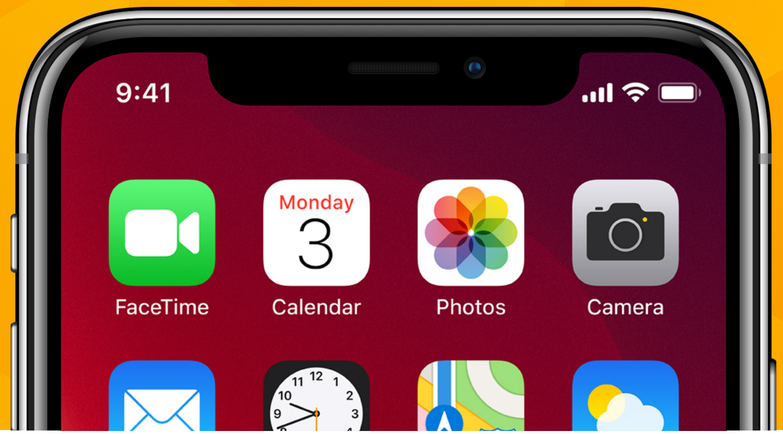


iOS Application Development

Lecture 6: ScrollViews and TableViews

Prof. Dr. Jan Borchers
Media Computing Group
RWTH Aachen University

WS ’22/’23 • hci.rwth-aachen.de/ios

RWTH AACHEN UNIVERSITY

Recap

- App and ViewController Lifecycle
- AutoLayout
- Protocols
- Extensions

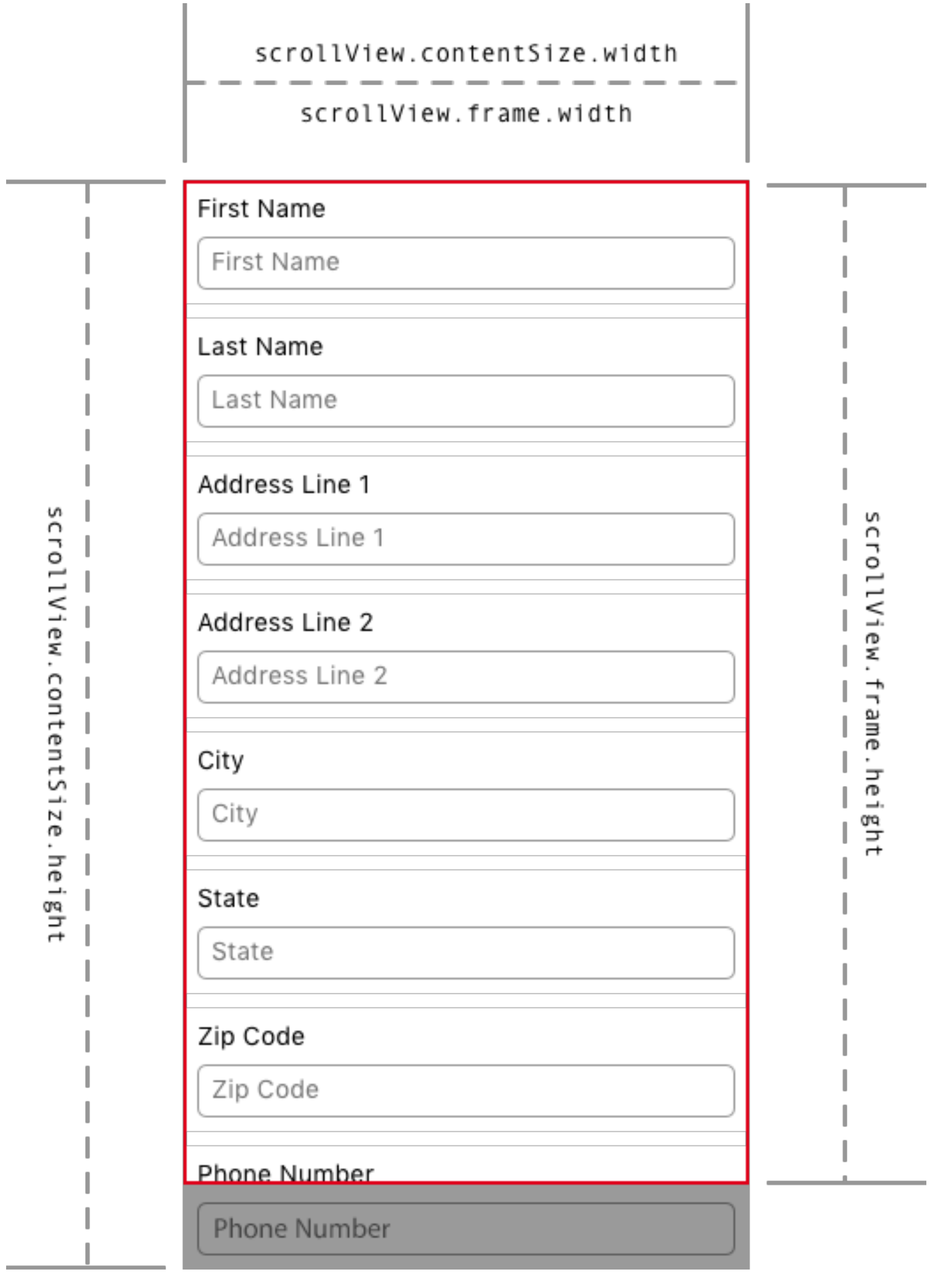
RWTH AACHEN UNIVERSITY

Scroll Views

RWTH AACHEN UNIVERSITY

ScrollView

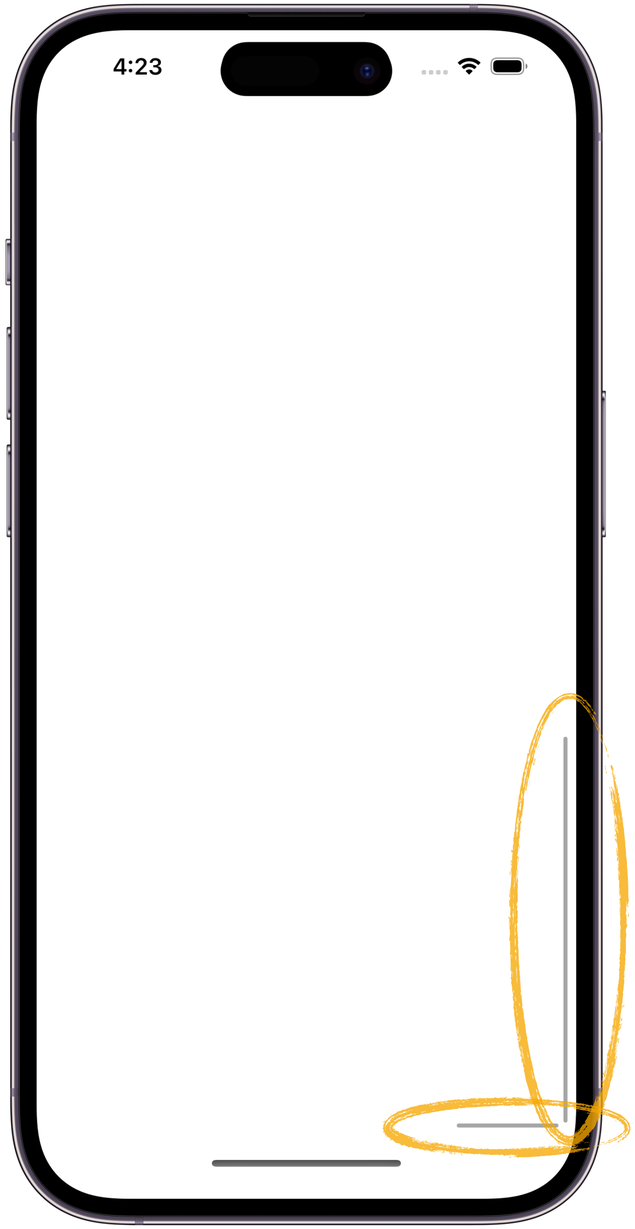
