WatchKit UI

CS193W - Spring 2016 - Lecture 2

Animating UI Changes

Use `WKInterfaceController`’s

```
func animateWithDuration(_ duration: NSTimeInterval,
              animations animations: () -> Void)
```

The following properties are animatable:

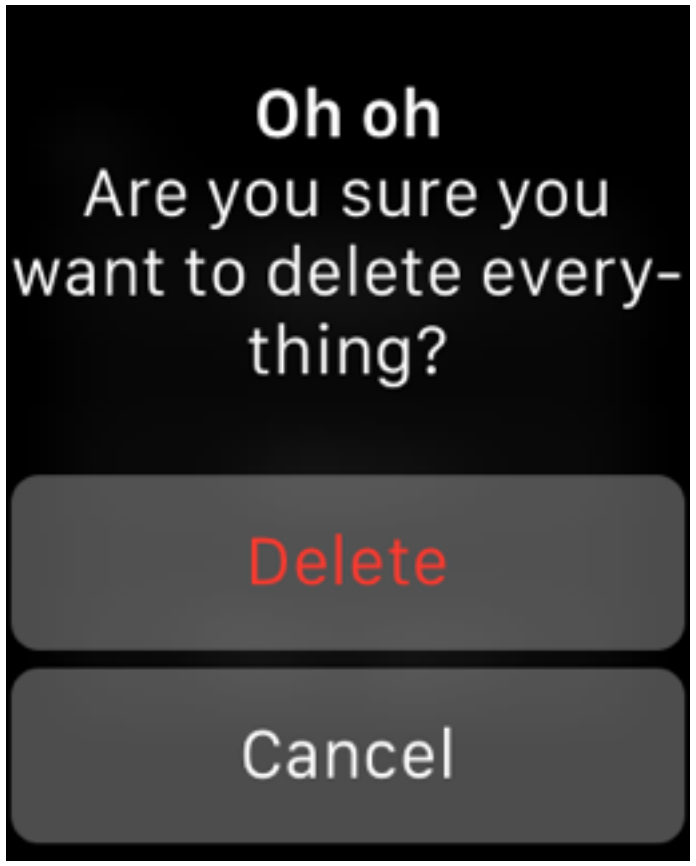
- alpha (opacity)
- width and height
- horizontal and vertical alignment
- background or tint color
- group content insets

Animation Example

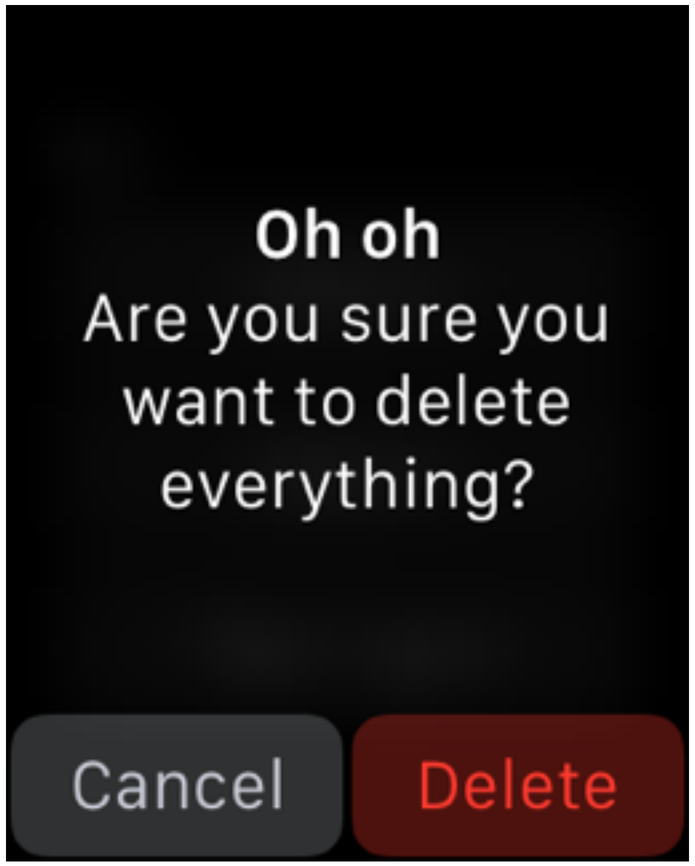
```
animateWithDuration(0.3, animations: {
    self.spacerGroup.setHeight(50)
    self.spacerGroup.setAlpha(.5)

    self.topGroup.setBackgroundColor(UIColor.redColor())
})
```

