


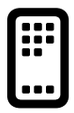


# Delegate Pattern

# Delegate Pattern in iOS

- ✅ Fundamental Design Pattern
- ✅ Allows object communication
- ✅ Promotes loose coupling
- ✅ Used in UIKit and Foundation

# Key Concepts of Delegates

- ✅ Delegation - Object hands off responsibility
- ✅ Protocols - Defines delegate methods
- ✅ Weak References - Prevents retain cycles
- ✅ Optional/Required Methods - Flexibility in implementation

# Common Use Cases for Delegates

- UITableViewDelegate - Handling table view interactions
- UITextFieldDelegate - Managing text input and editing
- URLSessionDelegate - Handling network request events
- CLLocationManagerDelegate - Location Updates
- WKNavigationDelegate - Managing Navigation Requests

# Implementing a Custom Delegate

- ✓ Define a protocol with required/optional methods
- ✓ Create a weak methods at appropriate times
- ✓ Call delegate methods at appropriate times
- ✓ Implement delegate methods in the delegate class

# Delegates vs Closures vs Notifications vs KVO

| Delegates | Closures | Notifications | KVO |
| --- | --- | --- | --- |
| One-to-one Communication | Simple inline & single use callbacks | One-to-many communication | One-to-many property observation |
| Strong type checking | Capture context easily | Broadcast events widely | Automatic updates on property changes |
| Multiple Callback methods | More flexible and concise | Useful for system-wide events | Decouples objects from property changes |
| Used for complex interfaces like UITableViewDelegate | Example: Completion Handlers | Example: Keyboard show/hide | Example: Observing model changes |