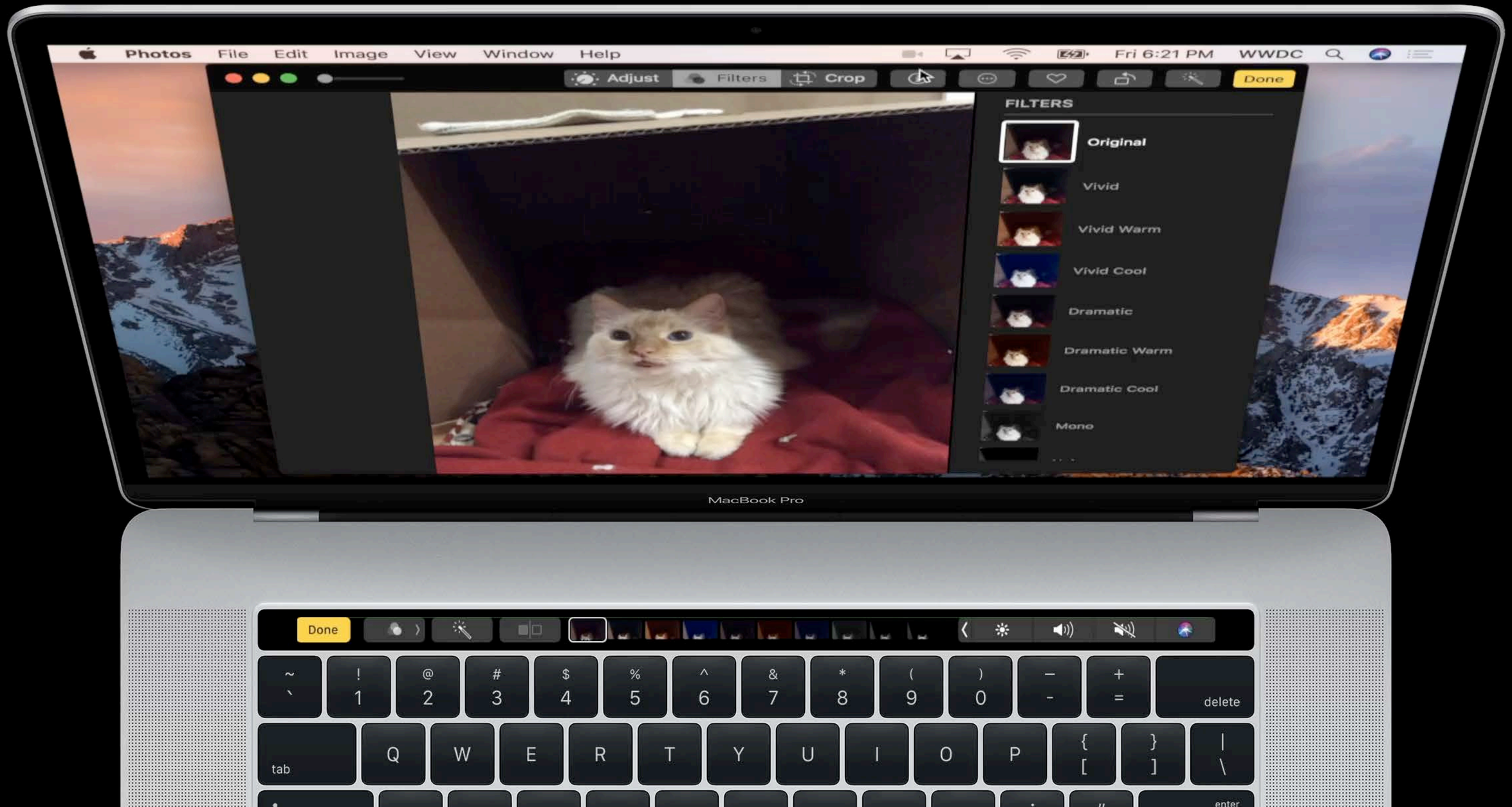
# Touch Bar Fundamentals

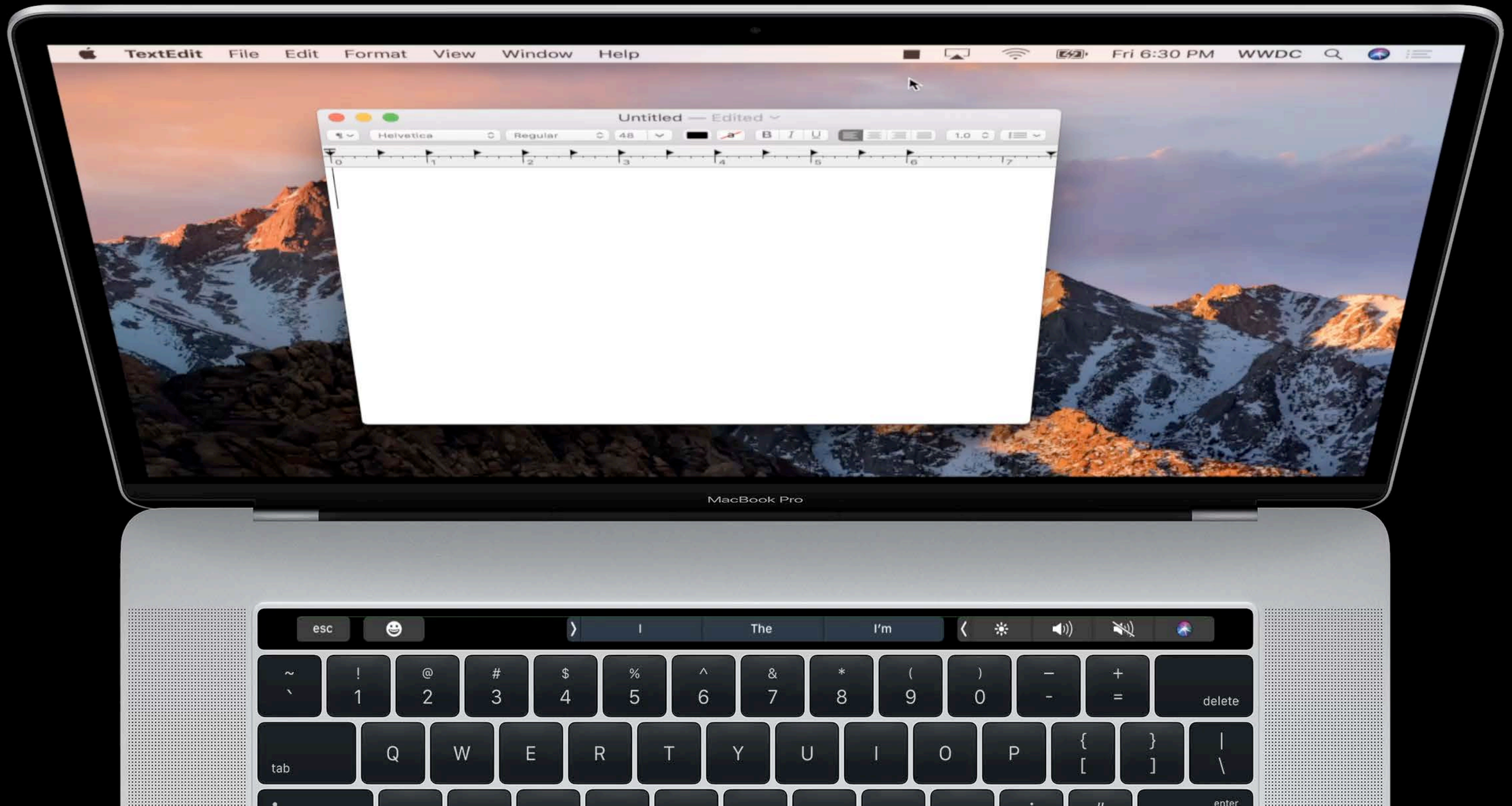
Session 211

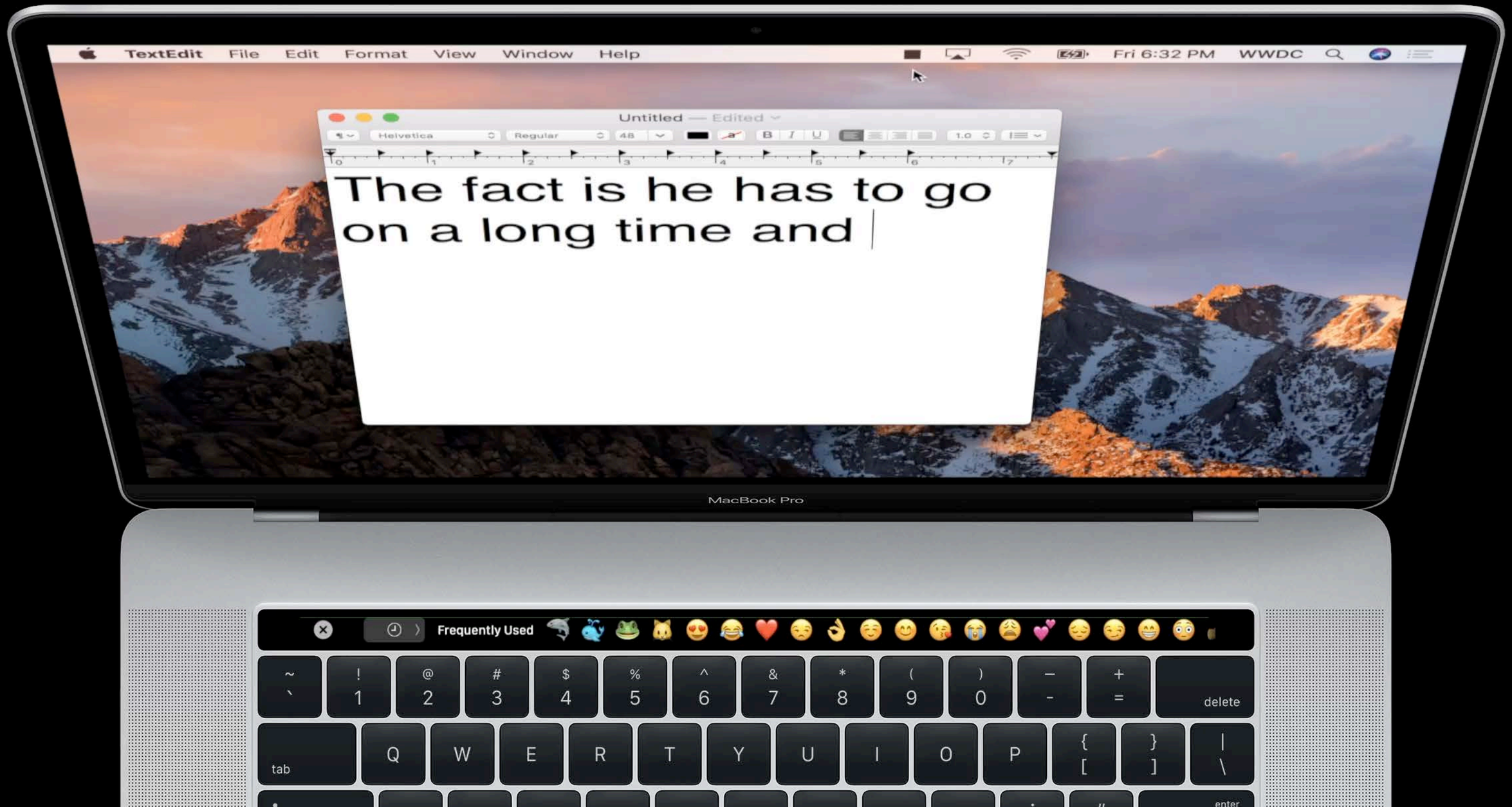
Chris Dreessen
John Tegtmeyer

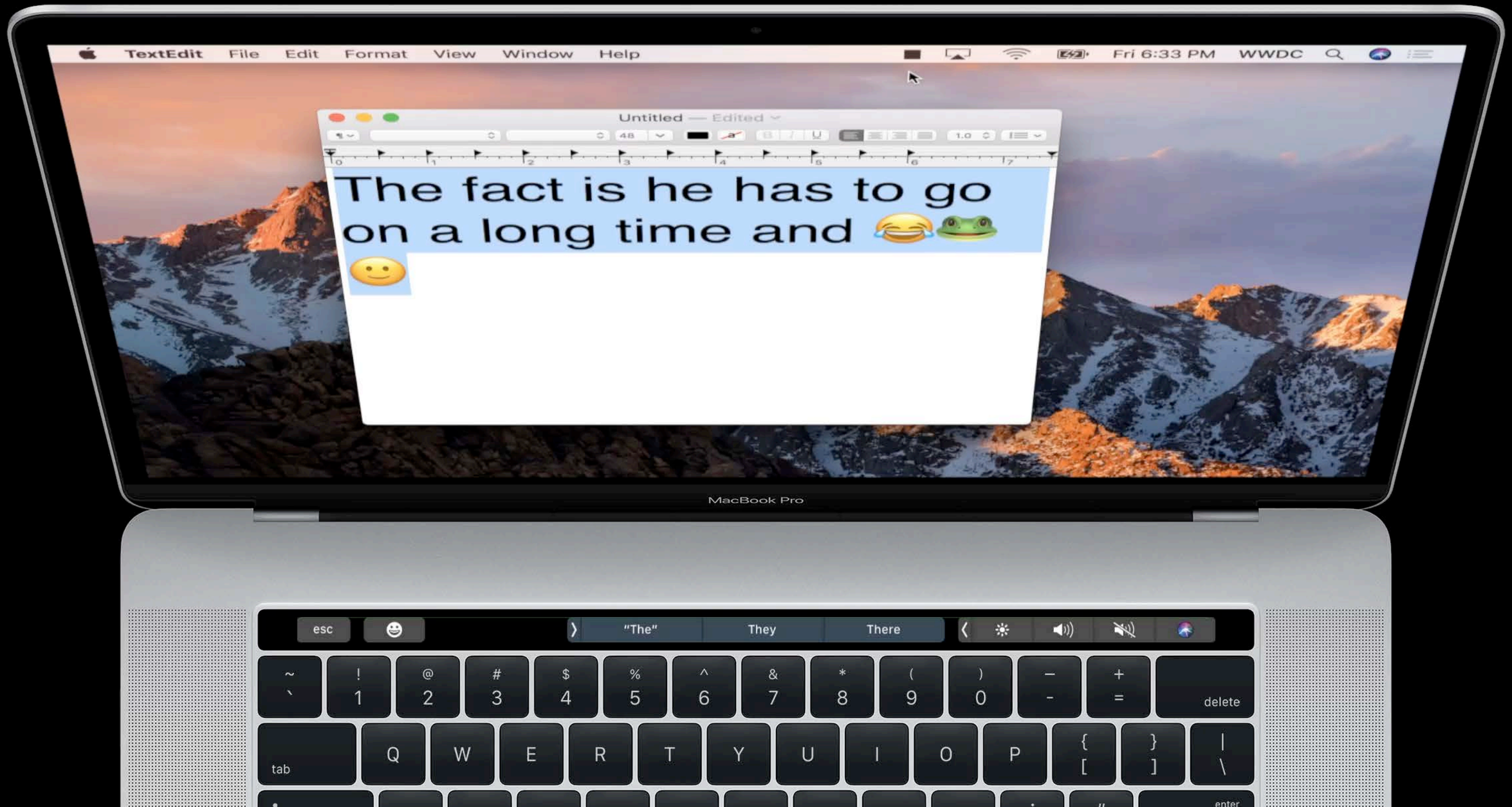

Finder File Edit View Go Window Help
Fri 6:16 PM WWDC
MacBook Pro
esc
delete
tab
Q W E R T Y U I O P
enter


Photos File Edit Image View Window Help
Fri 6:21 PM WWDC
Adjust Filters Crop Done
FILTERS
Original
Vivid
Vivid Warm
Vivid Cool
Dramatic
Dramatic Warm
Dramatic Cool
Mono
MacBook Pro
Done
delete
tab


QuickTime Player
File
Edit
View
Window
Help
Fri 6:24 PM
WWDC
b4-3.mov
Trim
Cancel
MacBook Pro
Cancel
Trim
delete
tab
Q
W
E
R
T
Y
U
I
O
P


TextEdit File Edit Format View Window Help
Fri 6:30 PM WWDC
Untitled — Edited
Helvetica
Regular
48
MacBook Pro
esc
I
The
I'm
delete
tab
Q W E R T Y U I O P
enter


TextEdit File Edit Format View Window Help
Fri 6:32 PM WWDC
Untitled — Edited
Helvetica Regular 48
The fact is he has to go on a long time and
MacBook Pro
Frequently Used
delete
tab
enter


TextEdit File Edit Format View Window Help
Fri 6:33 PM WWDC
Untitled — Edited
The fact is he has to go on a long time and
MacBook Pro
esc
"The" They There
delete
tab


Drag your favorite items to the bottom of the screen into the Touch Bar...
Show typing suggestions
Done
Default Set
Back/Forward
View Options
Quick Look
Share
Edit Tags
New Folder
New Folder with Selection
New Window
New Tab
Move To Trash
Connect To Server
Open With...
Open With...
Go To...
Go To
Move To
Copy To
View
Get Info
Space
MacBook Pro
Done
delete
tab
Q W E R T Y U I O P
enter

# Specifications

P3 retina display

Multi-Touch

Context sensitivity

Adaptive whitepoint and brightness

# Design Goals

Input device

Don’t hide functionality

Don’t distract

Core classes

Responder chain

NSTouchBarItem in depth

Core classes

Responder chain

NSTouchBarItem in depth

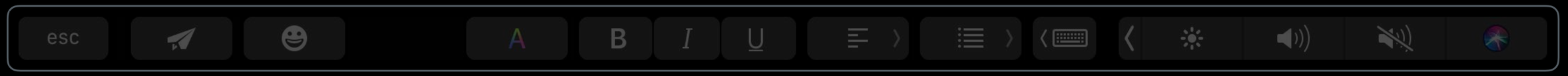
# Core Classes

NSTouchBarItem

NSTouchBar

NSResponder

# NSTouchBarItem



# NSTouchBarItem

# NSTouchBarItem

Attaches a view to the Touch Bar

Intended to be subclassed

Has a required identifier

# NSTouchBar

Has array of item identifiers

```
// Creating an NSTouchBar

let sharkIdentifier = NSTouchBarItem.Identifier(“shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
```

```
// Creating an NSTouchBar

let sharkIdentifier = NSTouchBarItem.Identifier(“shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
```

```
// Creating an NSTouchBar

let sharkIdentifier = NSTouchBarItem.Identifier(“shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
```

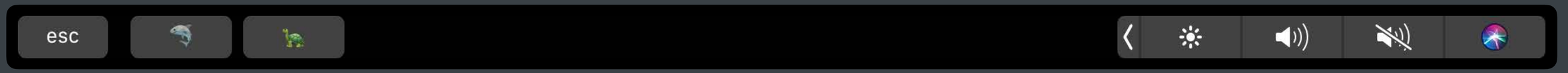
# NSTouchBar

Has array of item identifiers

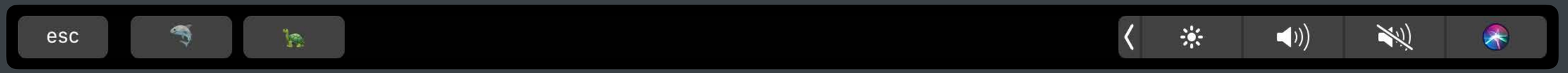
Looks up items by identifiers

- `templateItems`
- `delegate`

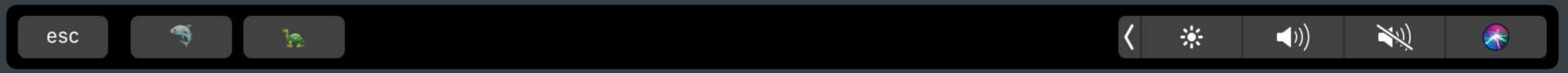
```
// Using NSTouchBar.templateItems

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)

let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🦕", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
```



