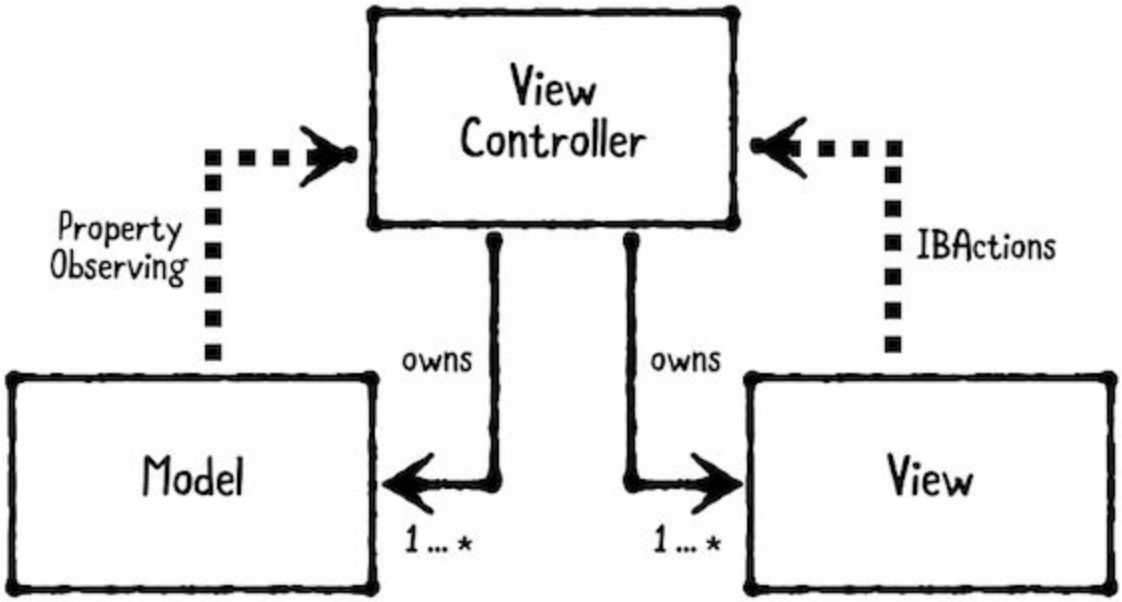
Fundamental Design Patterns

Patterns are frequently used throughout iOS development

Model-View-Controller Pattern
Delegation Pattern
Strategy Pattern
Singleton Pattern
Memento Pattern
Observer Pattern
Builder Pattern

Model-view-controller (MVC) [Structural pattern]

MVC pattern separates objects into three distinct types. are models, views and controllers!



Models hold application data. They are usually structs or simple classes.

Views display visual elements and controls on screen. They are usually subclasses of UIView.

Controllers coordinate between models and views. They are usually subclasses of UIViewController.

MVC is very common in iOS programming because it's the design pattern that Apple chose to adopt in UIKit.

Controllers are allowed to have strong properties for their model and view so they can be accessed directly. Controllers may also have more than one model and/or View.

Conversely, models and views **should not hold a strong** reference to their owning controller. This would cause a retain cycle.

Instead, models communicate to their controller via property observing (KVO), and views communicate to their controller via IBActions.

This lets you reuse models and views between several controllers

Note: Views may have a weak reference to their owning controller through a delegate (see “Delegation Pattern”).

Limitations of MVC
Controllers are much harder to reuse since their logic is often very specific to whatever task they are doing. Consequently, MVC doesn’t try to reuse them.

MVC is a good starting point, but it has limitations. Not every object will neatly fit into the category of model, view or controller. Consequently, applications that only use MVC tend to have a lot of logic in the controllers. This can result in view controllers getting very big! There's a rather quaint term for when this happens, called "**Massive View Controller**."

To solve this issue, you should introduce other design patterns as your app requires them.

When should you use it?
Use this pattern as a starting point for creating iOS apps. In nearly every app, you’ll likely need additional patterns besides MVC, but it’s okay to introduce more patterns as your app requires them.

Tutorial project

- Models we have: Question and QuestionGroup. View we have: QuestionView. Controller we have: QuestionViewController.
- QuestionViewController owns QuestionGroup (Model). QuestionViewController update QuestionView as per model QuestionGroup in method showQuestion()
- QuestionViewController update ui and update model in IBAction handleCorrect(_), handleIncorrect(_) and showNextQuestion()