美团·大众点评
meituan dianping

响应式编程在 iOS 开发中的应用

WELCOME

自我介绍

- 美团·大众点评 iOS 技术专家，国内 Functional Reactive Programming 技术爱好者。
- 2015年加入美团·大众点评，负责美团·大众点评北京侧发布工程系统的研发和流程优化梳理。
- 擅长多语言范式，对各种编程范式有着独到的见解。在美团·大众点评北京侧和 StuQ 组织过系统的 FRP 培训，参与人数达数百人，积累了一定经验。

内容

01 聊一个需求

02 想一个问题

03 讲一个方式

04 给一些建议

先从这一个需求开始

商家详情页

请求网络获取商家详情页

显示到UI上

控件产生动作

背后的需求

抓取数据的API
不只一个怎么办?

滚动改变导
航栏变化

……

网络抓取前的
数据如何处理?

显示的内容是
有选择性的

复杂点

- 异步数据拉取逻辑
- UI后期调整
- 控件相互作用

解决的方式

初始化状态　　改变状态　　判断状态

思考一个问题

学知识的时候和做业务的时候
差距有多大?

知识≠工作

我们所学到的知识

编程语言　　计算机网络　　各类API

```
func maxInArray<T: Comparable>(input: [T]) -> T {
    var max = input[0]
}
```

理想模型

```
NSURLSessionDownloadTask *downloadTask =
    [manager downloadTaskWithRequest:request
                            progress:nil
                         destination:^NSURL *(NSURL *targetPath,
                                              NSURLResponse *respo
{
    NSURL *documentsDirectoryURL =
        [[NSFileManager defaultManager]
            URLForDirectory:NSDocumentDirectory
                   inDomain:NSUserDomainMask
          appropriateForURL:nil
                     create:NO
                      error:nil];
    return [documentsDirectoryURL URLByAppendingPathComponent:
            [response suggestedFilename]];
} completionHandler:^(NSURLResponse *response,
                      NSURL *filePath,
                      NSError *error) {
    NSLog(@"File downloaded to: %@", filePath);
}];
[downloadTask resume];
```

我们该做什么？

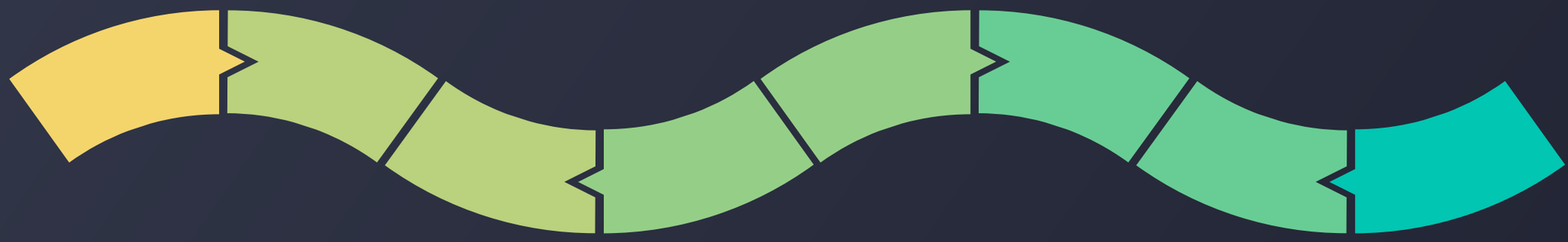
- 实践模型

- 理论知识

新知识如何应用?

