Mysteries of Auto Layout, Part 1
Session 218

Jason Yao Interface Builder Engineer
Kasia Wawer iOS Keyboards Engineer
Auto Layout
Auto Layout

- Constraints
  - Image.top = Top Layout Guide.bottom + 78
  - Image.leaning = leadingMargin + 25
  - View.centerX = centerX
  - View.top = Image.top
  - View.leaning = Image.trailing + 56
  - Label.top = View.bottom
  - Label.leaning = Image.trailing + 54
  - Label.leaning = Image.trailing + 49
  - Label.top = View.bottom + 31
  - Bottom Layout Guide.top = Label.bottom + 420
Getting Started with Auto Layout?

See previous WWDC sessions on

Outline

Part 1 — Morning
• Maintainable Layouts
• Changing Constraints
• View Sizing
• Self-Sizing Table View Cells
• Priorities
• Alignment

Part 2 — Afternoon
• The Layout Cycle
• Legacy Layout
• Constraint Creation
• Constraining Negative Space
• Unsatisfiable Constraints
• Resolving Ambiguity
Part 1 — Morning
- Maintainable Layouts
- Changing Constraints
- View Sizing
- Self-Sizing Table View Cells
- Priorities
- Alignment

Part 2 — Afternoon
- The Layout Cycle
- Legacy Layout
- Constraint Creation
- Constraining Negative Space
- Unsatisfiable Constraints
- Resolving Ambiguity
Maintainable Layouts

Mystery #1
Constraints
Spacing and alignment

Mysteries of Auto Layout
Constraints

Spacing and alignment

Mysteries of Auto Layout

Mysteries of Auto Layout
Many Constraints
Stack View

UIStackView (iOS 9) and NSStackView (OS X 10.9)
Stack View
Stack View

Built with Auto Layout
Stack View

Built with Auto Layout
Manages constraints
Stack View

Built with Auto Layout
Manages constraints
Horizontal or vertical
Stack View
Alignment

Top

Center

Bottom

Baseline
Stack View

Alignment

Fill
Leading
Center
Trailing
Stack View

Distribution

- Fill
- Fill Equally
- Fill Proportionally
- Equal Spacing
Demo

Stack View in Interface Builder
// iOS 9
UIView.animateWithDuration(1.0) { () -> Void in
    self.subviewToHide.hidden = !self.subviewToHide.hidden
}
// iOS 9
UIView.animateWithDuration(1.0) { () -> Void in
    self.subviewToHide.hidden = !self.subviewToHide.hidden
}
Animate

// iOS 9
UIView.animateWithDuration(1.0) {
    self.subviewToHide.hidden = !self.subviewToHide.hidden
}

// OS X 10.11
NSAnimationContext.runAnimationGroup({ context in
    context.duration = 1.0
    self.subviewToHide.animate().hidden = !self.subviewToHide.hidden
}, completionHandler: nil)
API

// iOS 9
class UIStackView {

var axis:UILayoutConstraintAxis
var distribution:UIStackViewDistribution
var alignment:UIStackViewAlignment
var spacing:CGFloat

func addArrangedSubview(view: UIView)

var arrangedSubviews:[UIView]
...
}

// OS X 10.11
class NSStackView {

var orientation:NSUserInterfaceOrientation
var distribution:NSStackViewDistribution
var alignment:NSLayoutAttribute
var spacing:CGFloat

func addArrangedSubview(view: NSView)

var arrangedSubviews:[NSView]
...
}
API

// iOS 9
class UIStackView {
var axis:UILayoutConstraintAxis
var distribution:UIStackViewDistribution
var alignment:UIStackViewAlignment
var spacing:CGFloat
func addArrangedSubview(view: UIView)
var arrangedSubviews:[UIView]
...
}

// OS X 10.11
class NSStackView {
var orientation:NSUserInterfaceOrientation
var distribution:NSStackViewDistribution
var alignment:NSLayoutAttribute
var spacing:CGFloat
func addArrangedSubview(view: NSView)
var arrangedSubviews:[NSView]
...
}
Stack View in Interface Builder
Stack View in Interface Builder

Easy to build
Stack View in Interface Builder

Easy to build
Easy to maintain
Stack View in Interface Builder

Easy to build
Easy to maintain
Composable Stack Views
Stack View in Interface Builder

Easy to build
Easy to maintain
Composable Stack Views
Lightweight
Before and After

Before, many constraints
Before and After

After, four constraints with Stack View
Start with Stack View, use constraints as needed
Feeding the Layout Engine
Mysteries of Auto Layout, part 1

Kasia Wawer iOS Keyboards Engineer
Getting from Constraints to Layout

Views

Constraints in IB

Constraints in VFL

Explicit constraints
Getting from Constraints to Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints

???
Getting from Constraints to Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints

Your Layout
Getting from Constraints to Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints

???