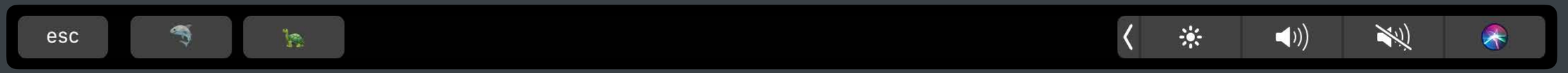
```
// Using NSTouchBar.templateItems

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)

let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🦕", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
```

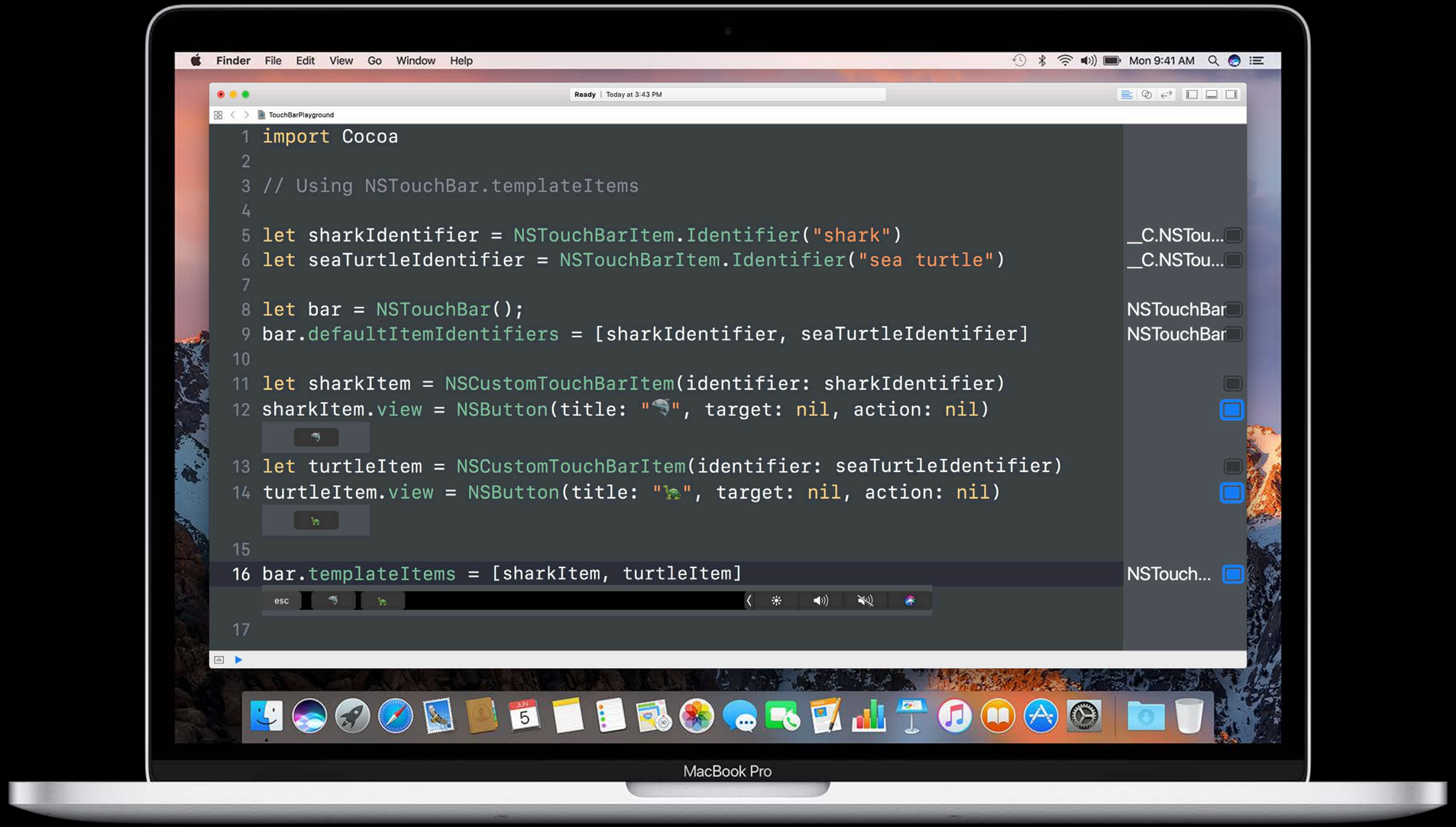


```
// Using NSTouchBar.templateItems

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)

let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🐢", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
```



```
// Using NSTouchBar.templateItems

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle”)

let bar = NSTouchBar()
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)

let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🐢", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
```




Finder File Edit View Go Window Help
Mon 9:41 AM
Ready | Today at 3:43 PM
TouchBarPlayground
1 import Cocoa
2
3 // Using NSTouchBar.templateItems
4
5 let sharkIdentifier = NSTouchBarItem.Identifier("shark")
6 let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle")
7
8 let bar = NSTouchBar();
9 bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
10
11 let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
12 sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)
13 let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
14 turtleItem.view = NSButton(title: "🐢", target: nil, action: nil)
15
16 bar.templateItems = [sharkItem, turtleItem]
17
__C.NSTou...
__C.NSTou...
NSTouchBar
NSTouchBar
NSTouch...
esc
MacBook Pro

```
// Implementing an NSTouchBarDelegate

class MyTouchBarDelegate: NSObject, NSTouchBarDelegate {
    func touchBar(_ touchBar: NSTouchBar, makeItemForIdentifier identifier:
NSTouchBarItem.Identifier) -> NSTouchBarItem? {
        switch identifier {
        case sharkIdentifier:
            let item = NSCustomTouchBarItem(identifier: identifier)
            item.view = NSButton(title: "🦈", target: nil, action: nil)
            return item
        case seaTurtleIdentifier:
            let item = NSCustomTouchBarItem(identifier: identifier)
            item.view = NSButton(title: "🐢", target: nil, action: nil)
            return item
        default:
            return nil
        }
    }
}
```

```
// Implementing an NSTouchBarDelegate

class MyTouchBarDelegate: NSObject, NSTouchBarDelegate {
    func touchBar(_ touchBar: NSTouchBar, makeItemForIdentifier identifier:
NSTouchBarItem.Identifier) -> NSTouchBarItem? {
        switch identifier {
        case sharkIdentifier:
            let item = NSCustomTouchBarItem(identifier: identifier)
            item.view = NSButton(title: "🦈", target: nil, action: nil)
            return item
        case seaTurtleIdentifier:
            let item = NSCustomTouchBarItem(identifier: identifier)
            item.view = NSButton(title: "🐢", target: nil, action: nil)
            return item
        default:
            return nil
        }
    }
}
```

```
// Implementing an NSTouchBarDelegate

class MyTouchBarDelegate: NSObject, NSTouchBarDelegate {
    func touchBar(_ touchBar: NSTouchBar, makeItemForIdentifier identifier:
NSTouchBarItem.Identifier) -> NSTouchBarItem? {
        switch identifier {
        case sharkIdentifier:
            let item = NSCustomTouchBarItem(identifier: identifier)
            item.view = NSButton(title: "🦈", target: nil, action: nil)
            return item
        case seaTurtleIdentifier:
            let item = NSCustomTouchBarItem(identifier: identifier)
            item.view = NSButton(title: "🐢", target: nil, action: nil)
            return item
        default:
            return nil
        }
    }
}
```

# NSTouchBar

Has array of item identifiers

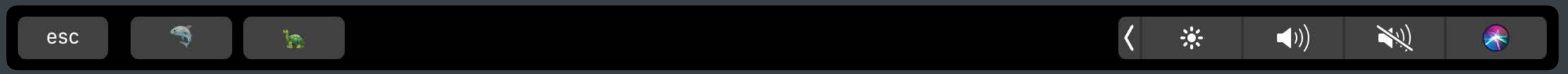
Looks up items by identifiers

- `templateItems`
- `delegate`

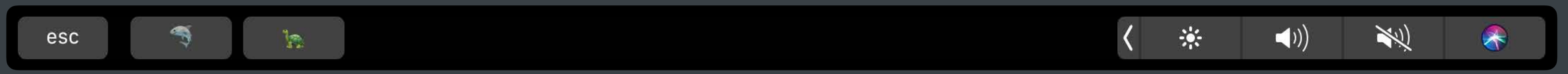
Customizable

```
// Making an NSTouchBar customizable
```

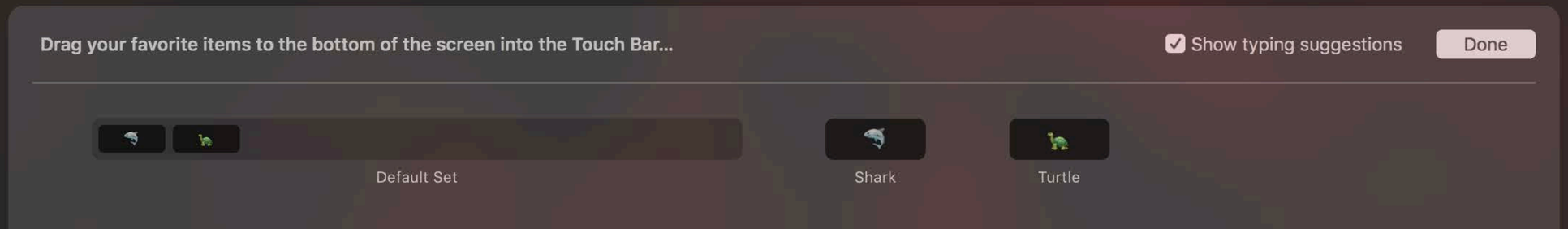
```
bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
```



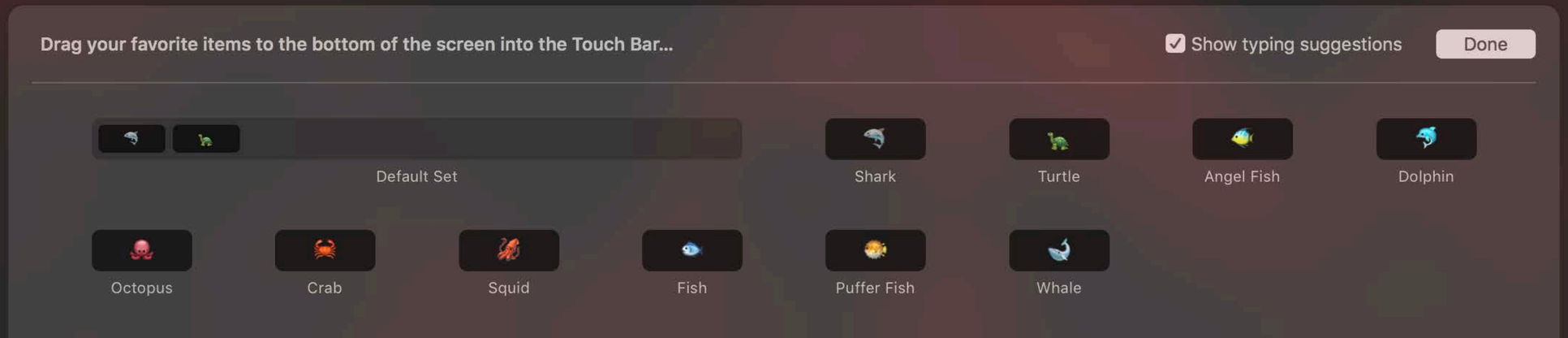
```
// Making an NSTouchBar customizable

NSApp.isAutomaticCustomizeTouchBarMenuItemEnabled = true

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
bar.customizationIdentifier = NSTouchBar.CustomizationIdentifier("ocean animals")
bar.customizationAllowedItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
```



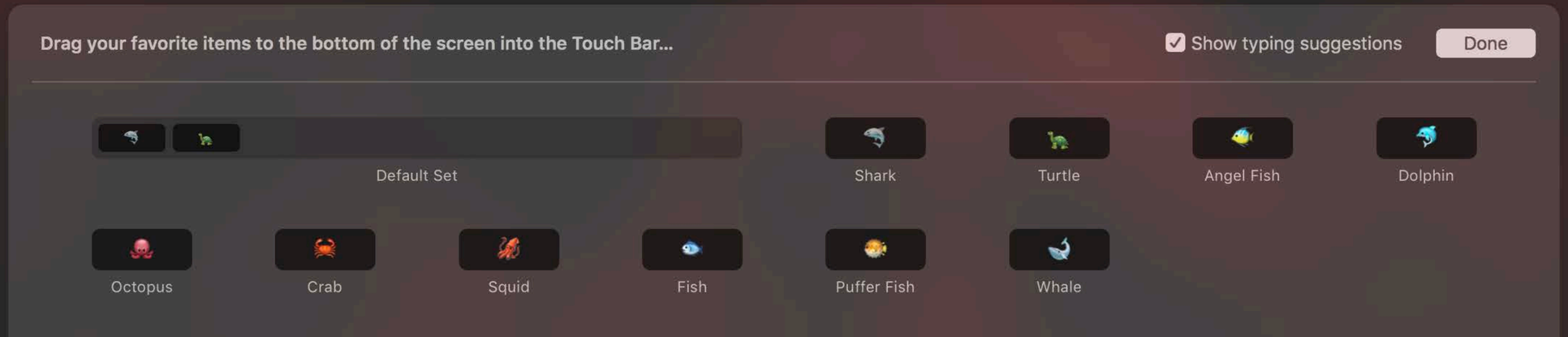
```
// Making an NSTouchBar customizable

NSApp.isAutomaticCustomizeTouchBarMenuItemEnabled = true

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
bar.customizationIdentifier = NSTouchBar.CustomizationIdentifier("ocean animals")
bar.customizationAllowedItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
```



```
// Making an NSTouchBar customizable

NSApp.isAutomaticCustomizeTouchBarMenuItemEnabled = true

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
bar.customizationIdentifier = NSTouchBar.CustomizationIdentifier("ocean animals")
bar.customizationAllowedItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier,
angelFishIdentifier, dolphinIdentifier, octopusIdentifier, crabIdentifier, squidIdentifier,
fishIdentifier, pufferFishIdentifier, whaleIdentifier]
```



```
// Making an NSTouchBar customizable

NSApp.isAutomaticCustomizeTouchBarMenuItemEnabled = true

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
bar.customizationIdentifier = NSTouchBar.CustomizationIdentifier("ocean animals")
bar.customizationAllowedItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier,
angelFishIdentifier, dolphinIdentifier, octopusIdentifier, crabIdentifier, squidIdentifier,
fishIdentifier, pufferFishIdentifier, whaleIdentifier]
bar.customizationRequiredItemIdentifiers = [seaTurtleIdentifier]
```



# NSTouchBarProvider Protocol

Connects NSTouchBars to the Touch Bar

Implemented by delegates and responders

```
extension NSTouchBarProvider : NSObject {
    let touchBar: NSTouchBar?
}
```

```
// Providing an NSTouchBar from an application delegate using the NSTouchBarProvider protocol

class ApplicationDelegate: NSObject, NSApplicationDelegate, NSTouchBarProvider {
    lazy var touchBar: NSTouchBar? = {
        let item = NSCustomTouchBarItem(identifier:NSTouchBarItem.Identifier("shark"))
        item.view = NSButton(title: "🦈", target: nil, action: nil)

        let bar = NSTouchBar()
        bar.defaultItemIdentifiers = [NSTouchBarItem.Identifier("shark")]
        bar.templateItems = [item]

        return bar
    }()
}
```

```
// Providing an NSTouchBar from an application delegate using the NSTouchBarProvider protocol

class ApplicationDelegate: NSObject, NSApplicationDelegate, NSTouchBarProvider {
    lazy var touchBar: NSTouchBar? = {
        let item = NSCustomTouchBarItem(identifier:NSTouchBarItem.Identifier("shark"))
        item.view = NSButton(title: "🦈", target: nil, action: nil)

        let bar = NSTouchBar()
        bar.defaultItemIdentifiers = [NSTouchBarItem.Identifier("shark")]
        bar.templateItems = [item]

        return bar
    }()
}
```

# NSResponder

Has a settable `touchBar property`

```
class NSResponder : NSObject, NSTouchBarProvider {
    var touchBar: NSTouchBar?
}
```

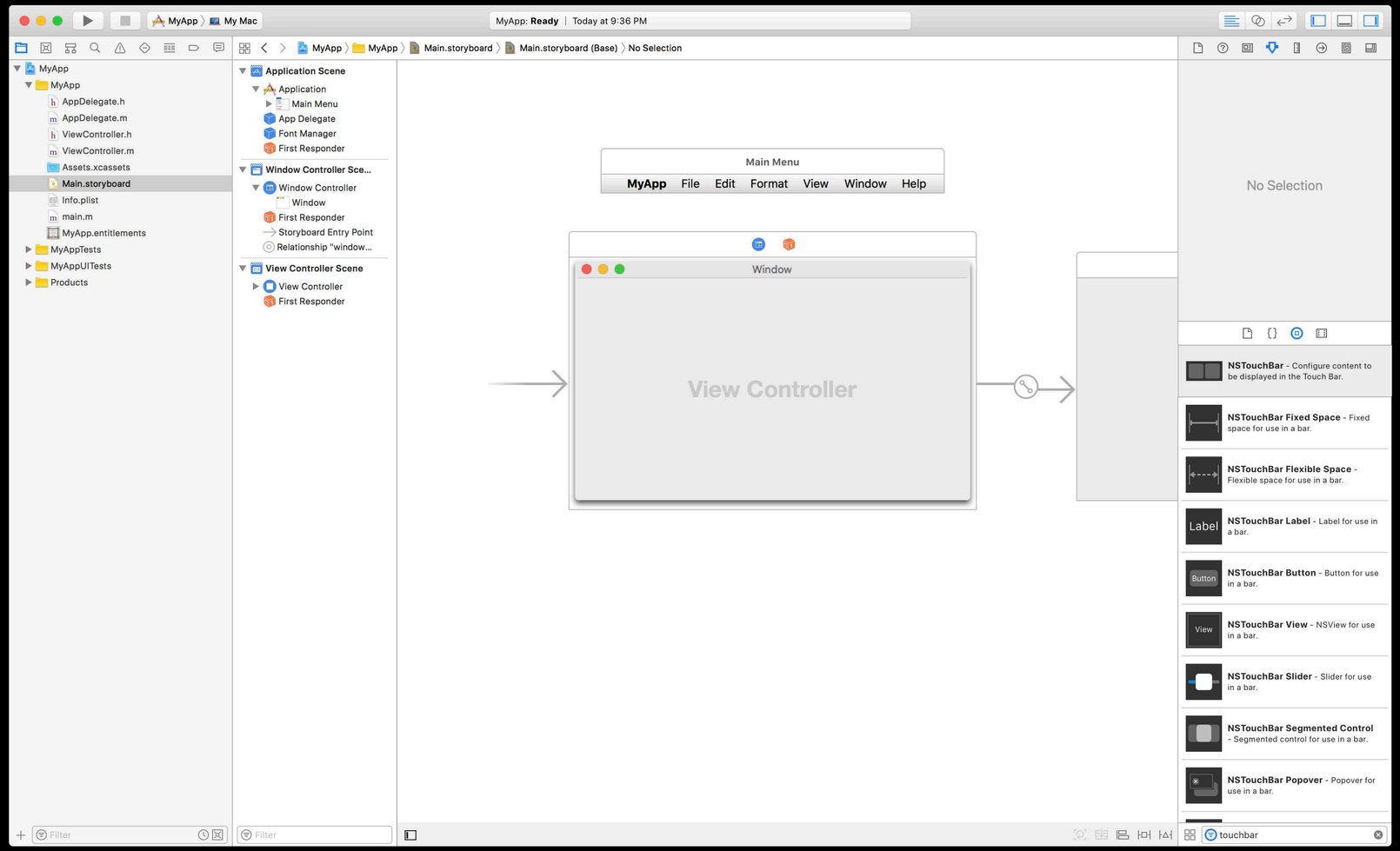
Subclasses can use `.makeTouchBar()`

```
// Providing an NSTouchBar from an NSResponder subclass

class MyWindowController: NSWindowController, NSTouchBarDelegate {
    override func makeTouchBar() -> NSTouchBar? {
        let touchBar = NSTouchBar()
        touchBar.delegate = self
        touchBar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
        return touchBar
    }
}
```