Alerts

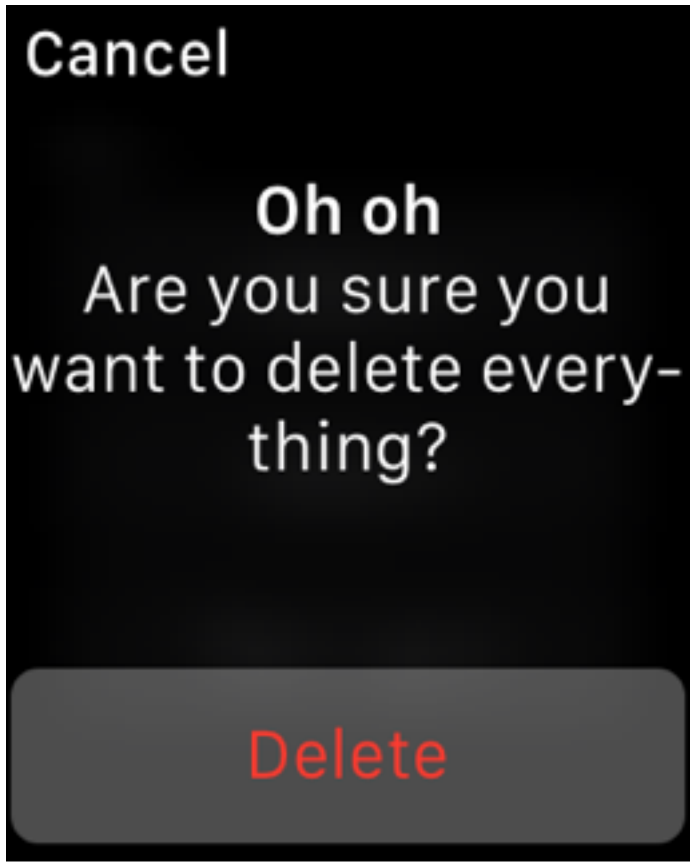
Alert Styles



.Alert



.SideBySideButtonsAlert



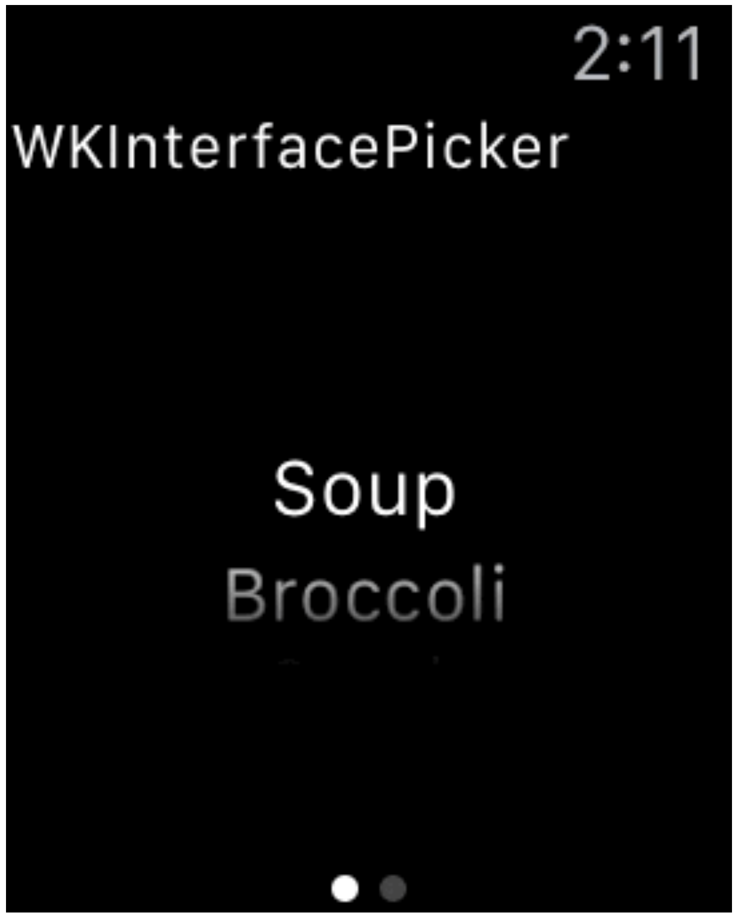
.ActionSheet

Alert Example

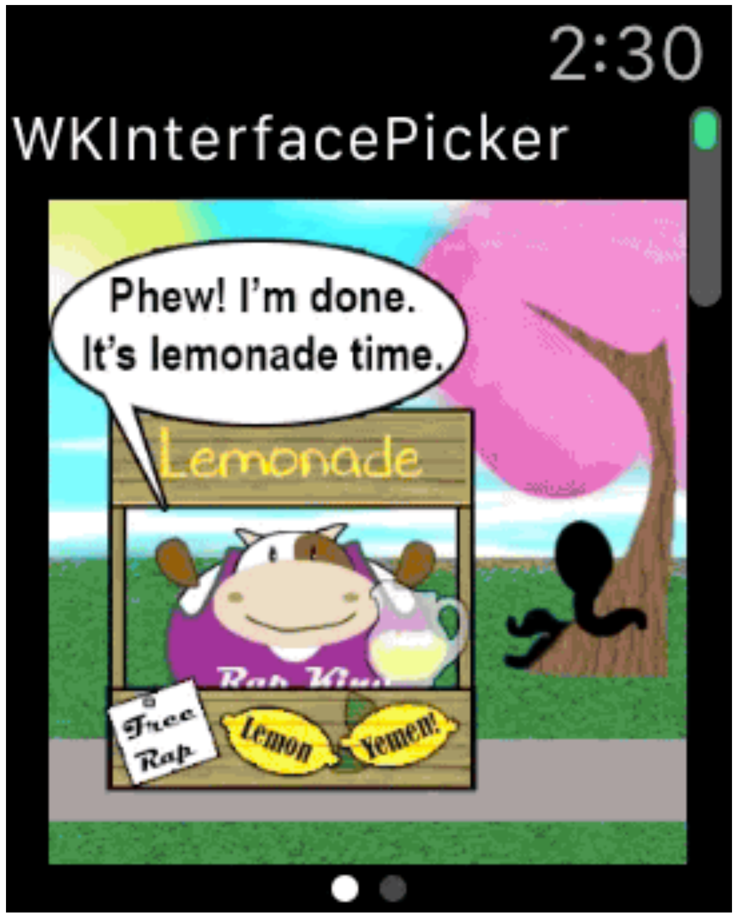
```
let cancelAction = WKAlertAction(title: "Nope", style: .Cancel,
  handler: {})

let agreeAction = WKAlertAction(title: "Delete", style: .Destructive,
  handler: {self.deleteEverything()})

self.presentAlertControllerWithTitle("Oh oh", message: "Are you sure
you want to delete everything?", preferredStyle: .ActionSheet, actions:
[cancelAction, agreeAction])
```

WKInterfacePicker

Picker Styles



List



Stack

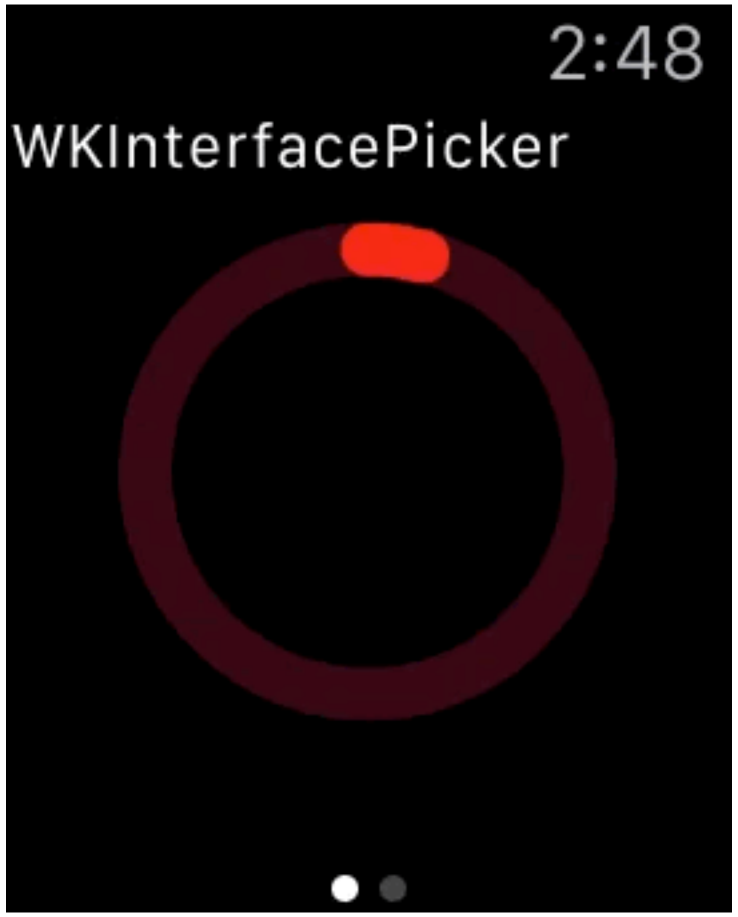
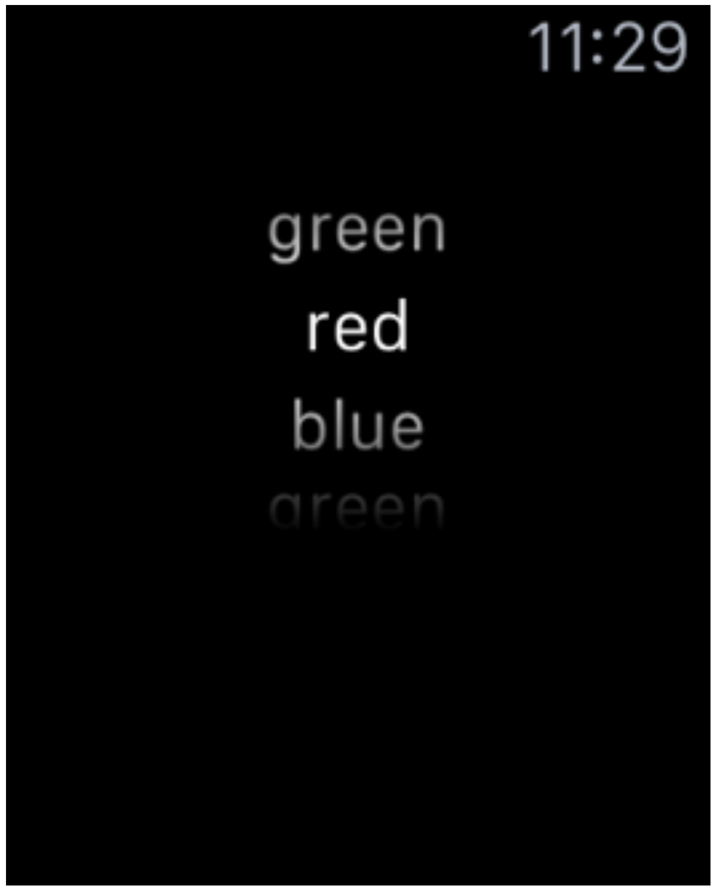


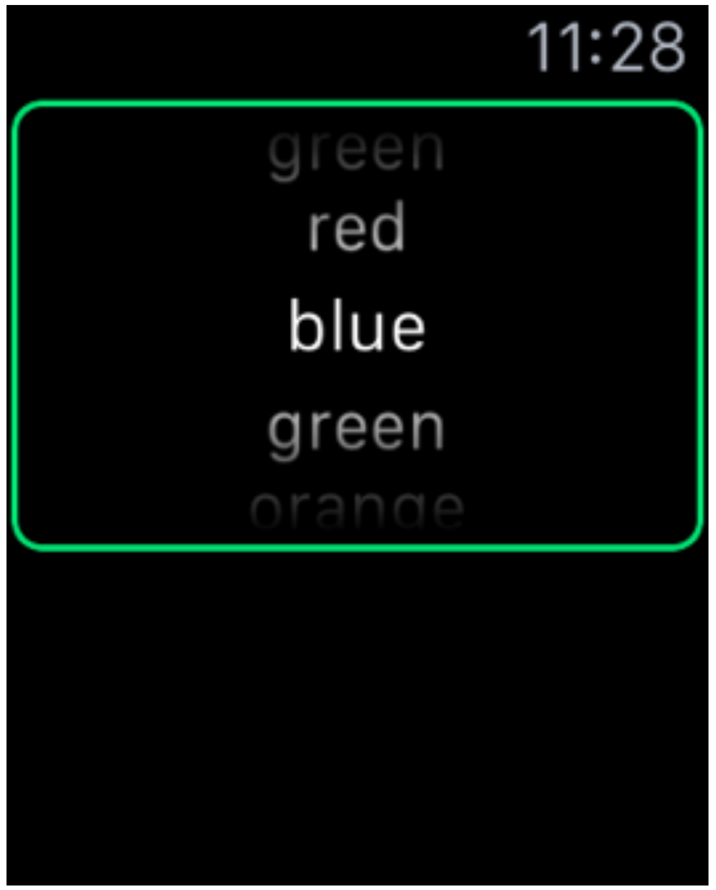
Image Sequence

Source: http://www.sneakycrab.com/blog/2015/6/12/wkinterfacepicker-in-watchkit-20-using-the-digital-crown

