#WWDC18

Accessible Drag and Drop

Session 241

Conor Hughes, iOS Accessibility

Agenda

Agenda

1. Drag and Drop refresher

Agenda

1. Drag and Drop refresher
2. Accessible Drag and Drop concepts

Agenda

1. Drag and Drop refresher
2. Accessible Drag and Drop concepts
3. Example—Exposing an ancestor’s drag

Agenda

1. Drag and Drop refresher
2. Accessible Drag and Drop concepts
3. Example—Exposing an ancestor’s drag
4. Example—Exposing multiple drops

Drag and Drop Refresher

Drag and Drop Refresher

Interactions are hosted by views

Drag and Drop Refresher

Interactions are hosted by views

Starting a drag: `UIDragInteraction`

Drag and Drop Refresher

Interactions are hosted by views

Starting a drag: `UIDragInteraction`

Accepting a drop: `UIDropInteraction`

Drag and Drop Refresher

Interactions are hosted by views

Starting a drag: `UIDragInteraction`

Accepting a drop: `UIDropInteraction`

Accessible Drag and Drop Concepts

Accessible Drag and Drop Concepts

Your accessibility element might not host interactions directly

Accessible Drag and Drop Concepts

Your accessibility element might not host interactions directly

- Subviews might host interactions

Accessible Drag and Drop Concepts

Your accessibility element might not host interactions directly

- Subviews might host interactions
- Element might descend from a view that hosts interactions

Accessible Drag and Drop Concepts

Your accessibility element might not host interactions directly

- Subviews might host interactions
- Element might descend from a view that hosts interactions

Solution

Accessible Drag and Drop Concepts

Your accessibility element might not host interactions directly

- Subviews might host interactions
- Element might descend from a view that hosts interactions

Solution

- Specify logical drags and drops to Accessibility

Accessible Drag and Drop Concepts

Accessible Drag and Drop Concepts

`UIAccessibilityDragging:` Logical drag sources and drop points

Accessible Drag and Drop Concepts

`UIAccessibilityDragging:` Logical drag sources and drop points

- Drag sources describe where to start drags

Accessible Drag and Drop Concepts

`UIAccessibilityDragging:` Logical drag sources and drop points

- Drag sources describe where to start drags
- Drop points describe where to drop

Accessible Drag and Drop Concepts

`UIAccessibilityDragging:` Logical drag sources and drop points

- Drag sources describe where to start drags
- Drop points describe where to drop

Users activate drags and drops like custom actions

Accessible Drag and Drop Concepts

`UIAccessibilityDragging:` Logical drag sources and drop points

- Drag sources describe where to start drags
- Drop points describe where to drop

Users activate drags and drops like custom actions

```
extension NSObject {
    @available(iOS 11.0, *)
    open var accessibilityDragSourceDescriptors: [UIAccessibilityLocationDescriptor]?

    @available(iOS 11.0, *)
    open var accessibilityDropPointDescriptors: [UIAccessibilityLocationDescriptor]?
}
```

Accessible Drag and Drop Concepts

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

- Assigned default name

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

- Assigned default name
- Only interactions in an element’s subtree are exposed

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

- Assigned default name
- Only interactions in an element’s subtree are exposed

`UIAccessibilityDragging`

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

- Assigned default name
- Only interactions in an element’s subtree are exposed

`UIAccessibilityDragging`

- Allows exposing exactly the interactions that make sense

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

- Assigned default name
- Only interactions in an element’s subtree are exposed

`UIAccessibilityDragging`

- Allows exposing exactly the interactions that make sense
- Allows specifying a specific name for each

Accessible Drag and Drop Concepts

Drags and drops often exposed automatically

- Assigned default name
- Only interactions in an element’s subtree are exposed

`UIAccessibilityDragging`

- Allows exposing exactly the interactions that make sense
- Allows specifying a specific name for each
- Implement for the best experience