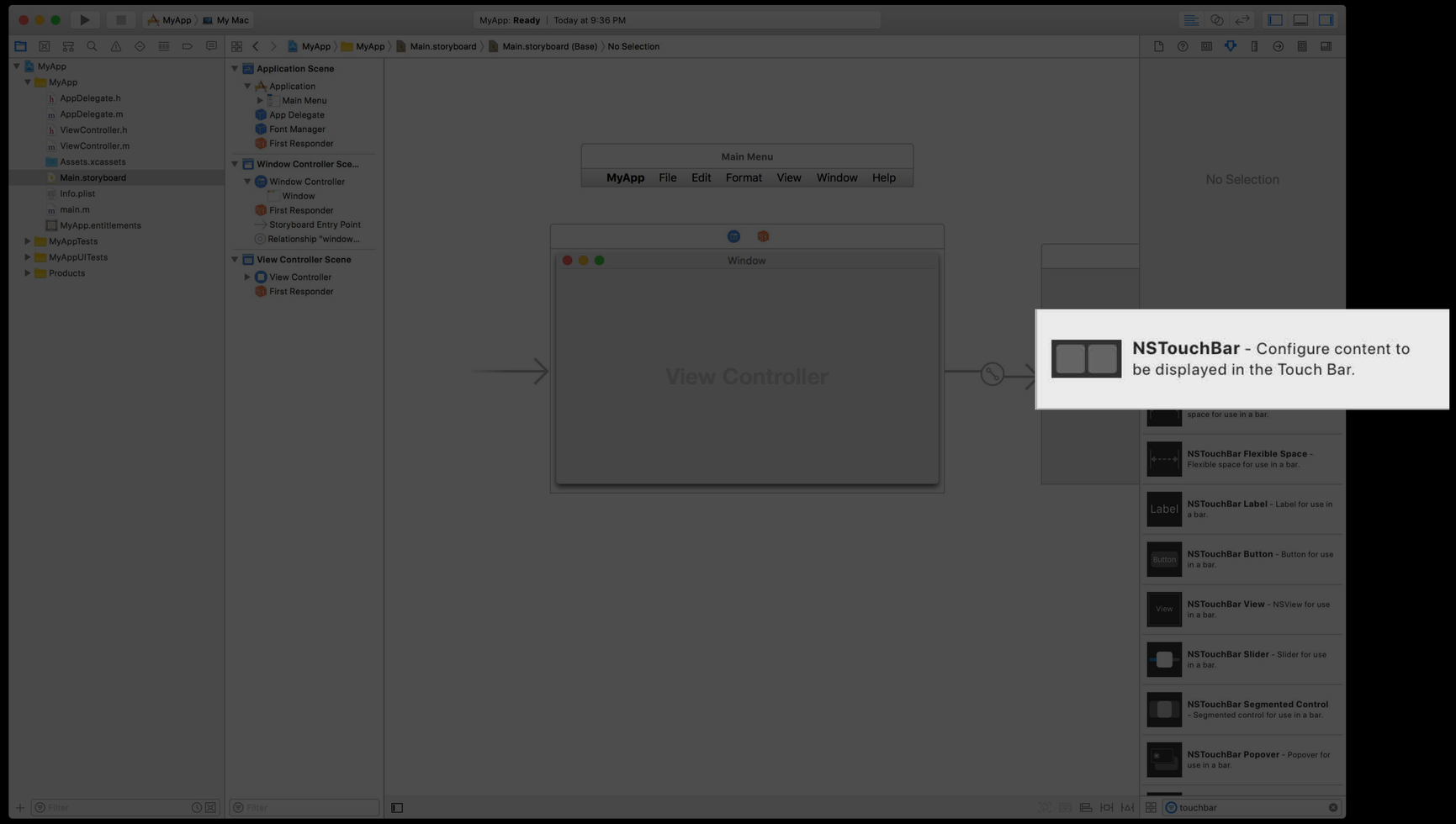
```
// Providing an NSTouchBar from an NSResponder subclass

class MyWindowController: NSWindowController, NSTouchBarDelegate {
    override func makeTouchBar() -> NSTouchBar? {
        let touchBar = NSTouchBar()
        touchBar.delegate = self
        touchBar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]
        return touchBar
    }
}
```

