第 3 章

函　　数

Dart 是面向对象的语言，即使函数也是对象，对应的类为 Function。函数也是对象意味着可以将函数赋值给变量，或者将函数作为参数传递给另一个函数。函数有时又被称为方法，它们是等同的，只是叫法上有差异。

声明函数的格式如下：

```
[返回值类型] 函数名([参数列表]){
  //函数体
  return expr;
}
```

每个函数都必须有返回值，如果不提供返回值类型，则默认为 void。函数名是标识符，应当遵循标识符命名规范。参数列表中可能包含零个到多个参数，这取决于实际情况。函数体由变量声明、语句组成。当函数存在返回值时，函数体应当提供 return 语句，例如：return expr;。如果函数返回值的类型为 void，则无须提供 return 语句，等价于 return null;。

定义函数的示例代码如下：

```
bool isEven(int x){
  return x % 2 == 0 ? true : false;
}
```

该函数返回值的类型为 bool，接收一个 int 类型的参数，函数体由一个 return 语句组成，返回语句的表达式是一个条件表达式，其返回值是布尔值。对于只有一个表达式的函数可以采用简写形式，示例代码如下：

```
bool isEven(int x) => x % 2 == 0 ? true : false;
```

=> expr;是{return expr;}的简写形式，常称为胖箭头语法。胖箭头(=>)与分号(;)之间只能是表达式而不能是语句，例如不能是 if 语句，但可以是条件表达式(expr1 ? expr2 : expr3)。

创建命令行应用程序，将项目命名为 chapter3。本章所有文件都在该项目的 bin 文件夹下创建与运行。

函数包含两种参数形式：必选参数和可选参数。参数列表先列出必选参数，再列出可选参数。

18min

3.1 可选参数

可选参数分为命名参数和位置参数。在参数列表中只能选择其中一种作为可选参数，不可同时出现。

3.1.1 命名参数

定义函数时使用花括号({})包裹可选参数列表，使用{arg1,arg2,…}的形式来定义命名参数，示例代码如下：

```
void  findAll({int currentPage, int pageSize}){…}
```

调用函数时，采用 arg1:value1,arg2:value2 的形式来传递参数。即先提供参数名，接着是冒号(:)，冒号后边接着参数值：

```
findAll(currentPage: 1, pageSize: 10);
```

示例代码如下：

```
//chapter3/bin/example_01.dart
main(){
  //以下定义了一个包含必选参数和可选命名参数的函数
  void message(String from, String content, {DateTime time, String device})
{
    //调用该函数必须提供参数 from 和 content
    //因此无须做任何判断即可直接打印这两个参数
    print('来自：$from,正文：$content');
    //如果参数 time 不为空,则打印 time
    if (time != null) {
      print('时间：$time');
    }
    //如果参数 device 不为空,则打印 device
    if (device != null) {
      print('发送设备：$device');
    }
  }
  //调用 message 函数且提供两个必选参数
  message('jobs', 'hello');
  //调用 message 函数且提供可选参数 time
```

```
  message('jobs', 'hello', time: DateTime.now());
  //调用 message 函数且提供可选参数 time 和 device
  message('jobs', 'hello', time: DateTime.now(), device: 'phone');
}
```

3.1.2 位置参数

使用方括号([])包裹可选参数列表来定义位置参数：

```
void  message(String from,String content,[DateTime time,String device]){...}
```

在调用包含可选位置参数的函数时应当按照位置参数列表的顺序依次为参数赋值。示例代码如下：

```
//chapter3/bin/example_02.dart
void main(){
  //定义带可选位置参数 time 和 device 的函数
  void message(String from,String content,[DateTime time,String device]){
    print('来自：$from,正文：$content');
    if(time != null){
      print('时间：$time');
    }
    if(device != null){
      print('发送设备：$device ');
    }
  }
  //调用函数时不指定可选参数
  message('jobs','hello');
  //调用函数时指定可选位置参数 time
  message('jobs','hello',DateTime.now());
  //调用函数时指定可选位置参数 time 和 device
  message('jobs','hello',DateTime.now(),'phone');
}
```