Your Layout

?
Getting from Constraints to Layout

???
Getting from Constraints to Layout
Changing Constraints

Mystery #2
Activate and Deactivate
Activate and Deactivate

Constraints find their own container
Activate and Deactivate

Constraints find their own container

Adds constraints efficiently
Activate and Deactivate

Constraints find their own container

Adds constraints efficiently

Do not need to own all views
Activate and Deactivate
Activate and Deactivate

Add and remove
Activate and Deactivate

Activate and deactivate
Things to Keep in Mind
Things to Keep in Mind

Never deactivate `self.view.constraints`
Things to Keep in Mind

Never deactivate `self.view.constraints`

• Not all of those constraints belong to you
Things to Keep in Mind

Never deactivate `self.view.constraints`

- Not all of those constraints belong to you
- Weird things will happen
Things to Keep in Mind

Never deactivate `self.view.constraints`

- Not all of those constraints belong to you
- Weird things will happen
- Just don’t do it!
Things to Keep in Mind

Never deactivate `self.view.constraints`

- Not all of those constraints belong to you
- Weird things will happen
- Just don’t do it!

Keep references to constraints that change
Demo
Changing constraints
Changing Constraints
Changing Constraints

Never deactivate `self.view.constraints`
Changing Constraints

Never deactivate `self.view.constraints`

Keep references to constraints
Changing Constraints

Never deactivate `self.view.constraints`
Keep references to constraints
Animate changing constraints with view animation
Building the Layout

Views

Constraints in IB

Constraints in VFL

Explicit constraints

Layout

Engine
View Sizing
Mystery #3
View Size
Intrinsic content size
View Size

Intrinsic content size

Certain views have an `intrinsicContentSize`

• For instance—labels and image views
View Size
Intrinsic content size

Certain views have an `intrinsicContentSize`

- For instance—labels and image views

Size derived from non-constraint internals
View Size
Intrinsic content size

Certain views have an intrinsicContentSize

- For instance—labels and image views

Size derived from non-constraint internals
System makes the size constraints
Intrinsic content size

Certain views have an `intrinsicContentSize`
- For instance—labels and image views
Size derived from non-constraint internals
System makes the size constraints
Layout size is not guaranteed
View Size
Defining a particular view size
View Size

Defining a particular view size

Use constraints first
View Size

Defining a particular view size

Use constraints first

Override `intrinsicContentSize` for specific reasons
View Size

Defining a particular view size

Use constraints first

Override \texttt{intrinsicContentSize} for specific reasons

\begin{itemize}
  \item If size information does not come from constraints
\end{itemize}
View Size
Defining a particular view size

Use constraints first

Override `intrinsicContentSize` for specific reasons

• If size information does not come from constraints
• If view has custom drawing (sometimes)
View Size
Defining a particular view size

Use constraints first

Override `intrinsicContentSize` for specific reasons

- If size information does not come from constraints
- If view has custom drawing (sometimes)
- You will be responsible for invalidating
View Size

Defining a particular view size

Use constraints first

Override `intrinsicContentSize` for specific reasons

- If size information does not come from constraints
- If view has custom drawing (sometimes)
- You will be responsible for invalidating

Size can change with size class changes
Example
Example

widthConstraint = 
NSLayoutConstraint(item: imageView,
    attribute: .Width,
    relatedBy: .Equal,
    toItem: self.view,
    attribute: .Width,
    multiplier: 0.75, constant: 0.0)
Example

```swift
widthConstraint =
NSLayoutConstraint(item: imageView,
    attribute: .Width,
    relatedBy: .Equal,
    toItem: self.view,
    attribute: .Width,
    multiplier: 0.75, constant: 0.0)
```
Example

widthConstraint = NSLayoutConstraint(item: imageView, attribute: .Width, relatedBy: .Equal, toItem: self.view, attribute: .Width, multiplier: 0.75, constant: 0.0)

heightConstraint = NSLayoutConstraint(item: imageView, attribute: .Height, relatedBy: .Equal, toItem: imageView, attribute: .Width, multiplier: 1.5, constant: 0.0)
Example

widthConstraint = NSLayoutConstraint(item: imageView,
 attribute: .Width,
 relatedBy: .Equal,
 toItem: self.view,
 attribute: .Width,
 multiplier: 0.75, constant: 0.0)

heightConstraint = NSLayoutConstraint(item: imageView,
 attribute: .Height,
 relatedBy: .Equal,
 toItem: imageView,
 attribute: .Width,
 multiplier: 1.5, constant: 0.0)
Self-Sizing Table View Cells