# Interface Builder

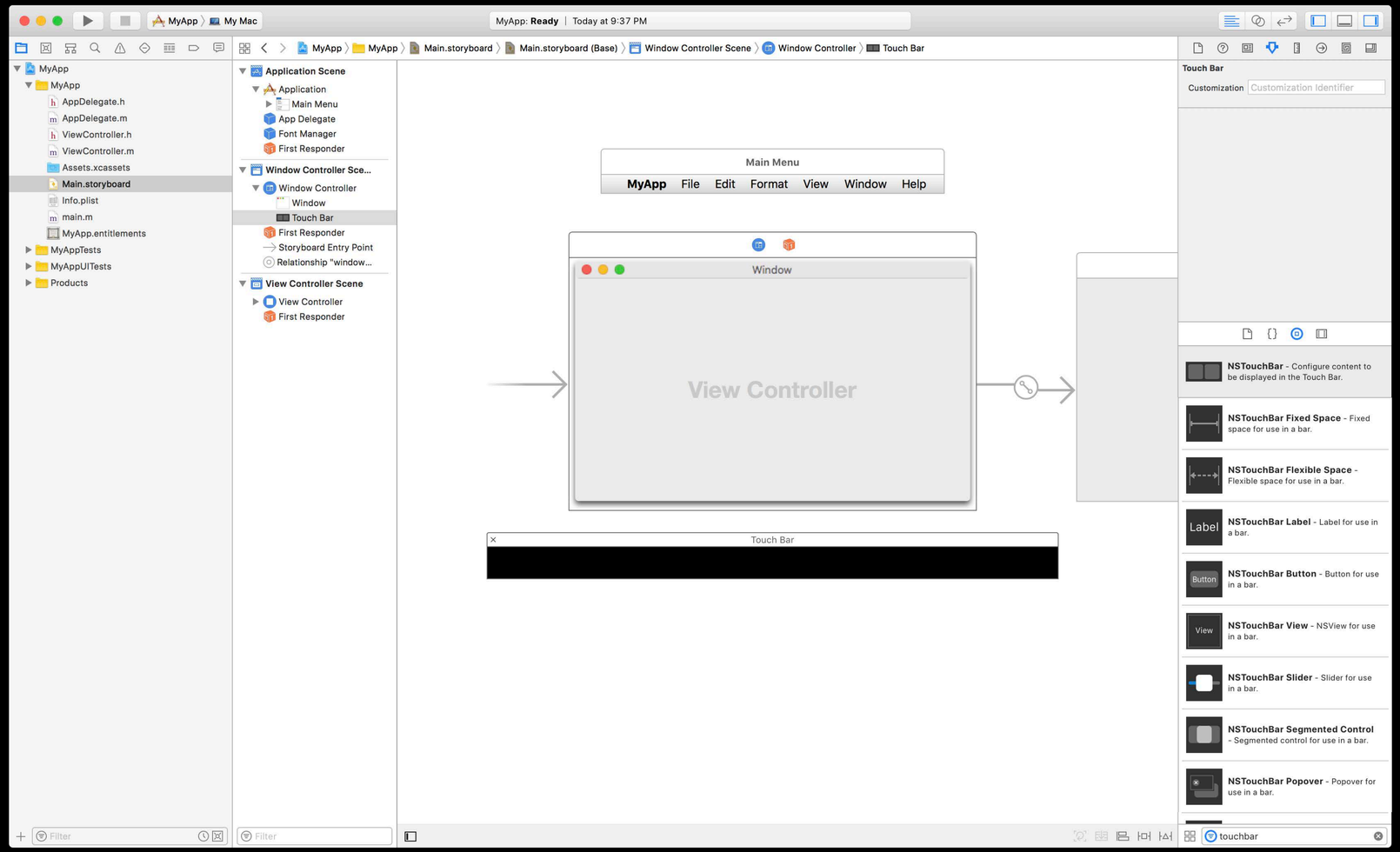
# Interface Builder



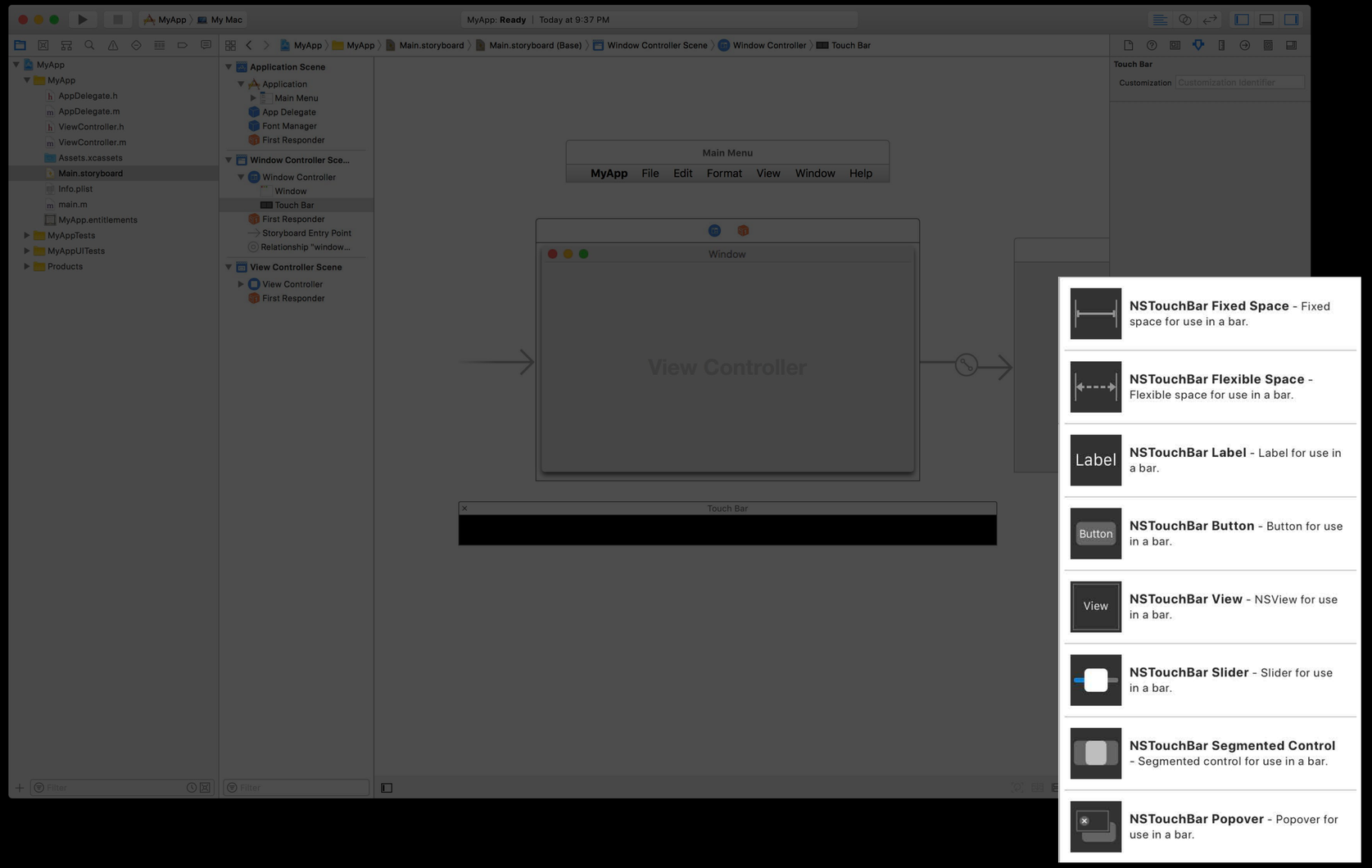
# Interface Builder



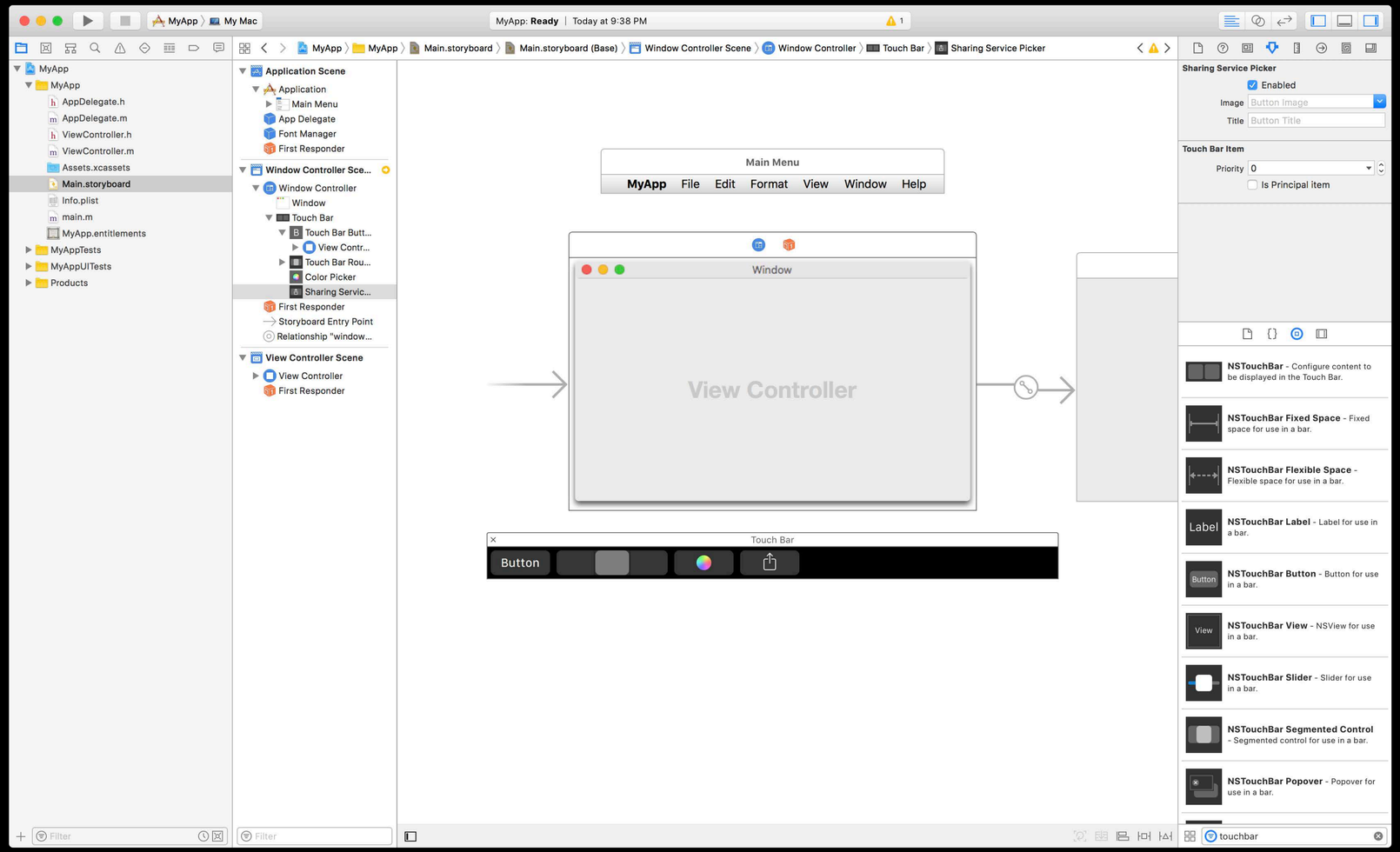
# Interface Builder



# Interface Builder



# Interface Builder



Core classes

Responder chain

NSTouchBarItem in depth

# Responder Chain

NSResponder

`nextResponder` property

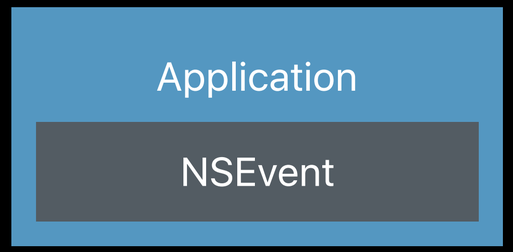
Linked to form chains

# Responder Chain

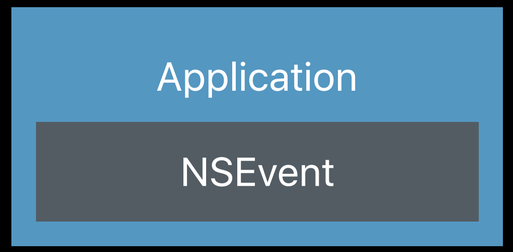
NSEvent

# Responder Chain

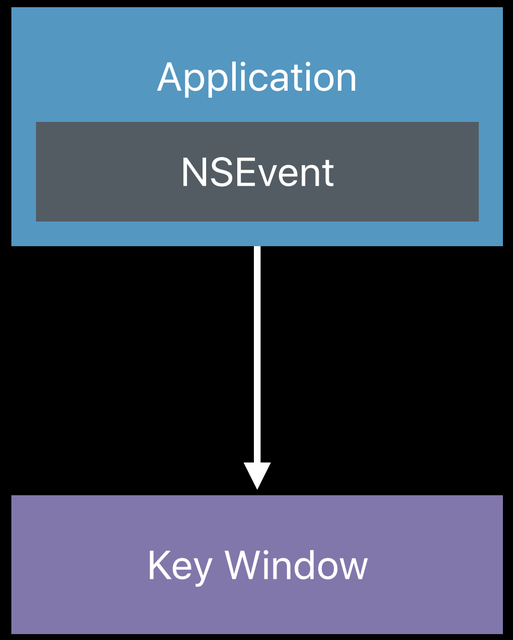
# Responder Chain



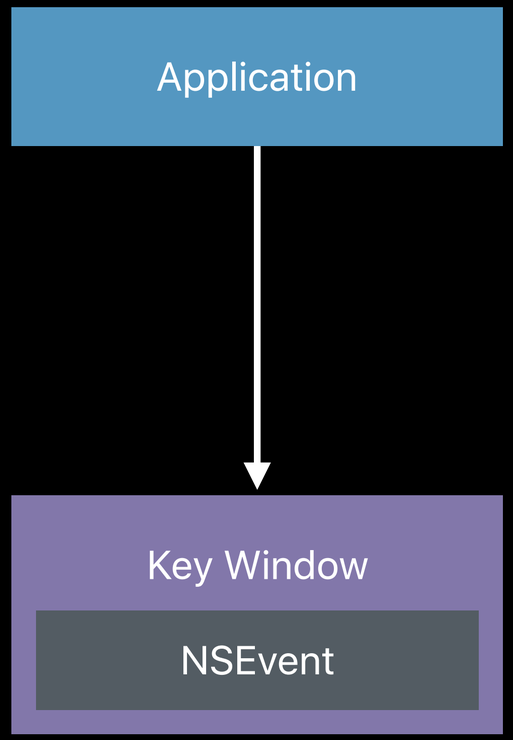
# Responder Chain



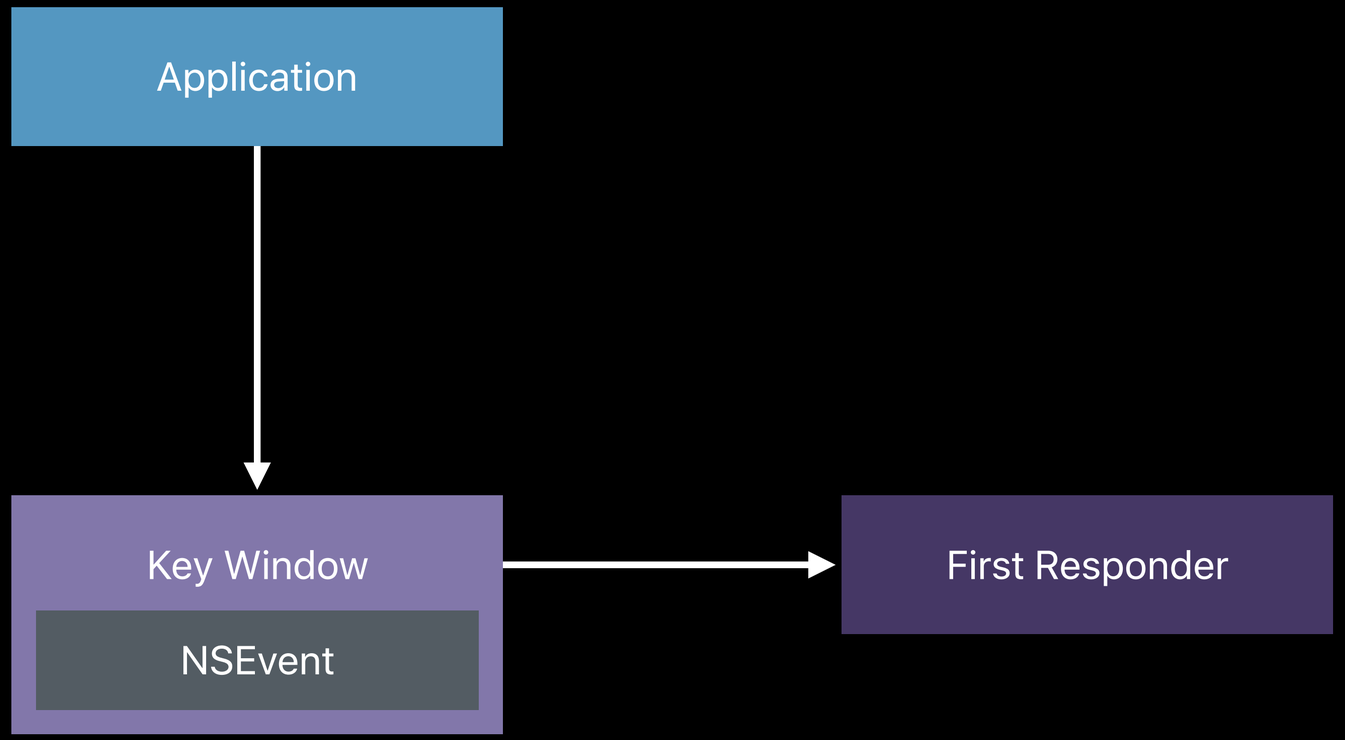