曲折的学习路径

- 知识不连贯
- 代码没人懂



- 概念多
- 调试难

新旧知识的桥梁

方法论

了解概念

深入需求

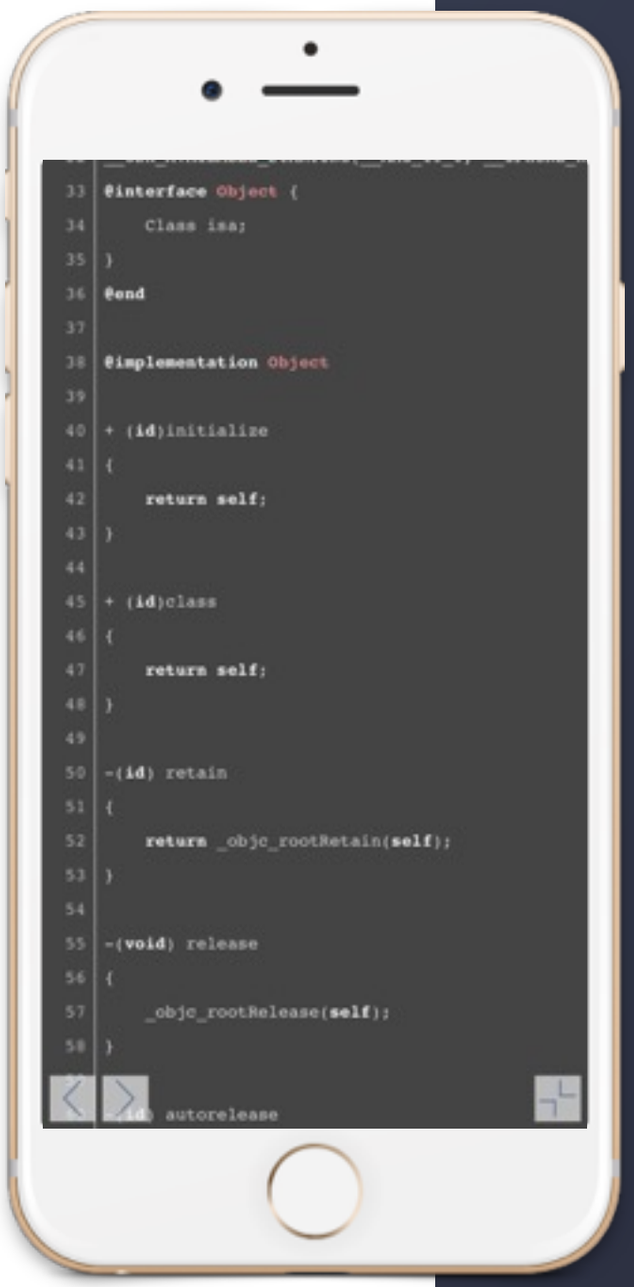
建立映射关系

处理特殊CASE

响应式编程

实践响应式编程

命令式编程



状态 + 命令

A = B + C

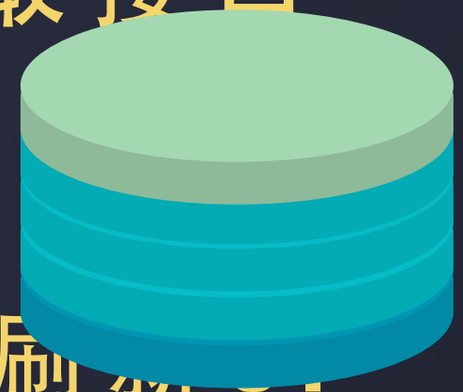
A = function(B, C)

A.method()

C = A.method2(B)

