CHAPTER 3

Setting Your Mind

Coming from a background of imperative languages, it is important to meet Erlang/OTP and Elixir with a mind that is not preset.

The following rules are my recommendations for setting your mind before you get into the actual implementation of a project:

1. Don’t try to translate from an OO (object-oriented) model to a functional model and don’t try to express OO constructs in Erlang or Elixir.
2. Think in a declarative way; do not write how it should happen, but what should happen.
3. Assignments are not actual assignments, but comparisons between the left side and the right side of an expression. In Erlang, this process is called pattern matching. In addition, the first assignment in the code is an initialization of a variable similar to *final* in Java or *readonly* in C#.
4. From the last rule follows that variables, despite their name, are immutable once they are initialized.
5. All types are dynamic. This does not mean that there are no built-in types; there are, but you don’t specify the type at compile time.
6. Leave the concepts of classes and interfaces behind. There are similarities and similar constructs, but it confuses more than it helps.
7. When you hear *processes*, forget what you fear about working with threads or operating processes. They are something completely different and there is no shared memory to hurt you.
8. There are no objects in an OO sense. There are processes and we send messages between them. And yes, this does not only sound like the Actor model, it is an implementation of it.
9. The language knows exceptions with the usual try-catch blocks. Now that you know this, you won’t hear it again in this book. I adhere to Erlang’s *Let it fail* premise. It sounds scary, but it works.
10. Libraries and frameworks are important, especially OTP. Erlang and Elixir as languages are only part of the whole.

You will see that all of the new concepts and patterns are not complicated. They are just different to the ones you, as a mostly imperative programmer, have encountered so far.

So keep these few rules in mind, and have more fun and less headaches.

W. Loder, *Erlang and Elixir for Imperative Programmers*, DOI 10.1007/978-1-4842-2394-9_3