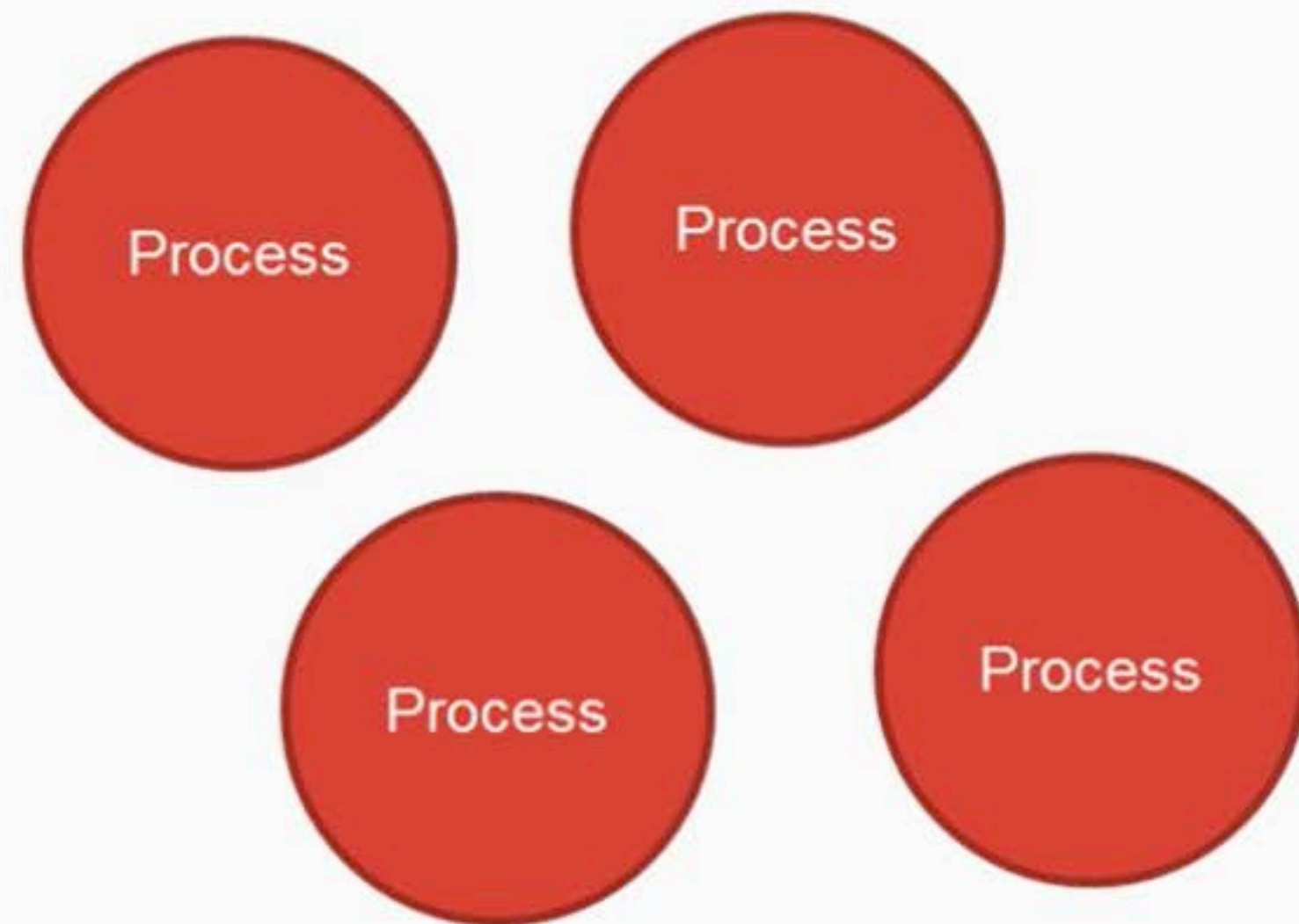


Application

In this Video, we are going to take a look at...

