TextInput

UITextField

UITextView

TextInput Delegates

Customizing TextInput Views

Responding to Keyboard Resizing

UIWebView/WKWebView

Custom Drawing With Core Graphics & UIBezierPath

Text Input

- UIKit offers 2 ways to input text
- UITextField: single lines
- UITextView: multiple lines
- UITextView inherits from UIScrollView
- Delegates are important for interacting with text input controls
- These inputs conform to the UITextInputTraits protocol which allows you to customize the keyboard type

==> Demo W2D5_TextFieldExample <==

==> Demo W2D5_TextViewExample <==

Handling Keyboard Covering TextField Using AutoLayout

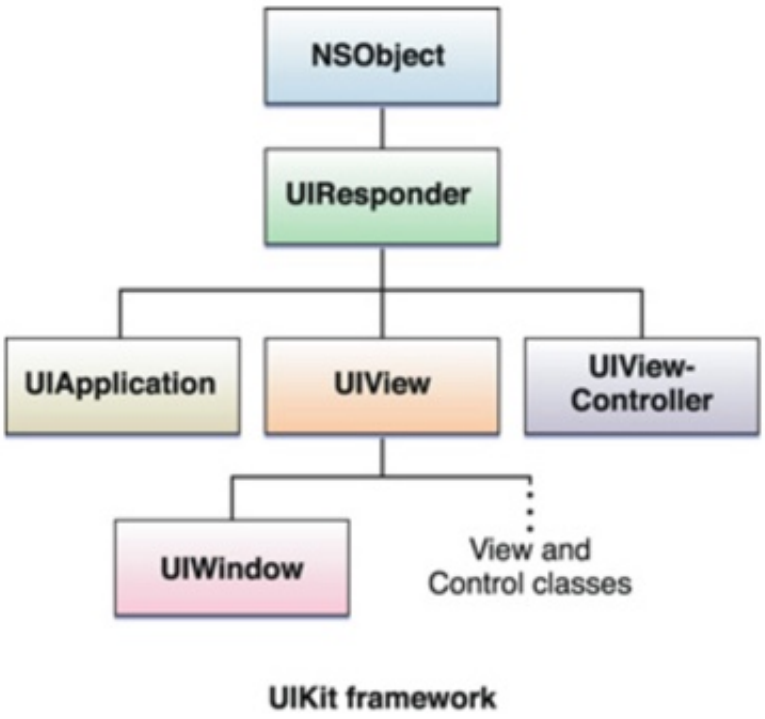
- When the keyboard pops up you may need to adjust your interface to avoid textfield's being covered
- To handle reorientation of interfaces when the keyboard pops up we need to add an observer to listen to the UIWindow's UIKeyboardWillShowNotification, and UIKeyboardWillHideNotification. To get the height of the keyboard.
- Why don't we just hard code the values?
- We get the values from the userInfo dictionary

==> Demo W2D5_KeyboardResizing With Autolayout <==

UIResponder

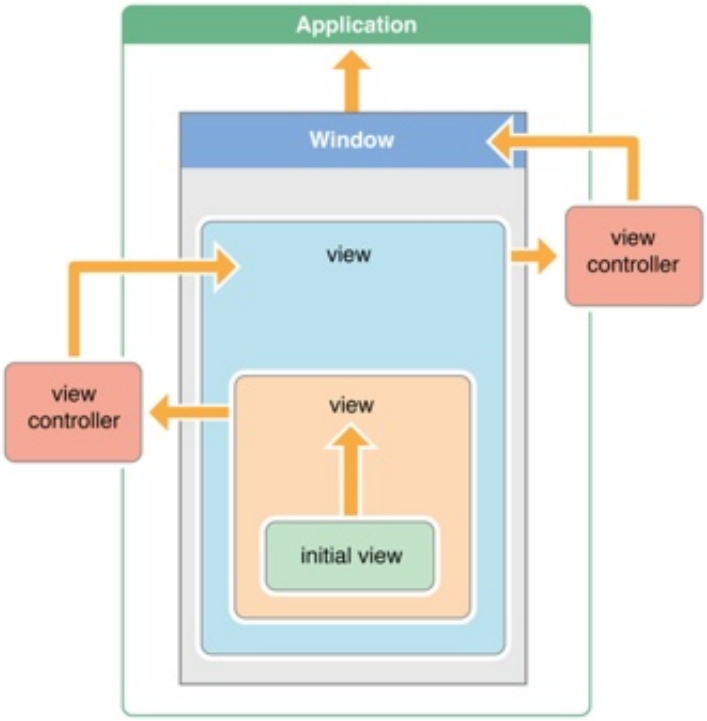
- Both UITextField & UITextView inherit from UIResponder
- Let’s talk about responders in iOS.
- All responder objects that inherit from the class UIResponder
- Any instance that inherits from UIResponder can handle events, like, touch events, motion events
- Visible elments of an app are almost always responders. e.g. all UIViews, controls (like

UIButton)
- ViewControllers and the UIApplication itself are responders.



Responder Chain

- A responder can receive action messages (sent by buttons and controls when touched) that have no target specified. (I will show a code example of this in a moment).
- If a responder cannot handle an event it *automatically* forwards it to the "next responder" in a linked series called the **responder chain**.



- The Responder Chain will always mark one responder as the First Responder & Next Responder. (We will come back to this).
- The responder chain allows flexibility in handling events. (How so?).
- Events travel up the chain starting at the leaf most responder.
- If nothing overrides the event in question it is forwarded to the next responder in the chain until the final responder in the chain the UIApplication.
- If UIApplication delegate doesn't handle it then the event is just discarded.
- *userInteractionEnabled* is a property on UIView that determines whether a view receives interactions. By default this is set to YES for UIView's but most subclasses set this to NO by default. So, if you want

a UIImageView to handle a touch event you must always set the *userInteractionEnabled* to YES. (This is a common beginner gotcha).

==> Demo W2D5_Responder_Example <==

UIWebView / WKWebView

- UIWebView is basically a slightly crippled version of Safari that you can just drop into your view controller.
- UIWebView has been replaced by WKWebView, which is much faster and flexible.
- You can use webviews for displaying web pages (obviously!)
- You can also load web content from other sources, like a data base, or network endpoint.
- You can interact with web content using javascript and do things like make content editable using *contentEditable*. So, you can make a blog editor from a webview, for instance.

==> Demo W2D5_WebView <==

==> Core Graphics Demo <==