ITM103 iOS Application Development

Topic 5: Table View Controller

Objectives

- By the end of the lesson, you will be able to:
  - Understand table view delegates and datasource
  - Bind data to table view
  - Manage table view events
  - Create customised table view cell
  - Self sizing cells and Dynamic Type in iOS 8
  - Edit UITableView –Adding Rows, Delete Rows, Moving Rows

Table View Controller

What is a Table View?

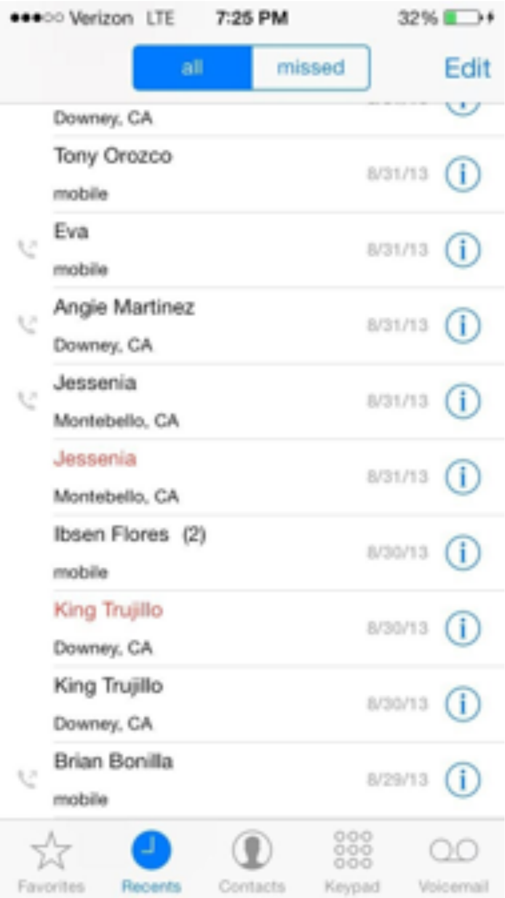
Table Views

- What is a Table View?
  - Allow users to navigate through hierarchically structured data
  - Present an indexed list of items
  - Display detail information and controls in visually distinct groupings
  - Present a selectable list of options

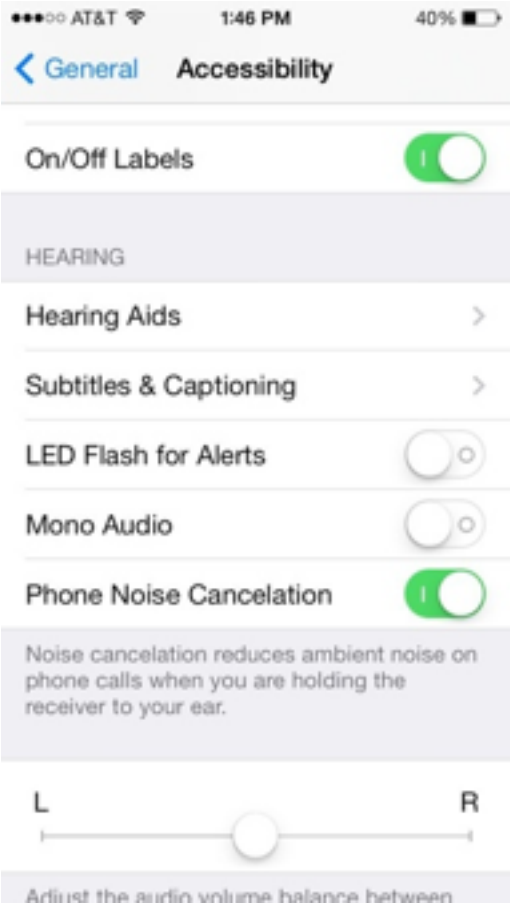
Table View Controller

Table View Styles

Table Views of Various Kinds



Downey, CA		
Tony Orozco mobile	8/31/13	ⓘ
Eva mobile	8/31/13	ⓘ
Angie Martinez Downey, CA	8/31/13	ⓘ
Jessenia Montebello, CA	8/31/13	ⓘ
Jessenia Montebello, CA	8/31/13	ⓘ
Ibsen Flores (2) mobile	8/30/13	ⓘ
King Trujillo Downey, CA	8/30/13	ⓘ
King Trujillo Downey, CA	8/30/13	ⓘ
Brian Bonilla mobile	8/29/13	ⓘ



On/Off Labels ●

HEARING

Hearing Aids >

Subtitles & Captioning >

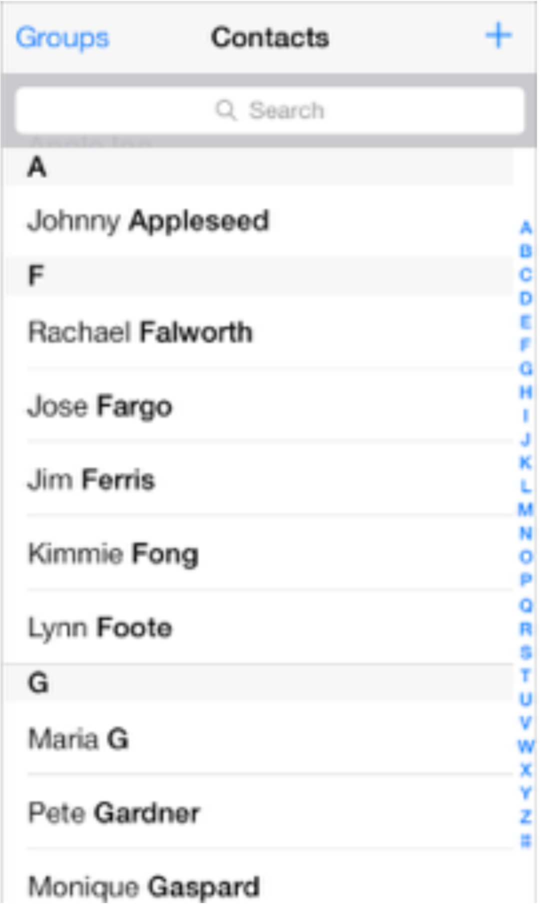
LED Flash for Alerts ○

Mono Audio ○

Phone Noise Cancelation ●

Noise cancelation reduces ambient noise on phone calls when you are holding the receiver to your ear.