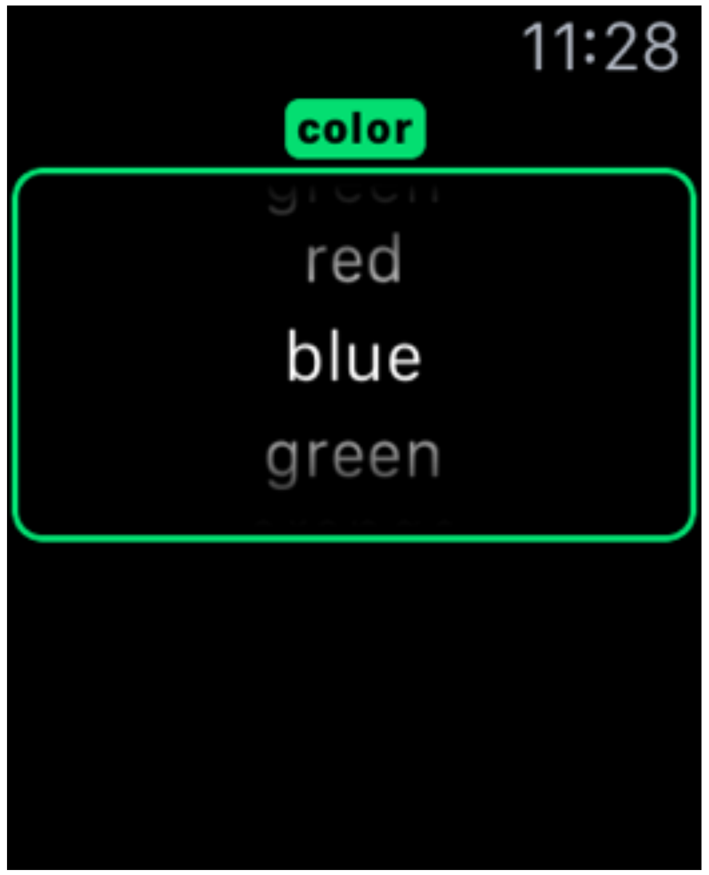
Indicator Styles



None



Outline



Outline With Caption

Note: scroll with two fingers on the trackpad to scroll a picker in the Simulator

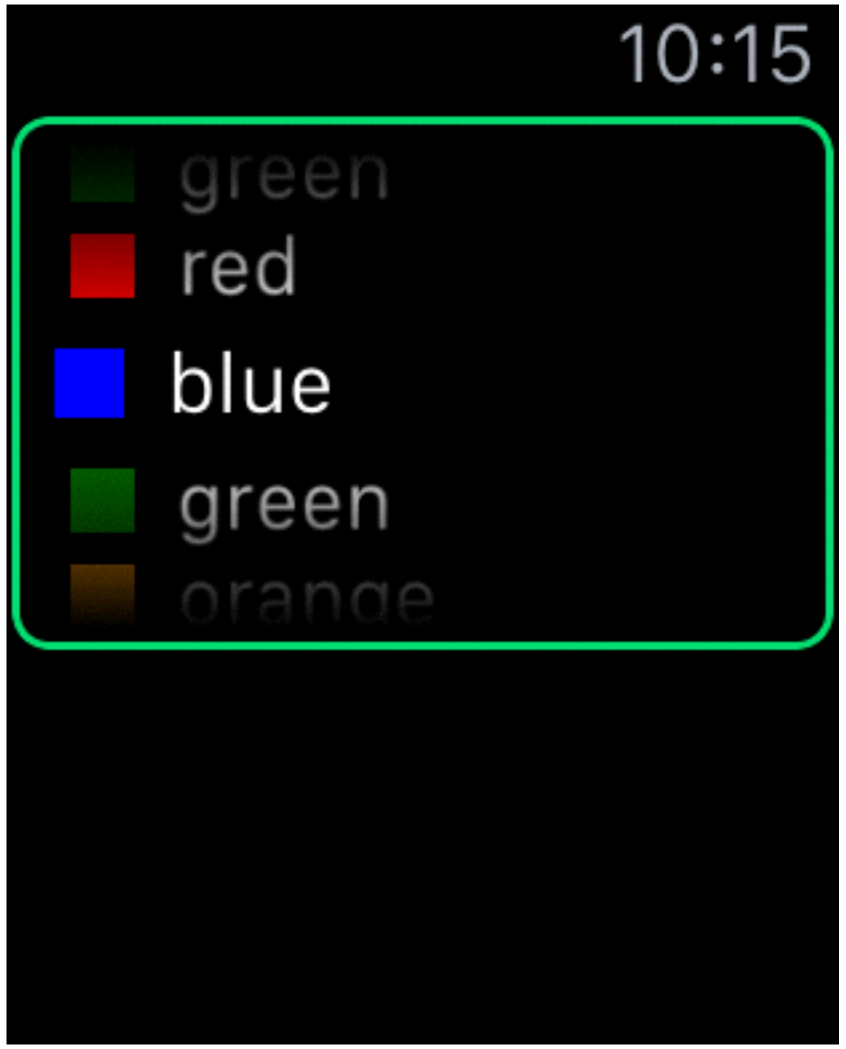
WKPickerItem (Textual List Style)

```
let pickerItem1 = WKPickerItem()
pickerItem1.title = "green"
pickerItem1.caption = “color"

let pickerItem2 = WKPickerItem()
pickerItem2.title = "red"
pickerItem2.caption = “color"

let pickerItem3 = WKPickerItem()
pickerItem3.title = "blue"
pickerItem3.caption = “color"

self.myPicker.setItems([pickerItem1, pickerItem2, pickerItem3])
```

WKPickerItem.accessoryImage

- Optional 13pt x 13pt image

```
let pickerItem1 = WKPickerItem()
pickerItem1.title = "green"
pickerItem1.caption = "color"
pickerItem1.accessoryImage = WKImage(imageName: "green")
```



WKImage

- A wrapper type used for some `WKInterfaceObject`s

```
convenience init(image image: UIImage)
convenience init(imageData imageData: NSData)
convenience init(imageName imageName: String)
```

WKPickerItem.contentImage

Can be used for any of the styles (List / Stack / Image Sequence)

```swift
let pickerItem1 = WKPickerItem()
pickerItem1.caption = "color"
pickerItem1.contentImage = WKImage(imageName: "green")

let pickerItem2 = WKPickerItem()
pickerItem2.caption = "color"
pickerItem2.contentImage = WKImage(imageName: "red")

let pickerItem3 = WKPickerItem()
pickerItem3.caption = "color"
pickerItem3.contentImage = WKImage(imageName: "blue")
```

Coordinated Animations

- The picker can cause other images to animate in a coordinated fashion

```
func setCoordinatedAnimations(_ coordinatedAnimations: [WKInterfaceObject]?)
```

`coordinatedAnimations` is an array of `WKInterfaceObject`s that conform to the `WKImageAnimatable` protocol (i.e. `WKInterfaceImage`)

- The number of frames in each animatable image need not correspond to the number of frames in the picker

- The percentage of the index of the selected frame in the picker drives the percentage of the selected frame in the coordinated animation
  - e.g. If the picker has 10 frames and the animatable image has 20 frames, the 3rd frame of the picker corresponds to the 6th frame of the coordinated animation

Focus and Selection

The user can focus by tapping a picker, and unfocused by scrolling the interface controller. You can focus / unfocus programmatically.

```
func focus()
```