# Responder Chain



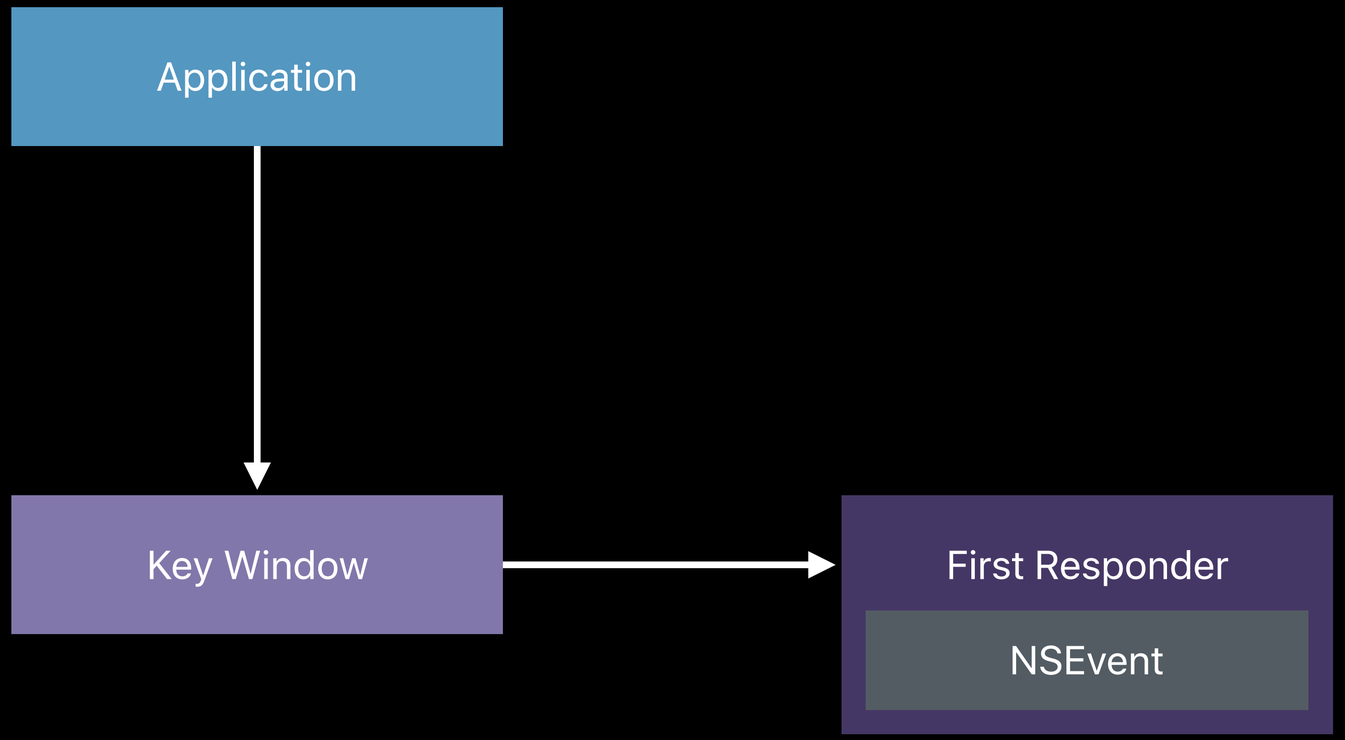
# Responder Chain



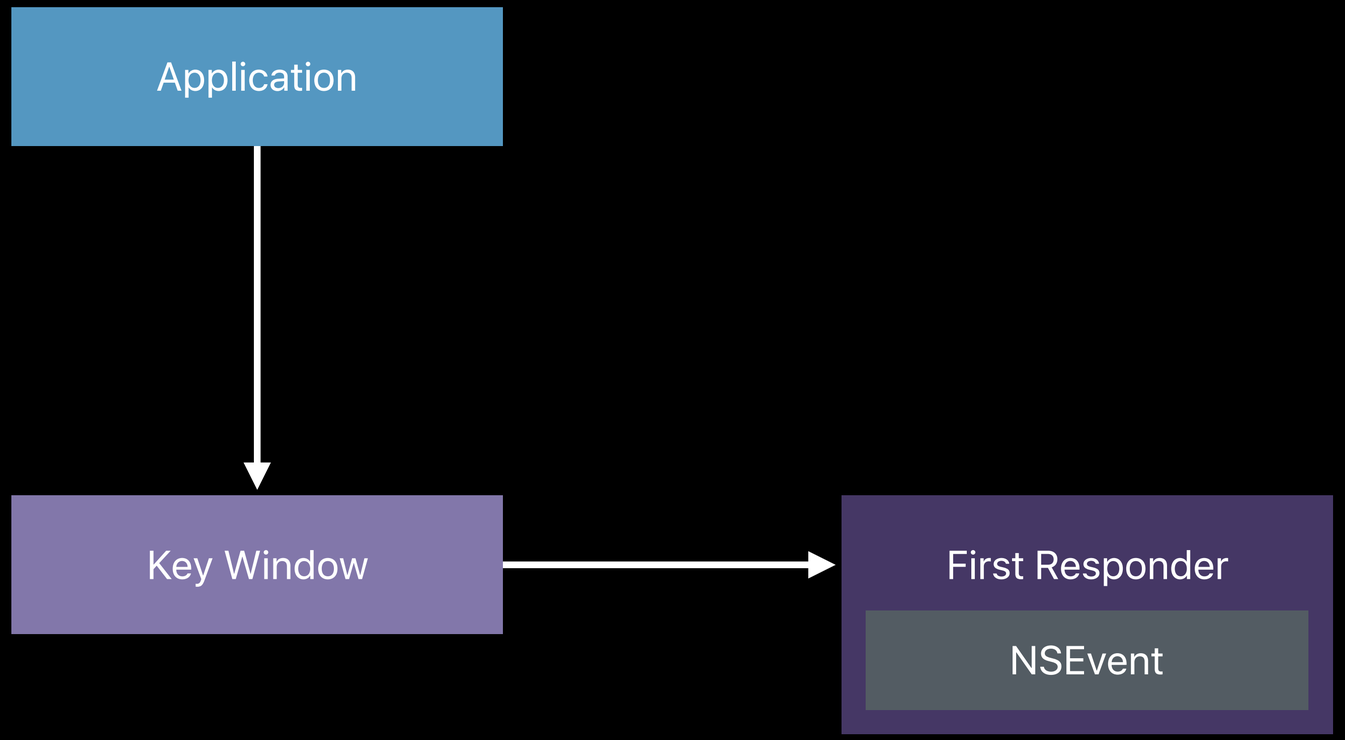
# Responder Chain



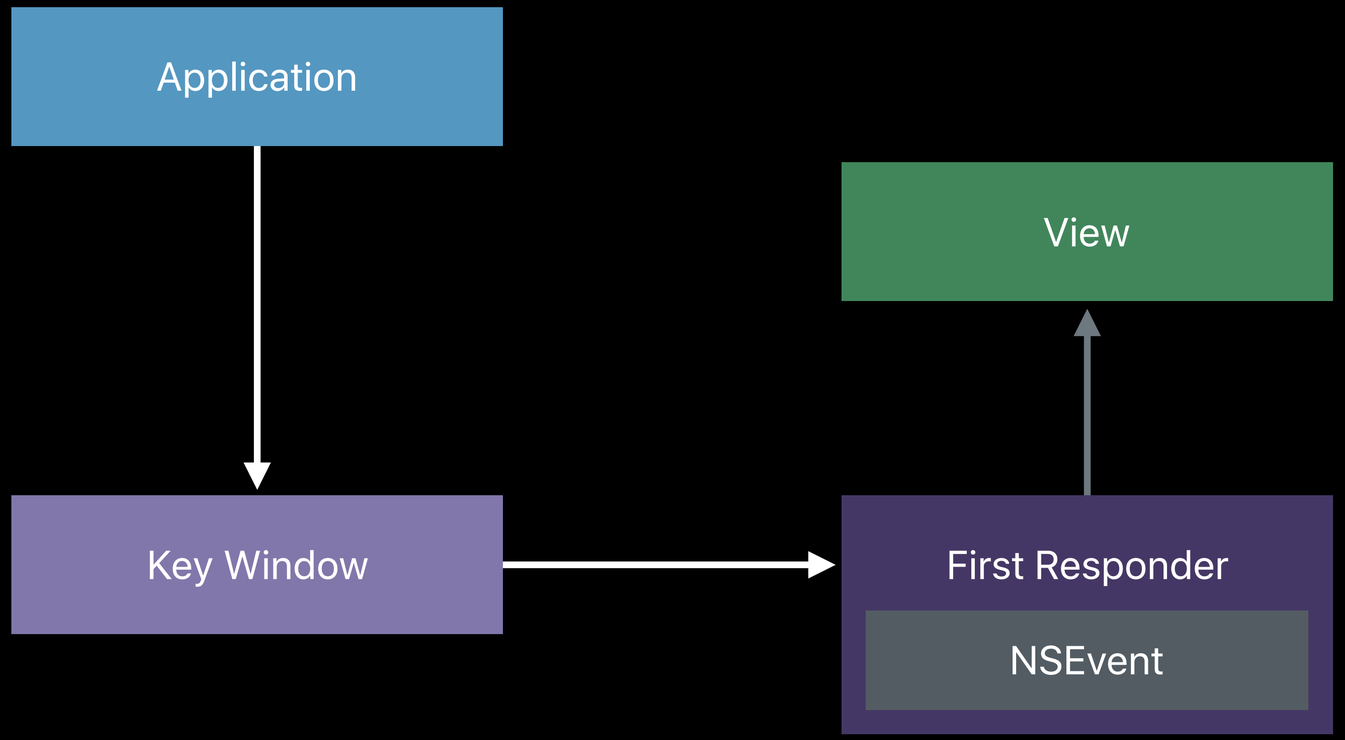
# Responder Chain



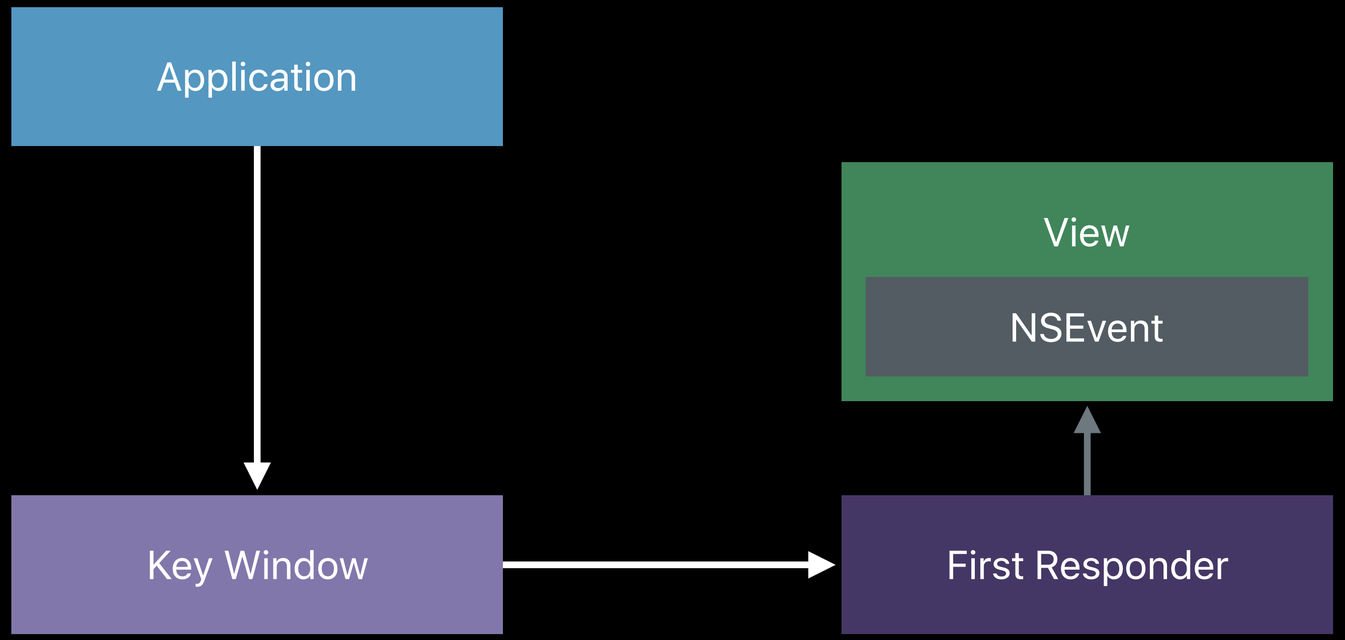
# Responder Chain



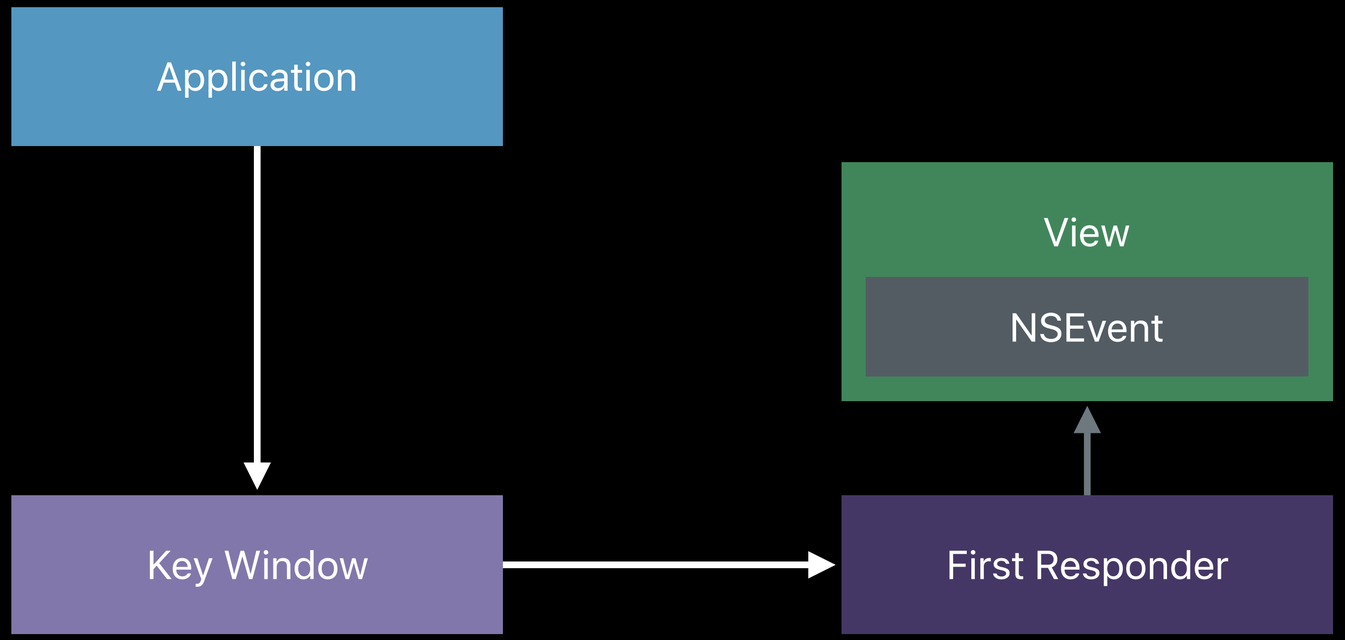
# Responder Chain



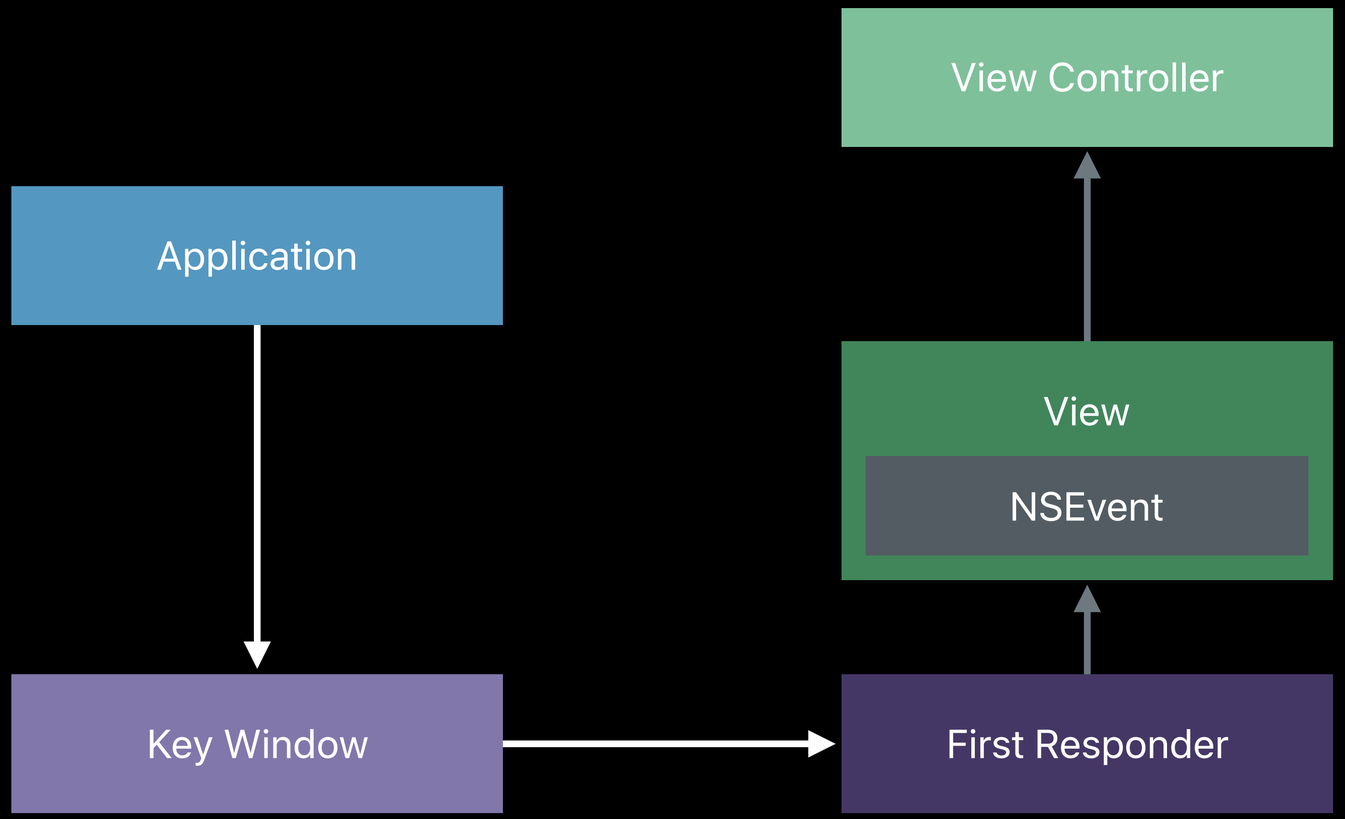
# Responder Chain



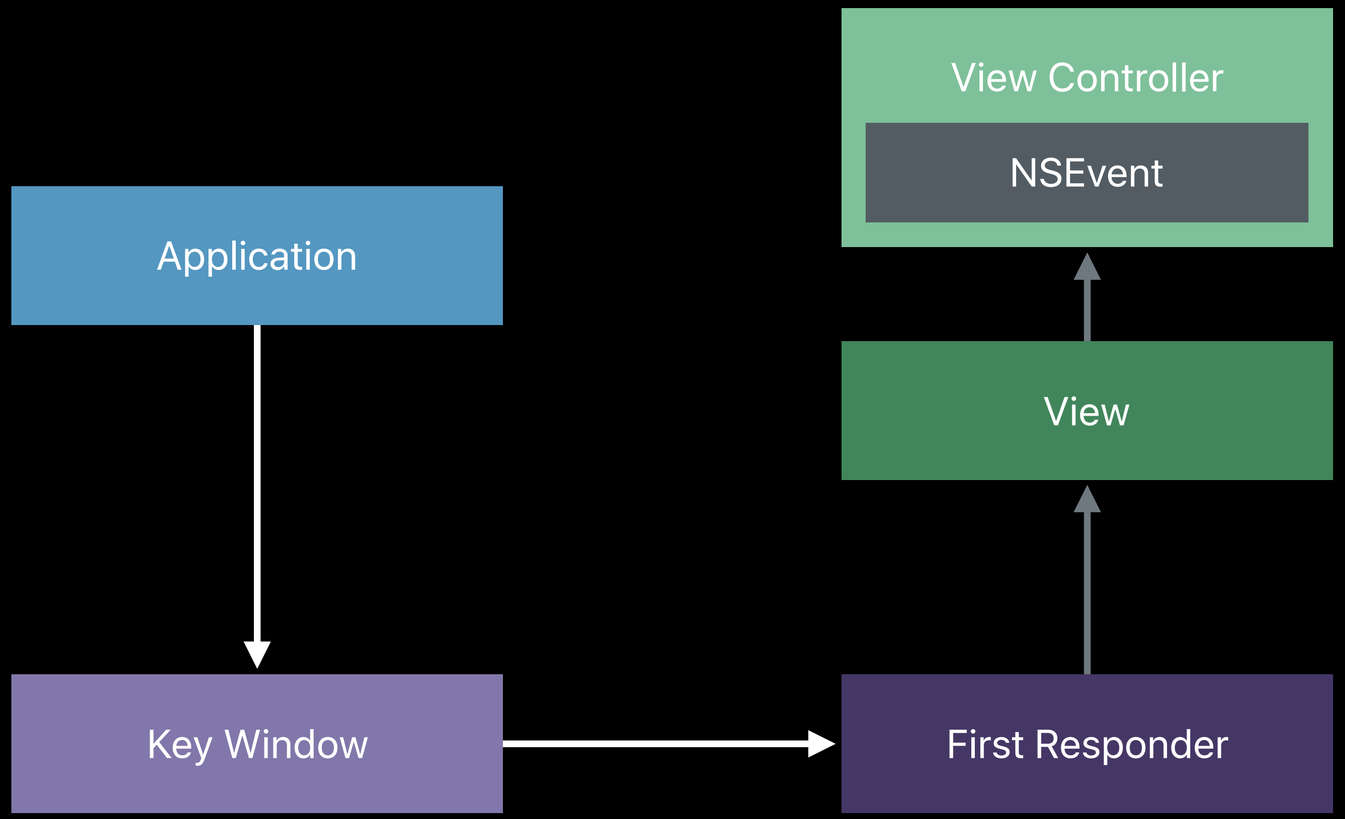
# Responder Chain



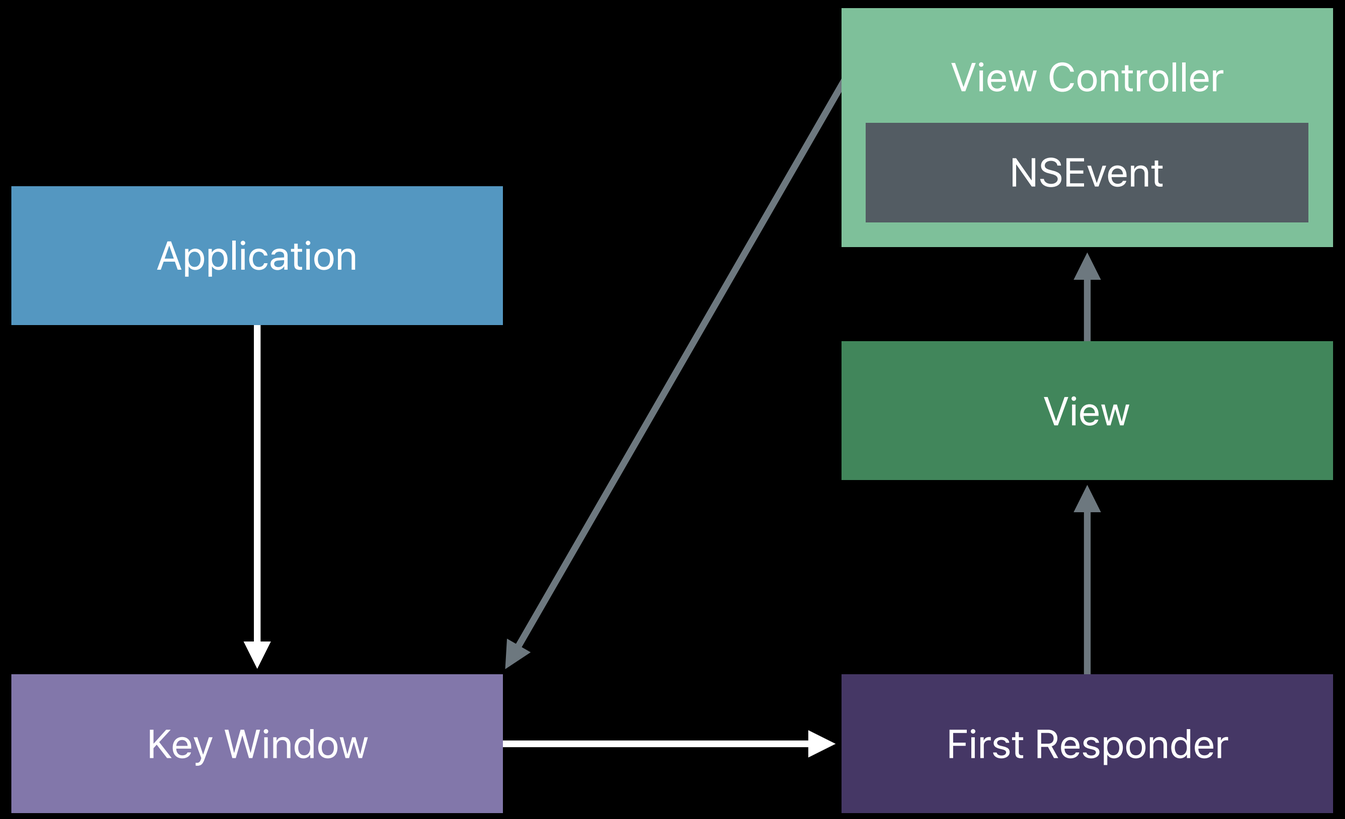
# Responder Chain



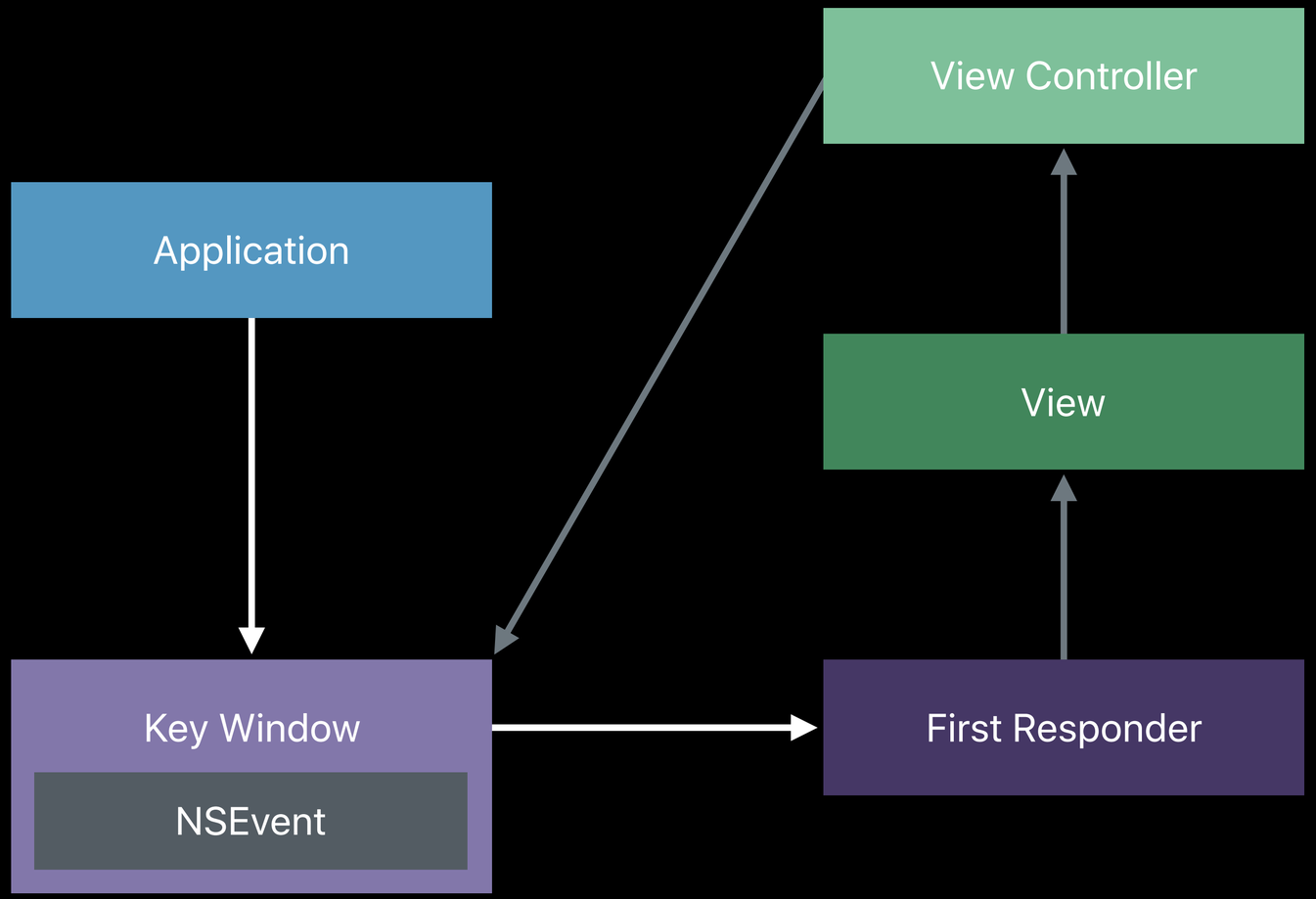
# Responder Chain



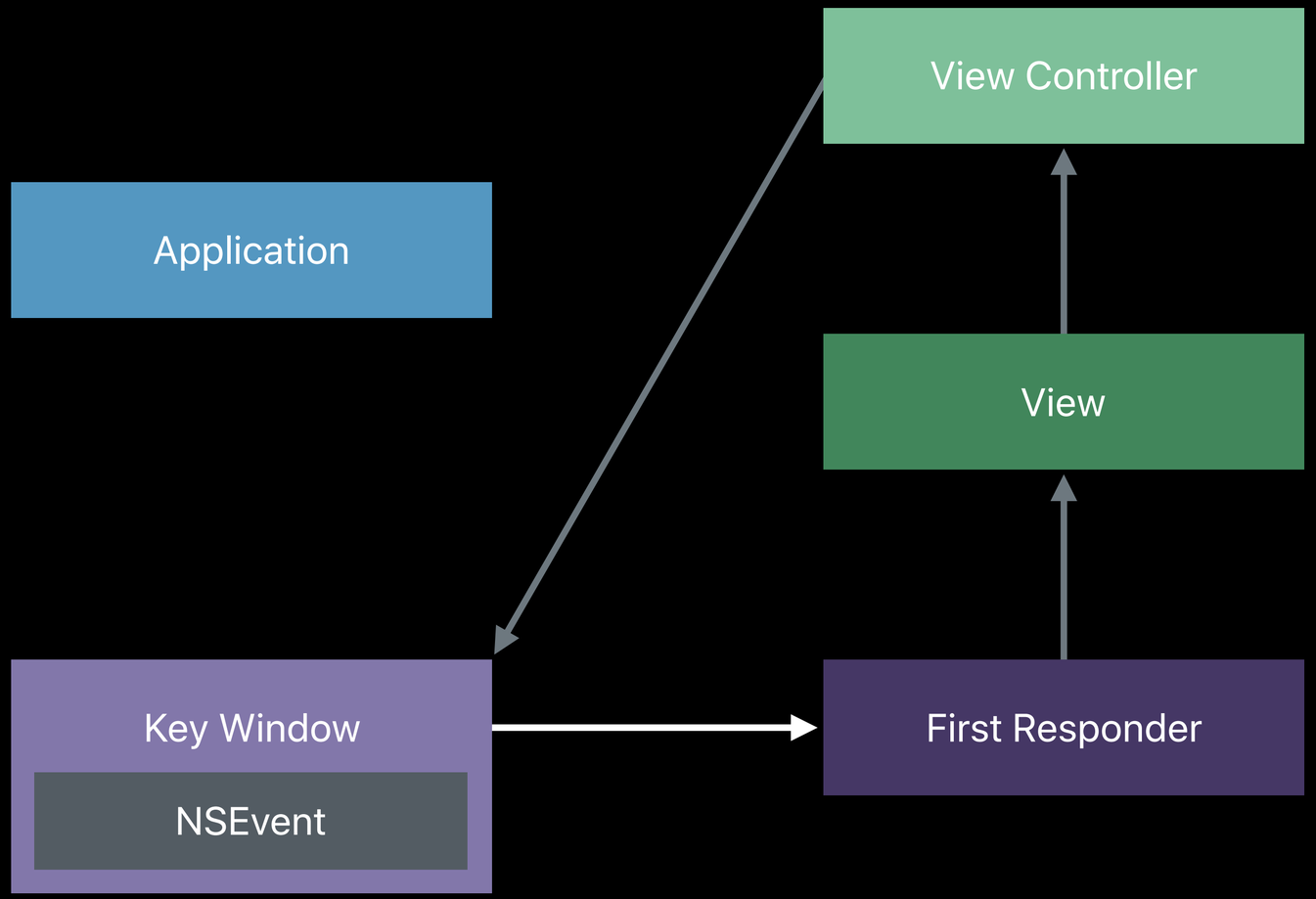
# Responder Chain



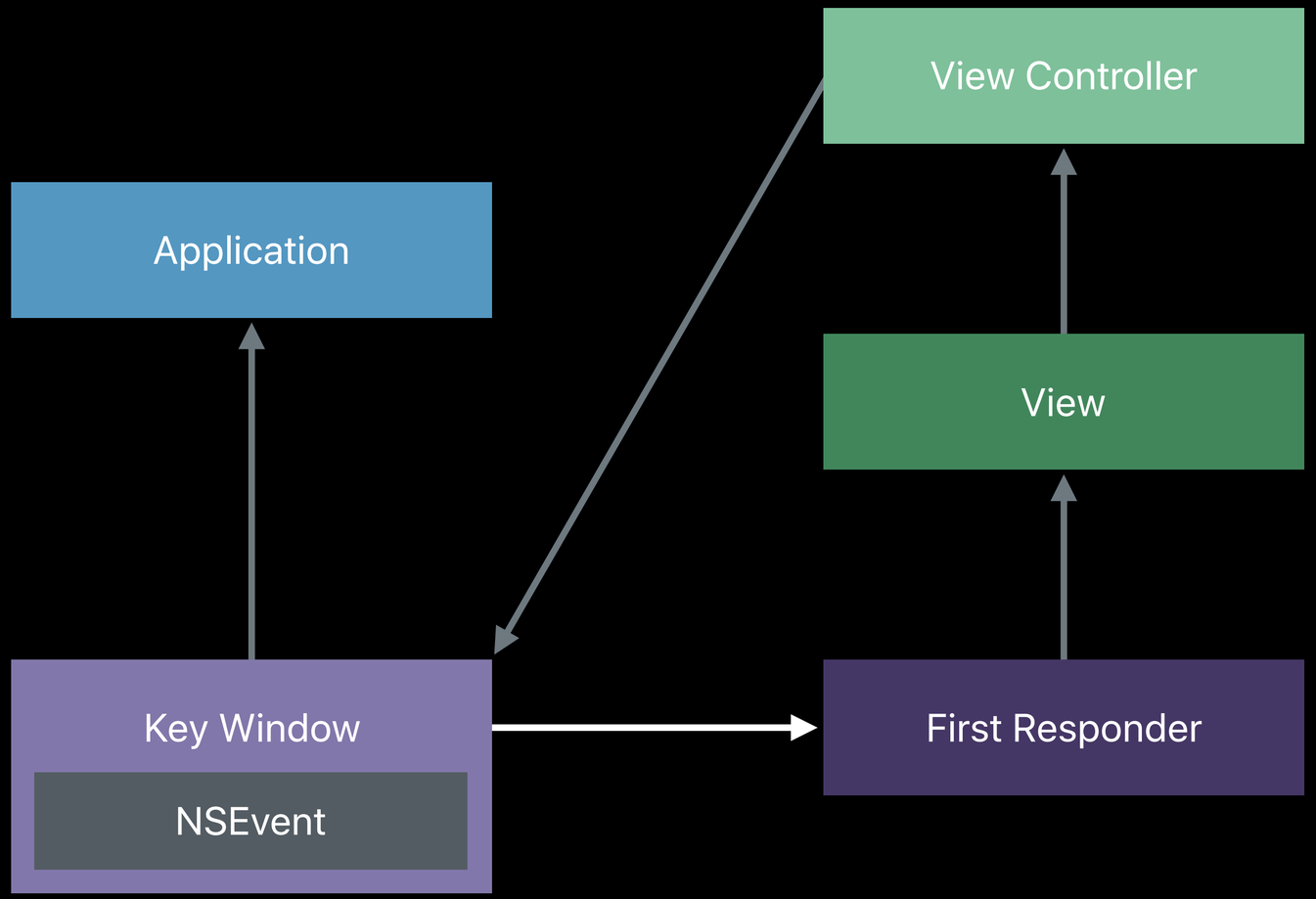
# Responder Chain



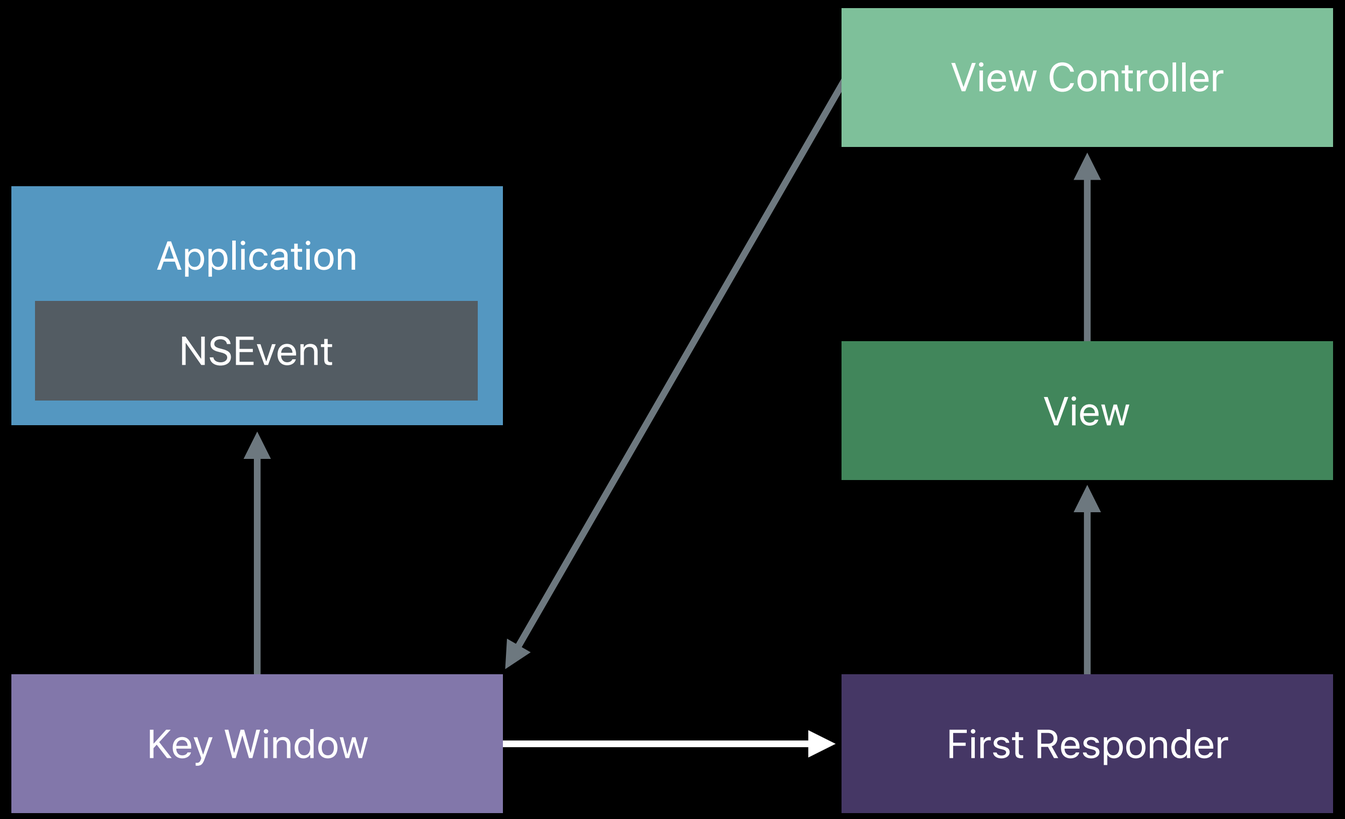
