New UIKit Support for International User Interfaces

Session 222

Sara Radi Internationalization Software Engineer
Aaltan Ahmad Internationalization Software Engineer
Paul Borokhov Internationalization Software Engineer
Designing UI for RTL Languages
Chapter 26

Mrs. Gardiner's caution to Elizabeth was punctually and kindly given on the first favourable opportunity of speaking to her alone; after honestly telling her what she thought, she thus went on:

"You are too sensible a girl, Lizzie. It is very easy for you to fall in love merely because you are exposed to it; and, therefore, I am not afraid of speaking openly. Settle it in your mind that you will not involve yourself or endanger yourself by involving him in an affection which, if not consummated, would have been mortifying. Do not attempt to persuade him to love you. But he is proud and cannot think of being in love with you. He must not know it to me; I shall not mention his name to him. But I shall try to prevent his being idle and not to know you better. But as you do not love him, you should not. I see fancy run away with you, but it must not.

Oh! that abstemious--"
Chapter 26

Mrs. Gardiner’s caution to Elizabeth was punctually and kindly given on the first favourable opportunity of speaking to her alone; after honestly telling her what she thought, she thus went on:

“You are too sensible a girl, Lizzie, to fall in love merely because you are warned against it; and, therefore, I am not afraid of speaking openly. Set yourself not to involve yourself or endanger any man’s mind; but when fortune would lead you to Mr. Darcy, be impudent. I have no doubt he would like it best if you were the most, and if he had, I should love better. But as it is, one should not see fancy run away with one.”
Over 500 million native speakers
Agenda

Right-to-Left (RTL) User Interface Challenges
Agenda

Right-to-Left (RTL) User Interface Challenges
Supporting RTL UI with UIKit Controls
Agenda

Right-to-Left (RTL) User Interface Challenges
Supporting RTL UI with UIKit Controls
Custom Layout
Agenda

Right-to-Left (RTL) User Interface Challenges
Supporting RTL UI with UIKit Controls
Custom Layout
Exceptions
Overview
Order
Order

1 2

LTR reading direction
Order

1 \hspace{1cm} 2 \hspace{1cm} 3

LTR reading direction
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas a scelerisque orci. Sed bibendum interdum orci, id luctus est luctus vitae.
LOREM

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas a scelerisque orci. Sed bibendum interdum orci, id luctus est luctus vitae.
New in UIKit

Right-to-left support
Standard Controls
Table Views
Table Views

Left-to-Right
Table Views

Left-to-Right

Right-to-Left
Navigation Controller
Tracking Gestures
Tracking Gestures
Enabling Right-to-Left Support
Enabling Right-to-Left Support

Link against iOS 9
Enabling Right-to-Left Support

Link against iOS 9

As simple as adding a RTL localization
Enabling Right-to-Left Support

Link against iOS 9
As simple as adding a RTL localization

Base.lproj/Main.storyboard
Enabling Right-to-Left Support

Link against iOS 9
As simple as adding a RTL localization
Enabling Right-to-Left Support

Link against iOS 9
As simple as adding a RTL localization
User Interface Testing
<table>
<thead>
<tr>
<th>Core Location</th>
<th>Allow Location Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Location</td>
<td>None</td>
</tr>
</tbody>
</table>

| Application Data | None |
| Routing App Coverage File | None |

| GPU Frame Capture | Automatically Enabled |
| Metal API Validation | Enabled |

| Background Fetch | Launch due to a background fetch event |
| Localization Debugging | Show non-localized strings |

| Application Language | System Language |
| Application Region | System Region |

<p>| XPC Services | Debug XPC services used by this application |
| View Debugging | Enable user interface debugging |</p>
<table>
<thead>
<tr>
<th>Core Location</th>
<th>Allow Location Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Location</td>
<td>None</td>
</tr>
<tr>
<td>Application Data</td>
<td>None</td>
</tr>
<tr>
<td>Routing App Coverage File</td>
<td>None</td>
</tr>
<tr>
<td>GPU Frame Capture</td>
<td>Automatically Enabled</td>
</tr>
<tr>
<td>Metal API Validation</td>
<td>Enabled</td>
</tr>
<tr>
<td>Background Fetch</td>
<td>Launch due to a background fetch event</td>
</tr>
<tr>
<td>Localization Debugging</td>
<td>Show non-localized strings</td>
</tr>
<tr>
<td>Application Language</td>
<td>System Language</td>
</tr>
<tr>
<td>Application Region</td>
<td>English</td>
</tr>
<tr>
<td>XP</td>
<td>Right to Left Pseudolanguage</td>
</tr>
<tr>
<td>View Debugging</td>
<td>Enable user interface debugging</td>
</tr>
</tbody>
</table>
Demo
Localization
Custom Layout
API Changes

