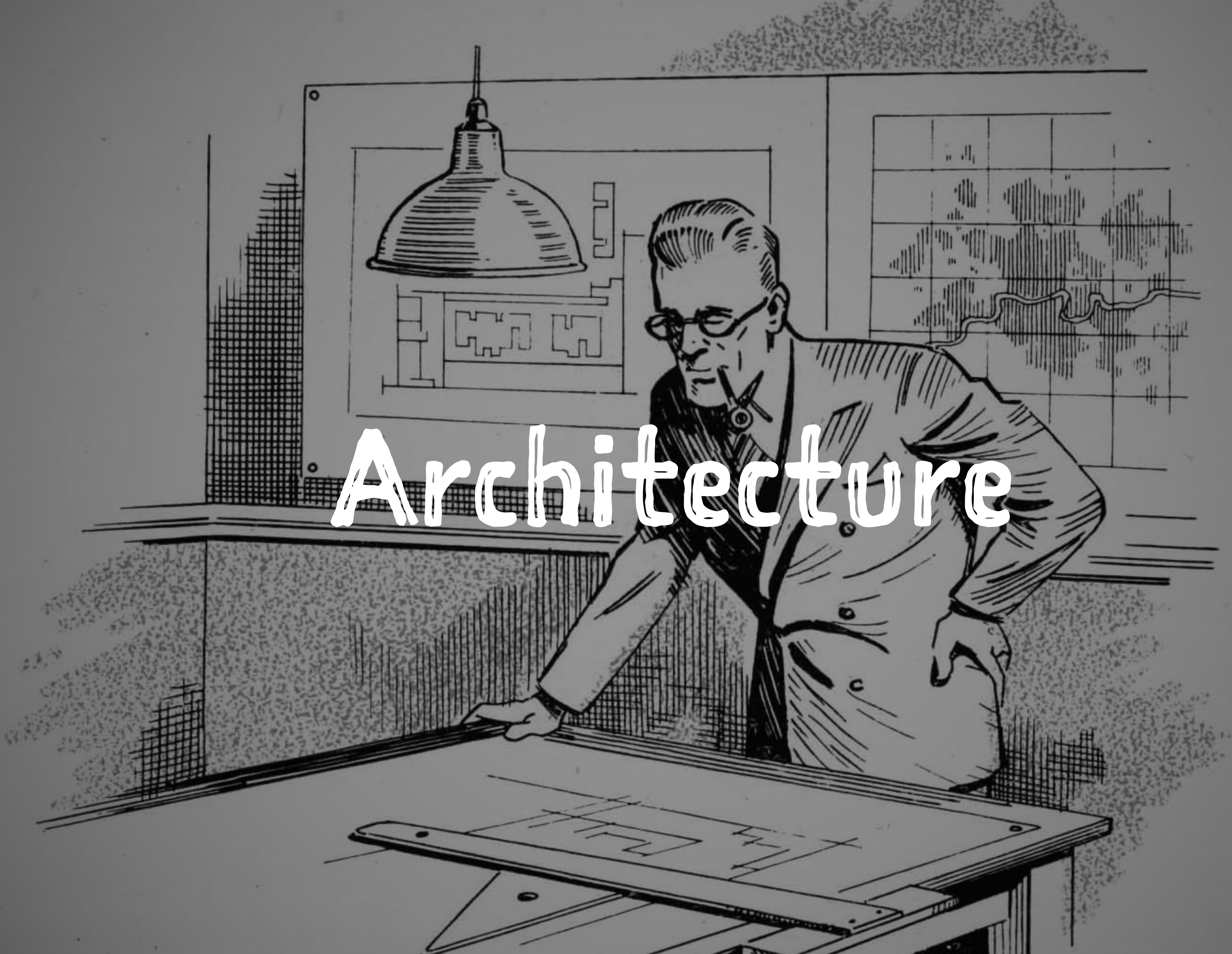
```
task = Task.async(fn -> call_to_api() end)
IO.puts("do some more work")
result = Task.await(task)
```

