@CocoaTalk #1

Agenda

- What’s New in UIKit Animations (iOS 6.0 - 7.0)
- ReactiveCocoa
- SBMVC_1.0 => SBMVC_2.0

Demo #1 : iPad Application

UIKit Animations

- New UIKit API in CoreAnimation
- UICollectionView + UIKit Dynamics
- Controller transition animations

Core Animation 101

- Implicit Transition:

```
view.frame = (CGRect){.origin={10,10},.size={100,100}}
[CATransition begin]  …  [CATransition commit]
```

- Explicit Transition:

```
[UIView animationWithDuration:animation:completion:]
```

- 2D Transform:

```
                                                                  not in UIKit
struct CGAffineTransform {                      [a, b, 0]
  CGFloat a, b, c, d;                                          x = ax0 + cy0 +tx;
  CGFloat tx, ty;         => (x0,y0,1)* [c, d, 0]  =>    y = bx0 + dy0 +ty;
};                                              [tx,ty,1]
```

- 3D Transform:

```
   struct CATransform3D{
 CGFloat m11, m12, m13, m14;
 CGFloat m21, m22, m23, m24;  =>  m34?  =>  openGL Projection Mode
 CGFloat m31, m32, m33, m34;
CGFloat m41, m42, m43, m44;};
```

What’s New?

- SpringAnimation:

```
[UIView animateWithDuration:1.5 delay:0.0 usingSpringWithDamping:0.4
initialSpringVelocity:5.0 options:0 animations:^{

        imageView.frame = CGRectMake((h-400)/2, (w-400)/2, 400, 400);

    } completion:^(BOOL finished) {

    }];
```

- KeyFrameAnimation:

```
[UIView animateKeyframesWithDuration:duration delay: options: animations:^{

[UIView addKeyframeWithRelativeStartTime:0.0 relativeDuration:0.5 animations:^{…}];

[UIView addKeyframeWithRelativeStartTime:0.5 relativeDuration:0.5 animations:^{…}];

…}];
```

Demo #2:CoreAnimation

UIKitDynamics

- New in iOS 7.0
- An UIKit physical engine
- Migrate from Sprite Kit:Rewrite of Box2D
- Gravity, Collision, Snap, Attachment,…etc.
- 9.8 m/s2 <==> 1000 pt/s2

Demo #3：UIKitDynamics

UICollectionView

- New in iOS 6.0
- UITableView + layout
- TBCitySBCollectionViewController

Demo #4：UICollectionView

Controller Transition Animation

- Hooks API

```
- (id <UIViewControllerAnimatedTransitioning>)navigationController:
- animationControllerForOperation:
 fromViewController:
 toViewController:
```

- Quick Snapshot API

```
- (UIView *)snapshotViewAfterScreenUpdates:(BOOL)afterUpdates

-  (UIView *)resizableSnapshotViewFromRect:(CGRect)rect afterScreenUpdates:
  (BOOL)afterUpdates withCapInsets:(UIEdgeInsets)capInsets

-  (BOOL)drawViewHierarchyInRect:(CGRect)rect afterScreenUpdates:(BOOL)afterUpdate
```

Best Practice

- Deep understanding of Core Animation/Quartz 2D
- Be careful with the “context”
- Don’t mess with UIKitDynamics
- if “screen update” is tricky ,try “renderInContext”

Demo #5：Transition

ReactiveCocoa Overview

- Github(Windows) -> Reactive Extension(RX) -> ReactiveCocoa (RAC)
- Design Philosophy : Functional Programming + Reactive = FRP
- Many concept comes from Haskell (x:xs,tuple,zip,zipwith…)
- It’s like a functional version of Cocoa Touch
- Block based
- Hard to debug
- Sometimes a little tricky :(