当命令式遇到异步

- 异步获取接口



- 获取后刷新UI
- 很多的状态量
- 事件异步通知

响应式编程

函数响应式编程

响应式编程的两种方式

Pull Driver

Push Driver

非响应式

量A
3

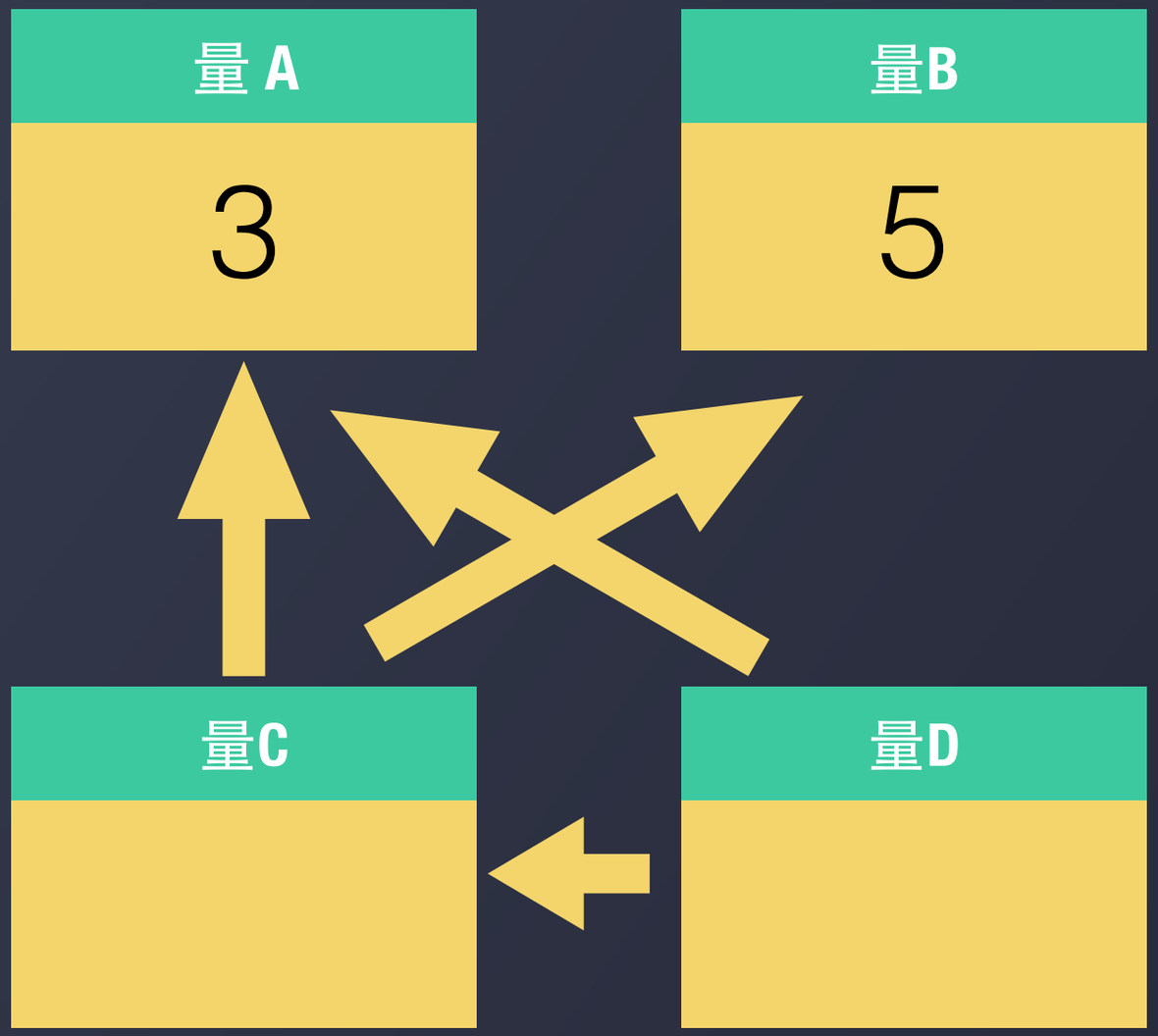
量B
5

量C
8

量D
11

```
var a = 3
var b = 5
var c = a + b
var d = c + a
b = 8
```

PULL DRIVER



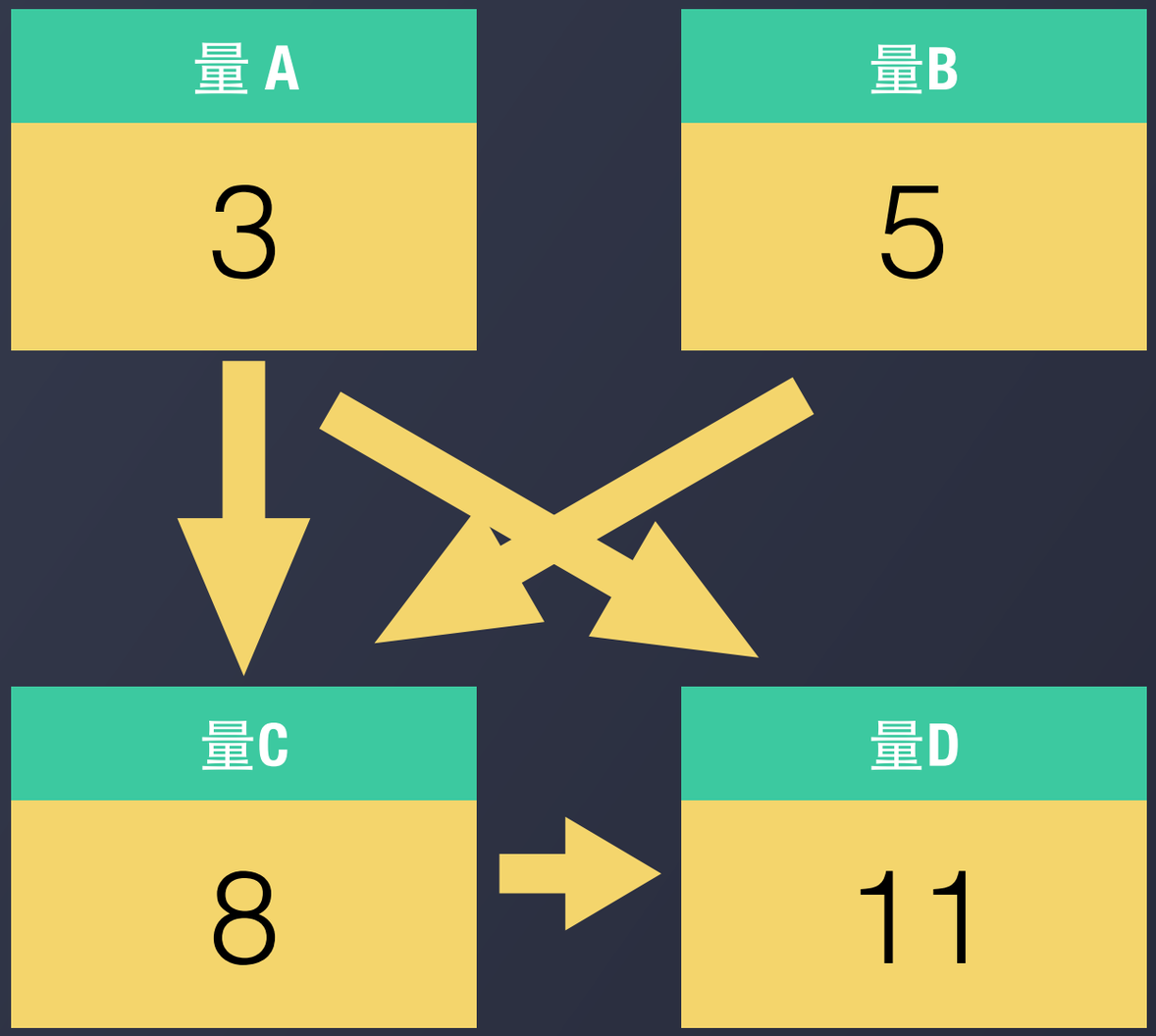

```
var a := 3
var b := 5
var c := a + b
var d := c + a
b = 8
```

```
func +(left: Value<Int>, right: Value<Int>) -> Value<Int> {
    return Value {
    }
}
```

```
var a = Value(5)                    Value<Int>
var b = Value(7)                    Value<Int>
var c = a + b                       Value<Int>

c.value                             12
a.value = 8                         Value<Int>
c.value                             15
b.value = 12                        Value<Int>
c.value                             20
c.value = 9                         Value<Int>
a = c + b                           Value<Int>
a.value                             21
```

