# 第 3 章 Swift基本语法



本章主要为大家介绍Swift的一些基本语法，其中包括标识符和关键字、常量、变量、表达式和注释等内容。

## 3.1 标识符和关键字

任何一种计算机语言都离不开标识符和关键字，下面我们将详细介绍Swift标识符和关键字。

### 3.1.1 标识符

**标识符**就是给变量、常量、方法、函数、枚举、结构体、类、协议等指定的名字。构成标识符的字母均有一定的规范，Swift语言中标识符的命名规则如下：

- 区分大小写，Myname与myname是两个不同的标识符；
- 标识符首字符可以以下划线（_）或者字母开始，但不能是数字；
- 标识符中其他字符可以是下划线（_）、字母或数字。

例如，identifier、userName、User_Name、_sys_val、身高等为合法的标识符，而2mail、room#和class为非法的标识符。其中，使用中文“身高”命名的变量是合法的。

---

**注意** Swift中的字母采用的是Unicode编码[①]。Unicode叫做统一编码制，它包含了亚洲文字编码，如中文、日文、韩文等字符，甚至是我们在聊天工具中使用的表情符号，如😄 😊 😃等，这些符号事实上也是Unicode，而非图片。这些符号在Swift中都可以使用。

---

如果一定要使用关键字作为标识符，可以在关键字前后添加重音符号（`），例如：

```
let π = 3.14159

let_Hello = "Hello"
```

① Unicode是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。Unicode用数字0-0x10FFFF来映射这些字符，最多可以容纳1 114 112个字符，或者说有1 114 112个码位。——百度百科http://baike.baidu.com/view/2602518.htm

```
let 您好 = "你好世界"

let `class` = "😄😊😃"

//诺亚方舟
let 🌍 = "🐶🐱🐭🐹🐰🦊🐻🐼🐨🐯🦁🐮🐷🐽🐸🐵🙈🙉🙊🐒🐔🐧🐦🐤🐣🐥🦆🦅🦉🦇🐺🐗🐴🦄🐝🐛🦋🐌🐞🐜🦟🦗🕷🦂🐢🐍🦎🦖🦕🐙🦑🦐🦞🦀🐡🐠🐟🐬🐳🐋🦈🐊🐅🐆🦓🦍🦧🐘🦛🦏🐪🐫🦒🦘🐃🐂🐄🐎🐖🐏🐑🦙🐐🦌🐕🐩🦮🐈🐓🦃🦚🦜🦢🦩🕊🐇🦝🦨🦡🦦🦥🐁🐀🐿🦔🐾🐉🐲🌵🎄🌲🌳🌴🌱🌿☘🍀🎍🎋🍃🍂🍁🍄🐚🌾💐🌷🌹🥀🌺🌸🌼🌻🌞🌝🌛🌜🌚🌕🌖🌗🌘🌑🌒🌓🌔🌙🌎🌍🌏🪐💫⭐🌟✨⚡☄💥🔥🌪🌈☀🌤⛅🌥☁🌦🌧⛈🌩🌨❄☃⛄🌬💨💧💦☔☂🌊🌫"
```

其中class是关键字，事实上重音符号（`）不是标识符的一部分，它也可以用于其他标识符，如π和`π`是等价的。因此使用关键字作为标识符是一种很不好的编程习惯。

### 3.1.2　关键字

**关键字**是类似于标识符的保留字符序列，除非用重音符号（`）将其括起来，否则不能用作标识符。关键字是对编译器具有特殊意义的预定义保留标识符。常见的关键字有以下4种。

- 与声明有关的关键字：class、deinit、enum、extension、func、import、init、let、protocol、static、struct、subscript、typealias和var。
- 与语句有关的关键字：break、case、continue、default、do、else、fallthrough、if、in、for、return、switch、where和while。
- 表达式和类型关键字：as、dynamicType、is、new、super、self、Self、Type、__COLUMN__、__FILE__、__FUNCTION__和__LINE__。
- 在特定上下文中使用的关键字：associativity、didSet、get、infix、inout、left、mutating、none、nonmutating、operator、override、postfix、precedence、prefix、rightset、unowned、unowned(safe)、unowned(unsafe)、weak和willSet。

对于上述关键字，目前我们没有必要全部知道它们的含义，但是要记住：在Swift中，关键字是区分大小写的，因此class和Class是不同的，当然Class不是Swift的关键字。