- UIScrollView
  - Parent of UITableView
- Show content that does not fit on one screen
- `.frame` property
  - Where on the screen is the ScrollView?
- `.contentSize` property
  - How large is the scrollable area?



RWTH AACHEN UNIVERSITY

Setting the contentSize

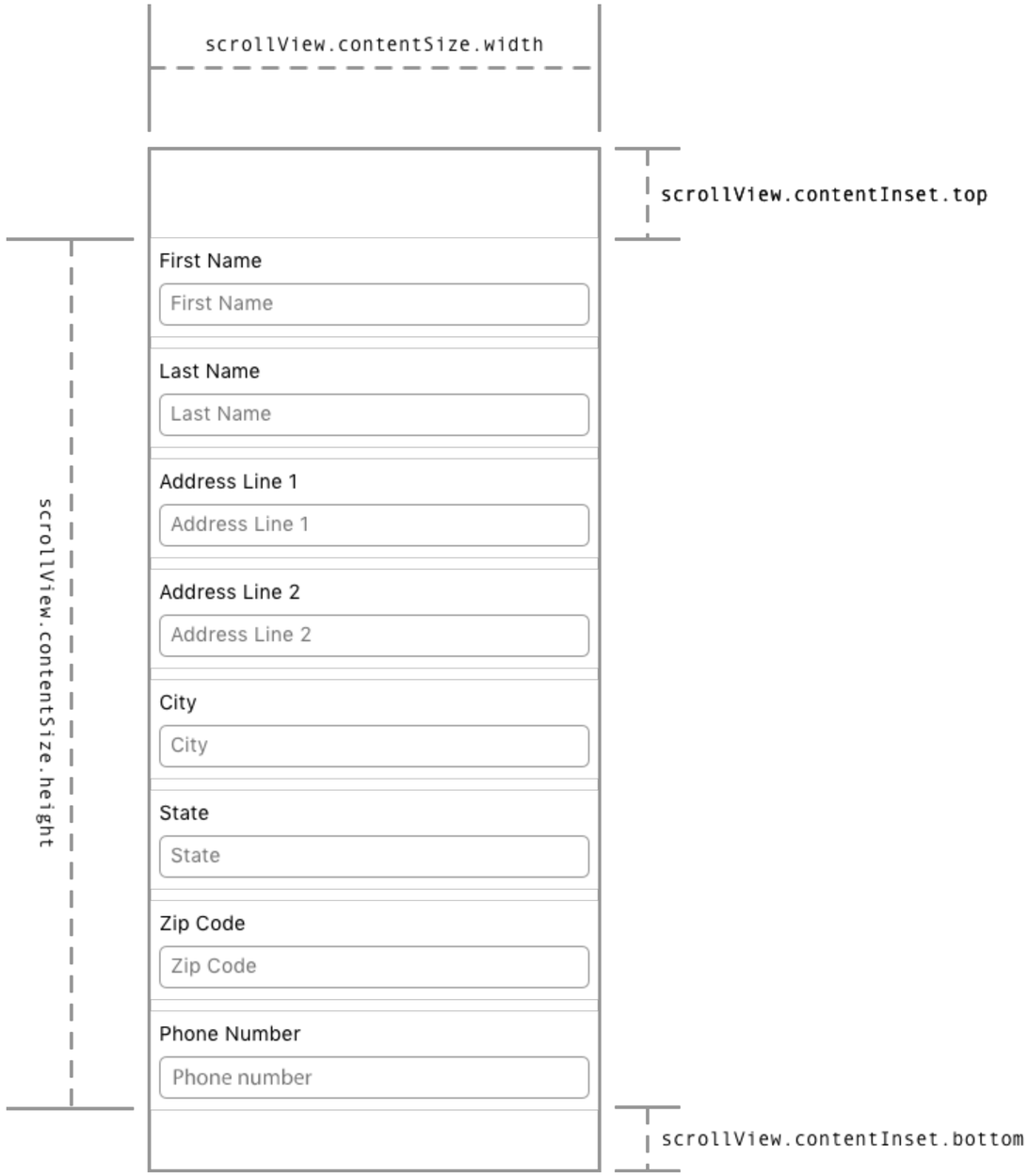
```
let aView = UIView.init(frame: CGRect.init(
  x: 0, y: 0, width: 2000, height: 2000))

self.theScrollView.addSubview(aView)
self.theScrollView.contentSize = aView.frame.size
```