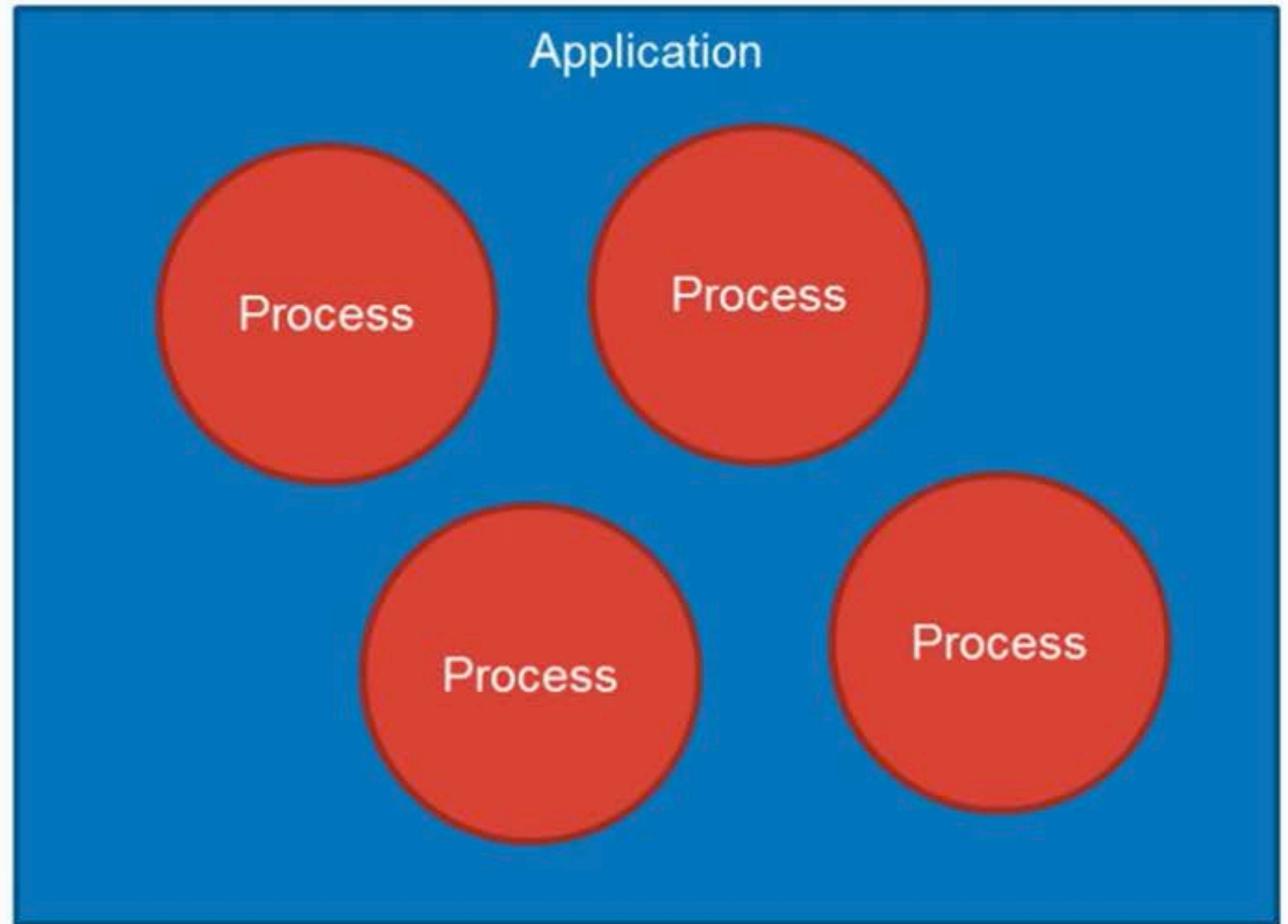
- Application behavior
- Starting applications
- Observing an application

