The Keys to a Better Text Input Experience
Crafting a better typing experience in your app
Session 242

Kasia Wawer, iOS Keyboards
Shuchen Li, iOS Keyboards
James Magahern, iOS Keyboards
Create a Better Input Experience
Create a Better Input Experience

Integrate the keyboard into your layout
Create a Better Input Experience

Integrate the keyboard into your layout

Create dynamic input accessory views
Create a Better Input Experience

Integrate the keyboard into your layout
Create dynamic input accessory views
Make your app multilingual
Create a Better Input Experience

Integrate the keyboard into your layout

Create dynamic input accessory views

Make your app multilingual

Use traits for smarter QuickType
Create a Better Input Experience

• Integrate the keyboard into your layout
• Create dynamic input accessory views
• Make your app multilingual
• Use traits for smarter QuickType
• Hardware keyboard support
Create a Better Input Experience

Integrate the keyboard into your layout
Create dynamic input accessory views
Make your app multilingual
Use traits for smarter QuickType
Hardware keyboard support
Custom input views
Create a Better Input Experience

Integrate the keyboard into your layout
Create dynamic input accessory views
Make your app multilingual
Use traits for smarter QuickType
Hardware keyboard support
Custom input views
Keyboard extension tips and best practices
Integrating the Keyboard into Your App
Adaptivity and input accessory views

Kasia Wawer, iOS Keyboards
CatChat

Simba
I left so much food this morning; how did you eat it all?

Faithful
Let me know when you wake up

Penny
You’re not lost, you’re probably in the closet again

Nala
Keep an eye on your brother, please
Accounting for changing keyboard heights
Working with non-scrolling layouts
Working with scrolling layouts
Adding an input accessory view
Accounting for the Keyboard
Accounting for the Keyboard

Heights vary by language and settings
Accounting for the Keyboard

Heights vary by language and settings
Accounting for the Keyboard

Heights vary by language and settings
Accounting for the Keyboard

Heights vary by language and settings
Accounting for the Keyboard

Heights vary by language and settings
Accounting for the Keyboard

Heights vary by language and settings

Register for notifications

- UIKeyboardDidShow
- UIKeyboardDidHide
- UIKeyboardDidChangeFrame
Undocked and Split Keyboards
Undocked and Split Keyboards

Dismissing and undocking both send Hide notifications
Undocked and Split Keyboards

Dismissing and undocking both send Hide notifications

Frame change notifications will continue when undocked
Undocked and Split Keyboards

Dismissing and undocking both send Hide notifications.

Frame change notifications will continue when undocked.

Track the most recent Hide or Show event.
Simba
Food?
I'll make sure to pick up extra on the way home
Please don't try to open the fridge again.

Penny
You're not lost, you're probably in the closet again
More food?
I left so much food this morning; how did you eat it all already??

Nala
Keep an eye on your brother, please
Food!
I'll be there as soon as I can
... food?
Sorry, you'll just have to be patient!

I promise you'll get fed as soon as I get home, ok?
Simba

Food?

I'll make sure to pick up extra on the way home.

Please don't try to open the fridge again.

Faithful

Let me know when you wake up

Penny

You're not lost, you're probably in the closet again.

Nala

Keep an eye on your brother, please

More food?

I left so much food this morning; how did you eat it all already??

Food!

I'll be there as soon as I can.

...food?

Sorry, you'll just have to be patient!

Food!

I promise you'll get fed as soon as I get home, ok?

😊
Being in the Right Space

@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        guard let userInfo = notification.userInfo,
            let frame = userInfo[UIKeyboardFrameEndUserInfoKey] as? CGRect else { return }
        let convertedFrame = view.convert(frame, from: UIScreen.main.coordinateSpace)
        let intersectedKeyboardHeight = view.frame.intersection(convertedFrame).height
    }
}
Being in the Right Space