Mystery #4
Self-Sizing Table View Cells

The start of another amazing sunset. Makes me want to move to the coast!
Self-Sizing Table View Cells

Self-sizing needs size from constraints

The start of another amazing sunset. Makes me want to move to the coast!
Self-Sizing Table View Cells

Self-sizing needs size from constraints
Width is defined with table view cells
Self-Sizing Table View Cells

Self-sizing needs size from constraints
Width is defined with table view cells
Constraints must determine height

The start of another amazing sunset. Makes me want to move to the coast!
Self-Sizing Table View Cells

Self-sizing needs size from constraints
Width is defined with table view cells
Constraints must determine height
• Take advantage of proportions

The start of another amazing sunset. Makes me want to move to the coast!
Self-Sizing Table View Cells

Self-sizing needs size from constraints
Width is defined with table view cells
Constraints must determine height
• Take advantage of proportions

The start of another amazing sunset. Makes me want to move to the coast!
Self-Sizing Table View Cells

Self-sizing needs size from constraints
Width is defined with table view cells
Constraints must determine height
• Take advantage of proportions

The start of another amazing sunset. Makes me want to move to the coast!
Self-Sizing Table View Cells

Self-sizing needs size from constraints
Width is defined with table view cells
Constraints must determine height
• Take advantage of proportions

Isn't this view incredible? I wanted to stay until the stars came out, but Kris forgot that it gets cold here and didn't bring a sweatshirt, so we had to leave before it got too dark. :P
Demo
Self-sizing Table View Cells
View Sizing
View Sizing

Certain views have an *intrinsicContentSize*.
View Sizing

Certain views have an `intrinsicContentSize`. Constraints should define size when possible.
View Sizing

Certain views have an intrinsicContentSize
Constraints should define size when possible
For self-sizing views, define size fully in constraints
Building the Layout

- Sizing constraints
- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints

Layout Engine
Building the Layout

Sizing constraints

Views
Constraints in IB
Constraints in VFL
Explicit constraints

Layout Engine

intrinsicContentSize
Priorities

Mystery #5
Ambiguity

Why does it happen?
Ambiguity

Why does it happen?

More than one layout solution
Ambiguity

Why does it happen?

More than one layout solution
- Not enough constraints
Ambiguity
Why does it happen?

More than one layout solution
• Not enough constraints
Ambiguity

Why does it happen?

More than one layout solution
- Not enough constraints
Ambiguity
Why does it happen?

More than one layout solution
- Not enough constraints
- Equal, non-required priorities
Ambiguity

Why does it happen?

More than one layout solution
- Not enough constraints
- Equal, non-required priorities

The engine makes a choice
Constraint Priorities
Constraint Priorities

Priorities go from 1–1000
Constraint Priorities

Priorities go from 1–1000
Required is 1000
DefaultHigh is 750
DefaultLow is 250
Constraint Priorities

Priorities go from 1–1000
Required is 1000
DefaultHigh is 750
DefaultLow is 250

@"H:|--[image]|"
@"V:|--[image]|[caption(==image@751)]|--"
Constraint Priorities

Priorities go from 1–1000
Required is 1000
DefaultHigh is 750
DefaultLow is 250

@"H:|-[image]-|"
@"V:|-[image]-[caption(==image@751)]-|"

Equal Widths Constraint

<table>
<thead>
<tr>
<th>First Item</th>
<th>Description</th>
<th>Label</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation</td>
<td>Equal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Second Item</td>
<td>Story Image</td>
<td>Width</td>
<td></td>
</tr>
<tr>
<td>Constant</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiplier</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Constraint Priorities

Priorities go from 1–1000
Required is 1000
DefaultHigh is 750
DefaultLow is 250

widthConstraint.priority = UILayoutPriorityDefaultHigh + 10;
Constraint Priorities

Priorities go from 1–1000
Required is 1000
DefaultHigh is 750
DefaultLow is 250
Highest priority wins

```swift
widthConstraint.priority = UILayoutPriorityDefaultHigh + 10;
```

```
@"H:|-[image]-|
@"V:|-[image]-[caption(==image@751)]-|
```
Constraint Priorities