Functional Programming

- imperative v.s Functional
    - F (Input) = output
- Languages:
    - Restricted : Lisp, Haskell (without IO/Monad)
    - Wide :Haskell, Erlang, Schema, Scala, Lua, Python, Javascript…
- Features:
    - First-Class Object -> Function
    - High Order Function : f(x) = 2x^2+1, g(x) = x+1, f(g(x)) = 2(x+1)^2+1
    - Map- > Filter -> Reduce(Fold)
    - 代数与求值
    - Lambda-calculas, Currying, Monad…

<Learn you a Haskell for great good >

<Functional Programming Principles in Scala>

Demo #6：Haskell

- High Order Function
- map -> filter -> reduce
- Lambda-calculus,Left fold, zipWith…

Reactive Programming

- Reactive to stimulation
- Consider the runloop on main thread: button, touch…
- a = 3, b = a+1, a=4, b = ?
- Hot signal v.s. Cold signal

<Reactive Programming in Scala>

Functional Style in OC

First Class:

HOF

```
[self doBlock:^id(id p) {
        return p;
}];
```

Enumerate:

HOF

```
NSArray* list = @[@(1),@(2),@(3)];

[list enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) { … }];
```

Filter:

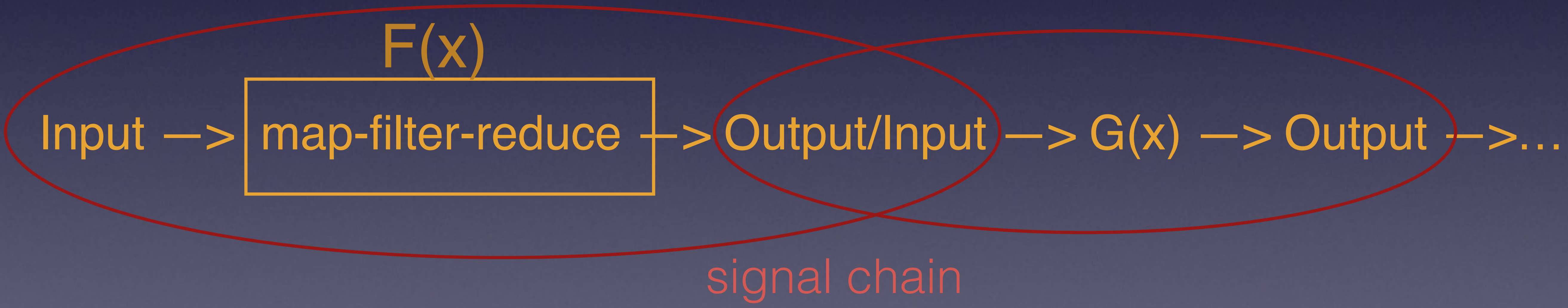
```
NSArray* list = @[@(1),@(2),@(3)];

NSArray* subList = [list filteredArrayUsingPredicate:[NSPredicate
predicateWithBlock:^BOOL(NSNumber* evaluatedObject, NSDictionary *bindings) {

        return evaluatedObject.intValue > 2;
}]];
```

Concept of ReactiveCocoa

- Event: stream, signal, sequence
- Event pipeline： 代数与求值


F(x)
Input —> map-filter-reduce —> Output/Input —> G(x) —> Output —>…
signal chain

Demo #7 : ReactiveCocoa

1, signal chain

2, binding

Why RAC?

side effect: local var

- Reduce side effect:

```
cancelBtn.rac_command = [[RACCommand alloc]initWithSignalBlock:^RACSignal *(id
input) {
        [searchBar resignFirstResponder];
        return [RACSignal empty];
    }];
```