RWTH AACHEN UNIVERSITY

Content Insets

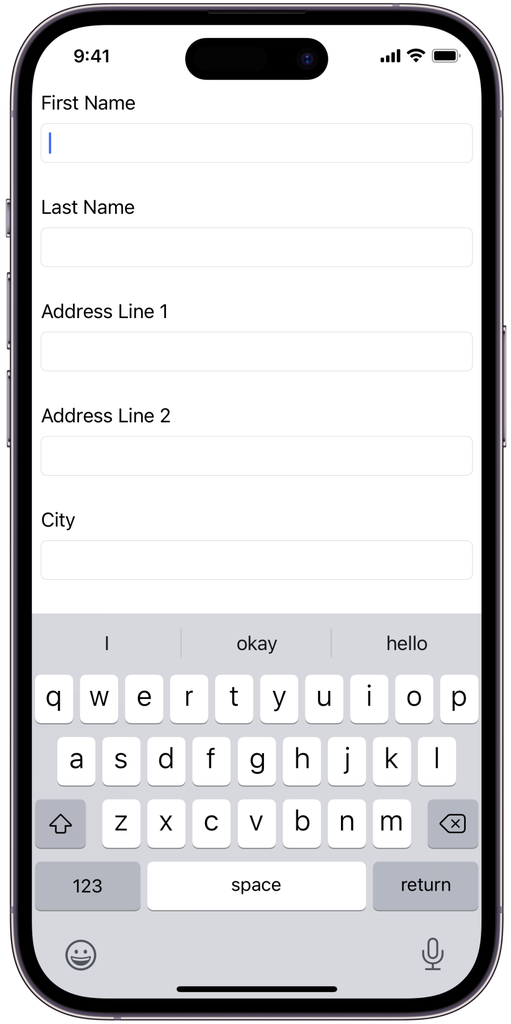
- Use insets to provide padding to your content
  - e.g., when showing toolbars or on the bottom displaying the keyboard



RWTH AACHEN UNIVERSITY

ScrollView Content Insets & Keyboard Events

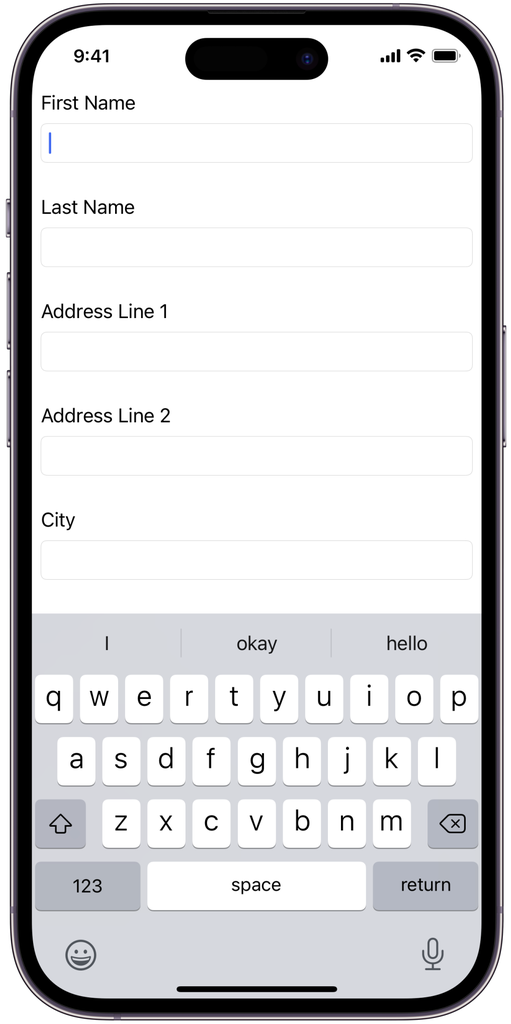
```
func registerForKeyboardNotifications() {
    NotificationCenter.default.addObserver(self, selector: #selector(keyboardWasShown(_:)),
      name: UIResponder.keyboardDidShowNotification, object: nil)

    NotificationCenter.default.addObserver(self, selector: #selector(keyboardWillBeHidden(_:)),
      name: UIResponder.keyboardWillHideNotification, object: nil)
}
```



