Advanced Touch Bar

Session 222

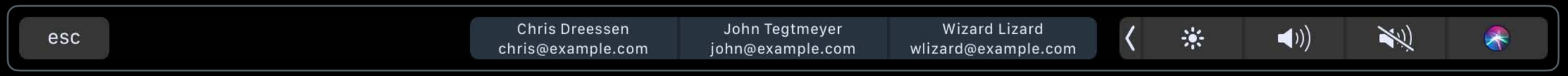
Donna Tom, TextKit
Jeff Nadeau, AppKit
Taylor Kelly, AppKit

Touch Bar Fundamentals

Grand Ballroom A

Wednesday, 10:00AM

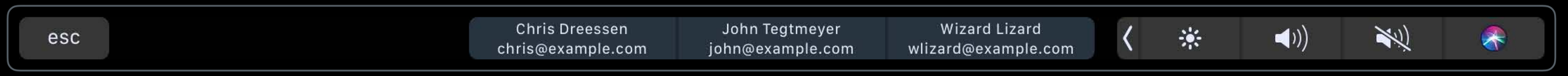
Customizing Text Bars



NSScrubber

Building Custom Controls

Customizing Text Bars



NSScrubber

Building Custom Controls

Standard items

Disabling standard items

Adding custom items

Candidate list item

Standard items

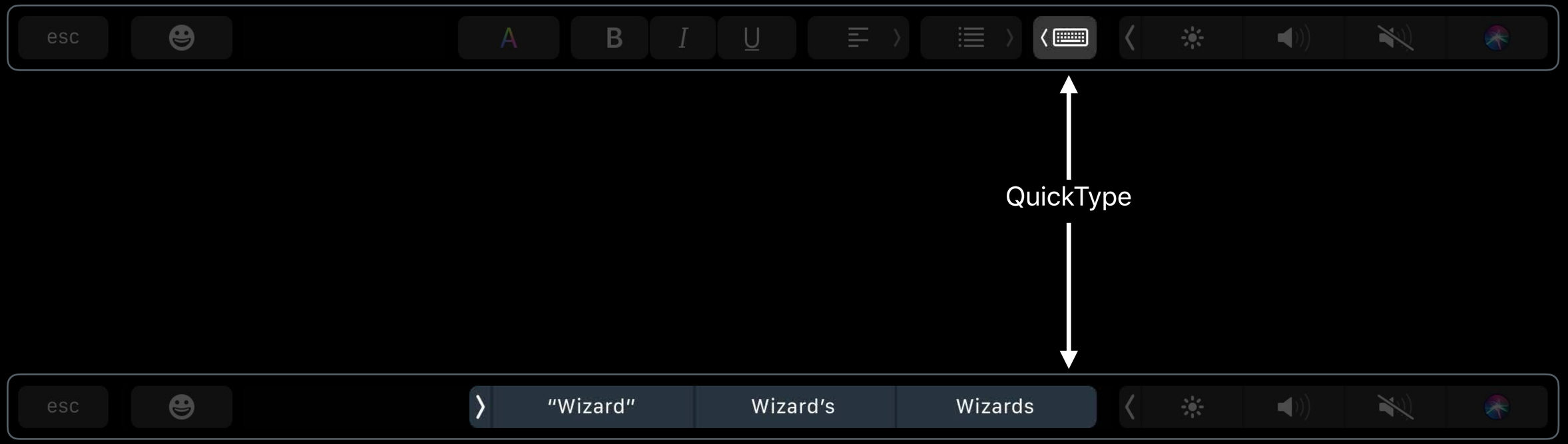
Disabling standard items

Adding custom items

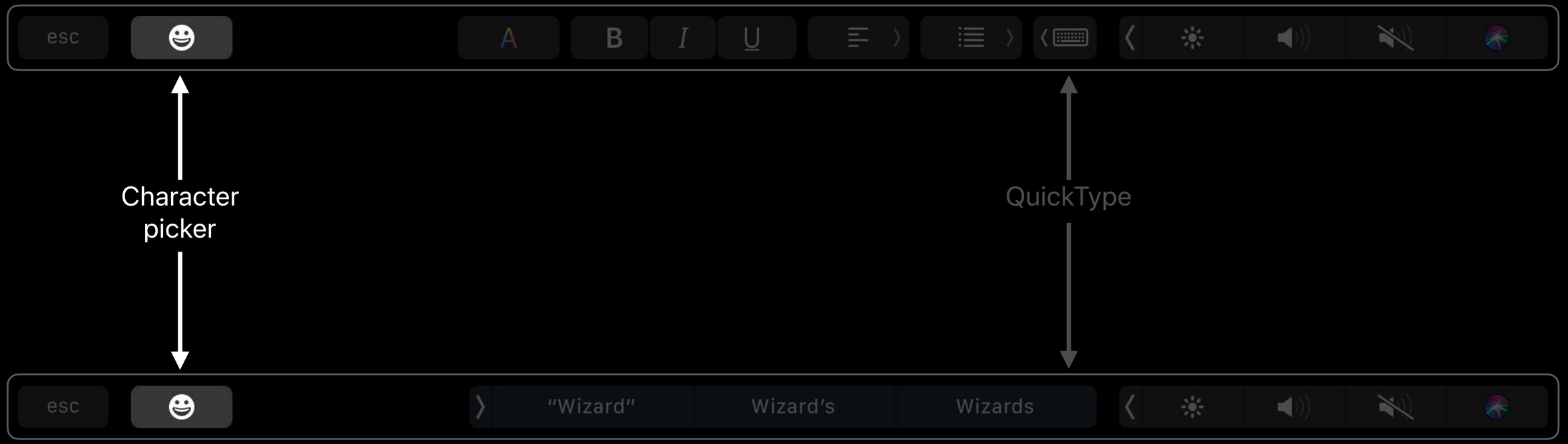
Candidate list item

Standard Items

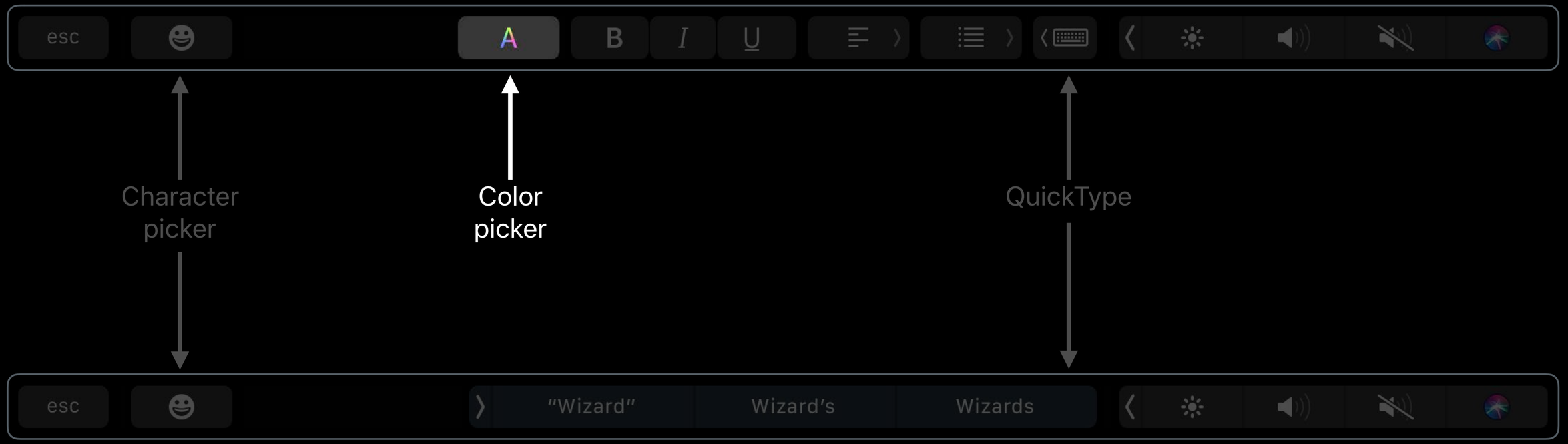
Standard Items



Standard Items

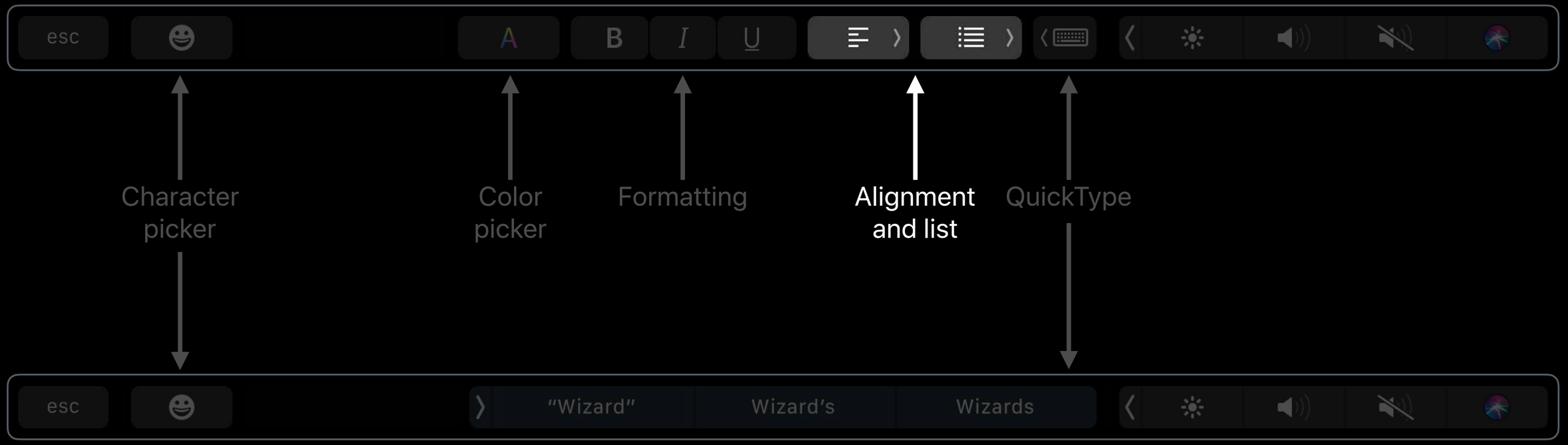


Standard Items



Standard Items

Standard Items

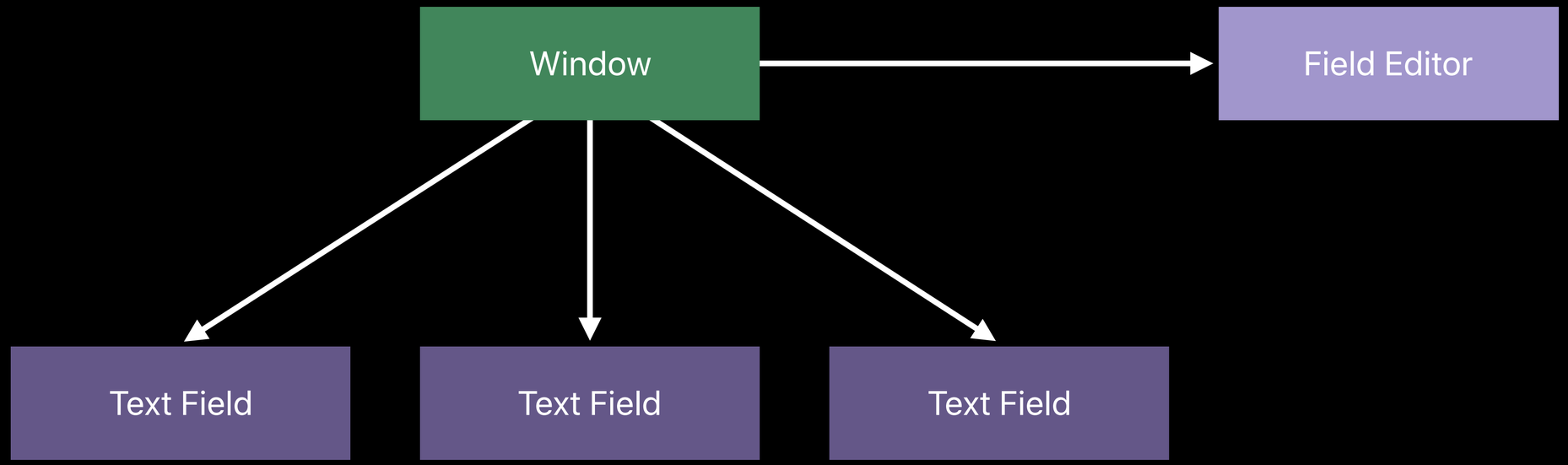



NSTextView
delegate
NSTouchBar


NSTextField
field editor
NSTextView
delegate
NSTouchBar

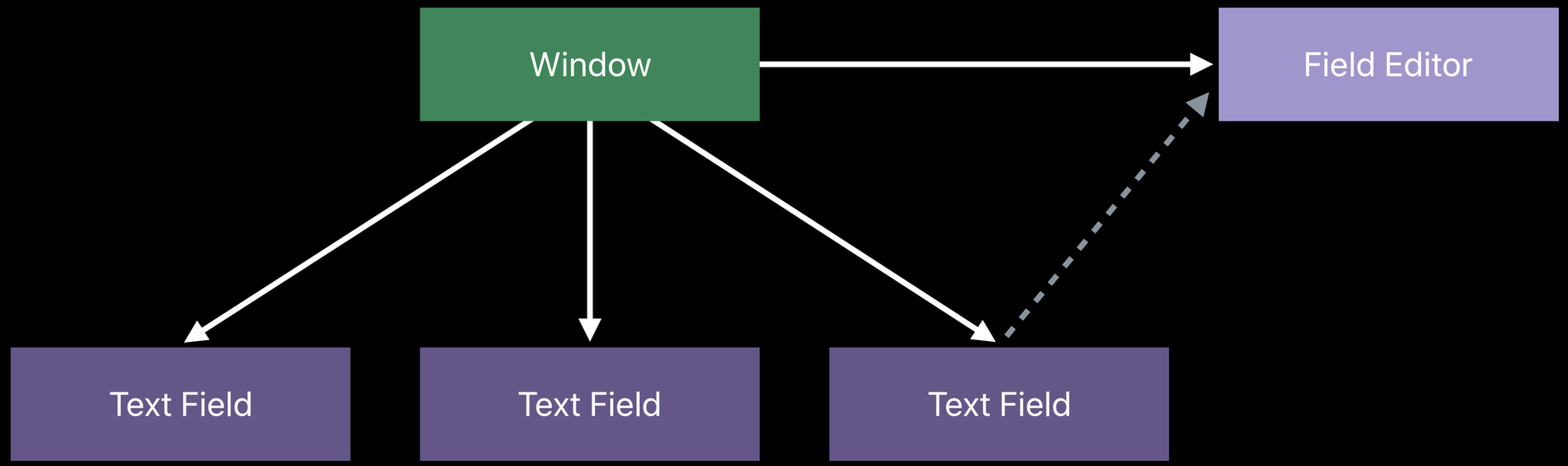
Standard Items

Field editor considerations



Standard Items

Field editor considerations



Standard items

Disabling standard items

Adding custom items

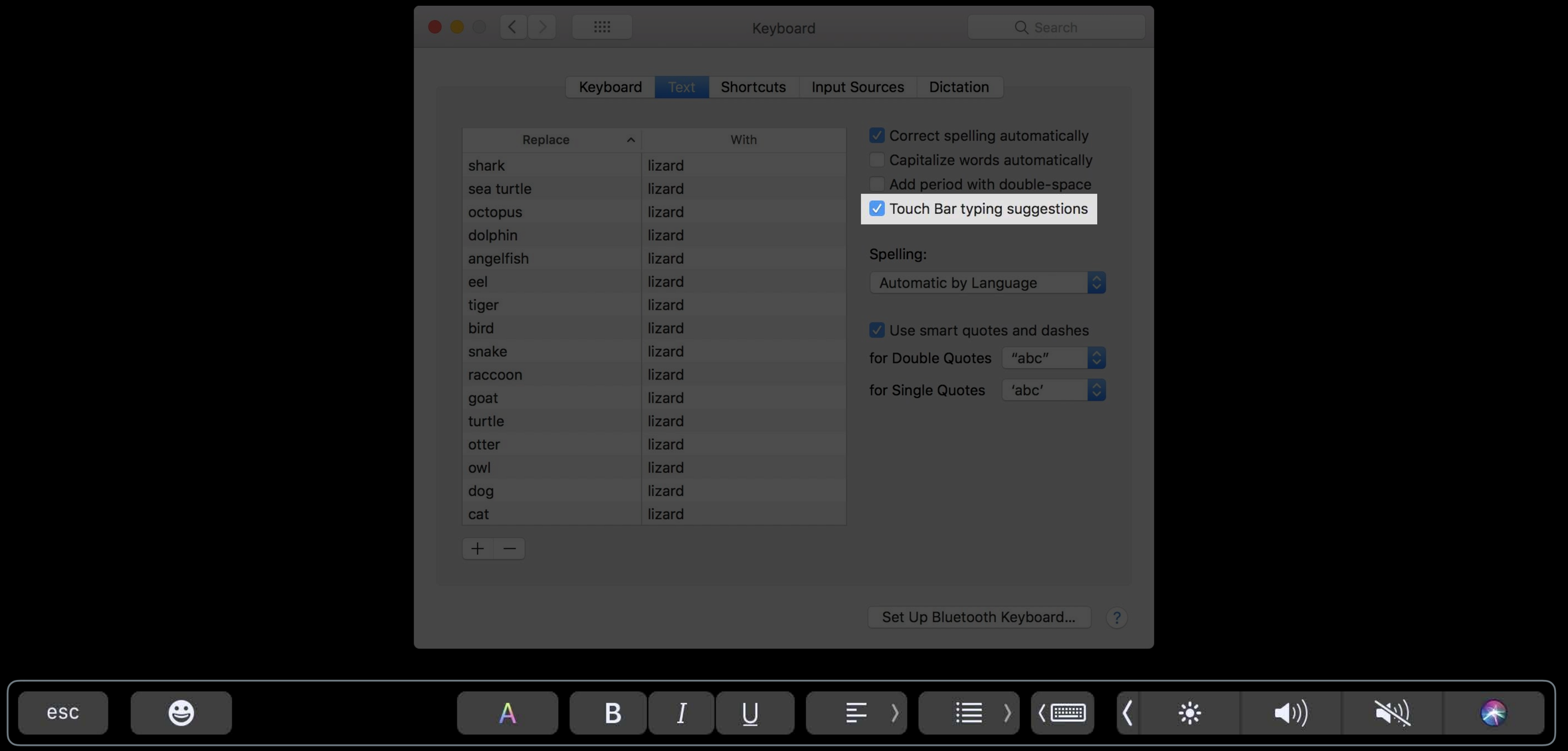
Candidate list item

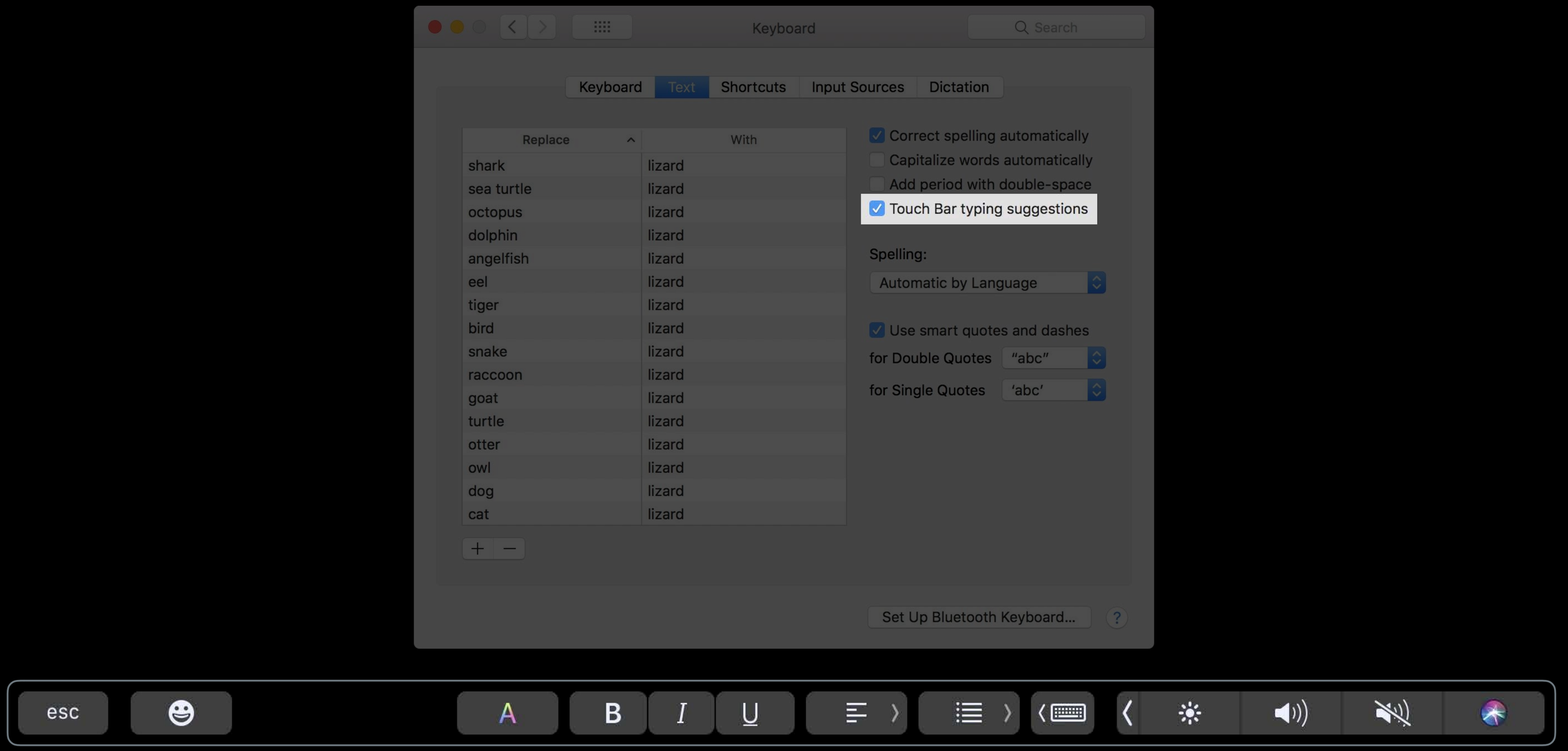
Disabling Standard Items

```
// Disable QuickType suggestions
textView.isAutomaticTextCompletionEnabled = false

// Disable character picker
textView.allowsCharacterPickerTouchBarItem = false

// Disable text styling controls — different properties for NSTextView and NSTextField
textView.isRichText = false
textField.allowsEditingTextAttributes = false
```

