Swift Study 09

2017. 01.21

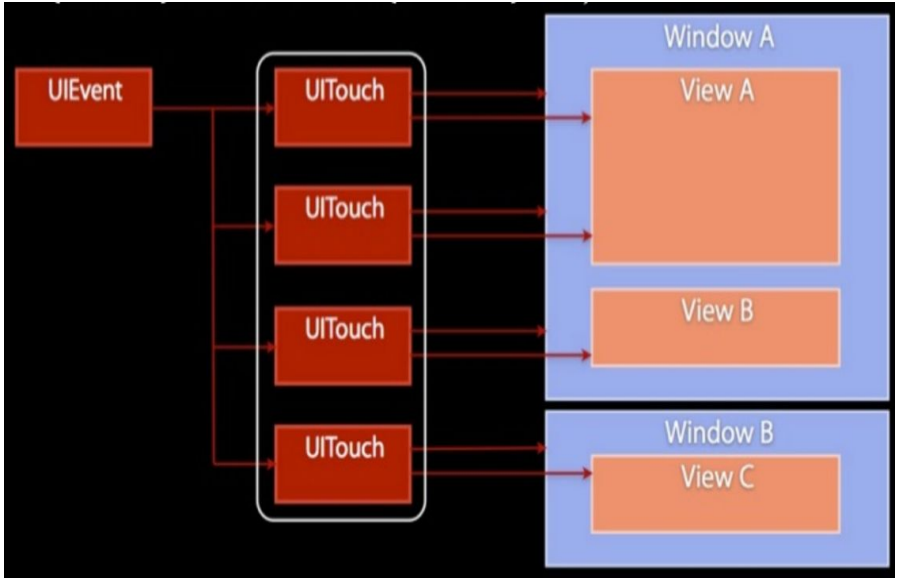
Swift ios

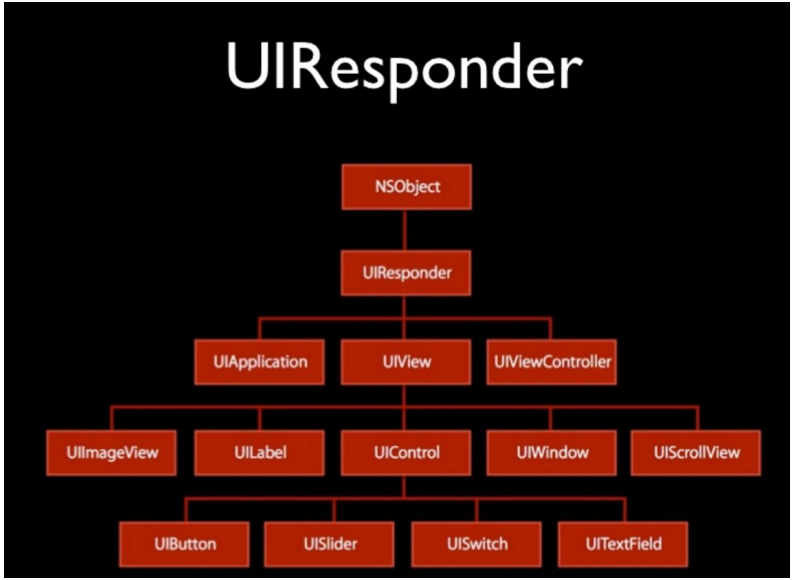
- Touch (UIResponder)

- Core Graphics

Touch (UIResponder)

- UITapGestureRecognizer 또는 UIViewController의 UIResponder로 터치기능 구현
- UIResponder는 기본적으로 touch / motion 두가지 이벤트 기능의 인터페이스 제공





Touch (UIResponder)

터치 이벤트에 응답하기

```
func touchesBegan(Set<UITouch>, with: UIEvent?)
```
하나 이상의 손가락이보기 또는 창에서 터치 다운되면 응답자에게 알립니다.

```
func touchesMoved(Set<UITouch>, with: UIEvent?)
```
이벤트와 연결된 하나 이상의 손가락이보기 또는 창 내에서 이동하면 응답자에게 알립니다.

```
func touchesEnded(Set<UITouch>, with: UIEvent?)
```
하나 이상의 손가락이 뷰 또는 창에서 들어올 때 응답자에게 알립니다.

```
func touchesCancelled(Set<UITouch>, with: UIEvent?)
```
시스템 이벤트 (예 : 메모리 부족 경고)가 터치 이벤트를 취소하면 응답자에게 보냅니다.

```
func touchesEstimatedPropertiesUpdated(Set<UITouch>)
```
터치에 대한 예상 속성이 변경되어 더 이상 예상치가 업데이트되지 않거나 업데이트가 더 이상 필요하지 않을 때 응답자에게 보냅니다.