Agents

```
defmodule Urlz.Shortener do
  @spec start_link() :: {:ok, pid}
  def start_link do
    Agent.start_link(fn -> %{} end)
  end

  @spec shorten(any, String.t) :: String.t
  def shorten(db, url) do
    shortened_url =
      url
      |> :erlang.md5
      |> Base.encode16(case: :lower)
      |> String.slice(0..3)

    Agent.update(db, fn(urls) -> Map.put(urls, shortened_url, url) end)
    shortened_url
  end
 end
```



Architecture

ETS

GenServers

```
defmodule Shortener do
  use GenServer
  def handle_call
  def handle_cast
  def handle_info
  def start_link
  def info
end
```

GenServers

```elixir
# client
@spec expand(pid, String.t) :: url
def expand(pid, shortened_url) do
  GenServer.call(pid, {:expand, shortened_url})
end

@spec shorten(pid, String.t) :: url
def shorten(pid, url) do
  GenServer.call(pid, {:shorten, url})
end
```

GenServers

```
# server

def handle_call({:shorten, url}, from, state) do
  shortened_url = shorten_url(url)
  result = Urlz.Cache.set(shortened_url, url)

  {:reply, result, state}
end

def handle_call({:expand, url}, from, state) do
  result = Urlz.Cache.get(url)
  {:reply, result, state}
end
```

GenServers

```elixir
# business logic

@spec shorten_url(String.t) :: String.t
def shorten_url(url) do
  url
  |> :erlang.md5
  |> Base.encode16(case: :lower)
  |> String.slice(0..3)
end
```

Supervisors

```elixir
def init(:ok) do
  children = [
    worker(Urlz.Cache, [[name: Urlz.Cache]])
  ]

  supervise(children, strategy: :one_for_one)
end
```

Supervisor Trees

Supervisors

```
  def init(:ok) do
    children = [
      :poolboy.child_spec(:shortener_pool, poolboy_config(), []),
      supervisor(Urlz.CacheSupervisor, []),
      Plug.Adapters.Cowboy.child_spec(:http, Urlz.Router, [], [port: 4001, acceptors: 10])
    ]

    supervise(children, strategy: :one_for_one)
  end
end
```

Pools

Pools

```
:poolboy.transaction(:shortener_pool,
  fn(w) -> do_stuff(w, "qcon") end)
```

OTP Application

```
defmodule Urlz do
  use Application

  def start(_type, _args) do
    Urlz.Supervisor.start_link
  end
end
```

```
def application do
  # Specify extra applications you'll use from Erlang/Elixir
  [extra_applications: [:logger],
   mod: {Urlz, []}]
end
```

Operations

Distillery

WHISKY DISTILLERY
№DB3613
POT STILL
PURE MALT WHISKEY
1999
YAMAZAKI

```
defp deps do
  [{:distillery, "~> 1.4"}]
end
```

```
mix release          # Build a release for the current mix application
mix release.clean    # Clean up any release-related files
mix release.init     # initialize a new release configuration
```

Release Contents

```
tree _build/prod/rel/urlz/releases/0.1.0/
_build/prod/rel/urlz/releases/0.1.0/
├── sys.config
├── urlz.boot
├── urlz.rel
├── urlz.script
├── urlz.tar.gz
└── vm.args
```

```
{release,{"urlz","0.1.0"},
         {erts,"8.3"},
         [{kernel,"5.2"},
          {stdlib,"3.3"},
          ...
          {runtime_tools,"1.11.1"}]}
```