Keyboard is always in the screen coordinate space

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        guard let userInfo = notification.userInfo,
        let frame = userInfo[UIKeyboardFrameEndUserInfoKey] as? CGRect else { return }
        let convertedFrame = view.convert(frame, from: UIScreen.main.coordinateSpace)
        let intersectedKeyboardHeight = view.frame.intersection(convertedFrame).height
    }
}
```
Being in the Right Space

Keyboard is always in the screen coordinate space

Convert the frame from screen coordinates

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        guard let userInfo = notification.userInfo,
             let frame = userInfo[UIKeyboardFrameEndUserInfoKey] as? CGRect else { return }
        let convertedFrame = view.convert(frame, from: UIScreen.main.coordinateSpace)
        let intersectedKeyboardHeight = view.frame.intersection(convertedFrame).height
    }
}
```
Being in the Right Space

Keyboard is always in the screen coordinate space

Convert the frame from screen coordinates

Use the intersection of the converted keyboard frame and your view

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        guard let userInfo = notification.userInfo,
            let frame = userInfo[UIKeyboardFrameEndUserInfoKey] as? CGRect else { return }
        let convertedFrame = view.convert(frame, from: UIScreen.main.coordinateSpace)
        let intersectedKeyboardHeight = view.frame.intersection(convertedFrame).height
    }
}
```
Accounting for changing keyboard heights

Working with non-scrolling layouts

Working with scrolling layouts

Adding an input accessory view
Adaptive Keyboard-Inclusive Layouts

Create a custom `UILayoutGuide` and height constraint

```swift
func setUpViews() {
    // ... view set up ...
    let keyboardGuide = UILayoutGuide()
    view.addLayoutGuide(keyboardGuide)
    heightConstraint = keyboardGuide.heightAnchor.constraint(equalToConstant: kDefaultHeight)
    heightConstraint.isActive = true
    // ... view set up ...
}
```
Adaptive Keyboard-Inclusive Layouts

Create a custom `UILayoutGuide` and height constraint

Tie lowest view to top of your layout guide

```swift
func setUpViews() {
    // ...
    keyboardGuide.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor)
        .isActive = true
    bottomSpacer.bottomAnchor.constraint(equalTo: keyboardGuide.topAnchor).isActive = true
    // ...
}
```
Adaptive Keyboard-Inclusive Layouts

Create a custom `UILayoutGuide` and height constraint

Tie lowest view to top of your layout guide

```swift
func setUpViews() {
    // ...
    keyboardGuide.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor)
        .isActive = true
    bottomSpacer.bottomAnchor.constraint(equalTo: keyboardGuide.topAnchor).isActive = true
    // ...
}
```
Adaptive Keyboard-Inclusive Layouts

Create a custom `UILayoutGuide` and height constraint

Tie lowest view to top of your layout guide

Use the converted frame to set the height constraint of layout guide

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        UIView.animate(withDuration: 0.2) {
            heightConstraint.constant = intersectedKeyboardHeight
            view.layoutIfNeeded()
        }
    }
}
```
Adaptive Keyboard-Inclusive Layouts

Create a custom `UILayoutGuide` and height constraint

Tie lowest view to top of your layout guide

Use the converted frame to set the height constraint of layout guide

```objc
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        UIView.animate(withDuration: 0.2) {
            heightConstraint.constant = intersectedKeyboardHeight
            view.layoutIfNeeded()
        }
    }
}
```
Accounting for changing keyboard heights
Working with non-scrolling layouts
Working with scrolling layouts
Adding an input accessory view
Just checking in!

How's it going today?

Food?

I'll pick up some more on the way home.

Please don't try to open the fridge again.

More food now?

I left so much food this morning; how did you eat it all?

Food!

I'll be there as soon as I can!

... food?
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        scrollView.contentInset.bottom = intersectedKeyboardHeight
        // ... Handle content scrolling ...
    }
}
Scrolling Views and Keyboards

Make sure the keyboard is visible

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        scrollView.contentInset.bottom = intersectedKeyboardHeight
        // ... Handle content scrolling ...
    }
}
```
Scrolling Views and Keyboards

Make sure the keyboard is visible

Convert the frame and get the height

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        scrollView.contentInset.bottom = intersectedKeyboardHeight
        // ... Handle content scrolling ...
    }
}
```
Scrolling Views and Keyboards