# Responder Chain



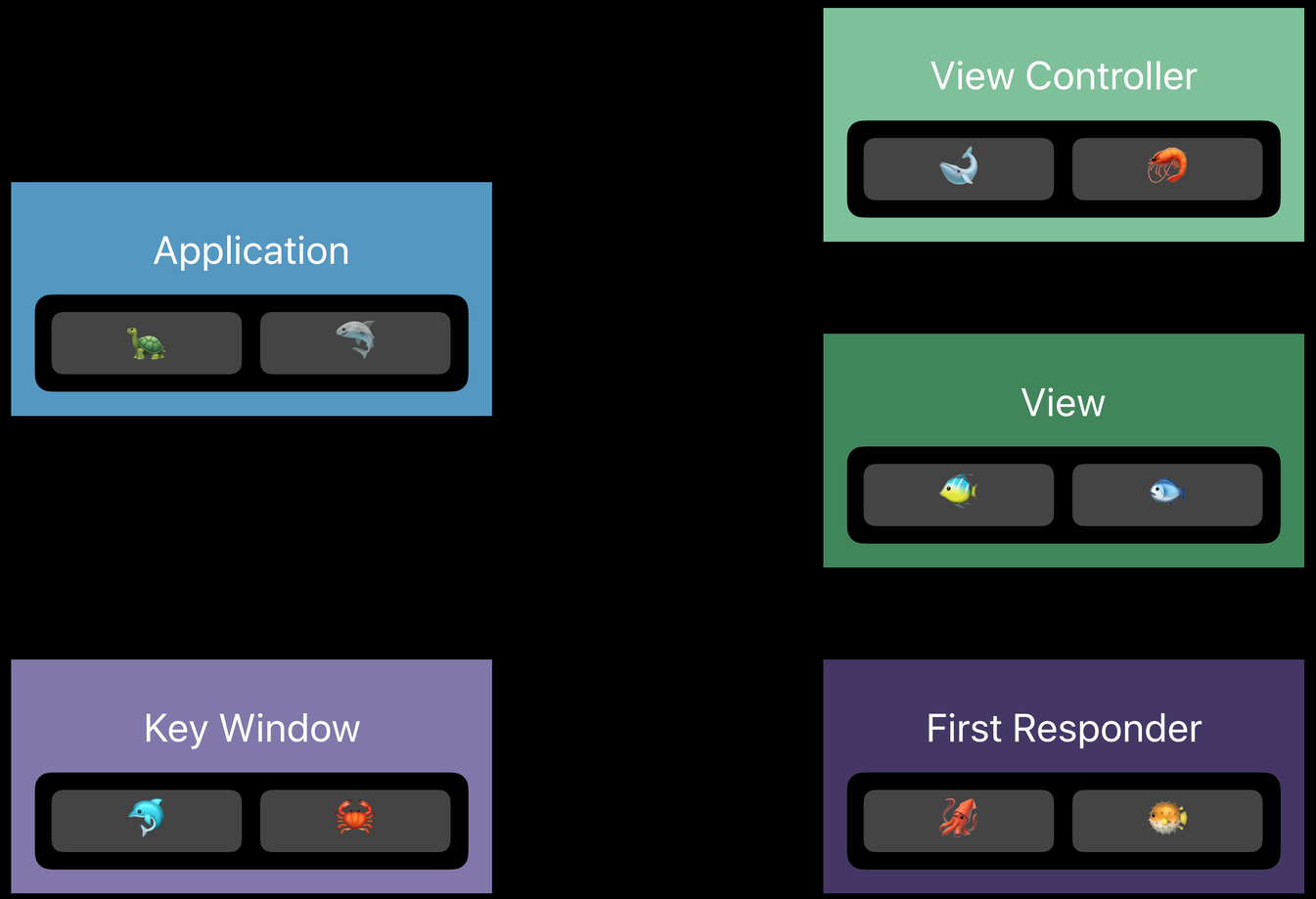
# Responder Chain



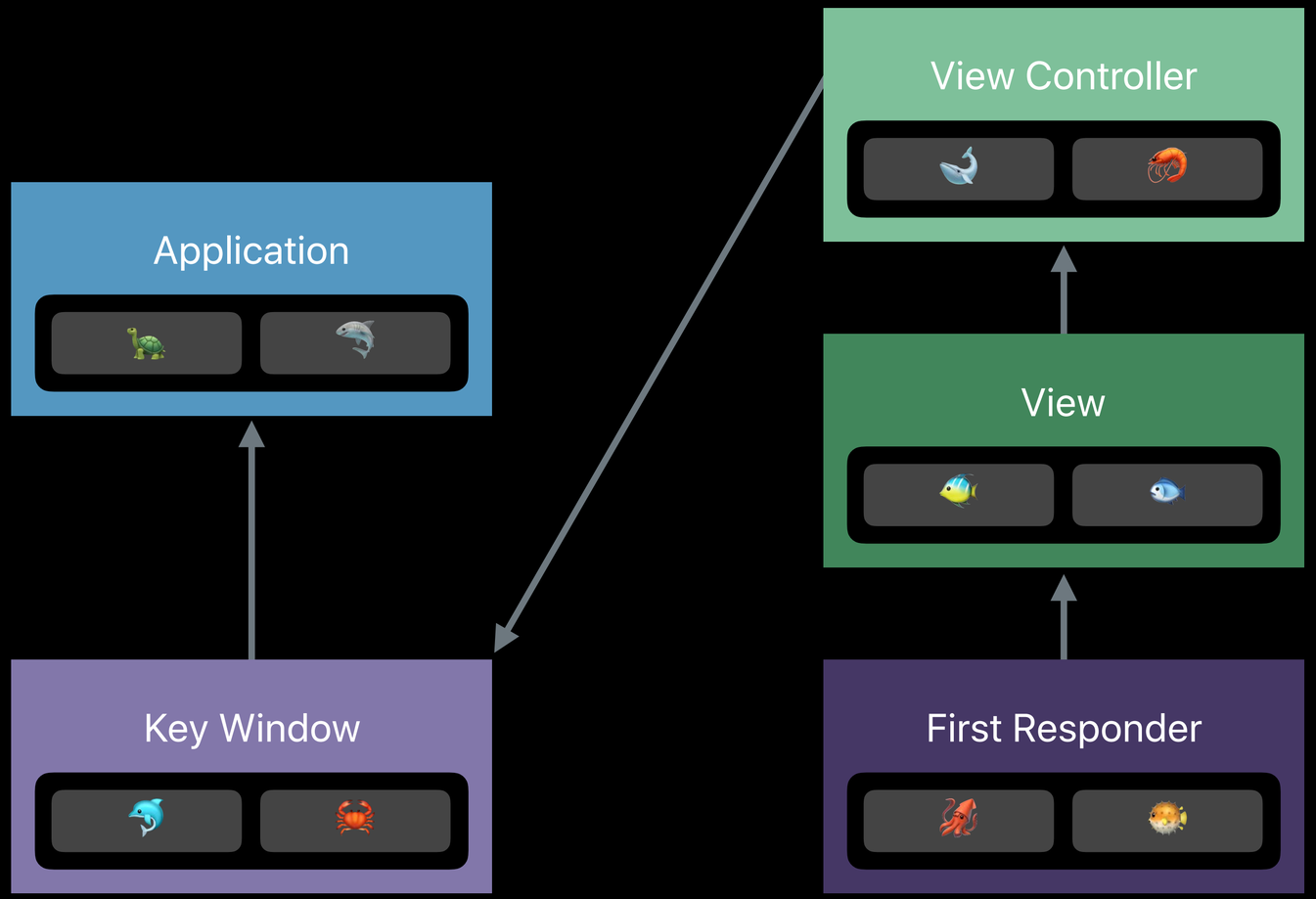
# Responder Chain



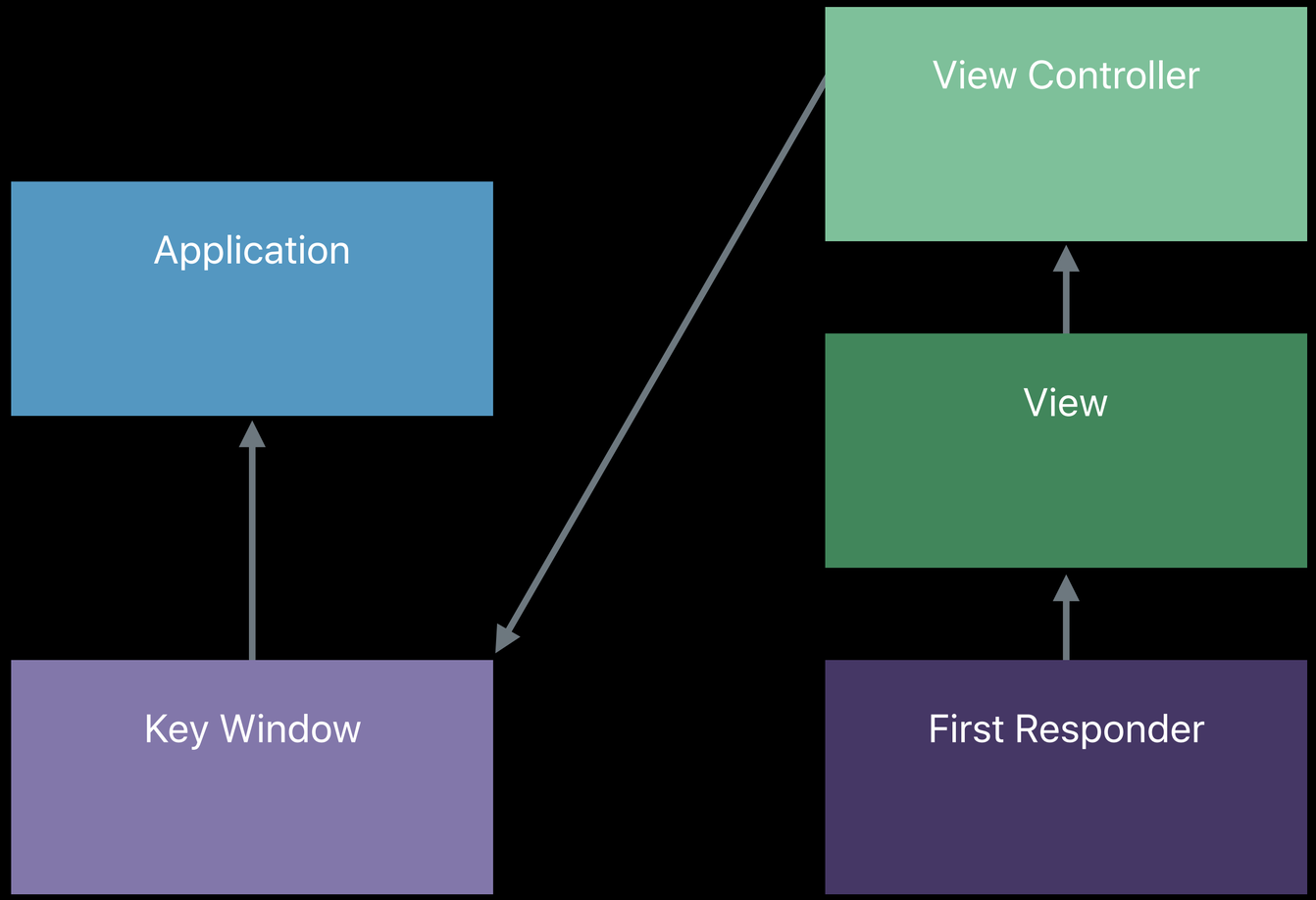
# Responder Chain



# Responder Chain



# Responder Chain



# A Special Note About Views and View Controllers

To be the first responder of a window, views must opt-in

```
class MyViewSubclass: NSView {
    func override acceptsFirstResponder -> Bool {
        return true
    }
}
```

# Responder Chain

Multiple `NSTouchBar` instances

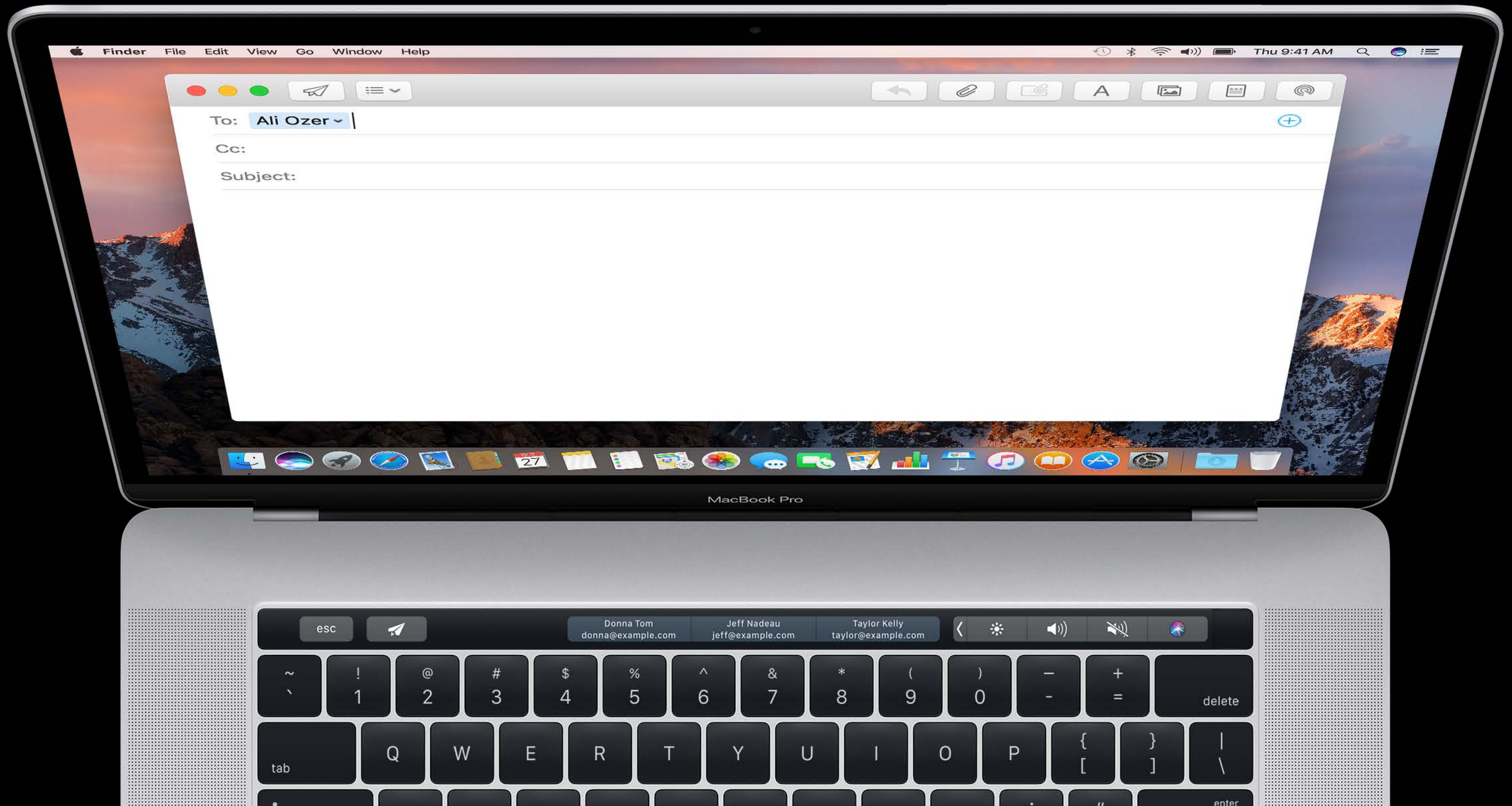
Closest to keyboard input wins

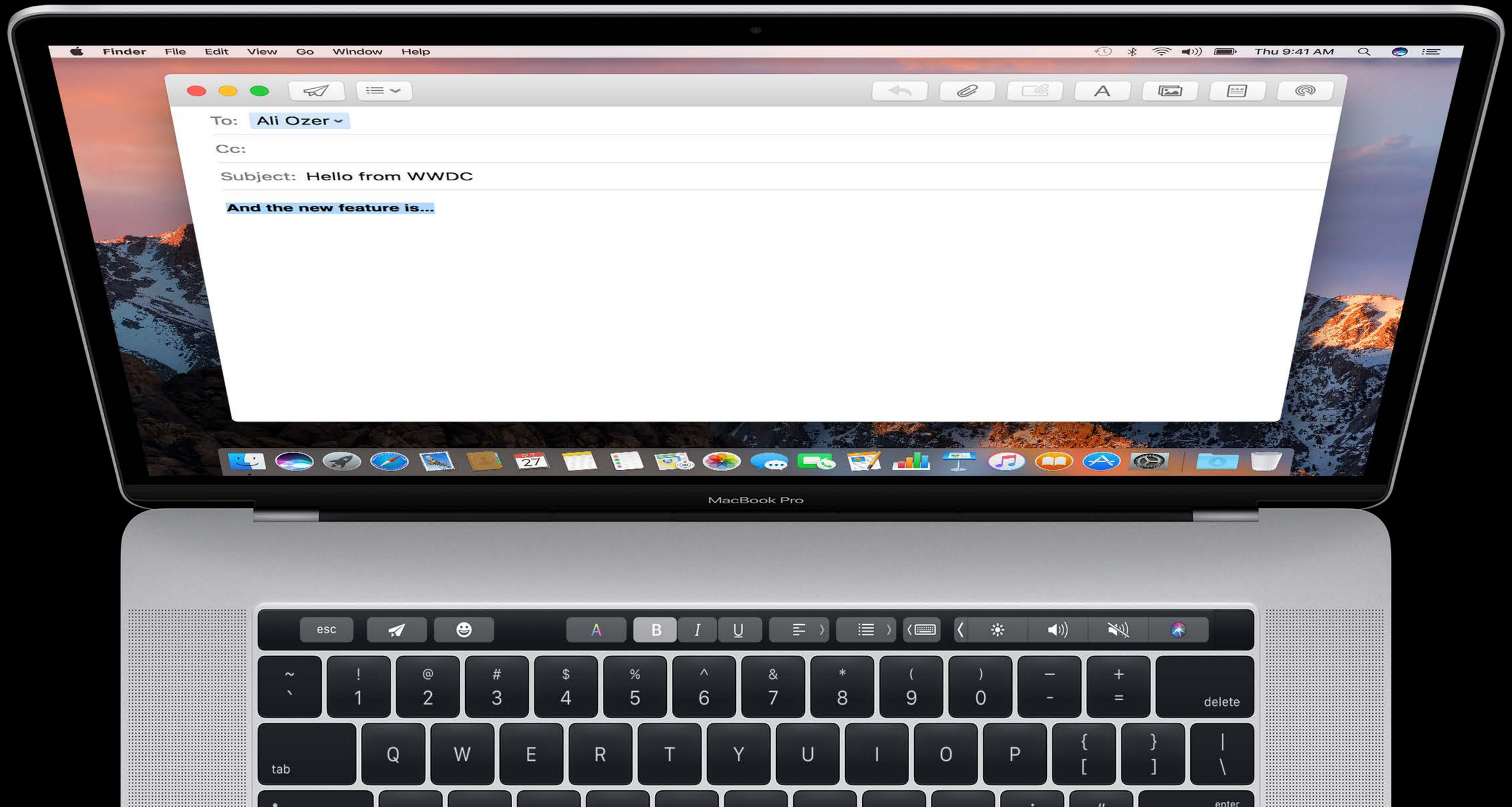
Many system controls have automatic support

