**1.1. 明确声明(Explicit)**

明确声明变量的方式, 格式如下:

变量类型 变量名称 = 赋值;

示例代码:

String name = 'coderwhy';

int age = 18;

double height = 1.88;

print('${name}, ${age}, ${height}'); // 拼接方式后续会讲解

注意事项: 定义的变量可以修改值, 但是不能赋值其他类型

String content = 'Hello Dart';

content = 'Hello World'; // 正确的

content = 111; // 错误的, 将一个int值赋值给一个String变量

**1.2. 类型推导(Type Inference)**

类型推导声明变量的方式, 格式如下:

var/dynamic/const/final 变量名称 = 赋值;

**1.3.1. var的使用**

var的使用示例:

* runtimeType用于获取变量当前的类型

var name = 'coderwhy';

name = 'kobe';

print(name.runtimeType); // String

var的错误用法:

var age = 18;

age = 'why'; // 不可以将String赋值给一个int类型

**1.3.2. dynamic的使用**

如果确实希望这样做,可以使用dynamic来声明变量:

* 但是在开发中, 通常情况下不使用dynamic, 因为类型的变量会带来潜在的危险

dynamic name = 'coderwhy';

print(name.runtimeType); // String

name = 18;

print(name.runtimeType); // int

**1.3.3. final&const的使用**

final和const都是用于定义常量的, 也就是定义之后值都不可以修改

final name = 'coderwhy';

name = 'kobe'; // 错误做法

const age = 18;

age = 20; // 错误做法

final和const有什么区别呢?

* const在赋值时, 赋值的内容必须是在编译期间就确定下来的
* final在赋值时, 可以动态获取, 比如赋值一个函数

String getName() {

return 'coderwhy';

}

main(List<String> args) {

const name = getName(); // 错误的做法, 因为要执行函数才能获取到值

final name = getName(); // 正确的做法

}

final和const小案例:

* 首先, const是不可以赋值为DateTime.now()
* 其次, final一旦被赋值后就有确定的结果, 不会再次赋值

// const time = DateTime.now(); // 错误的赋值方式

final time = DateTime.now();

print(time);

sleep(Duration(seconds: 2));

print(time);

const放在赋值语句的右边，可以共享对象，提高性能:

* 这里可以暂时先做了解，后面讲解类的常量构造函数时，我会再次提到这个概念

class Person {

const Person();

}

main(List<String> args) {

final a = const Person();

final b = const Person();

print(identical(a, b)); // true

final m = Person();

final n = Person();

print(identical(m, n)); // false

}