Make sure the keyboard is visible

Convert the frame and get the height

Set content insets appropriately

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        scrollView.contentInset.bottom = intersectedKeyboardHeight
        // ... Handle content scrolling ...
    }
}
```
Scrolling Views and Keyboards

Make sure the keyboard is visible

Convert the frame and get the height

Set content insets appropriately

Handle scrolling the content if needed

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    if !keyboardIsHidden {
        // ... Convert frame ...
        scrollView.contentInset.bottom = intersectedKeyboardHeight
        // ... Handle content scrolling ...
    }
}
```
UITableViewController

Sets insets for you

Handle scrolling the content if needed
UITableViewController

Sets insets for you

Handle scrolling the content if needed

```swift
@objc func keyboardFrameChanged(_ notification: Notification) -> Void {
    let bottomRow = IndexPath(row: items.count - 1, section: 0)
    tableView.scrollToRow(at: bottomRow, at: UITableViewScrollPosition.bottom, animated: true)
}
```
Accounting for changing keyboard heights
Working with non-scrolling layouts
Working with scrolling layouts
Adding an input accessory view
CatChat

Simba

I left so much food this morning; how did you eat it all?

Food!

I'll be there as soon as I can!

...food?

I can't right now; you'll just have to be more patient!
Accessorizing the Keyboard
Accessorizing the Keyboard

Host view controller returns `true` for `canBecomeFirstResponder`
Accessorizing the Keyboard

Host view controller returns `true` for `canBecomeFirstResponder`

Use custom view or view controller

• For view, override `inputAccessoryView`
• For view controller, override `inputAccessoryViewController`
Accessorizing the Keyboard

Host view controller returns `true` for `canBecomeFirstResponder`.

Use custom view or view controller

• For view, override `inputAccessoryView`
• For view controller, override `inputAccessoryViewController`

Changing height sends frame change notifications
Making Dynamic Input Accessory Views
Making Dynamic Input Accessory Views

Use expanding view to define height
Making Dynamic Input Accessory Views

Use expanding view to define height

Pin to top and bottom of content view
Making Dynamic Input Accessory Views

Use expanding view to define height

Pin to top and bottom of content view

Use or define intrinsicContentSize

```swift
expandingTextView.textContainer.heightTracksTextView = true
expandingTextView.isScrollEnabled = false
```
Making Dynamic Input Accessory Views

Use expanding view to define height
Pin to top and bottom of content view
Use or define intrinsicContentSize

expandingTextView.textContainer.heightTracksTextView = true
expandingTextView.isScrollEnabled = false

I’ll let you know if I can come by then.
/ Using intrinsicContentSize to determine height

override var intrinsicContentSize: CGSize {
    var newSize = self.bounds.size
    newSize.height = kMinimumHeight
    if expandingTextView.bounds.size.height > 0.0 {
        newSize.height = expandingTextView.bounds.size.height + kVerticalPadding
    }
    if newSize.height > kMaximumHeight {
        newSize.height = kMaximumHeight
    }
    return newSize
}
// Using intrinsicContentSize to determine height

override var intrinsicContentSize: CGSize {
  var newSize = self.bounds.size
  newSize.height = kMinimumHeight
  if expandingTextView.bounds.size.height > 0.0 {
    newSize.height = expandingTextView.bounds.size.height + kVerticalPadding
  }
  if newSize.height > kMaximumHeight {
    newSize.height = kMaximumHeight
  }
  return newSize
}
// Using intrinsicContentSize to determine height

override var intrinsicContentSize: CGSize {
  var newSize = self.bounds.size
  newSize.height = kMinimumHeight
  if expandingTextView.bounds.size.height > 0.0 {
    newSize.height = expandingTextView.bounds.size.height + kVerticalPadding
  }
  if newSize.height > kMaximumHeight {
    newSize.height = kMaximumHeight
  }
  return newSize
}
// Using intrinsicContentSize to determine height