Priorities go from 1–1000
Required is 1000
DefaultHigh is 750
DefaultLow is 250
Highest priority wins
System uses some priorities
• Set around, not equal to

```
widthConstraint.priority = UILayoutPriorityDefaultHigh + 10;
```

@"H:|-[image]-|
@"V:|-[image]-[caption(==image@751)]-|

Content Priorities
Content Priorities

How a view handles its content
Content Priorities

How a view handles its content

By default, these are not set as required

- Do not set as required
- Can cause unsatisfiable constraints
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Content Priorities

How a view handles its content
By default, these are not set as required
  • Do not set as required
  • Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
  • Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity

Types
• Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
• Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity

Types
• Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
• Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints

Equal priorities can cause ambiguity
Types
• Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
• Content hugging

Content hugging priorities

answer

send answer

250

251
Content Priorities

How a view handles its content

By default, these are not set as required

• Do not set as required
• Can cause unsatisfiable constraints

Equal priorities can cause ambiguity

Types

• Content hugging
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
• Content hugging
Content Priorities

How a view handles its content

By default, these are not set as required

• Do not set as required
• Can cause unsatisfiable constraints

Equal priorities can cause ambiguity

Types

• Content hugging
• Compression resistance
Content Priorities

How a view handles its content

By default, these are not set as required

• Do not set as required
• Can cause unsatisfiable constraints

Equal priorities can cause ambiguity

Types

• Content hugging
• Compression resistance
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
• Content hugging
• Compression resistance
Content Priorities

How a view handles its content
By default, these are not set as required
• Do not set as required
• Can cause unsatisfiable constraints
Equal priorities can cause ambiguity
Types
• Content hugging
• Compression resistance
Priorities
Priorities

Can help keep constraints from unsatisfiability
Priorities

Can help keep constraints from unsatisfiability

• But look out for competing priorities!
Priorities

Can help keep constraints from unsatisfiability

• But look out for competing priorities!

Results are more consistent
Priorities

Can help keep constraints from unsatisfiability
• But look out for competing priorities!

Results are more consistent

Use content priorities to get to the right layout
Priorities

Can help keep constraints from unsatisfiability
• But look out for competing priorities!

Results are more consistent

Use content priorities to get to the right layout
• Hugging priorities hug content
Priorities

Can help keep constraints from unsatisfiability
• But look out for competing priorities!

Results are more consistent

Use content priorities to get to the right layout
• Hugging priorities hug content
• Compression resistance resists squishing
Building the Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints
- Sizing constraints
- intrinsicContentSize
Building the Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints
- Sizing constraints
- Constraint priorities
- Content priorities
- intrinsicContentSize
Alignment
Mystery #6
Aligning Baselines
Aligning Baselines

Use `firstBaseline` and `lastBaseline`
Aligning Baselines

Use firstBaseline and lastBaseline
Aligns text better than top or bottom
Aligning Baselines

Use `firstBaseline` and `lastBaseline`.

Aligns text better than top or bottom.

Better control over changing views.
Aligning Baselines

Use `firstBaseline` and `lastBaseline`.

Aligns text better than top or bottom.

Better control over changing views.

Label aligned to button by bottom.
Aligning Baselines

Use `firstBaseline` and `lastBaseline`.
Aligns text better than top or bottom.
Better control over changing views.

Label aligned to button by bottom
Add second line of text and a third while we’re at it.
Aligning Baselines

Use `firstBaseline` and `lastBaseline`
Aligns text better than top or bottom
Better control over changing views

Label aligned to button by bottom
Add second line of text and a third while we’re at it
Aligning Baselines

Use `firstBaseline` and `lastBaseline` to align text better than top or bottom.
Better control over changing views.

Label aligned to button by bottom:
Add second line of text and a third while we’re at it.
Aligning Baselines

Use `firstBaseline` and `lastBaseline`

Aligns text better than top or bottom

Better control over changing views

Label aligned to button by bottom

Add second line of text

and a third while we’re at it
Aligning Baselines

Use `firstBaseline` and `lastBaseline`  
Aligns text better than top or bottom  
Better control over changing views  

Label aligned to button by bottom  
Add second line of text  
and a third while we’re at it  

Button
One of the best photos I took on the trip.

I love how the water looks like mist!
Leading and Trailing

Use leading/trailing instead of left/right
Helps with prep for localization

One of the best photos I took on the trip.

