TableViews

Ana Nogal
@anainogal
@iOSStepByStep
http://ananogal.com

Agenda

What are we learning today?

- ***Anatomy of a Table View***
- ***Index Paths and Arrays***
- ***Table View Protocols***
- ***Custom TableView Cells***

TableViews

Overview:

- Are the most used control in iOS.
- Display data objects
- Present data as a single column list
- Can present rows in sections or groups

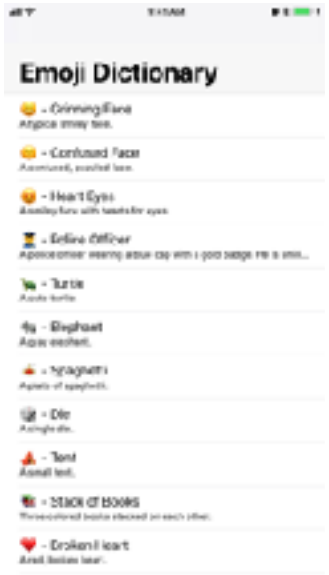
TableViews

How to use them

- Can add a tableView to a ViewController
- Create a TableViewController

TableViews - Type

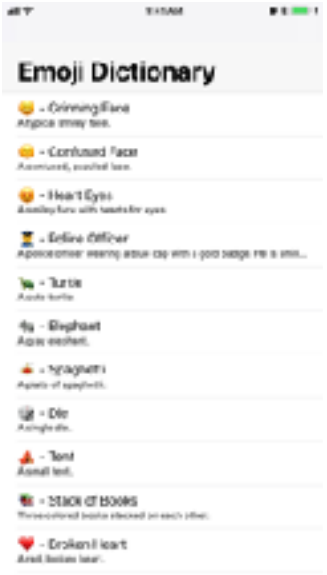
Dynamic



Static

TableViews - Style

Plain



Grouped

TableViews

TableViewCell

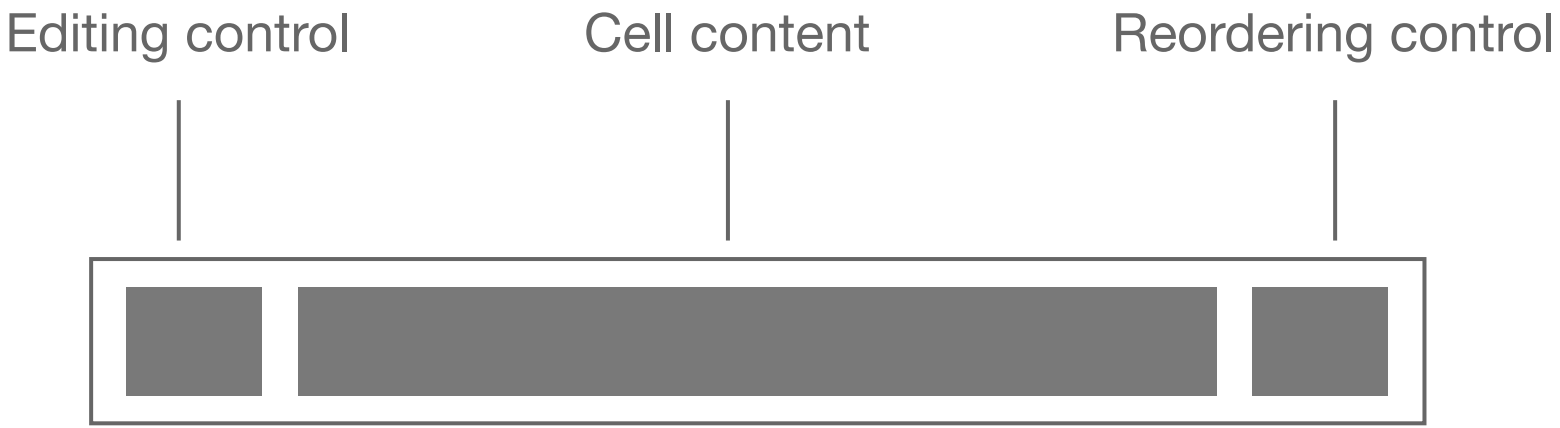
- They represent a row in the list
- They are reusable views that can display text, images or even other views.

TableViews

Every row is represented with a table view cell

TableViews

In editing mode, the cell content shrinks



TableViews

UITableViewCell properties

- textLabel - UILabel for the title
- detailTextLabel - UILabel for the subtitle
- imageView - UIImageView for the an image

TableViews

UITableViewCellStyle

- **Basic** - Displays a ***textLabel*** and a ***imageView***
- **Subtitle** - Displays a ***textLabel***, a ***detailTextLabel*** and ***imageView***
- **Right detail** - Displays a ***textLabel***, a ***detailTextLabel*** and a ***imageView***
- **Left detail** - Displays a ***textLabel*** and a ***detailTextLabel***

TableViews

IndexPath

- Has two properties: ***row*** and ***section***

Arrays

- TableViews display collections of items and an array is the most common used.