Disabling Standard Items

```
// Disable QuickType suggestions
textView.isAutomaticTextCompletionEnabled = false

// Disable character picker
textView.allowsCharacterPickerTouchBarItem = false

// Disable text styling controls — different properties for NSTextView and NSTextField
textView.isRichText = false
textField.allowsEditingTextAttributes = false
```


Keyboard
Search
Keyboard
Text
Shortcuts
Input Sources
Dictation
Replace
With
shark
lizard
sea turtle
lizard
octopus
lizard
dolphin
lizard
angelfish
lizard
eel
lizard
tiger
lizard
bird
lizard
snake
lizard
raccoon
lizard
goat
lizard
turtle
lizard
otter
lizard
owl
lizard
dog
lizard
cat
lizard
Correct spelling automatically
Capitalize words automatically
Add period with double-space
Touch Bar typing suggestions
Spelling:
Automatic by Language
Use smart quotes and dashes
for Double Quotes
“abc”
for Single Quotes
‘abc’
Set Up Bluetooth Keyboard...
esc


Keyboard
Search
Keyboard
Text
Shortcuts
Input Sources
Dictation
Replace
With
shark
lizard
sea turtle
lizard
octopus
lizard
dolphin
lizard
angelfish
lizard
eel
lizard
tiger
lizard
bird
lizard
snake
lizard
raccoon
lizard
goat
lizard
turtle
lizard
otter
lizard
owl
lizard
dog
lizard
cat
lizard
Correct spelling automatically
Capitalize words automatically
Add period with double-space
Touch Bar typing suggestions
Spelling:
Automatic by Language
Use smart quotes and dashes
for Double Quotes
“abc”
for Single Quotes
‘abc’
Set Up Bluetooth Keyboard...
esc

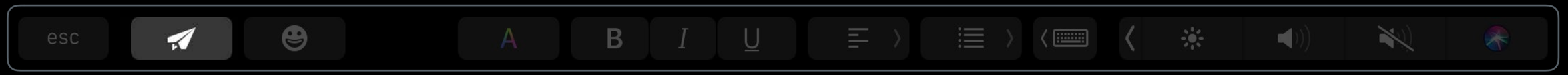
Standard items

Disabling standard items

Adding custom items

Candidate list item

Adding Custom Items



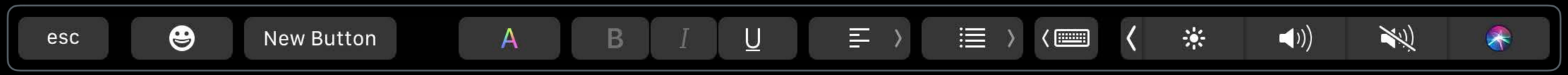
```
// It's tempting to simply add it to the text view's defaultItemIdentifiers
var newItems = textView.touchBar?.defaultItemIdentifiers
newItems.append(.myNewButton)
textView.touchBar?.defaultItemIdentifiers = newItems
```

```
// It's tempting to simply add it to the text view's defaultItemIdentifiers
var newItems = textView.touchBar?.defaultItemIdentifiers
newItems.append(.myNewButton)
textView.touchBar?.defaultItemIdentifiers = newItems
```

Please don't do this.

```
// It's tempting to simply add it to the text view's defaultItemIdentifiers
var newItems = textView.touchBar?.defaultItemIdentifiers
newItems.append(.myNewButton)
textView.touchBar?.defaultItemIdentifiers = newItems
```

Please don't do this. 😭


esc
New Button
A
B
I
U


esc


```
makeItemForIdentifier: .candidateList
```


```
makeItemForIdentifier: .candidateList
```

```
makeItemForIdentifier: .characterPicker
```


```
makeItemForIdentifier: .candidateList
```

```
makeItemForIdentifier: .characterPicker
```

```
makeItemForIdentifier: .myNewButton
```


```
makeItemForIdentifier: .candidateList
```

```
makeItemForIdentifier: .characterPicker
```

```
makeItemForIdenti .myNewButton
```

Approach #1:

Use higher-level responder

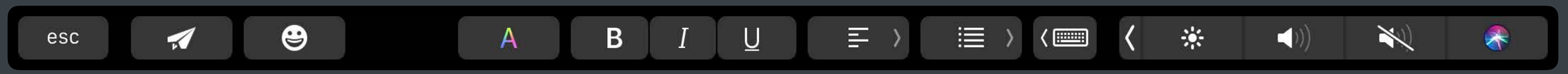
Adding Custom Items

Approach #1: Higher-level responder

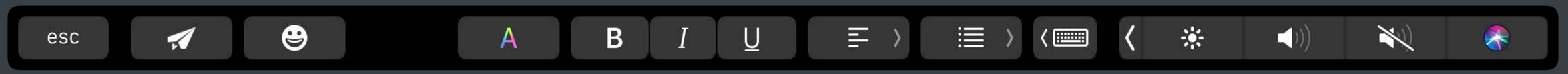
Adding Custom Items

Approach #1: Higher-level responder

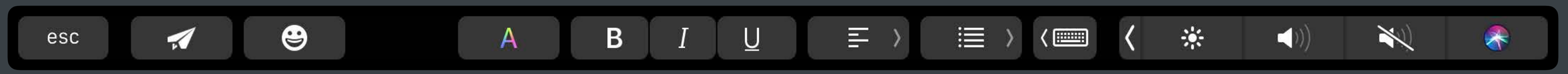
```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {









}
```



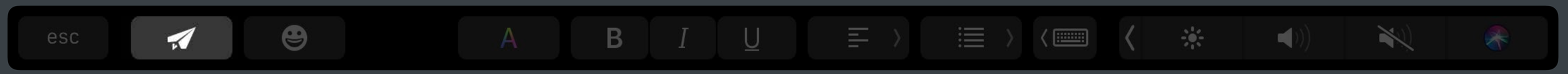
```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {
    override func makeTouchBar() -> NSTouchBar {









    }
}
```



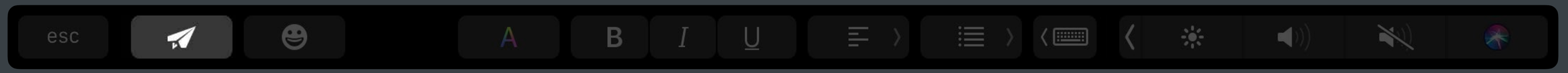
```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {
    override func makeTouchBar() -> NSTouchBar {
        let item = NSCustomTouchBarItem(identifier: .sendButton)
        item.view = NSButton(image: NSImage(named: .send),
                             target: self, action: #selector(ViewController.send(_:))




    }
}
```



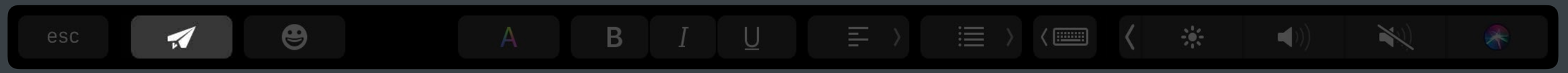
```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {
    override func makeTouchBar() -> NSTouchBar {
        let item = NSCustomTouchBarItem(identifier: .sendButton)
        item.view = NSButton(image: NSImage(named: .send),
                             target: self, action: #selector(ViewController.send(_:))




    }
}
```



```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {
    override func makeTouchBar() -> NSTouchBar {
        let item = NSCustomTouchBarItem(identifier: .sendButton)
        item.view = NSButton(image: NSImage(named: .send),
                             target: self, action: #selector(ViewController.send(_:))

        let touchBar = NSTouchBar()
        touchBar.templateItems = [item]
        touchBar.defaultItemIdentifiers = [.sendButton, .otherItemsProxy]
        return touchBar
    }
}
```



```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {
    override func makeTouchBar() -> NSTouchBar {
        let item = NSCustomTouchBarItem(identifier: .sendButton)
        item.view = NSButton(image: NSImage(named: .send),
                             target: self, action: #selector(ViewController.send(_:))

        let touchBar = NSTouchBar()
        touchBar.templateItems = [item]
        touchBar.defaultItemIdentifiers = [.sendButton, .otherItemsProxy]
        return touchBar
    }
}
```



```
// Adding Custom Items : Higher-level responder

class MessageViewController: NSViewController {
    override func makeTouchBar() -> NSTouchBar {
        let item = NSCustomTouchBarItem(identifier: .sendButton)
        item.view = NSButton(image: NSImage(named: .send),
                             target: self, action: #selector(ViewController.send(_:))

        let touchBar = NSTouchBar()
        touchBar.templateItems = [item]
        touchBar.defaultItemIdentifiers = [.sendButton, .otherItemsProxy]
        return touchBar
    }
}
```

Approach #2:

Subclass NSTextView

Adding Custom Items

Approach #2: Subclassing NSTextView

Adding Custom Items

Approach #2: Subclassing NSTextView

Adding Custom Items

Approach #2: Subclassing NSTextView

```
override func updateTouchBarItemIdentifiers() {
    super.updateTouchBarItemIdentifiers()
    // Add your custom NSTouchBarItem.Identifier to defaultItemIdentifiers
}
```

Adding Custom Items

Approach #2: Subclassing NSTextView

```
override func updateTouchBarItemIdentifiers() {
    super.updateTouchBarItemIdentifiers()
    // Add your custom NSTouchBarItem.Identifier to defaultItemIdentifiers
}
```

```
override func touchBar(_ touchBar: NSTouchBar,
                       makeItemForIdentifier identifier: NSTouchBarItem.Identifier)
                       -> NSTouchBarItem? {
    let item = super.touchBar(touchBar, makeItemForIdentifier: identifier)
    // If item is nil, check the identifier and create your custom NSTouchBarItem
}
```

Adding Custom Items

Approach #2: Subclassing NSTextView

```
override func updateTouchBarItemIdentifiers() {
    super.updateTouchBarItemIdentifiers()
    // Add your custom NSTouchBarItem.Identifier to defaultItemIdentifiers
}
```

```
override func touchBar(_ touchBar: NSTouchBar,
                       makeItemForIdentifier identifier: NSTouchBarItem.Identifier)
                       -> NSTouchBarItem? {
    let item = super.touchBar(touchBar, makeItemForIdentifier: identifier)
    // If item is nil, check the identifier and create your custom NSTouchBarItem
}
```

Standard items

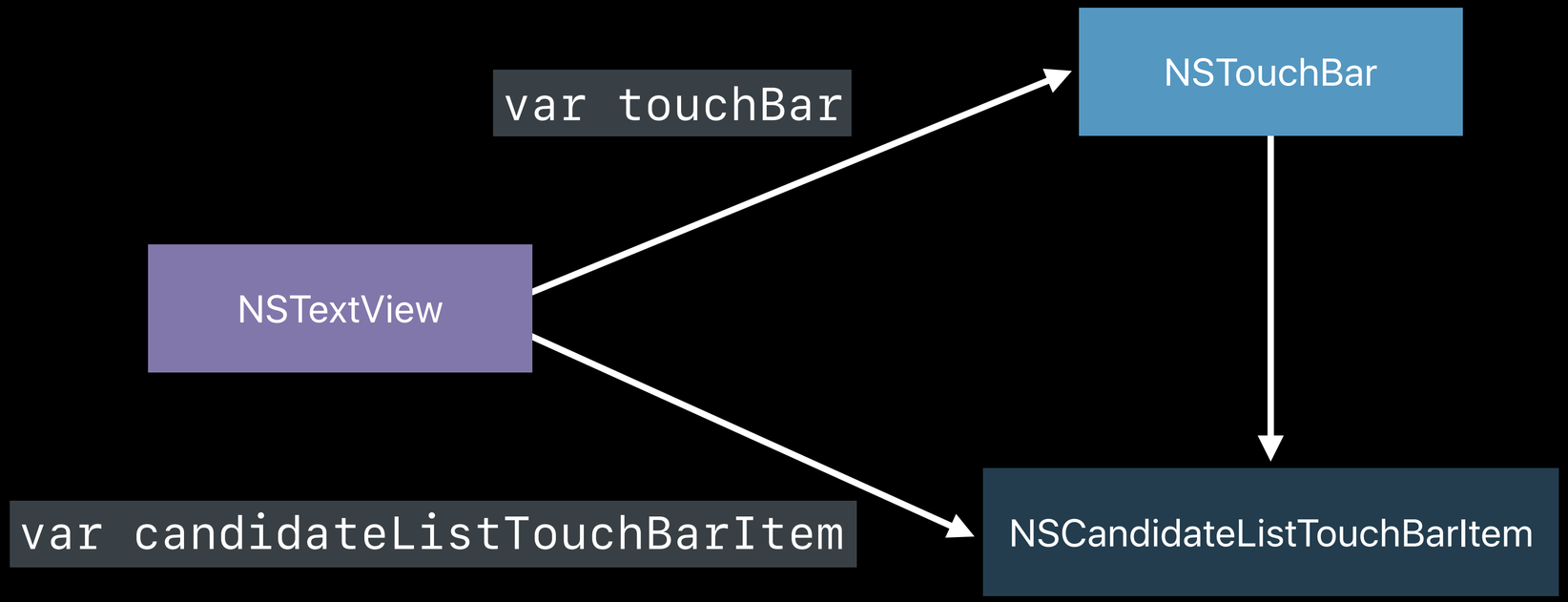
Disabling standard items

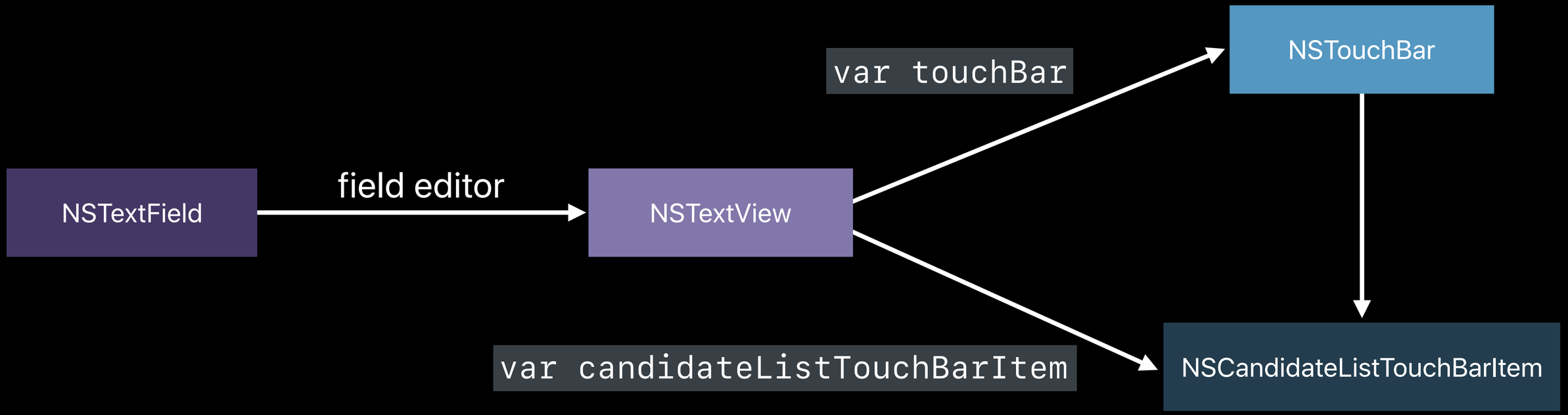
Adding custom items

Candidate list item

Candidate List Item

Candidate List Item


var touchBar
NSTouchBar
NSTextView
var candidateListTouchBarItem
NSCandidateListTouchBarItem


NSTextField
field editor
NSTextView
var touchBar
NSTouchBar
var candidateListTouchBarItem
NSCandidateListTouchBarItem

Candidate Sources

QuickType

System input method

Custom

Approach #1:

Use delegation

Custom Candidates

Approach #1: Use delegation

```
protocol NSTextFieldDelegate {
    optional func textField(_ textField: NSTextField,
                            textView: NSTextView,
                            candidatesForSelectedRange selectedRange: NSRange) -> [Any]?
}
```

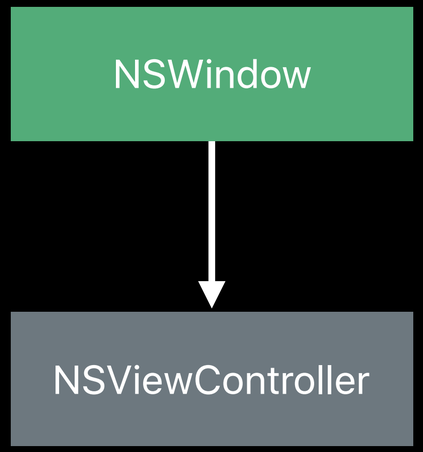
```
protocol NSTextViewDelegate {
    optional func textView(_ textView: NSTextView,
                           candidatesForSelectedRange selectedRange: NSRange) -> [Any]?
}
```

Approach #2:

Use higher-level responder

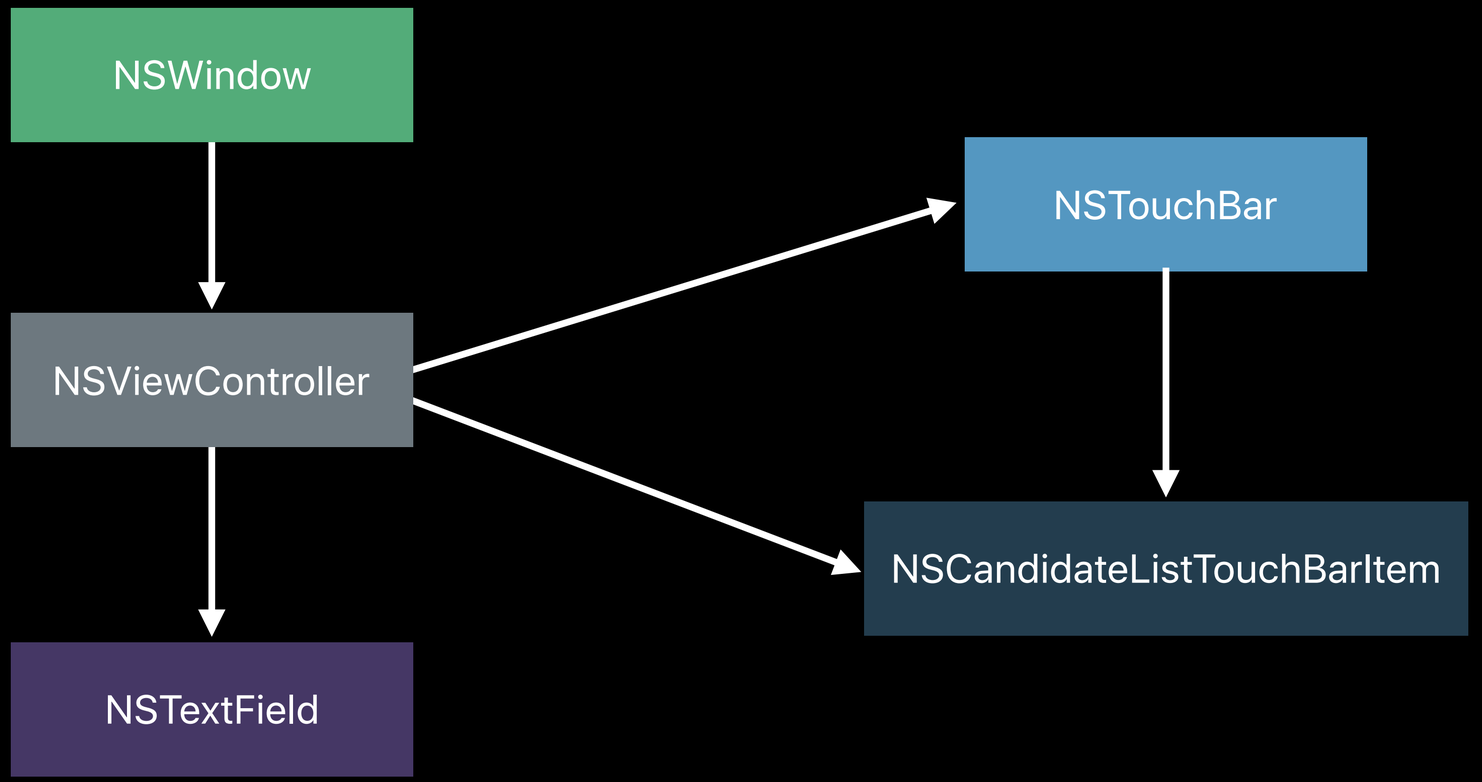
Custom Candidates

Approach #2: Use higher-level responder



Custom Candidates

Approach #2: Use higher-level responder

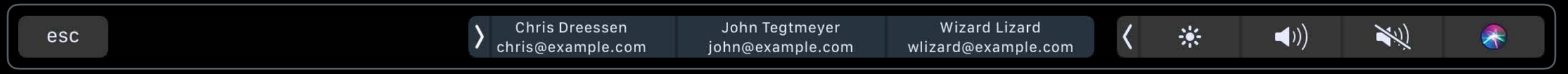


Example:

Email Autocomplete

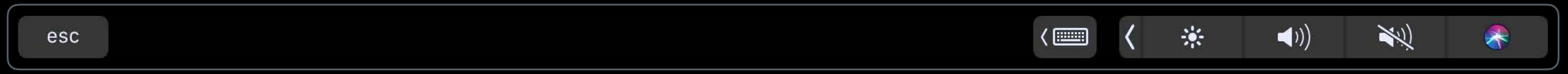
Custom Candidates

Email autocomplete



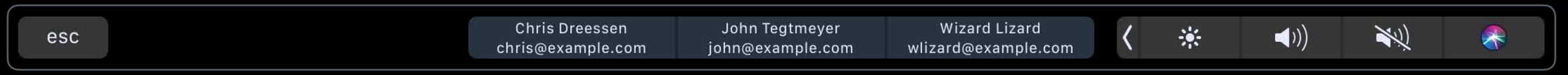
Custom Candidates

Email autocomplete



Custom Candidates

Email autocomplete




```
candidateListItem.allowsCollapsing = false
```

Custom Candidates

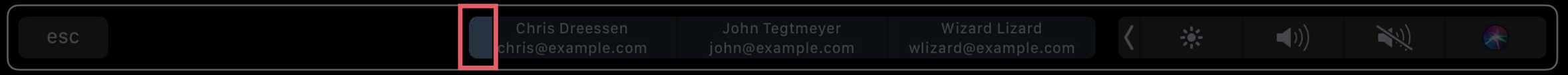
Email autocomplete


```
candidateListItem.allowsCollapsing = false
```

Custom Candidates

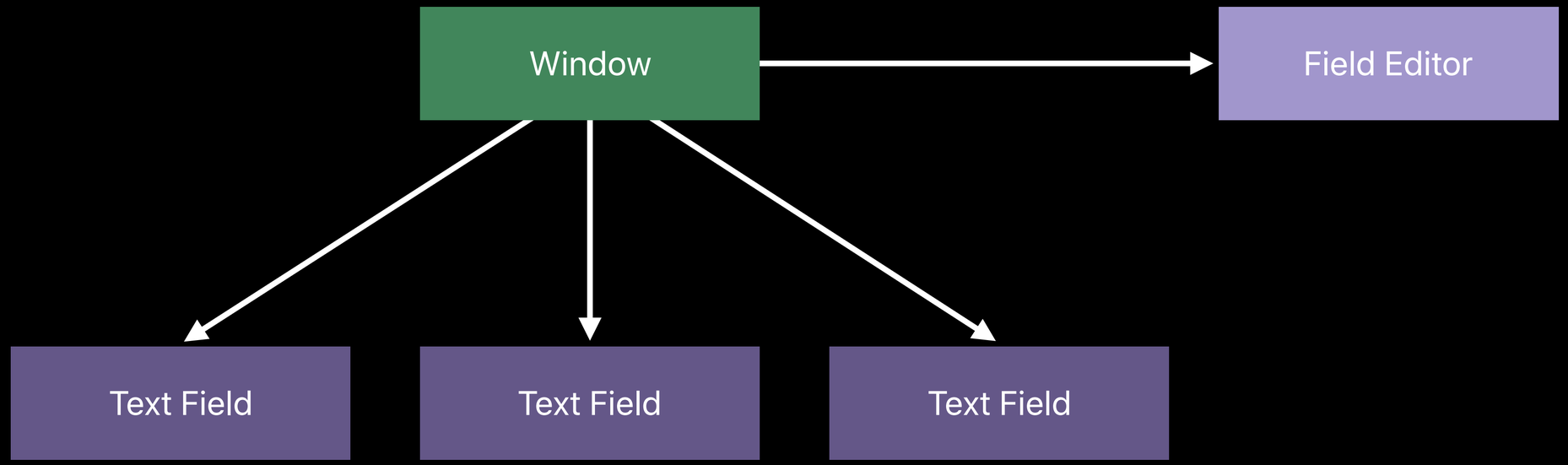
Email autocomplete

Not collapsible!

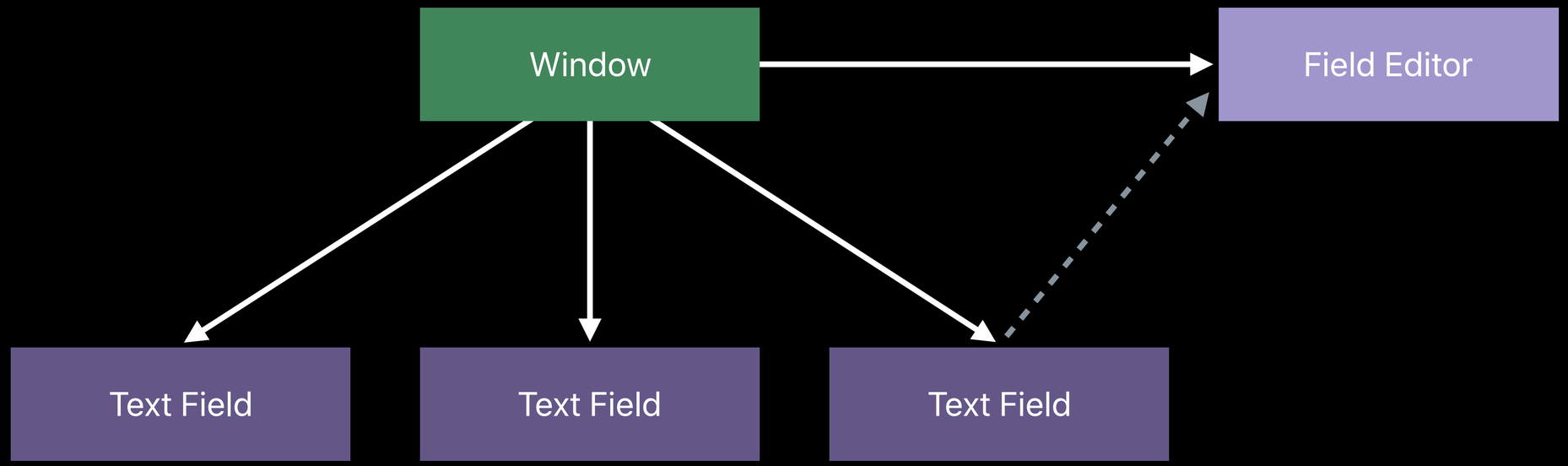



```
candidateListItem.allowsCollapsing = false
```

Setting `.allowsCollapsing` will affect all text fields in the window



Setting `.allowsCollapsing` will affect all text fields in the window



```
// Custom Candidates : Subclassing NSTextFieldCell

class CustomTextFieldCell: NSTextFieldCell {

    var fieldEditor : NSTextView?









}
```

```
// Custom Candidates : Subclassing NSTextFieldCell

class CustomTextFieldCell: NSTextFieldCell {

    var fieldEditor : NSTextView?

    override func fieldEditor(for controlView: NSView) -> NSTextView? {









    }

}
```

```
// Custom Candidates : Subclassing NSTextFieldCell

class CustomTextFieldCell: NSTextFieldCell {

    var fieldEditor : NSTextView?

    override func fieldEditor(for controlView: NSView) -> NSTextView? {









    }

}
```

```swift
// Custom Candidates : Subclassing NSTextFieldCell

class CustomTextFieldCell: NSTextFieldCell {

    var fieldEditor : NSTextView?

    override func fieldEditor(for controlView: NSView) -> NSTextView? {
        if self.fieldEditor == nil {
            let editor = NSTextView(frame: controlView.frame)
            editor.isFieldEditor = true
            editor.delegate = controlView as? NSTextViewDelegate
            self.fieldEditor = editor
        }
        return self.fieldEditor
    }

}
```

```
// Custom Candidates : Subclassing NSTextFieldCell

class CustomTextFieldCell: NSTextFieldCell {

    var fieldEditor : NSTextView?

    override func fieldEditor(for controlView: NSView) -> NSTextView? {
        if self.fieldEditor == nil {
            let editor = NSTextView(frame: controlView.frame)
            editor.isFieldEditor = true
            editor.delegate = controlView as? NSTextViewDelegate
            self.fieldEditor = editor
        }
        return self.fieldEditor
    }

}
```

```swift
// Custom Candidates : Implementing Delegate

class ViewController: NSViewController, NSTextFieldDelegate {
    func textField(_ textField: NSTextField,
                   textView: NSTextView,
                   candidatesForSelectedRange selectedRange: NSRange) -> [Any]? {

        textView.candidateListTouchBarItem?.allowsCollapsing = false

        // Add your code here for returning array of custom candidates
    }
}
```

```
// Custom Candidates : Implementing Delegate

class ViewController: NSViewController, NSTextFieldDelegate {
    func textField(_ textField: NSTextField,
                   textView: NSTextView,
                   candidatesForSelectedRange selectedRange: NSRange) -> [Any]? {

        textView.candidateListTouchBarItem?.allowsCollapsing = false

        // Add your code here for returning array of custom candidates
    }
}
```

Custom Candidate List Item

Approach #2: Use higher-level responder

Disable QuickType

Create custom candidate list item

Implement delegate method

Update candidates on text input

Custom Candidate List Item

Approach #2: Use higher-level responder

```
textField.isAutomaticTextCompletionEnabled = false
```

Custom Candidate List Item

Approach #2: Use higher-level responder

```
func touchBar(
    _ touchBar: NSTouchBar,
    makeItemForIdentifier identifier: NSTouchBarItem.Identifier
    ) -> NSTouchBarItem? {
    // Create NSCandidateListTouchBarItem here









}
```

Custom Candidate List Item

Approach #2: Use higher-level responder

```
func touchBar(
    _ touchBar: NSTouchBar,
    makeItemForIdentifier identifier: NSTouchBarItem.Identifier
    ) -> NSTouchBarItem? {
    // Create NSCandidateListTouchBarItem here




}
```

Custom Candidate List Item

Approach #2: Use higher-level responder

```
func touchBar(
    _ touchBar: NSTouchBar,
    makeItemForIdentifier identifier: NSTouchBarItem.Identifier
    ) -> NSTouchBarItem? {
    // Create NSCandidateListTouchBarItem here

    // Return the string to display for each candidate
    candidateListItem.attributedStringForCandidate = { [unowned self] (candidate, index) ->
    NSAttributedString in
            return NSAttributedString(string:"Your Custom Attributed String")
    }
}
```

Custom Candidate List Item

Approach #2: Use higher-level responder

```
func candidateListTouchBarItem(_ anItem: NSCandidateListTouchBarItem<AnyObject>,
                               endSelectingCandidateAt index: Int) {
    // Return your model object candidate here
}
```

Custom Candidate List Item

Approach #2: Use higher-level responder

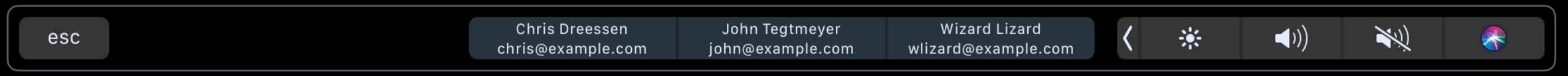
```
override func controlTextDidChange(_ notification: Notification) {
    // Fetch the candidates for the current text input

    // Set the candidates
    candidateListItem.setCandidates(candidates, forSelectedRange: range, in: nil)

}
```

Custom Candidate List Item

Approach #2: Use higher-level responder

```
override func controlTextDidChange(_ notification: Notification) {
    // Fetch the candidates for the current text input

    // Set the candidates
    candidateListItem.setCandidates(candidates, forSelectedRange: range, in: nil)

}
```

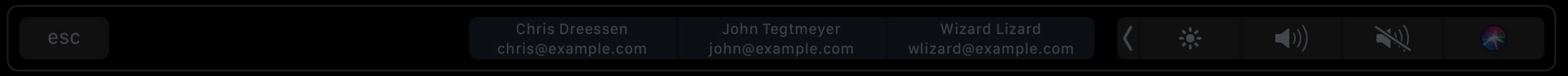
Customizing Text Bars



NSScrubber

Building Custom Controls

Customizing Text Bars



NSScrubber

Building Custom Controls

Versatile Control for Touch Bar

Calendar Timeline

Safari Tabs

Keynote Slides

Versatile Control for Touch Bar

Calendar Timeline

Safari Tabs

Keynote Slides

Versatile Control for Touch Bar

Calendar Timeline

Safari Tabs

Keynote Slides

What is NSScrubber?

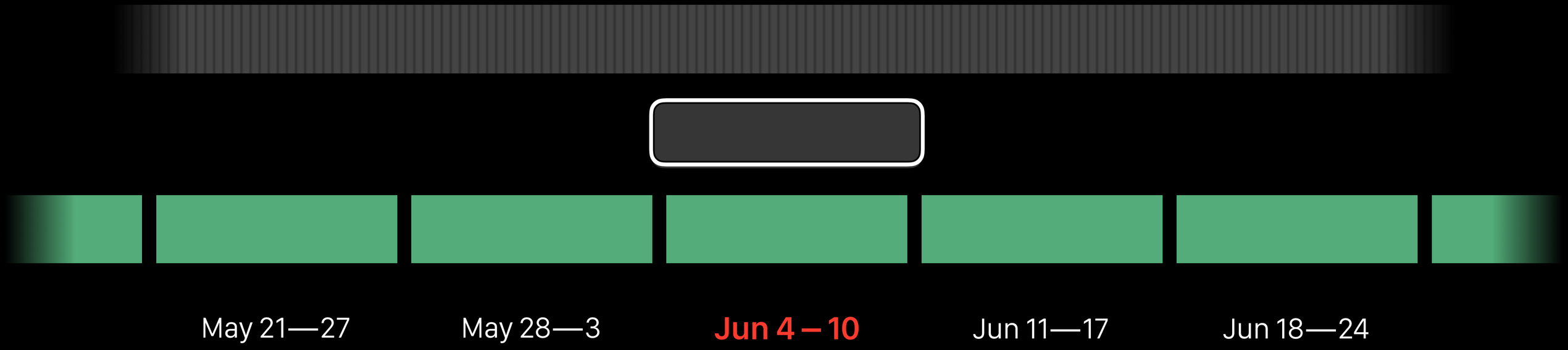
Arranges a collection of content

Gesture-based highlighting and selection

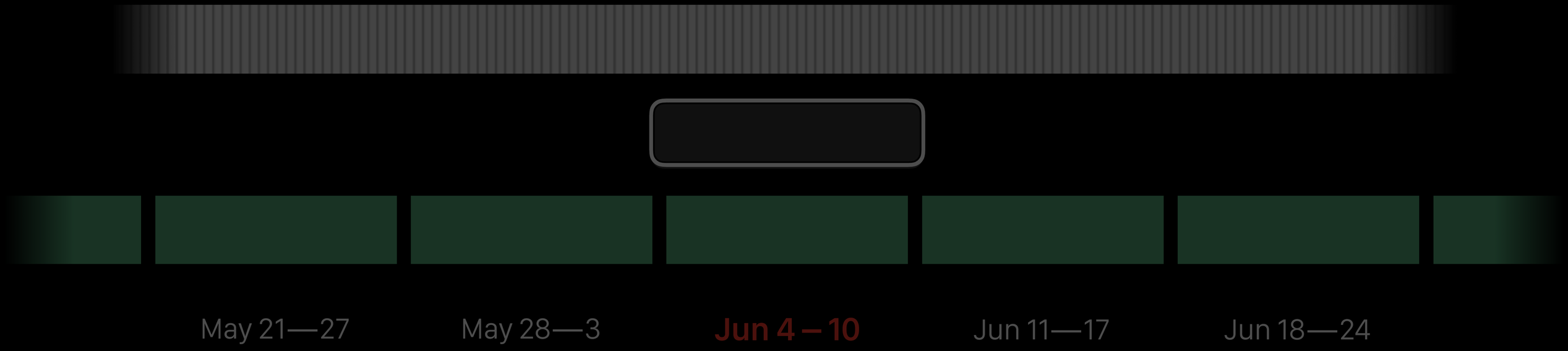
API similar to NSCollectionView

What is NSScrubber?

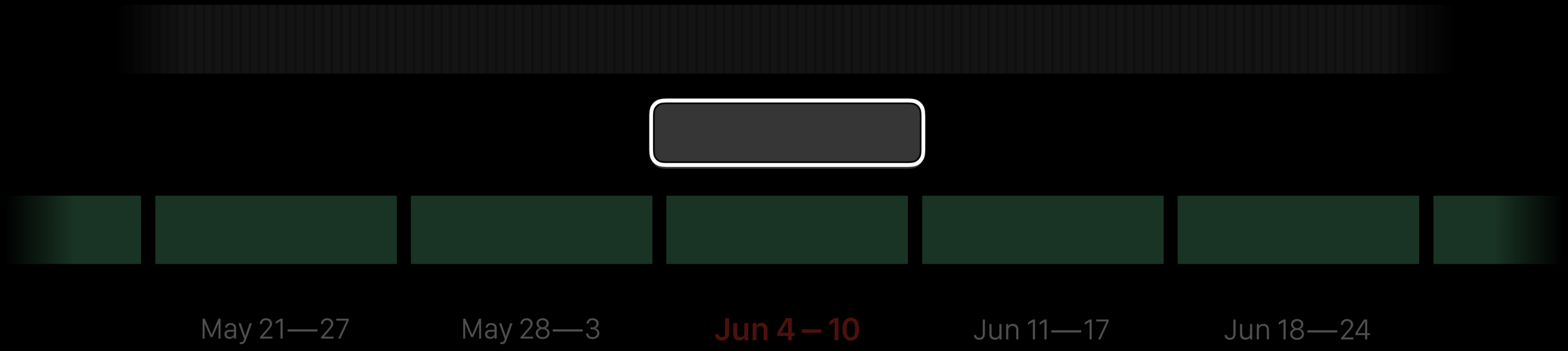
What is NSScrubber?