Deploy Anywhere

Docker

mix_docker

```
defp deps do
  [{:mix_docker, "~> 0.5.0"}]
end
```

Docker Build

```
mix docker.build
```

Docker Release

```
mix docker.release
```

Sys module

```
:sys.get_status(pid)
```

Sys module

```
{:status, #PID<0.100.0>, {:module, :gen_server},
 [["$initial_call": {:erl_eval, :"-expr/5-fun-3-", 0},
   "$ancestors": [#PID<0.97.0>, #PID<0.73.0>]], :running, #PID<0.97.0>, [],
  [header: 'Status for generic server <0.100.0>',
   data: [{'Status', :running}, {'Parent', #PID<0.97.0>},
    {'Logged events', []}], data: [{'State', %{}}]]]}
```

Sys module

```
:sys.trace(pid, true)
```

Sys module

```
Agent.get(pid, fn(state) -> Map.get(state, "does-not-exit") end)
*DBG* <0.116.0> got call {get,#Fun<erl_eval.6.118419387>} from <0.97.0>
*DBG* <0.116.0> sent nil to <0.97.0>, new state #{<<108,111,99,97,116,105,111,110>>=><<78,101,119,32,89,111,114,107>>}
```

DBG

```
defp deps do
  {:dbg, github: "fishcakez/dbg"}
end
```

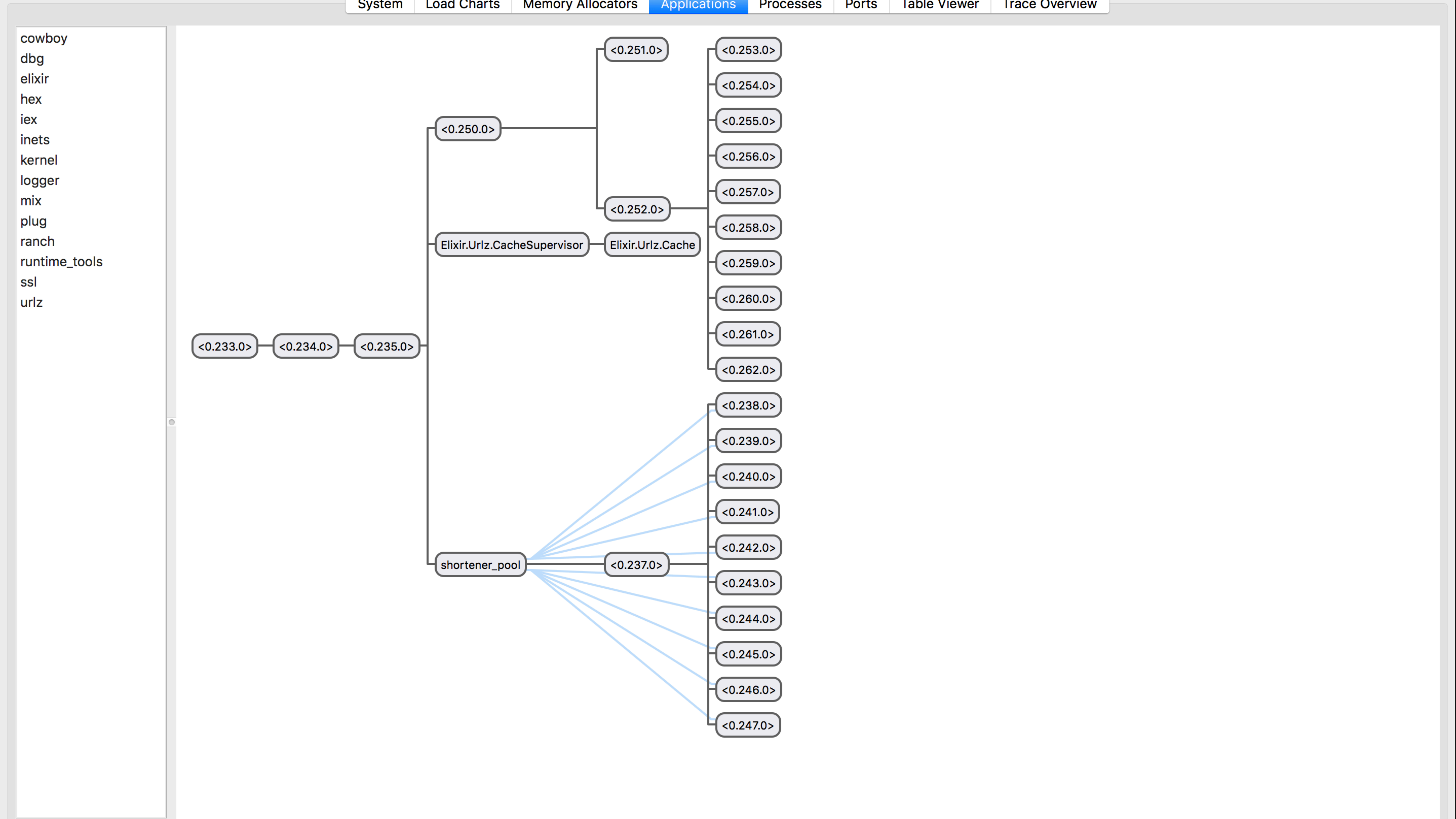
DBG

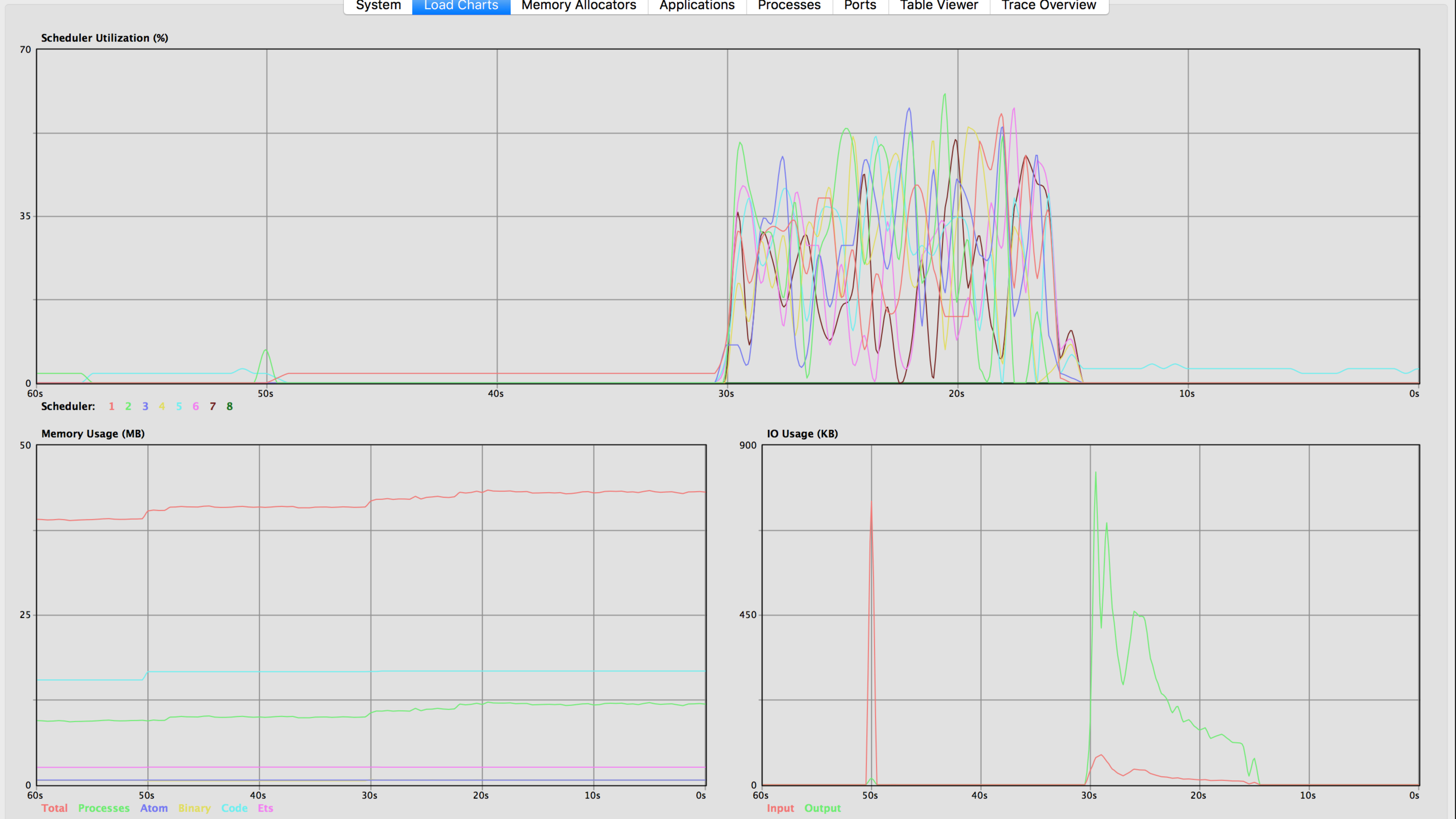
```
Dbg.trace(n, :call)
Dbg.call(&String.slice/2)
```

```
** (Dbg) #PID<0.333.0> calls String.slice/2 with arguments:
    ["69599bcfd1e2792d426f3d2b843a6885", 0..3]
```

