Introducing Compositional UI Styling

@takecian

今日は

- 関数型プログラミングで UI を設定する話

関数型プログラミング

関数を第一級オブジェクトとして扱う

第一級オブジェクト

その言語における基本的な操作を制限なしに使用できる対象

* 生成
* 代入
* 演算
*（引数・戻り値としての）受け渡し

多分こういうのとか

```
// 生成&代入
let handler: (Bool) -> Void = { isSuccess in
    if isSuccess {
        print("Yes!")
    }else{
        print("Nooo")
    }
}

// 受け渡し
someFunction(handler: handler)
```

第一級オブジェクト

その言語における基本的な操作を制限なしに使用できる対象

* 生成
* 代入
* 演算
*（引数・戻り値としての）受け渡し

演算？

関数を足したり引いたり？

他の言語にはあるらしい

Haskell, PureScript とか

演算用 operator

関数を結合する operator <> を定義してみる

```
infix operator<>: SingleTypeComposition
public func <> <A: AnyObject>(f: @escaping (A) -> Void, g: @escaping (A) -> Void) -> (A) -> Void {
    return { a in
        f(a)
        g(a)
    }
}
```

<>: 関数fと関数g を結合した関数を返す

何に使うの？

今回UIの設定に使ってみた

- backgroundColor
- cornerRadius

とか

もうちょっと具体的に

- UIView を受け取ってbackgroundColor を設定する関数
- UIView を受け取ってcornerRadius を設定する関数

を作って結合する

- UIView を受け取ってbackgroundColor を green にして、cornerRadius を 30 に設定する関数とか作れる

デモ

https://github.com/takecian/ComposableStyling

何が便利？

UI の設定を分離できる

- アプリでbackgroundColorを統一する時
- いくつかの View で cornerRadius の値を統一する時

参考

- https://www.pointfree.co/episodes/ep3-uikit-styling-with-functions
- https://www.youtube.com/watch?v=XJreRR0cC3E