override var intrinsicContentSize: CGSize {
    var newSize = self.bounds.size
    newSize.height = kMinimumHeight
    if expandingTextView.bounds.size.height > 0.0 {
        newSize.height = expandingTextView.bounds.size.height + kVerticalPadding
    }
    if newSize.height > kMaximumHeight {
        newSize.height = kMaximumHeight
    }
    return newSize
}
override var intrinsicContentSize: CGSize {
    var newSize = self.bounds.size
    newSize.height = kMinimumHeight
    if expandingTextView.bounds.size.height > 0.0 {
        newSize.height = expandingTextView.bounds.size.height + kVerticalPadding
    }
    if newSize.height > kMaximumHeight {
        newSize.height = kMaximumHeight
    }
    return newSize
}
// Using intrinsicContentSize to determine height

override var intrinsicContentSize: CGSize {
    var newSize = self.bounds.size
    newSize.height = kMinimumHeight
    if expandingTextView.bounds.size.height > 0.0 {
        newSize.height = expandingTextView.bounds.size.height + kVerticalPadding
    }
    if newSize.height > kMaximumHeight {
        newSize.height = kMaximumHeight
    }
    return newSize
}
More food now?

I left so much food this morning; how did you eat it all?

Food!

I'll be there as soon as I can!

... food?

I'm busy right now,
Making Your App Feel Magical

Using context to enrich text input experience

Shuchen Li, iOS Keyboards
Multilingual
Being aware of context
The new “smarts”
Marked text
Hardware keyboard
Great! Do you want to grab a coffee before we go?

Sure. Do you want to meet in front of the building?

OK. I'll see you in 5 mins.

Need to return the projector to admin before they close.

Take your time.
周末聚会还有什么需要帮忙的吗？
都准备得差不多了😊
你的蛋糕需要盘子和蜡烛吗？
我有蜡烛，你准备盘子吧😊
我还准备了❤️
小朋友们肯定会非常开心的😊
😊
Remembering User Selected Keyboard

Your App

User-selected Keyboard
Context Identifier String

Look up Identifier
Set Selected Keyboard

User Defaults

<unique identifier 1> : en_US
<unique identifier 2> : en_US
<unique identifier 3> : zh_Hans
<unique identifier 4> : pl_PL
...

Remembering User Selected Keyboard

User-selected Keyboard

Context Identifier String

Look up Identifier

Set Selected Keyboard

Your App

User Defaults

<unique identifier 1> : en_US
<unique identifier 2> : en_US
<unique identifier 3> : zh_Hans
<unique identifier 4> : pl_PL
...


What kind of food does Misiu like?

Anything is fine. No sweets!
ChatInputAccessoryView becomesFirstResponder
ChatInputAccessoryView

becomesFirstResponder
ChatInputAccessoryView

becomesFirstResponder
ChatInputAccessoryView
implemented by
ConversationViewController
becomesFirstResponder
becomesFirstResponder
implemented by
Navigation Controller
becomesFirstResponder

ChatInputAccessoryView

ConversationViewController

Navigation Controller

becomesFirstResponder
class ConversationViewController: UITableViewController, UITextViewDelegate {

    // ... other code ...

    override var textInputContextIdentifier: String? {
        // Returning some unique identifier here allows the keyboard to remember
        // which language the user was typing in when they were last communicating
        // with this person.

        // It can be anything, as long as it's unique to each
        // recipient (here we're just returning the name)
        return self.conversation?.otherParticipant
    }

    // ... other code ...
}
class ConversationViewController: UITableViewController, UITextViewDelegate {

    // ... other code ... 

    override var textInputContextIdentifier: String? {
        return self.conversation?.otherParticipant
    }

    // ... other code ...
}
Demo

Give your app a memory

Kasia Wawer, iOS Keyboards
UIResponder to Remember Keyboard

textInputContextIdentifier

UITextInputMode
Multilingual
Being aware of context
The new “smarts”
Marked text
Hardware keyboard
Being Aware of Context

UIKeyboardType

UITextContentTypeTypes
QuickType personal information
QuickType personal information
Add new pet

Penny

John Appleseed

(505) 555-0155

j.appleseed@icloud.com

Pet's home address
QuickType address
QuickType sources
UITextContentType

Provides contextual predictions

Displays on QuickType bar
Content Types for Password AutoFill

Introducing Password AutoFill for Apps

UITextContentTypeUsername

UITextContentTypePassword

Log In

WWDC 2017
Multilingual
Being aware of context
The new “smarts”
Marked text
Hardware keyboard
Smart Quote and Smart Dash
# Smart Quote and Smart Dash