Adjust the audio volume balance between



A

Johnny Appleseed

F

Rachael Falworth

Jose Fargo

Jim Ferris

Kimmie Fong

Lynn Foote

G

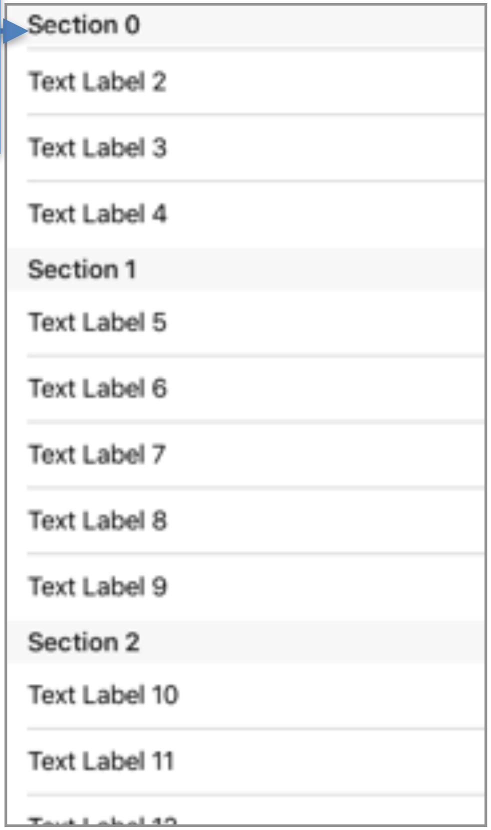
Maria G

Pete Gardner

Monique Gaspard

Table View Styles

This header floats as you scroll up



Section 0

Text Label 2

Text Label 3

Text Label 4

Section 1

Text Label 5

Text Label 6

Text Label 7

Text Label 8

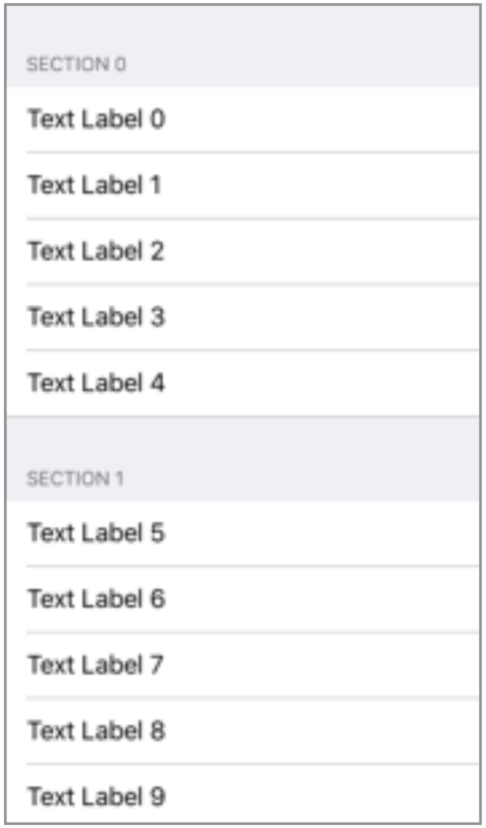
Text Label 9

Section 2

Text Label 10

Text Label 11

Text Label 12



SECTION 0

Text Label 0

Text Label 1

Text Label 2

Text Label 3

Text Label 4

SECTION 1

Text Label 5

Text Label 6

Text Label 7

Text Label 8

Text Label 9

Plain (UITableViewStyle.plain)
A plain table view. Any section headers or footers are displayed as inline separators and float when the table view is scrolled.

Grouped (UITableViewStyle.grouped)
A table view whose sections present distinct groups of rows. The section headers and footers do not float.

Table View Styles

In XCode,
after selecting a Table View,
change the Table View Style here.

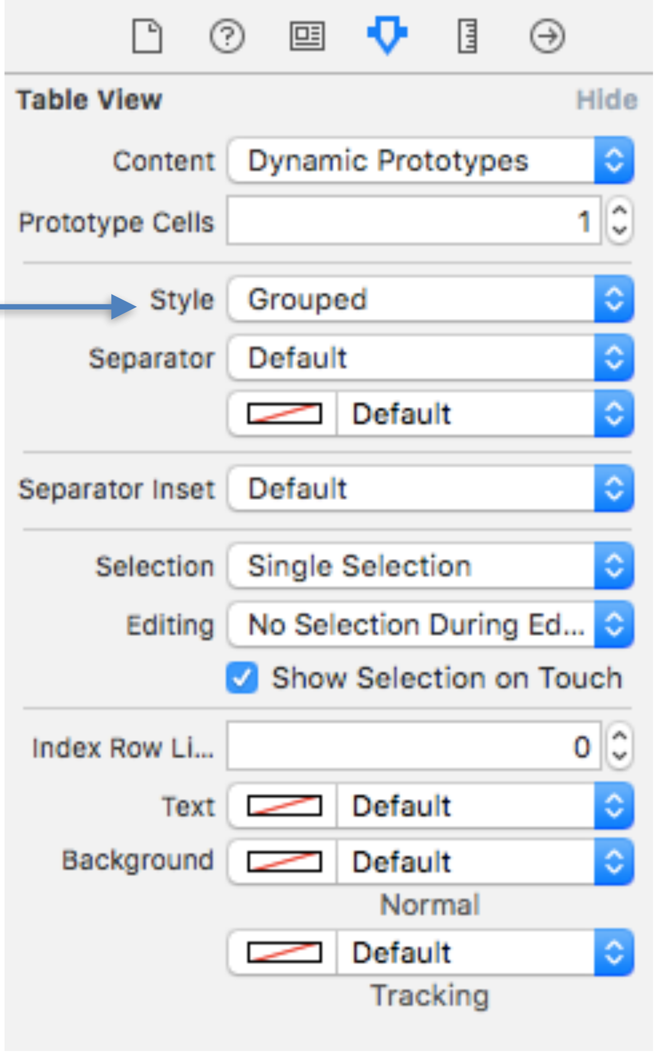


Table View Hide

Content: Dynamic Prototypes

Prototype Cells: 1

Style: Grouped

Separator: Default

Default

Separator Inset: Default

Selection: Single Selection

Editing: No Selection During Ed...