Exposing an Ancestor’s Drag

Example

Exposing an Ancestor’s Drag

Example

Bar graph built with `CALayer`

Exposing an Ancestor’s Drag

Example

Bar graph built with `CALayer`

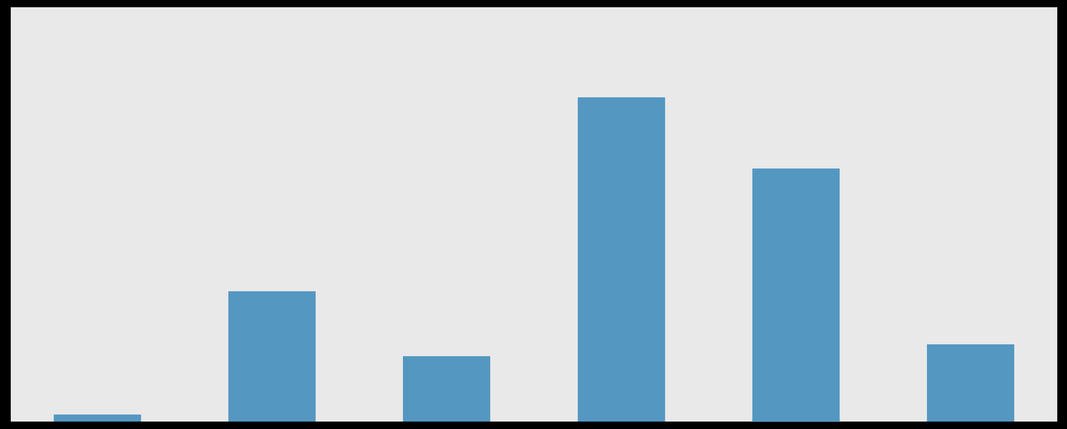
Drag and drop bar data by dragging the bar itself

Exposing an Ancestor's Drag

Example

Bar graph built with `CALayer`

Drag and drop bar data by dragging the bar itself

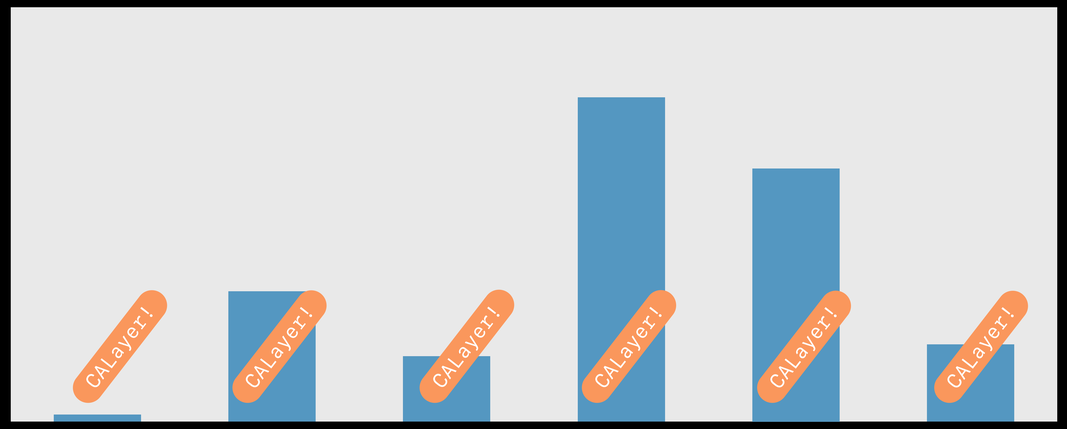


Exposing an Ancestor's Drag

Example

Bar graph built with `CALayer`

Drag and drop bar data by dragging the bar itself



```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    if let index = self.indexOfBar(point: session.location(in: self)) {
        let provider = NSItemProvider(object: "Bar: \(series[index])" as NSString)
        let dragItem = UIDragItem(itemProvider: provider)
        dragItem.localObject = index
        return [dragItem]
    }
    return []
}
```

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    if let index = self.indexOfBar(point: session.location(in: self)) {
        let provider = NSItemProvider(object: "Bar: \(series[index])" as NSString)
        let dragItem = UIDragItem(itemProvider: provider)
        dragItem.localObject = index
        return [dragItem]
    }
    return []
}
```

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    if let index = self.indexOfBar(point: session.location(in: self)) {
        let provider = NSItemProvider(object: "Bar: \(series[index])" as NSString)
        let dragItem = UIDragItem(itemProvider: provider)
        dragItem.localObject = index
        return [dragItem]
    }
    return []
}
```

