**1.1. 函数的基本定义**

Dart是一种真正的面向对象语言，所以即使函数也是对象，所有也有类型, 类型就是Function。

这也就意味着函数可以作为变量定义或者作为其他函数的参数或者返回值.

函数的定义方式:

返回值 函数的名称(参数列表) {

函数体

return 返回值

}

按照上面的定义方式, 我们定义一个完整的函数:

int sum(num num1, num num2) {

return num1 + num2;

}

Effective Dart建议对公共的API, 使用类型注解, 但是如果我们省略掉了类型, 依然是可以正常工作的

sum(num1, num2) {

return num1 + num2;

}

另外, 如果函数中只有一个表达式, 那么可以使用箭头语法(*arrow* syntax)

* 注意, 这里面只能是一个表达式, 不能是一个语句

sum(num1, num2) => num1 + num2;

**1.2. 函数的参数问题**

函数的参数可以分成两类: 必须参数和可选参数

前面使用的参数都是必须参数.

**1.2.1. 可选参数**

可选参数可以分为 **命名可选参数** 和 **位置可选参数**

定义方式:

命名可选参数: {param1, param2, ...}

位置可选参数: [param1, param2, ...]

命名可选参数的演示:

// 命名可选参数

printInfo1(String name, {int age, double height}) {

print('name=$name age=$age height=$height');

}

// 调用printInfo1函数

printInfo1('why'); // name=why age=null height=null

printInfo1('why', age: 18); // name=why age=18 height=null

printInfo1('why', age: 18, height: 1.88); // name=why age=18 height=1.88

printInfo1('why', height: 1.88); // name=why age=null height=1.88

位置可选参数的演示:

// 定义位置可选参数

printInfo2(String name, [int age, double height]) {

print('name=$name age=$age height=$height');

}

// 调用printInfo2函数

printInfo2('why'); // name=why age=null height=null

printInfo2('why', 18); // name=why age=18 height=null

printInfo2('why', 18, 1.88); // name=why age=18 height=1.88

命名可选参数, 可以指定某个参数是必传的(使用@required, 有问题)

// 命名可选参数的必须

printInfo3(String name, {int age, double height, @required String address}) {

print('name=$name age=$age height=$height address=$address');

}

**1.2.2. 参数默认值**

参数可以有默认值, 在不传入的情况下, 使用默认值

* 注意, 只有可选参数才可以有默认值, 必须参数不能有默认值

// 参数的默认值

printInfo4(String name, {int age = 18, double height=1.88}) {

print('name=$name age=$age height=$height');

}

Dart中的main函数就是一个接受可选的列表参数作为参数的, 所以在使用main函数时, 我们可以传入参数, 也可以不传入

**1.3. 函数是一等公民**

在很多语言中, 函数并不能作为一等公民来使用, 比如Java/OC. 这种限制让编程不够灵活, 所以现代的编程语言基本都支持函数作为一等公民来使用, Dart也支持.

这就意味着你可以将函数赋值给一个变量, 也可以将函数作为另外一个函数的参数或者返回值来使用.

main(List<String> args) {

// 1.将函数赋值给一个变量

var bar = foo;

print(bar);

// 2.将函数作为另一个函数的参数

test(foo);

// 3.将函数作为另一个函数的返回值

var func =getFunc();

func('kobe');

}

// 1.定义一个函数

foo(String name) {

print('传入的name:$name');

}

// 2.将函数作为另外一个函数的参数

test(Function func) {

func('coderwhy');

}

// 3.将函数作为另一个函数的返回值

getFunc() {

return foo;

}

**1.4. 匿名函数的使用**

大部分我们定义的函数都会有自己的名字， 比如前面定义的foo、test函数等等。

但是某些情况下，给函数命名太麻烦了，我们可以使用没有名字的函数，这种函数可以被称之为匿名函数( *anonymous function*)，也可以叫*lambda*或者*closure*。

main(List<String> args) {

// 1.定义数组

var movies = ['盗梦空间', '星际穿越', '少年派', '大话西游'];

// 2.使用forEach遍历: 有名字的函数

printElement(item) {

print(item);

}

movies.forEach(printElement);

// 3.使用forEach遍历: 匿名函数

movies.forEach((item) {

print(item);

});

movies.forEach((item) => print(item));

}

**1.5. 词法作用域**

dart中的词法有自己明确的作用域范围，它是根据代码的结构({})来决定作用域范围的

优先使用自己作用域中的变量，如果没有找到，则一层层向外查找。

var name = 'global';

main(List<String> args) {

// var name = 'main';

void foo() {

// var name = 'foo';

print(name);

}

foo();

}

**1.6. 词法闭包**

闭包可以访问其词法范围内的变量，即使函数在其他地方被使用，也可以正常的访问。

main(List<String> args) {

makeAdder(num addBy) {

return (num i) {

return i + addBy;

};

}

var adder2 = makeAdder(2);

print(adder2(10)); // 12

print(adder2(6)); // 8

var adder5 = makeAdder(5);

print(adder5(10)); // 15

print(adder5(6)); // 11

}

**1.7. 返回值问题**

所有函数都返回一个值。如果没有指定返回值，则语句返回null;隐式附加到函数体。

main(List<String> args) {

print(foo()); // null

}

foo() {

print('foo function');

}