Swift 基本语法 Swift 变量

Swift 数据类型

在我们使用任何程序语言编程时，需要使用各种数据类型来存储不同的信息。

变量的数据类型决定了如何将代表这些值的位存储到计算机的内存中。在声明变量时也可指定它的数据类型。

所有变量都具有数据类型，以决定能够存储哪种数据。

内置数据类型

Swift 提供了非常丰富的数据类型，以下列出了常用了几种数据类型：

Int

一般来说，你不需要专门指定整数的长度。Swift 提供了一个特殊的整数类型`Int`，长度与当前平台的原生字长相同：

- 在32位平台上，`Int`和`Int32`长度相同。
- 在64位平台上，`Int`和`Int64`长度相同。

除非你需要特定长度的整数，一般来说使用`Int`就够了。这可以提高代码一致性和可复用性。即使是在32位平台上，`Int`可以存储的整数范围也可以达到`-2,147,483,648`~`2,147,483,647`，大多数时候这已经足够大了。

UInt

Swift 也提供了一个特殊的无符号类型`UInt`，长度与当前平台的原生字长相同：

- 在32位平台上，`UInt`和`UInt32`长度相同。
- 在64位平台上，`UInt`和`UInt64`长度相同。

> ***注意：*** *尽量不要使用`UInt`，除非你真的需要存储一个和当前平台原生字长相同的无符号整数。除了这种情况，最好使用`Int`，即使你要存储的值已知是非负的。统一使用`Int`可以提高代码的可复用性，避免不同类型数字之间的转换，并且匹配数字的类型推断。*

整数类型需要注意以下几点：

- 在 32 位系统上, Int 和 Int32 长度相同。
- 在 64 位系统上, Int 和 Int64 长度相同。
- 在 32 位系统上, UInt 和 UInt32 长度相同。
- 在 64 位系统上, UInt 和 UInt64 长度相同。
- Int8, Int16, Int32, Int64 分别表示 8 位, 16 位, 32 位, 和 64 位的有符号整数形式。
- UInt8, UInt16, UInt32, UInt64 分别表示 8 位, 16 位, 32 位 和 64 位的无符号整数形式。

浮点数：Float、Double

浮点数是有小数部分的数字，比如 `3.14159`，`0.1` 和 `-273.15`。

浮点类型比整数类型表示的范围更大，可以存储比 `Int` 类型更大或者更小的数字。Swift 提供了两种有符号浮点数类型：

- **Double** 表示64位浮点数。当你需要存储很大或者很高精度的浮点数时请使用此类型。
- **Float** 表示32位浮点数。精度要求不高的话可以使用此类型。

> *注意：*
> *`Double`精确度很高，至少有15位数字，而 `Float` 最少只有6位数字。选择哪个类型取决于你的代码需要处理的值的范围。*

布尔值：Bool

Swift 有一个基本的布尔（Boolean）类型，叫做 Bool。布尔值指逻辑上的值，因为它们只能是真或者假。Swift 有两个布尔常量，true 和 false。

字符串：String

字符串是字符的序列集合，例如：

```
"Hello, World!"
```

字符：Character

字符指的是单个字母，例如：

```
"C"
```

可选类型：Optional

使用可选类型来处理值可能缺失的情况。可选类型表示有值或没有值。

数值范围

下表显示了不同变量类型内存的存储空间，及变量类型的最大最小值：

类型	大小（字节）	区间值
Int8	1 字节	-128 到 127
UInt8	1 字节	0 到 255
Int32	4 字节	-2147483648 到 2147483647
UInt32	4 字节	0 到 4294967295
Int64	8 字节	-9223372036854775808 到 9223372036854775807

UInt64	8 字节	0 到 18446744073709551615
Float	4 字节	1.2E-38 到 3.4E+38 (~6 digits)
Double	8 字节	2.3E-308 到 1.7E+308 (~15 digits)

类型别名

类型别名对当前的类型定义了另一个名字，类型别名通过使用 typealias 关键字来定义。语法格式如下：

```
typealias newname = type
```

例如以下定义了 Int 的类型别名为 Feet：

```
typealias Feet = Int
```

现在，我们可以通过别名来定义变量：

```
import Cocoa

typealias Feet = Int
var distance: Feet = 100
print(distance)
```

我们使用 playground 执行以上程序，输出结果为：

```
100
```

类型安全

Swift 是一个类型安全（type safe）的语言。

由于 Swift 是类型安全的，所以它会在编译你的代码时进行类型检查（type checks），并把不匹配的类型标记为错误。这可以让你在开发的时候尽早发现并修复错误。

```
import Cocoa

var varA = 42
varA = "This is hello"
print(varA)
```

以上程序，会在 Xcode 中报错：

```
error: cannot assign value of type 'String' to type 'Int'
varA = "This is hello"
```

意思为不能将 'String' 字符串赋值给 'Int' 变量。

类型推断

当你要处理不同类型的值时，类型检查可以帮你避免错误。然而，这并不是说你每次声明常量和变量的时候都需要显式指定类型。

如果你没有显式指定类型，Swift 会使用类型推断（type inference）来选择合适的类型。

例如，如果你给一个新常量赋值42并且没有标明类型，Swift 可以推断出常量类型是Int，因为你给它赋的初始值看起来像一个整数：

```
let meaningOfLife = 42
// meaningOfLife 会被推测为 Int 类型
```

同理，如果你没有给浮点字面量标明类型，Swift 会推断你想要的是Double：

```
let pi = 3.14159
// pi 会被推测为 Double 类型
```

当推断浮点数的类型时，Swift 总是会选择Double而不是Float。

如果表达式中同时出现了整数和浮点数，会被推断为Double类型：

```
let anotherPi = 3 + 0.14159
// anotherPi 会被推测为 Double 类型
```

原始值3没有显式声明类型，而表达式中出现了一个浮点字面量，所以表达式会被推断为Double类型。

实例

```
import Cocoa

// varA 会被推测为 Int 类型
var varA = 42
print(varA)

// varB 会被推测为 Double 类型
var varB = 3.14159
print(varB)

// varC 也会被推测为 Double 类型
```

```
var varC = 3 + 0.14159
print(varC)
```

执行以上代码，输出结果为：

```
42
3.14159
3.14159
```

Swift 基本语法 Swift 变量

点我分享笔记