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    if let index = self.indexOfBar(point: session.location(in: self)) {
        let provider = NSItemProvider(object: "Bar: \(series[index])" as NSString)
        let dragItem = UIDragItem(itemProvider: provider)
        dragItem.localObject = index
        return [dragItem]
    }
    return []
}
```

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    if let index = self.indexOfBar(point: session.location(in: self)) {
        let provider = NSItemProvider(object: "Bar: \(series[index])" as NSString)
        let dragItem = UIDragItem(itemProvider: provider)
        dragItem.localObject = index
        return [dragItem]
    }
    return []
}
```

```
func dragInteraction(_ interaction: UIDragInteraction,
                     itemsForBeginning session: UIDragSession) -> [UIDragItem] {
    if let index = self.indexOfBar(point: session.location(in: self)) {
        let provider = NSItemProvider(object: "Bar: \(series[index])" as NSString)
        let dragItem = UIDragItem(itemProvider: provider)
        dragItem.localObject = index
        return [dragItem]
    }
    return []
}
```

```
func makeAccessibilityElements() {
    self.accessibilityElements = bars.enumerated().map { (index, barLayer) in
        let element = UIAccessibilityElement(accessibilityContainer: self)
        element.accessibilityFrameInContainerSpace = barLayer.frame
        element.accessibilityLabel = seriesLabels[index]
        element.accessibilityValue = "\(series[index])"
        return element
    }
}
```

```
func makeAccessibilityElements() {
    self.accessibilityElements = bars.enumerated().map { (index, barLayer) in
        let element = UIAccessibilityElement(accessibilityContainer: self)
        element.accessibilityFrameInContainerSpace = barLayer.frame
        element.accessibilityLabel = seriesLabels[index]
        element.accessibilityValue = "\(series[index])"
        return element
    }
}
```

UIAccessibilityLocationDescriptor

UIAccessibilityLocationDescriptor

Names and describes where to activate an interaction

UIAccessibilityLocationDescriptor

Names and describes where to activate an interaction

- A point

UIAccessibilityLocationDescriptor

Names and describes where to activate an interaction

- A point
- In a view

UIAccessibilityLocationDescriptor

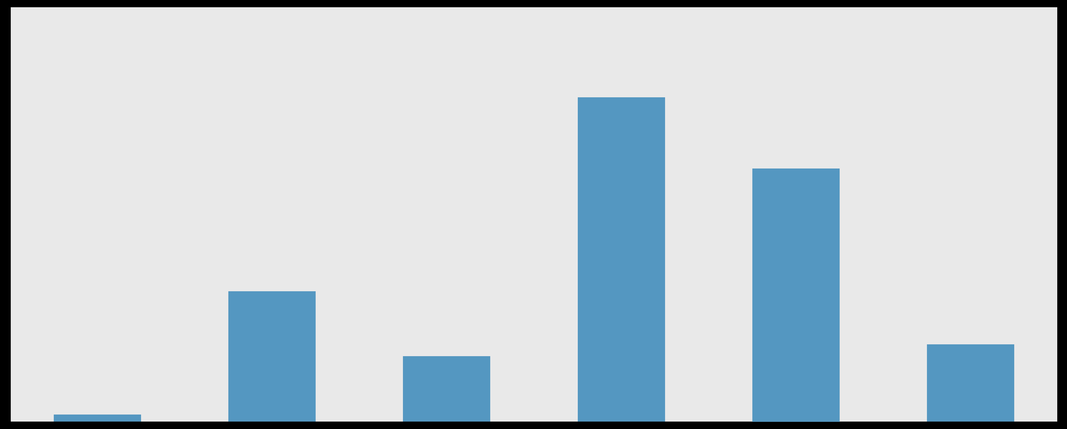
Names and describes where to activate an interaction