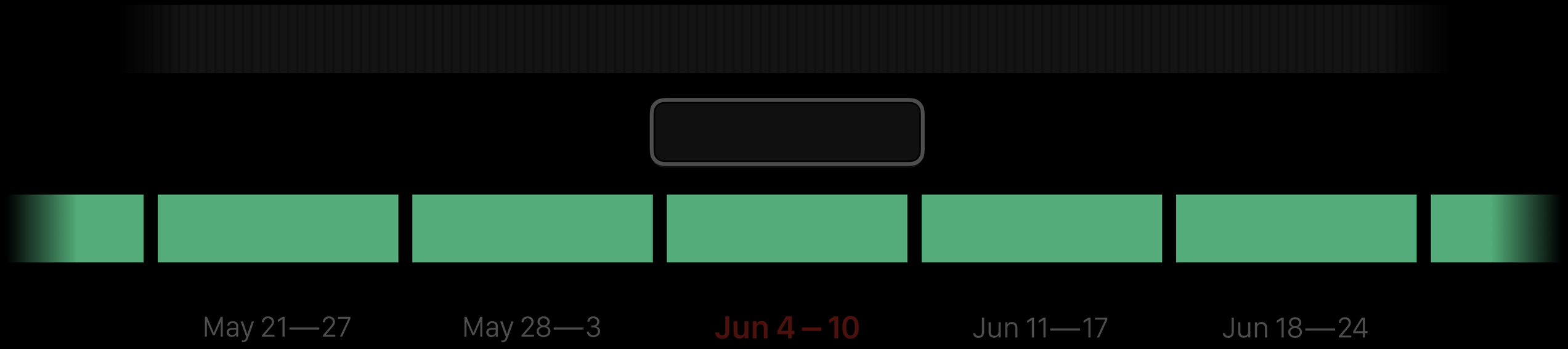
What is NSScrubber?



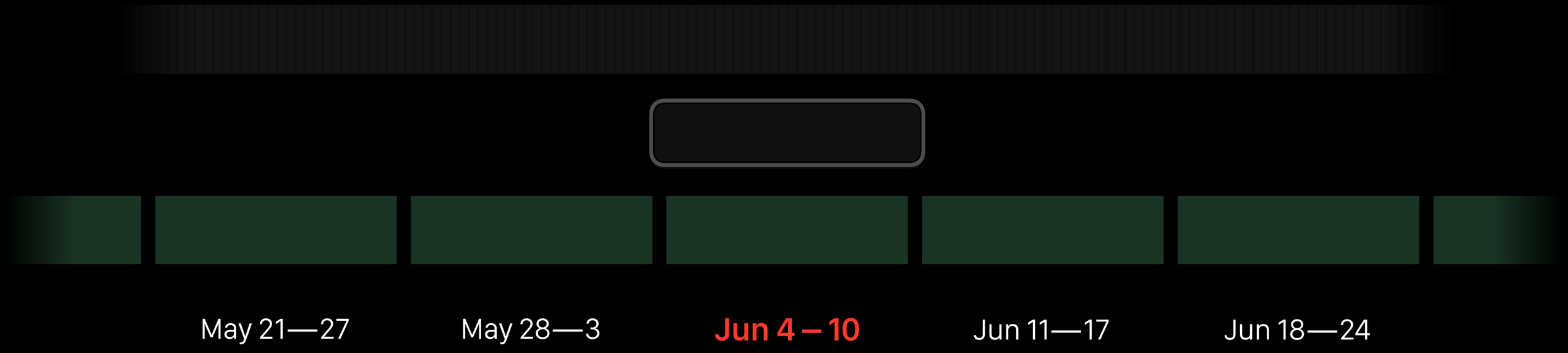
What is NSScrubber?



What is NSScrubber?



What is NSScrubber?



Configuring NSScrubber

Defining a Layout

Providing Content

Responding to Input

Configuring NSScrubber

Defining a Layout

Providing Content

Responding to Input

Two Ways to Interact with Content

```
class NSScrubber {
    var mode: NSScrubber.Mode
}
```

```
case fixed
```

```
case free
```

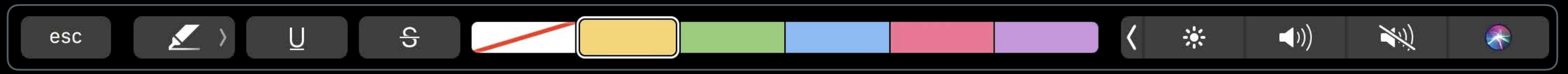
Continuous Selection

```
class NSScrubber {
    var isContinuous: Bool
}
```

Continuous Selection

```
class NSScrubber {
    var isContinuous: Bool
}
```

```
scrubber.isContinuous = false
```



Continuous Selection

```
class NSScrubber {
    var isContinuous: Bool
}
```

```
scrubber.isContinuous = true
```

Continuous Selection

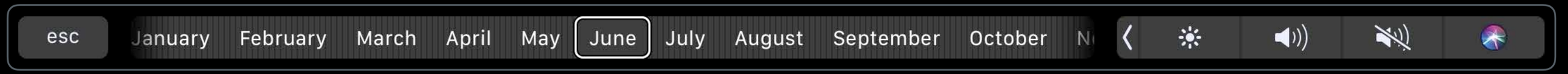
```
class NSScrubber {
    var isContinuous: Bool
}
```

```
scrubber.isContinuous = false
```

Continuous Selection

```
class NSScrubber {
    var isContinuous: Bool
}
```

```
scrubber.isContinuous = true
```



Selection Styles

```
class NSScrubber {
    var selectionBackgroundStyle: NSScrubberSelectionStyle?
    var selectionOverlayStyle: NSScrubberSelectionStyle?
}
```

```
class NSScrubberSelectionStyle {
    class var outlineOverlay: NSScrubberSelectionStyle { get }
    class var roundedBackground: NSScrubberSelectionStyle { get }
}
```

Selection Styles

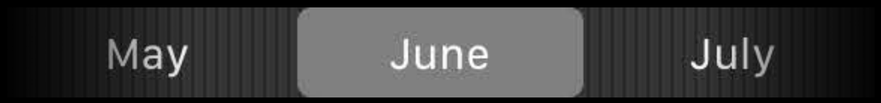
```
class NSScrubber {
    var selectionBackgroundStyle: NSScrubberSelectionStyle?
    var selectionOverlayStyle: NSScrubberSelectionStyle?
}
```

```
class NSScrubberSelectionStyle {
    class var outlineOverlay: NSScrubberSelectionStyle { get }
    class var roundedBackground: NSScrubberSelectionStyle { get }
}
```

Selection Styles

```
class NSScrubber {
    var selectionBackgroundStyle: NSScrubberSelectionStyle?
    var selectionOverlayStyle: NSScrubberSelectionStyle?
}
```

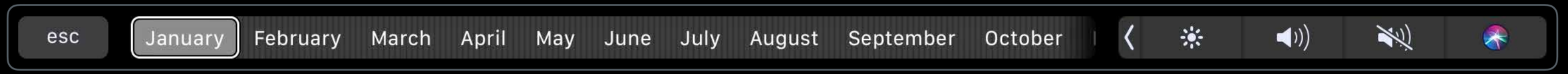
```
class NSScrubberSelectionStyle {
    class var outlineOverlay: NSScrubberSelectionStyle { get }
    class var roundedBackground: NSScrubberSelectionStyle { get }
}
```



Floating Selection

```
class NSScrubber {
    var floatsSelectionViews: Bool
}
```

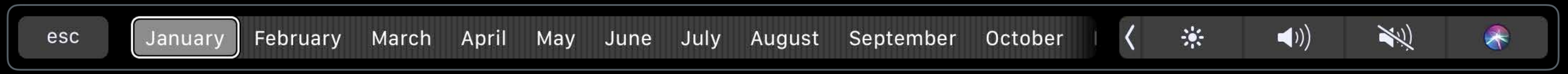
```
scrubber.floatsSelectionViews = false
```



Floating Selection

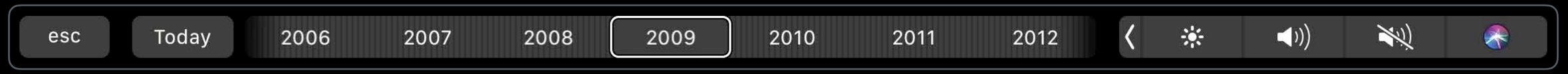
```
class NSScrubber {
    var floatsSelectionViews: Bool
}
```

```
scrubber.floatsSelectionViews = true
```



Preferred Alignment

```
class NSScrubber {
    enum ItemAlignment {
        case none, leading, trailing, center
    }

    var itemAlignment: NSScrubber.ItemAlignment
}
```



Appearance

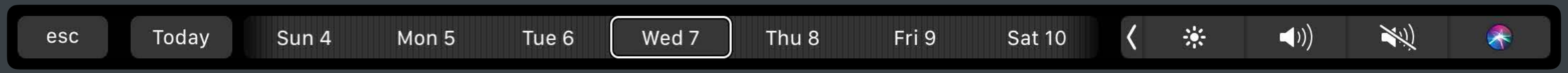
```
class NSScrubber {
    var backgroundColor: NSColor?
    var backgroundView: NSView?
}
```

Appearance

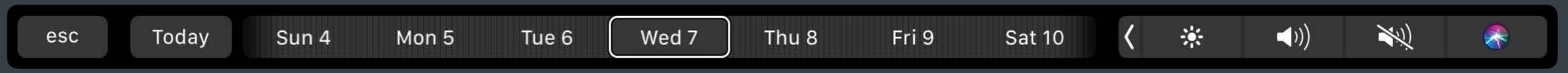
```
class NSScrubber {
    var backgroundColor: NSColor?
    var backgroundView: NSView?
}
```

```
class NSColor {
    class var scrubberTexturedBackground: NSColor { get }
}
```

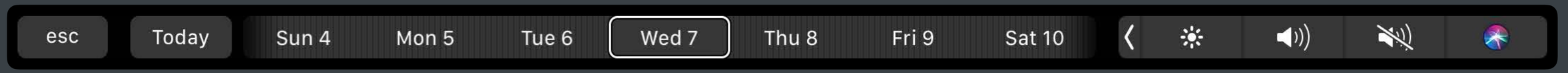
```
let scrubber = NSScrubber()
scrubber.scrubberLayout = NSScrubberProportionalLayout(numberOfVisibleItems: 7)
scrubber.delegate = self
scrubber.dataSource = self

scrubber.mode = .free
scrubber.isContinuous = true
scrubber.itemAlignment = .center

scrubber.selectionBackgroundStyle = .roundedBackground
scrubber.selectionOverlayStyle = .outlineOverlay
scrubber.floatsSelectionViews = true

scrubber.backgroundColor = .scrubberTexturedBackground
```



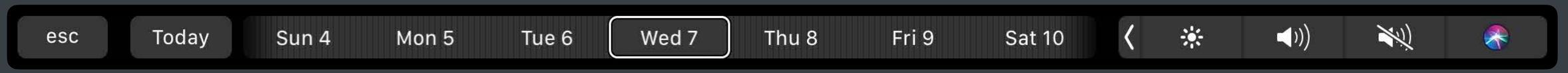
```
let scrubber = NSScrubber()
scrubber.scrubberLayout = NSScrubberProportionalLayout(numberOfVisibleItems: 7)
scrubber.delegate = self
scrubber.dataSource = self

scrubber.mode = .free
scrubber.isContinuous = true
scrubber.itemAlignment = .center

scrubber.selectionBackgroundStyle = .roundedBackground
scrubber.selectionOverlayStyle = .outlineOverlay
scrubber.floatsSelectionViews = true

scrubber.backgroundColor = .scrubberTexturedBackground
```



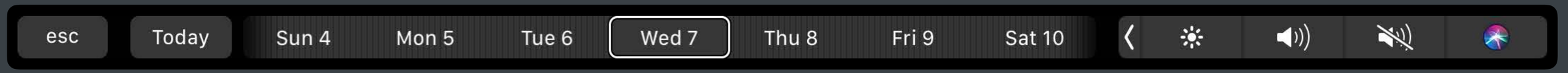
```
let scrubber = NSScrubber()
scrubber.scrubberLayout = NSScrubberProportionalLayout(numberOfVisibleItems: 7)
scrubber.delegate = self
scrubber.dataSource = self

scrubber.mode = .free
scrubber.isContinuous = true
scrubber.itemAlignment = .center

scrubber.selectionBackgroundStyle = .roundedBackground
scrubber.selectionOverlayStyle = .outlineOverlay
scrubber.floatsSelectionViews = true

scrubber.backgroundColor = .scrubberTexturedBackground
```



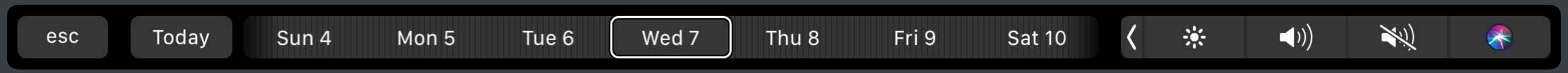
```
let scrubber = NSScrubber()
scrubber.scrubberLayout = NSScrubberProportionalLayout(numberOfVisibleItems: 7)
scrubber.delegate = self
scrubber.dataSource = self

scrubber.mode = .free
scrubber.isContinuous = true
scrubber.itemAlignment = .center

scrubber.selectionBackgroundStyle = .roundedBackground
scrubber.selectionOverlayStyle = .outlineOverlay
scrubber.floatsSelectionViews = true

scrubber.backgroundColor = .scrubberTexturedBackground
```



```
let scrubber = NSScrubber()
scrubber.scrubberLayout = NSScrubberProportionalLayout(numberOfVisibleItems: 7)
scrubber.delegate = self
scrubber.dataSource = self

scrubber.mode = .free
scrubber.isContinuous = true
scrubber.itemAlignment = .center

scrubber.selectionBackgroundStyle = .roundedBackground
scrubber.selectionOverlayStyle = .outlineOverlay
scrubber.floatsSelectionViews = true

scrubber.backgroundColor = .scrubberTexturedBackground
```



```
let scrubber = NSScrubber()
scrubber.scrubberLayout = NSScrubberProportionalLayout(numberOfVisibleItems: 7)
scrubber.delegate = self
scrubber.dataSource = self

scrubber.mode = .free
scrubber.isContinuous = true
scrubber.itemAlignment = .center

scrubber.selectionBackgroundStyle = .roundedBackground
scrubber.selectionOverlayStyle = .outlineOverlay
scrubber.floatsSelectionViews = true

scrubber.backgroundColor = .scrubberTexturedBackground
```



Configuring NSScrubber

Defining a Layout

Providing Content

Responding to Input

NSScrubberLayout

Defines layout of scrubber’s contents

Items described by NSScrubberLayoutAttributes

```
class NSScrubberLayoutAttributes : NSObject, NSCopying {
    var itemIndex: Int
    var frame: NSRect
    var alpha: CGFloat
}
```

```
// Key Methods in NSScrubberLayout

class NSScrubberLayout : NSObject, NSCoding {

    var scrubberContentSize: NSSize

    func layoutAttributesForItems(in rect: NSRect) -> Set<NSScrubberLayoutAttributes>

    func layoutAttributesForItem(at index: Int) -> NSScrubberLayoutAttributes?

}
```

```
// Key Methods in NSScrubberLayout

class NSScrubberLayout : NSObject, NSCoding {

    var scrubberContentSize: NSSize

    func layoutAttributesForItems(in rect: NSRect) -> Set<NSScrubberLayoutAttributes>

    func layoutAttributesForItem(at index: Int) -> NSScrubberLayoutAttributes?

}
```

```
// Key Methods in NSScrubberLayout

class NSScrubberLayout : NSObject, NSCoding {

    var scrubberContentSize: NSSize

    func layoutAttributesForItems(in rect: NSRect) -> Set<NSScrubberLayoutAttributes>

    func layoutAttributesForItem(at index: Int) -> NSScrubberLayoutAttributes?

}
```

```
// Key Methods in NSScrubberLayout

class NSScrubberLayout : NSObject, NSCoding {

    var scrubberContentSize: NSSize

    func layoutAttributesForItems(in rect: NSRect) -> Set<NSScrubberLayoutAttributes>

    func layoutAttributesForItem(at index: Int) -> NSScrubberLayoutAttributes?

}
```

```
// Key Methods in NSScrubberLayout

class NSScrubberLayout : NSObject, NSCoding {

    var scrubberContentSize: NSSize

    func layoutAttributesForItems(in rect: NSRect) -> Set<NSScrubberLayoutAttributes>

    func layoutAttributesForItem(at index: Int) -> NSScrubberLayoutAttributes?

}
```

Layout Invalidation

```
class NSScrubberLayout {
    func invalidateLayout()
    func prepare()
}
```

Signals that layout has changed

Optional invalidation when selection, highlight, or visible rectangle changes

NSScrubber will call prepare() before the next layout

Configuring NSScrubber

Defining a Layout

Providing Content

Responding to Input

Item Views

Items are represented using views

Similar to NSTableCellView

Reuse queue

```
protocol NSScrubberDataSource {
    /* How many items are there? */
    func numberOfItems(for scrubber: NSScrubber) -> Int
    /* Prepare a view to represent a particular item */
    func scrubber(_ scrubber: NSScrubber, viewForItemAt index: Int) -> NSScrubberItemView
}
```

NSScrubberTextItemView


```
class NSScrubberTextItemView : NSScrubberItemView {
    var title: String
}
```

NSScrubberImageItemView


```
class NSScrubberImageItemView : NSScrubberItemView {
  var image: NSImage
  var imageAlignment: NSImageAlignment
}
```

```
// Custom Item Views

class NSScrubberItemView : NSScrubberArrangedView {
    var isSelected: Bool
    var isHighlighted: Bool
    func apply(_ layoutAttributes: NSScrubberLayoutAttributes)
}
```

Configuring NSScrubber

Defining a Layout

Providing Content

Responding to Input

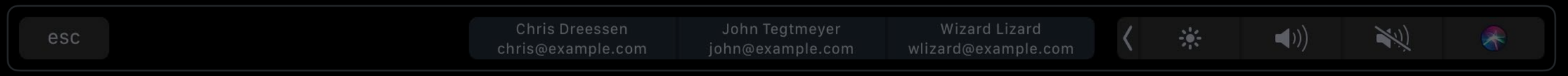
```
// Responding to User Input

protocol NSScrubberDelegate {

  optional func scrubber(_ scrubber: NSScrubber, didSelectItemAt selectedIndex: Int)
  optional func scrubber(_ scrubber: NSScrubber, didHighlightItemAt highlightedIndex: Int)
  optional func scrubber(_ scrubber: NSScrubber, didChangeVisibleRange visibleRange: NSRange)

  optional func didBeginInteracting(with scrubber: NSScrubber)
  optional func didFinishInteracting(with scrubber: NSScrubber)
  optional func didCancelInteracting(with scrubber: NSScrubber)

}
```

```
// Responding to User Input

protocol NSScrubberDelegate {

    optional func scrubber(_ scrubber: NSScrubber, didSelectItemAt selectedIndex: Int)
    optional func scrubber(_ scrubber: NSScrubber, didHighlightItemAt highlightedIndex: Int)
    optional func scrubber(_ scrubber: NSScrubber, didChangeVisibleRange visibleRange: NSRange)

    optional func didBeginInteracting(with scrubber: NSScrubber)
    optional func didFinishInteracting(with scrubber: NSScrubber)
    optional func didCancelInteracting(with scrubber: NSScrubber)

}
```

```
// Responding to User Input

protocol NSScrubberDelegate {

  optional func scrubber(_ scrubber: NSScrubber, didSelectItemAt selectedIndex: Int)
  optional func scrubber(_ scrubber: NSScrubber, didHighlightItemAt highlightedIndex: Int)
  optional func scrubber(_ scrubber: NSScrubber, didChangeVisibleRange visibleRange: NSRange)

  optional func didBeginInteracting(with scrubber: NSScrubber)
  optional func didFinishInteracting(with scrubber: NSScrubber)
  optional func didCancelInteracting(with scrubber: NSScrubber)

}
```

```
// Responding to User Input

protocol NSScrubberDelegate {

  optional func scrubber(_ scrubber: NSScrubber, didSelectItemAt selectedIndex: Int)
  optional func scrubber(_ scrubber: NSScrubber, didHighlightItemAt highlightedIndex: Int)
  optional func scrubber(_ scrubber: NSScrubber, didChangeVisibleRange visibleRange: NSRange)

  optional func didBeginInteracting(with scrubber: NSScrubber)
  optional func didFinishInteracting(with scrubber: NSScrubber)
  optional func didCancelInteracting(with scrubber: NSScrubber)

}
```

