

How to Make a Touch Bar App

Outline

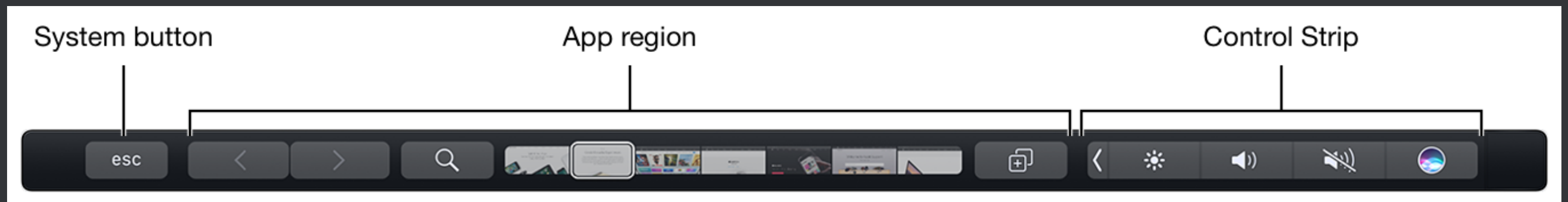
- What is a **Touch Bar** 🤔
- Read the **Docs** 📖
- Write the **Code** ✍️

What is a **Touch Bar**



About the Touch Bar¹

- Retina display
- Input device
- Dynamic interface
- Touch ID



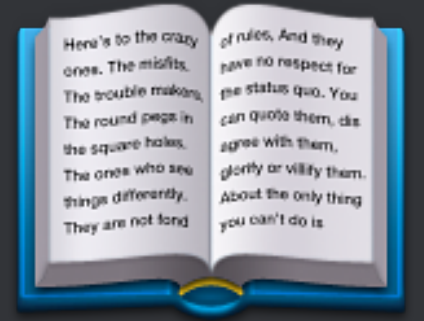
¹ <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/OSXHIGuidelines/AbouttheTouchBar.html>

Gestures²

- Tap
- Touch and hold
- Horizontal swipe (pan)
- Multitouch

² <https://developer.apple.com/library/content/documentation/UserExperience/Conceptual/OSXHIGuidelines/Interaction.html>

