Brief Overview by Francis Joanis

What is Elixir?

- New functional programming language
- Built using Erlang
- Runs on top of Erlang VM
  - Interacts with Erlang
- Feels more like Ruby than Erlang

Why Elixir?

- Syntax can be less intimidating than Erlang
- Leverage all of Erlang's strengths
  - Processes, Pattern Matching, Immutability, ...
- Easier, more flexible meta-programming
  - Build DSLs easily (like Ruby)
- Supports polymorphism
- Variables can be re-bound

Hello World

```
$ cat Hello.exs

IO.puts “Hello World”

$ elixir Hello.exs

Hello World
```

Data Types

- They are pretty much like Erlang's
  - Integer, Float
  - Atom
  - Tuple
  - List
  - Bitstring
  - Binary
  - Record

Hello World, Shell

```
$ iex

Interactive Elixir (0.9.2.dev) - press Ctrl+C
to exit (type h() ENTER for help)

iex(1)> IO.puts “Hello World”

Hello World

:ok

iex(2)>
```

Tools

- Elixir has its own tools similar to Erlang's
  - `iex` - shell (like `erl`)
  - `mix` - build tool (like `rebar`)
  - `ExUnit` - unit test lib (like `eunit`)

Modules

```
defmodule HelloWorld do
  def say_hello() do
    IO.puts “Hello World”
  end
end
```

Pattern Matching

```
def say_hello(“Bill”) do
  IO.puts “Hi Bill!”
end
def say_hello(name) do
  IO.puts “Hello “ <> name
end
```

Pattern Matching

```
def first([head|_]) do
  IO.puts "First: " <> head
end
```

Regular Expressions

```
def regex(name) do
  if Regex.run(%r/jim/i, name) do
    IO.puts "Jim was found"
  else
    IO.puts "Jim not found"
  end
end
```

Protocols

- Protocols are used to implement polymorphism
- Act on native types

Protocols

```
defprotocol K9 do
  @doc "Returns whether it barks or not"
  def barks?(kind)
end

defimpl K9, for: BitString do
  def barks?("Dog") do
    true
  end
  def barks?(_) do
    false
  end
end

IO.puts K9.barks?("Cat") # false
IO.puts K9.barks?("Dog") # true
```

Processes and Messages

- Elixir has the same semantics for spawning processes and passing messages as Erlang
- In fact... it is the same as Erlang
  - Elixir is implemented in Erlang
  - Compiles to Erlang bytecode
  - A lot of code/concepts are 1:1

Interacting with Erlang

```
# Calling Erlang from Elixir
:timer.sleep(1000)


% Calling Elixir form Erlang

'Elixir.IO':puts("Hello").
```

More info

- http://elixir-lang.org