Observer

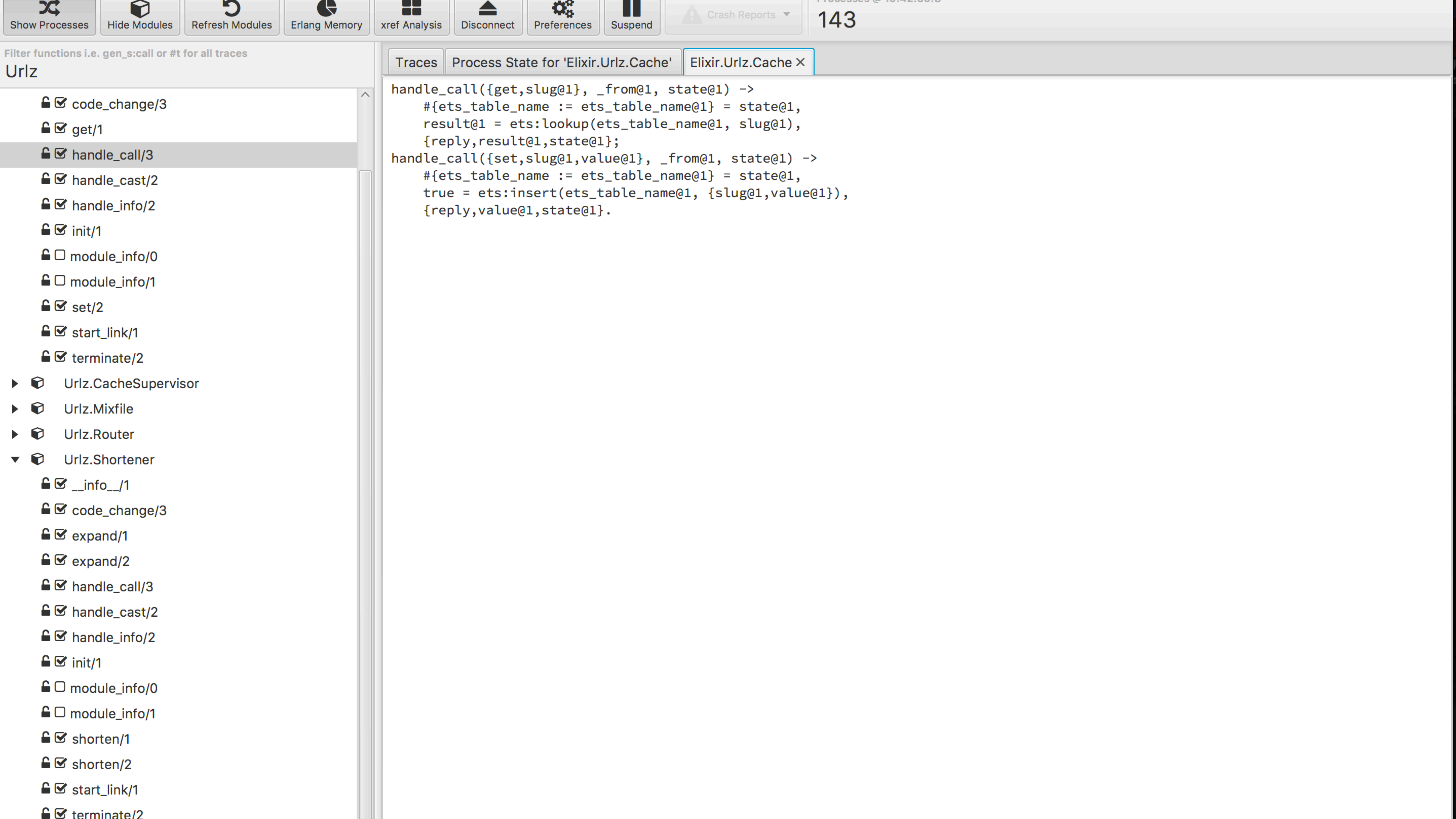
```
:observer.start()
```


System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
cowboy
dbg
elixir
hex
iex
inets
kernel
logger
mix
plug
ranch
runtime_tools
ssl
urlz
<0.233.0>
<0.234.0>
<0.235.0>
<0.250.0>
<0.251.0>
<0.252.0>
<0.253.0>
<0.254.0>
<0.255.0>
<0.256.0>
<0.257.0>
<0.258.0>
<0.259.0>
<0.260.0>
<0.261.0>
<0.262.0>
Elixir.Urlz.CacheSupervisor
Elixir.Urlz.Cache
shortener_pool
<0.237.0>
<0.238.0>
<0.239.0>
<0.240.0>
<0.241.0>
<0.242.0>
<0.243.0>
<0.244.0>
<0.245.0>
<0.246.0>
<0.247.0>



System
Load Charts
Memory Allocators
Applications
Processes
Ports
Table Viewer
Trace Overview
Scheduler Utilization (%)
70
35
0
60s
50s
40s
30s
20s
10s
0s
Scheduler: 1 2 3 4 5 6 7 8
Memory Usage (MB)
50
25
0
60s
50s
40s
30s
20s
10s
0s
Total Processes Atom Binary Code Ets
IO Usage (KB)
900
450
0
60s
50s
40s
30s
20s
10s
0s
Input Output

ERLYBERLY



```
handle_call({get,slug@1}, _from@1, state@1) ->
    #{ets_table_name := ets_table_name@1} = state@1,
    result@1 = ets:lookup(ets_table_name@1, slug@1),
    {reply,result@1,state@1};
handle_call({set,slug@1,value@1}, _from@1, state@1) ->
    #{ets_table_name := ets_table_name@1} = state@1,
    true = ets:insert(ets_table_name@1, {slug@1,value@1}),
    {reply,value@1,state@1}.
```

Remote Systems

- Cookie
- Name
- EPMD
- SSH forwarding

```
ssh -i ~/.ssh/id_rsa -L 4369:127.0.0.1:4369 -L 34579:127.0.0.1:34579 user@server
```

```
iex --name debug@127.0.0.1 --cookie my_cookie --remsh urlz@127.0.0.1
```

Summary

Discussion / Questions?

Reach Out

Twitter: @jearvon

Email: j.dharrie@gmail.com

github repo: https://github.com/iamjarvo/urlz