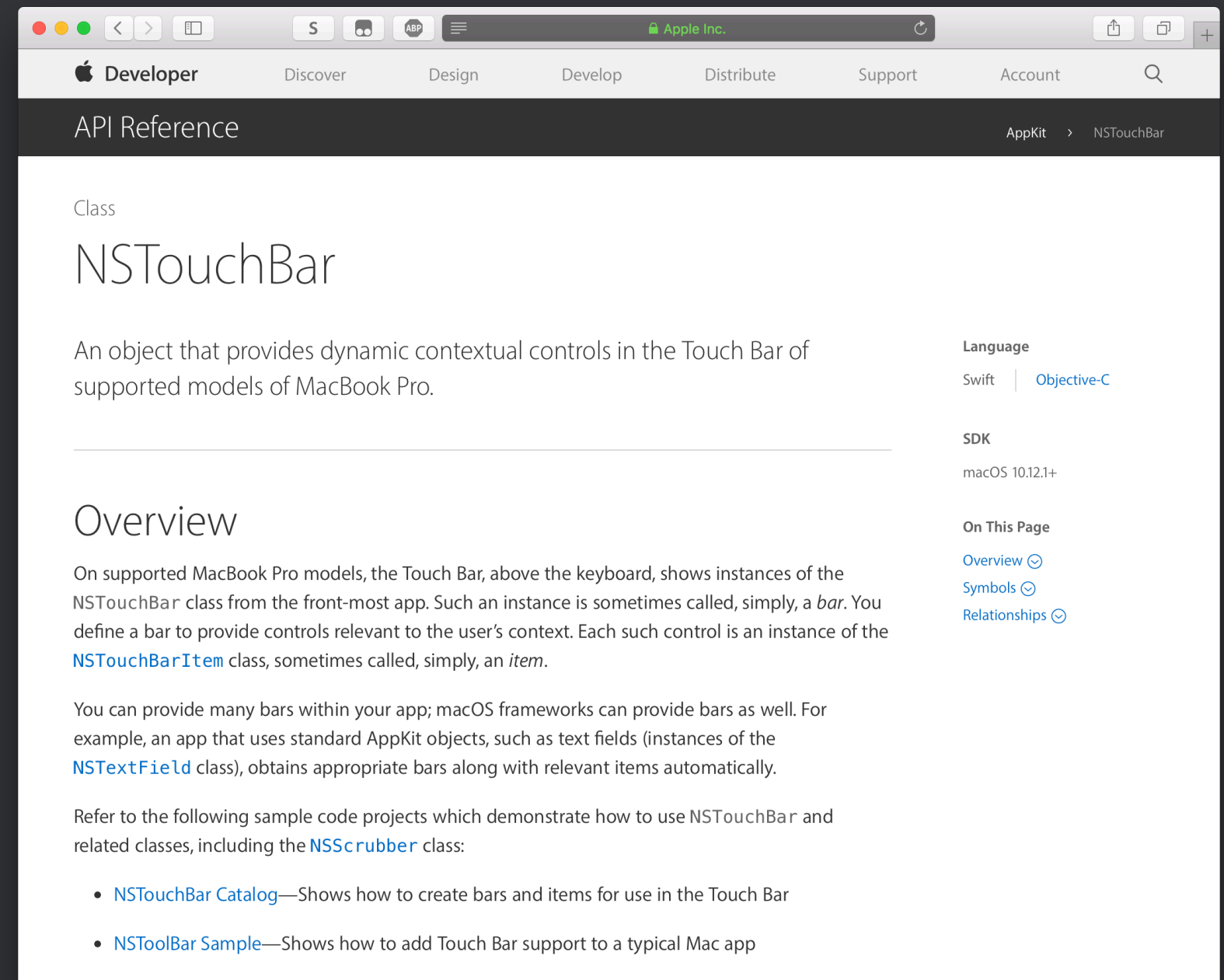
Read the Docs



Open this **NSTouchBar API Reference**

At a Glance

- SDK: macOS 10.12.1+
- **NSTouchBar**: a *bar*
- **NSTouchBarItem**: an *item*



Musts

1. Be a responder (an instance of an **NSResponder** subclass) that is present within a responder chain at runtime
2. Conform to the **NSTouchBarProvider** protocol
3. Implement the **makeTouchBar()** method within that protocol

At a Glance (cont.)

- `NSTouchBar: customizationIdentifier`
- `NSTouchBarItem:`
 - `defaultItemIdentifiers`
 - `customizationAllowedItemIdentifiers`
 - `customizationRequiredItemIdentifiers`
- `NSTouchBarDelegate: delegate`

Write the Code



Environment³

- macOS Sierra 10.12.1 (16B2657)
- Xcode 8.1 (8B62)



Xcode and macOS Sierra

To develop apps that use the Touch Bar, your Mac will need to run Xcode 8.1 on macOS Sierra 10.12.1 (16B2657) or later.