## 3.2　常量和变量

我们在上一章中介绍了如何使用Swift编写一个HelloWorld小程序，其中就用到了变量。常量和变量是构成表达式的重要组成部分。

### 3.2.1　常量

在声明和初始化变量时，在标识符的前面加上关键字let，就可以把该变量指定为一个常量。顾名思义，**常量**是其值在使用过程中不会发生变化的量，实例代码如下：

```
let_Hello = "Hello"
```

_Hello标识符就是常量，只能在初始化的时候被赋值，如果我们再次给_Hello赋值，代码如下：

```
_Hello = "Hello, World"
```

则程序会报错，如图3-1所示，时间轴中显示了错误信息。

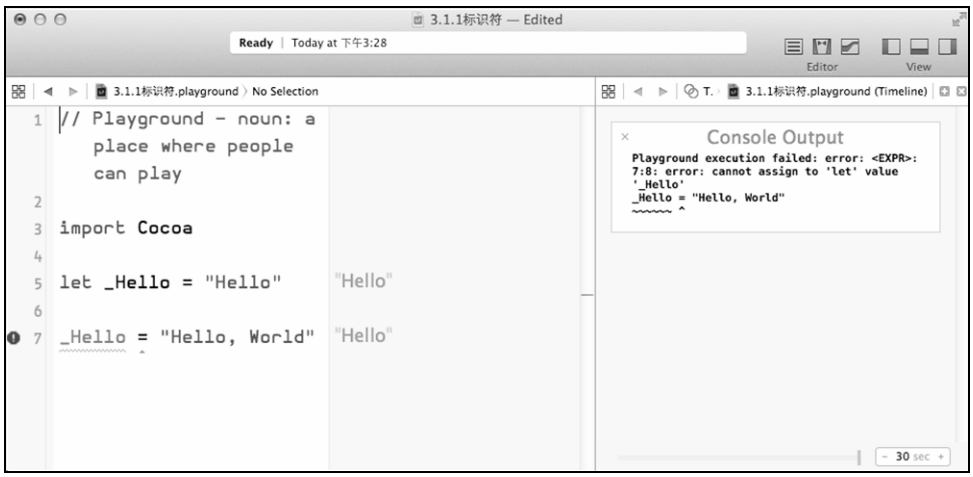


图3-1 时间轴中错误信息

从错误信息可以获知_Hello是let分配的值，不能被赋值。

### 3.2.2 变量

在Swift中声明变量，就是在标识符的前面加上关键字var，实例代码如下：

```
var scoreForStudent = 0.0
```

该语句声明Double类型scoreForStudent变量，并且初始化为0.0。如果在一个语句中声明和初始化了多个变量，那么所有的变量都具有相同的数据类型：

```
var x = 10, y = 20
```

在多个变量的声明中，我们也能指定不同的数据类型：

```
var x = 10, y = true
```

其中x为整型，y为布尔型。

### 3.2.3 命名规范

在使用常量和变量的时候，要保证它们的命名符合规范，这样程序才具有良好的可读性。这也是一种良好的编程习惯。

**1. 常量名**

基本数据类型的常量名全为大写，如果由多个单词构成，则可以用下划线隔开，例如：

```
let YEAR = 60
let WEEK_OF_MONTH = 3
```

### 2. 变量名

变量的命名有多种风格，主要以清楚易懂为主。有些程序员为了方便，使用单个字母来作为变量名称，如j和i等，这会为日后程序维护带来困难，变量同名的概率也会增加。单个字母变量一般只用于循环变量，因为它们只作用于循环体内。

在过去，计算机语言对变量名称的长度会有所限制，但现在已经没有这种限制了，因此我们鼓励用清楚的名称来表明变量的作用，通常会以小写字母作为开始，其余单词首字母大写，例如：

```
var maximumNumberOfLoginAttempts = 10
var currentLoginAttempt = 0
```

这样的名称可以令变量的作用一目了然。

除了常量和变量的命名要规范之外，其他语言对象的命名也要规范。其中类名、协议名、结构体、枚举等类型的命名规范通常是，大写字母作为开始，其余单词首字母大写，例如类名HelloWorldApp。

