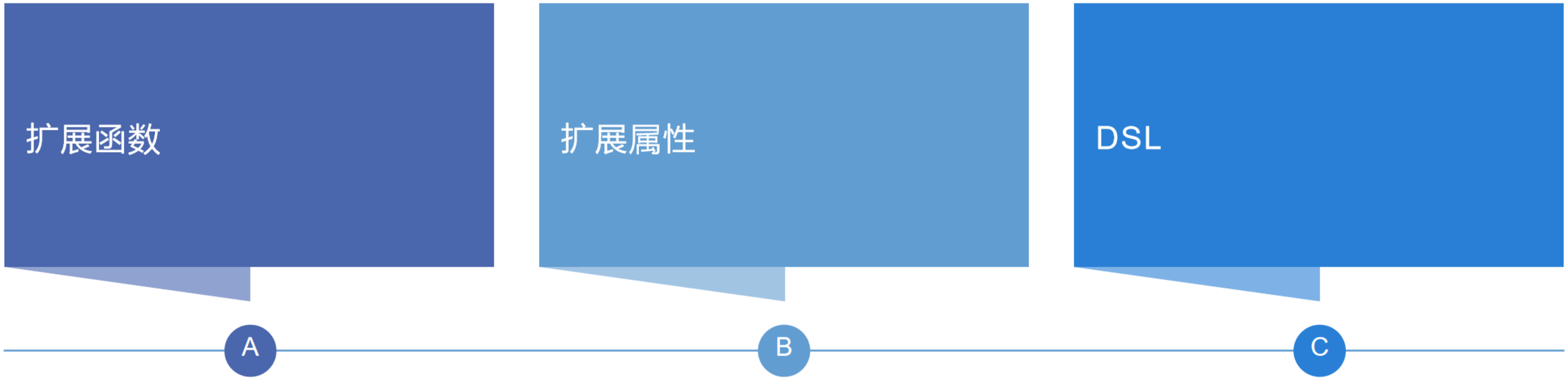
彻底掌握Kotlin



讲师简介

宁传奇

Jason

曾任华为高级工程师、架构师，全栈开发爱好者。

技术这条路，不和你一比高下，只伴你一同成长。

扩展函数

定义扩展函数

- 扩展可以在不直接修改类定义的情况下增加类功能，扩展可以用于自定义类，也可以用于比如List、String，以及Kotlin标准库里的其他类。和继承相似，扩展也能共享类行为，在你无法接触某个类定义，或者某个类没有使用open修饰符，导致你无法继承它时，扩展就是增加类功能的最好选择。

定义扩展函数

- 定义扩展函数和定义一般函数差不多，但有一点大不一样，除了函数定义，你还需要指定接受功能扩展的接收者类型。

```
1   //给字符串追加若干个感叹号
2   fun String.addExt(amount: Int = 1) = this + "!".repeat(amount)
3
4   //在超类上定义扩展函数，Any的所有子类都能使用该函数了
5   fun Any.easyPrint() = println(this)
6
7   fun main() {
8       println("abc".addExt( amount: 10))
9       "test".easyPrint()
10      15.easyPrint()
11  }
```

泛型扩展函数

- 如果想在调用addExt扩展函数之前和之后分别打印字符串怎么办？

```
1   fun String.addExt(amount: Int = 1) = this + "!".repeat(amount)
2
3   fun Any.easyPrint(): Any {
4       println(this)
5       return this
6   }
7
8   fun main() {
9       "abc".easyPrint().addExt(10).easyPrint()
10  }
```

泛型扩展函数

- 新的泛型扩展函数不仅可以支持任何类型的接收者，还保留了接收者的类型信息，

  使用泛型类型后，扩展函数能够支持更多类型的接收者，适用范围更广了。

```
1   fun String.addExt(amount: Int = 1) = this + "!".repeat(amount)
2
3   fun <T> T.easyPrint(): T {
4       println(this)
5       return this
6   }
7
8   fun main() {
9       "abc".easyPrint().addExt( amount: 10).easyPrint()
10  }
```

泛型扩展函数

- 泛型扩展函数在Kotlin标准库里随处可见，例如let函数，let函数被定义成了泛型扩展函数，所以能支持任何类型，它接收一个lambda表达式，这个lambda表达式接收者T作为值参，返回的R-lambda表达式返回的任何新类型。

```
1    public inline fun <T, R> T.let(block: (T) -> R): R {
2        return block(this)
3    }
```

扩展属性

扩展属性

- 除了给类添加功能扩展函数外，你还可以给类定义扩展属性，给String类添加一个扩展，这个扩展属性可以统计字符串里有多少个元音字母。

```
1   val String.numVowels
2       get() = count{"aeiouy".contains(it)}
3
4   fun <T> T.easyPrint(): T {
5       println(this)
6       return this
7   }
8
9   fun main() {
10      "The people's Republic of China".numVowels.easyPrint()
11  }
```

可空类扩展

- 你也可以定义扩展函数用于可空类型，在可空类型上定义扩展函数，你就可以直接在扩展函数体内解决可能出现的空值问题。

```
1  infix fun String?.printWithDefault(default: String) = print(this ?: default)
2
3  fun main() {
4      val nullableString: String? = null
5      nullableString printWithDefault "abc"
6  }
```

infix关键字

- infix关键字适用于有单个参数的扩展和类函数，可以让你以更简洁的语法调用函数，

  如果一个函数定义使用了infix关键字，那么调用它时，接收者和函数之间的点操作以及参数的一对括号都可以不要。

定义扩展文件

- 扩展函数需要在多个文件里面使用，可以将它定义在单独的文件，然后import。

```
1  package com.jason.kotlin.extension
2
3  fun <T> Iterable<T>.randomTake(): T = this.shuffled().first()
```

```
1  import com.jason.kotlin.extension.randomTake
2
3  fun main() {
4      val list: List<String> = listOf("Jason", "Jack", "Tom")
5      val set: List<String> = listOf("love", "hate", "like")
6      list.randomTake()
7  }
```

重命名扩展

- 有时候，你想使用一个扩展或一个类，但它的名字不和你的意。

```
1   import com.jason.kotlin.extension.randomTake as randomizer
2
3   fun main() {
4       val list: List<String> = listOf("Jason", "Jack", "Tom")
5       val set: List<String> = listOf("love", "hate", "like")
6       list.randomizer()
7   }
```

Kotlin标准库中的扩展

- Kotlin标准库提供的很多功能都是通过扩展函数和扩展属性来实现的，包含类扩展的标准库文件通常都是以类名加s后缀来命名的，例如Sequences.kt，Ranges.kt，Maps.kt。

DSL

带接收者的函数字面量

- apply函数是如何做到支持接收者对象的隐式调用的。

```
10    public inline fun <T> T.apply(block: T.() -> Unit): T {
11        block()
12        return this
13    }
```

DSL

- 使用这样的编程范式，就可以写出业界知名的“领域特定语言”（DSL），一种API编程范式，暴露接收者的函数和特性，以便于使用你定义的lambda表达式来读取和配置它们。