# L3-函数与lambda表达式

### 3.5.1函数

默认参数这里要介绍和无默认参数方法的区别。如果在默认参数之后的最后一个参数是 [lambda 表达式](https://www.kotlincn.net/docs/reference/lambdas.html" \l "lambda-%E8%A1%A8%E8%BE%BE%E5%BC%8F%E8%AF%AD%E6%B3%95)，那么它既可以作为具名参数在括号内传入，也可以在[括号外](https://www.kotlincn.net/docs/reference/lambdas.html" \l "passing-a-lambda-to-the-last-parameter)传入,括号外传入可以省略参数的括号！

介绍Unit概念。

介绍varargs 的概念与原理

中缀表示法简单介绍即可 注意优先级

对于函数类型我们应该心中有数：

按作用域划分：局部函数 成员函数 顶层函数

函数分类

1. 单表达式函数
2. 范型函数
3. 内联函数
4. 扩展函数
5. 尾递归函数

### 3.5.2 高阶函数与Lambda表达式

介绍高阶函数的概念，主要是阐述高阶函数和普通函数的区别。

高阶函数是将函数用作参数或返回值的函数。

在介绍此部分之前先简述Lambda语法是怎么回事。另外先交代下方法引用的概念。

使用fold对集合进行自定义计算，以此为高阶函数概念的切入点。

下面对一下代码进行解析

fun <T, R> Collection<T>.fold(initial: R, combine: (acc: R, nextElement: T) -> R): R {

var accumulator: R = initial

for (element: T in this) {

accumulator = combine(accumulator, element)

}

return accumulator

}

1. 这是一个扩展函数，此扩展函数涉及两个范型,返回类型是范型R的类型。
2. 范型T是集合元素的参数化类型，R是输入的操作类型
3. 扩展的方法名为fold,参数有两个initial是输入，combine是一个函数参数
4. 这个函数参数参数的 模版为:(acc: R, nextElement: T) -> R

可以理解为，对此函数参数的规定，不按规定的传不进来

规定如下:此函数需要有两个参数一个类型是R,一个类型是T然后需要返回值，返回值是R

5.方法体内的代码就是正常的操作了，即先做一个赋值，然后遍历集合当前集合this。

然后进行操作，即调用我们传入的函数参数。遍历完成后返回结果。

另外此部分请尽量展示用例，并做出解释。

如：

如需将函数类型指定为[可空](https://www.kotlincn.net/docs/reference/null-safety.html" \l "%E5%8F%AF%E7%A9%BA%E7%B1%BB%E5%9E%8B%E4%B8%8E%E9%9D%9E%E7%A9%BA%E7%B1%BB%E5%9E%8B)，请使用圆括号：((Int, Int) -> Int)?。

函数类型可以使用圆括号进行接合：(Int) -> ((Int) -> Unit)

箭头表示法是右结合的，(Int) -> (Int) -> Unit 与前述示例等价，但不等于 ((Int) -> (Int)) -> Unit。

函数类型实例化

带与不带接收者的函数类型非字面值可以互换，其中接收者可以替代第一个参数，反之亦然。例如，(A, B) -> C 类型的值可以传给或赋值给期待 A.(B) -> C 的地方。

class HTML {

fun body() { …… }

}

fun html(init: HTML.() -> Unit): HTML {

val html = HTML() // 创建接收者对象

html.init() // 将该接收者对象传给该 lambda

return html

}

html { // 带接收者的 lambda 由此开始

body() // 调用该接收者对象的一个方法

}

### 3.5.3 内联函数

首先要阐述，函数也是对象，创建对象会产生性能消耗。所以我们使用内联函数的方式，直接通过编译将代码拿过来，这样就不创建对象了，降低了性能消耗。

可以内联的 lambda 表达式只能在内联函数内部调用或者作为可内联的参数传递， 但是 noinline 的可以以任何我们喜欢的方式操作：存储在字段中、传送它等等。

非局部返回，成员函数结合lambda或局部函数进行介绍。

具体化的类型参数 reified ，让我们的范型操作更优雅。