<table>
<thead>
<tr>
<th>Font</th>
<th>Smart Quote</th>
<th>Smart Dash</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF Hello</td>
<td>&quot;a&quot;</td>
<td></td>
</tr>
<tr>
<td>Helvetica Neue</td>
<td>&quot;a&quot;</td>
<td></td>
</tr>
<tr>
<td>Lucida Grande</td>
<td>&quot;a&quot;</td>
<td></td>
</tr>
<tr>
<td>Avenir</td>
<td>&quot;a&quot;</td>
<td></td>
</tr>
<tr>
<td>Myriad Set</td>
<td>&quot;a&quot;</td>
<td></td>
</tr>
</tbody>
</table>
## Smart Quote and Smart Dash

<table>
<thead>
<tr>
<th>Font</th>
<th>Smart Quote</th>
<th>Smart Dash</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF Hello</td>
<td>&quot;a&quot;</td>
<td>&quot;a&quot;</td>
</tr>
<tr>
<td>Helvetica Neue</td>
<td>&quot;a&quot;</td>
<td>&quot;a&quot;</td>
</tr>
<tr>
<td>Lucida Grande</td>
<td>&quot;a&quot;</td>
<td>&quot;a&quot;</td>
</tr>
<tr>
<td>Avenir</td>
<td>&quot;a&quot;</td>
<td>&quot;a&quot;</td>
</tr>
<tr>
<td>Myriad Set</td>
<td>&quot;a&quot;</td>
<td>&quot;a&quot;</td>
</tr>
</tbody>
</table>
# Smart Quote and Smart Dash

<table>
<thead>
<tr>
<th>Font</th>
<th>&quot;a&quot;</th>
<th>“a”</th>
<th>Hyphen: 1-dash</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF Hello</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td>- → -</td>
</tr>
<tr>
<td>Helvetica Neue</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
</tr>
<tr>
<td>Lucida Grande</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
</tr>
<tr>
<td>Avenir</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
</tr>
<tr>
<td>Myriad Set</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
</tr>
</tbody>
</table>
# Smart Quote and Smart Dash

<table>
<thead>
<tr>
<th>Font</th>
<th>Double Quote</th>
<th>Single Quote</th>
<th>Hyphen: 1-dash</th>
<th>En dash: 2-dash</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF Hello</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Helvetica Neue</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lucida Grande</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Avenir</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Myriad Set</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Smart Quote and Smart Dash

<table>
<thead>
<tr>
<th>Font</th>
<th>Single Quote</th>
<th>Double Quote</th>
<th>Hyphen: 1-dash</th>
<th>En dash: 2-dash</th>
<th>Em dash: 3-dash</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF Hello</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td>– → –</td>
<td>– – → –</td>
<td>– – – → – –</td>
</tr>
<tr>
<td>Helvetica Neue</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td>– – → –</td>
<td>– – – → – –</td>
</tr>
<tr>
<td>Lucida Grande</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td>– – → –</td>
<td>– – – → – –</td>
</tr>
<tr>
<td>Avenir</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td>– – → –</td>
<td>– – – → – –</td>
</tr>
<tr>
<td>Myriad Set</td>
<td>&quot;a&quot;</td>
<td>“a”</td>
<td></td>
<td>– – → –</td>
<td>– – – → – –</td>
</tr>
</tbody>
</table>
Smart Insertion and Deletion

Vivian will bring Misiu to the party.
Vivian will bring Misiu and Jiwang to the party.
Vivian will bring Misiu and Jiwang to the party.
Vivian will bring Misiu and Jiwang to the party.
Vivian will bring Misiu and Jiwang to the party.
Vivian will bring Misiu and Jiwang to the party.
Vivian will bring Misiu to the party.
Vivian will bring Misiu to the party.
UITextInputTraits
UITextInputTraits

.default
.yes
.no
UITextInputTraits

- default
- yes
- no

Understand your text entry
Marked text search
Giving text widget a memory
Being aware of context
The new “smarts”
Marked text
Hardware keyboard
Working with Hardware Keyboard