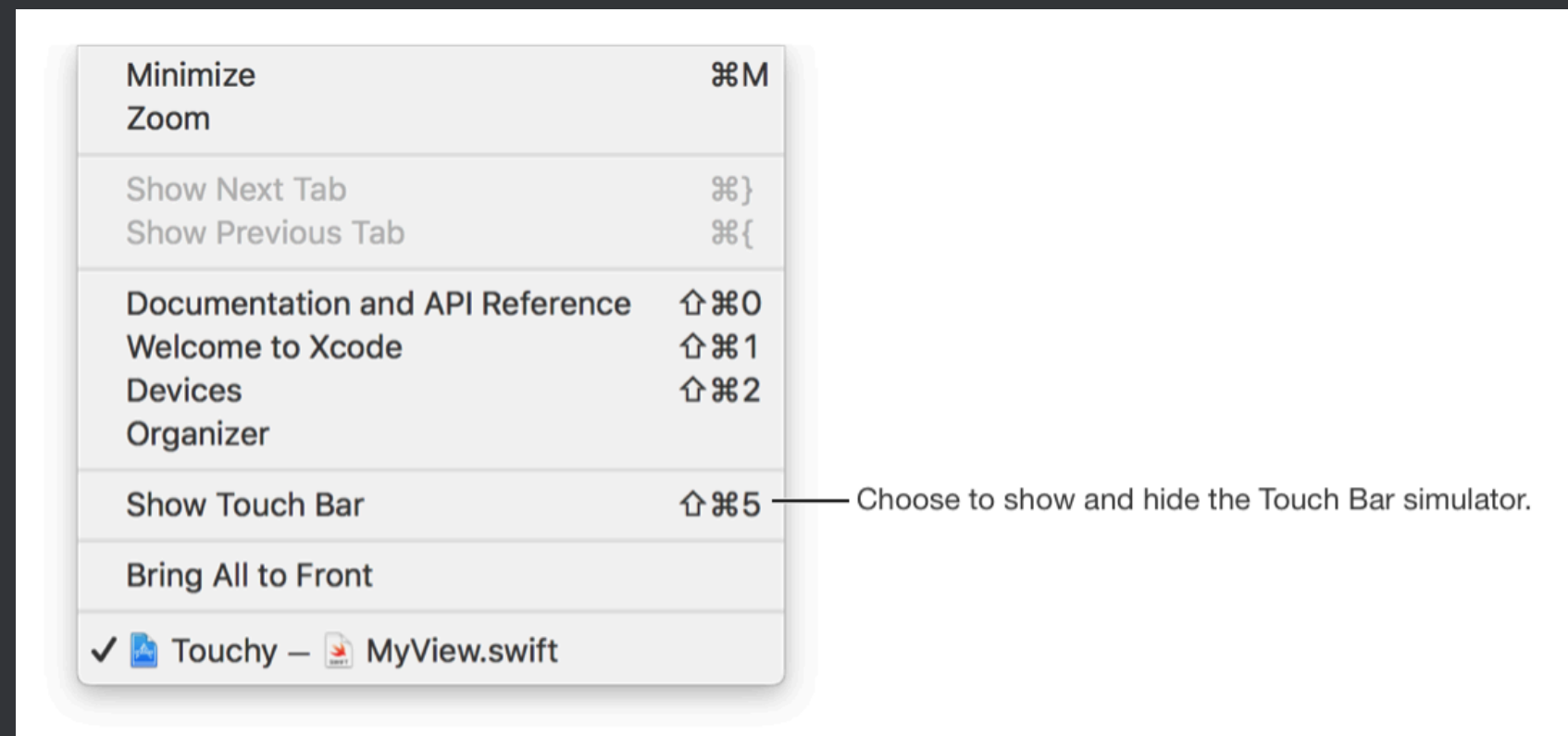
- ⬇ [Xcode 8.1](#)
- ⬇ [macOS 10.12.1 \(Build 16B2657\)](#)
- ☰ [Xcode 8.1 Release Notes](#)
- ☰ [What's New in Xcode](#)