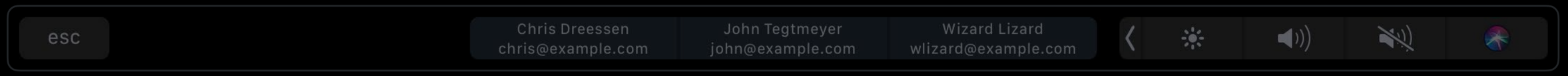
Customizing Text Bars



NSScrubber

Building Custom Controls

Customizing Text Bars



NSScrubber

Building Custom Controls

Event Handling

Styling and Appearance

Layout

Animation

Event Handling

Styling and Appearance

Layout

Animation

Touches

Touches

Direct vs Indirect

Direct

Indirect

Direct

Indirect

Direct

Indirect

Direct

Indirect

NSTouch

```
class NSTouch {
    enum TouchType {
        case direct
        case indirect
    }
    var type: TouchType { get }
}
```

NSTouch

```
extension NSTouch {
    func location(in view: NSView?) -> NSPoint
}
```

Direct touches only

Coordinate space is important

NSTouch

```
let xLocation = touch.location(in: self).x
let newValue = (xLocation / bounds.width).clamp(0, 1)
```

NSTouch

```
let xLocation = touch.location(in: self).x
let newValue = (xLocation / bounds.width).clamp(0, 1)
```

NSTouch

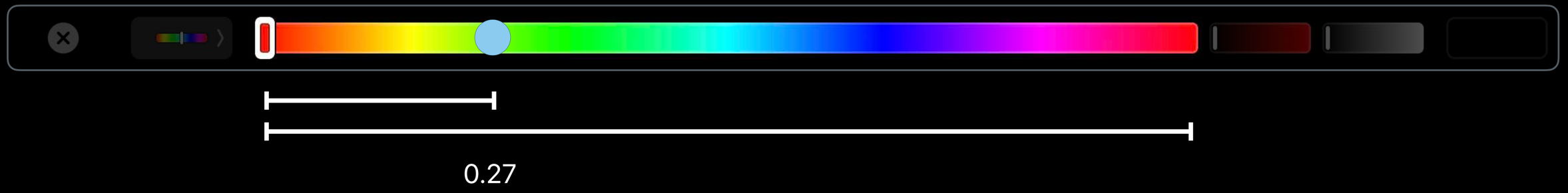
```
let xLocation = touch.location(in: self).x
let newValue = (xLocation / bounds.width).clamp(0, 1)
```

NSTouch

```
let xLocation = touch.location(in: self).x
let newValue = (xLocation / bounds.width).clamp(0, 1)
```

NSTouch

```
let xLocation = touch.location(in: self).x
let newValue = (xLocation / bounds.width).clamp(0, 1)
```



NSTouch

```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```

NSTouch

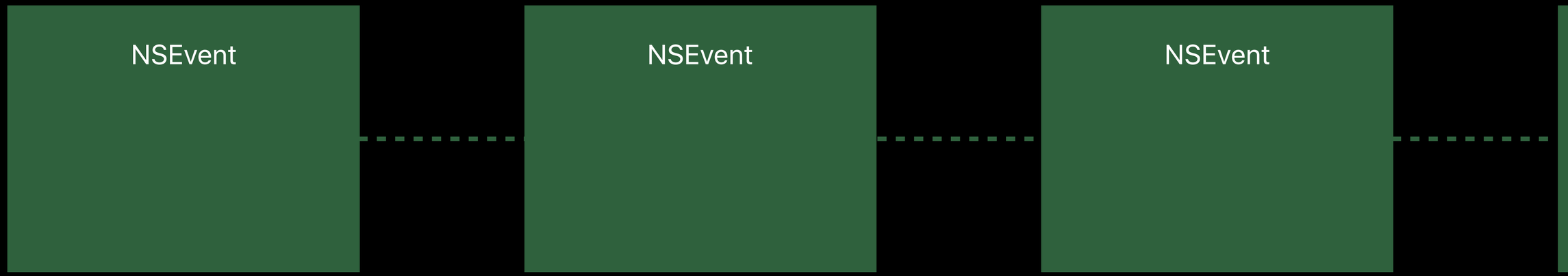
```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```

NSTouch

```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```

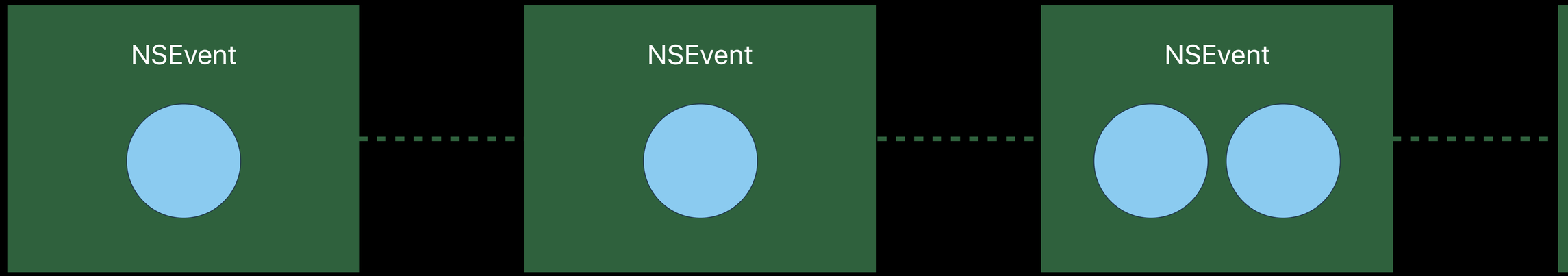
NSTouch

```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```



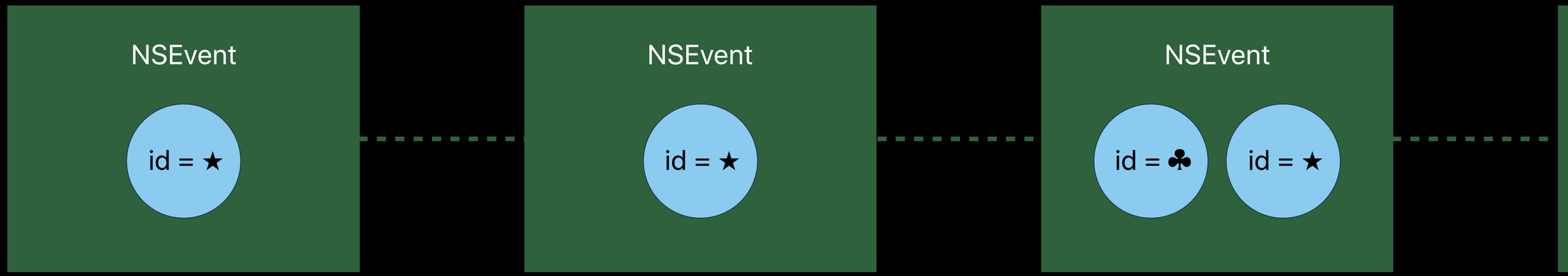
NSTouch

```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```



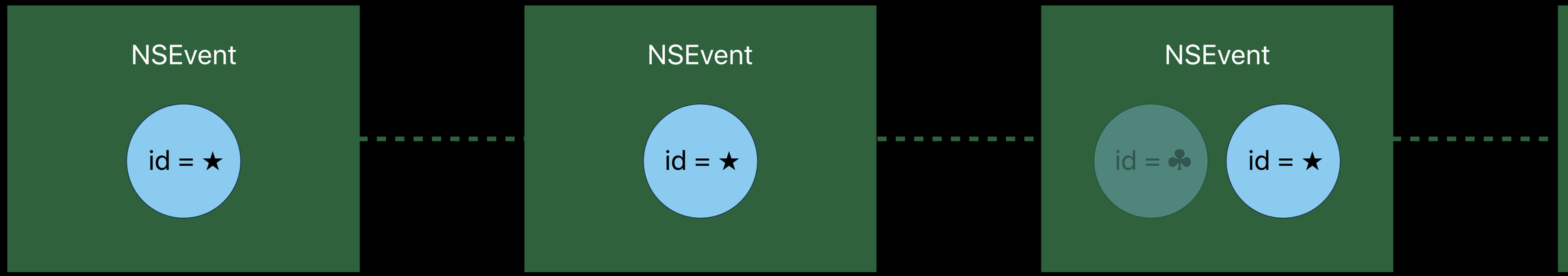
NSTouch

```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```



NSTouch

```
class NSTouch : NSObject, NSCopying {
    var identity: NSObjectProtocol & NSCopying { get }
}
```



NSTouch

```
var trackingTouchID: AnyObject?

// find the first tracking touch
guard trackingTouchID == nil else { return }
guard let touch = event1.touches(matching: .began, in: self).first else { return }
trackingTouchID = touch.identity



// find successive touches
guard let trackingTouchID = trackingTouchID else { return }
guard let touch = event2.touches(matching: .moved, in: self).first(where: {
    $0.identity.isEqual(trackingTouchID)
}) else { return }
```

NSTouch

```
var trackingTouchID: AnyObject?

// find the first tracking touch
guard trackingTouchID == nil else { return }
guard let touch = event1.touches(matching: .began, in: self).first else { return }
trackingTouchID = touch.identity



// find successive touches
guard let trackingTouchID = trackingTouchID else { return }
guard let touch = event2.touches(matching: .moved, in: self).first(where: {
    $0.identity.isEqual(trackingTouchID)
}) else { return }
```

NSTouch

```
var trackingTouchID: AnyObject?

// find the first tracking touch
guard trackingTouchID == nil else { return }
guard let touch = event1.touches(matching: .began, in: self).first else { return }
trackingTouchID = touch.identity



// find successive touches
guard let trackingTouchID = trackingTouchID else { return }
guard let touch = event2.touches(matching: .moved, in: self).first(where: {
    $0.identity.isEqual(trackingTouchID)
}) else { return }
```

NSTouch

```
var trackingTouchID: AnyObject?

// find the first tracking touch
guard trackingTouchID == nil else { return }
guard let touch = event1.touches(matching: .began, in: self).first else { return }
trackingTouchID = touch.identity


// find successive touches
guard let trackingTouchID = trackingTouchID else { return }
guard let touch = event2.touches(matching: .moved, in: self).first(where: {
    $0.identity.isEqual(trackingTouchID)
}) else { return }
```

NSTouch

```
var trackingTouchID: AnyObject?

// find the first tracking touch
guard trackingTouchID == nil else { return }
guard let touch = event1.touches(matching: .began, in: self).first else { return }
trackingTouchID = touch.identity



// find successive touches
guard let trackingTouchID = trackingTouchID else { return }
guard let touch = event2.touches(matching: .moved, in: self).first(where: {
    $0.identity.isEqual(trackingTouchID)
}) else { return }
```

NSTouch

```
var trackingTouchID: AnyObject?

// find the first tracking touch
guard trackingTouchID == nil else { return }
guard let touch = event1.touches(matching: .began, in: self).first else { return }
trackingTouchID = touch.identity



// find successive touches
guard let trackingTouchID = trackingTouchID else { return }
guard let touch = event2.touches(matching: .moved, in: self).first(where: {
    $0.identity.isEqual(trackingTouchID)
}) else { return }
```

NSResponder and NSView

```
extension NSResponder {
    func touchesBegan(with event: NSEvent)
    func touchesMoved(with event: NSEvent)
    func touchesEnded(with event: NSEvent)
    func touchesCancelled(with event: NSEvent)
}
```

```
extension NSView {
    var allowedTouchTypes: NSTouch.TouchTypeMask
}
```

Automatic opt-in with 10.12 linking

NSResponder and NSView

```
extension NSResponder {
    func touchesBegan(with event: NSEvent)
    func touchesMoved(with event: NSEvent)
    func touchesEnded(with event: NSEvent)
    func touchesCancelled(with event: NSEvent)
}
```

```
extension NSView {
    var allowedTouchTypes: NSTouch.TouchTypeMask
}
```

Automatic opt-in with 10.12 linking

NSResponder and NSView

```
func touchesCancelled(with event: NSEvent)
```

Very important in Touch Bar

Changes to Touch Bar content

- Focus changes on screen
- State changes

Accidental touches

NSGestureRecognizer

```
extension NSGestureRecognizer {
    func touchesBegan(with event: NSEvent)
    func touchesMoved(with event: NSEvent)
    func touchesEnded(with event: NSEvent)
    func touchesCancelled(with event: NSEvent)

    var allowedTouchTypes: NSTouch.TouchTypeMask
}
```

Requires opt-in

Only direct touches supported

NSGestureRecognizer

AppKit	Touch Bar Support	UIKit
NSPressGestureRecognizer	✓	UIPressGestureRecognizer
NSPanGestureRecognizer	✓	UIPanGestureRecognizer
NSClickGestureRecognizer	✓	UITapGestureRecognizer
NSMagnificationGestureRecognizer	✓	UIMagnificationGestureRecognizer

NSGestureRecognizer

AppKit	Touch Bar Support	UIKit
NSPressGestureRecognizer	✓	UIPressGestureRecognizer

```
recognizer.allowedTouchTypes = .direct
```

NSClickGestureRecognizer	✓	UITapGestureRecognizer
NSMagnificationGestureRecognizer	✓	UIMagnificationGestureRecognizer

Press-and-Hold Popovers

`NSPopoverTouchBarItem.pressAndHoldTouchBar`

Simple list-like UI

`NSResponder.touchesBegan(with:)` on enter

`NSResponder.touchesCancelled(with:)` on exit

No gesture recognizers

Press-and-Hold Popovers

`NSPopoverTouchBarItem.pressAndHoldTouchBar`

Simple list-like UI

`NSResponder.touchesBegan(with:)` on enter

`NSResponder.touchesCancelled(with:)` on exit

No gesture recognizers

Press-and-Hold Popovers

`NSPopoverTouchBarItem.pressAndHoldTouchBar`

Simple list-like UI

`NSResponder.touchesBegan(with:)` on enter

`NSResponder.touchesCancelled(with:)` on exit

No gesture recognizers

Event Modality

Touch Bar is a responsive input device

No modal interactions

- Event tracking loops
- Drag and Drop
- Touch Bar interaction

Multiple Input Devices

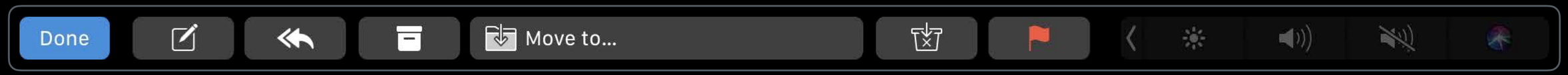
Multi-Touch on Touch Bar

Keyboard Input

Cursor Input

Multiple Input Devices

Multi-Touch



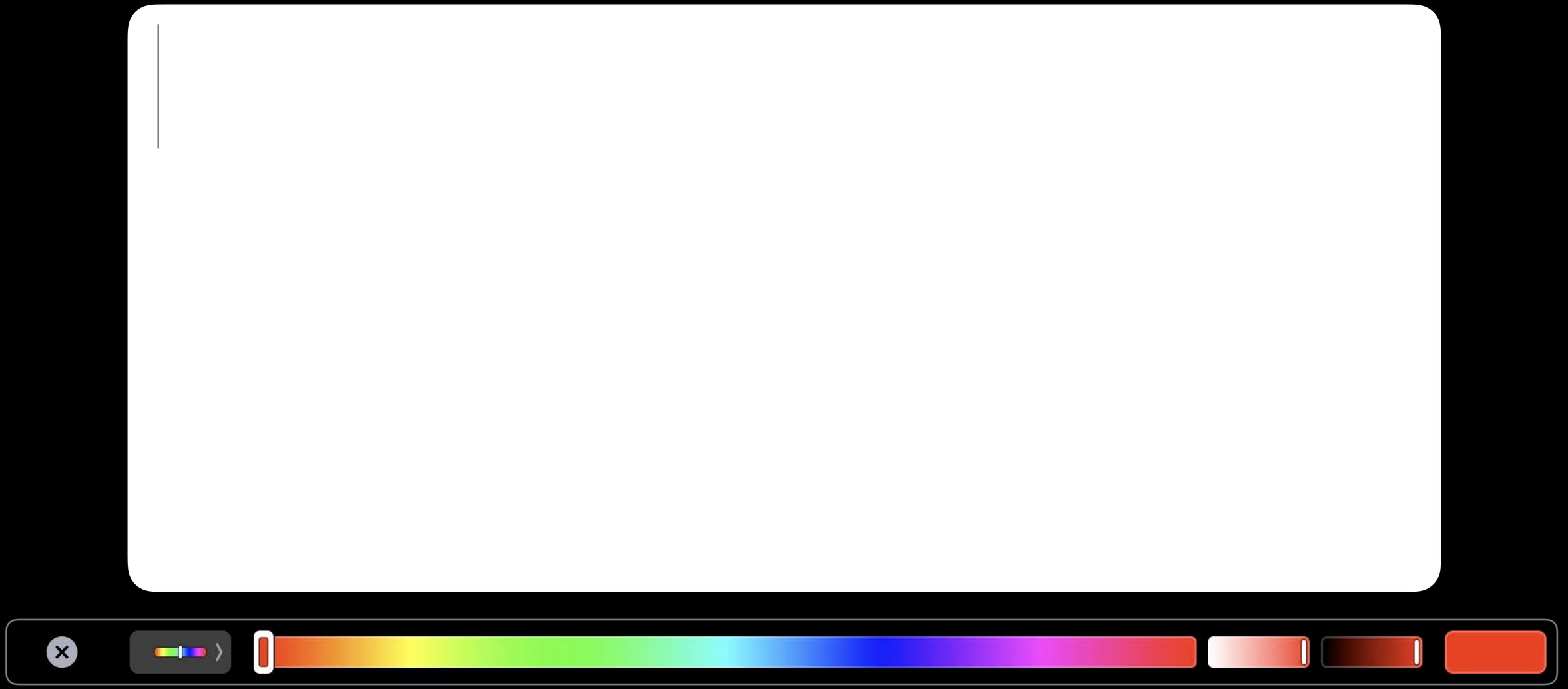
Multiple Input Devices

Multi-Touch

Multiple Input Devices

Multi-Touch

Multiple Input Devices

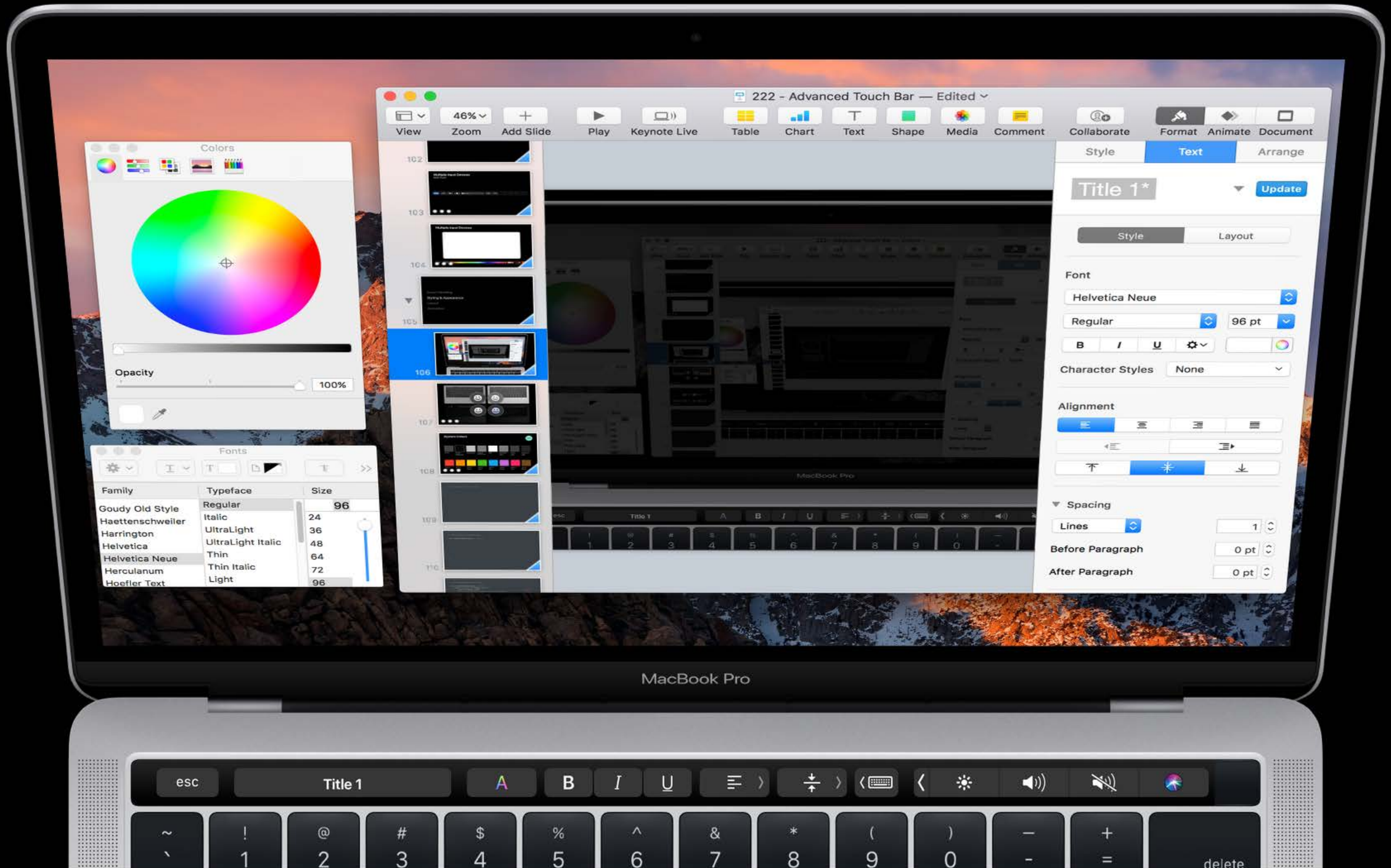


Event Handling

Styling and Appearance

Layout

Animation


222 - Advanced Touch Bar — Edited
View
Zoom
Add Slide
Play
Keynote Live
Table
Chart
Text
Shape
Media
Comment
Collaborate
Format
Animate
Document
Colors
Opacity
100%
Fonts
Family
Typeface
Size
Goudy Old Style
Haettenschweiler
Harrington
Helvetica
Helvetica Neue
Herculanum
Hoefler Text
Regular
Italic
UltraLight
UltraLight Italic
Thin
Thin Italic
Light
96
24
36
48
64
72
96
Style
Text
Arrange
Title 1*
Update
Style
Layout
Font
Helvetica Neue
Regular
96 pt
Character Styles
None
Alignment
Spacing
Lines
1
Before Paragraph
0 pt
After Paragraph
0 pt
MacBook Pro
esc
Title 1
delete