- A point
- In a view
- With a name

UIAccessibilityLocationDescriptor

Names and describes where to activate an interaction

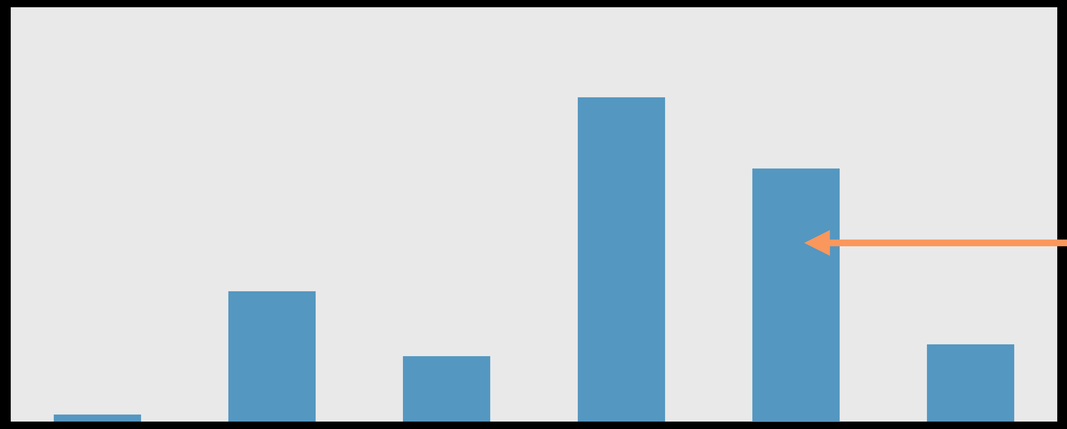
- A point
- In a view
- With a name



UIAccessibilityLocationDescriptor

Names and describes where to activate an interaction

- A point
- In a view
- With a name



accessibilityDragSourceDescriptors and accessibilityDropPointDescriptors

accessibilityDragSourceDescriptors and accessibilityDropPointDescriptors

`accessibilityDragSourceDescriptors`

accessibilityDragSourceDescriptors and accessibilityDropPointDescriptors

`accessibilityDragSourceDescriptors`

- Expose drag sources logically associated with this element

accessibilityDragSourceDescriptors and accessibilityDropPointDescriptors

`accessibilityDragSourceDescriptors`

- Expose drag sources logically associated with this element

`accessibilityDropPointDescriptors`

accessibilityDragSourceDescriptors and accessibilityDropPointDescriptors

`accessibilityDragSourceDescriptors`

- Expose drag sources logically associated with this element

`accessibilityDropPointDescriptors`

- Expose drop points logically associated with this element

accessibilityDragSourceDescriptors and accessibilityDropPointDescriptors

`accessibilityDragSourceDescriptors`

- Expose drag sources logically associated with this element

`accessibilityDropPointDescriptors`

- Expose drop points logically associated with this element

Descriptors must reference the views with the relevant interaction

```
let descriptor = UIAccessibilityLocationDescriptor(name: “Drag from specified point",
                                                   point: dragPoint, in: view)
element.accessibilityDragSourceDescriptors = [descriptor]
```

```
func makeAccessibilityElements() {
    self.accessibilityElements = bars.enumerated().map { (index, barLayer) in
        let element = UIAccessibilityElement(accessibilityContainer: self)
        element.accessibilityFrameInContainerSpace = barLayer.frame
        element.accessibilityLabel = seriesLabels[index]
        element.accessibilityValue = "\(series[index])"




        return element
    }
}
```

```
func makeAccessibilityElements() {
    self.accessibilityElements = bars.enumerated().map { (index, barLayer) in
        let element = UIAccessibilityElement(accessibilityContainer: self)
        element.accessibilityFrameInContainerSpace = barLayer.frame
        element.accessibilityLabel = seriesLabels[index]
        element.accessibilityValue = "\(series[index])"
        let dragPoint = CGPoint(x: barLayer.frame.midX, y: barLayer.frame.midY)
        let descriptor = UIAccessibilityLocationDescriptor(name: "Drag bar data",
                                                           point:dragPoint, in: self)
        element.accessibilityDragSourceDescriptors = [descriptor]
        return element
    }
}
```