# NSTouchBarItemIdentifier.otherItemsProxy

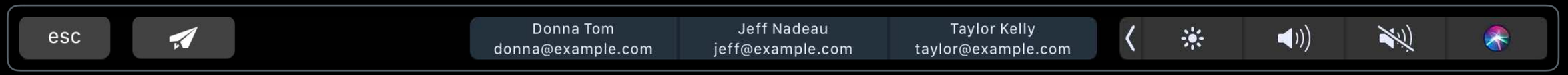
A special item identifier

Incorporates items from other `NSTouchBar` instances


Finder File Edit View Go Window Help
Thu 9:41 AM
To: Ali Ozer
Cc:
Subject:
MacBook Pro
esc
Donna Tom
donna@example.com
Jeff Nadeau
jeff@example.com
Taylor Kelly
taylor@example.com
delete
tab
Q W E R T Y U I O P
enter


Finder File Edit View Go Window Help
Thu 9:41 AM
To: Ali Ozer
Cc:
Subject: Hello from WWDC
And the new feature is...
MacBook Pro
esc
delete
tab

Touch Bar




Window Controller’s NSTouchBar


Recipient Field’s NSTouchBar

Touch Bar


Window Controller’s NSTouchBar


Edit View’s NSTouchBar

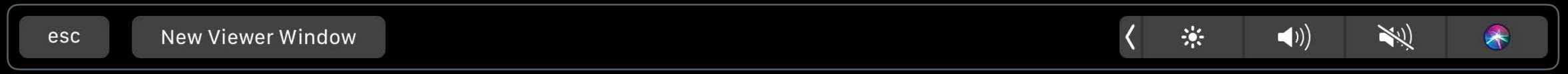

Drag your favorite items to the bottom of the screen into the Touch Bar...
Show typing suggestions
Done
Default Set
Send
Include Attachments
Markup
Character Picker
Text Color
Text Style
Text Alignment
Text List
Space

# NSTouchBarItemIdentifier.otherItemsProxy

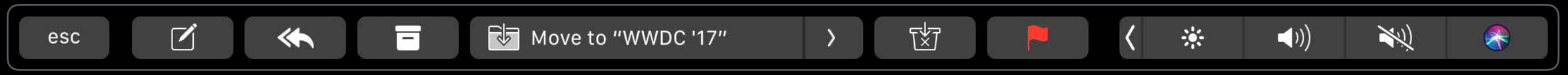
## Excluding an NSTouchBar

### Mail Viewer

No window



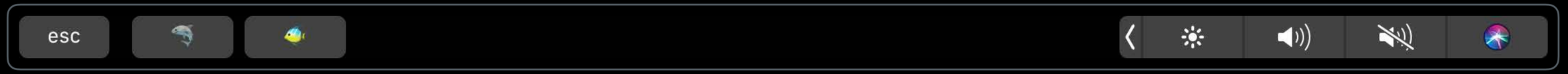

Viewer window



# Space Item Identifiers

# Space Item Identifiers

no space



# Space Item Identifiers

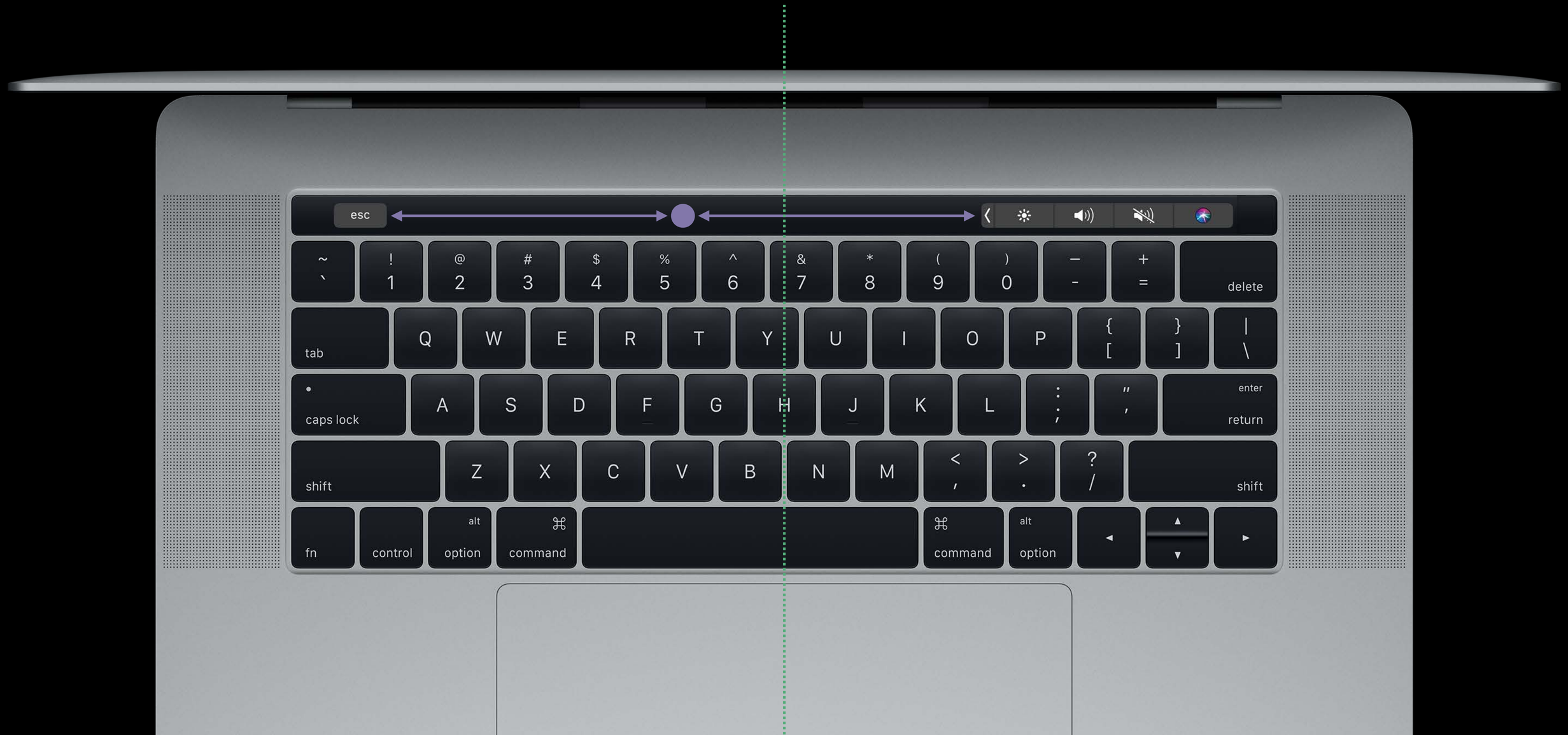
`.fixedSpaceLarge`

# Space Item Identifiers

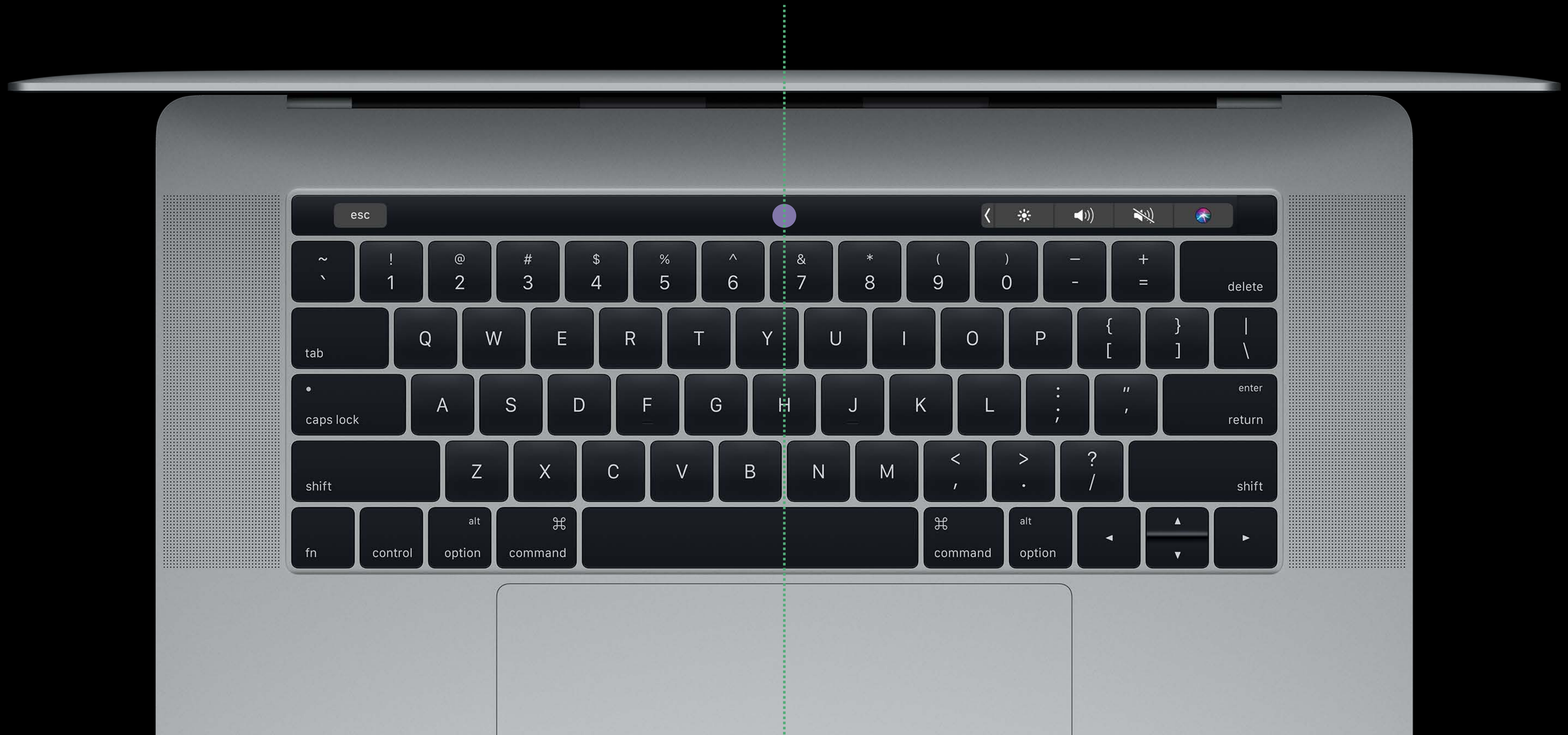
`.flexibleSpace`

# Flexible Space

# Flexible Space



# Principal Item Identifier



# Principal Item Identifier

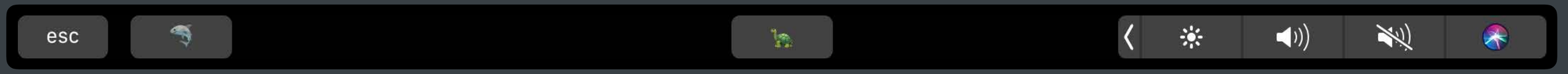
`NSTouchBar.principalItemIdentifier`

Physically centered

```
// Using NSTouchBar.principalItemIdentifier

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle")

let bar = NSTouchBar()

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)
let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🐢", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
bar.principalItemIdentifier = sharkIdentifier
```

```
// Using NSTouchBar.principalItemIdentifier

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle")

let bar = NSTouchBar()

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)
let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🐢", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
bar.principalItemIdentifier = sharkIdentifier
```

```
// Using NSTouchBar.principalItemIdentifier

let sharkIdentifier = NSTouchBarItem.Identifier("shark")
let seaTurtleIdentifier = NSTouchBarItem.Identifier("sea turtle")

let bar = NSTouchBar()

bar.defaultItemIdentifiers = [sharkIdentifier, seaTurtleIdentifier]

let sharkItem = NSCustomTouchBarItem(identifier: sharkIdentifier)
sharkItem.view = NSButton(title: "🦈", target: nil, action: nil)
let turtleItem = NSCustomTouchBarItem(identifier: seaTurtleIdentifier)
turtleItem.view = NSButton(title: "🐢", target: nil, action: nil)

bar.templateItems = [sharkItem, turtleItem]
bar.principalItemIdentifier = seaTurtleIdentifier
```



Core classes

Responder chain

NSTouchBarItem in Depth

# NSTouchBarItem in Depth

John Tegtmeyer

# NSTouchBarItem

```
class NSTouchBarItem : NSObject, NSCoding {

    var identifier: NSTouchBarItem.Identifier { get }

    var view: NSView? { get }

    var viewController: NSViewController? { get }

    var customizationLabel: String { get }
}
```

# NSTouchBarItem

```
class NSTouchBarItem : NSObject, NSCoding {

    var identifier: NSTouchBarItem.Identifier { get }

    var view: NSView? { get }

    var viewController: NSViewController? { get }

    var customizationLabel: String { get }
}
```

# NSTouchBarItem

```
class NSTouchBarItem : NSObject, NSCoding {

    var identifier: NSTouchBarItem.Identifier { get }

    var view: NSView? { get }

    var viewController: NSViewController? { get }

    var customizationLabel: String { get }
}
```

# NSTouchBarItem

```
class NSTouchBarItem : NSObject, NSCoding {

    var identifier: NSTouchBarItem.Identifier { get }

    var view: NSView? { get }

    var viewController: NSViewController? { get }

    var customizationLabel: String { get }
}
```

# NSTouchBarItem

```
class NSTouchBarItem : NSObject, NSCoding {

    var identifier: NSTouchBarItem.Identifier { get }

    var view: NSView? { get }

    var viewController: NSViewController? { get }

    var customizationLabel: String { get }
}
```

# NSTouchBarItem

Constraint-based sizing

- `intrinsicContentSize`
- Inequality constraints

NSCustomTouchBarItem

NSPopoverTouchBarItem

NSSliderTouchBarItem

NSGroupTouchBarItem

Other Items

NSCustomTouchBarItem

NSPopoverTouchBarItem

NSSliderTouchBarItem

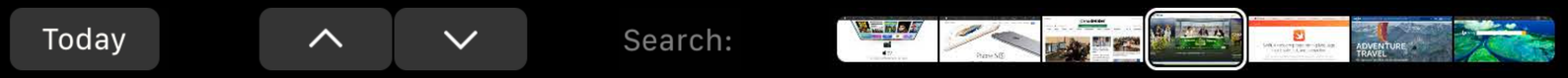
NSGroupTouchBarItem

Other Items

# NSCustomTouchBarItem

Set view or view controller

Convenience constructors




Button | Segmented Control | Label | Scrubber

```
// customTouchBarItem : NSButton

let buttonItem = NSCustomTouchBarItem(identifier: identifier)
buttonItem.view = NSButton(title: "Hello World!", target: nil, action: nil)
```

```
// customTouchBarItem : NSSegmentedControl

let segmentedItem = NSCustomTouchBarItem(identifier: identifier)
segmentedItem.view = NSSegmentedControl(labels: ["foo", "bar", "buzz"],
                        trackingMode:.selectOne, target: nil, action: nil)
```

NSCustomTouchBarItem

NSPopoverTouchBarItem

NSSliderTouchBarItem

NSGroupTouchBarItem

Other Items

# NSPopoverTouchBarItem

Collapsed representation


Popover NSTouchBar

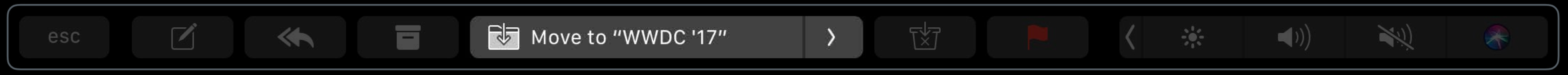
# NSPopoverTouchBarItem

Default


```
class NSPopoverTouchBarItem: NSTouchBarItem {
    var collapsedRepresentationImage: NSImage?

    var collapsedRepresentationLabel: String
}
```

# NSPopoverTouchBarItem

Custom




```
class NSPopoverTouchBarItem: NSTouchBarItem {
    var collapsedRepresentation: NSView

    func showPopover(_ sender: Any?)

    func dismissPopover(_ sender: Any?)
}
```

# NSPopoverTouchBarItem

Touch and show

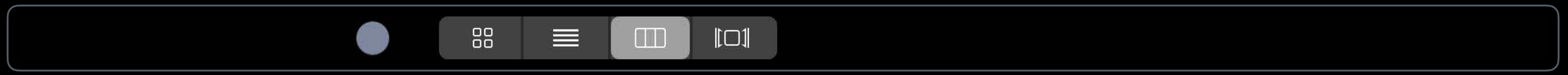
# NSPopoverTouchBarItem

Press and hold

# NSPopoverTouchBarItem

Touch and show


Press and hold



# NSPopoverTouchBarItem

```
class NSPopoverTouchBarItem: NSTouchBarItem {
    var popoverTouchBar: NSTouchBar

    var pressAndHoldTouchBar: NSTouchBar?
}
```

NSCustomTouchBarItem

NSPopoverTouchBarItem

NSSliderTouchBarItem

NSGroupTouchBarItem

Other Items

# NSSliderTouchBarItem


Slider

Value accessories

Label

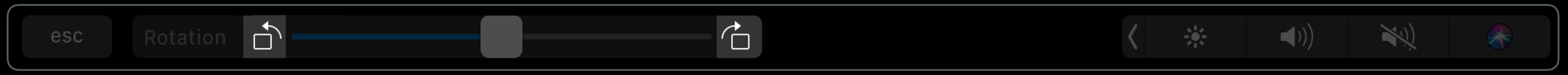
# NSSliderTouchBarItem


Slider

Value accessories

Label

# NSSliderTouchBarItem




Slider

Value accessories

Label

# NSSliderTouchBarItem


Slider

Value accessories

Label

# NSSliderTouchBarItem

Press and hold

# NSSliderTouchBarItem

Press and hold

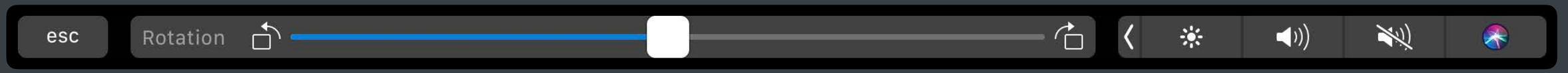
```
// NSSliderTouchBarItem with value accessories and a label

let sliderItem = NSSliderTouchBarItem(identifier: identifier)
sliderItem.minimumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateLeftTemplate)!)
sliderItem.maximumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateRightTemplate)!)
sliderItem.label = "Rotation"
```

```
// NSSliderTouchBarItem with value accessories and a label

let sliderItem = NSSliderTouchBarItem(identifier: identifier)
sliderItem.minimumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateLeftTemplate)!)
sliderItem.maximumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateRightTemplate)!)
sliderItem.label = "Rotation"
```