☑ Show Selection on Touch

Index Row Li...: 0

Text: Default

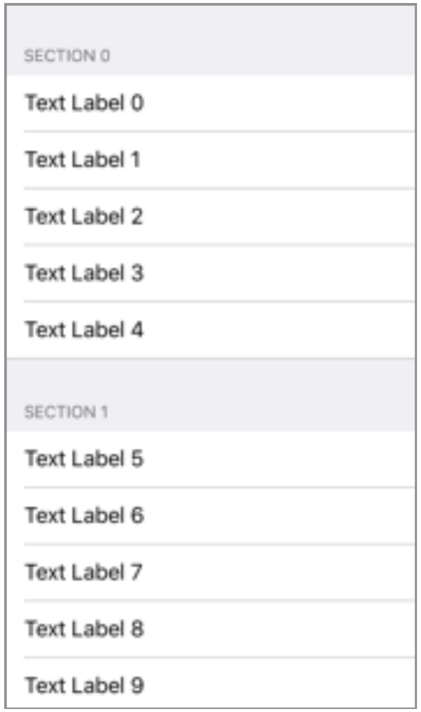
Background: Default

Normal

Default

Tracking

Table View Cell Styles



SECTION 0

Text Label 0

Text Label 1

Text Label 2

Text Label 3

Text Label 4

SECTION 1

Text Label 5

Text Label 6

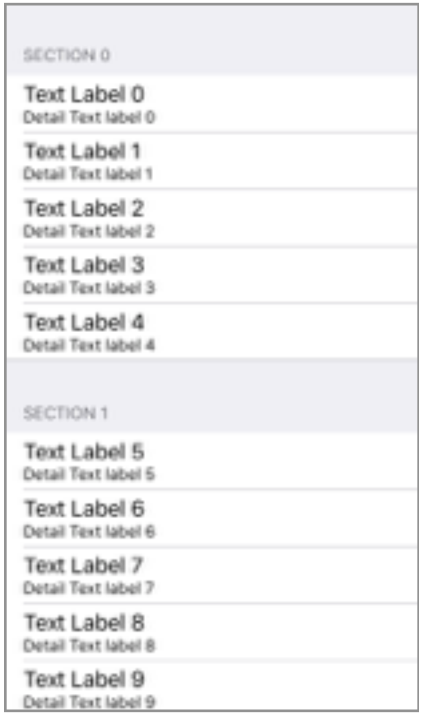
Text Label 7

Text Label 8

Text Label 9

Basic

UITableViewCellStyle
.default



SECTION 0

Text Label 0
Detail Text label 0

Text Label 1
Detail Text label 1

Text Label 2
Detail Text label 2

Text Label 3
Detail Text label 3

Text Label 4
Detail Text label 4

SECTION 1

Text Label 5
Detail Text label 5

Text Label 6
Detail Text label 6

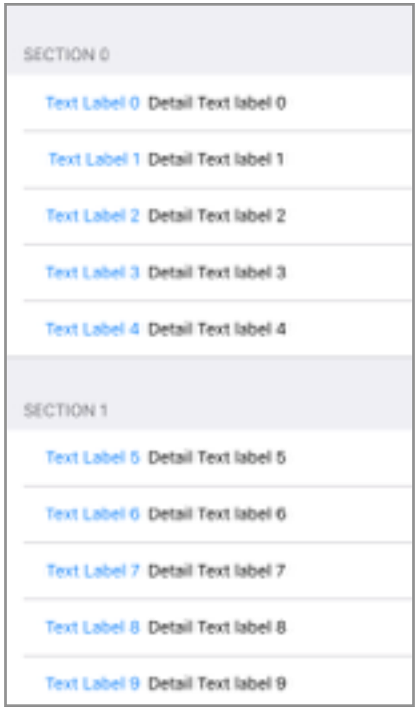
Text Label 7
Detail Text label 7

Text Label 8
Detail Text label 8

Text Label 9
Detail Text label 9

Subtitle

UITableViewCellStyle
.subtitle



SECTION 0

Text Label 0 Detail Text label 0

Text Label 1 Detail Text label 1

Text Label 2 Detail Text label 2

Text Label 3 Detail Text label 3

Text Label 4 Detail Text label 4

SECTION 1

Text Label 5 Detail Text label 5

Text Label 6 Detail Text label 6

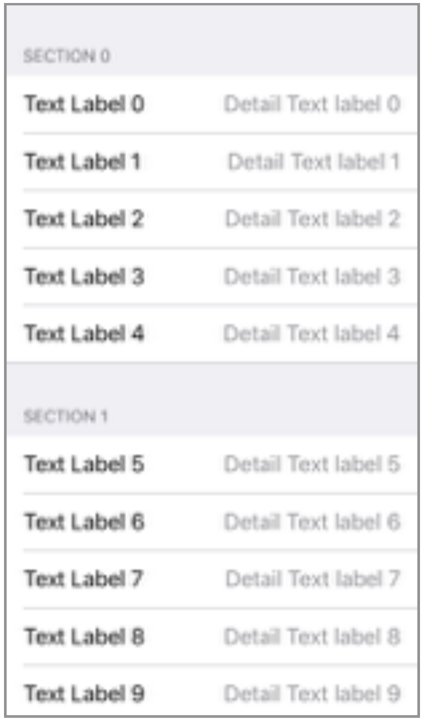
Text Label 7 Detail Text label 7

Text Label 8 Detail Text label 8

Text Label 9 Detail Text label 9

Left Detail

UITableViewCellStyle
.value2



SECTION 0

Text Label 0 Detail Text label 0

Text Label 1 Detail Text label 1

Text Label 2 Detail Text label 2

Text Label 3 Detail Text label 3

Text Label 4 Detail Text label 4

SECTION 1

Text Label 5 Detail Text label 5

Text Label 6 Detail Text label 6

Text Label 7 Detail Text label 7

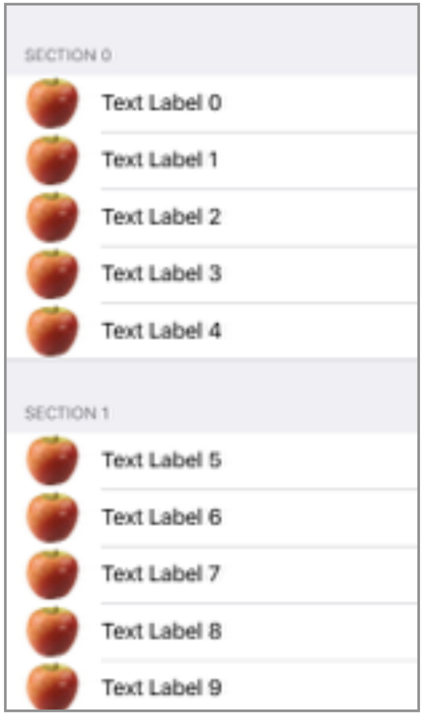
Text Label 8 Detail Text label 8

Text Label 9 Detail Text label 9

Right Detail

UITableViewCellStyle
.value1

Table View Cell Styles (With Images)



SECTION 0

Text Label 0

Text Label 1

Text Label 2

Text Label 3

Text Label 4

SECTION 1

Text Label 5

Text Label 6

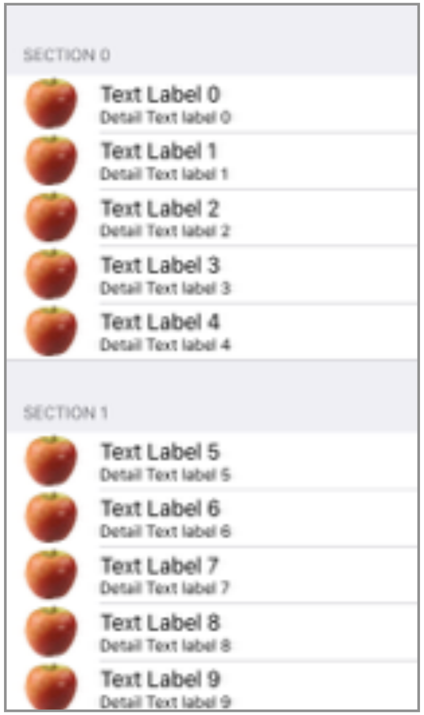
Text Label 7

Text Label 8

Text Label 9

Basic

UITableViewCellStyle
.default



SECTION 0

Text Label 0
Detail Text label 0

Text Label 1
Detail Text label 1

Text Label 2
Detail Text label 2

Text Label 3
Detail Text label 3

Text Label 4
Detail Text label 4

SECTION 1

Text Label 5
Detail Text label 5

Text Label 6
Detail Text label 6

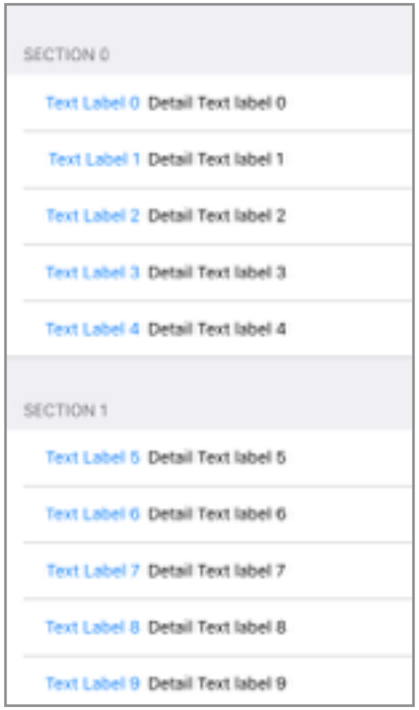
Text Label 7
Detail Text label 7

Text Label 8
Detail Text label 8

Text Label 9
Detail Text label 9

Subtitle

UITableViewCellStyle
.subtitle



SECTION 0

Text Label 0 Detail Text label 0

Text Label 1 Detail Text label 1

Text Label 2 Detail Text label 2

Text Label 3 Detail Text label 3

Text Label 4 Detail Text label 4

SECTION 1

Text Label 5 Detail Text label 5

Text Label 6 Detail Text label 6

Text Label 7 Detail Text label 7

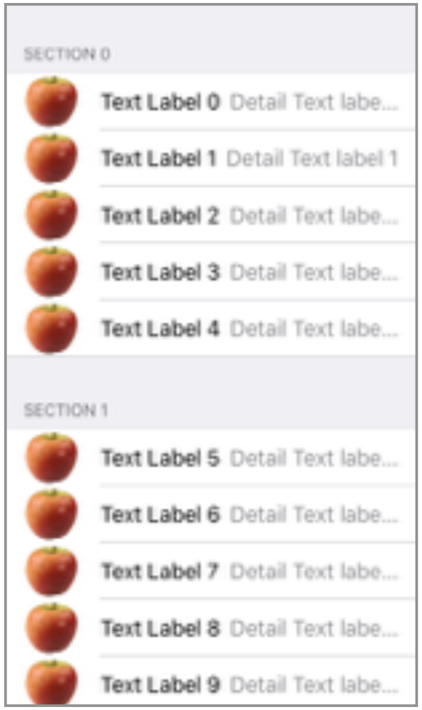
Text Label 8 Detail Text label 8

Text Label 9 Detail Text label 9

Left Detail

UITableViewCellStyle
.value2

Left Detail style does **not** display images.