Exposing Multiple Drops

Example

Exposing Multiple Drops

Example

Contact card

Exposing Multiple Drops

Example

Contact card

Exposing Multiple Drops

Example

Contact card

- Card is one element

Exposing Multiple Drops

Example

Contact card

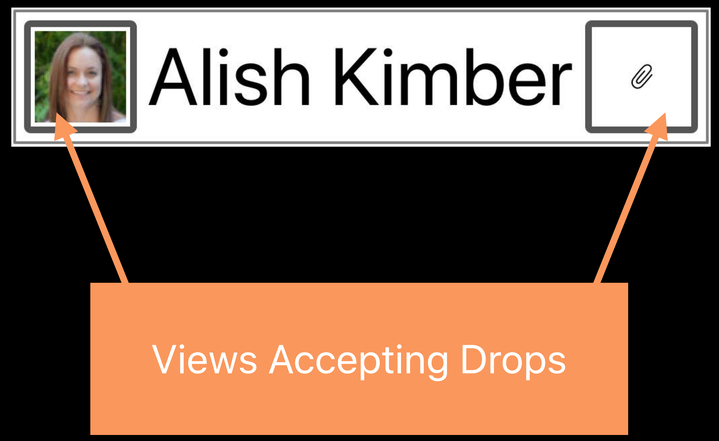
- Card is one element
- Drop into multiple “wells” in the card

Exposing Multiple Drops

Example

Contact card

- Card is one element
- Drop into multiple “wells” in the card



```
override var accessibilityDropPointDescriptors: [UIAccessibilityLocationDescriptor]? {
    get {
        let photoWellMidpoint = CGPoint(x: self.contactPhotoWell.bounds.midX,
                                        y: self.contactPhotoWell.bounds.midY)
        let attachmentsWellMidpoint = CGPoint(x: self.attachmentsWell.bounds.midX,
                                              y: self.attachmentsWell.bounds.midY)
        return [UIAccessibilityLocationDescriptor(name: "Drop into portrait",
                                                  point: photoWellMidpoint,
                                                  in: self.contactPhotoWell),
                UIAccessibilityLocationDescriptor(name: "Drop into attachments",
                                                  point: attachmentsWellMidpoint,
                                                  in:self.attachmentsWell)]
    }
    set {}
}
```

```
override var accessibilityDropPointDescriptors: [UIAccessibilityLocationDescriptor]? {
    get {
        let photoWellMidpoint = CGPoint(x: self.contactPhotoWell.bounds.midX,
                                        y: self.contactPhotoWell.bounds.midY)
        let attachmentsWellMidpoint = CGPoint(x: self.attachmentsWell.bounds.midX,
                                              y: self.attachmentsWell.bounds.midY)
        return [UIAccessibilityLocationDescriptor(name: "Drop into portrait",
                                                  point: photoWellMidpoint,
                                                  in: self.contactPhotoWell),
                UIAccessibilityLocationDescriptor(name: "Drop into attachments",
                                                  point: attachmentsWellMidpoint,
                                                  in:self.attachmentsWell)]
    }
    set {}
}
```

```
override var accessibilityDropPointDescriptors: [UIAccessibilityLocationDescriptor]? {
    get {
        let photoWellMidpoint = CGPoint(x: self.contactPhotoWell.bounds.midX,
                                        y: self.contactPhotoWell.bounds.midY)
        let attachmentsWellMidpoint = CGPoint(x: self.attachmentsWell.bounds.midX,
                                              y: self.attachmentsWell.bounds.midY)
        return [UIAccessibilityLocationDescriptor(name: "Drop into portrait",
                                                  point: photoWellMidpoint,
                                                  in: self.contactPhotoWell),
                UIAccessibilityLocationDescriptor(name: "Drop into attachments",
                                                  point: attachmentsWellMidpoint,
                                                  in:self.attachmentsWell)]
    }
    set {}
}
```

Summary

Summary

Expose the drags and drops associated with your elements to Accessibility

Summary

Expose the drags and drops associated with your elements to Accessibility

`accessibilityDragSourceDescriptors` for drags

Summary

Expose the drags and drops associated with your elements to Accessibility

`accessibilityDragSourceDescriptors` for drags

`accessibilityDropPointDescriptors` for drops

Summary

Expose the drags and drops associated with your elements to Accessibility

`accessibilityDragSourceDescriptors` for drags

`accessibilityDropPointDescriptors` for drops

`UIAccessibilityLocationDescriptor` describes drags and drops to Accessibility

More Information

https://developer.apple.com/wwdc18/241


WWDC18