3.1.3 默认参数值

在声明可选参数时可以使用等号运算符为可选参数指定默认值，默认值必须是编译时常量，没有指定默认值的参数将被赋值为 null。

为命名参数提供默认值的示例代码如下：

```
//chapter3/bin/example_03.dart
main(){
  //为可选命名参数 device 提供默认值
```

```
  void message(String from,String content,{DateTime time,String device = 'phone'}){
    print('来自:$from,正文:$content');
    if(time != null){
      print('时间:$time');
    }
    //参数 device 始终会被打印
    if(device != null){
      print('发送设备:$device');
    }
  }
  //调用函数时不指定可选参数
  message('jobs','hello');
  //调用函数且指定可选命名参数 time
  message('jobs','hello',time: DateTime.now());
  //调用函数且指定可选命名参数 time 和 device
  message('jobs','hello',time: DateTime.now(),device: 'pc');
}
```

为位置参数提供默认值的示例代码如下：

```
//chapter3/bin/example_04.dart
main(){
  //为可选位置参数 device 提供默认值
  void message(String from,String content,[DateTime time,String device = 'phone']){
    print('来自:$from,正文:$content');
    if(time != null){
      print('时间:$time');
    }
    //参数 device 始终会被打印
    if(device != null){
      print('发送设备:$device');
    }
  }
  //调用函数且不指定可选参数
  message('jobs','hello');
  //调用函数且指定可选命名参数 time
  message('jobs','hello',DateTime.now());
  //调用函数且指定可选命名参数 time 和 device
  message('jobs','hello',DateTime.now(),'pc');
}
```

3.2 main 函数

3min

每个 Dart 程序都必须有一个 main 函数作为入口，main 函数的返回值为 void，它有一个 List<String>类型的可选参数。

使用命令行访问带参数的 main 函数的示例代码如下：

```
//chapter3/bin/example_05.dart
//带有参数列表的 main 函数
void main(List < String > args){
  //打印所有参数值
  print(args);
  var len = args.length;
  //打印参数数量
  print('参数数量: $ len');
}
```

保存上述代码，文件名为 example_05.dart。打开编辑器的命令行工具 Terminal，运行命令并携带参数：

```
dart bin/bin/example_05.dart first 2 dog
```

运行结果如下：

```
[first, 2, dog]
参数数量:3
```

也可以省略参数：

```
void main(){
//函数体
}
```

4min

3.3 函数对象

可以将函数作为参数传递给另一个函数，下例中将函数 printElement 传递给 List 类型的对象 list 的 forEach 函数。示例代码如下：

```
//chapter3/bin/example_06.dart
void main(){
  //定义一个带有 int 类型参数的函数
  void printElement(int element) {
    //打印传入的参数值
    print(element);
  }
  //定义一个 List 类型的对象并赋初值
  var list = [1, 2, 3];
```

```
  //将 printElement 函数作为参数传递给 List 对象的 forEach 函数
  //forEach 函数会按迭代顺序将函数 printElement 应用于 List 集合的每个元素
  list.forEach(printElement);
}
```

也可以将函数赋值给一个变量。示例代码如下：

```
//chapter3/bin/example_07.dart
void main() {
  void printElement(int element) {
    print(element);
  }
  //将函数 printElement 赋值给变量 show
  var show = printElement;
  //像执行函数一样使用变量 show
  show(1);
}
```

5min

3.4 匿名函数

大多数方法都是有名字的，例如 main 或 printElement。也可以创建一个没有名字的函数，称为匿名函数。匿名函数常以回调函数或另一个函数的参数的形式出现。

匿名函数看起来与命名函数类似，在括号之间可以定义参数，参数之间用逗号分隔，后面大括号中的内容则为函数体，也可以有返回值。

```
([类型][参数,…]){
//函数体;
}
```

匿名函数常用于集合迭代中，下面的代码定义了只有一个参数 item 且没有参数类型的匿名函数。List 中的每个元素都会调用这个函数，函数体打印元素在集合中的下标和值。示例代码如下：

```
//chapter3/bin/example_08.dart
void main(){
  var list = ['apples', 'bananas', 'oranges'];
  //向 forEach 函数提供匿名函数
  list.forEach((item){
    print('${list.indexOf(item)}: $item');
  });
}
```

如果函数体内只有一行语句，则可以使用胖箭头语法：

```
list.forEach((item) => print('${list.indexOf(item)}: $item'));
```

匿名函数也可以作为另一个函数的返回对象，返回语句 return（num r）=> 2 * pi * r; 中的返回值就是匿名函数。示例代码如下：

```
//chapter3/bin/example_09.dart
void main(){
  //定义一个返回值类型为 Function 的函数
  Function perimeter(){
    var pi = 3.14;
    //return 关键字后面跟着的是一个匿名函数
    return (num r) => 2 * pi * r;
  }
  //将函数赋值给变量
  var per = perimeter();
  var r = 9;
  var l = per(r);
  print('圆的周长：$l');
}
```