SECTION 0

Text Label 0 Detail Text labe...

Text Label 1 Detail Text label 1

Text Label 2 Detail Text labe...

Text Label 3 Detail Text labe...

Text Label 4 Detail Text labe...

SECTION 1

Text Label 5 Detail Text labe...

Text Label 6 Detail Text labe...

Text Label 7 Detail Text labe...

Text Label 8 Detail Text labe...

Text Label 9 Detail Text labe...

Right Detail

UITableViewCellStyle
.value1

Table View Cell Style

In XCode,
after selecting a Table View Cell,
change the Table View Cell Style here.

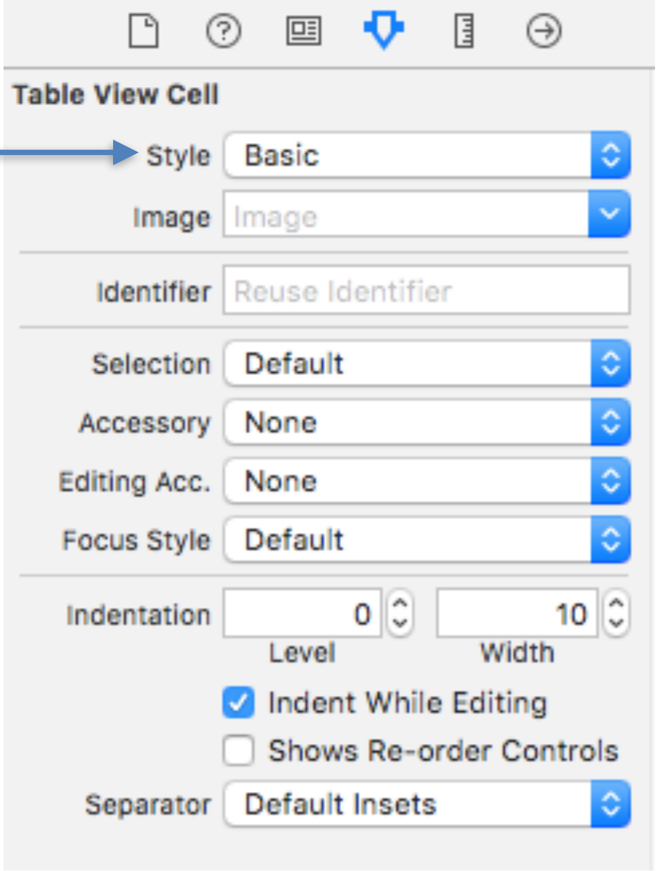


Table View Cell

Style: Basic

Image: Image

Identifier: Reuse Identifier

Selection: Default

Accessory: None

Editing Acc.: None

Focus Style: Default

Indentation: 0 Level, 10 Width

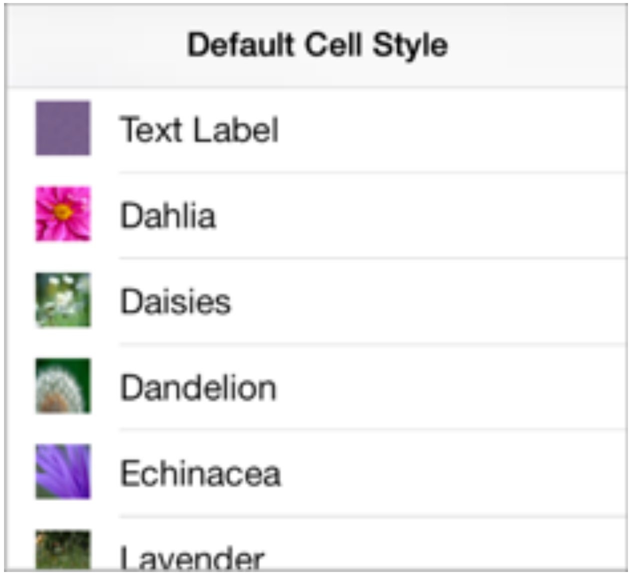
☑ Indent While Editing

□ Shows Re-order Controls

Separator: Default Insets

Table View Cell Styles - Realistic Examples

- UITableViewCellStyle.default
  Default style with label on the left.



Default Cell Style

Text Label

Dahlia

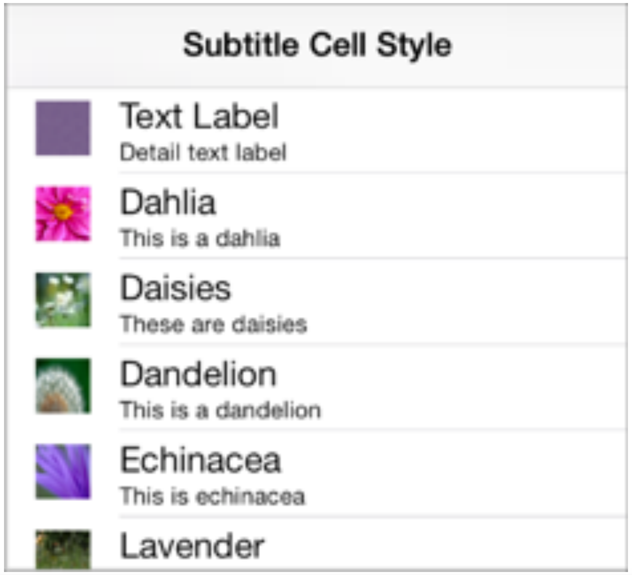
Daisies

Dandelion

Echinacea

Lavender

- UITableViewCellStyle.subtitle
  Label on the left, and a blue label on the right.