PUSH DRIVER




```
var a := 3
var b := 5
var c := a + b
var d := c + a
b = 8
```

```
clas
func +(left: Value<Int>, right: Value<Int>) -> Value<Int>{
    var leftValue = left.value
```

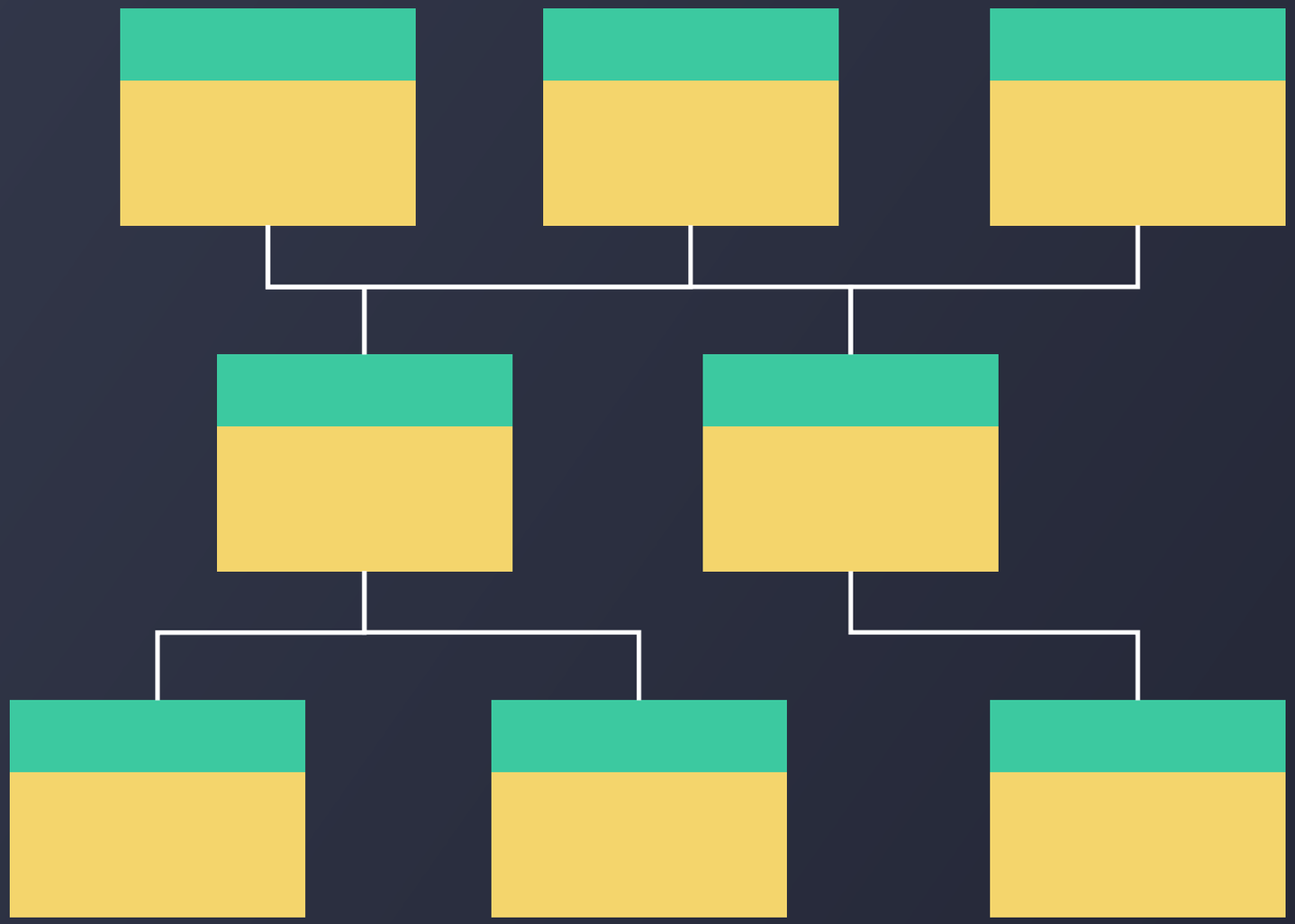
```
    var a = Value(3)                    Value<Int>
    var b = Value(5)                    Value<Int>
    var c = a + b                       Value<Int>

    c.observe {                         Value<Int>
        print("\tvalue: \($0)")         (4 times)
    }
    a.value = 8                         Value<Int>
    c.value                             13
    a.value = 12                        Value<Int>
    c.value                             17
    b.value = 9                         Value<Int>
```

```
    }
}
```

```
value: 8
value: 13
value: 17
value: 21
```

