Modules

In Elixir we group several functions into modules.

A module is defined using the `defmodule` macro. Use `def` to define functions and `defp` to define private functions. Public functions are globally available and private functions can only be used inside its own module.

```
defmodule Greetings do
  def hello(name) do
    IO.puts concat_greeting("hello ", name)
  end

  def morning(name) do
    IO.puts concat_greeting("Good morning ", name)
  end

  defp concat_greeting(greeting, name) do
    greeting <> name
  end
end
```

To call our hello/1 we would do `Greetings.hello("Uriel")`. But if we tried to use `concat_greeting/2` we would get the error:

```
iex> Greetings.concat_greeting("Oi ", "billy")
** (UndefinedFunctionError) function Greetings.concat_greeting/2 is undefined or private
    Greetings.concat_greeting("Oi ", "billy")
    (elixir) lib/code.ex:376: Code.require_file/2
```

Import

When importing a module we remove the need to use the fully-qualified name (prefix its methods calls with the module name).

```
iex(1)> Greetings.hello("Marcola")
hello Marcola
:ok
iex(2)> import Greetings, only: [hello: 1]
Greetings
iex(3)> hello("Filter")
hello Filter
:ok
```

The :only is optional, but is recommended to avoid importing all the functions into the namespace.

Resources

- Elixir Docs - Modules and functions