UITextField

- `leftView/rightView` and `leftViewMode/rightViewMode` flip automatically
- `leftViewRectForBounds(_:)/rightViewRectForBounds(_:)` stay unchanged
API Changes

UITextField

- `leftView/rightView` and `leftViewMode/rightViewMode` flip automatically
- `leftViewRectForBounds(_)/rightViewRectForBounds(_)` stay unchanged

UITableView

- Insets set using the `separatorInset` property automatically flip left and right measurements
API Changes
API Changes

UISlider

• `minimumValueImage` and `maximumValueImage` flip automatically
• Be aware of adjustments done in `minimumValueImageRectForBounds(_:)` and `maximumValueImageRectForBounds(_:)`
API Changes

UISlider

- `minimumValueImage` and `maximumValueImage` flip automatically
- Be aware of adjustments done in `minimumValueImageRectForBounds(_:)` and `maximumValueImageRectForBounds(_:)`

UINavigationItem

- `leftBarButtonItem(s)` and `rightBarButtonItem(s)` flip automatically
- Beware of views added outside of API
Table View Cells
Table View Cells

Standard cells flip automatically
Table View Cells

Standard cells flip automatically
Custom layouts need to be flipped too
Collection View Flow Layouts
UICollectionViewFlowLayout supports right to left
UICollectionViewFlowLayout supports right to left
Reverse math for custom flow layouts
UICollectionViewFlowLayout supports right to left

Reverse math for custom flow layouts

• Subclass UICollectionViewFlowLayout
UICollectionViewFlowLayout supports right to left

Reverse math for custom flow layouts

- Subclass UICollectionViewFlowLayout
Auto Layout

Many reasons to use Auto Layout
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization
Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization

One more reason—right-to-left!
Auto Layout

Many reasons to use Auto Layout

• Available since iOS 6
• Different screen sizes
• Split-screen multitasking
• Localization

One more reason—right-to-left!

Can be used in storyboards, programmatically, or both
Auto Layout
Auto Layout

Use leading and trailing constraints
Auto Layout

Use leading and trailing constraints
Auto Layout

Use leading and trailing constraints
Auto Layout

Use leading and trailing constraints
Auto Layout

Use leading and trailing constraints

Xcode

First name Paul

Left-to-right

First name Paul
Auto Layout

Use leading and trailing constraints

Xcode

First name  Paul

Right-to-left

Paul  First name
Auto Layout

Use leading and trailing constraints

Storyboards

Xcode

First name Paul

Right-to-left

Paul First name
Auto Layout

Use leading and trailing constraints

Storyboards
• The default

Xcode

First name  Paul

Right-to-left

Paul  First name
Auto Layout

Use leading and trailing constraints

Storyboards
- The default

Code

Xcode

First name  Paul

Right-to-left
Paul  First name
Auto Layout

Use leading and trailing constraints

Storyboards
- The default

Code
- The default in visual format language
Auto Layout

Use leading and trailing constraints

Storyboards
• The default

Code
• The default in visual format language
• Use explicitly for manual constraints and layout anchors
Animations

Flip your x-axis animations if using frames

• Not recommended
Animations

Flip your x-axis animations if using frames

- Not recommended
Animations

Flip your x-axis animations if using frames

- Not recommended

Use Auto Layout with leading and trailing constraints instead
Animations
Animations

let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to
self.layoutIfNeeded() // make sure all frames are at the starting position
UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
Animations

```swift
let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to
self.layoutIfNeeded() // make sure all frames are at the starting position
UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
```
Animations

```swift
let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to
self.layoutIfNeeded() // make sure all frames are at the starting position
UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
```
Animations

let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to

self.layoutIfNeeded() // make sure all frames are at the starting position

UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
Animations

let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to
self.layoutIfNeeded() // make sure all frames are at the starting position

UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to
self.layoutIfNeeded() // make sure all frames are at the starting position
UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
Animations

let duration = 0.5 // time in seconds
let newOffset = 10 // new constraint value to animate to
self.layoutIfNeeded() // make sure all frames are at the starting position
UIView.animateWithDuration(duration) {
    self.animatedConstraint?.constant = newOffset
    self.layoutIfNeeded() // layout again to update the frames
}
Tracking Gestures

Gesture recognizers remain unchanged

- Inherently physical, map-to-finger movement
- A “flipped” recognizer wouldn’t make sense
Tracking Gestures
Tracking Gestures