Application



Application

Logical group of processes.




```
defmodule MyApp.Mixfile do
  use Mix.Project

  def project do
    [
      app: :my_app,
      version: "0.1.0",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  def application do
    [
      extra_applications: [:logger],
    ]
  end

  defp deps do
    []
  end
end
```



```
defmodule MyApp.Mixfile do
  use Mix.Project

  def project do
    [
      app: :my_app,
      version: "0.1.0",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  def application do
    [
      extra_applications: [:logger],
    ]
  end

  defp deps do
    []
  end
end
```



```
defmodule MyApp.Mixfile do
  use Mix.Project

  def project do
    [
      app: :my_app,
      version: "0.1.0",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  def application do
    [
      extra_applications: [:logger],
    ]
  end

  defp deps do
    []
  end
end
```

```
> iex -S mix
```



```
iex(1)> Application.started_applications
```



```
defmodule MyApp.Mixfile do
  use Mix.Project

  def project do
    [
      app: :my_app,
      version: "0.1.0",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  def application do
    [
      extra_applications: [:logger],
    ]
  end

  defp deps do
    []
  end
end
```

```
> iex -S mix
```



```
iex(1)> Application.started_applications
```



```
[
  {:my_app, 'my_app', '0.1.0'},
  {:logger, 'logger', '1.5.1'},
  {:mix, 'mix', '1.5.1'},
  {:iex, 'iex', '1.5.1'},
  {:elixir, 'elixir', '1.5.1'},
  {:compiler, 'ERTS CXC 138 10', '7.0'},
  {:stdlib, 'ERTS CXC 138 10', '3.0'},
  {:kernel, 'ERTS CXC 138 10', '5.0'}
]
```


How Do We Get “Stuff” Starting with Our Application?

How Do We Get “Stuff”
Starting with Our Application?
Implementing the Application behavior!

Application — Deck of Cards Revisited



Application — Deck of Cards Revisited

Press ? for neotree help

```
<r/Projects/Section6/card_deck/  
+_build/  
+config/  
+lib/  
+test/  
.gitignore  
README.md  
mix.exs
```

```
defmodule Deck.Application do  
  use Application  
  
  def start(_type, _args) do  
    Deck.start_link()  
  end  
end
```

[4/7] card_deck (D:4 F:3)

1 - 110 application.ex Elixir unix | 5:

Application — Deck of Cards Revisited

```
defmodule CardDeck.Mixfile do
  use Mix.Project

  def project do
    [
      app: :card_deck,
      version: "0.1.0",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about applications.
  def application do
    [
      extra_applications: [:logger],
    ]
  end

  # Run "mix help deps" to learn about dependencies.
  defp deps do
```


Application — Deck of Cards Revisited

```
defmodule CardDeck.Mixfile do
  use Mix.Project

  def project do
    [
      app: :card_deck,
      version: "0.1.0",
      elixir: "~> 1.5",
      start_permanent: Mix.env == :prod,
      deps: deps()
    ]
  end

  # Run "mix help compile.app" to learn about applications.
  def application do
    [
      extra_applications: [:logger],
      mod: {Deck.Application, []}
    ]
  end

  # Run "mix help deps" to learn about dependencies.
  defp deps do
```

