Introduction to DSLs in Kotlin

As humans we don't communicate the same way all the time. Sometimes we're formal, sometimes we use slang, and at times all we do is nod or grunt. And we vary between these and other forms throughout the day, depending on the situation, what we're communicating, and with whom. Programming applications is similar—we communicate with the system and fellow programmers who end up maintaining the code.

Sometimes we need the rigor and full power of general purpose programming languages. At other times we're better with highly specialized, small, and effective languages called domain-specific languages or DSLs—for a thorough discussion of DSLs see the book Domain-Specific Languages. DSLs can't be used to program an entire application, but many of them may be used in different parts of a software system.

As programmers we use a number of DSLs each day, sometimes without realizing or thinking about them as DSLs: CSS, Regular Expressions, XML configuration files, Gradle or Rake build files, React JSX, and so on. When designed well, DSLs can make programmers productive, reducing time and effort to implement parts of an application. They may also help reduce errors, while giving programmers greater flexibility.

This chapter is not about using DSLs, but how to design them so the users of your libraries can enjoy the benefits of the DSLs you create. DSLs are easy to use, but the effort to design and implement them is anything but easy, in general. A good language can alleviate some of that pain, and we'll learn how Kotlin helps.

Even when a language is flexible, we must be willing to push the boundaries to breathe fluency and ease into the mini language we're designing. "All is fair in love and war" and when creating DSLs—to achieve a certain syntax we must be willing to experiment, improvise, and adapt to different styles. That's part of what we'll explore here.

We'll start with a quick discussion about types of DSLs and their essential characteristics. Then we'll review the facilities in Kotlin that make it a suitable

language to design DSLs. After that we'll look at a number of small DSLs and explore how to design and implement each one of them. The techniques you learn here will help you to create and maintain your own DSLs and also to design your APIs for more fluent use in Kotlin.