Subtitle Cell Style

Text Label
Detail text label

Dahlia
This is a dahlia

Daisies
These are daisies

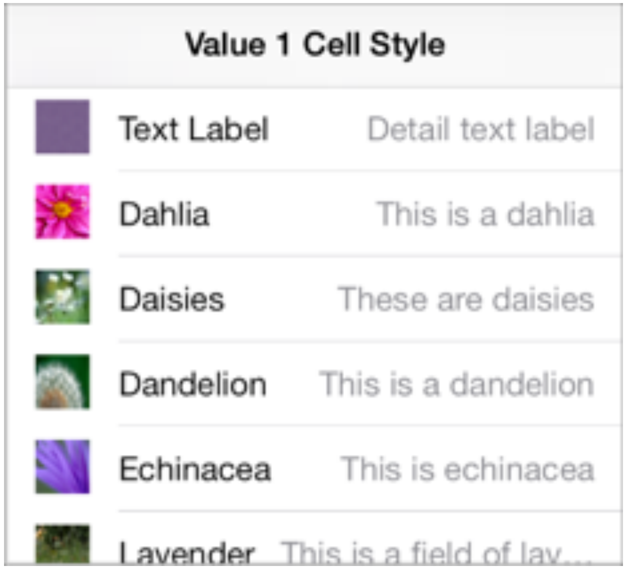
Dandelion
This is a dandelion

Echinacea
This is echinacea

Lavender

Table View Cell Styles - Realistic Examples

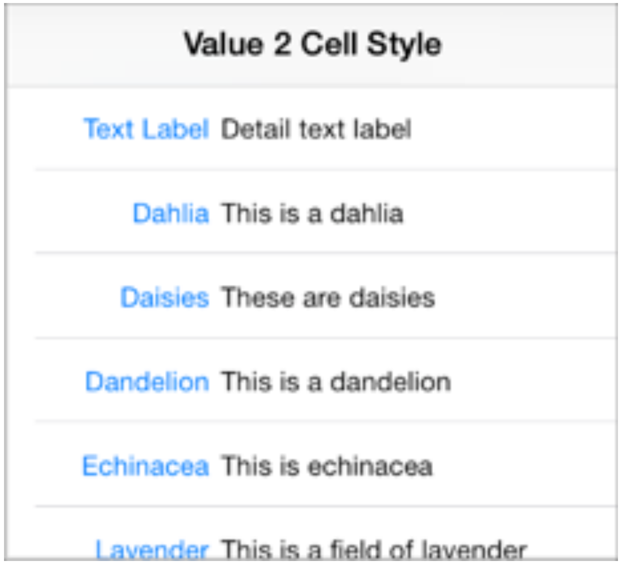
- UITableViewCellStyle.value1
  The value 1 style displays a left-aligned title with, on the same line, a right-aligned subtitle in a lighter font.



Value 1 Cell Style

Text Label	Detail text label
Dahlia	This is a dahlia
Daisies	These are daisies
Dandelion	This is a dandelion
Echinacea	This is echinacea
Lavender	This is a field of lav...

- UITableViewCellStyle.value2
  The value 2 style displays a right-aligned title in a blue font, followed on the same line by a left-aligned subtitle in a black font. Images don’t fit well in this style.



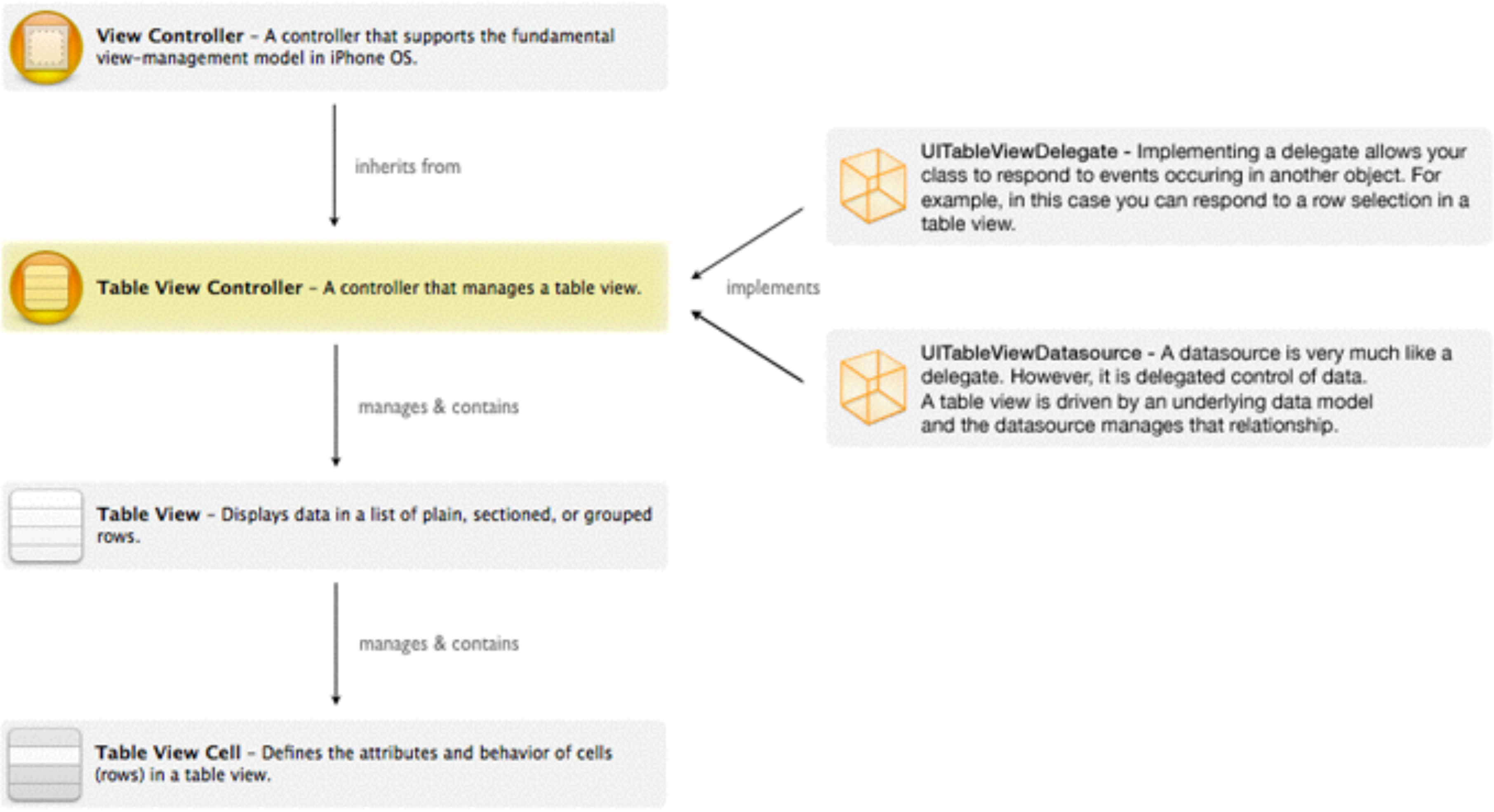
Value 2 Cell Style

Text Label	Detail text label
Dahlia	This is a dahlia
Daisies	These are daisies
Dandelion	This is a dandelion
Echinacea	This is echinacea
Lavender	This is a field of lavender

Table View Controller

UITableViewDataSource Protocol

UITableViewController



View Controller – A controller that supports the fundamental view-management model in iPhone OS.

Table View Controller – A controller that manages a table view.

UITableViewDelegate - Implementing a delegate allows your class to respond to events occuring in another object. For example, in this case you can respond to a row selection in a table view.

UITableViewDatasource - A datasource is very much like a delegate. However, it is delegated control of data. A table view is driven by an underlying data model and the datasource manages that relationship.

Table View – Displays data in a list of plain, sectioned, or grouped rows.

Table View Cell – Defines the attributes and behavior of cells (rows) in a table view.

Reference: http://blog.teamtreehouse.com/introduction-to-the-ios-uitableviewcontroller

UITableViewDataSource Protocol

- UITableViewDataSource (2 **required** methods):