函数和方法名往往由多个单词合成，第一个单词通常为动词，以小写字母作为开始，其余单词首字母大写，例如balanceAccount和isButtonPressed。

## 3.3　注释

Swift程序有两类注释：单行注释（ // ）和多行注释（ /*...*/ ）。注释方法与C、C++和Objective-C语言都是类似的，下面详细介绍一下。

### 1. 单行注释

单行注释可以注释整行或者一行中的一部分，一般不用于连续多行的注释文本。当然，它也可以用来注释连续多行的代码段。以下是两种注释风格的例子：

```
if x > 1 {
    //注释1
} else {
    return false; //注释2
}

//if x > 1 {
//    //注释1
//} else {
//    return false; //注释2
//}
```

**提示**　在Xcode中对连续多行的注释文本可以使用快捷键：选择多行然后按住“command+/”键进行注释。去掉注释也是按住“command+/”键。

### 2. 块注释

一般用于连续多行的注释文本，但也可以对单行进行注释。以下是几种注释风格的例子：

```
if x > 1 {
    /* 注释1 */
} else {
    return false; /* 注释2 */
}

/*
if x > 1 {
    //注释1
} else {
    return false; //注释2
}
*/

/*
if x > 1 {
    /* 注释1 */
} else {
    return false; /* 注释2 */
}
*/
```



**提示** Swift多行注释有一个其他语言没有的优点，就是可以嵌套，上述示例的最后一种情况便实现了多行注释嵌套。

在程序代码中，对容易引起误解的代码进行注释是必要的，但应避免对已清晰表达信息的代码进行注释。需要注意的是，频繁的注释有时反映了代码的低质量。当你觉得被迫要加注释的时候，不妨考虑一下重写代码使其更清晰。

## 3.4 表达式

**表达式**是程序代码的重要组成部分，在Swift中，表达式有3种形式。

### 1. 不指定数据类型

```
var a1 = 10
let a2 = 20
var a = a1 > a2 ? "a1" : "a2"
```

在上述代码中，我们直接为变量或常量赋值，并没有指定数据类型，因为在Swift中可以自动推断数据类型。

### 2. 指定数据类型

```
var a1:Int  = 10
let a2:Int = 20
var a = a1 > a2 ? "a1" : "a2"
```

在上述代码中，:Int是为变量和常量指定数据类型。这种写法使程序可读性良好，我们推荐明确指定变量和常量的数据类型。

3. 使用分号

```
var a1:Int = 10; var a2:Int = 20
var a = a1 > a2 ? "a1" : "a2"
```

在Swift语言中，一条语句结束后可以不加分号也可以添加分号，但是有一种情况必须要用分号，那就是多条语句写在一行的时候，需要通过分号来区别语句。例如：

```
var a1:Int = 10; var a2:Int = 20;
```

## 3.5　本章小结

通过对本章内容的学习，我们可以了解到Swift语言的基本语法，其中包括标识符和关键字、常量、变量、表达式和注释等内容。

## 3.6　同步练习

1. 下列是Swift合法标识符的是（　　）。

A. `2variable`　　B. `variable2`　　C. `_whatavariable`　　D. `_3_`

E. `$anothervar`　　F. `#myvar`　　G. `体重`　　H. 

I. `` `class` ``

2. 下列不是Swift关键字的是（　　）。

A. `if`　　B. `then`　　C. `goto`　　D. `while`

E. `case`　　F. `__COLUMN__`　　G. `where`　　H. `Class`

3. 描述下列代码的运行结果。

```
let _Hello1 = "Hello"          ①
_Hello1 = "Hello, World"       ②
println(_Hello1)               ③
var _Hello2 = "Hello"          ④
_Hello2 = "Hello, World"       ⑤
println(_Hello2)               ⑥
```

4. 下列有关Swift注释使用正确的是（　　）。

A.

```
if x > 1 {
      //注释1
 } else {
      return false; //注释2
 }
```

B.

```
//let _Hello1 = "Hello"
//_Hello1 = "Hello, World"
//println(_Hello1)
```

C.

```
/*
let _Hello1 = "Hello"
_Hello1 = "Hello, World"
println(_Hello1)
*/
```

D.

```
/**
let _Hello1 = "Hello"
_Hello1 = "Hello, World"
println(_Hello1)
*/
```

5. 下列表达式不正确的是（ ）。

A. `var n1:Int = 10;`

B. `var n1:Int = 10`

C. `var n1 = 10`

D. `var n1:Int = 10; var str:String = 20`

E. `var n1:Int = 10; var str:String = "20"`

F. `var n1:Int = 10; var str:String = '20'`