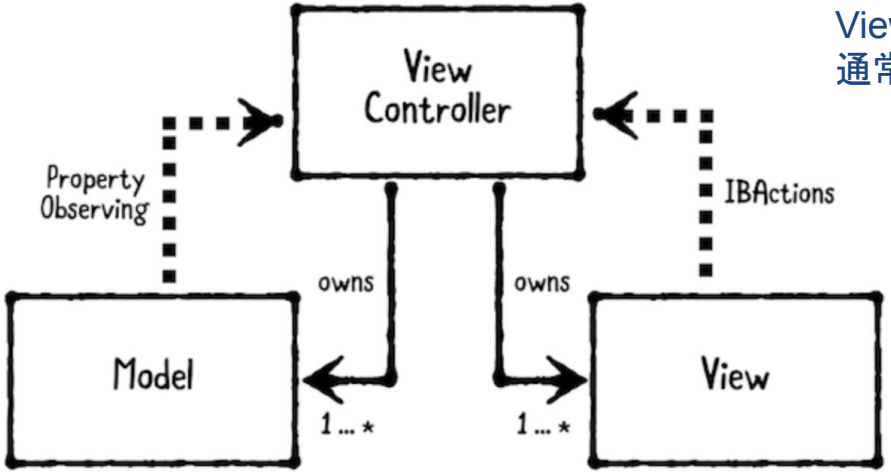
# MVVM pattern

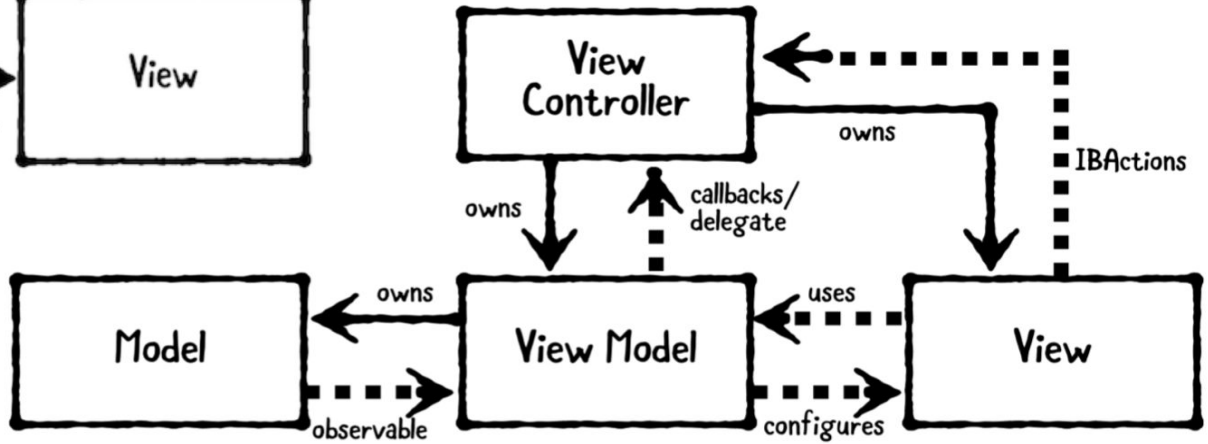
# APPLE MVC v.s. MVVM

Models: 用來保存資訊的結構，通常是 struct 或 class

Views: 在螢幕上顯示可見的元素或是控制，
通常是 UIView 或 subclass

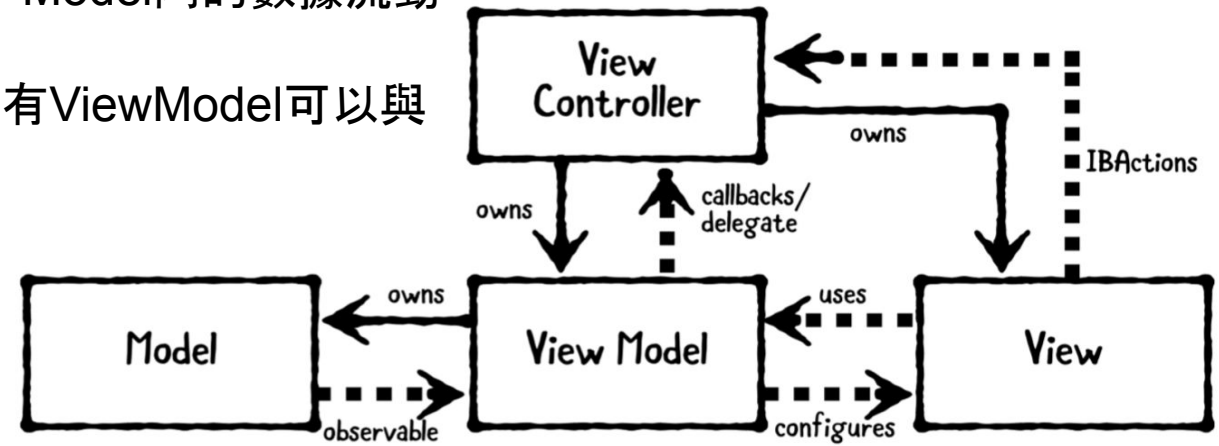



ViewModels: 轉換model的資訊為可讀的格式，使之可以顯示在 View上，通常是class類別，所以可 references 傳遞



# MVVM pattern

1. View / ViewController 負責將Model的資料顯示出來，也可以更新資料。
2. View / ViewController 只能透過ViewModel 與 Model 溝通。
3. ViewModel 為一中介者，Model內的數據流動都要通過它
4. Model為一儲存數據，只有ViewModel可以與Model溝通。



# When we should use it?

1. 如果你的app用到許多 model-to-view 的轉換，那使用 MVVM 是很不錯的，當然不是所有的物件都適合使用 MVVM，你可能需要配合使用其他的設計模式。

2. 開始寫一個新的 project 時就使用 MVVM 架構可能不是很有用，可以先試著使用 MVC，之後再根據不斷變化的需求來選擇相對應的設計模式，做到某個程度後發現真的需要 MVVM 架構，再導入也是可行的。要有計畫的去做，不要怕改變。

# Key points

1. MVVM 可幫 ViewController 瘦身，使之變得簡潔，不再是 ” Massive View Controller ”。
2. ViewModels 可以將原本的資料轉為View可直接使用的類型。比如說今天有個Model裡的資料money其別為Int,但View中的Label顯示資料為String，這時ViewModel就可以先行處理，將Int轉為String，這樣Label在顯示時就不用再轉型。
3. 如果你的viewModel只用在單一個View上，最好將所有的設置放在viewModel中，但如果你有使用多個View，則建議把各個View的設置做分離，以免造成混亂。
4. 當你的app規模還小時，還是以MVC架構為主，之後若需求有變動，再選用相對應的設計模式。