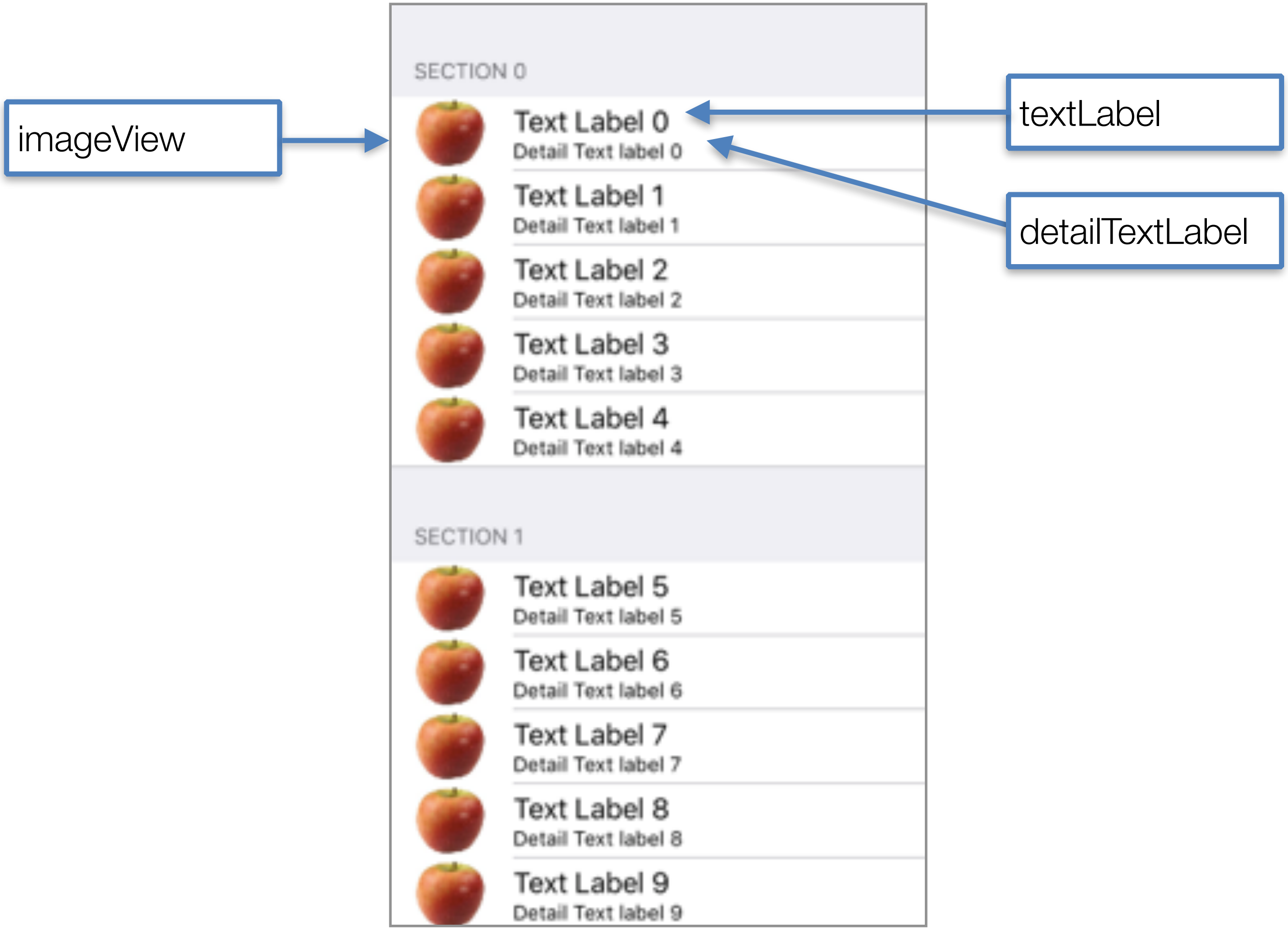
```
func tableView(_ tableView: UITableView,
    numberOfRowsInSection section: Int) -> Int
```

Tells the table view how many rows to display in each section

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

Creates and provides the cell to display the contents for each row in the table

UITableViewDataSource Views



imageView

textLabel

detailTextLabel

SECTION 0

Text Label 0
Detail Text label 0

Text Label 1
Detail Text label 1

Text Label 2
Detail Text label 2

Text Label 3
Detail Text label 3

Text Label 4
Detail Text label 4

SECTION 1

Text Label 5
Detail Text label 5

Text Label 6
Detail Text label 6

Text Label 7
Detail Text label 7

Text Label 8
Detail Text label 8

Text Label 9
Detail Text label 9

UITableViewDataSource

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















}
```

UITableViewDataSource

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
    -> UITableViewCell
{
    // Dequeues a cell marked for re-use if available. If not,
    // iOS will create a new cell for you.
    //
    let cell : UITableViewCell! = tableView.dequeueReusableCell(
        withIdentifier: "cell", for: indexPath)







}
```

Gets the dequeued cell and reuses it.

UITableViewDataSource

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
    -> UITableViewCell
{
    // Dequeues a cell marked for re-use if available. If not,
    // iOS will create a new cell for you.
    //
    let cell : UITableViewCell! = tableView.dequeueReusableCell(
        withIdentifier: "cell", for: indexPath)

    // Using the re-used cell, or the newly created
    // cell, we update the text label's text property.
    //
    let movie = movieList[indexPath.row]
    cell.textLabel?.text = "\(movie.MovieName) \(movie.Length)
        mins"