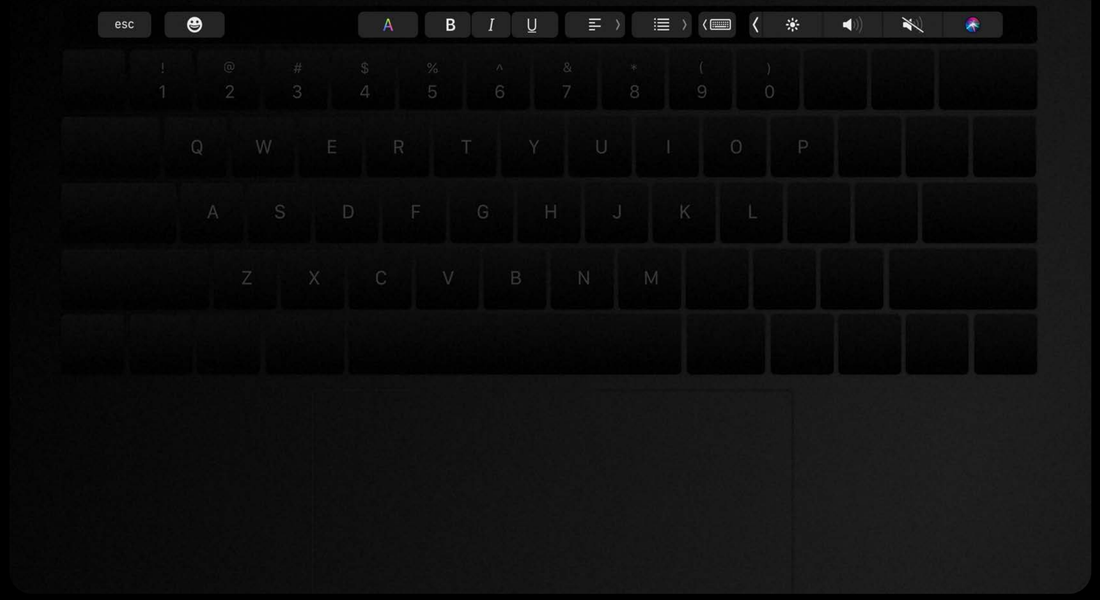

esc
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M


esc
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M


esc
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M

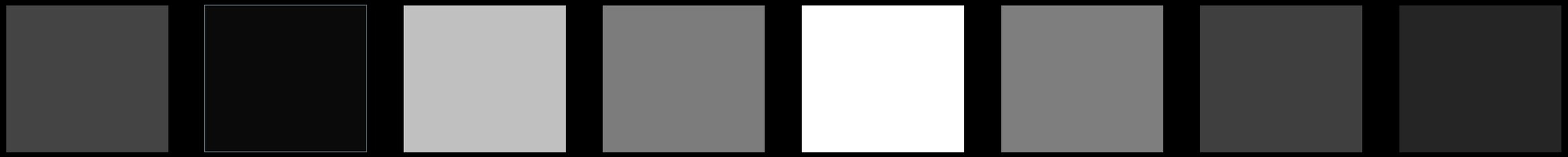

esc
A
B I U
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M


esc
A
B I U
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M


esc
A
B I U
! 1
@ 2
# 3
$ 4
% 5
^ 6
& 7
* 8
(9
) 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M


esc
B I U
! @ # $ % ^ & * ( )
1 2 3 4 5 6 7 8 9 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
esc
B I U
! @ # $ % ^ & * ( )
1 2 3 4 5 6 7 8 9 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
esc
B I U
! @ # $ % ^ & * ( )
1 2 3 4 5 6 7 8 9 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
esc
B I U
! @ # $ % ^ & * ( )
1 2 3 4 5 6 7 8 9 0
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M

System Colors



Control

Control Background

Control Text

Disabled Control Text

Label

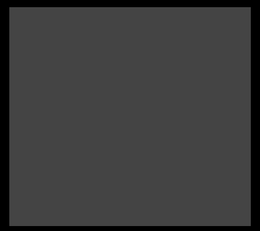
Secondary Label

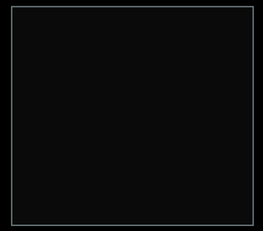
Tertiary Label

Quaternary Label

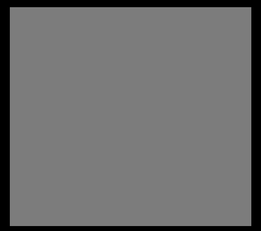
System Colors

NEW

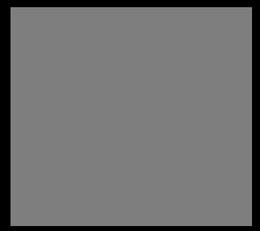

Control

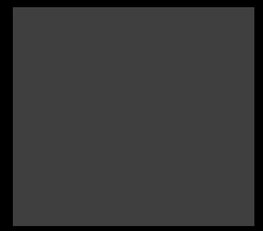

Control Background

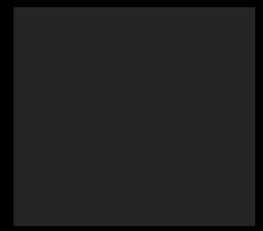

Control Text


Disabled Control Text


Label


Secondary Label


Tertiary Label


Quaternary Label

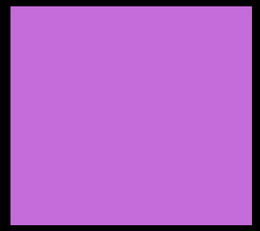

system Red

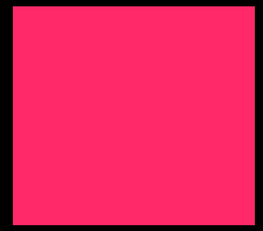

system Orange


system Yellow


system Green


system Blue


system Purple


system Pink


system Brown

```
// Custom Drawing with System Colors
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {









}
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {
    required override init(frame: NSRect) {
        super.init(frame: frame)
        aLayer.backgroundColor = NSColor.controlColor.cgColor
    }

    var bezelColor = NSColor.controlColor {
        didSet {
            aLayer.backgroundColor = bezelColor.cgColor
        }
    }
}
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {









}
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {
    override func updateLayer() {
        super.updateLayer()
        aLayer.backgroundColor = NSColor.controlColor.cgColor
    }




}
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {
    override func updateLayer() {
        super.updateLayer()
        aLayer.backgroundColor = bezelColor.cgColor
    }

    var bezelColor: NSColor = .controlColor {
        didSet {
            needsDisplay = true
        }
    }
}
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {
    override func updateLayer() {
        super.updateLayer()
        aLayer.backgroundColor = bezelColor.cgColor
    }

    var bezelColor: NSColor = .controlColor {
        didSet {
            needsDisplay = true
        }
    }
}
```

```
// Custom Drawing with System Colors

class LayerBasedDrawingView : NSView {
    override func updateLayer() {
        super.updateLayer()
        aLayer.backgroundColor = bezelColor.cgColor
    }

    var bezelColor: NSColor = .controlColor {
        didSet {
            needsDisplay = true
        }
    }
}
```

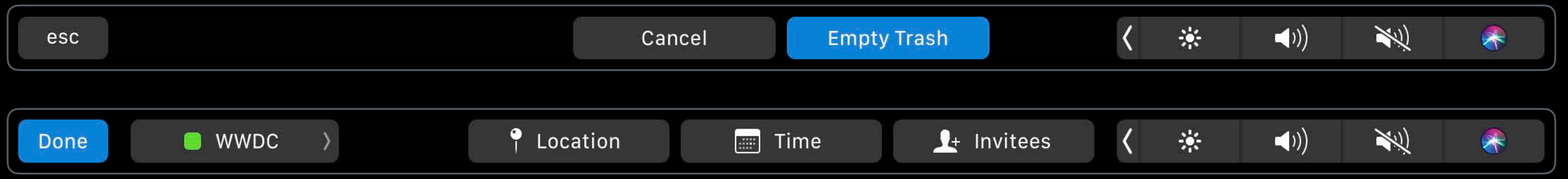
```
// Custom Drawing with System Colors

class DrawRectBasedView : NSView {
    override func draw(_ rect: NSRect) {
        bezelColor.set()
        rect.fill()
    }

    var bezelColor: NSColor = .controlColor {
        didSet {
            needsDisplay = true
        }
    }
}
```

```
// Custom Drawing with System Colors

class DrawRectBasedView : NSView {
    override func draw(_ rect: NSRect) {
        bezelColor.set()
        rect.fill()
    }

    var bezelColor: NSColor = .controlColor {
        didSet {
            needsDisplay = true
        }
    }
}
```

Control Tinting

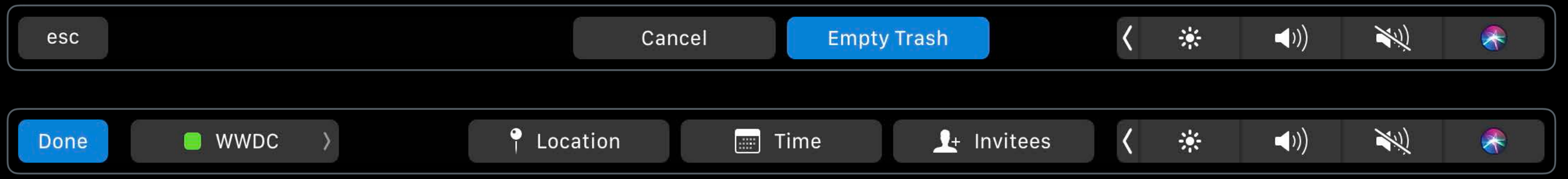

```
button.bezelColor = .systemYellow
segmentedControl.selectedSegmentBezelColor = .systemPurple
slider.trackFillColor = .systemGreen
```

Default Button



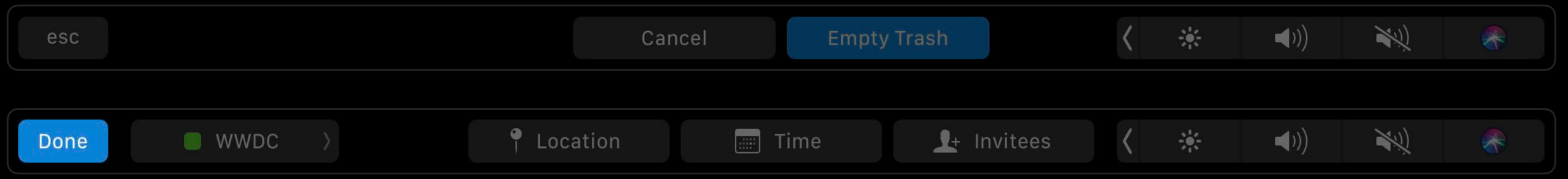

```
button.bezelColor = .systemBlue
```

Default Button




```
button.keyEquivalent = "\r"
```

Default Button




```
button.keyEquivalent = "\r"
```

Esc Key Replacement

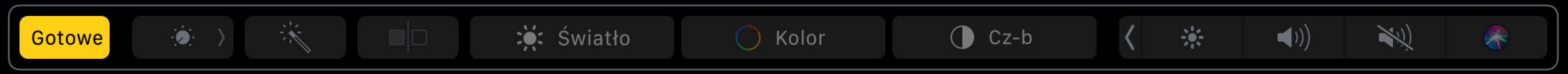

```
touchBar.escapeKeyReplacementItemIdentifier = doneButtonItemIdentifier
```

Modal contexts

Undoable action

Automatic metrics

Esc Key Replacement



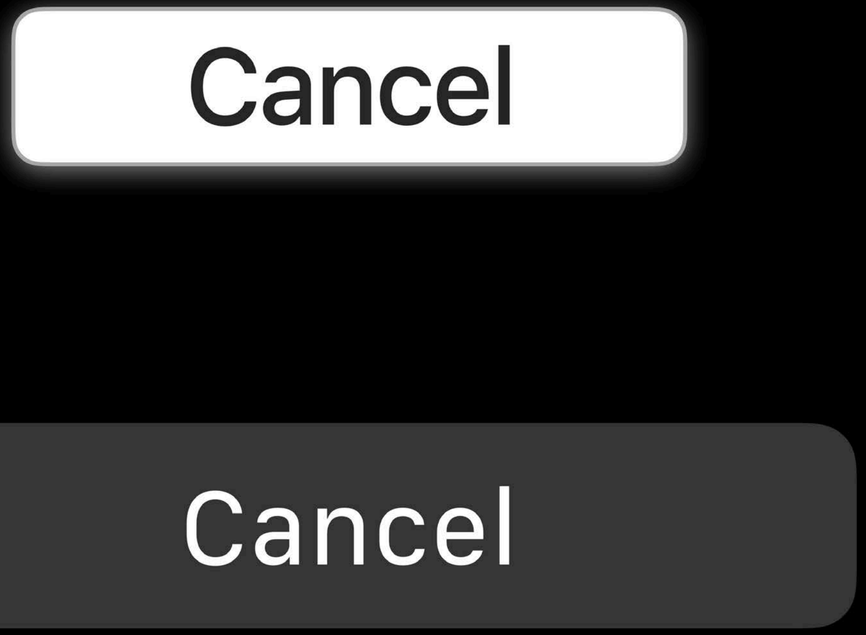

```
touchBar.escapeKeyReplacementItemIdentifier = doneButtonItemIdentifier
```

Modal contexts

Undoable action

Automatic metrics

Font



Font

```
let font = NSFont.systemFont(ofSize: 0)
let boldFont = NSFont.systemFont(ofSize: 0, weight: .bold)
```

Font

```
let font = NSFont.systemFont(ofSize: 0)
let boldFont = NSFont.systemFont(ofSize: 0, weight: .bold)
```

Font

```
let font = NSFont.systemFont(ofSize: 0)
let boldFont = NSFont.systemFont(ofSize: 0, weight: .bold)
```

Font

```
let font = NSFont.systemFont(ofSize: 0)
let boldFont = NSFont.systemFont(ofSize: 0, weight: .bold)
```


```
let font = NSFont.monospacedDigitSystemFont(ofSize: 0, weight: .regular)
```

Font

```
let font = NSFont.systemFont(ofSize: 0)
let boldFont = NSFont.systemFont(ofSize: 0, weight: .bold)
```


```
let font = NSFont.monospacedDigitSystemFont(ofSize: 0, weight: .regular)
```

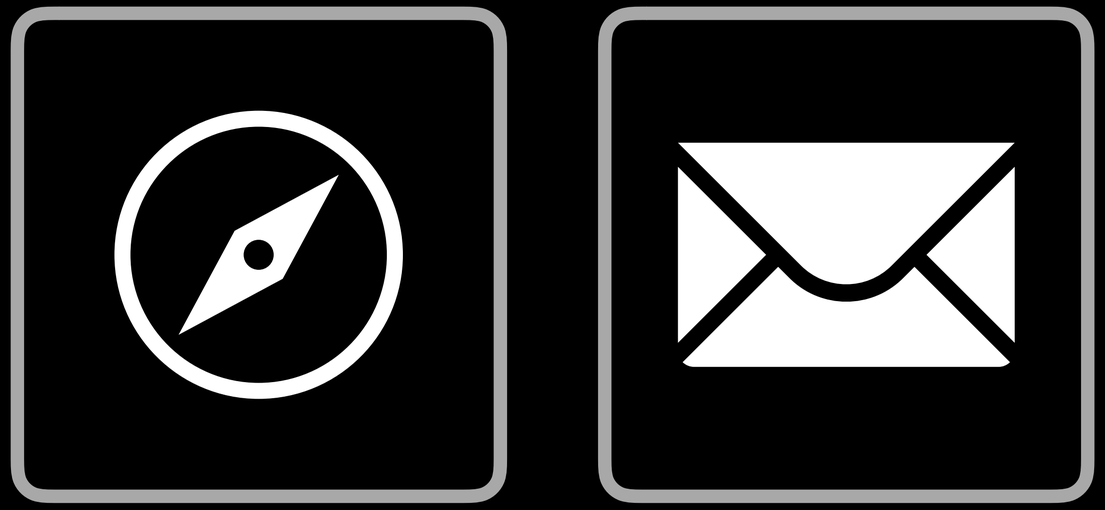
Introducing the New System Fonts	WWDC 2015
Typography and Font	WWDC 2016