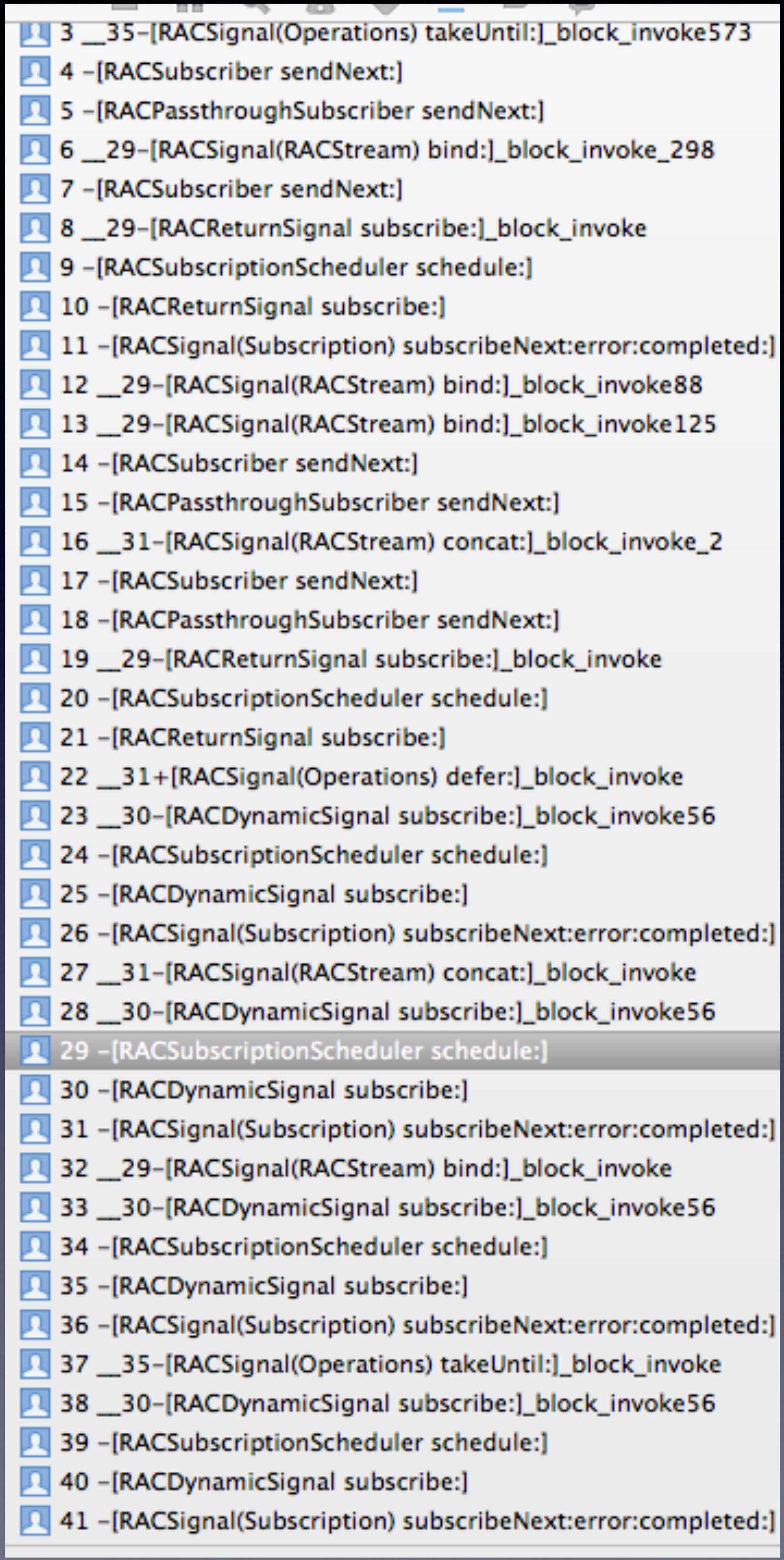
- Binding:

```
RAC(self,totalPriceLabel.text) = [RACObserve(self.tableViewVM, totalPrice)
map:^id(NSNumber* value) {
        return [NSString stringWithFormat:@"%.1f",value.floatValue];
    }];
```

- Dealing with list:

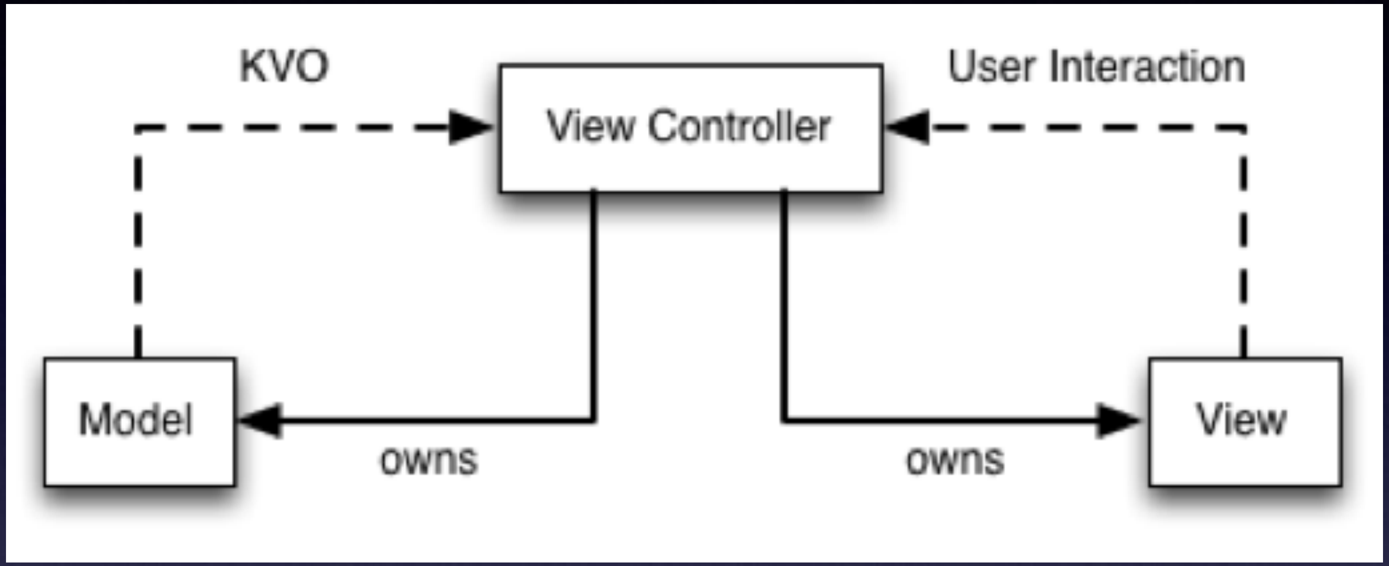
```
map -> filter -> reduce
```

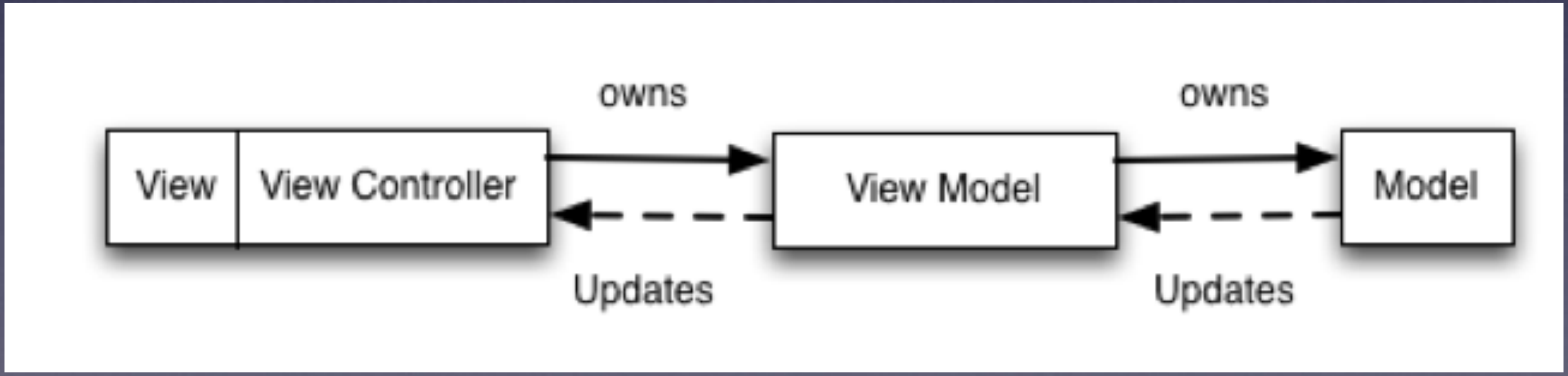
- Extremely useful in some cases
- Going fully functional could make things worse
- Productivity v.s. code efficiency



```
3 __35-[RACSignal(Operations) takeUntil:]_block_invoke573
4 -[RACSubscriber sendNext:]
5 -[RACPassthroughSubscriber sendNext:]
6 __29-[RACSignal(RACStream) bind:]_block_invoke_298
7 -[RACSubscriber sendNext:]
8 __29-[RACReturnSignal subscribe:]_block_invoke
9 -[RACSubscriptionScheduler schedule:]
10 -[RACReturnSignal subscribe:]
11 -[RACSignal(Subscription) subscribeNext:error:completed:]
12 __29-[RACSignal(RACStream) bind:]_block_invoke88
13 __29-[RACSignal(RACStream) bind:]_block_invoke125
14 -[RACSubscriber sendNext:]
15 -[RACPassthroughSubscriber sendNext:]
16 __31-[RACSignal(RACStream) concat:]_block_invoke_2
17 -[RACSubscriber sendNext:]
18 -[RACPassthroughSubscriber sendNext:]
19 __29-[RACReturnSignal subscribe:]_block_invoke
20 -[RACSubscriptionScheduler schedule:]
