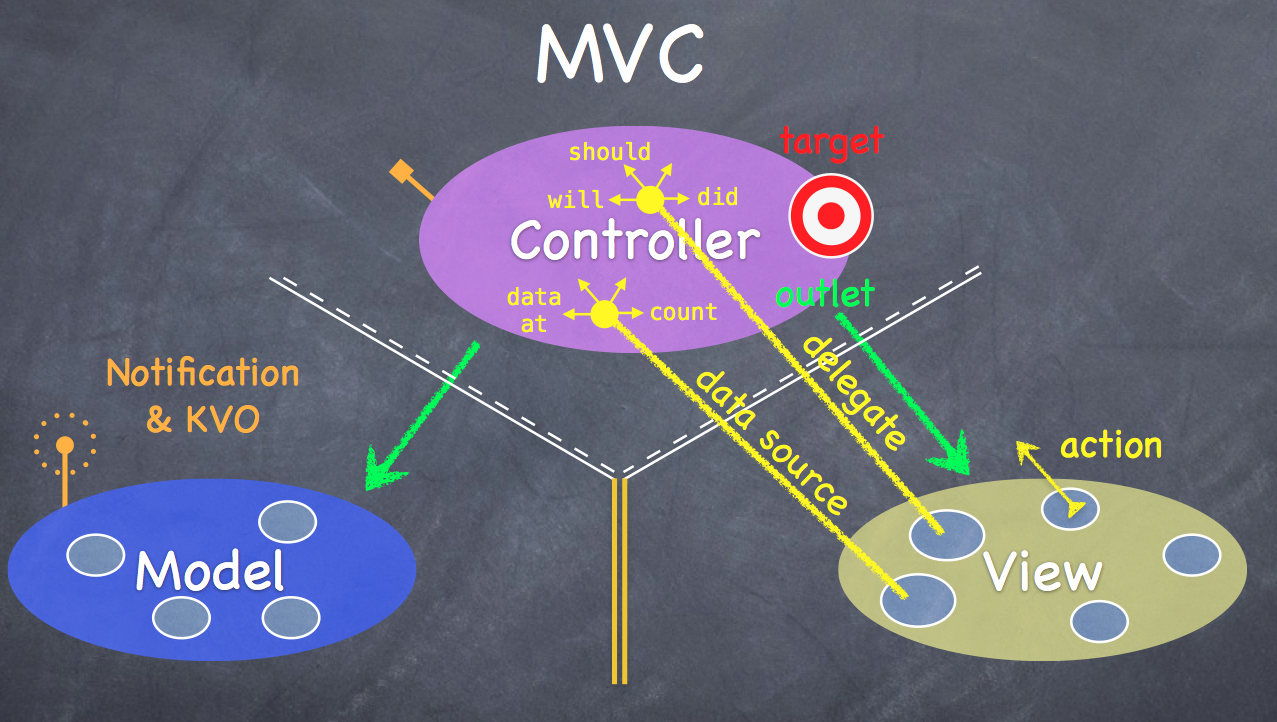
**Lecture 1**



**Model** - what an app is [but not how it is displayed].

**Controller** - How Model is presented to the user [UI logic].

**View** - Controller’s minions.

**Controllers** 🡪 **View**.

**View** 🡪 **Controller [2 ways]**:

**Target-Action [button tap]:** The **View** sends the action to the **Controller** [View’s target] when action happens in the UI.

**Data-source delegation**: The Controller [View’s delegate]. Views do not own the data they display – data are acquired via a protocol. Controllers are almost always that data source [not Model!]. Controllers interpret/format Model information for the View.

**Controllers** 🡪 **Model**.

**Model x Controller [**The Model is UI independent]. If the Model has info to update, it uses **notifications**.

**Model x View**.

**View x Model.**

**Objective-C**

No multiple inheritance.

Contents of the superclass header file is **public**.

Import **Foundation** [NSDictionary, NSArray, NSStrings].

All objects live on the **heap**. Objects are always pointed to.

**No garbage collection** – use reference counting [When count goes to zero, - clean up the object from the heap].

**Encapsulation**: Use private properties but not private methods [setter, getter]. Setting and getting the pointer.

**Strong property** – objects stays on the heap as long as it is pointed to.

**Weak property –** only keep it on the heap if someone else points to it strongly.

**Nonatomic** – the property is not a thread safe.

**Do not initialize** random properties – it start out at 0.

**Card Game:**

**Card.h/Card.m** - model

**Card.h**

@property (strong, nonatomic) NSString \*contents;

**Card.m [behind the scenes]:**

@synthesize contents = \_contents;

// Getter method.

- (NSString \*)contents

{

return \_contents;

}

// Setter method.

- (void)setContents:(NSString \*)contents

{

\_contents = contents;

}