I love how the water looks like mist!
Leading and Trailing

Use leading/trailing instead of left/right
Helps with prep for localization
Leading and Trailing

Use leading/trailing instead of left/right
Helps with prep for localization
Alignment Rects
Alignment Rects

Usually (not always) same as frame

Tap to choose character
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Override `alignmentRectInsets` if needed
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Override `alignmentRectInsets` if needed
Find out the calculated rects
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Override `alignmentRectInsets` if needed

Find out the calculated rects
• Use Show Alignment Rectangles in Debug menu
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Override `alignmentRectInsets` if needed
Find out the calculated rects
- Use Show Alignment Rectangles in Debug menu
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Override `alignmentRectInsets` if needed
Find out the calculated rects
- Use Show Alignment Rectangles in Debug menu
- Get using `alignmentRectForFrame:`
Alignment Rects

Usually (not always) same as frame
Includes the critical content only
Does not change when view is transformed
Override `alignmentRectInsets` if needed
Find out the calculated rects
  • Use Show Alignment Rectangles in Debug menu
  • Get using `alignmentRectForFrame`
More in Part 2
Alignment
Alignment

First and last baseline for better aligned text
Alignment

First and last baseline for better aligned text
Leading and trailing instead of left and right
Alignment

First and last baseline for better aligned text
Leading and trailing instead of left and right
Override `alignmentRectInsets` to adjust alignment rects
Building the Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints

Layout Engine

Sizing constraints

Constraint priorities

Content priorities

intrinsicContentSize
Building the Layout

Views

Constraints in IB

Constraints in VFL

Explicit constraints

Sizing constraints

Constraint priorities

Content priorities

Layout Engine

Horizontal alignment

Vertical alignment

Baseline alignment

intrinsicContentSize
Building the Layout

- Views
- Constraints in IB
- Constraints in VFL
- Explicit constraints
- Sizing constraints
- Constraint priorities
- Content priorities
- Alignment rect insets
  - Horizontal alignment
  - Vertical alignment
  - Baseline alignment
  - intrinsicContentSize
Building the Layout

Layout
Engine
Building the Layout
Building the Layout

Layout Engine

Alignment rects
Building the Layout

Layout Engine → Alignment rects → Your Layout
Building the Layout

Your Layout
Summary

Stack Views help build easily maintainable layouts
Summary

Stack Views help build easily maintainable layouts
Use activate and deactivate for constraints
Summary

Stack Views help build easily maintainable layouts
Use activate and deactivate for constraints
Determine size through constraints
Summary

Stack Views help build easily maintainable layouts
Use activate and deactivate for constraints
Determine size through constraints
  • Override intrinsicContentSize judiciously
Summary

Stack Views help build easily maintainable layouts
Use activate and deactivate for constraints
Determine size through constraints
  • Override intrinsicContentSize judiciously
Use priorities to properly solve your layout
Summary

Stack Views help build easily maintainable layouts
Use activate and deactivate for constraints
Determine size through constraints
  • Override intrinsicContentSize judiciously
Use priorities to properly solve your layout
Alignment goes beyond top, bottom, and center
Summary

Stack Views help build easily maintainable layouts
Use activate and deactivate for constraints
Determine size through constraints
  • Override intrinsicContentSize judiciously
Use priorities to properly solve your layout
Alignment goes beyond top, bottom, and center
  • Keep localization in mind
More Information

Documentation and Videos
Swift Language Documentation
http://developer.apple.com/swift

Technical Support
Apple Developer Forums
http://devforums.apple.com

Sample Code
AstroLayout

General Inquiries
Paul Marcos, App Frameworks Evangelist
pmarcos@apple.com
# Related Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mysteries of Auto Layout, Part 2</td>
<td>Presidio</td>
<td>Thursday 1:30PM</td>
</tr>
<tr>
<td>What’s New in Cocoa</td>
<td>Presidio</td>
<td>Tuesday 1:30PM</td>
</tr>
<tr>
<td>What’s New in UIKit Dynamics and Visual Effects</td>
<td>Mission</td>
<td>Friday 10:00AM</td>
</tr>
<tr>
<td>Cocoa Touch Best Practices</td>
<td>Presidio</td>
<td>Friday 1:30PM</td>
</tr>
<tr>
<td>What’s New in Internationalization</td>
<td>Pacific Heights</td>
<td>Friday 9:00AM</td>
</tr>
<tr>
<td>New UIKit Support for International User Interfaces</td>
<td>Nob Hill</td>
<td>Thursday 2:30PM</td>
</tr>
<tr>
<td>Lab Name</td>
<td>Room</td>
<td>Time</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>Interface Builder and Auto Layout Lab</td>
<td>Lab C</td>
<td>Thursday 2:30PM</td>
</tr>
</tbody>
</table>