RWTH AACHEN UNIVERSITY

ScrollView Content Insets for Keyboard Events

```swift
func keyboardWasShown(_ notificiation: NSNotification) {
    //Get keyboard size
    guard let info = notificiation.userInfo,
        let keyboardFrameValue = info[UIResponder.keyboardFrameBeginUserInfoKey] as? NSValue
          else { return }

    let keyboardFrame = keyboardFrameValue.cgRectValue
    let keyboardSize = keyboardFrame.size

    //Set insets with keyboard size
    let contentInsets = UIEdgeInsets(top: 0.0, left: 0.0, bottom: keyboardSize.height,
      right: 0.0)
    scrollView.contentInset = contentInsets
    scrollView.scrollIndicatorInsets = contentInsets
}

func keyboardWillBeHidden(_ notification: NSNotification) {
    //Reset the insets to be 0 for each edge
    let contentInsets = UIEdgeInsets.zero
    scrollView.contentInset = contentInsets
    scrollView.scrollIndicatorInsets = contentInsets
}
```



RWTH AACHEN UNIVERSITY

UIScrollViewDelegate

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {}

func viewForZooming(in scrollView: UIScrollView) -> UIView? {}

func scrollViewDidEndScrollingAnimation(_ scrollView: UIScrollView) {}

func scrollViewWillBeginDragging(_ scrollView: UIScrollView) {}

func scrollViewWillBeginDecelerating(_ scrollView: UIScrollView) {}

...
```


RWTH AACHEN UNIVERSITY

Scroll View Demo

RWTH AACHEN UNIVERSITY

Table View