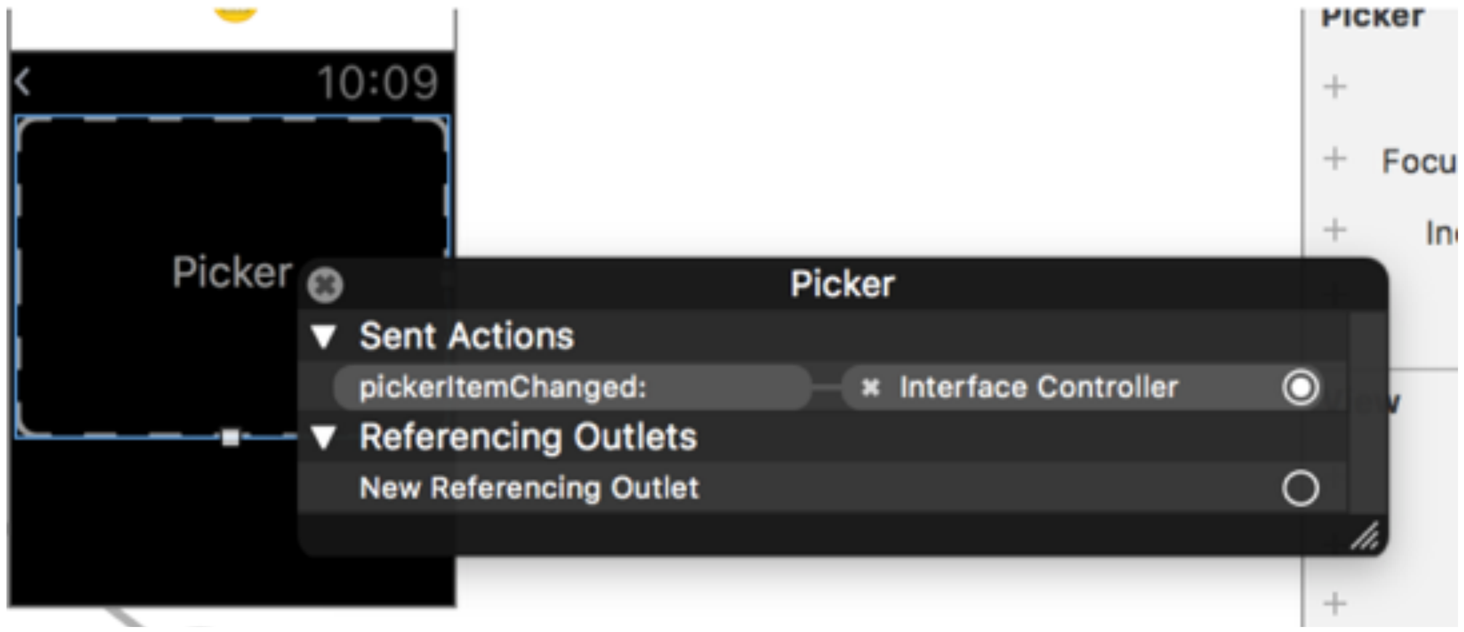
```
func resignFocus()
```

```
func setSelectedItemIndex(_ itemIndex: Int)
```

Handling Picker Actions in WKInterfaceController

```
func pickerDidFocus(_ picker: WKInterfacePicker)
func pickerDidResignFocus(_ picker: WKInterfacePicker)

func pickerDidSettle(_ picker: WKInterfacePicker)
```

Obtaining the Picked Item



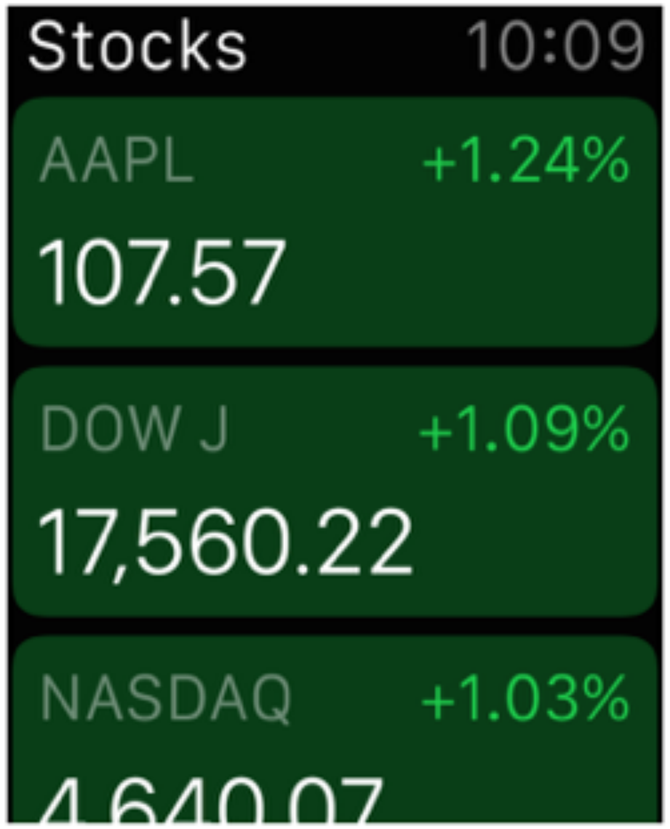
```
- (IBAction)pickerItemChanged:(NSInteger)value
```

`value` is the index of the selected item.

Tables

WKInterfaceTable

- Can have multiple row types
- HIG recommends that no more than 20 rows are displayed at once
- Each row is managed by a Row Controller (which is just a subclass of `NSObject`)



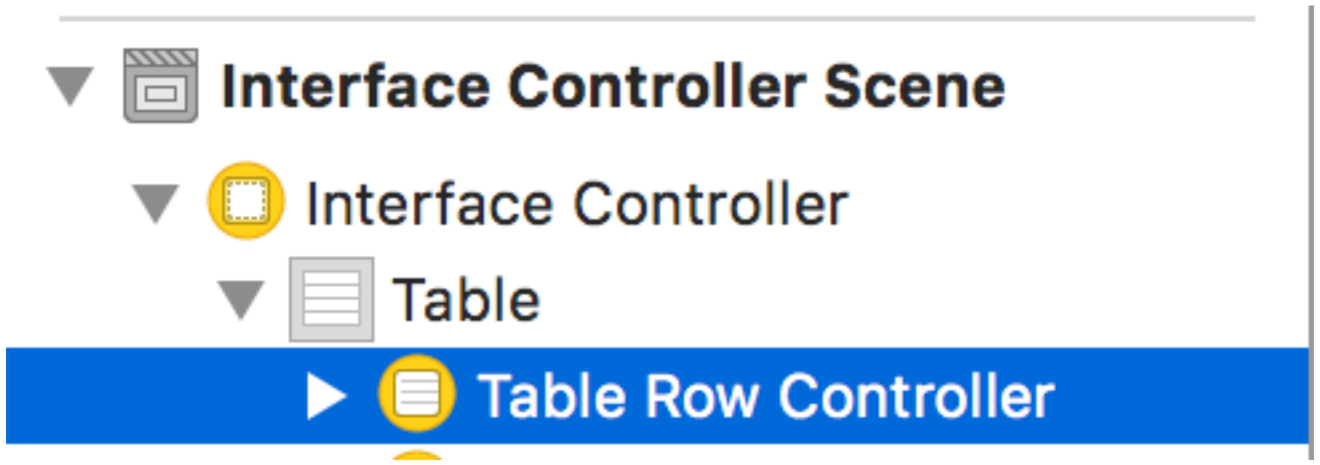
Tables in the Storyboard

- When you create a table in the storyboard, Interface Builder will include 1 prototype row
- You can add more prototypes if you have more types of row controllers
- The number of prototype rows in the storyboard does not affect the number of rows in the table; rows are specified programmatically

Creating a Row Controller

1. Create a new class for your row controller that is a subclass of `NSObject`

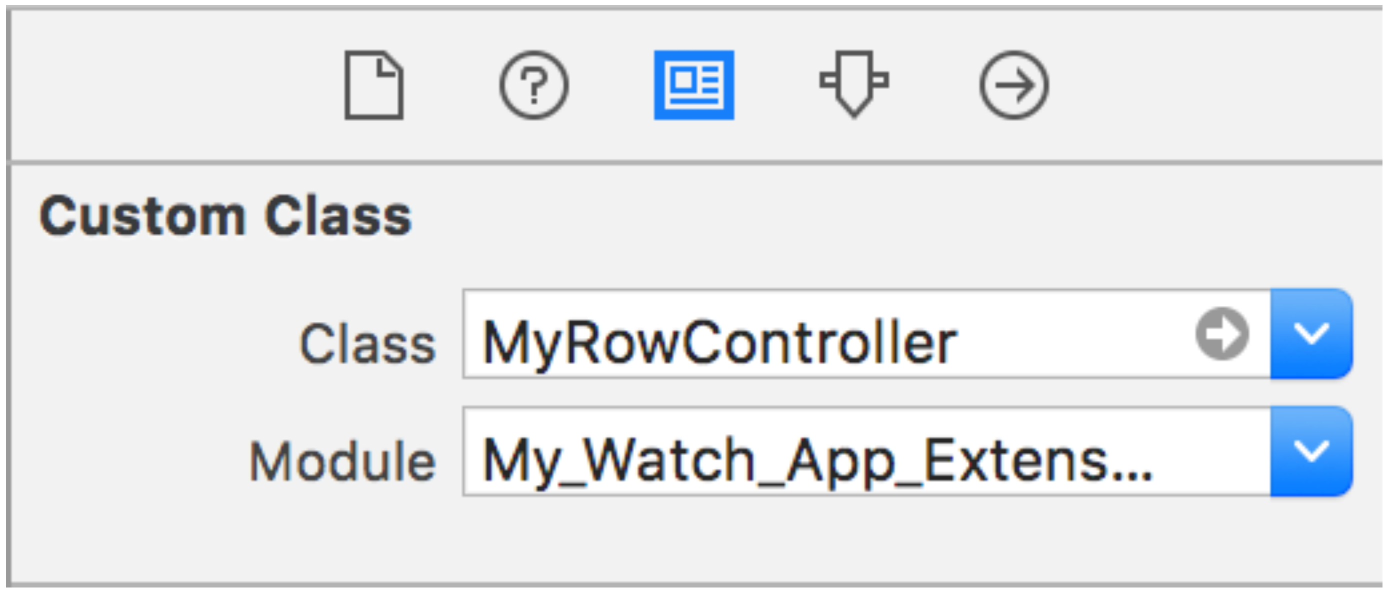
2. Select your row controller in the storyboard



Creating a Row Controller

3.Specify the class of the row controller

Note: If you do not create the class first, the module will not be set!