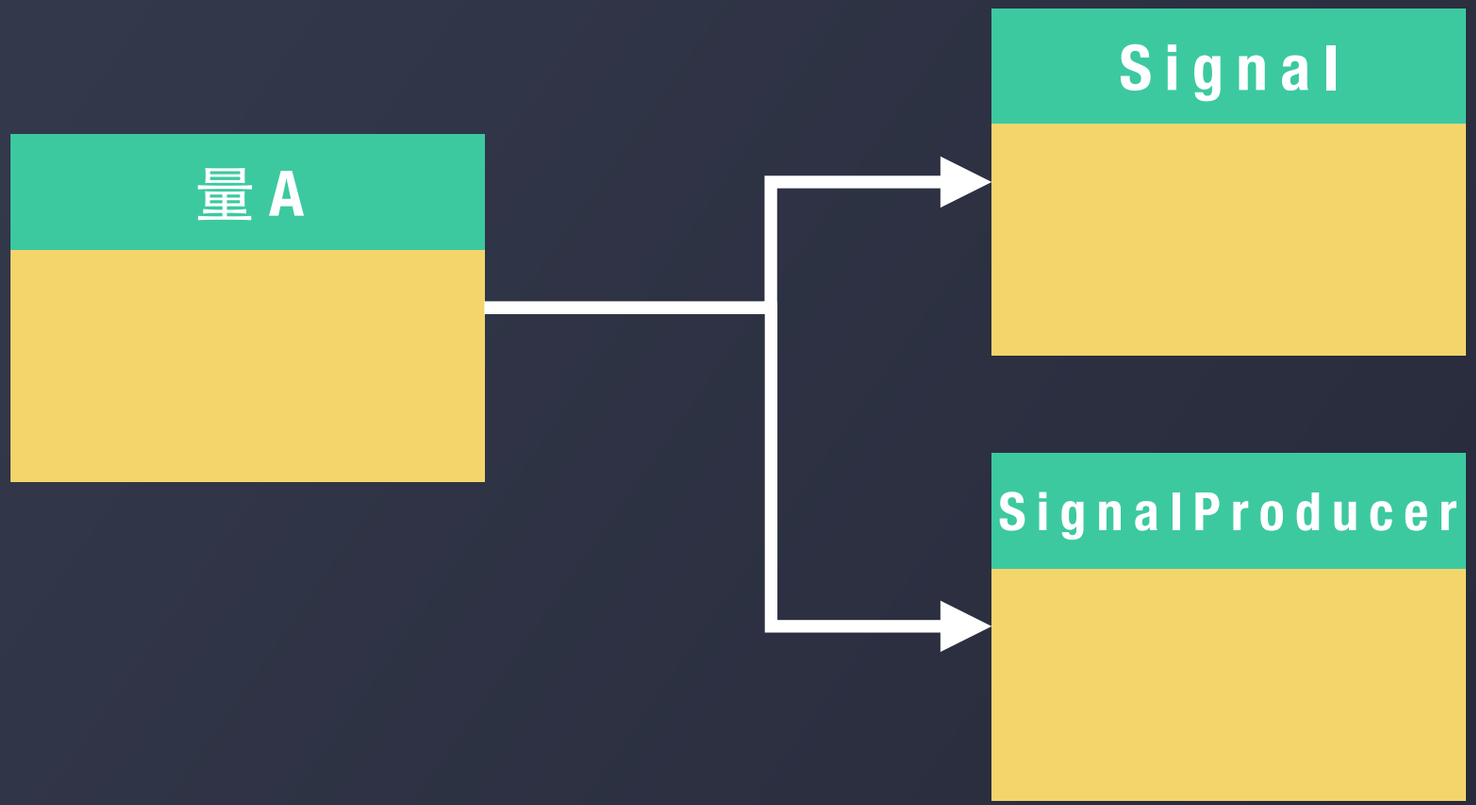
PULL VS PUSH



成熟的框架

Reactive Cocoa	RxSwift	Other
		

Value<T> VS ReactiveCocoa

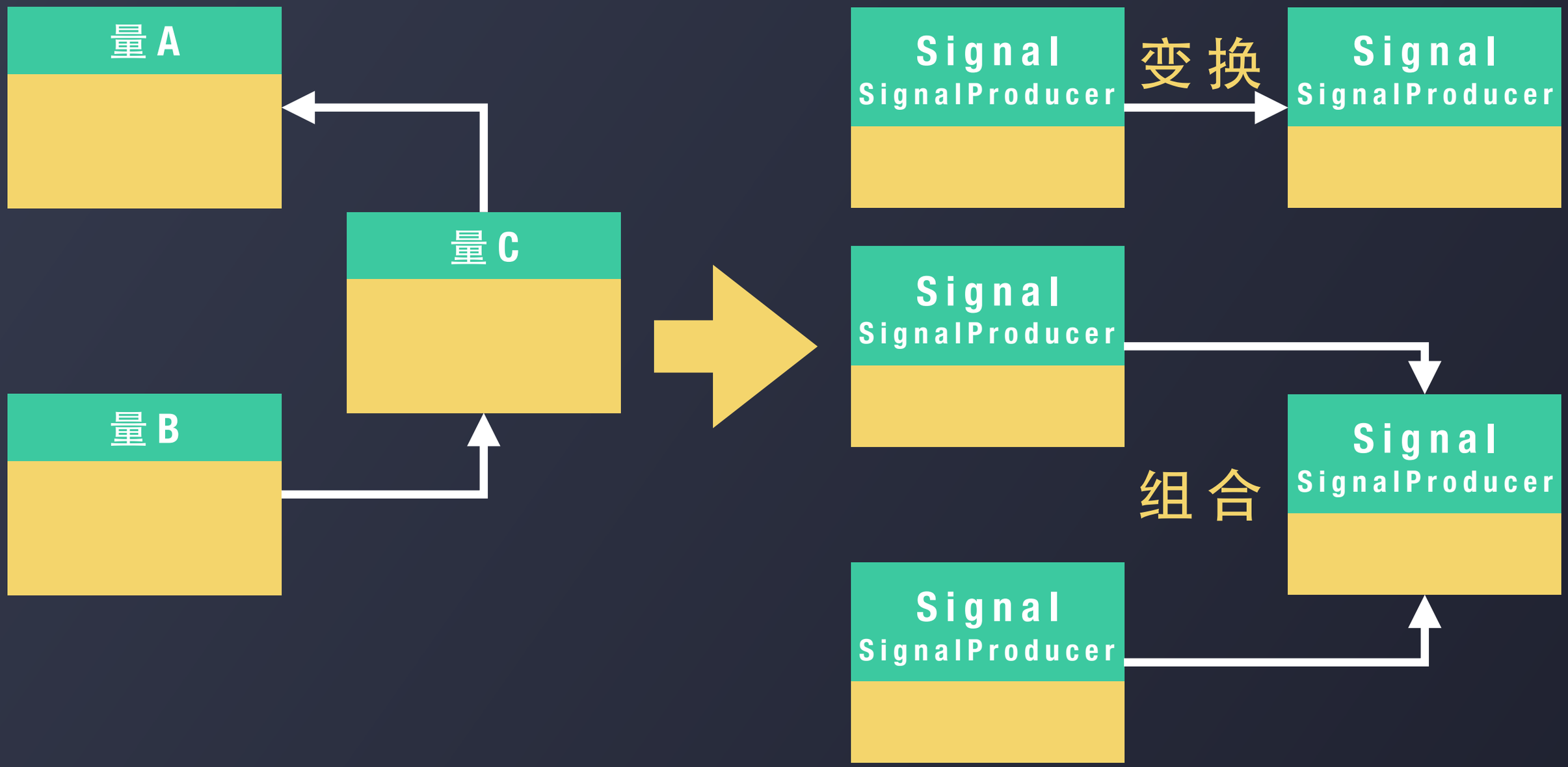


纯异步

初始化值

可取消

Value<T> VS ReactiveCocoa



实践响应式编程

两个请求并行处理

- 默认文案
- 某一个返回
- 两者都返回

分析过程

LABEL
默认

量A
nil

量B
nil

判断趋势

评估

是否适用

额外扩展

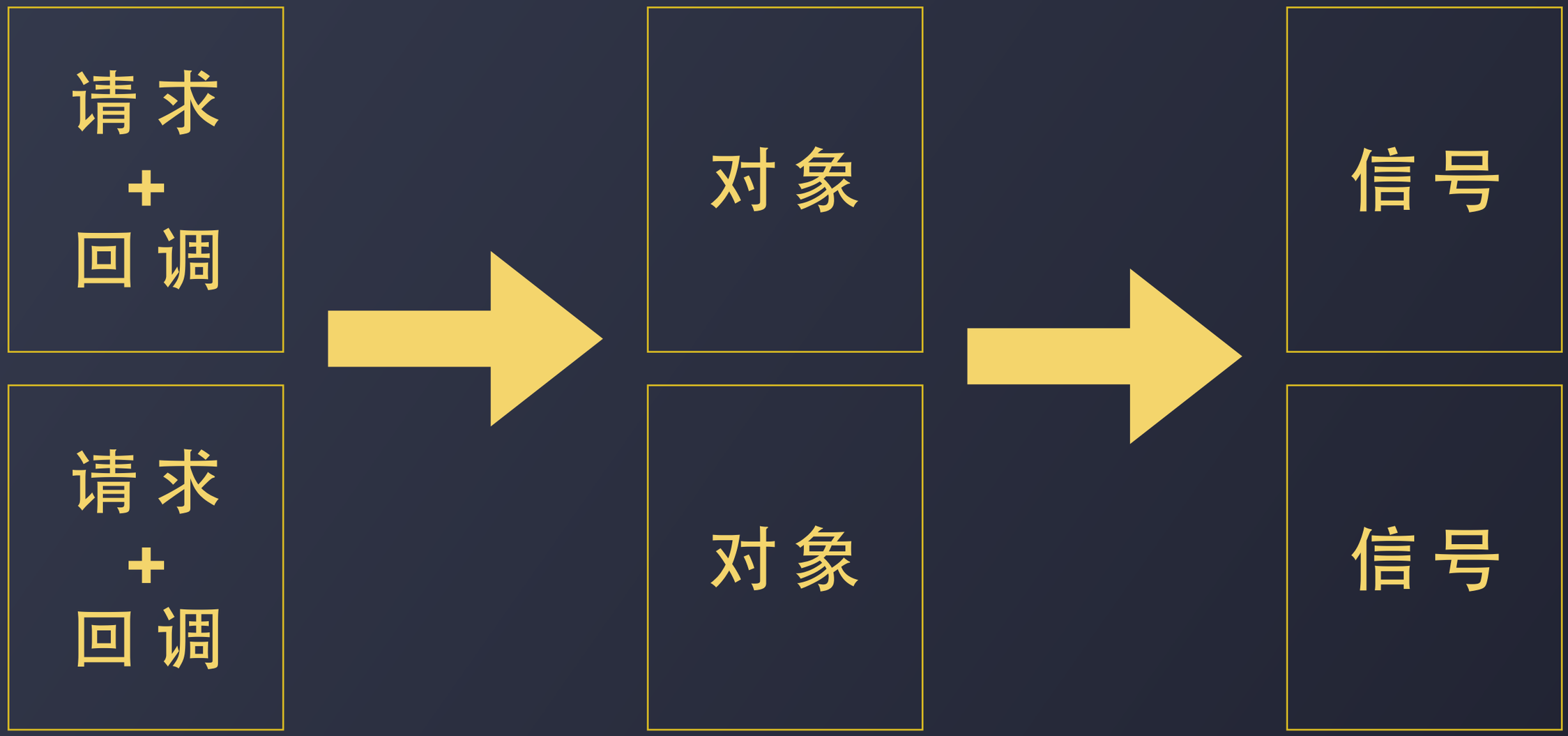