21 -[RACReturnSignal subscribe:]
22 __31+[RACSignal(Operations) defer:]_block_invoke
23 __30-[RACDynamicSignal subscribe:]_block_invoke56
24 -[RACSubscriptionScheduler schedule:]
25 -[RACDynamicSignal subscribe:]
26 -[RACSignal(Subscription) subscribeNext:error:completed:]
27 __31-[RACSignal(RACStream) concat:]_block_invoke
28 __30-[RACDynamicSignal subscribe:]_block_invoke56
29 -[RACSubscriptionScheduler schedule:]
30 -[RACDynamicSignal subscribe:]
31 -[RACSignal(Subscription) subscribeNext:error:completed:]
32 __29-[RACSignal(RACStream) bind:]_block_invoke
33 __30-[RACDynamicSignal subscribe:]_block_invoke56
34 -[RACSubscriptionScheduler schedule:]
35 -[RACDynamicSignal subscribe:]
36 -[RACSignal(Subscription) subscribeNext:error:completed:]
37 __35-[RACSignal(Operations) takeUntil:]_block_invoke
38 __30-[RACDynamicSignal subscribe:]_block_invoke56
39 -[RACSubscriptionScheduler schedule:]
40 -[RACDynamicSignal subscribe:]
41 -[RACSignal(Subscription) subscribeNext:error:completed:]
```