RWTH AACHEN UNIVERSITY

Table Views

- Subclasses UIScrollView
- Displays a vertical list of items
- Provides customizable options

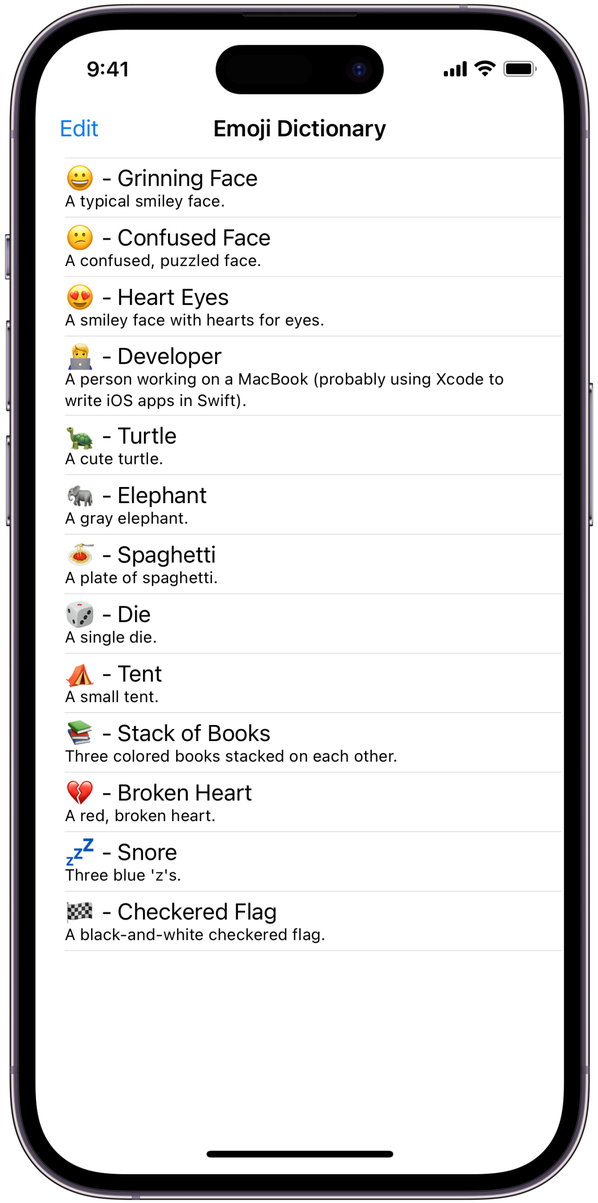
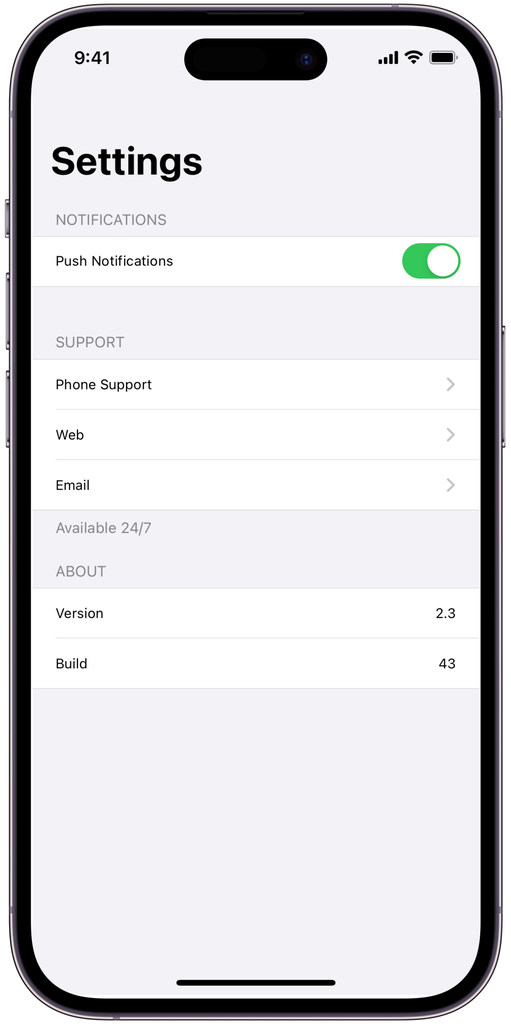
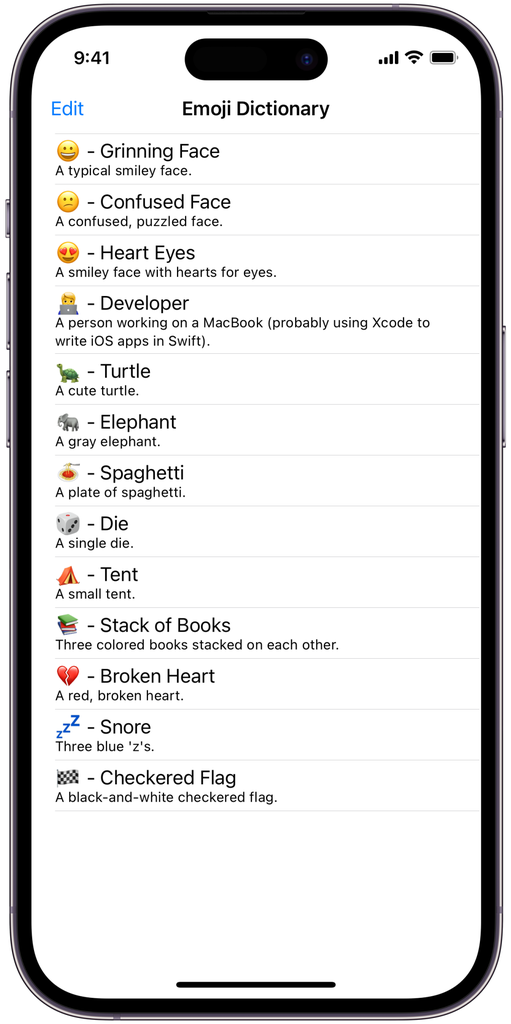


Table View Types



Static

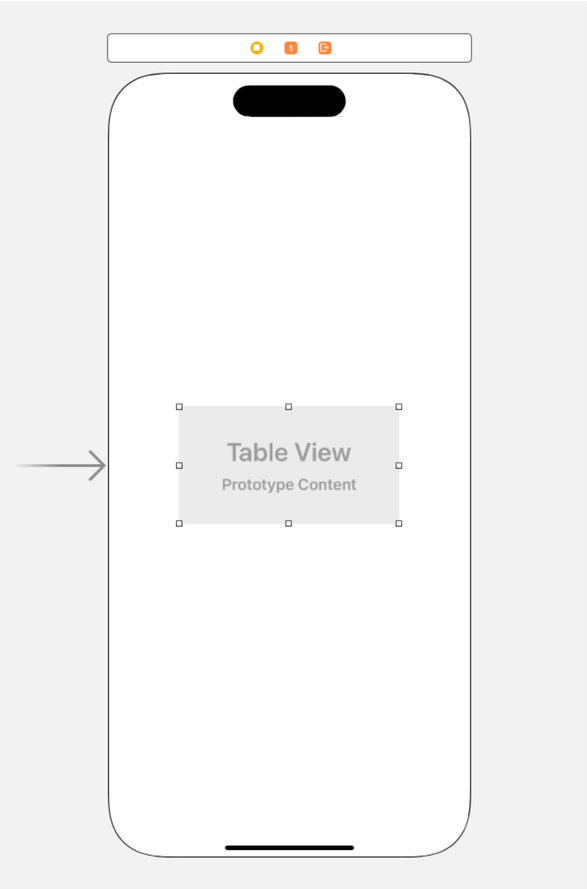


Dynamic

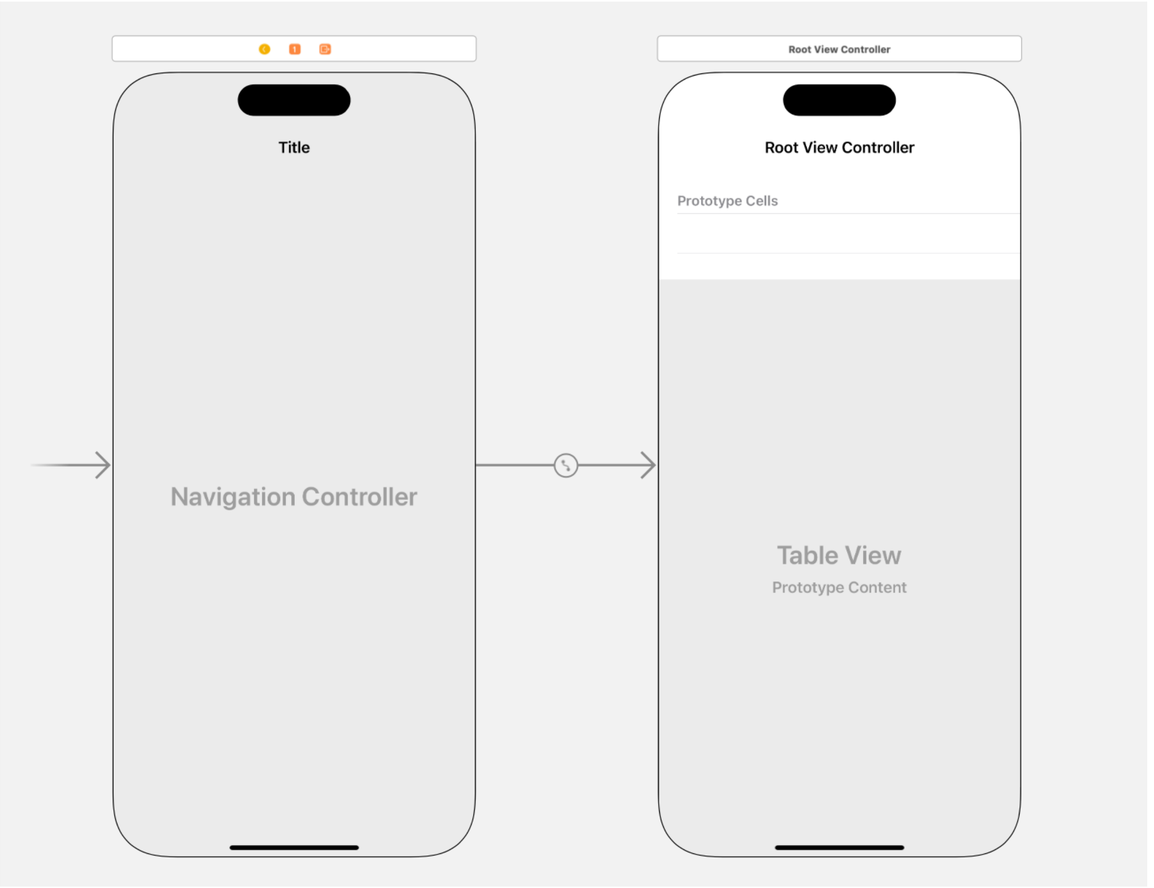
RWTH AACHEN UNIVERSITY

Adding Table Views

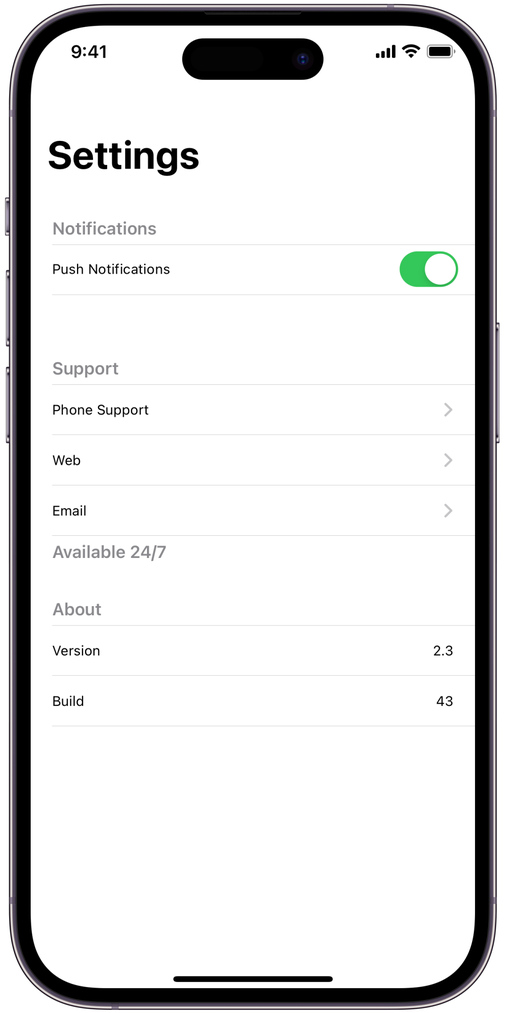
Add TableView to existing ViewController



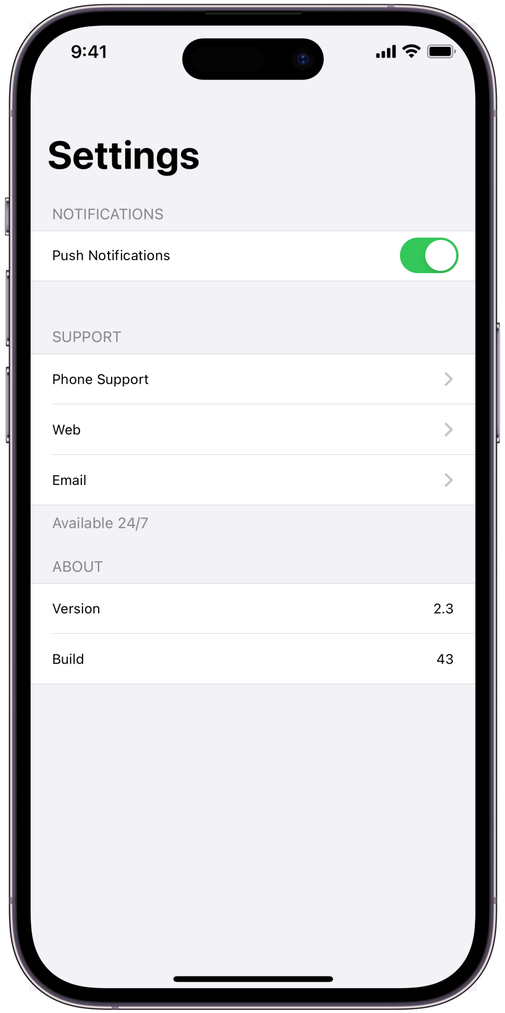
Add a TableViewController



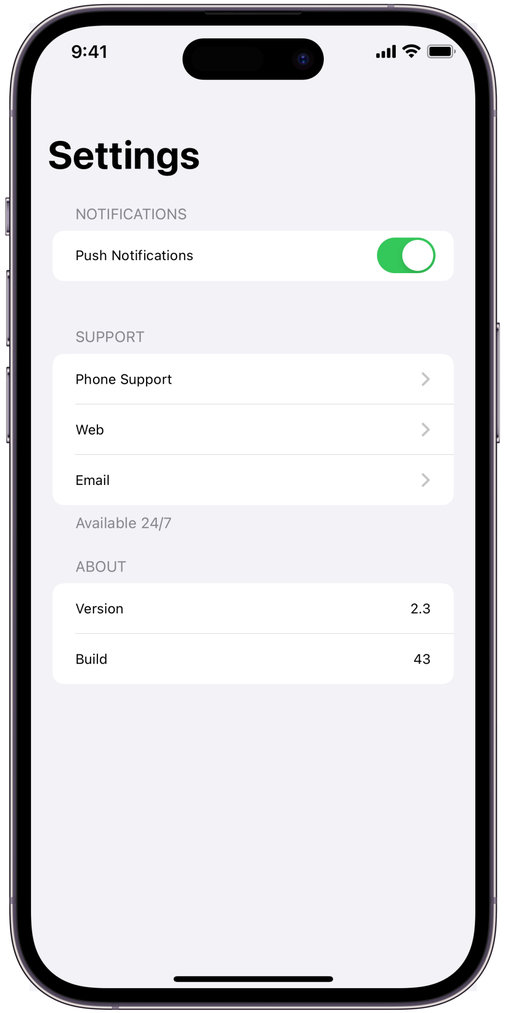
TableView Styles



Plain



Grouped



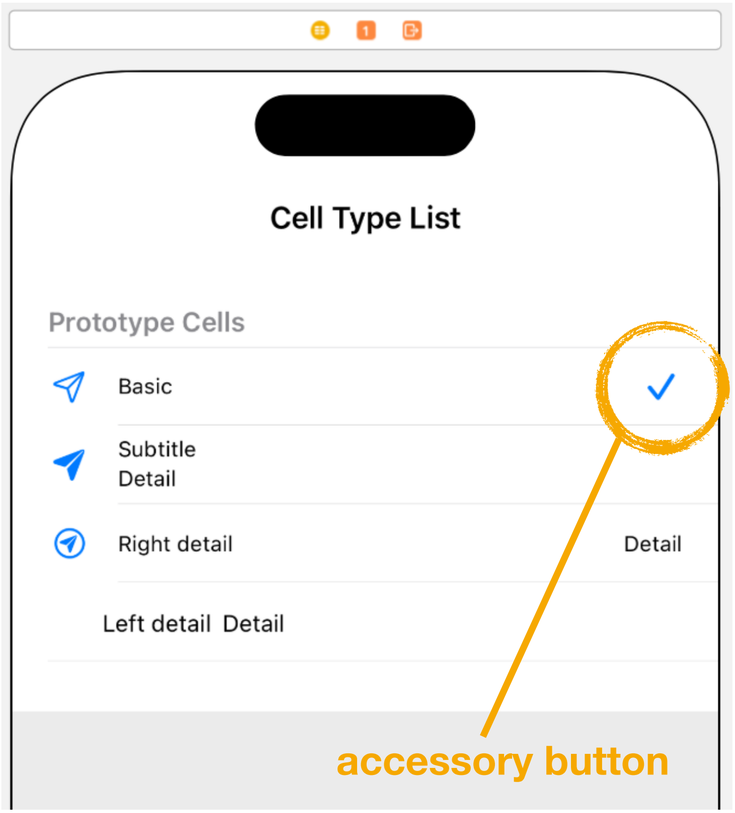