    return cell
}
```

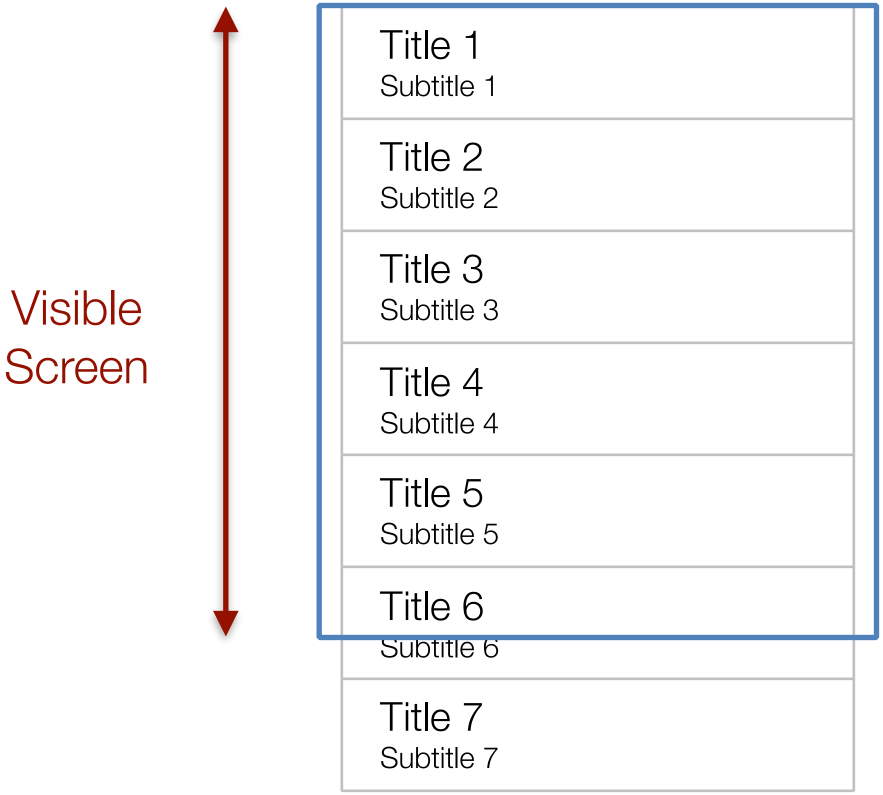
Gets the dequeued cell and reuses it.

Update the cell to display our contents.

Using the default cells, you can use its:

- **textLabel**
- **detailTextLabel**
- **imageView**

UITableViewCells - What Happens



Visible Screen

Title 1
Subtitle 1

Title 2
Subtitle 2

Title 3
Subtitle 3

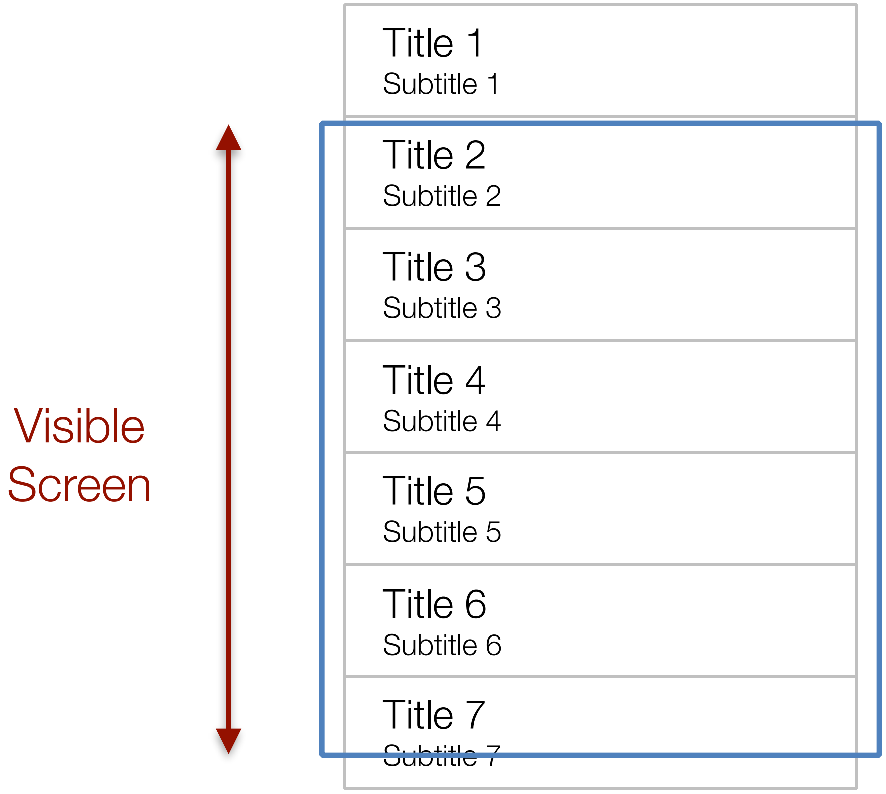
Title 4
Subtitle 4

Title 5
Subtitle 5

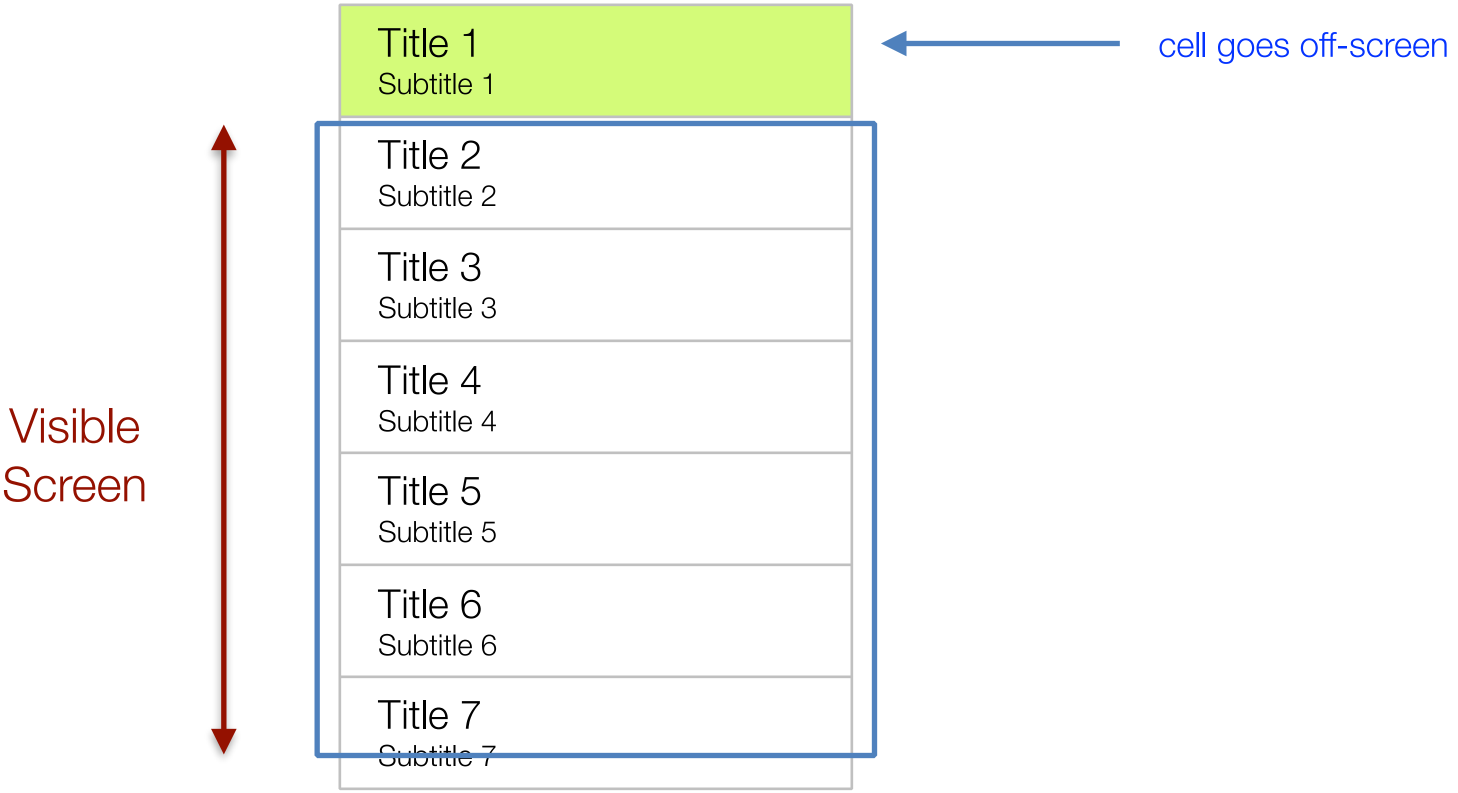
Title 6
Subtitle 6

Title 7
Subtitle 7

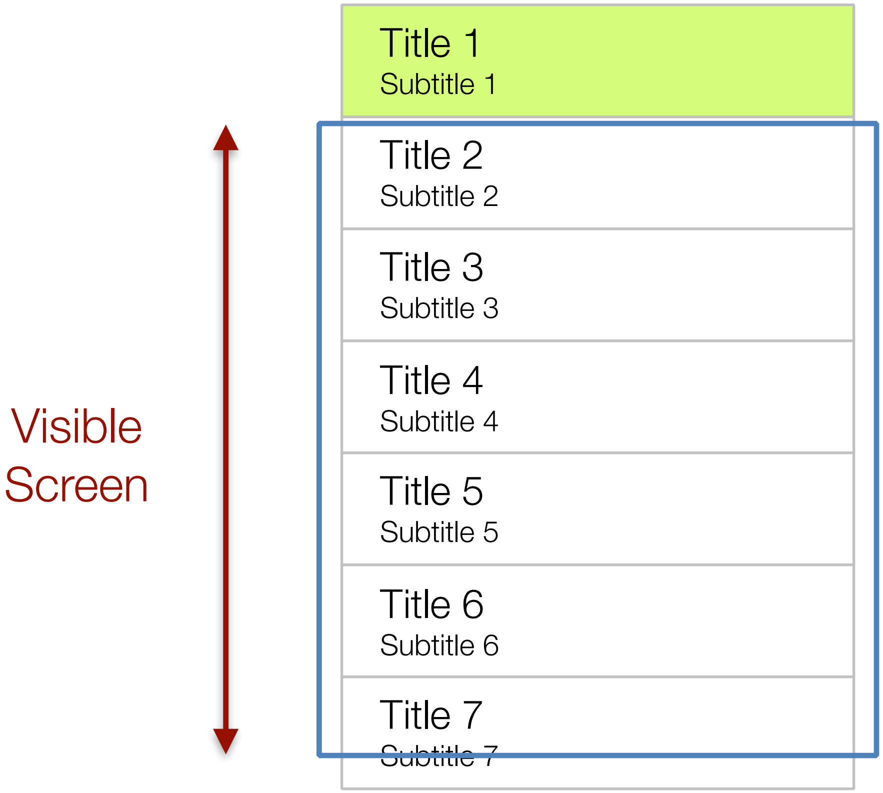
UITableViewCells - What Happens