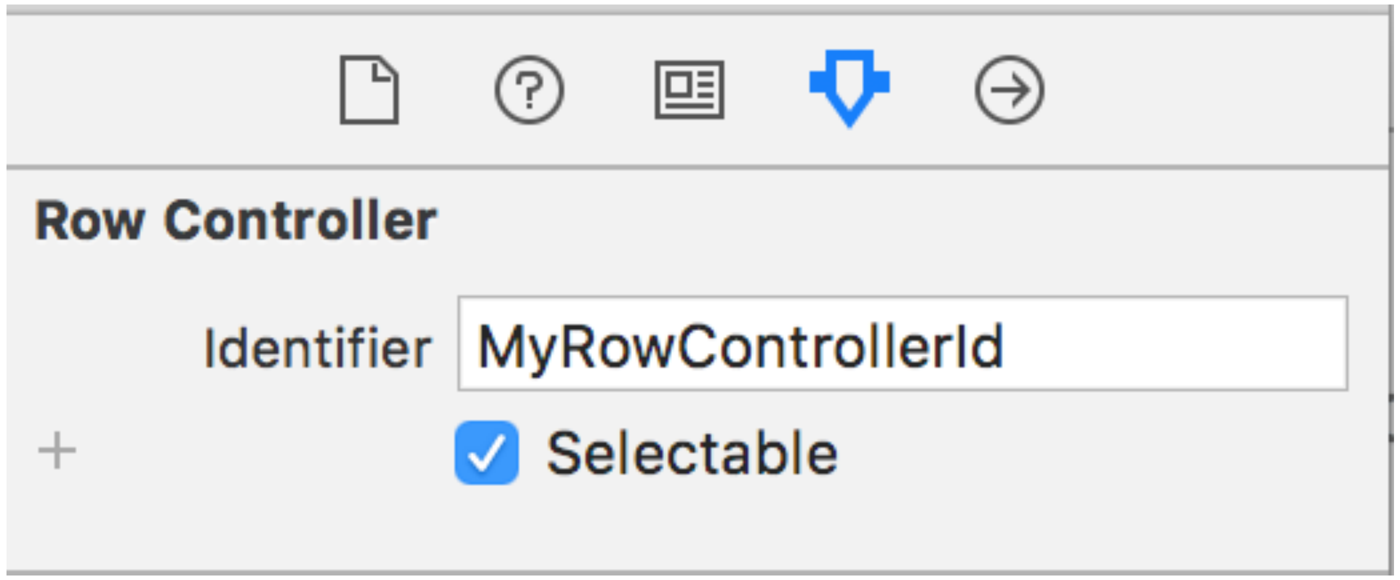


Custom Class

Class: MyRowController

Module: My_Watch_App_Extens...

4.Specify an identifier for the row controller



Row Controller

Identifier: MyRowControllerId

✓ Selectable

Specifying the number of rows and their types

```
func setRowTypes(_ rowTypes: [String])

func setNumberOfRows(_ numberOfRows: Int,
         withRowType rowType: String)
```

Specifying the number of rows and their types

```
func insertRowsAtIndexes(_ rows: NSIndexSet,
               withRowType rowType: String)

func removeRowsAtIndexes(_ rows: NSIndexSet)
```

Querying Tables

```swift
var numberOfRows: Int { get }

func rowControllerAtIndex(_ index: Int) -> AnyObject?
```

Scrolling Tables

```
func scrollToRowAtIndex(_ index: Int)
```

Row Selections

```
WKInterfaceController:

func table(_ table: WKInterfaceTable, didSelectRowAtIndex rowIndex: Int)


func contextForSegueWithIdentifier(_ segueIdentifier: String,
                                    inTable table: WKInterfaceTable,
                                   rowIndex rowIndex: Int) -> AnyObject?

func contextsForSegueWithIdentifier(_ segueIdentifier: String,
                                     inTable table: WKInterfaceTable,
                                    rowIndex rowIndex: Int) -> [AnyObject]?
```

Table Example

```
// MyRowController.swift

import WatchKit

class MyRowController: NSObject {

    @IBOutlet var myLabel: WKInterfaceLabel!
}
```

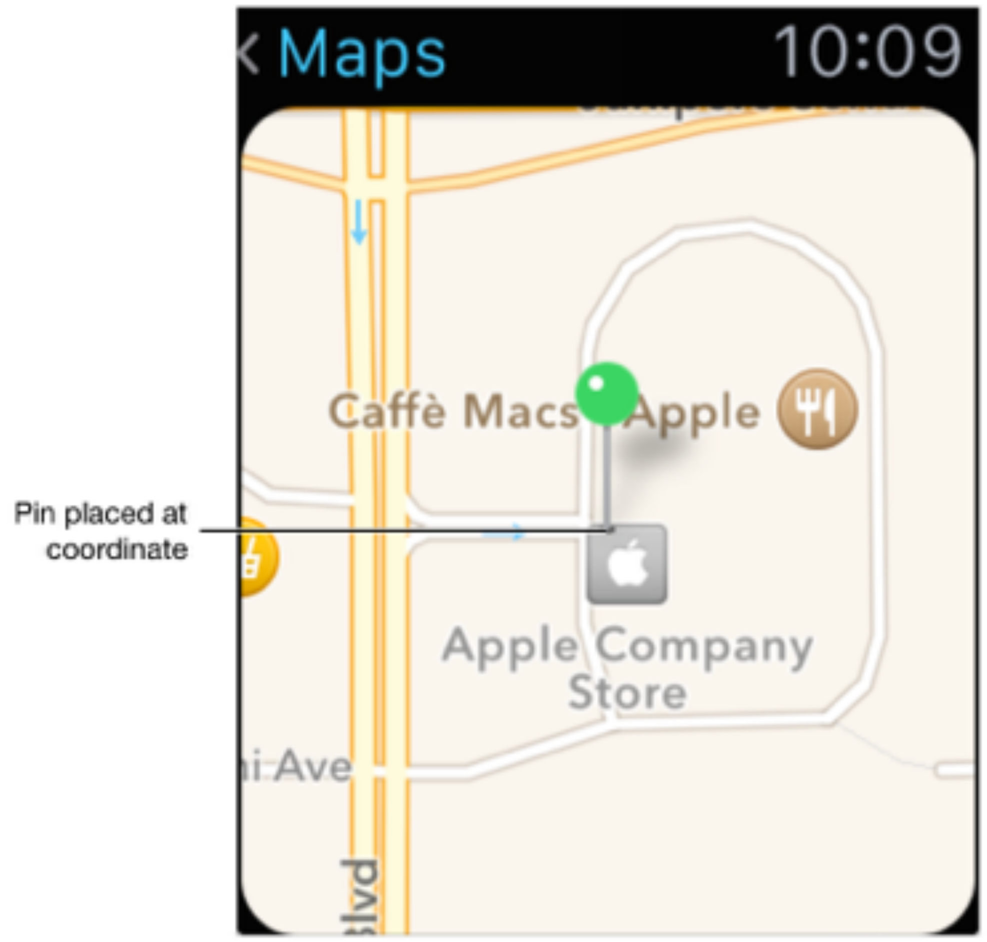
```
// MyInterfaceController.swift

@IBOutlet var myTable: WKInterfaceTable!

override func awakeWithContext(context: AnyObject?) {
    super.awakeWithContext(context)

    myTable.setNumberOfRows(3, withRowType: "MyRowControllerId")

    let row0 = myTable.rowControllerAtIndex(0) as! MyRowController
    row0.myLabel.setText("The Mayflower”)

   …
}

func table(table: WKInterfaceTable, didSelectRowAtIndex rowIndex: Int) {
   …
}
```

Maps

WKInterfaceMap

- Static (tapping on them opens Maps app)

