Flutter基础语法（六）var、final、const、late

明似水

前言

在使用flutter开发已经有一段时间了，在之前都是快速开发，没有时间去复盘自己的知识体系，现在就从flutter的基本语法开始，一步步分析flutter涉及到的细节问题，希望对你有所帮助。

一、var

1.var是什么?

- var就是定义变量的意思。
- var定义的变量会自动推断类型
- dart 中使用 var声明变量，可以赋值不同类型的值，会自动推断变量的类型。
- var 声明的变量如果没有初始化，那么它的值是nil

2.var如何使用

var 声明的变量，如果没有指定类型，是一个dynamic动态类型

```
var a;
print("打印==${a.runtimeType}");//打印==Null
```

```
var b;
b = "hello world";
print("打印==$b");//打印==hello world
print("打印==${b.runtimeType}");//打印==String
```

3.var自动推断类型

```
var b;
b = true;
print("打印==$b"); //打印==true
print("打印==${b.runtimeType}"); //打印==bool
```

4.var可以再次赋值

本来是bool类型的，后面变成了int类型

```
var b;    没有指定类型，可以赋值给不同的类型
b = true;
print("打印==$b"); //打印==true
print("打印==${b.runtimeType}"); //打印==bool


b = 100;
print("打印==$b"); //打印==100
print("打印==${b.runtimeType}"); //打印==int
```

5.var指定类型

在声明时候就赋值，则相当于指定类型，如果再赋值其他类型会报错

```
var b = 100;   已经指定了类型，不能赋值给不同的类型
b = true;//Error: A value of type 'bool' can't be assigned to a variable of type 'int'.b = true;

print("打印==$b"); //打印==true
print("打印==${b.runtimeType}"); //打印==bool
```

二、final

1.final是什么?

- 使用final声明的变量，它只能赋值一次。

- final修饰的是一个最终的变量，不能再次赋值，否则会报错
- 可以先声明再次赋值，但是只能赋值一次

2.final声明但不赋值

final声明的变量，如果不赋值，则无法使用，如runtimeType等方法。

final类型的变量，不是动态的，因此没有赋值前，不能使用。

```
final a;//The final variable 'a' can't be read because it's potentially unassigned at this point
print("打印==${a.runtimeType}");//报错
```

3.final赋值多次

final声明的变量，不能多次赋值，否则报错。

```
final a;
a = 100;
a = 'hello world';//Final variable 'a' might already be assigned at this point.a = 'hello world';
print("打印==${a.runtimeType}"); //
```

4.final正常使用

```
final a = 100;
print("打印==$a"); //打印==100
print("打印==${a.runtimeType}"); //打印==int
```

三、const

1.const是什么？

- const修饰常量，声明的时候就得赋值，这也是和变量最大的区别

2.const声明但不赋值

声明不赋值会报错：

```
///lib/modules/dart/xxx.dart:20:11: Error: The
const variable 'a' must be initialized.
/// Try adding an initializer ('= expression') to
the declaration.
///     const a;
const a;    const a = 100;
print("打印==$a"); //打印==100
print("打印==${a.runtimeType}"); //打印==int
```

3.const赋值多次

const声明的~~变量~~常量，不能多次赋值，否则报错。

```
///Error: Can't assign to the const variable 'a'.
///     a = 200;
const a = 100;
a = 200;
print("打印==$a"); //打印==100
print("打印==${a.runtimeType}"); //打印==int
```

4.const正常使用

```
const a = 100;
print("打印==$a"); //打印==100
print("打印==${a.runtimeType}"); //打印==int
```

三、late

1.late是什么?

- Dart 2.12引入了late修饰符
- 显式声明一个非空的变量，但不初始化
- 如果不加late关键字，类实例化时此值是不确定的，无法通过静态检查，加上late关键字可以通过静态检查，但由此会带来运行时风险
- 延迟初始化变量。如果这个变量没有被使用的话就不会被初始化，初始化语句不会执行。

2.late声明但不赋值

声明不赋值不会报错：

```
///Error: Late variable 'a' without initializer is definitely unassigned.
///     print("打印==${a.runtimeType}"); //打印==int
late int a;  声明非空的变量，可以通过静态检查
a = 100;
print("打印==$a"); //打印==100
print("打印==${a.runtimeType}"); //打印==int
```

3.late赋值多次

已经指定了类型，不能赋值别的类型

```
late int a;   late修饰的变量，必须指定类型；已经指定了类型，就不能赋值其他类型的值
a = 100;
a = 300; 可以再次赋值为int类型的值，但是不能赋值为String类型的值
print("打印==$a"); //打印==300
print("打印==${a.runtimeType}"); //打印==int
```

4.late正常使用

```
late int a;
a = 100;
print("打印==$a"); //打印==100
print("打印==${a.runtimeType}"); //打印==int
```

总结

以上就是今天要讲的内容，本文仅仅简单介绍了var、final、const、late的使用。希望文章对你有所帮助，后续继续完善。