7min

3.5 语法作用域

Dart 是有词法作用域的语言，变量的作用域在写代码的时候就确定了。大括号内定义的变量只能在大括号内被访问，在大括号内可以访问大括号外的变量，与 Java 类似。嵌套函数中变量在多个作用域中使用的示例代码如下：

```
//chapter3/bin/example_10.dart
//定义一个顶层变量
String topLevel = 'top variable';
void main(){
  //在 main 函数中定义变量
  var insideMain = 'insideMain variable';
  void myFunction(){
    //在 myFunction 函数中定义变量
    var insideFunction = 'insideFunction variable';
    void nestedFunction(){
      //在嵌套函数中定义变量
      var insideNestedFunction = 'insideNestedFunction variable';
      print('$topLevel');
      print('$insideMain');
```

```
        print('$insideFunction');
        print('$insideNestedFunction');
      }
    }
}
```

注意：nestedFunction()函数可以访问包括顶层变量在内的所有变量。

3.6 语法闭包

4min

语法闭包即一个函数对象，即使函数对象的调用在它原始作用域之外，依然能够访问在它词法作用域内的变量。

函数可以封闭定义到它作用域内的变量。下面的示例中，函数 makeAdder()捕获了变量 addBy。无论函数在什么时候返回，它都可以使用捕获的 addBy 变量。示例代码如下：

```
//chapter3/bin/example_11.dart
void main(){
  //返回一个函数,该函数的参数将与 addBy 相加
  Function makeAdder(num addBy){
    return (num i) => addBy + i;
  }
  //生成加 2 的函数
  var add2 = makeAdder(2);
  //生成加 4 的函数
  var add4 = makeAdder(4);
  print(add2(3));
  print(add4(3));
}
```

3.7 函数相等性测试

7min

函数相等性测试用来判断两个函数是否是同一个对象。

顶层函数、静态方法和实例方法相等性的测试示例代码如下：

```
//chapter3/bin/example_12.dart
//定义顶层函数
void foo(){}

//定义一个类
class A {
```

```
  //定义静态方法
  static void bar(){}
  //定义实例方法
  void baz(){}
}

void main() {
  var x;
  x = foo;
  //比较顶层函数是否相等
  print(foo == x);

  //比较静态方法是否相等
  x = A.bar;
  print(A.bar == x);

  //A 的实例 #1
  var v = A();
  //A 的实例 #2
  var w = A();
  var y = w;
  x = w.baz;
  //这两个闭包引用了相同的实例对象,因此它们相等
  print(y.baz == x);
  //这两个闭包引用了不同的实例对象,因此它们不相等
  print(v.baz != w.baz);
}
```

3min

3.8 返回值

所有的函数都有返回值,返回值本质上是对象。返回值类型可以是内置类型也可以是自定义类型,返回值类型放在函数名前面。示例代码如下:

```
//chapter3/bin/example_13.dart
void main(){
  //返回值类型为 num
  num area(num pi,num r) {
    //返回语句
    return pi * r * r;
  }
  print('圆面积为:${area(3.14,6)}');
}
```

当函数名前有类型修饰时，函数体最后必须提供 return 语句。当函数名前没有类型修饰且未提供 return 语句时，最后一行默认为执行 return null;。示例代码如下：

```
//chapter3/bin/example_14.dart
void main() {
  //定义没有提供返回值类型的函数
  foo(){
    //函数体未提供 return 语句
    var a;
  }
  //判断函数 foo 的返回值是否为 null
  print(foo() == null);
}
```

4min

3.9 回调函数

回调函数是指作为参数传递的函数，回调函数还会获得原函数提供的值。

首先将函数作为参数：

```
void printProgress({Function(int) callback}){...}
```

有条件地在函数内执行回调函数：

```
if(callback! = null) callback(progress);
```

接收原函数提供的参数：

```
printProgress(callback: (int progress){
  print('打印进度 : $progress');
});
```

完整的示例代码如下：

```
//chapter3/bin/example_15.dart
void main(){
  //定义 printProgress 函数，将回调函数 callback 作为可选参数
  void printProgress({Function(int) callback}){
    for (int progress = 0; progress <= 10; progress++){
      //判断回调函数是否为空
      if(callback!= null){
        //若提供则调用回调函数并传递循环变量 progress
        callback(progress);
```

```
        }
      }
    }
    //调用 printProgress 函数并提供匿名回调函数
    printProgress(callback: (int progress){
      print('打印进度 : $progress%');
    });
  }
```