- Can contain up to 5 custom annotations



WKInterfaceMap API

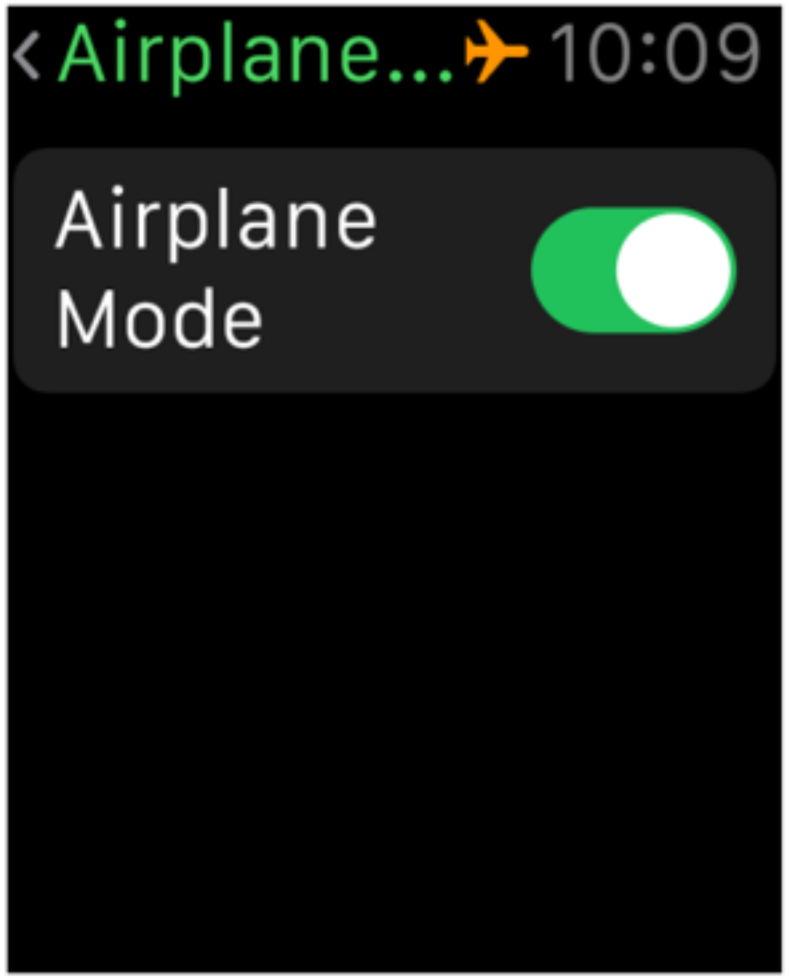
```
func setVisibleMapRect(_ mapRect: MKMapRect)

func setRegion(_ coordinateRegion: MKCoordinateRegion)
```

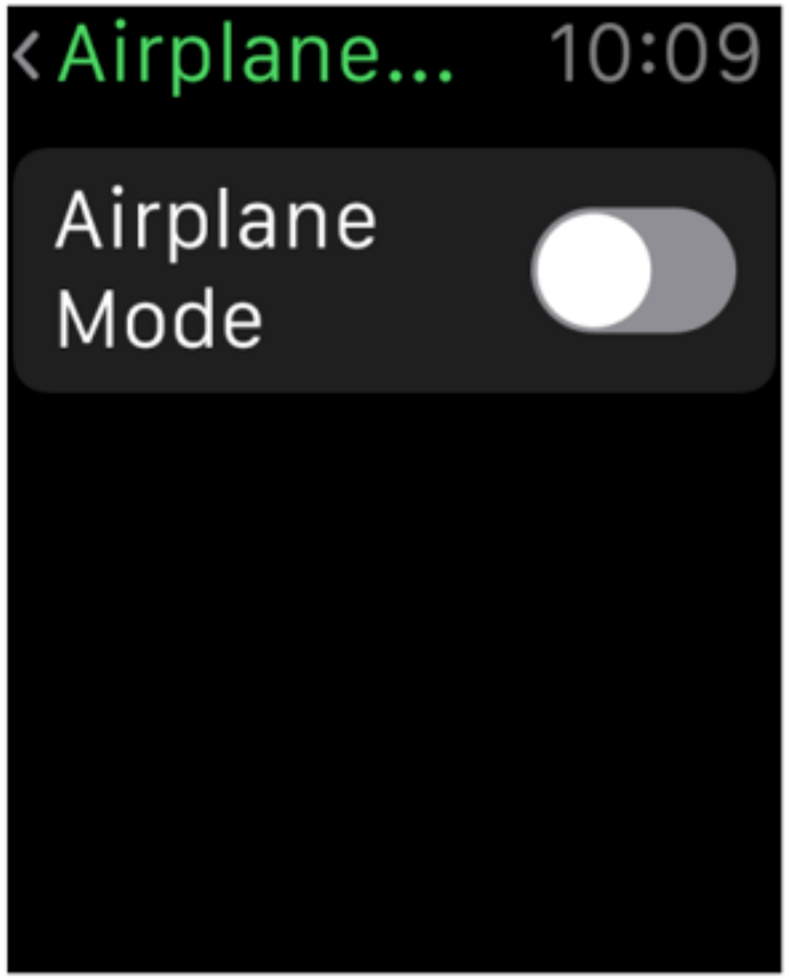
Annotations

```
func addAnnotation(_ location: CLLocationCoordinate2D,
          withImage image: UIImage?,
       centerOffset offset: CGPoint)

func addAnnotation(_ location: CLLocationCoordinate2D,
    withImageNamed name: String?,
       centerOffset offset: CGPoint)

func addAnnotation(_ location: CLLocationCoordinate2D,
       withPinColor pinColor: WKInterfaceMapPinColor)

func removeAllAnnotations()



enum WKInterfaceMapPinColor : Int {
    case Red
    case Green
    case Purple
}
```

Switches

WKInterfaceSwitch





- Always contains a label

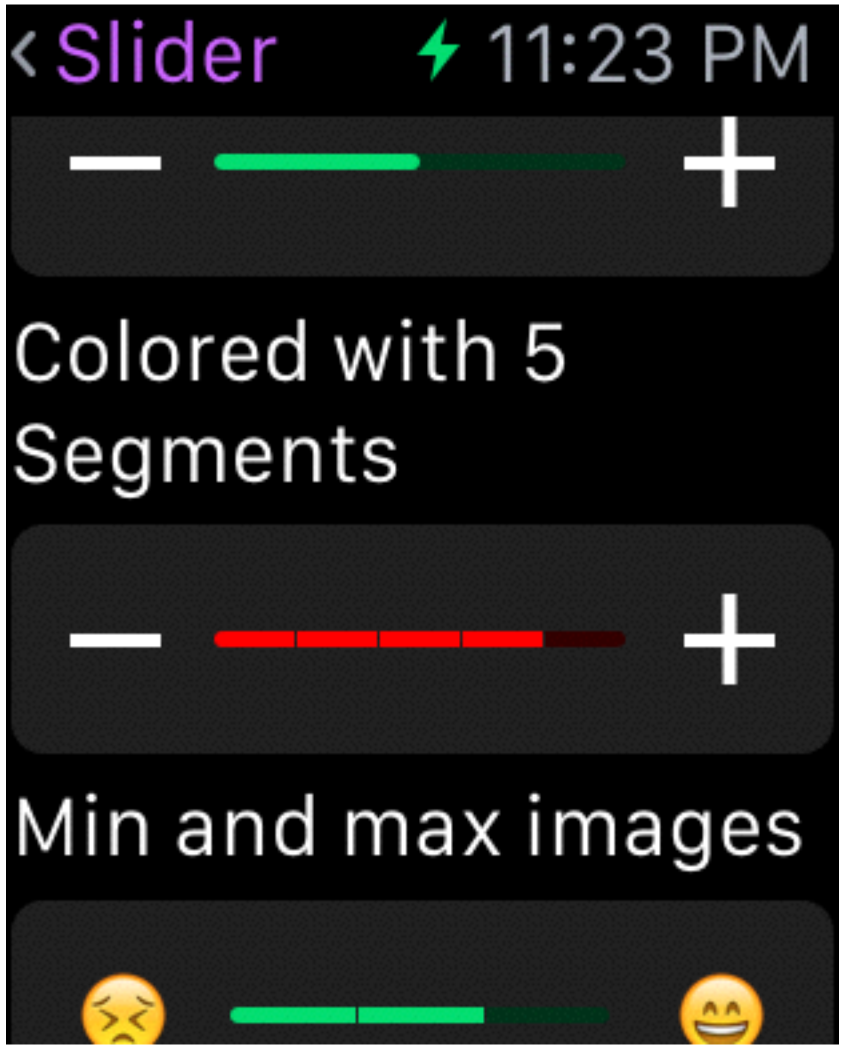
WKInterfaceSwitch API

```
func setTitle(_ title: String?)

func setAttributedTitle(_ attributedTitle:
NSAttributedString?)



func setOn(_ on: Bool)

func setColor(_ color: UIColor?)



func setEnabled(_ enabled: Bool)
```