实践响应式编程

映射关系

可以表示未来的值

映射关系

信号的组合处理

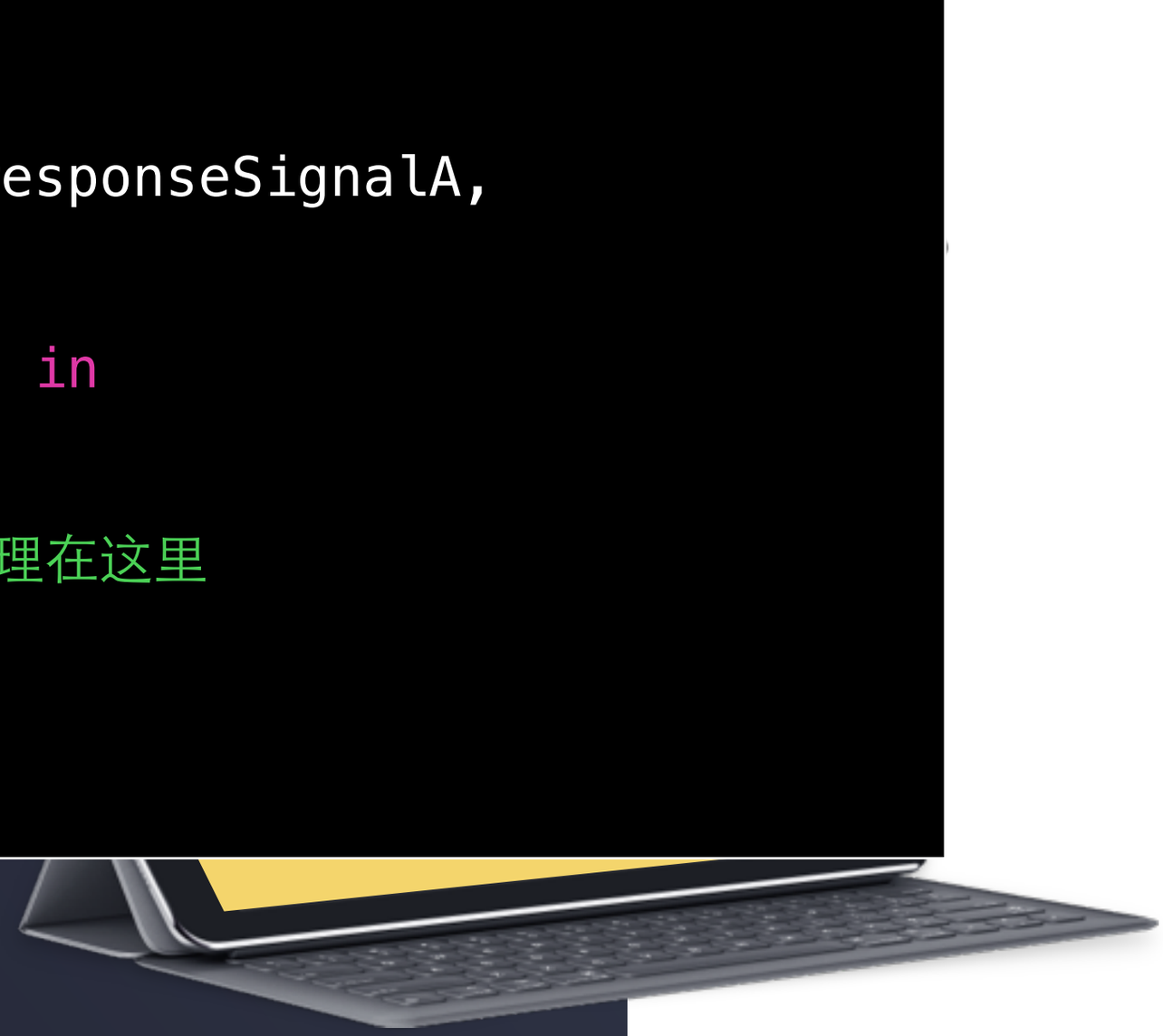


```swift
let requestA = URLRequest(url: baidu)
let requestB = URLRequest(url: google)

let responseSignalA = sessionManager.reactive.data(with:
requestA)
let responseSignalB = sessionManager.reactive.data(with:
requestB)

let responseSignalAll =
SignalProducer.combineLatest([responseSignalA,
responseSignalB])

responseSignalAll.start { event in
    switch event {
    case let .value(value):
        print("\(value)") // 处理在这里
    default:
        break
    }
}
```



实践响应式编程

遇到特殊CASE

绕开 & 记录

统计 & 评估

深挖 & 解决

一些建议

如何在团队中推广新技术

- 搞得清的人
- 愿意搞的伙伴
- 可以搞的项目

避免实践的极端

道听途说

因噎废食

归纳与总结

更高抽象的层次

理论与业务的结合

Q & A

THANKS