Be aware of what’s being manipulated in UI
Tracking Gestures

Be aware of what’s being manipulated in UI

• Paintbrush on a canvas?
Tracking Gestures

Be aware of what’s being manipulated in UI

- Paintbrush on a canvas?
- Table view cell?
Tracking Gestures

Be aware of what’s being manipulated in UI

• Paintbrush on a canvas?
• Table view cell?
Tracking Gestures

Be aware of what’s being manipulated in UI

• Paintbrush on a canvas?
• Table view cell?
• Navigation?
Tracking Gestures

Be aware of what’s being manipulated in UI

- Paintbrush on a canvas?
- Table view cell?
- Navigation?

Make sure that position changes correspond to movement
Tracking Gestures

Be aware of what’s being manipulated in UI

• Paintbrush on a canvas?
• Table view cell?
• Navigation?

Make sure that position changes correspond to movement

• Use Auto Layout
Demo
Custom layout
Exceptions

Views
Exceptions

Semantic content attributes
Exceptions
Semantic content attributes

var semanticContentAttribute: UISemanticContentAttribute
Exceptions
Semantic content attributes

var semanticContentAttribute: UISemanticContentAttribute

Not all UI flips
Exceptions

Semantic content attributes

```swift
var semanticContentAttribute: UISemanticContentAttribute
```

Not all UI flips

Default is `.Unspecified`
Exceptions
Semantic content attributes

var semanticContentAttribute: UISemanticContentAttribute

Not all UI flips
Default is .Unspecified
Some UI needs different semantic content attribute for correct layout
Exceptions

Semantic content attributes

```swift
var semanticContentAttribute: UISemanticContentAttribute
```

Not all UI flips

Default is `.Unspecified`

Some UI needs different semantic content attribute for correct layout

Affects resolution of leading and trailing constraints
Exceptions
Semantic content attributes

UISemanticContentAttribute.Playback

Containers of playback controls, playhead scrubbers, etc.
Exceptions

Semantic content attributes

`UISemanticContentAttribute.Playback`

Containers of playback controls, playhead scrubbers, etc.
Exceptions
Semantic content attributes

UISemanticContentAttribute.Spatial

Groups of controls for manipulating objects or directional input on the screen
- Game controllers
- Text alignment controls
Exceptions

Semantic content attributes

**UISemanticContentAttribute.Spatial**

Groups of controls for manipulating objects or directional input on the screen

- Game controllers
- Text alignment controls
Exceptions

Semantic content attributes
Exceptions
Semantic content attributes

UISemanticContentAttribute.ForceLeftToRight
  .ForceRightToLeft
Exceptions

Semantic content attributes

UISemanticContentAttribute.ForceLeftToRight
  .ForceRightToLeft

Explicitly set the layout direction you want
Exceptions

Semantic content attributes

`UISemanticContentAttribute.ForceLeftToRight`

`ForceRightToLeft`

Explicitly set the layout direction you want

Only `ForceRightToLeft` affects layout in left-to-right localizations
Exceptions
Semantic content attributes

UISemanticContentAttribute.ForceLeftToRight
  .ForceRightToLeft

Explicitly set the layout direction you want
Only .ForceRightToLeft affects layout in left-to-right localizations
Come talk to us in a lab if you want to use these
Best Practices
User interface and text
Right-to-Left User Interface
Right-to-Left User Interface

Use formatters for region-appropriate formatting
Right-to-Left User Interface

Use formatters for region-appropriate formatting

What’s New in Internationalization
Right-to-Left User Interface

Use formatters for region-appropriate formatting

Never use `NSLocale` or `NSBundle` for UI layout branching
Right-to-Left User Interface

Use formatters for region-appropriate formatting

Never use `NSLocale` or `NSBundle` for UI layout branching

```swift
let preferredLang = NSLocale.preferredLanguages().first!
if NSLocale.characterDirectionForLanguage(preferredLang) == .RightToLeft {
    // ...
}
```
Right-to-Left User Interface

Use formatters for region-appropriate formatting

Never use `NSLocale` or `NSBundle` for UI layout branching

```swift
    let preferredLang = NSLocale.preferredLanguages().first!
    if NSLocale.characterDirectionForLanguage(preferredLang) == .RightToLeft {
        // ...
    }
```
Right-to-Left User Interface
Right-to-Left User Interface

class func userInterfaceLayoutDirectionForSemanticContentAttribute(
    attribute: UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection
Right-to-Left User Interface

class func userInterfaceLayoutDirectionForSemanticContentAttribute(
    attribute: UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection

For custom UI layout

• Do not use to determine regional or formatting settings
Right-to-Left User Interface

class func userInterfaceLayoutDirectionForSemanticContentAttribute(
    attribute: UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection

For custom UI layout

• Do not use to determine regional or formatting settings

let semanticAttr = myView.semanticContentAttribute
let layoutDirection = UIView.userInterfaceLayoutDirectionForSemanticContentAttribute(semanticAttr)
if layoutDirection == .RightToLeft {
    // ...
}
Right-to-Left User Interface

class func userInterfaceLayoutDirectionForSemanticContentAttribute(
    attribute: UISemanticContentAttribute) -> UIUserInterfaceLayoutDirection

For custom UI layout
• Do not use to determine regional or formatting settings

let semanticAttr = myView.semanticContentAttribute
let layoutDirection = UIView.userInterfaceLayoutDirectionForSemanticContentAttribute(semanticAttr)
if layoutDirection == .RightToLeft {
    // ...
}
Right-to-Left Text
Right-to-Left Text

Leave alignment and directionality at their default values
Right-to-Left Text

Leave alignment and directionality at their default values

• Natural alignment is now default on iOS 9
Right-to-Left Text

Leave alignment and directionality at their default values

- Natural alignment is now default on iOS 9
- Natural base writing direction is default since iOS 7
Right-to-Left Text

Leave alignment and directionality at their default values

- Natural alignment is now default on iOS 9
- Natural base writing direction is default since iOS 7

Do not make layout decisions based on the alignment or writing direction
Exceptions

Images
Exceptions
Images
func imageFlippedForRightToLeftLayoutDirection() -> UIImage
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context
• Obeys the UIImageView’s semantic content attribute
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context

• Obey the `UIImageView`'s semantic content attribute

Only for directional images
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context
• Obey the UIImageView's semantic content attribute

Only for directional images
• Arrows
• Chevrons
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context

• Obey the UIImageView’s semantic content attribute

Only for directional images

• Arrows

• Chevrons
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context

• Obeys the UIImageView’s semantic content attribute

Only for directional images

• Arrows
• Chevrons
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context

- Obeys the UIImageView’s semantic content attribute

Only for directional images

- Arrows
- Chevrons
- Some UI icons
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context

- Obeys the `UIImageView`'s semantic content attribute

Only for directional images

- Arrows
- Chevrons
- Some UI icons
Exceptions
Images

```swift
func imageFlippedForRightToLeftLayoutDirection() -> UIImage
```

Horizontally flips image in a right-to-left context

- Obey the `UIImageView`'s semantic content attribute

Only for directional images

- Arrows
- Chevrons
- Some UI icons
func imageFlippedForRightToLeftLayoutDirection() -> UIImage

Horizontally flips image in a right-to-left context

- Obeys the UIImageView’s semantic content attribute

Only for directional images

- Arrows
- Chevrons
- Some UI icons
Demo

Exceptions and best practices
Conclusion
Natives of right-to-left languages expect right-to-left UI
Summary

Natives of right-to-left languages expect right-to-left UI

Perfect opportunity to add right-to-left localizations
Summary

Natives of right-to-left languages expect right-to-left UI.

Perfect opportunity to add right-to-left localizations.

API accessible to non-natives.
Summary

Natives of right-to-left languages expect right-to-left UI
Perfect opportunity to add right-to-left localizations
API accessible to non-natives
Reach millions of users in new markets
More Information

Documentation and Videos
Internationalization Guide
http://developer.apple.com/internationalization/

Technical Support
Apple Developer Forums
http://developer.apple.com/forums
# Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mysteries of Auto Layout, Part 1</td>
<td>Presidio</td>
<td>Thursday 11:00 AM</td>
</tr>
<tr>
<td>Mysteries of Auto Layout, Part 2</td>
<td>Presidio</td>
<td>Thursday 1:30 PM</td>
</tr>
<tr>
<td>What’s New in Internationalization</td>
<td>Pacific Heights</td>
<td>Friday 9:00 AM</td>
</tr>
<tr>
<td>Cocoa Touch Best Practices</td>
<td>Presidio</td>
<td>Friday 1:30 PM</td>
</tr>
<tr>
<td>Labs</td>
<td>Developer Tools</td>
<td>Now</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Interface Builder and Auto Layout Lab</td>
<td>Lab C</td>
<td></td>
</tr>
<tr>
<td>Internationalization Lab</td>
<td>Frameworks Lab A</td>
<td>Friday 11:00 AM</td>
</tr>
</tbody>
</table>