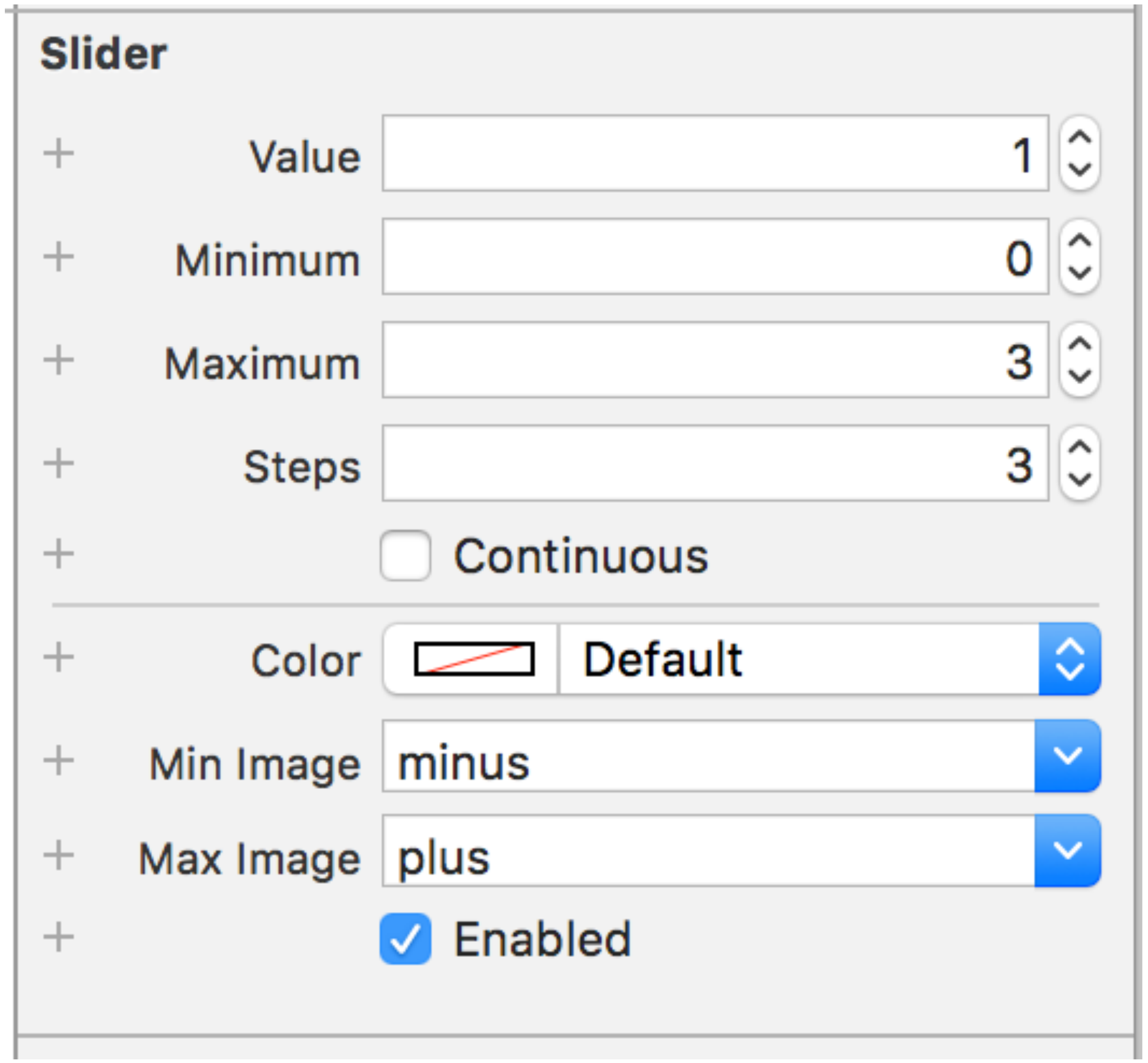
Sliders

WKInterfaceSlider

- Always represents discrete values
- Segmented or continuous look
- Has images at either end to manipulate value



Configuring WKInterfaceSlider



Slider

Value	1
Minimum	0
Maximum	3
Steps	3
	☐ Continuous
Color	Default
Min Image	minus
Max Image	plus
	✓ Enabled

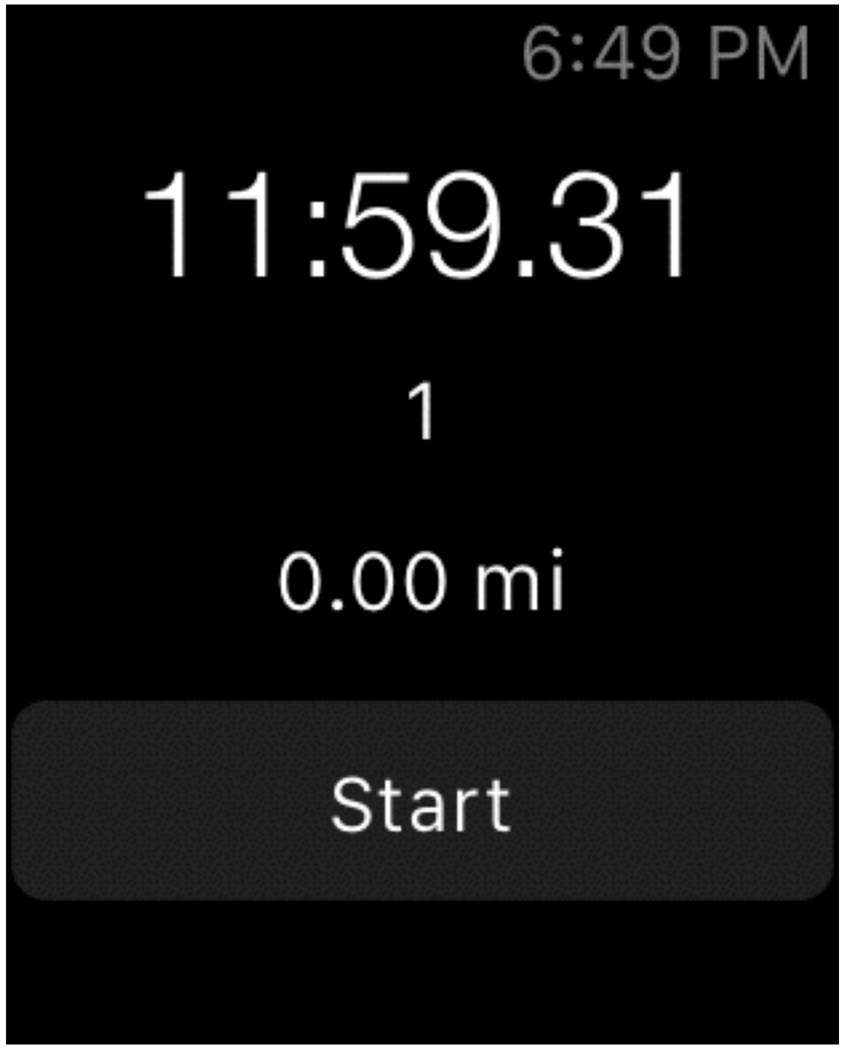
WKInterfaceSlider API

```
func setValue(_ value: Float)

func setColor(_ color: UIColor?)

func setNumberOfSteps(_ numberOfSteps: Int)


func setEnabled(_ enabled: Bool)
```

