Swift 可选链 Swift 扩展

Swift 类型转换

Swift 语言类型转换可以判断实例的类型。也可以用于检测实例类型是否属于其父类或者子类的实例。

Swift 中类型转换使用 is 和 as 操作符实现，is 用于检测值的类型，as 用于转换类型。

类型转换也可以用来检查一个类是否实现了某个协议。

定义一个类层次

以下定义了三个类：Subjects、Chemistry、Maths，Chemistry 和 Maths 继承了 Subjects。

代码如下：

```
class Subjects {
    var physics: String
    init(physics: String) {
        self.physics = physics
    }
}

class Chemistry: Subjects {
    var equations: String
    init(physics: String, equations: String) {
        self.equations = equations
        super.init(physics: physics)
    }
}

class Maths: Subjects {
    var formulae: String
    init(physics: String, formulae: String) {
        self.formulae = formulae
        super.init(physics: physics)
    }
}

let sa = [
    Chemistry(physics: "固体物理", equations: "赫兹"),
    Maths(physics: "流体动力学", formulae: "千兆赫")]


let samplechem = Chemistry(physics: "固体物理", equations: "赫兹")
print("实例物理学是: \(samplechem.physics)")
print("实例方程式: \(samplechem.equations)")
```

```
let samplemaths = Maths(physics: "流体动力学", formulae: "千兆赫")
print("实例物理学是: \(samplemaths.physics)")
print("实例公式是: \(samplemaths.formulae)")
```

以上程序执行输出结果为：

```
实例物理学是: 固体物理
实例方程式: 赫兹
实例物理学是: 流体动力学
实例公式是: 千兆赫
```

检查类型

类型转换用于检测实例类型是否属于特定的实例类型。

你可以将它用在类和子类的层次结构上，检查特定类实例的类型并且转换这个类实例的类型成为这个层次结构中的其他类型。

类型检查使用 **is** 关键字。

操作符 **is** 来检查一个实例是否属于特定子类型。若实例属于那个子类型，类型检查操作符返回 true，否则返回 false。

```
class Subjects {
    var physics: String
    init(physics: String) {
        self.physics = physics
    }
}

class Chemistry: Subjects {
    var equations: String
    init(physics: String, equations: String) {
        self.equations = equations
        super.init(physics: physics)
    }
}

class Maths: Subjects {
    var formulae: String
    init(physics: String, formulae: String) {
        self.formulae = formulae
        super.init(physics: physics)
    }
}

let sa = [
    Chemistry(physics: "固体物理", equations: "赫兹"),
    Maths(physics: "流体动力学", formulae: "千兆赫"),
    Chemistry(physics: "热物理学", equations: "分贝"),
    Maths(physics: "天体物理学", formulae: "兆赫"),
```

```
    Maths(physics: "微分方程", formulae: "余弦级数")]


let samplechem = Chemistry(physics: "固体物理", equations: "赫兹")
print("实例物理学是: \(samplechem.physics)")
print("实例方程式: \(samplechem.equations)")


let samplemaths = Maths(physics: "流体动力学", formulae: "千兆赫")
print("实例物理学是: \(samplemaths.physics)")
print("实例公式是: \(samplemaths.formulae)")

var chemCount = 0
var mathsCount = 0
for item in sa {
    // 如果是一个 Chemistry 类型的实例，返回 true，相反返回 false。
    if item is Chemistry {
        ++chemCount
    } else if item is Maths {
        ++mathsCount
    }
}

print("化学科目包含 \(chemCount) 个主题，数学包含 \(mathsCount) 个主题")
```

以上程序执行输出结果为：

```
实例物理学是: 固体物理
实例方程式: 赫兹
实例物理学是: 流体动力学
实例公式是: 千兆赫
化学科目包含 2 个主题，数学包含 3 个主题
```

向下转型

向下转型，用类型转换操作符(as? 或 as!)