³ <https://developer.apple.com/macos/touch-bar/>

Sample Project is here

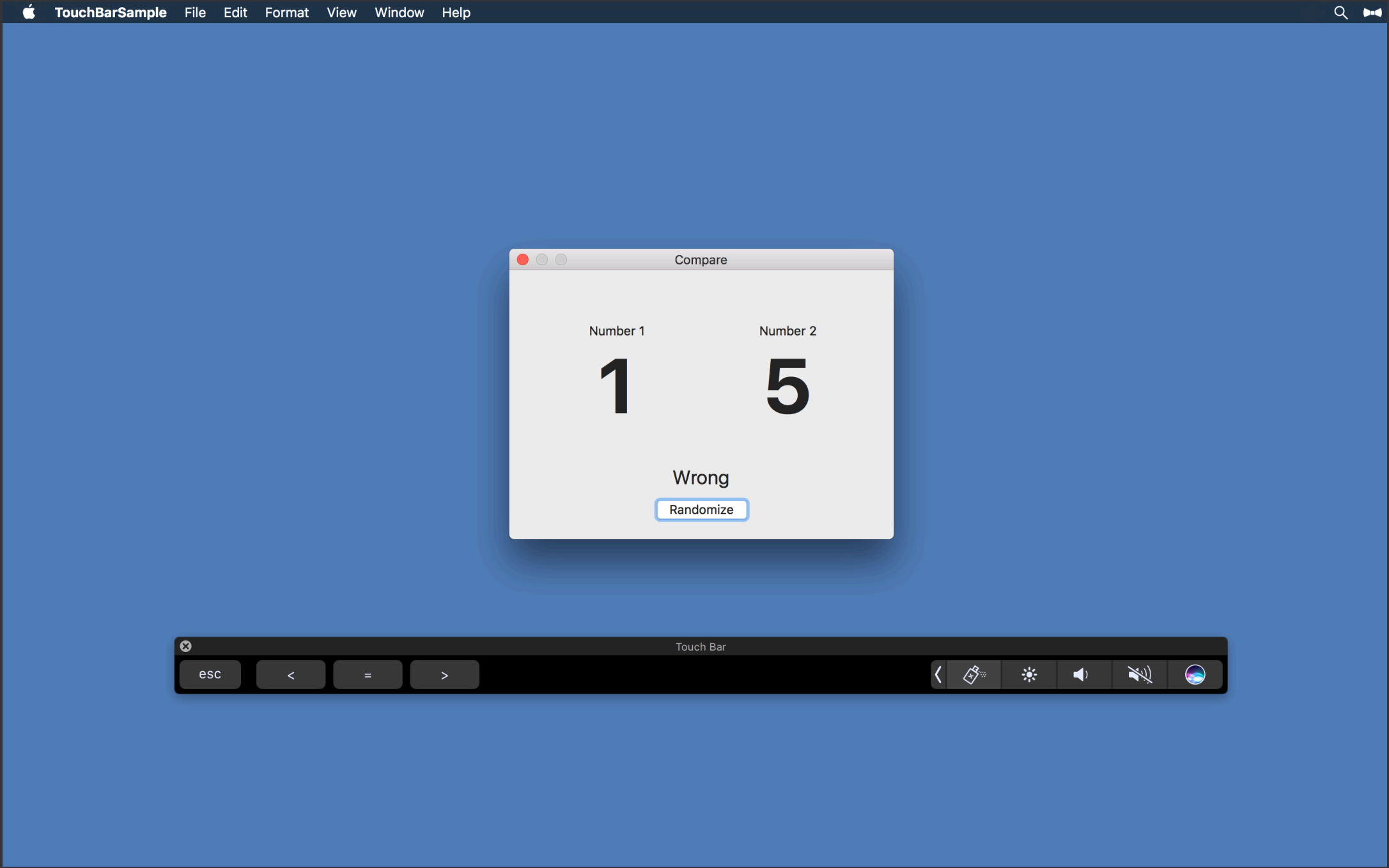
Show Touch Bar Simulator⁴

- Show the Touch Bar simulator by choosing Window > Show Touch Bar (⇧⌘5)



