- 但是这和咱们SwiftUI的MV模式有什么关系呢?

- 答：首先，MV模式是死的
- Model驱动View，实际是用户先修改Model的值，监听View的Model就变啦！
  就像一个空荡荡的口号，数据和界面，让数据驱动界面固然是非常美好的，可是这nm也太不现实了吧！我们的原数据（Outline）总不可能白花花地显示在界面上，它需要经过parser处理，也就是一堆函数的磨炼！
- 可是函数怎么调用，谁去调用，MV这个抽象的口号根本没有给出实际的解释

- 那Redux模式是活的吗?
- 下面是一个Reducer的实现～上面的两个case就是state的抽象（上框），而下面的就是根据Action和现state去调用函数，并返回未来的状态（下框）

```
enum KeyboardState {
    //行号和处于行的状态
    case startline(linenumber : Int)
    case inline(linenumber : Int)

// apply是一个状态机函数， 它根据旧状态，用户的行动，确定新的状态
func apply(inparser : Parser, thisaction : KeyboardAction)->KeyboardState{
    switch self {
    case .startline( _):
        switch thisaction {
        case .editinline(let linenumber,let newstring):
            // 这里随便写了parser的调用的函数
            inparser.chageOutline(lineid: linenumber, pageid: 0, newstring: newstring)
            inparser.drawSingleSlide()
            return .inline(linenumber: linenumber)
        default:
            inparser.drawSingleSlide()
            return .startline(linenumber: 0)
        }
    default:
        switch thisaction {
        case .deleteline(linenumber: 0):
            inparser.drawSingleSlide()
            return .startline(linenumber: 0)
        default:
```