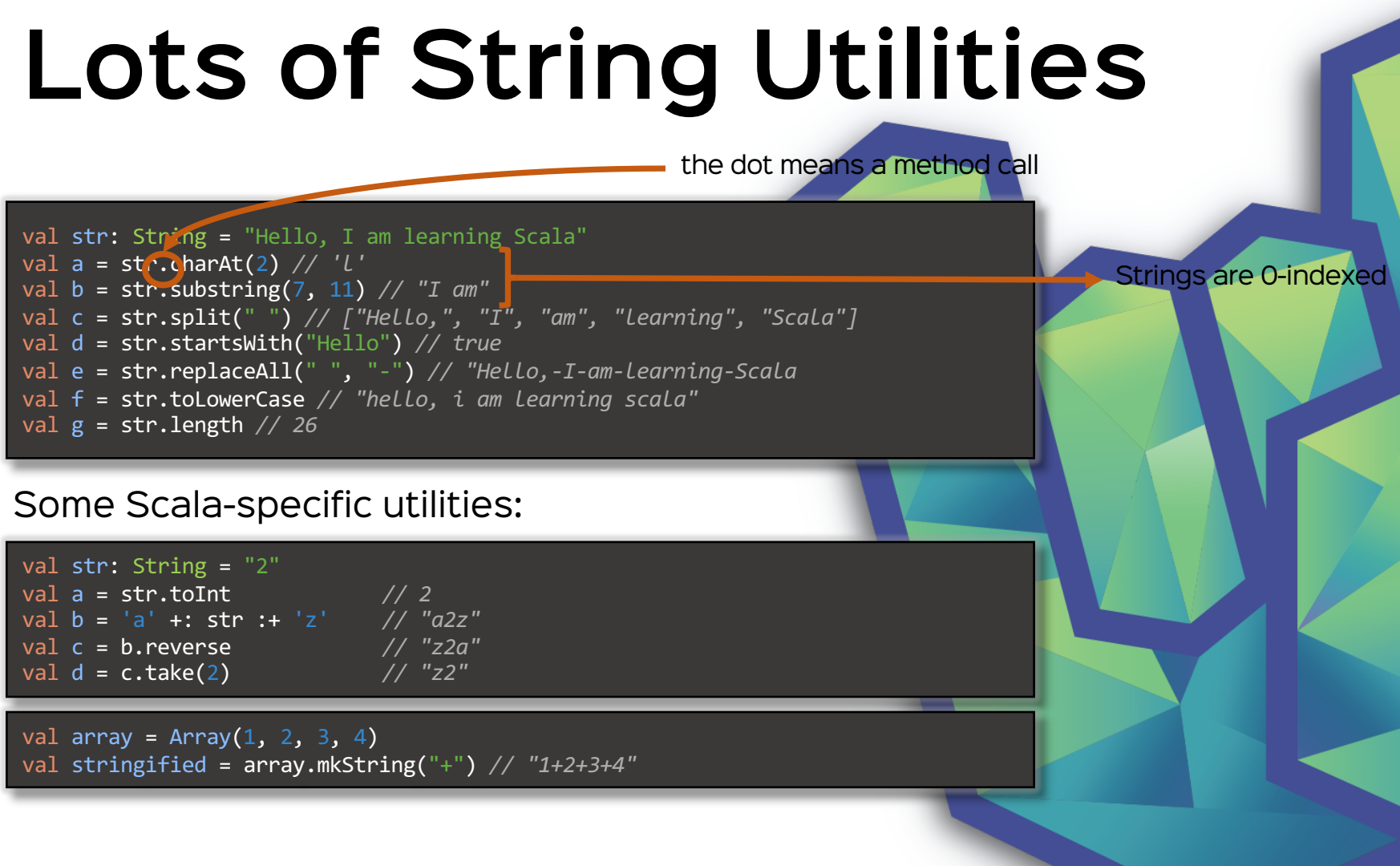
Smart Operations on Strings

Lots of String Utilities




```
val str: String = "Hello, I am learning Scala"
val a = str.charAt(2) // 'l'
val b = str.substring(7, 11) // "I am"
val c = str.split(" ") // ["Hello,", "I", "am", "learning", "Scala"]
val d = str.startsWith("Hello") // true
val e = str.replaceAll(" ", "-") // "Hello,-I-am-learning-Scala
val f = str.toLowerCase // "hello, i am learning scala"
val g = str.length // 26
```

Some Scala-specific utilities:

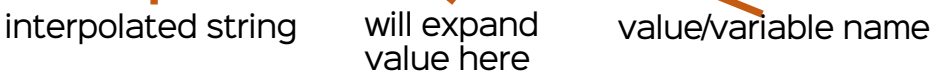
```
val str: String = "2"
val a = str.toInt            // 2
val b = 'a' +: str :+ 'z'    // "a2z"
val c = b.reverse            // "z2a"
val d = c.take(2)            // "z2"
```

```
val array = Array(1, 2, 3, 4)
val stringified = array.mkString("+") // "1+2+3+4"
```

S-String Interpolators

Expands variables inside strings:

```
val greetings = "hello"
println(s"Say $greetings, world!") // "Say hello, world!"
```



Can also evaluate results:

```
val x = 2
val y = 3
println(s"$x + $y = ${x + y}") // "2 + 3 = 5"
```

Scala expression inside

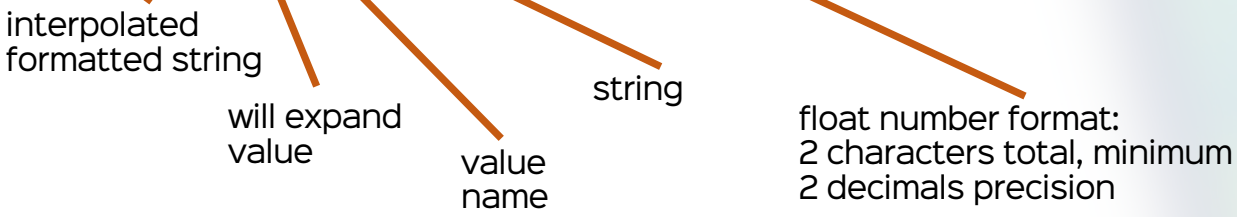
F-String Interpolators

For formatted strings, similar to *printf*:

```
val speed = 1.2f
val name = "Daniel"
println(f"$name%s can eat $speed%2.2f burgers per minute")
```

```
Daniel can eat 1.20 burgers per minute
```

insane!



Can check for type correctness:

```
val x = 1.1f

val str = f"$x%3d"
```

value is Float

format requires Int

Compile error!

Raw-String Interpolators

Same as the S interpolator, prints characters literally:

```
println(raw"Un-escaped: this is a \n newline")
```

```
Un-escaped: this is a \n newline
```

But escaped characters inside variables are kept:

```
val anEscapedString = "this is a \n newline"
println(raw"Un-escaped: $anEscapedString")
```

```
Un-escaped: this is a
 newline
```

Scala rocks