Telling the Current Time

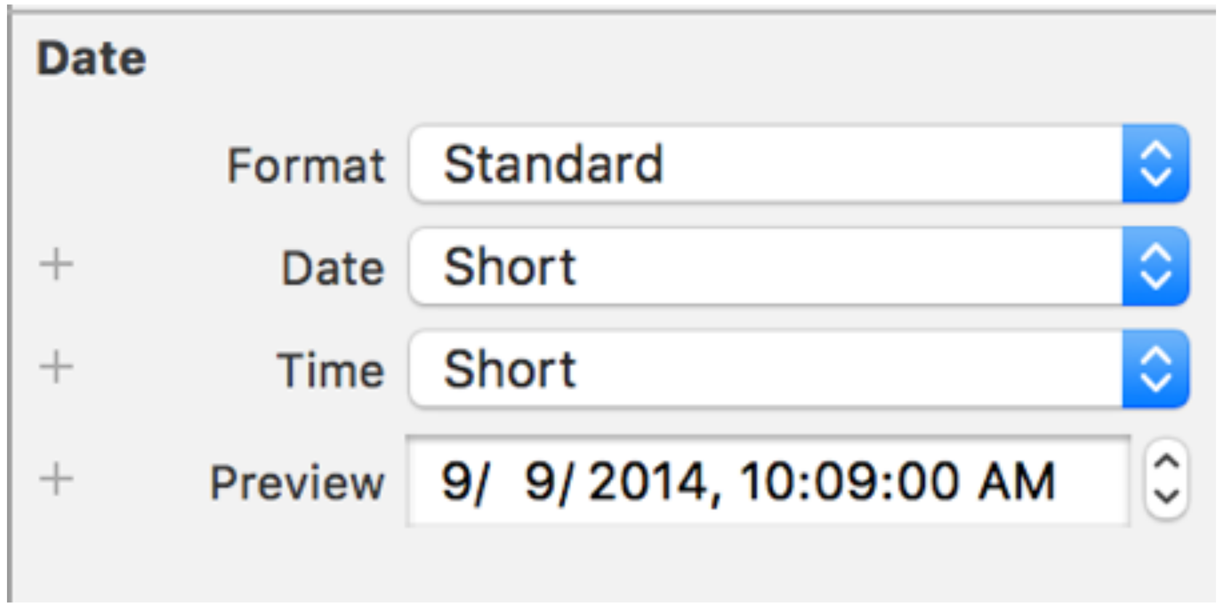
WKInterfaceDate

- Tells the current time
- Requires no updates from your code



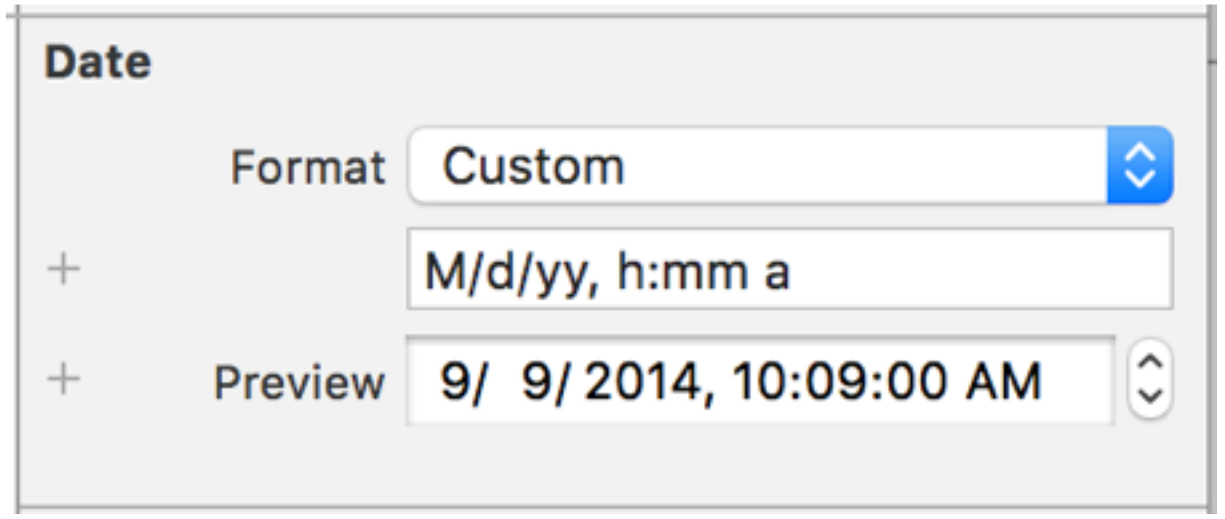
Configuring WKInterfaceDate

- You can use `NSDateFormatterStyles` or create a format string using the Unicode tr-35-31 Standard.



Date

Format: Standard
Date: Short
Time: Short
Preview: 9/ 9/ 2014, 10:09:00 AM



Date

Format: Custom
M/d/yy, h:mm a
Preview: 9/ 9/ 2014, 10:09:00 AM

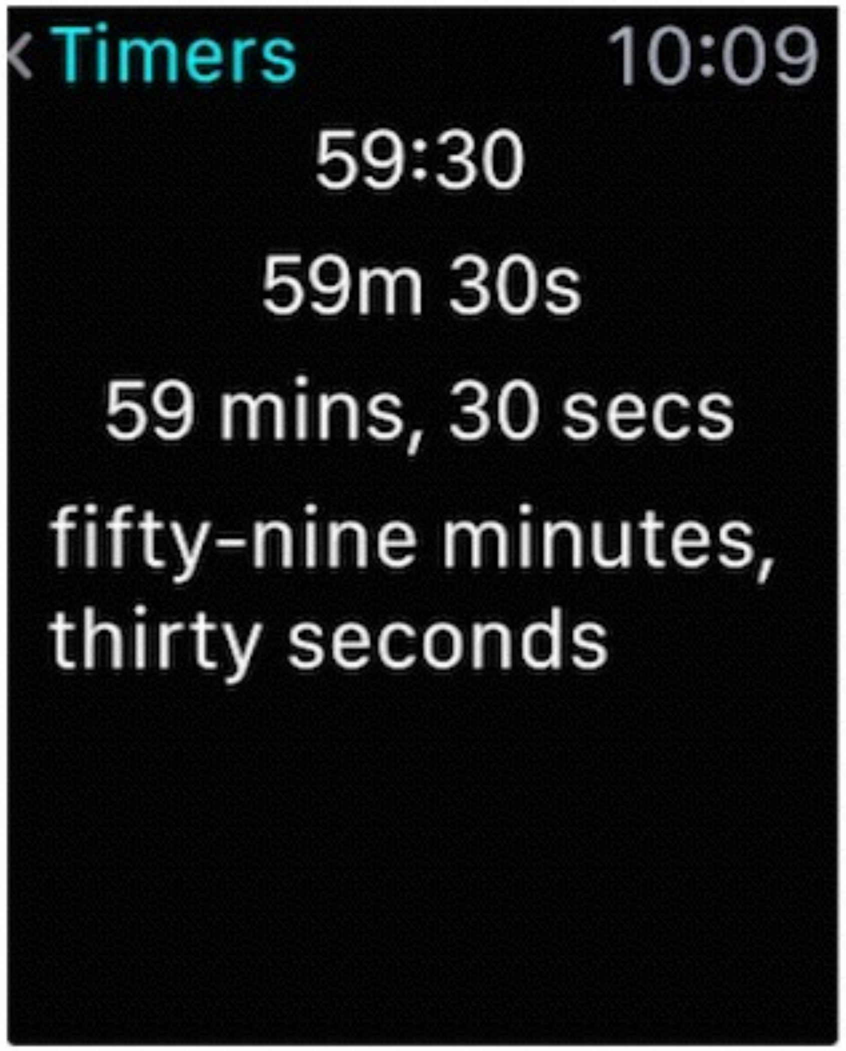
WKInterfaceDate API

```
func setTextColor(_ color: UIColor?)

func setTimeZone(_ timeZone: NSTimeZone?)

func setCalendar(_ calendar: NSCalendar?)
```

Timers

WKInterfaceTimer

- Can count up or down
- Requires no updates from your code


‹ Timers 10:09

59:30

59m 30s

59 mins, 30 secs

fifty-nine minutes, thirty seconds

Starting and Stopping the Timer

```
func start()
```

Starts updating the label.

```
func stop()
```

Stops updating the label. Does **not** stop the timer.

WKInterfaceTimer API

```
func setDate(_ date: NSDate)
```

The timer will count up from the current time or down to a time in the future. Call `start`() after setting this.

Haptic Feedback

Haptic Feedback

```
WKInterfaceDevice.currentDevice().playHaptic(_ type: WKHapticType)
```

```
enum WKHapticType : Int {
    case Notification     // Alerts the user to an arrived notification when the Watch app is not running in the foreground.
    case DirectionUp      // An increase in a specific value or when a value has gone above a certain threshold.
    case DirectionDown    // A decrease in a specific value or when a value has gone below a certain threshold.
    case Success          // the successful completion of a task or the answering of a question.
    case Failure  // The failed completion of a task or answering of a question.
    case Retry    // The user should retry a task that temporarily failed.
    case Start    // The beginning of an action.
    case Stop     // The end of an action
    case Click    // Mark fixed points along a path.
}
```

https://developer.apple.com/watch/human-interface-guidelines/watch-technologies/#haptic-feedback

Haptic Guidelines

- Use haptics sparingly
  - “Strong” feedback to the user
  - Drains the battery quickly
- There is a slight delay when playing a haptic
- Playing two haptic close together results in the first haptic being interrupted, followed by a delay of at least 100ms before playing the second haptic

Accessibility

Accessibility Concepts

- Apple Watch can speak aloud accessibility *labels*, *hints*, and *values*
- *Label*. A short, localized word or phrase that succinctly describes the control or view, but does not identify the element’s type. Examples are “Add” or “Play.”.
- *Hint*. A brief, localized phrase that describes the results of an action on an element. Examples are “Adds a title” or “Opens the shopping list.”
- *Value*. The current value of an element, when the value is not represented by the label. For example, the label for a slider might be “Speed,” but its current value might be “50%.”

WKAccessibilityImageRegion

```
func setAccessibilityImageRegions(_ accessibilityImageRegions: [ AnyObject ])
```

```
let region1 = WKAccessibilityImageRegion()
region1.frame = CGRect(x: 0, y: 0, width: 30, height: 30)
region1.label = "Clown Face"

let region2 = WKAccessibilityImageRegion()
region2.frame = CGRect(x: 0, y: 30, width: 30, height: 30)
region2.label = "Mini-Golf Set"

todoTable.setAccessibilityImageRegions([ region1, region2 ])
```