⁴ <https://help.apple.com/xcode/mac/8.1/#/dev7a8cb8a8c>

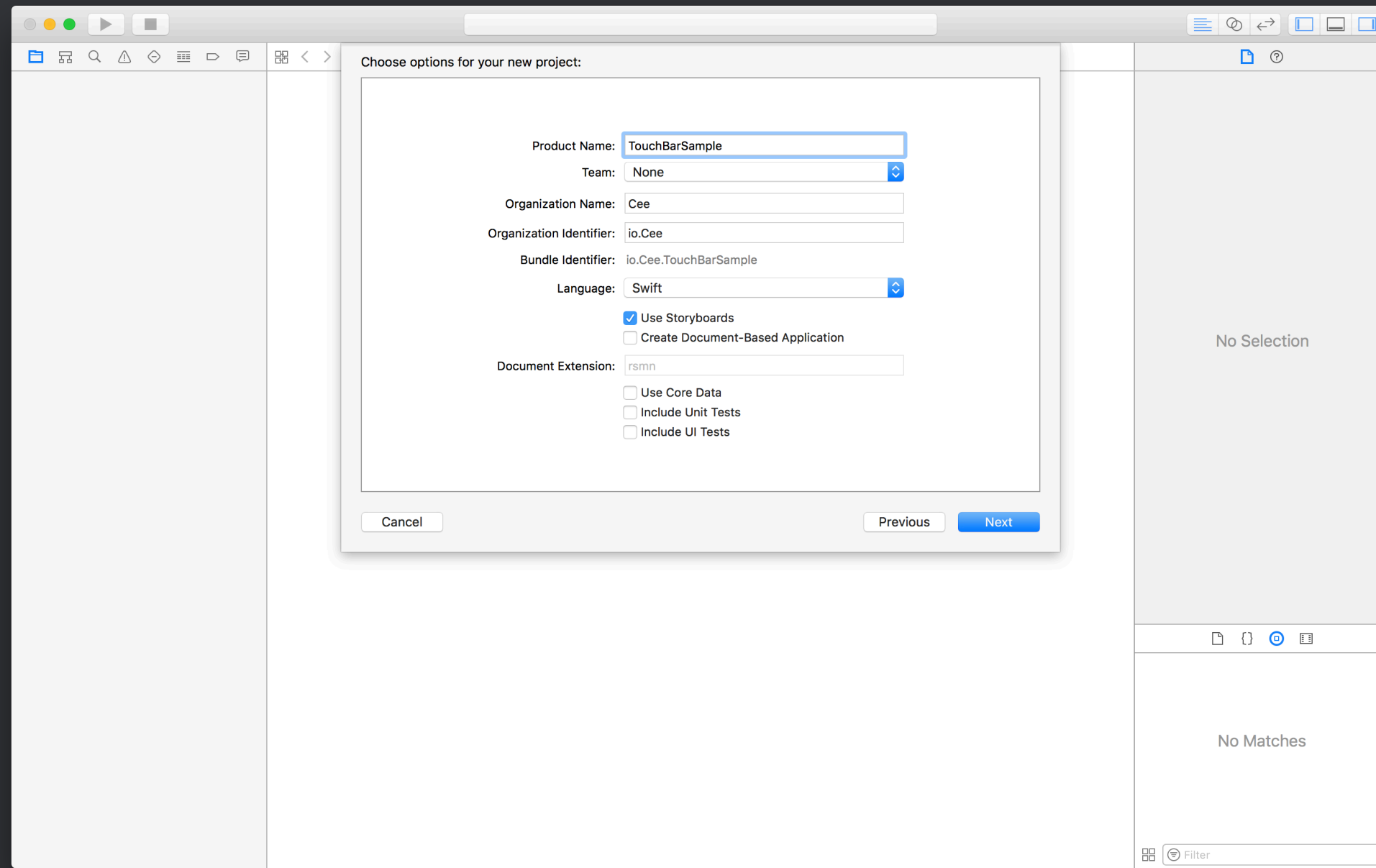
Preview



Project Init

- Create a new Xcode project
- Template > macOS > Cocoa Application > Next
- Product name, organization name, etc.

Project Init (cont.)




Folder Tree

- AppDelegate.swift
- ViewController.swift
- Main.storyboard
- Assets.xcassets
- Info.plist