Key commands for hardware keyboards
class ConversationViewController: UITableViewController, UITextViewDelegate {

// ... some code ...

override var keyCommands: [UIKeyCommand]? {
    return [
        // Command + Down arrow goes to the next conversation
        UIKeyCommand(input: UIKeyInputDownArrow,
                      modifierFlags: .command,
                      action: #selector(switchToConversationKeyCommandInvoked(sender:)),
                      discoverabilityTitle: NSLocalizedString("GO_TO_NEXT_CONVERSATION", comment: "")),

        // Command + Up arrow goes to the previous conversation
        UIKeyCommand(input: UIKeyInputUpArrow,
                      modifierFlags: .command,
                      action: #selector(switchToConversationKeyCommandInvoked(sender:)),
                      discoverabilityTitle: NSLocalizedString("GO_TO_PREV_CONVERSATION", comment: ""))
    ]
}

//... some code ...
}
class ConversationViewController: UITableViewController, UITextViewDelegate {

    // ... some code ...

    override var keyCommands: [UIKeyCommand]? {
        return [
            // Command + Down arrow goes to the next conversation
            UIKeyCommand(input: UIKeyInputDownArrow,
                          modifierFlags: .command,
                          action: #selector(switchToConversationKeyCommandInvoked(sender:)),
                          discoverabilityTitle: NSLocalizedString("GO_TO_NEXT_CONVERSATION", comment: "")
            ),

            // Command + Up arrow goes to the previous conversation
            UIKeyCommand(input: UIKeyInputUpArrow,
                          modifierFlags: .command,
                          action: #selector(switchToConversationKeyCommandInvoked(sender:)),
                          discoverabilityTitle: NSLocalizedString("GO_TO_PREV_CONVERSATION", comment: "")
            ]
        }

        //... some code ...
    }
class ConversationViewController: UITableViewController, UITextViewDelegate {

    // ... some code ...

    override var keyCommands: [UIKeyCommand]?
    {
        return [
            // Command + Down arrow goes to the next conversation
            UIKeyCommand(input: UIKeyInputDownArrow,
                          modifierFlags: .command,
                          action: #selector(switchToConversationKeyCommandInvoked(sender:)),
                          discoverabilityTitle: NSLocalizedString("GO_TO_NEXT_CONVERSATION", comment: "")

            // Command + Up arrow goes to the previous conversation
            UIKeyCommand(input: UIKeyInputUpArrow,
                          modifierFlags: .command,
                          action: #selector(switchToConversationKeyCommandInvoked(sender:)),
                          discoverabilityTitle: NSLocalizedString("GO_TO_PREV_CONVERSATION", comment: "")
        ]
    }

    //... some code ...
}
class ConversationViewController: UITableViewController, UITextViewDelegate {

    // ... some code ...

    override var keyCommands: [UIKeyCommand]? {
        return [
            // Command + Down arrow goes to the next conversation
            UIKeyCommand(input: UIKeyInputDownArrow, 
                         modifierFlags: .command, 
                         action: #selector(switchToConversationKeyCommandInvoked(sender:)), 
                         discoverabilityTitle: NSLocalizedString("GO_TO_NEXT_CONVERSATION", comment: "") ),
            // Command + Up arrow goes to the previous conversation
            UIKeyCommand(input: UIKeyInputUpArrow, 
                         modifierFlags: .command, 
                         action: #selector(switchToConversationKeyCommandInvoked(sender:)), 
                         discoverabilityTitle: NSLocalizedString("GO_TO_PREV_CONVERSATION", comment: ""))
        ]
    }

    //... some code ...
}
<table>
<thead>
<tr>
<th>Name</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misau</td>
<td>Running late! I promise I'll feed you soon!</td>
</tr>
<tr>
<td>Jiwang</td>
<td></td>
</tr>
<tr>
<td>Vivian Li</td>
<td>Anything is fine. No sweets!</td>
</tr>
</tbody>
</table>
Multilingual
Being aware of context
The new “smarts”
Marked text
Hardware keyboard
• Multilingual
• Being aware of context
• The new “smarts”
• Marked text
• Hardware keyboard
Creating Custom Input Views