Images

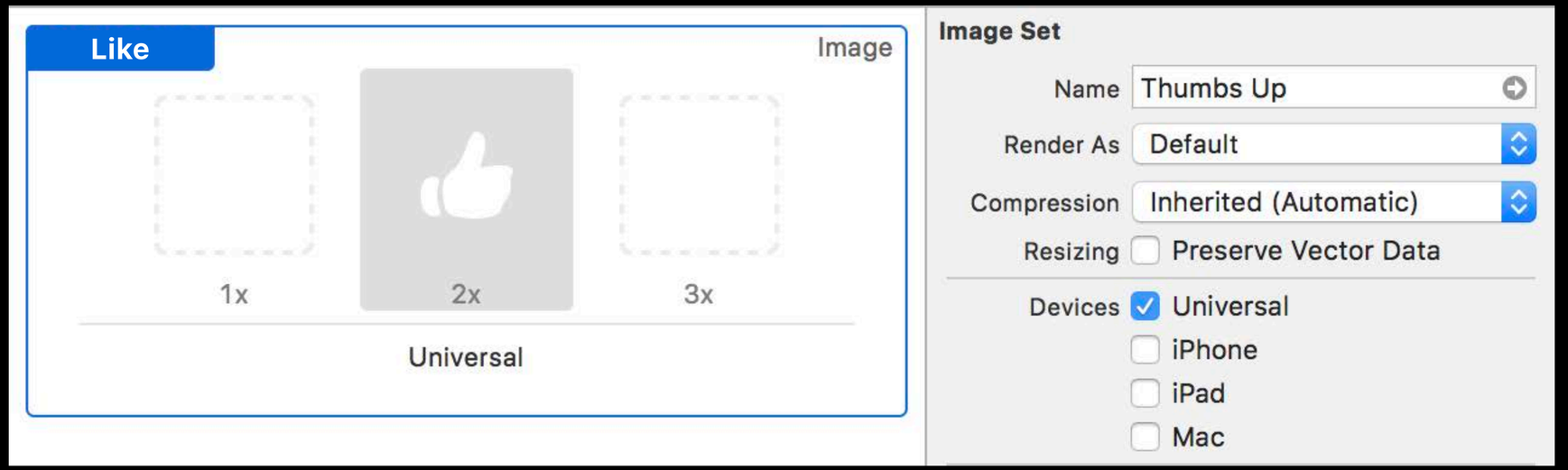
2x assets

Designed for Touch Bar

Template rendering

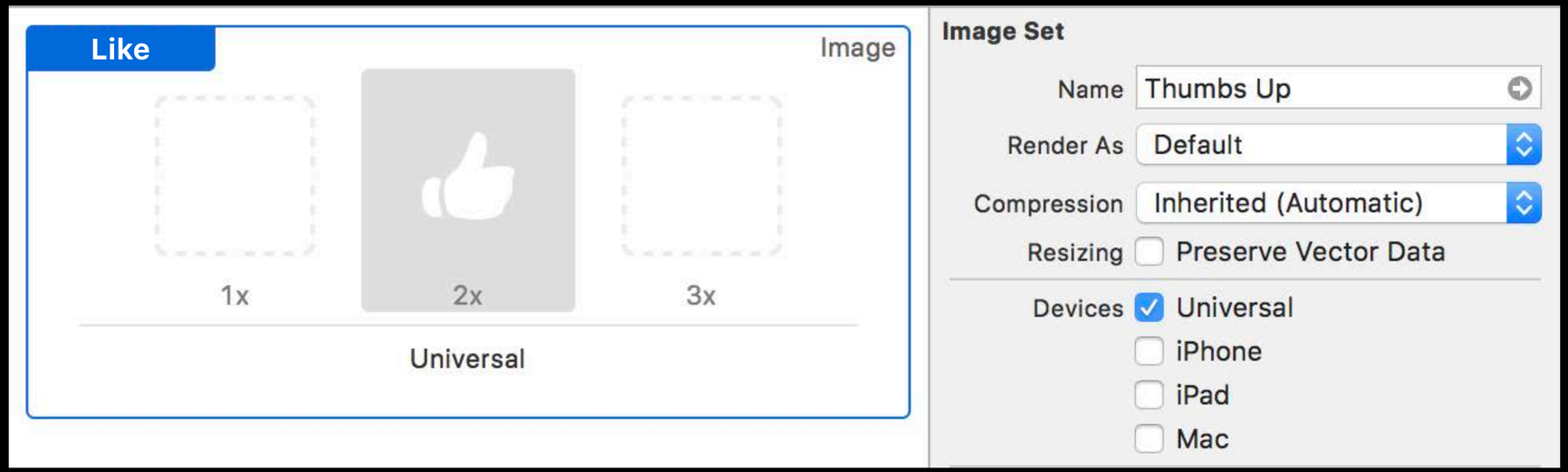


Images


Like
Image
1x
2x
3x
Universal
Image Set
Name
Thumbs Up
Render As
Default
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
Mac

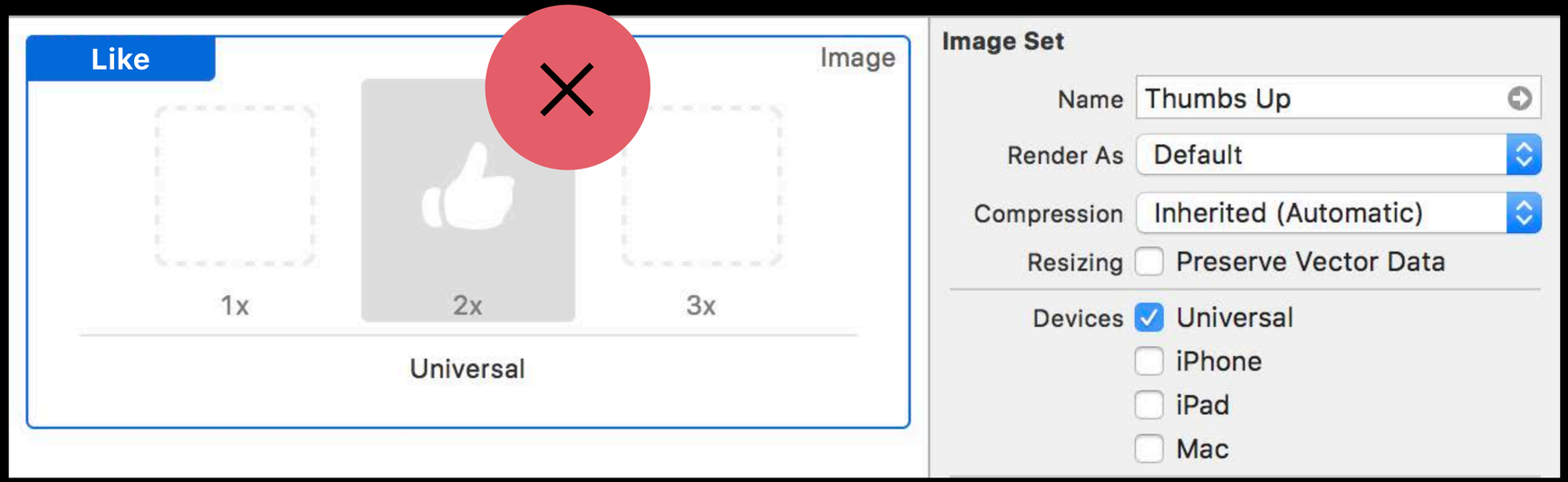

esc

Images


Like
Image
1x
2x
3x
Universal
Image Set
Name
Thumbs Up
Render As
Default
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
Mac

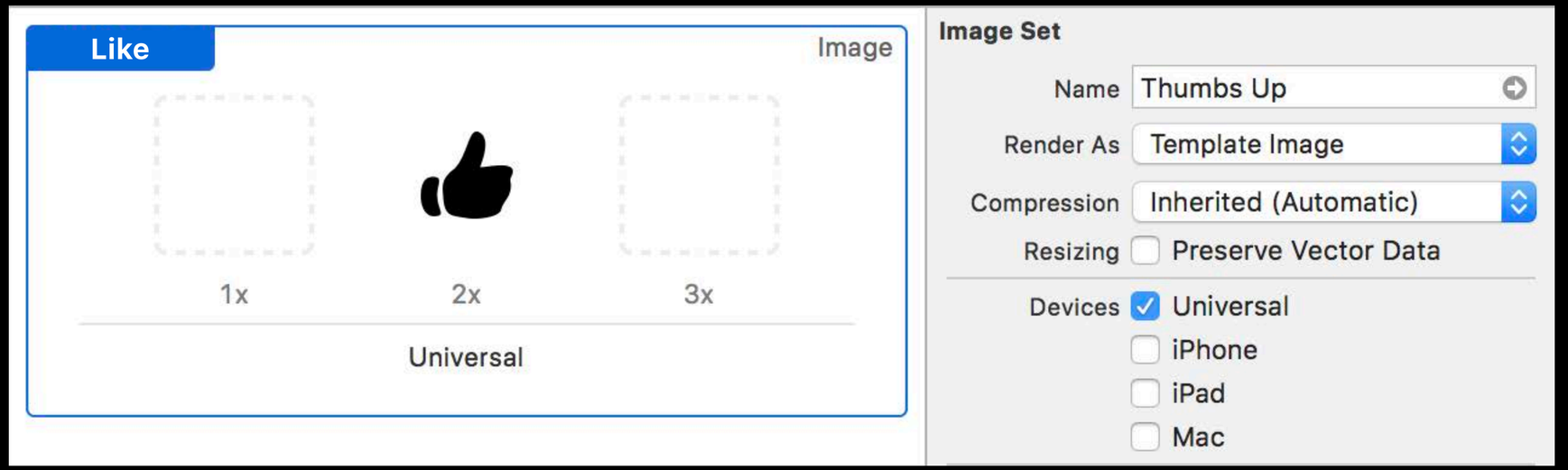

esc

Images


Like
Image
1x
2x
3x
Universal
Image Set
Name
Thumbs Up
Render As
Default
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
Mac

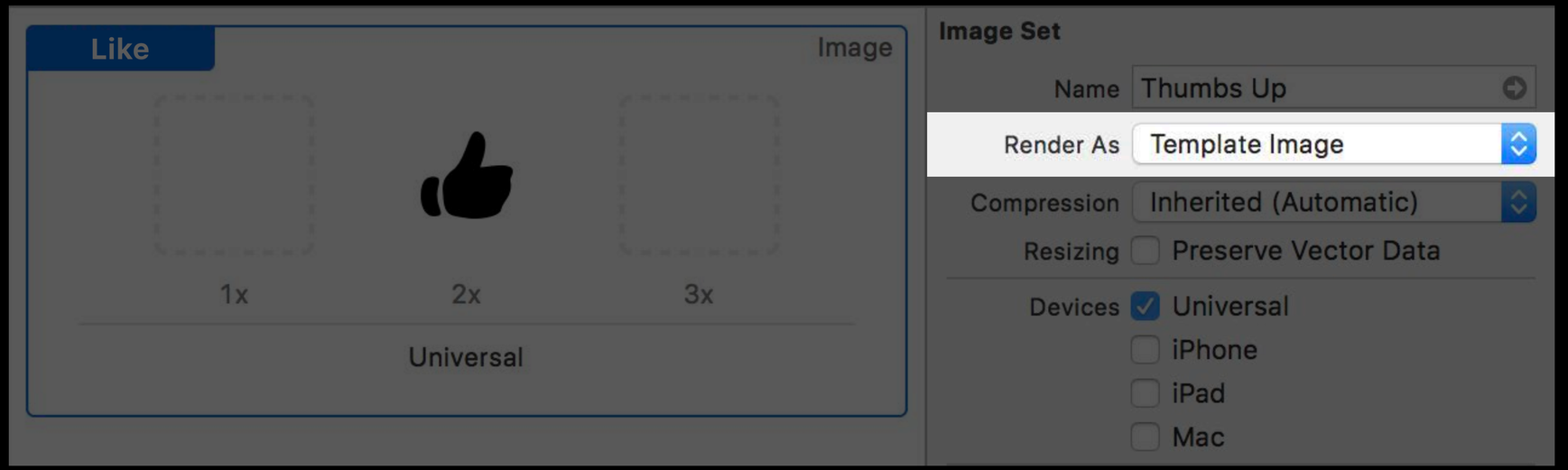

esc

Images


Like
Image
1x
2x
3x
Universal
Image Set
Name
Thumbs Up
Render As
Template Image
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
Mac

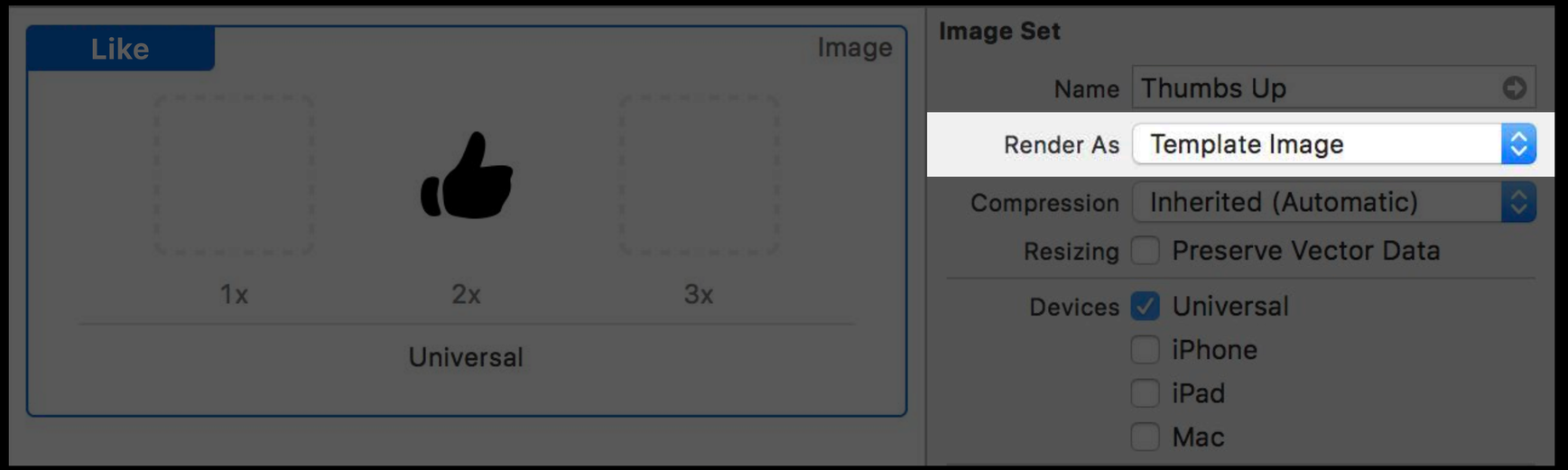

esc

Images


Like
Image
1x
2x
3x
Universal
Image Set
Name
Thumbs Up
Render As
Template Image
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
Mac


esc

Images


Like
Image
1x
2x
3x
Universal
Image Set
Name
Thumbs Up
Render As
Template Image
Compression
Inherited (Automatic)
Resizing
Preserve Vector Data
Devices
Universal
iPhone
iPad
Mac


esc


Add


Add Detail


Alarm


Audio Input


Audio Input Mute


Audio Output Mute


Audio Output Volume High


Audio Output Volume Low


Audio Output Volume Medium


Audio Output Volume Off


Bookmarks


Color Picker Fill


Color Picker Font


Color Picker Stroke


Communication Audio


Communication Video


Compose


Delete


Download


Enter Full Screen


Exit Full Screen


Fast Forward


Folder


Folder Copy To


Folder Move To


Get Info


Go Back


Go Down


Go Forward


Go Up


History


Icon View


List View


Mail


New Folder


New Message


Open In Browser


Pause


Play


Play Pause

Quick Look

Record Start

Record Stop

Refresh

Rewind

Rotate Left

Rotate Right

Search

Share

Sidebar

Skip Ahead

Skip Ahead 15 Seconds

Skip Ahead 30 Seconds

Skip Back

Skip Back 15 Seconds

Skip Back 30 Seconds

Skip To End

Skip To Start

Slideshow

Tag Icon

Text Bold

Text Box

Text Center Align

Text Italic

Text Justified Align

Text Left Align

Text List

Text Right Align

Text Strikethrough

Text Underline

User

User Add

User Group

Volume Down

Volume Up

Event Handling

Styling and Appearance

Layout

Animation

NSTouchBar Item Layout

Standard Item Layout

Flexible Spaces

Principal Item

Item Sizing

Fixed Size Controls


```
override var intrinsicContentSize: NSSize {
    let buttonWidth = // calculate from text + padding
    return NSSize(width: buttonWidth, height: 30)
}
```

Item Sizing

Fixed Size Controls


```
override var intrinsicContentSize: NSSize {
    let buttonWidth = // calculate from text + padding
    return NSSize(width: buttonWidth, height: 30)
}
```

Item Sizing

Fixed Size Controls


```
override var intrinsicContentSize: NSSize {
    let buttonWidth = // calculate from text + padding
    return NSSize(width: buttonWidth, height: 30)
}


invalidateIntrinsicContentSize()
```

Item Sizing

Fixed Size Controls


```
let widthConstraint = widthAnchor.constraint(equalToConstant: 72)




widthConstraint.constant = newWidth
```

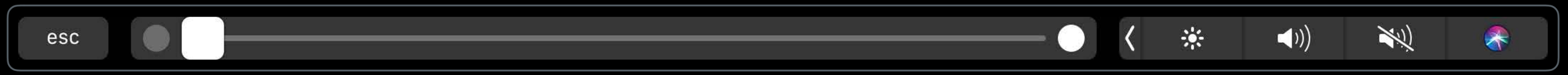
Item Sizing

Fixed Size Controls


```
let widthConstraint = widthAnchor.constraint(equalToConstant: 72)




widthConstraint.constant = newWidth
```

Item Sizing

Flexible Controls




```
override var intrinsicContentSize: NSSize {
    return NSSize(width: NSView.noIntrinsicMetric, height: 72)
}
```

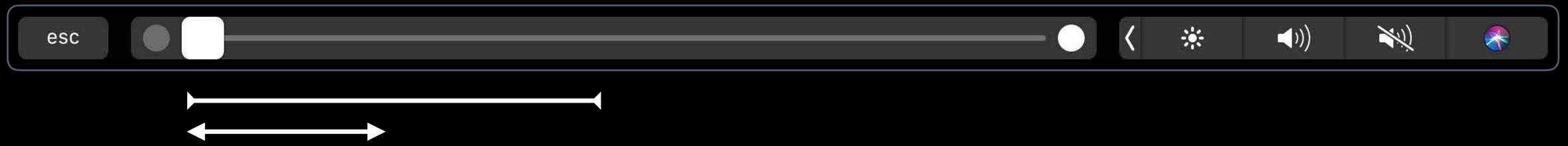
Item Sizing

Flexible Controls


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: NSView.noIntrinsicMetric, height: 72)
}
```

Item Sizing

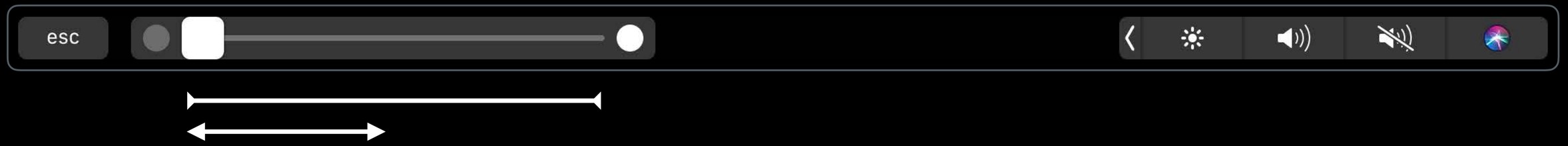
Flexible Controls




```
override var intrinsicContentSize: NSSize {
    return NSSize(width: NSView.noIntrinsicMetric, height: 72)
}

sliderItem.slider.widthAnchor.constraint(lessThanOrEqualToConstant: 300).isActive = true
sliderItem.slider.widthAnchor.constraint(greaterThanOrEqualToConstant: 150).isActive = true
```

Item Sizing

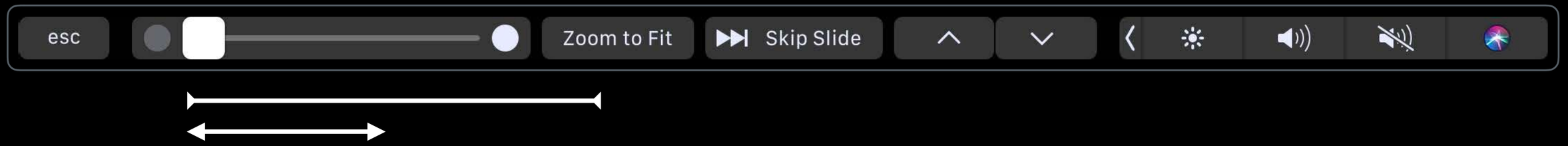
Flexible Controls




```
override var intrinsicContentSize: NSSize {
    return NSSize(width: NSView.noIntrinsicMetric, height: 72)
}

sliderItem.slider.widthAnchor.constraint(lessThanOrEqualToConstant: 300).isActive = true
sliderItem.slider.widthAnchor.constraint(greaterThanOrEqualToConstant: 150).isActive = true
```

Item Sizing

Flexible Controls




```
override var intrinsicContentSize: NSSize {
    return NSSize(width: NSView.noIntrinsicMetric, height: 72)
}

sliderItem.slider.widthAnchor.constraint(lessThanOrEqualToConstant: 300).isActive = true
sliderItem.slider.widthAnchor.constraint(greaterThanOrEqualToConstant: 150).isActive = true
```

Equal Sizing

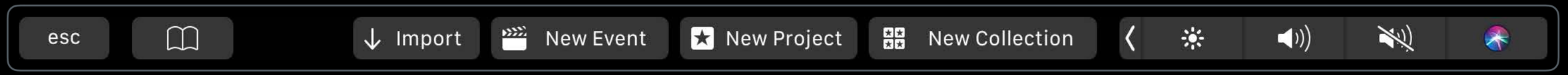
NEW