Motion (UIResponder)

모션 이벤트에 응답

```
func motionBegan(UIEventSubtype, with: UIEvent?)
```

모션 이벤트가 시작되었음을 수신기에 알립니다.

```
func motionEnded(UIEventSubtype, with: UIEvent?)
```

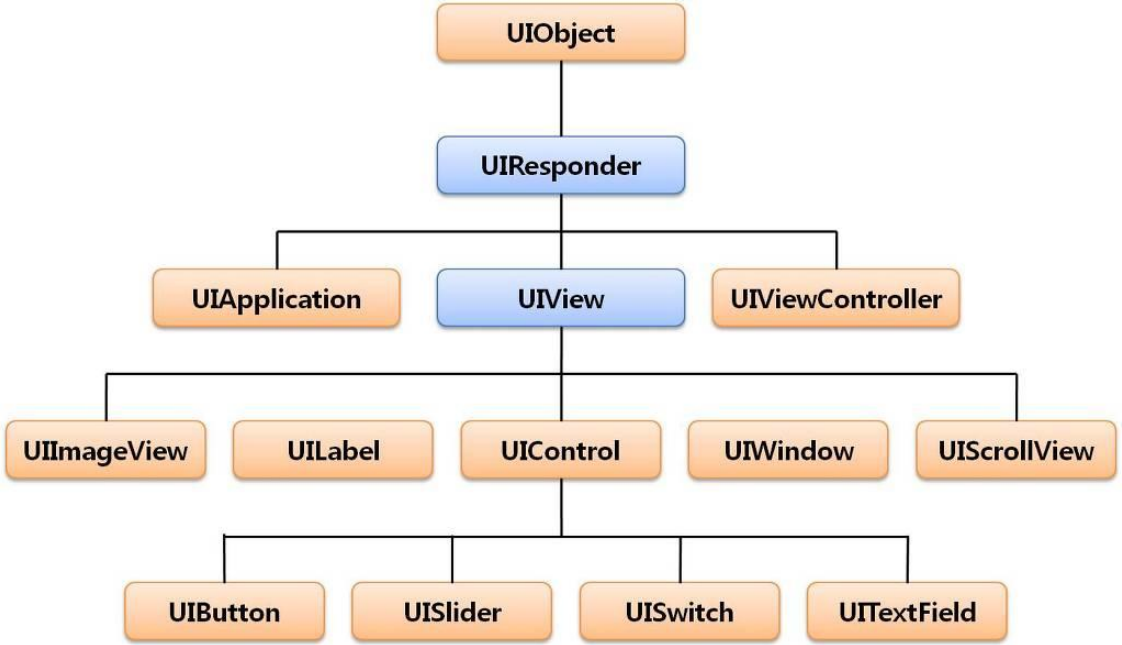
수신기에 모션 이벤트가 종료되었음을 알립니다.

```
func motionCancelled(UIEventSubtype, with: UIEvent?)
```

수신기에 모션 이벤트가 취소되었음을 알립니다.

UIResponder

- UIApplication, UIView, UIWindow의 슈퍼클래스
- 위 클래스들의 인스턴스에 대한 이벤트/응답객체



UIResponder

```
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
```

```
@available(iOS 2.0, *)
open class UIViewController : UIResponder, NSCoding, UIAppearanceContainer,
    UITraitEnvironment, UIContentContainer, UIFocusEnvironment {
```

```
@available(iOS 2.0, *)
open class UIView : UIResponder, NSCoding, UIAppearance, UIAppearanceContainer,
    UIDynamicItem, UITraitEnvironment, UICoordinateSpace, UIFocusItem,
    CALayerDelegate {
```

UIResponder

Responder Chain 관리

```
var next: UIResponder?
```
수신기 '????의 다음 응답자를 돌려 `nil`경우 아무 것도 없다.

```
var isFirstResponder: Bool
```
수신자가 첫 번째 응답자인지 여부를 나타내는 부울 값을 반환합니다.

```
var canBecomeFirstResponder: Bool
```
리시버가 최초 응답자가 될지 어떨지를 나타내는 불리언 값을 돌려줍니다.

```
func becomeFirstResponder()
```
수신기가 창에서 첫 번째 응답자가 될 것임을 수신기에 알립니다.

```
var canResignFirstResponder: Bool
```
리시버가 first-responder 상태를 기꺼이 포기할 지 여부를 나타내는 부울 값을 반환합니다.

```
func resignFirstResponder()
```
수신자에게 창에서 첫 번째 응답자로 상태를 양도하도록 요청되었음을 알립니다.