Inset Grouped

RWTH AACHEN UNIVERSITY

Cell Styles and Accessory Buttons

Storyboard name	Programmatic enum name	Displays
Basic	`.default`	`textLabel`, `imageView`
Subtitle	`.subtitle`	`textlabel` , `detailTextLabel`, `imageView`
Right detail	`.value1`	`textlabel` , `detailTextLabel`, `imageView`
Left detail	`.value2`	`textLabel` , `detailTextLabel`

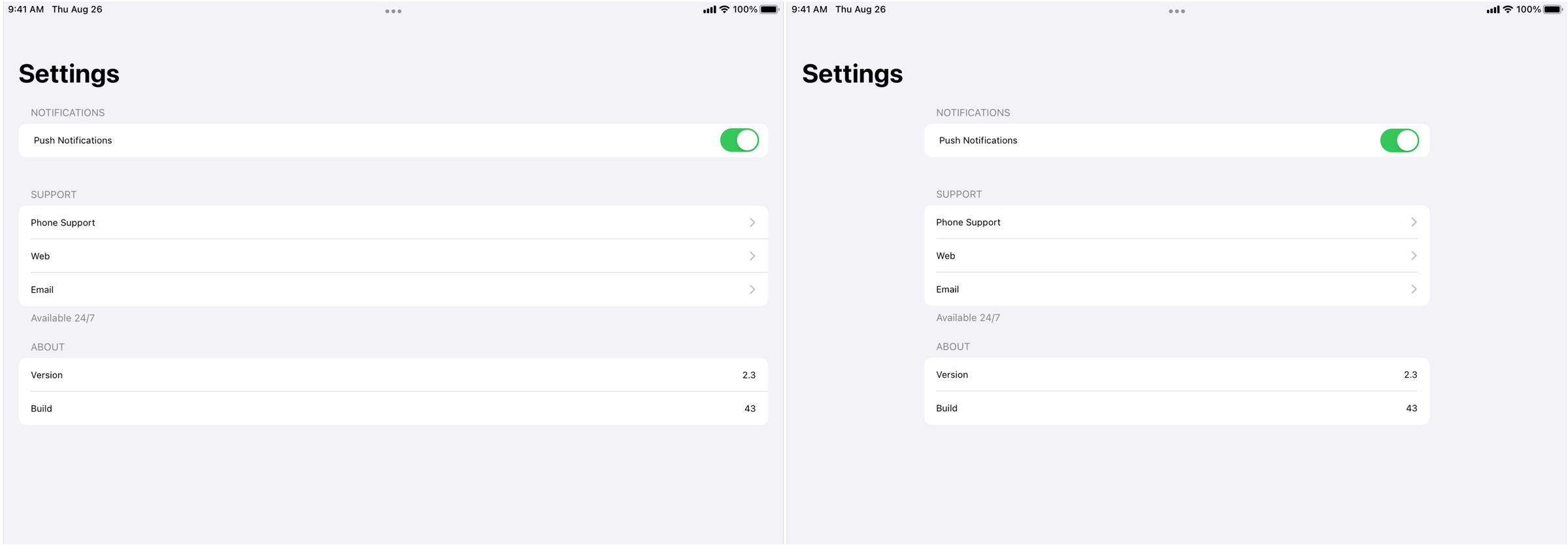
```
var content = cell.defaultContentConfiguration()
content.text = "😀" + "Grinning Face"
content.secondaryText = "A typical smiley face."
cell.contentConfiguration = content
```



- You can add accessory buttons for all styles in the Interface Builder

Readability Margins