当你不确定向下转型可以成功时，用类型转换的条件形式(as?)。条件形式的类型转换总是返回一个可选值（optional value），并且若下转是不可能的，可选值将是 nil。

只有你可以确定向下转型一定会成功时，才使用强制形式(as!)。当你试图向下转型为一个不正确的类型时，强制形式的类型转换会触发一个运行时错误。

```
class Subjects {
    var physics: String
    init(physics: String) {
        self.physics = physics
```

```
    }
}

class Chemistry: Subjects {
    var equations: String
    init(physics: String, equations: String) {
        self.equations = equations
        super.init(physics: physics)
    }
}

class Maths: Subjects {
    var formulae: String
    init(physics: String, formulae: String) {
        self.formulae = formulae
        super.init(physics: physics)
    }
}

let sa = [
    Chemistry(physics: "固体物理", equations: "赫兹"),
    Maths(physics: "流体动力学", formulae: "千兆赫"),
    Chemistry(physics: "热物理学", equations: "分贝"),
    Maths(physics: "天体物理学", formulae: "兆赫"),
    Maths(physics: "微分方程", formulae: "余弦级数")]


let samplechem = Chemistry(physics: "固体物理", equations: "赫兹")
print("实例物理学是: \(samplechem.physics)")
print("实例方程式: \(samplechem.equations)")


let samplemaths = Maths(physics: "流体动力学", formulae: "千兆赫")
print("实例物理学是: \(samplemaths.physics)")
print("实例公式是: \(samplemaths.formulae)")

var chemCount = 0
var mathsCount = 0

for item in sa {
    // 类型转换的条件形式
    if let show = item as? Chemistry {
        print("化学主题是: '\(show.physics)', \(show.equations)")
        // 强制形式
    } else if let example = item as? Maths {
        print("数学主题是: '\(example.physics)',  \(example.formulae)")
    }
}
```

以上程序执行输出结果为：

```
实例物理学是: 固体物理
实例方程式: 赫兹
实例物理学是: 流体动力学
实例公式是: 千兆赫
化学主题是: '固体物理', 赫兹
数学主题是: '流体动力学',  千兆赫
化学主题是: '热物理学', 分贝
数学主题是: '天体物理学',  兆赫
数学主题是: '微分方程',  余弦级数
```

Any和AnyObject的类型转换

Swift为不确定类型提供了两种特殊类型别名：

- `AnyObject`可以代表任何class类型的实例。
- `Any`可以表示任何类型，包括方法类型（function types）。

> *注意：*
> *只有当你明确的需要它的行为和功能时才使用`Any`和`AnyObject`。在你的代码里使用你期望的明确的类型总是更好的。*

Any 实例

```
class Subjects {
    var physics: String
    init(physics: String) {
        self.physics = physics
    }
}

class Chemistry: Subjects {
    var equations: String
    init(physics: String, equations: String) {
        self.equations = equations
        super.init(physics: physics)
    }
}

class Maths: Subjects {
    var formulae: String
    init(physics: String, formulae: String) {
        self.formulae = formulae
        super.init(physics: physics)
    }
```

```
}

let sa = [
    Chemistry(physics: "固体物理", equations: "赫兹"),
    Maths(physics: "流体动力学", formulae: "千兆赫"),
    Chemistry(physics: "热物理学", equations: "分贝"),
    Maths(physics: "天体物理学", formulae: "兆赫"),
    Maths(physics: "微分方程", formulae: "余弦级数")]


let samplechem = Chemistry(physics: "固体物理", equations: "赫兹")
print("实例物理学是: \(samplechem.physics)")
print("实例方程式: \(samplechem.equations)")


let samplemaths = Maths(physics: "流体动力学", formulae: "千兆赫")
print("实例物理学是: \(samplemaths.physics)")
print("实例公式是: \(samplemaths.formulae)")

var chemCount = 0
var mathsCount = 0

for item in sa {
    // 类型转换的条件形式
    if let show = item as? Chemistry {
        print("化学主题是: '\(show.physics)', \(show.equations)")
        // 强制形式
    } else if let example = item as? Maths {
        print("数学主题是: '\(example.physics)',  \(example.formulae)")
    }
}

// 可以存储Any类型的数组 exampleany
var exampleany = [Any]()

exampleany.append(12)
exampleany.append(3.14159)
exampleany.append("Any 实例")
exampleany.append(Chemistry(physics: "固体物理", equations: "兆赫"))

for item2 in exampleany {
    switch item2 {
    case let someInt as Int:
        print("整型值为 \(someInt)")
    case let someDouble as Double where someDouble > 0:
        print("Pi 值为 \(someDouble)")
    case let someString as String:
        print("\(someString)")
    case let phy as Chemistry:
```

```
        print("主题 '\(phy.physics)', \(phy.equations)")
    default:
        print("None")
    }
}
```