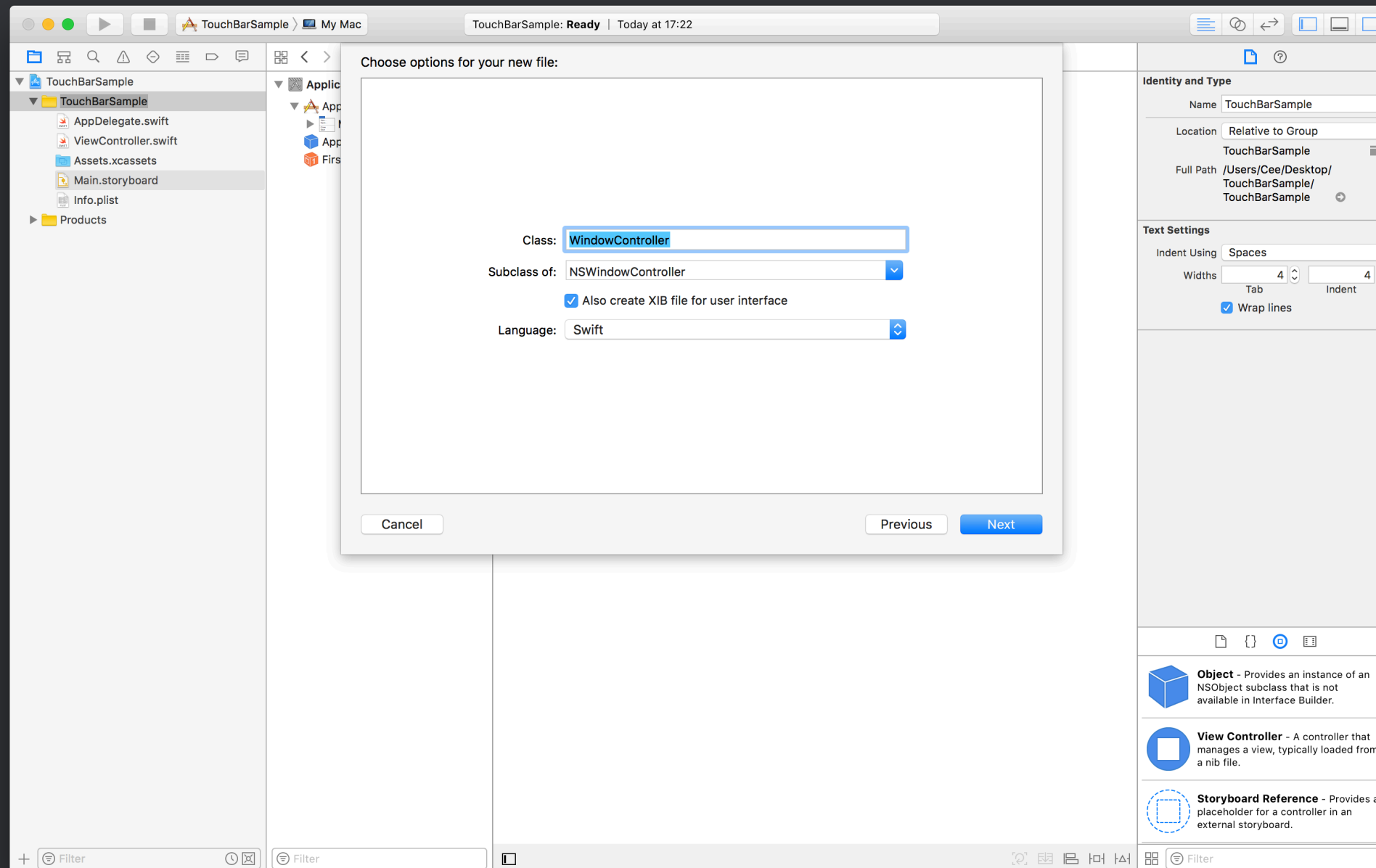
Main.storyboard

- Delete Window Controller Scene
- Delete View Controller Scene

Add our WindowController

- File > New > File... (⌘N)
- Cocoa Class > Next
- Subclass of: **NSWindowController**
-  Also create XIB file for user interface

Add our WindowController (cont.)



Window Controller

- Single-window / Multi-window
- Entrance:
 - AppDelegate.swift
 - application(_:didFinishLaunchingWithOptions:)

AppDelegate - iOS

```
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(application: UIApplication,
                      didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
        window = UIWindow(frame: UIScreen.mainScreen().bounds)

        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        let rootVC = storyboard.instantiateViewControllerWithIdentifier("RootVC")

        window?.rootViewController = rootVC
        window?.makeKeyAndVisible()
        return true
    }
}
```

AppDelegate - macOS

```
import Cocoa
```

```
@NSApplicationMain
```

```
class AppDelegate: NSObject, NSApplicationDelegate {
```

```
    var windowController: NSWindowController?
```

```
    func applicationDidFinishLaunching(_ aNotification: Notification) {  
        let windowController = WindowController(windowNibName: "WindowController")  
        windowController.showWindow(self)  
        self.windowController = windowController  
    }
```

```
}
```