```
tableView.cellLayoutMarginsFollowReadableWidth = true
```

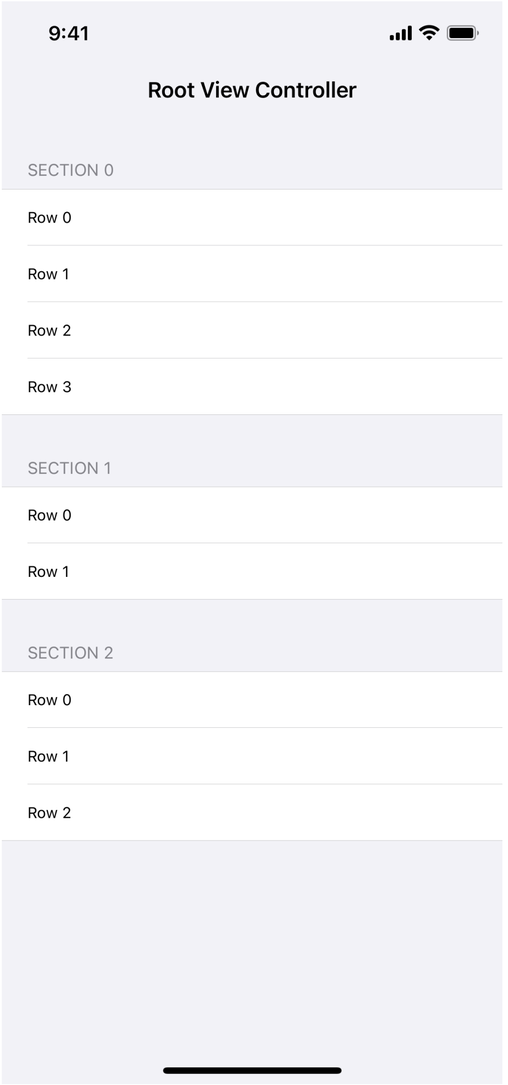


Index Paths and Reusable Cells

- TableViews consist of sections and rows you can access with `indexPath.row & indexPath.section`

```
let cell: UITableViewCell = tableView.dequeueReusableCell(withIdentifier: "Cell",
  for: indexPath)
```

  - Only visible cells are loaded to save memory
    - Allows for smooth scrolling



TableView Protocols & Reloading

- UITableViewDataSource: Provides data for the TableView

```
//Set the number of sections. Assumes 1 if not overridden
optional func numberOfSections(in tableView: UITableView) -> Int {}

//Set the number of rows per section
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {}

//Set the content of the cells
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {}
```

- UITableViewDelegate (optional): Set appearance and behavior

```
//React to clicks on the row
func tableView(_:didSelectRowAt:)

//React to taps on the accessory button
func tableView(_:accessoryButtonTappedForRowWith:)
```

- Use `reloadData()` to refresh the TableView

RWTH AACHEN UNIVERSITY

Table View Demo

RWTH AACHEN UNIVERSITY

Summary

- ScrollViews
  - To show content that does not fit on one screen
  - .contentSize and .frame
- TableView
  - Add a view or a TableViewViewController
  - Different types and cell types
  - Cells are reused for dynamic TableViews
  - Check out the demo in Lesson 1.6 in the Data Collection book

RWTH AACHEN UNIVERSITY