UIResponder

입력보기 관리

```
var inputView: UIView?
```
리시버가 첫 번째 응답자가 될 때 표시 할 사용자 정의 입력보기입니다.

```
var inputViewController: UIInputViewController?
```
수신기가 첫 번째 응답자가 될 때 사용할 맞춤 입력보기 컨트롤러입니다.

```
var inputAccessoryView: UIView?
```
수신기가 첫 번째 응답자가 될 때 표시 할 사용자 정의 입력 액세서리보기입니다.

```
var inputAccessoryViewController: UIInputViewController?
```
수신기가 첫 번째 응답자가 될 때 표시 할 사용자 정의 입력 액세서리보기 컨트롤러입니다.

```
func reloadInputViews()
```
객체가 첫 번째 응답자 일 때 사용자 정의 입력 및 액세서리보기를 업데이트합니다.

UIResponder

보도 자료에 대한 응답

일반적으로 커스텀 프레스 처리를 수행하는 모든 응답자는이 네 가지 방법 모두를 무시해야합니다. 받는 사람 각각의 호출 `pressesBegan(_:with:)`방법, 당신 응답자의 `pressesEnded(_:with:)`또는 `pressesCancelled(_:with:)`방법이라고합니다. 그만큼`pressesChanged(_:with:)`

```
func pressesBegan(Set<UIPress>, with: UIPressesEvent?)
```
관련 뷰에서 물리적 버튼을 누르면 수신자에게 보냅니다.

```
func pressesCancelled(Set<UIPress>, with: UIPressesEven
t?)
```
시스템 이벤트 (예 : 메모리 부족 경고)가 프레스 이벤트를 취소 할 때 수신기로 보냅니다.

```
func pressesChanged(Set<UIPress>, with: UIPressesEvent?)
```
때 수신기에 전송 `force`언론이 관련보기에서 변경되었습니다.

```
func pressesEnded(Set<UIPress>, with: UIPressesEvent?)
```
연결된보기에서 버튼을 놓을 때 수신기로 보냅니다.

UIResponder

원격 제어 이벤트에 대한 응답

```
func remoteControlReceived(with: UIEvent?)
```

원격 제어 이벤트가 수신되면 수신자에게 전송됩니다.

실행 취소 관리자 가져 오기

```
var undoManager: UndoManager?
```

응답자 체인에서 가장 가까운 공유 실행 취소 관리자를 반환합니다.

UIResponder

텍스트 입력 모드 관리

```
var textInputMode: UITextInputMode?
```
이 리스폰 더 객체의 텍스트 입력 모드입니다.

```
var textInputContextIdentifier: String?
```
응답자가 텍스트 입력 모드 정보를 유지해야한다는 것을 나타내는 식별자.

```
class func clearTextInputContextIdentifier(String)
```
앱의 사용자 기본값에서 텍스트 입력 모드 정보를 지 웁니다.

```
var inputAssistantItem: UITextInputAssistantItem
```
키보드의 바로 가기 바를 구성 할 때 사용할 입력 보조 장치입니다.

UIResponder

유효성 검사 명령

```
func canPerformAction(Selector, withSender: Any?)
```

수신 응답자에게 사용자 인터페이스에서 지정된 명령을 사용하거나 사용하지 않도록 요청합니다.

```
func target(forAction: Selector, withSender: Any?)
```

액션에 응답하는 대상 객체를 반환합니다.

사용 가능한 키 명령 액세스

```
var keyCommands: [UIKeyCommand]?
```

이 응답자에서 작업을 트리거하는 주요 명령입니다.

UIResponder

사용자 활동 지원

```
var userActivity: NSUserActivity?
```

이 응답자가 지원하는 사용자 작업을 캡슐화하는 객체입니다.

```
func restoreUserActivityState(NSUserActivity)
```

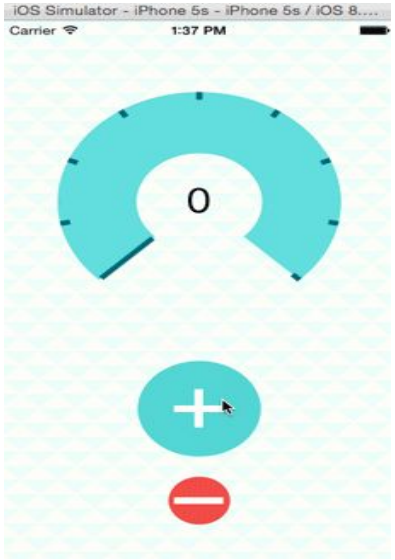
지정된 사용자 활동을 계속하는 데 필요한 상태를 복원합니다.

```
func updateUserActivityState(NSUserActivity)
```

지정된 사용자 활동의 상태를 업데이트합니다.