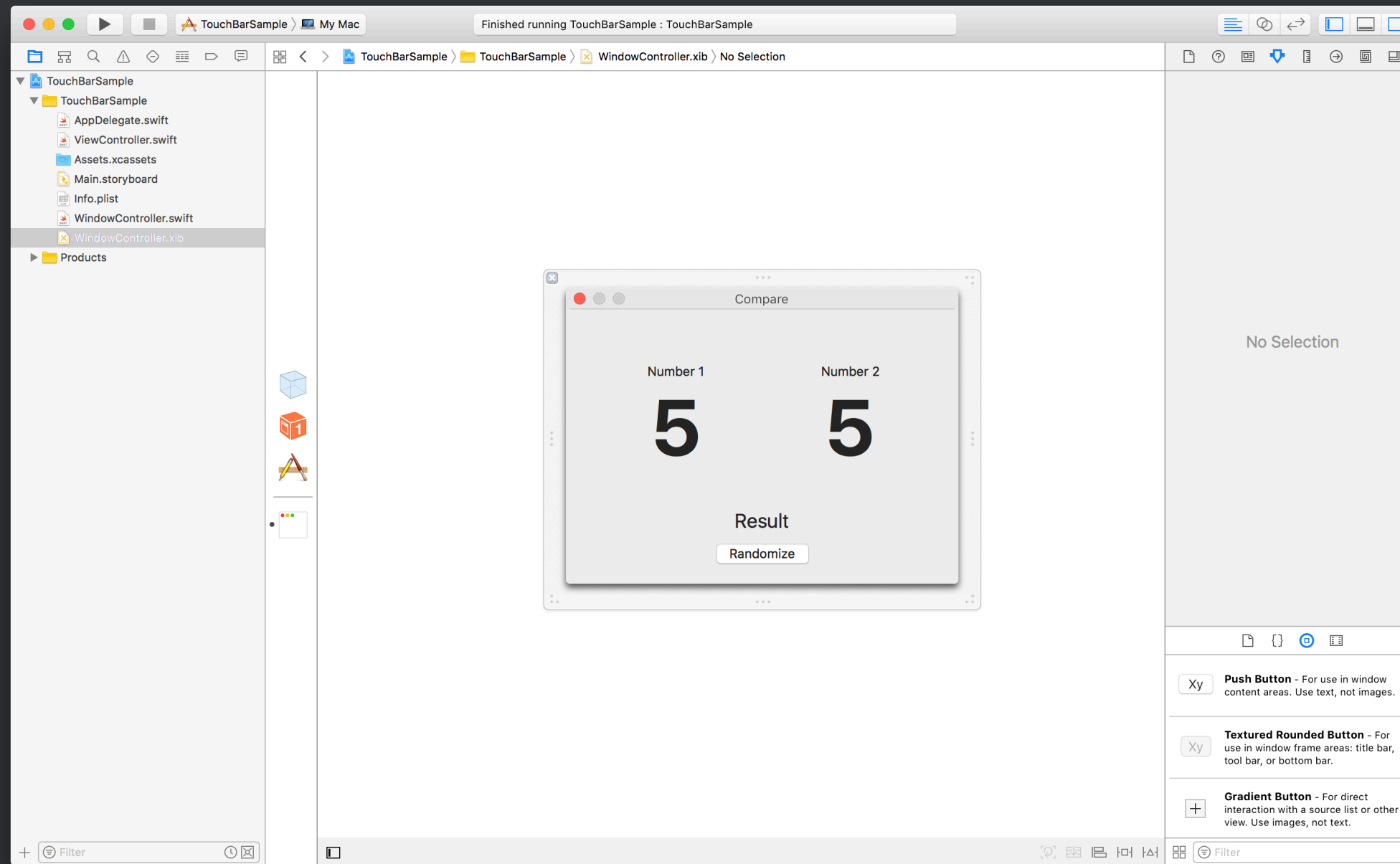

Differences

- `import UIKit` → `import Cocoa`
- UI- prefix → NS- prefix
- Simple enough! 😊

WindowController.xib

- 5 NSTextFields
- 1 NSButton

WindowController.xib (cont.)



Musts Again

1. Be a responder (an instance of an **NSResponder** subclass) that is present within a responder chain at runtime
2. Conform to the **NSTouchBarProvider** protocol
3. Implement the **makeTouchBar()** method within that protocol

Step 1: Become a Responder

- We've created a `WindowController`

```
open class NSWindowController : NSResponder, NSCoding, NSSeguePerforming {  
    public init(window: NSWindow?)  
}
```

- Yes, it's an instance of an `NSResponder` subclass!

