Function Clauses

Notes

HTTP Request

Here's the request we pasted in the video:

```
request = """
GET /bears HTTP/1.1
Host: example.com
User-Agent: ExampleBrowser/1.0
Accept: */*

"""
```

There's Another Way!

If you've played around with Elixir before this course, then you might be thinking: "Hey, there's another way to do this!" And you'd be right! Instead of passing three arguments to the `route` function clauses, you can pass just the `conv` map and pattern match on it, like so:

```
def route(%{method: "GET", path: "/wildthings"} = conv) do
  %{ conv | resp_body: "Bears, Lions, Tigers" }
end

def route(%{method: "GET", path: "/bears"} = conv) do
  %{ conv | resp_body: "Teddy, Smokey, Paddington" }
end
```

If however you're new to Elixir, we haven't yet learned how to pattern match maps in this course. Hang with us! We'll explore this technique in depth starting in Module 8.

Parentheses

When calling a function with arguments, parentheses are optional as long as their absence does not introduce ambiguity.

For example, in the video we used parentheses when calling the `route/3` function which takes three arguments:

```
route(conv, conv.method, conv.path)
```

If you prefer, you can remove the parentheses:

```
route conv, conv.method, conv.path
```

There's no ambiguity there.

However, parentheses are required when piping into a function call. For example, you'll get a warning about the following code being ambiguous:

```
[method, path, _] =
  request
  |> String.split "\n"
  |> List.first
  |> String.split " "
```

Using parentheses makes it unambiguous:

```
[method, path, _] =
  request
  |> String.split("\n")
  |> List.first
  |> String.split(" ")
```

Our general rule is to use parentheses when calling functions to avoid any ambiguity. However, sometimes we're inconsistent. For example, when calling `IO.puts` it just feels more natural to omit the parentheses.

Code So Far

The code for this video is in the `function-clauses` directory found within the `video-code` directory of the code bundle.

Go To Next Video