1

* 605 mix.exs

Elixir

alchemist Y S P @ K

unix | 18:31

Top

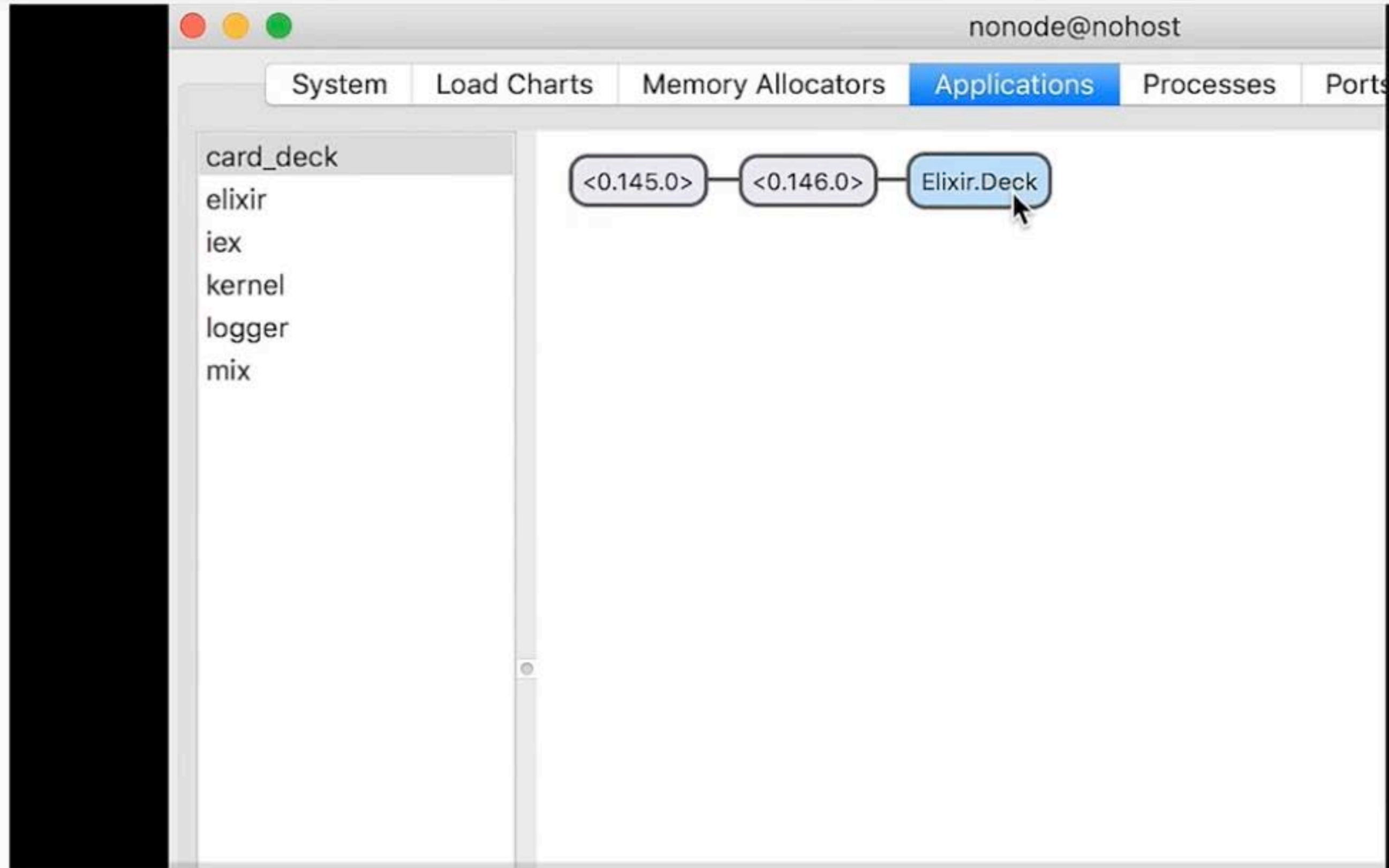
:w

Application — Deck of Cards Revisited

```
→ card_deck iex -S mix
Erlang/OTP 20 [erts-9.0.4] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:10] [hipe] [kernel-poll:false] [dtrace]

Compiling 2 files (.ex)
Generated card_deck app
Interactive Elixir (1.5.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)> Deck.take_card
51
iex(2)> :observer
```

Application — Deck of Cards Revisited



Summary

- Explored introduction to OTP
- Discussed OTP feature set – GenServer
- Explained application