Step 2: Conform to the Protocol

```
public protocol NSTouchBarProvider : NSObjectProtocol {  
    @available(OSX 10.12.1, *)  
    public var touchBar: NSTouchBar? { get }  
}
```

```
extension NSResponder : NSTouchBarProvider {  
    @available(OSX 10.12.1, *)  
    open var touchBar: NSTouchBar?  
  
    @available(OSX 10.12.1, *)  
    open func makeTouchBar() -> NSTouchBar?  
}
```

- Yes, it conforms to the `NSTouchBarProvider` protocol!

Step 3: Implement the Method

- Haven't yet! 😅

Make Touch Bar Great Again

- Open `WindowController.swift`
- Code!

Identifiers

```
fileprivate extension NSTouchBarCustomizationIdentifier {  
    static let touchBar = NSTouchBarCustomizationIdentifier("io.Cee.TouchBarSample.touchBar")  
}
```

```
fileprivate extension NSTouchBarItemIdentifier {  
    static let smaller = NSTouchBarItemIdentifier("io.Cee.TouchBarSample.smaller")  
    static let equal = NSTouchBarItemIdentifier("io.Cee.TouchBarSample.equal")  
    static let bigger = NSTouchBarItemIdentifier("io.Cee.TouchBarSample.bigger")  
}
```

Create a Bar

```
// MARK: - NSTouchBar
@available(OSX 10.12.1, *)
override func makeTouchBar() -> NSTouchBar? {
    let touchBar = NSTouchBar()

    touchBar.delegate = self

    touchBar.customizationIdentifier = .touchBar

    touchBar.defaultItemIdentifiers = [.smaller, .equal, .bigger]
    touchBar.customizationAllowedItemIdentifiers = [.smaller, .equal, .bigger]

    return touchBar
}
```

Create Bar Items

```
extension WindowController: NSTouchBarDelegate {
    @available(OSX 10.12.1, *)
    func touchBar(_ touchBar: NSTouchBar, makeItemForIdentifier identifier: NSTouchBarItemIdentifier) -> NSTouchBarItem? {
        let touchBarItem = NSCustomTouchBarItem(identifier: identifier)

        var title: String
        switch identifier {
            case NSTouchBarItemIdentifier.smaller: title = "<"
            case NSTouchBarItemIdentifier.equal: title = "="
            case NSTouchBarItemIdentifier.bigger: title = ">"
            default: title = ""
        }

        let touchBarButton = NSButton(title: title, target: self, action: #selector(compare(with:)))
        touchBarItem.view = touchBarButton

        return touchBarItem;
    }
}
```

Components

```
@IBOutlet weak var numberA: NSTextField!
```

```
@IBOutlet weak var numberB: NSTextField!
```

```
@IBOutlet weak var resultLabel: NSTextField!
```

Compare

```
func compare(with symbol: NSButton) {  
    let number1 = numberA.intValue  
    let number2 = numberB.intValue  
    var result: Bool  
    switch symbol.title {  
        case "<": result = (number1 < number2)  
        case "=": result = (number1 == number2)  
        case ">": result = (number1 > number2)  
        default: result = false  
    }  
    resultLabel.stringValue = (result == true) ? "Correct" : "Wrong"  
}
```

Randomize

```
// MARK: - Button Action
```

```
@IBAction func randomize(_ sender: NSButton) {  
    reset()  
}
```

```
// MARK: - Private Method
```

```
func reset() {  
    numberA.intValue = randomAInt()  
    numberB.intValue = randomAInt()  
    resultLabel.stringValue = ""  
}
```

```
func randomAInt() -> Int32 {  
    return Int32(arc4random_uniform(10))  
}
```

Initialization

```
override func windowDidLoad() {  
    super.windowDidLoad()  
    reset()  
}
```

Where to Go From Here

- How to write a macOS app
- Cocoa programming

About Me

- 王天宇 a.k.a. Cee
- Serve at: Raven Lab
- Mail: i@cee.moe
- GitHub: [Cee](#)
- Twitter: [Ceecirno](#)
- Weibo: [毫无存在感的Cee](#)



**Thank
You For
Listening**