以上程序执行输出结果为：

```
实例物理学是: 固体物理
实例方程式: 赫兹
实例物理学是: 流体动力学
实例公式是: 千兆赫
化学主题是: '固体物理', 赫兹
数学主题是: '流体动力学',  千兆赫
化学主题是: '热物理学', 分贝
数学主题是: '天体物理学',  兆赫
数学主题是: '微分方程',  余弦级数
整型值为 12
Pi 值为 3.14159
Any 实例
主题 '固体物理', 兆赫
```

AnyObject 实例

```
class Subjects {
    var physics: String
    init(physics: String) {
        self.physics = physics
    }
}

class Chemistry: Subjects {
    var equations: String
    init(physics: String, equations: String) {
        self.equations = equations
        super.init(physics: physics)
    }
}

class Maths: Subjects {
    var formulae: String
    init(physics: String, formulae: String) {
        self.formulae = formulae
        super.init(physics: physics)
    }
}
```

```
// [AnyObject] 类型的数组
let saprint: [AnyObject] = [
    Chemistry(physics: "固体物理", equations: "赫兹"),
    Maths(physics: "流体动力学", formulae: "千兆赫"),
    Chemistry(physics: "热物理学", equations: "分贝"),
    Maths(physics: "天体物理学", formulae: "兆赫"),
    Maths(physics: "微分方程", formulae: "余弦级数")]


let samplechem = Chemistry(physics: "固体物理", equations: "赫兹")
print("实例物理学是: \(samplechem.physics)")
print("实例方程式: \(samplechem.equations)")


let samplemaths = Maths(physics: "流体动力学", formulae: "千兆赫")
print("实例物理学是: \(samplemaths.physics)")
print("实例公式是: \(samplemaths.formulae)")

var chemCount = 0
var mathsCount = 0

for item in saprint {
    // 类型转换的条件形式
    if let show = item as? Chemistry {
        print("化学主题是: '\(show.physics)', \(show.equations)")
        // 强制形式
    } else if let example = item as? Maths {
        print("数学主题是: '\(example.physics)',  \(example.formulae)")
    }
}

var exampleany = [Any]()
exampleany.append(12)
exampleany.append(3.14159)
exampleany.append("Any 实例")
exampleany.append(Chemistry(physics: "固体物理", equations: "兆赫"))

for item2 in exampleany {
    switch item2 {
    case let someInt as Int:
        print("整型值为 \(someInt)")
    case let someDouble as Double where someDouble > 0:
        print("Pi 值为 \(someDouble)")
    case let someString as String:
        print("\(someString)")
    case let phy as Chemistry:
        print("主题 '\(phy.physics)', \(phy.equations)")
    default:
        print("None")
```

```
    }
}
```

以上程序执行输出结果为：

```
实例物理学是: 固体物理
实例方程式: 赫兹
实例物理学是: 流体动力学
实例公式是: 千兆赫
化学主题是: '固体物理', 赫兹
数学主题是: '流体动力学', 千兆赫
化学主题是: '热物理学', 分贝
数学主题是: '天体物理学', 兆赫
数学主题是: '微分方程', 余弦级数
整型值为 12
Pi 值为 3.14159
Any 实例
主题 '固体物理', 兆赫
```

在一个switch语句的case中使用强制形式的类型转换操作符（as, 而不是 as?）来检查和转换到一个明确的类型。

← Swift 可选链 Swift 扩展 →