Accessible input for cats, dogs, and more

James Magahern, iOS Keyboards
\[ \sum_{i=1}^{n} 573x + 12i + 4i^2 \]
The Shortest Route

Learned up to now. (You might need your debugging skills, too!)

moveForward();
    moveForward();
collectGem();
moveForward();
moveForward();
moveForward();
<table>
<thead>
<tr>
<th>Song</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monkey Business</td>
<td>999870</td>
</tr>
<tr>
<td>Blew My Mind</td>
<td>999980</td>
</tr>
<tr>
<td>Come To Life</td>
<td>999990</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Custom Accessible Input Views for Pets
Custom Accessible Input Views for Pets

Our pets have no way to talk back
Custom Accessible Input Views for Pets

Our pets have no way to talk back

Cats and dogs have trouble using QWERTY
Custom Accessible Input Views for Pets

Our pets have no way to talk back

Cats and dogs have trouble using QWERTY

Limited vocabulary
Food!

Okay I'll be home in a bit. Please don't eat the furniture!
UIInputViewController

UIInputView
Providing a Custom Input View

class ConversationViewController: UITableViewController, UITextViewDelegate {

Providing a Custom Input View

class ConversationViewController: UITableViewController, UITextViewDelegate {

    private let customInputView = AnimalInputView()

    override var canBecomeFirstResponder: Bool {
        return true
    }
}
class ConversationViewController: UITableViewController, UITextViewDelegate {

    private let customInputView = AnimalInputView()

    override var canBecomeFirstResponder: Bool {
        return true
    }

    override var inputView: UIInputView? {
        // Return an instance of our custom UIInputView subclass
        return customInputView
    }

    // ... other code ...
}
How's my pup?
Demo
Converting to a keyboard extension

James Magahern, iOS Keyboards
Converting to a Keyboard Extension
Converting to a Keyboard Extension

Create a new target
Converting to a Keyboard Extension

Create a new target

Set up your `UIInputViewController` subclass
Converting to a Keyboard Extension

Create a new target

Set up your `UIInputViewController` subclass

Users can now enable keyboard from your settings bundle
New APIs in iOS 11
New APIs in iOS 11

Selected text
New APIs in iOS 11

Selected text
documentIdentifier handle
New APIs in iOS 11

Selected text

documentIdentifier handle

Ability to query for full access
Other Third Party Keyboard APIs
Other Third Party Keyboard APIs

Incorporate the system input menu
Other Third Party Keyboard APIs

Incorporate the system input menu

Personalization with the supplementary lexicon
Other Third Party Keyboard APIs

- Incorporate the system input menu
- Personalization with the supplementary lexicon
- Multilingual support
Designing for User Trust
Designing for User Trust

Privacy
Designing for User Trust

Privacy

Enhance with user data
Designing for User Trust

Privacy

Enhance with user data

Requesting full access
Full Access and Privacy
Full Access and Privacy

Value in not asking for full access
Full Access and Privacy

Value in not asking for full access

Communicating with your main app
Full Access and Privacy

Value in not asking for full access

Communicating with your main app

Networking
Full Access and Privacy

Value in not asking for full access

Communicating with your main app

Networking

Current location
Full Access and Privacy

Value in not asking for full access

Communicating with your main app

Networking

Current location

Address book
Full Access and Privacy

Value in not asking for full access

Communicating with your main app

Networking

Current location

Address book

Keyboard needs to work without it
Summary

Design your app with the keyboard in mind

Use advanced traits to enhance the user’s experience

Building keyboard extensions is a lot easier than you think
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>Session Title</th>
<th>WWDC Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing Password AutoFill for Apps</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Localizing for Xcode 9</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Increase Usage of Your App With Proactive Suggestions</td>
<td>WWDC 2016</td>
</tr>
<tr>
<td>Internationalization Best Practices</td>
<td>WWDC 2016</td>
</tr>
<tr>
<td>Mysteries of Auto Layout, Part 1</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>Labs</td>
<td>Technology Lab C</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Cocoa Touch and Haptics Lab</td>
<td></td>
</tr>
</tbody>
</table>