Play along

Only prerequisite is to have Elixir installed

v1.0 or greater is fine

```
$ mix -v
Mix 1.0.5

$ elixir -v
Elixir 1.0.5
```

You are not alone

- → Join us throughout the week on Triangle Devs Slack (https://triangle-devs-slack-inviter.herokuapp.com) in the **#elixir** channel
- → Weekly exercism.io Elixir exercises (http://exercism.io/languages/elixir)
- → Me @rwdaigle

Getting To Know the Elixir Development Environment

What is a language environment?

- → Build tool
- → Dependency resolution
- → Configuration management
- → Testing framework
- → Interactive shell

Mix

- → Project bootstrapper (like `script/rails g`)
- → Build utility/script runner (like Rake)
- → Dependency manager (like Bundler)

Create new project*

```
$ mix help
mix                     # Run the default task (current: mix run)
mix app.start           # Start all registered apps
…

$ mix new triangle
$ cd triangle
$ ls
README.md config    lib       mix.exs   test
```

Test*

```
# test/triangle_test.exs
test "area" do
  assert Triangle.area(3, 5) == 3 * 5 / 2
end

$ mix test

# lib/triangle.ex
def area(base, height), do: base * height / 2
```

IEx

- → Interactive elixir, REPL
- → Loads Elixir environment w/ shell access
- → Dynamically reload code

IEx playground*

```
$ iex
> c "lib/triangle.ex"
> Triangle.area 2, 3
3.0
> r Triangle
> h Enum.<tab>

$ iex -S mix
> Triangle.area 2, 3
```

Debugging

- → Print messages to stdout
- → Attach interactive shell to running process (IEx.pry)

Print to stdout*

```
def area(base, height) do
    IO.puts "base: #{base}, height: #{height}"
    base * height / 2
end
```

Attached shell*

```
# triangle.ex
require IEx
def area(base, height), do: IEx.pry && base * height / 2

$ iex -S mix
> Triangle.area 2, 3
pry(1)> base
2
pry(2)> respawn
```

Dependencies

- → Package manager is called Hex (https://hex.pm)
- → Dependency resolution handled by Mix
- → Project dependencies defined in `mix.exs`

Add dependency*

```
# triangle.ex
require Metrix
def area(base, height) do
  Metrix.measure "triangle.area", fn -> base * height / 2 end
end


# mix.exs
def application do
  [applications: [:logger, :metrix]]
end

defp deps do
  [
    {:metrix, "~> 0.2.0"}
  ]
end


$ mix deps.get
```

Configuration

- → `config/` contains configurations
  - → `Config` sets up app values

Using config values*

```
# config/config.ex
config :triangle, :default_length, 4

# triangle.ex
def equilateral(length), do: {length, length, length}
def equilateral do
  Application.get_env(:triangle, :default_length))
  |> equilateral
end

# triangle_test.exs
test "default equilateral side length of 4" do
  assert Triangle.equilateral == {4, 4, 4}
end
```

Environments

- → `Mix.env` available at runtime
- → Environment specific configs in `config/`

Environment configuration*

```
$ touch config/dev.exs config/test.exs

# config/config.exs
import_config "#{Mix.env}.exs"

# config/dev.exs
use Mix.Config
config :triangle, :default_length, 3

# config/test.exs
use Mix.Config
config :triangle, :default_length, 2
```

Bye

This talk/script can be found at:

https://github.com/rwdaigle/elixir-environment-basics

I can be found

@rwdaigle