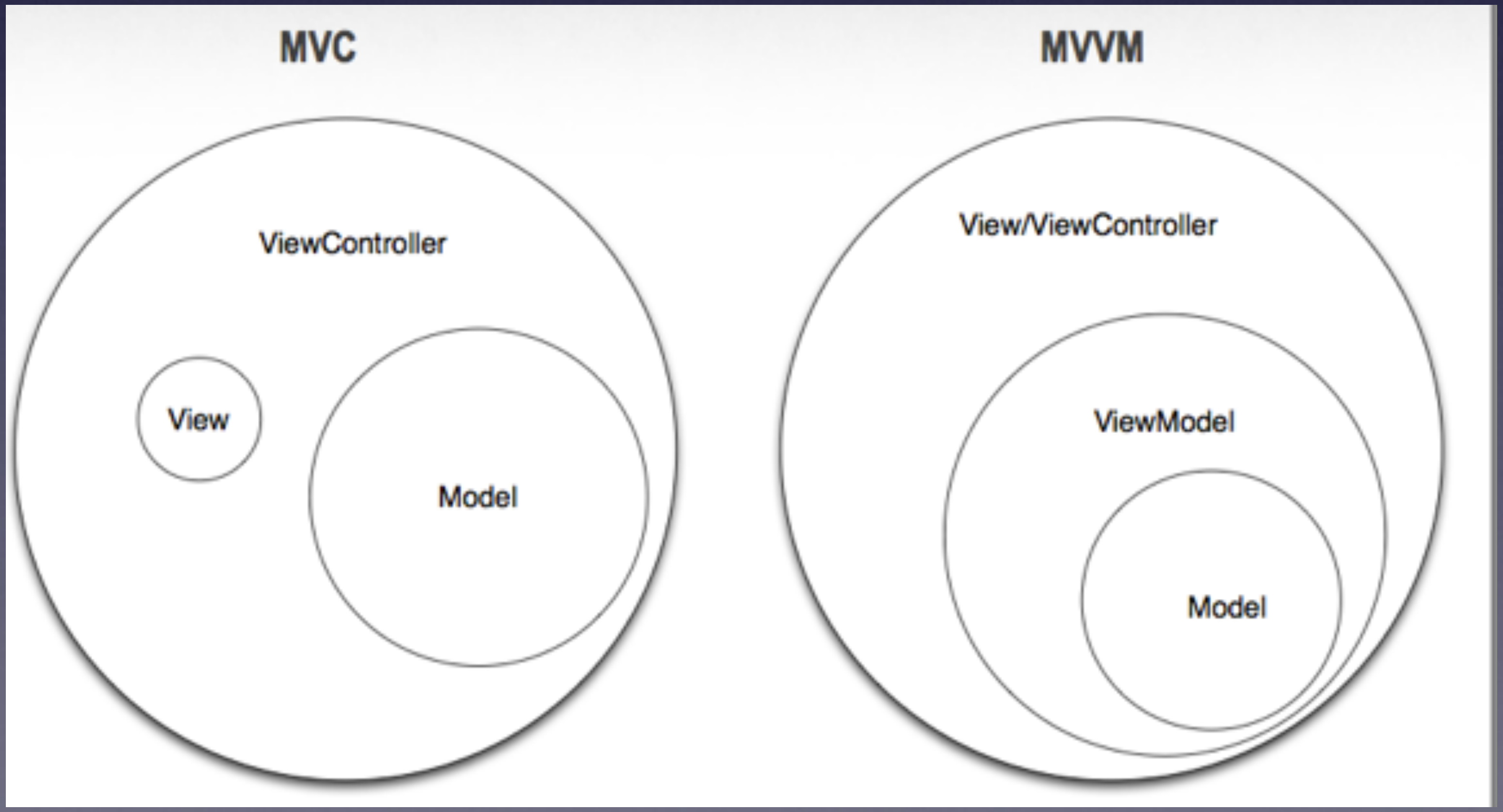
MVC -> MVVM


KVO
View Controller
User Interaction
Model
owns
owns
View


owns
owns
View
View Controller
View Model
Model
Updates
Updates

ViewModel

- ViewModel is highly abstracted
- ViewModel can be tested
- ViewModel can be cross platform



Demo #8 : ViewModel + binding

Demo #8 : LLDB + Python

```
(lldb) command script import ~/my_script.py

(lldb) breakpoint command add -F my.breakpoint_func
```