Core Graphics

- 쿼츠 (Quartz) 기술의 힘을 이용하여 고해상도 출력으로 가벼운 2D 렌더링
- 경로 기반 드로잉, 앤티 앨리어싱 렌더링, 그라디언트, 이미지, 색상 관리, PDF 문서 등 이용



Core Graphics

기하 데이터 유형

`CGFloat`

Core Graphics 및 관련 프레임 워크에서 부동 소수점 스칼라 값의 기본 유형입니다.

`CGPoint`

2 차원 좌표계에 점을 포함하는 구조체입니다.

`CGSize`

너비와 높이 값을 포함하는 구조체입니다.

`CGRect`

사각형의 위치와 크기가 포함 된 구조체입니다.

`CGVector`

2 차원 벡터를 포함하는 구조체입니다.

`CGAffineTransform`

2D 그래픽을 그리는 데 사용하는 아핀 변환 행렬입니다.

Core Graphics

2D 도면

`CGContext`

Quartz 2D 드로잉 환경.

`CGImage`

비트 맵 이미지 또는 이미지 마스크입니다.

`CGPath`

변경 불가능한 그래픽 경로 : 그래픽 컨텍스트에서 그려지는 모양이나 선을 수학적으로 설명합니다.

`CGMutablePath`

변경 가능한 그래픽 경로 : 그래픽 컨텍스트에서 그려지는 모양이나 선을 수학적으로 설명합니다.

`CGLayer`

Core Graphics로 그려진 콘텐트를 재사용하기위한 오프 스크린 컨텍스트.

Core Graphics

색상 및 글꼴

`CGColor`

색상을 정의하는 구성 요소 세트로, 색상을 해석하는 방법을 지정하는 색상 공간이 있습니다.

`CGColorConversionInfo`

다른 시스템 서비스로 사용하기 위해서 칼라 스페이스 간의 변환 방법을 기술하는 오브젝트입니다.

`CGColorSpace`

표시 할 색상 값을 해석하는 방법을 지정하는 프로파일.

`CGFont`

그리기 텍스트의 문자 모양 및 레이아웃 정보 집합입니다.

Core Graphics

PDF 문서 작업

`CGPDFDocument`

PDF (Portable Document Format) 도면 정보가 포함 된 문서.

유틸리티 및 지원 클래스

`CGDataConsumer`

원시 메모리 버퍼를 관리 할 필요가없는 데이터 쓰기 작업을위한 추상화.

`CGDataProvider`

원시 메모리 버퍼를 관리 할 필요가없는 데이터 읽기 작업을위한 추상화.

`CGShading`

방사형 및 축 방향 그라디언트 채우기를 그리는 데 사용자가 제공하는 사용자 지정 함수에 의해 제어되는 색상 간 부드러운 전환에 대한 정의입니다.

`CGGradient`

레이디 얼 그라디언트 및 축 그라디언트 채우기를 그리기위한 색상 간 부드러운 전환에 대한 정의입니다.

`CGFunction`

콜백 함수를 정의하고 사용하기위한 일반적인 기능.

`CGPattern`

그래픽 경로를 그리는 데 사용할 2D 패턴입니다.

CGContext

- 코어 그래픽스(Quartz 2D)에서 그림을 그리는 영역을 가르키는 객체
- 윈도우, 비트 맵 이미지, PDF 문서 등 페이지의 대상을 대상에 렌더링하는 데 필요한 그리기 매개 변수와 모든 장치 관련 정보가 들어 있음.

```
UIGraphicsBeginImageContext(imageView.frame.size)  //CGContext 시작
let context = UIGraphicsGetCurrentContext()    //CGContext 얻기
context?.setLineWidth(2.0)      //선의 굵기 설정
context?.setStrokeColor(UIColor.red.cgColor)        //그려지는 색
context?.move(to: CGPoint(x: 50, y: 50))        //기준점 좌표이동 (기본 0,0)
context?.addLine(to: CGPoint(x: 250, y: 250))        //선 추가
context?.strokePath()      //설정한 선 그리기
imageView.image = UIGraphicsGetImageFromCurrentImageContext()  //Context 적용
UIGraphicsEndImageContext()        //CGContext 끝
```

CGContext