TableViews

Cell Dequeuing

It’s the process of reusing a cell that is no longer being used in the TableView.

Use the cell identifier to ensure you are dequeuing the same type of cell.

TableViews

TableView DataSource

```
func numberOfSections(in tableView: UITableView) -> Int {}

func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {}

func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {}
```

https://developer.apple.com/documentation/uikit/uitableviewdatasource

TableViews

TableView Delegate

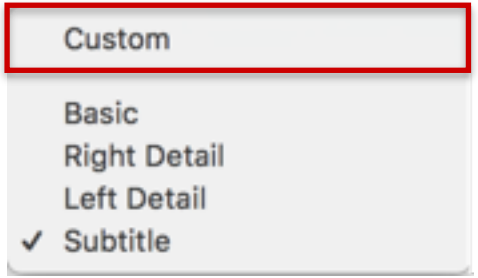
```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath)
func tableView(_ tableView: UITableView, editingStyleForRowAt indexPath:
IndexPath) -> UITableViewCellEditingStyle {}
```

https://developer.apple.com/documentation/uikit/uitableviewdelegate

TableViews

Custom TableView Cells

When you select the tableCell and go to the Attributes Inspector panel, in Style you have an option ***Custom***



TableViews

Custom TableView Cells

A UITableViewCell inherits from UIView

```
Declaration  class UITableViewCell : UIView, NSCoding,
             UIGestureRecognizerDelegate
```

Demo

Demo

Delete ViewController

- Delete the ViewController.swift file
- Go to Main.storyboard and delete the ViewController scene.

Demo

Create a model

- Create a new Swift file and name it Emoji
- Create a class called Emoji that has a 4 String properties:
  - symbol
  - name
  - description
  - usage
- Create a memberwise initialiser

Demo

Create a TableViewController

- Go to Main.Storyboard
- From the Object Library Drag a NavigationViewController
- Select the Navigation Controller and set it as the initial ViewController

Demo

Create a EmojiTableViewController file

- Create a new Cocoa Touch file and name it **EmojiTableViewController**
- Go to Main.storyboard and set the TableViewController class to EmojiTableViewController class.

Demo

Style the cell

- Go to Main.storyboard and select the TableViewController
- Select the Navigation item and change the title to “**Emoji Dictionary**”
- Under the “**Prototype Cells**” select the tableViewCell.
- Change it’s style to **Subtitle**
- Set the ***identifier*** to “***EmojiCell***”

Demo

Add data to the tableView:

- Go to this link http://codingboard.org/boards/swiftbootcamp and copy the variable into the ***EmojiTableViewController***
- In ***tableView(_:numberOfRowsInSection:)*** return the number of elements of the array
- Find the method ***numberOfSections(in:)*** and make it return 1.

Challenge

Configure the cell:

- Go to EmojiTableViewController.swift
- Find the method ***tableView(_:cellForRowAt:)*** and uncomment It
- In the dequeue method set the parameter "***withIdentifier***" to "***EmojiCell***"
- Using the indexPath, get the emoji for the row.
- Set the ***textLabel*** property of the cell to the **symbol** + **name** of the emoji
- Set the ***detailTextLabel*** property of the cell to the emoji **description**
- Build and run the app.

Demo

Create a Custom TableView Cell:

- First go to Main.Storyboard and select the cell
- In the Attributes inspector panel, set the Style property to Custom
- Create a new Cocoa Touch Class name ***EmojiTableViewCell*** *and* set its subclass to UITableViewCell.

Demo

In the Storyboard:

- Select the TableView cell
- Add a horizontal stack view to the cell's content view
- Add constraints to the stack view to the margin of the cell
- Add a label and a vertical stack view inside the existent one.
- Set the vertical stack view's "***Distribution***" property to "***Fill Equally***"

Demo

In the Storyboard:

- Set the text of the label to a Emoji of your preference. Center it.
- Add two labels to the vertical stack view
- Change the “***Horizontal Content Hugging Priority***” of the emoji label to 252
- Select the cell and set it’s class to the ***EmojiTableViewCell***

Demo

Add Outlets to the labels:

- Select the cell, open Assistant Editor.
- Add 3 outlets to the class for each label:
  - symbolLabel
  - nameLabel
  - descriptionLabel

Challenge

In the EmojiTableViewCell.swift:

- Add a new function: ***update(emoji: Emoji)***
- In the function body write code to update the labels with the data from the emoji

In the EmojiTableViewController.swift:

- Update ***tableView(_:cellForRowAt:)*** to use your new cell.

  Tip: use if let cell = cell as? EmojiTableViewCell { }
- Build and run the app.

iOS Step by Step

Follow us on Twitter:

@iOSStepByStep

@anainogal

@Vasy_1st

Resources

RayWenderlich.com

HackingWithSwift.com

Apple Swift Book

Human Interface Guidelines iOS