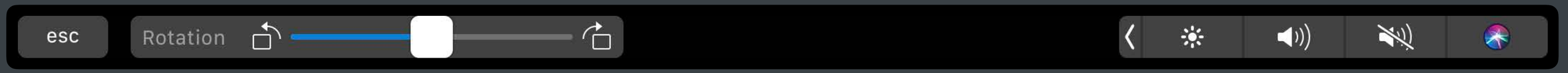
```
// NSSliderTouchBarItem with value accessories and a label

let sliderItem = NSSliderTouchBarItem(identifier: identifier)
sliderItem.minimumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateLeftTemplate)!)
sliderItem.maximumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateRightTemplate)!)
sliderItem.label = "Rotation"
```

```
// NSSliderTouchBarItem with value accessories and a label

let sliderItem = NSSliderTouchBarItem(identifier: identifier)
sliderItem.minimumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateLeftTemplate)!)
sliderItem.maximumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateRightTemplate)!)
sliderItem.label = "Rotation"
```

```
// NSSliderTouchBarItem with value accessories and a label

let sliderItem = NSSliderTouchBarItem(identifier: identifier)
sliderItem.minimumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateLeftTemplate)!)
sliderItem.maximumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateRightTemplate)!)
sliderItem.label = "Rotation"
```



```
// NSSliderTouchBarItem with value accessories and a label

let sliderItem = NSSliderTouchBarItem(identifier: identifier)
sliderItem.minimumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateLeftTemplate)!)
sliderItem.maximumValueAccessory = NSSliderAccessory(image:
                                    NSImage(named: .touchBarRotateRightTemplate)!)
sliderItem.label = "Rotation"

sliderItem.slider.widthAnchor.constraint(equalToConstant: 200).isActive = true
```



NSCustomTouchBarItem

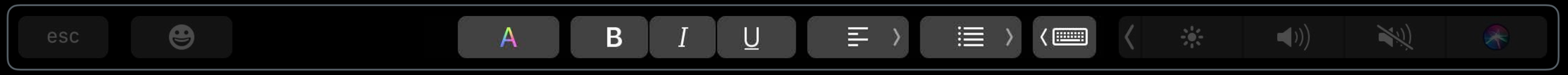
NSPopoverTouchBarItem

NSSliderTouchBarItem

NSGroupTouchBarItem

Other Items

# NSGroupTouchBarItem




```
class NSGroupTouchBarItem: NSTouchBarItem {
    convenience init(identifier: NSTouchBarItem.Identifier, items: [NSTouchBarItem])

    var groupTouchBar: NSTouchBar
}
```

# NSGroupTouchBarItem

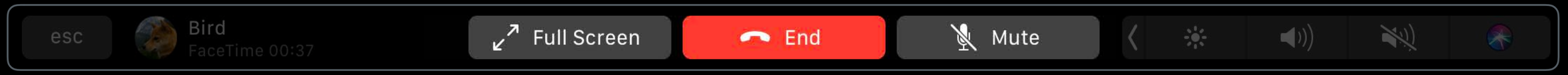
Center multiple items

Customization

Localization

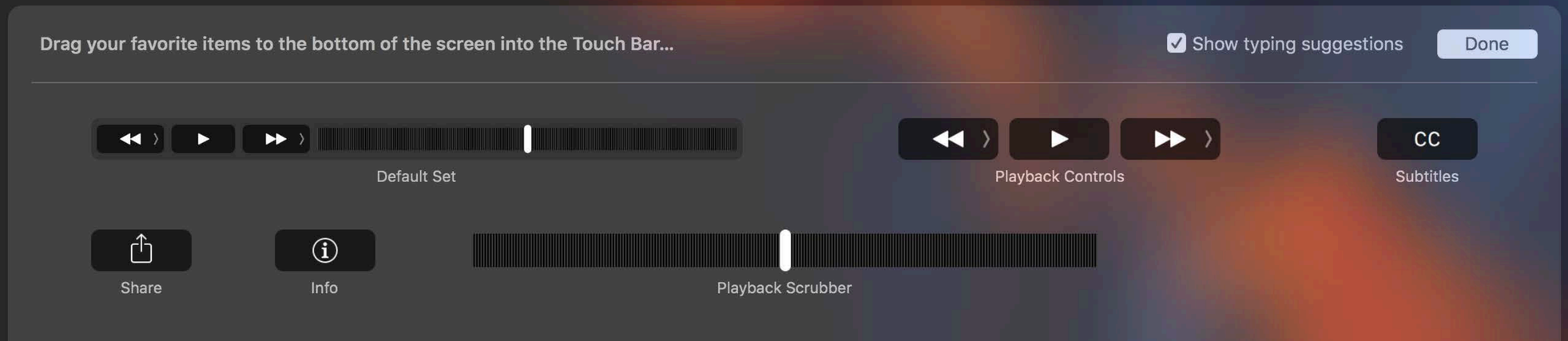
# NSGroupTouchBarItem

Center multiple items



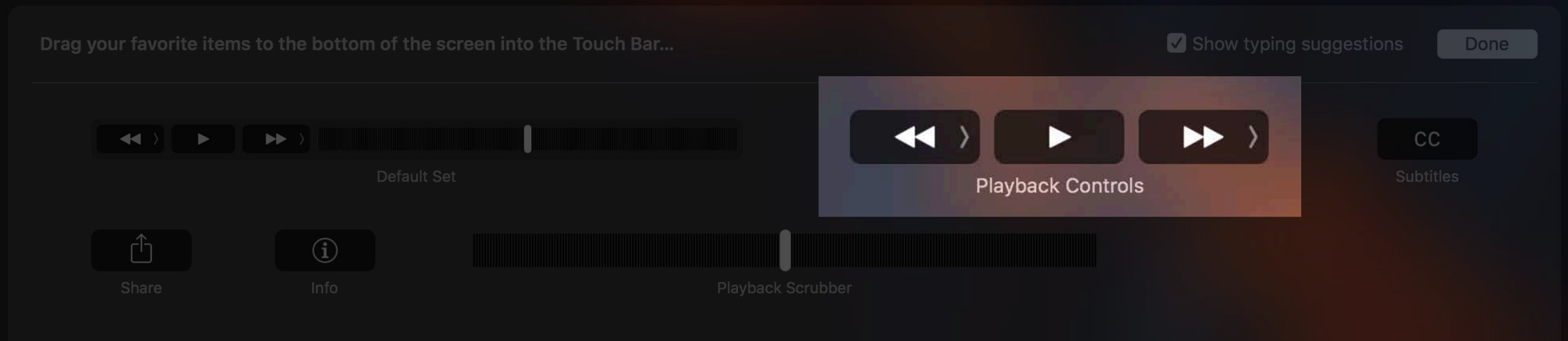
# NSGroupTouchBarItem

Non-customizable



# NSGroupTouchBarItem

Non-customizable



# NSGroupTouchBarItem

Customizable


Within-group reordering

Items remain within group

# NSGroupTouchBarItem

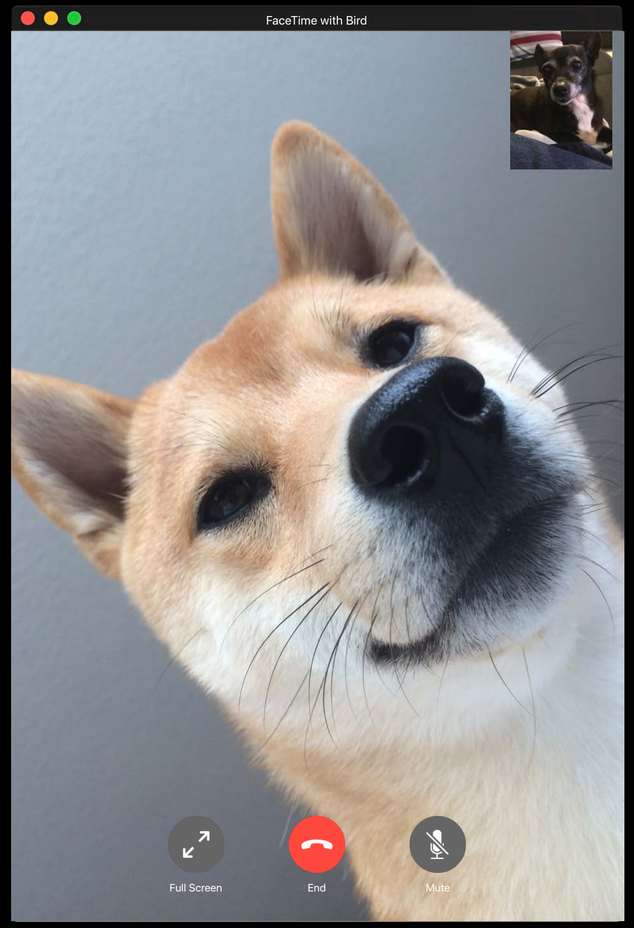
Customizable


Within-group reordering

Items remain within group

# NSGroupTouchBarItem

## Localization




# NSGroupTouchBarItem

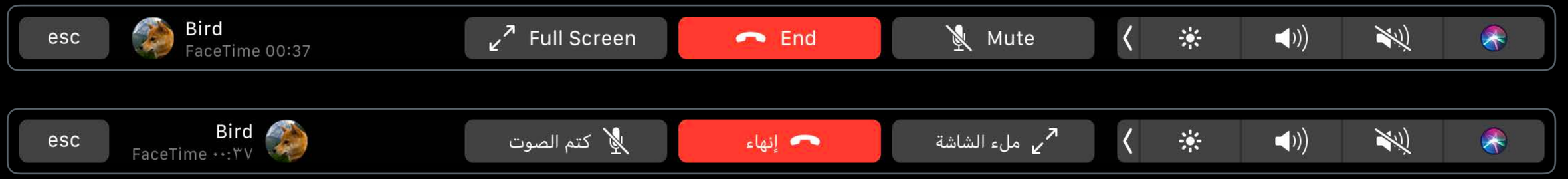
## Localization

# NSGroupTouchBarItem

## Localization

# NSGroupTouchBarItem

## Localization



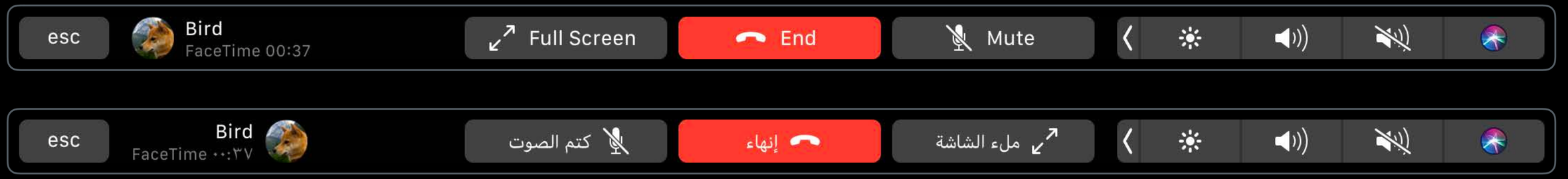

Control Strip and esc key do not flip left-to-right

Opt-in per group

# NSGroupTouchBarItem

## Localization

NEW



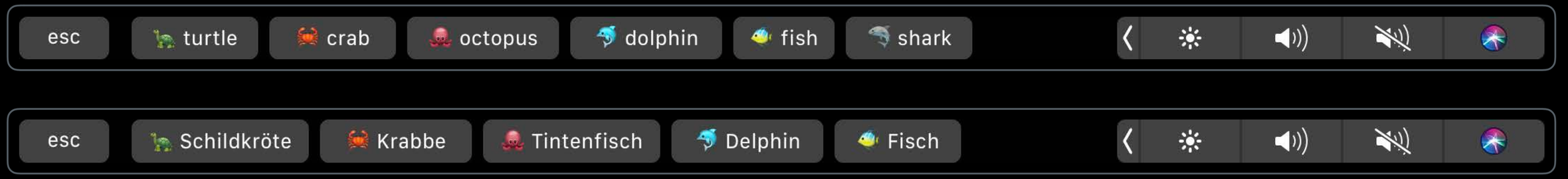

Control Strip and esc key do not flip left-to-right

Opt-in per group

```
groupItem.groupUserInterfaceLayoutDirection = NSApp.userInterfaceLayoutDirection
```

# NSGroupTouchBarItem

## Localization

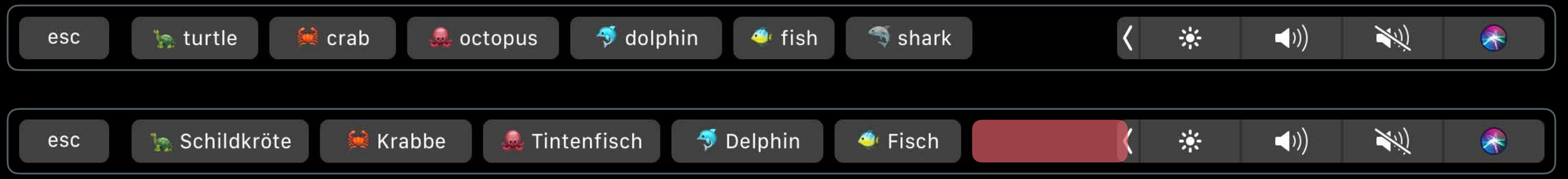



Design for variable length strings

Rigid constraints can cause clipping

# NSGroupTouchBarItem

## Localization

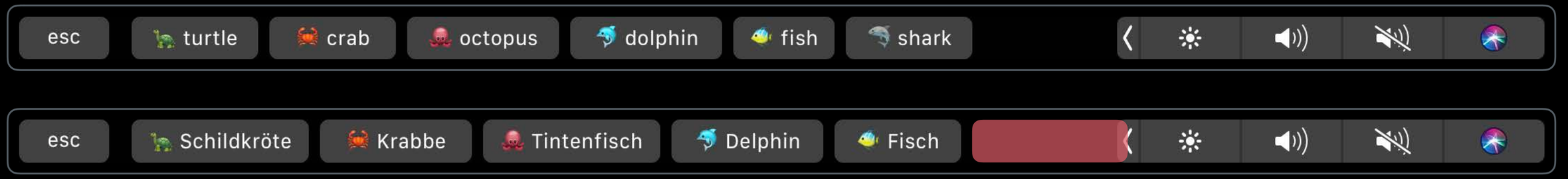



Design for variable length strings

Rigid constraints can cause clipping

# NSGroupTouchBarItem

## Localization




Design for variable length strings

Rigid constraints can cause clipping

NSCustomTouchBarItem

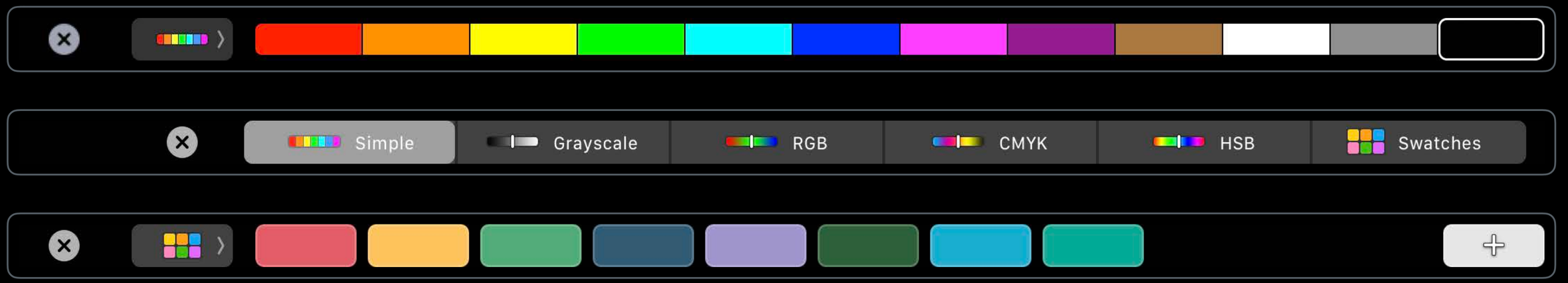
NSPopoverTouchBarItem

NSSliderTouchBarItem

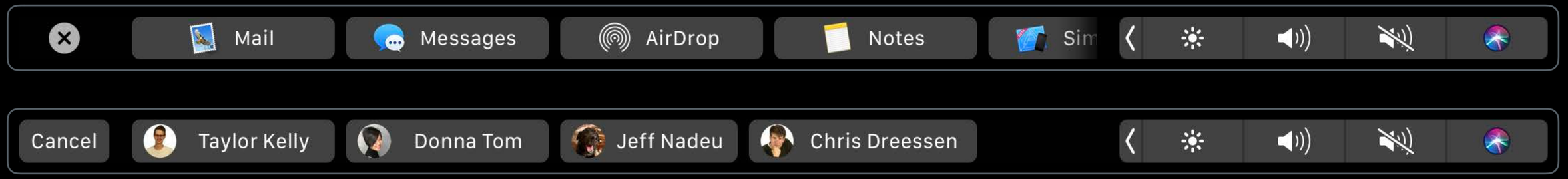
NSGroupTouchBarItem

Other Items

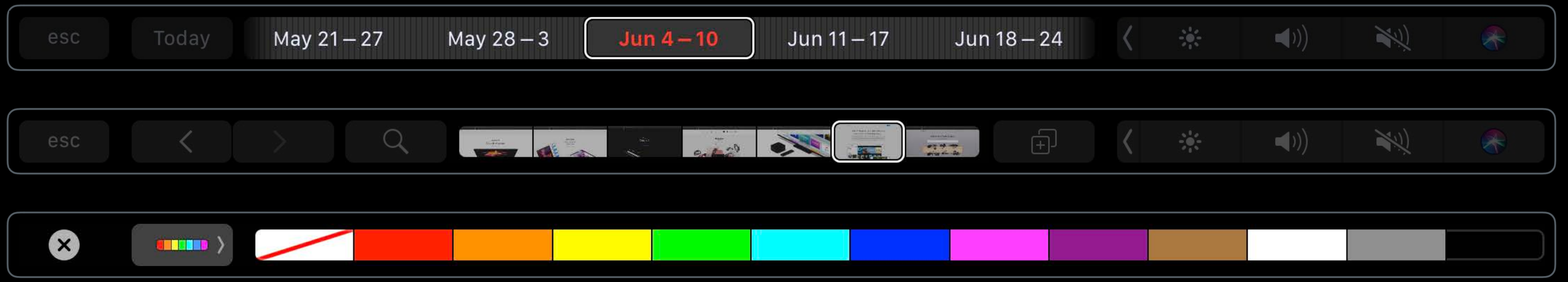
# NSColorPickerTouchBarItem



# NSSharingServicePickerTouchBarItem



# NSScrubber



# Next Steps

Adopt NSTouchBar

User customization

Localization

# More Information

https://developer.apple.com/wwdc17/211

# Related Sessions

| | | |
|---|---|---|
| Choosing the Right Cocoa Container View | Grand Ballroom B | Wednesday 3:10PM |
| Advanced Touch Bar | Grand Ballroom A | Wednesday 5:10PM |

# Labs

| | | |
|---|---|---|
| Cocoa Lab | Technology Lab C | Wed 11:00AM-1:00PM |
| Cocoa Touch Bar Lab | Technology Lab C | Thurs 9:00AM-11:00AM |
| Cocoa Lab | Technology Lab B | Fri 1:50PM-3:20PM |


WWDC17