```
extension NSGroupTouchBarItem {
    var prefersEqualWidths: Bool
}
```

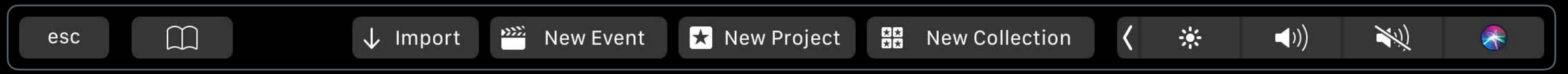
User customization

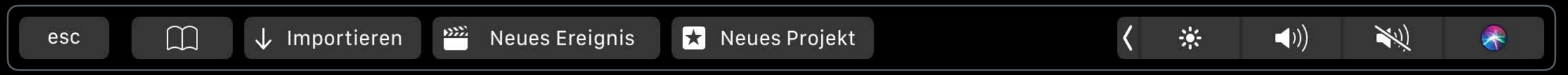
Localization

Visibility Priority

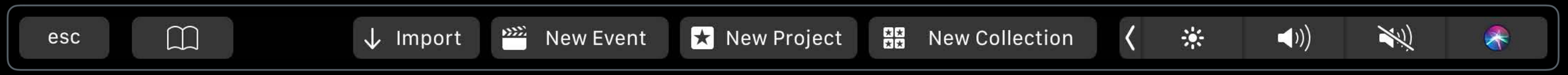


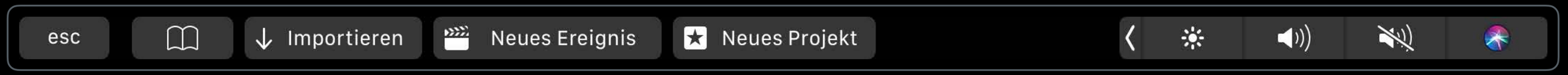
Visibility Priority





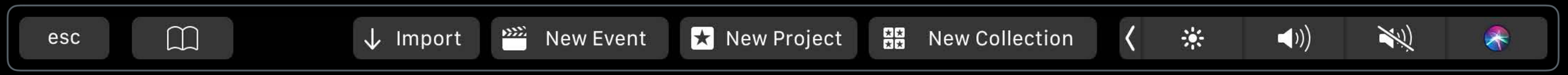
Visibility Priority

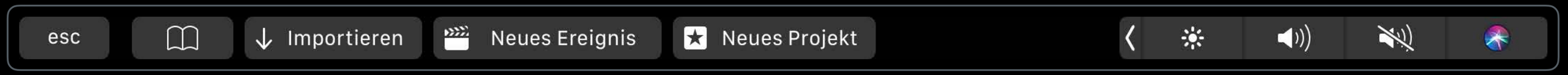




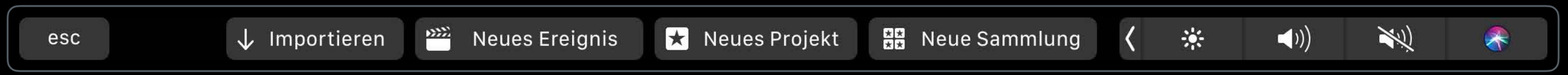

```
bookmarkItem.visibilityPriority = .low
```

Visibility Priority



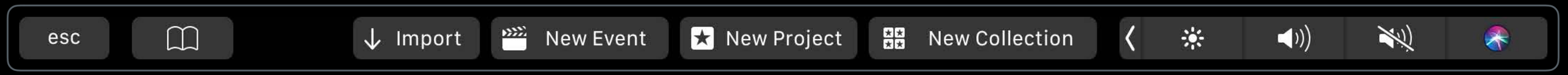



```
bookmarkItem.visibilityPriority = .low
```



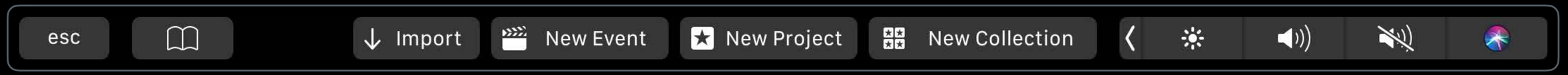
Compression Behavior

NEW



Compression Behavior

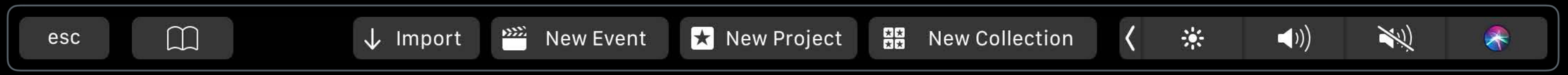
NEW



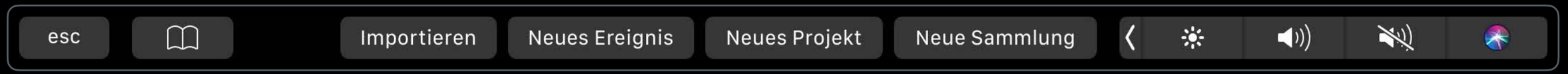

```
centerGroup.prioritizedCompressionOptions = [.hideImages]
```

Compression Behavior

NEW

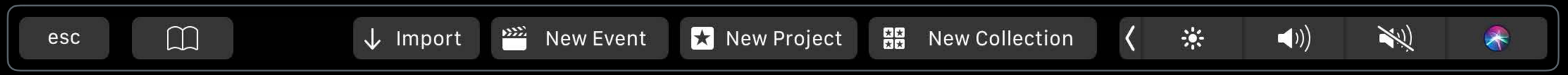



```
centerGroup.prioritizedCompressionOptions = [.hideImages]
```

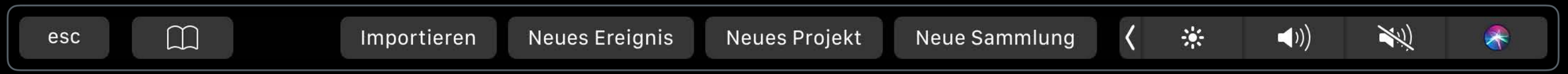


Compression Behavior

NEW



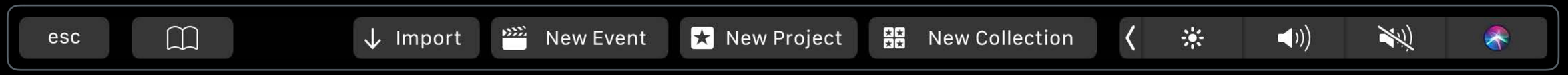

```
centerGroup.prioritizedCompressionOptions = [.hideImages]
```



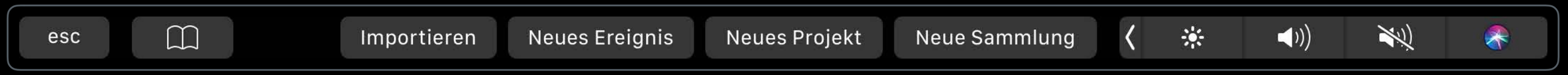

```
centerGroup.prioritizedCompressionOptions = [.hideText]
```

Compression Behavior

NEW




```
centerGroup.prioritizedCompressionOptions = [.hideImages]
```




```
centerGroup.prioritizedCompressionOptions = [.hideText]
```

Container Views in Touch Bar

NSStackView

NSScrollView

NSScrubber

NSCollectionView

Container Views in Touch Bar

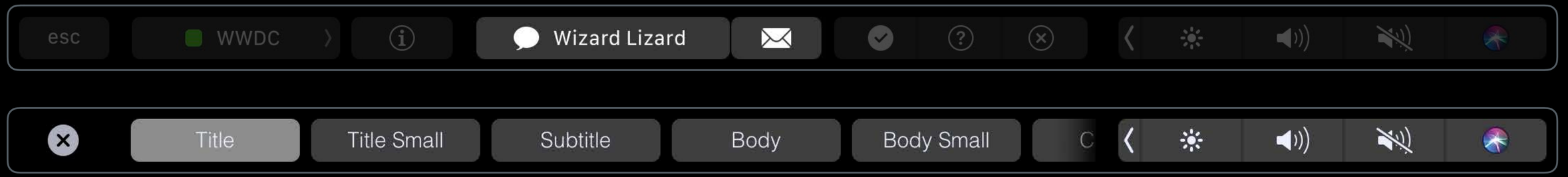
NSStackView

Container Views in Touch Bar

NSStackView

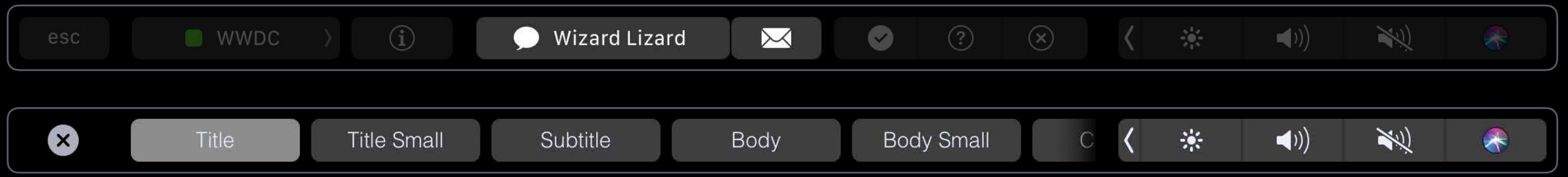
Container Views in Touch Bar

NSStackView



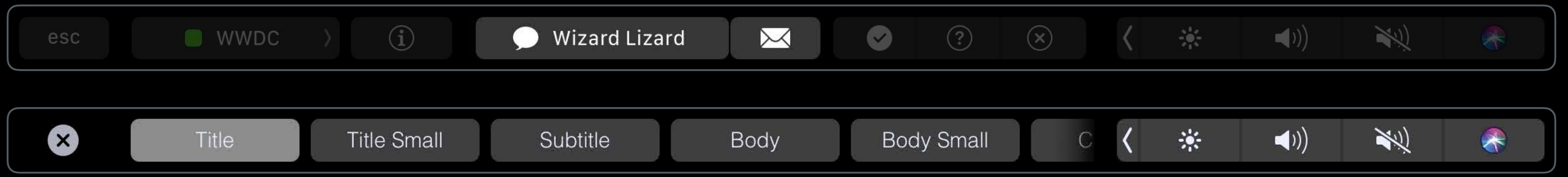
Container Views in Touch Bar

NSStackView



Container Views in Touch Bar

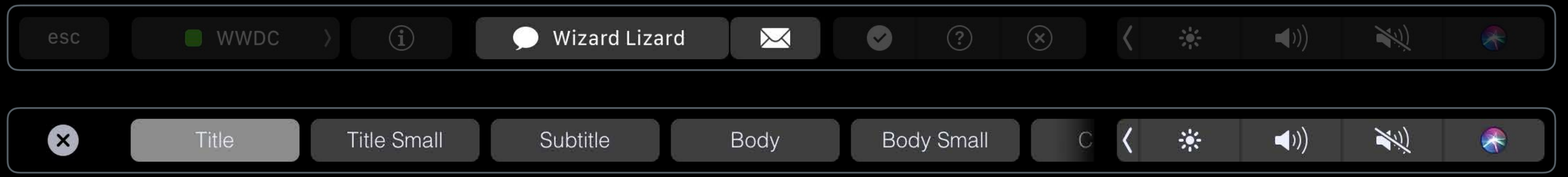
NSStackView



NSScrubber

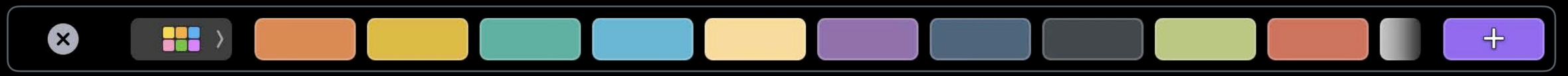
Container Views in Touch Bar

NSStackView



NSScrubber

NSCollectionView



Event Handling

Styling and Appearance

Layout

Animation

Animation

Input device, not display

Don’t distract

Relative to user interaction

Interruptible

Animation

Input device, not display

Don’t distract

Relative to user interaction

Interruptible

Item Size Animation

Constant Size


```
let widthConstraint = widthAnchor.constraint(equalToConstant: 72)

// on tap recognition
widthConstraint.constant = newWidth
```

Item Size Animation

Constant Size


```
let widthConstraint = widthAnchor.constraint(equalToConstant: 72)

// on tap recognition
widthConstraint.constant = newWidth
```

Item Size Animation

Constant Size


```
let widthConstraint = widthAnchor.constraint(equalToConstant: 72)

// on tap recognition
widthConstraint.animator().constant = newWidth
```

Item Size Animation

Constant Size


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? 152 : 72, height: 30)
}



isActive = !isActive
invalidateIntrinsicContentSize()
```

Item Size Animation

Constant Size


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? 152 : 72, height: 30)
}



isActive = !isActive
invalidateIntrinsicContentSize()
```

Item Size Animation

Constant Size


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? 152 : 72, height: 30)
}

NSAnimationContext.runAnimationGroup({ context in
    isActive = !isActive
    invalidateIntrinsicContentSize()
    context.allowsImplicitAnimation = true
    window?.layoutIfNeeded()
})
```

Item Size Animation

Constant Size


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? 152 : 72, height: 30)
}

NSAnimationContext.runAnimationGroup({ context in
    isActive = !isActive
    invalidateIntrinsicContentSize()
    context.allowsImplicitAnimation = true
    window?.layoutIfNeeded()
})
```

Item Size Animation

Flexibility


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? NSView.noIntrinsicMetric : 72, height: 30)
}

NSAnimationContext.runAnimationGroup({ context in
    isActive = !isActive
    invalidateIntrinsicContentSize()
    context.allowsImplicitAnimation = true
    window?.layoutIfNeeded()
})
```

Item Size Animation

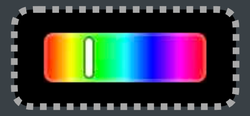
Flexibility


```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? NSView.noIntrinsicMetric : 72, height: 30)
}

NSAnimationContext.runAnimationGroup({ context in
    isActive = !isActive
    invalidateIntrinsicContentSize()
    context.allowsImplicitAnimation = true
    window?.layoutIfNeeded()
})
```

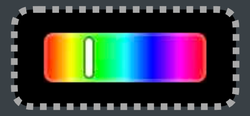
Item Size Animation

Flexibility

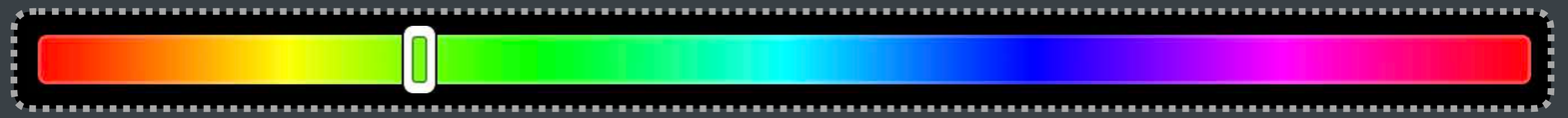

```
override var intrinsicContentSize: NSSize {
    return NSSize(width: isActive ? NSView.noIntrinsicMetric : 72, height: 30)
}

NSAnimationContext.runAnimationGroup({ context in
    isActive = !isActive
    invalidateIntrinsicContentSize()
    context.allowsImplicitAnimation = true
    window?.layoutIfNeeded()
})
```



```
class AccordionView : NSView {
    var isActive: Bool {
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    override var intrinsicContentSize: NSSize {
        return NSSize(width: isActive ? NSView.noIntrinsicMetric : 72, height: 30)
    }

    // drawing, slider event handling ...
}
```



```
class AccordionView : NSView {
    var isActive: Bool    false
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    override var intrinsicContentSize: NSSize {
        return NSSize(width: isActive ? NSView.noIntrinsicMetric : 72, height: 30)
    }

    // drawing, slider event handling ...
}
```



```
class AccordionView : NSView {
    var isActive: Bool      true
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    override var intrinsicContentSize: NSSize {
        return NSSize(width: isActive ? NSView.noIntrinsicMetric : 72, height: 30)
    }

    // drawing, slider event handling ...
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

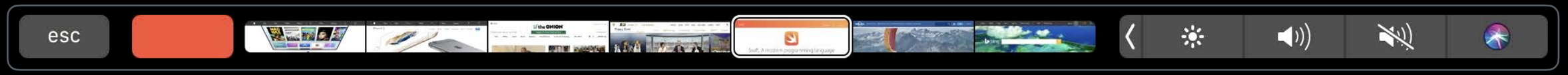
```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

```
var activeAccordionView: AccordionView

func handlePress(_ recognizer: NSPressGestureRecognizer) {
    let oldActiveAccordionView = activeAccordionView
    let newActiveAccordionView = recognizer.view as! AccordionView

    NSAnimationContext.runAnimationGroup({ context in
        oldActiveAccordionView.isActive = false
        newActiveAccordionView.isActive = true
        context.allowsImplicitAnimation = true
        activeAccordionView.window?.layoutIfNeeded()
    })

    activeAccordionView = newActiveAccordionView
}
```

Within-Item Animation

Layout



Constraints or layout() override

No extra work!

Within-Item Animation

Button Content

NEW


```
button1.title = "Rotate"
button1.image = NSImage(named: .touchBarRotateRightTemplate)
button2.bezelColor = .systemRed
button3.imagePosition = .imageLeading
```

Within-Item Animation

Button Content

NEW


```
button1.title = "Rotate"
button1.image = NSImage(named: .touchBarRotateRightTemplate)
button2.bezelColor = .systemRed
button3.imagePosition = .imageLeading
```

Within-Item Animation

Button Content

NEW


```
button1.animator().title = "Rotate"
button1.animator().image = NSImage(named: .touchBarRotateRightTemplate)
button2.animator().bezelColor = .systemRed
button3.animator().imagePosition = .imageLeading
```

Within-Item Animation

Button Content

NEW


```
button1.animator().title = "Rotate"
button1.animator().image = NSImage(named: .touchBarRotateRightTemplate)
button2.animator().bezelColor = .systemRed
button3.animator().imagePosition = .imageLeading
```

Within-Item Animation

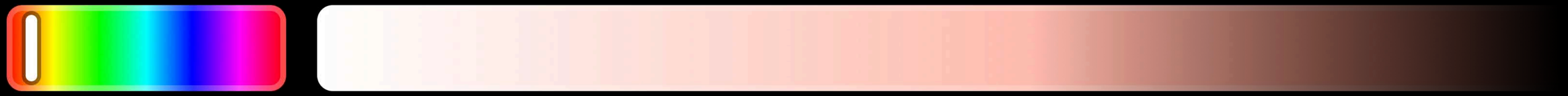
Button Content

NEW


```
button1.animator().title = "Rotate"
button1.animator().image = NSImage(named: .touchBarRotateRightTemplate)
button2.animator().bezelColor = .systemRed
button3.animator().imagePosition = .imageLeading

NSAnimationContext.runAnimationGroup({ context in
    button2.animator().imagePosition = .imageLeading
    context.allowsImplicitAnimation = true
    button2.window?.layoutIfNeeded()
})
```

Within-Item Animation

Custom



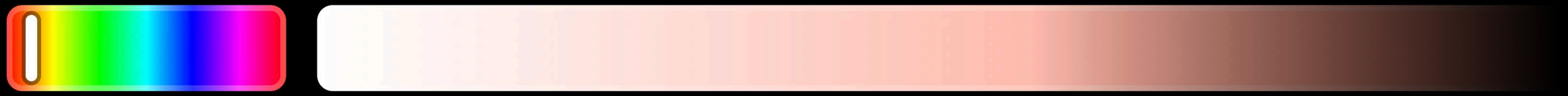
Animator proxy

CAAnimation

layout() override and allowsImplicitAnimation

Within-Item Animation

Custom



Animator proxy

CAAnimation

layout() override and allowsImplicitAnimation

Summary

Customize text view bar content

Use and customize NSScrubber

Create unique Touch Bar interactions

More Information

https://developer.apple.com/wwdc17/222

Related Sessions

Touch Bar Fundamentals	Grand Ballroom A	Wednesday 10:00AM
Choosing the Right Cocoa Container View	Grand Ballroom B	Wednesday 3:10PM
Building Visually Rich User Experiences	Hall 2	Thursday 5:10PM
Best Practices for Cocoa Animation		WWDC 2013
Introducing the New System Fonts		WWDC 2015
Typography and Font		WWDC 2016

Labs

Cocoa Touch Bar Lab	Technology Lab C	Thu 9:00AM–11:00AM
Cocoa Lab	Technology Lab B	Fri 1:50PM–3:20PM


WWDC17