UITableViewCells - What Happens



UITableViewCells - What Happens



Visible Screen

Title 1
Subtitle 1

Title 2
Subtitle 2

Title 3
Subtitle 3

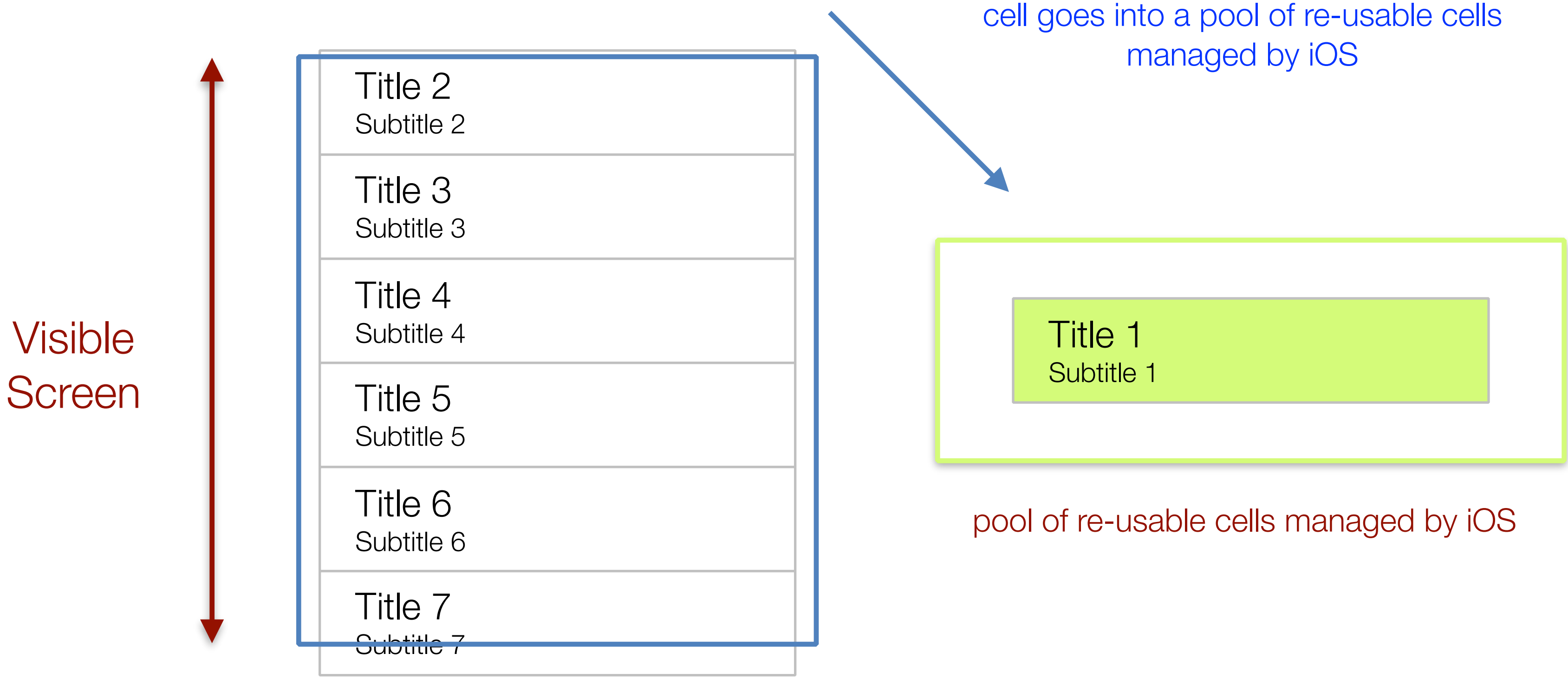
Title 4
Subtitle 4

Title 5
Subtitle 5

Title 6
Subtitle 6

Title 7
Subtitle 7

UITableViewCells - What Happens



Visible Screen

Title 2
Subtitle 2

Title 3
Subtitle 3

Title 4
Subtitle 4

Title 5
Subtitle 5

Title 6
Subtitle 6

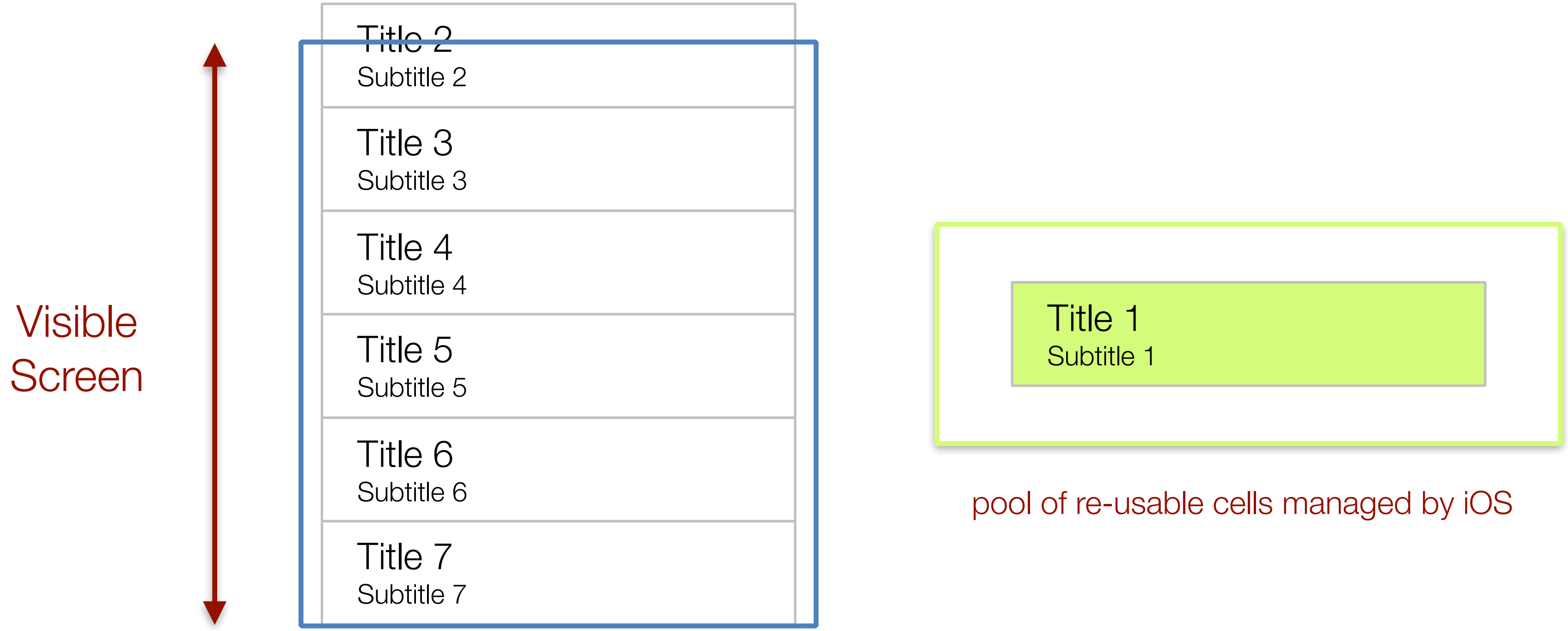
Title 7
Subtitle 7

cell goes into a pool of re-usable cells managed by iOS

Title 1
Subtitle 1

pool of re-usable cells managed by iOS

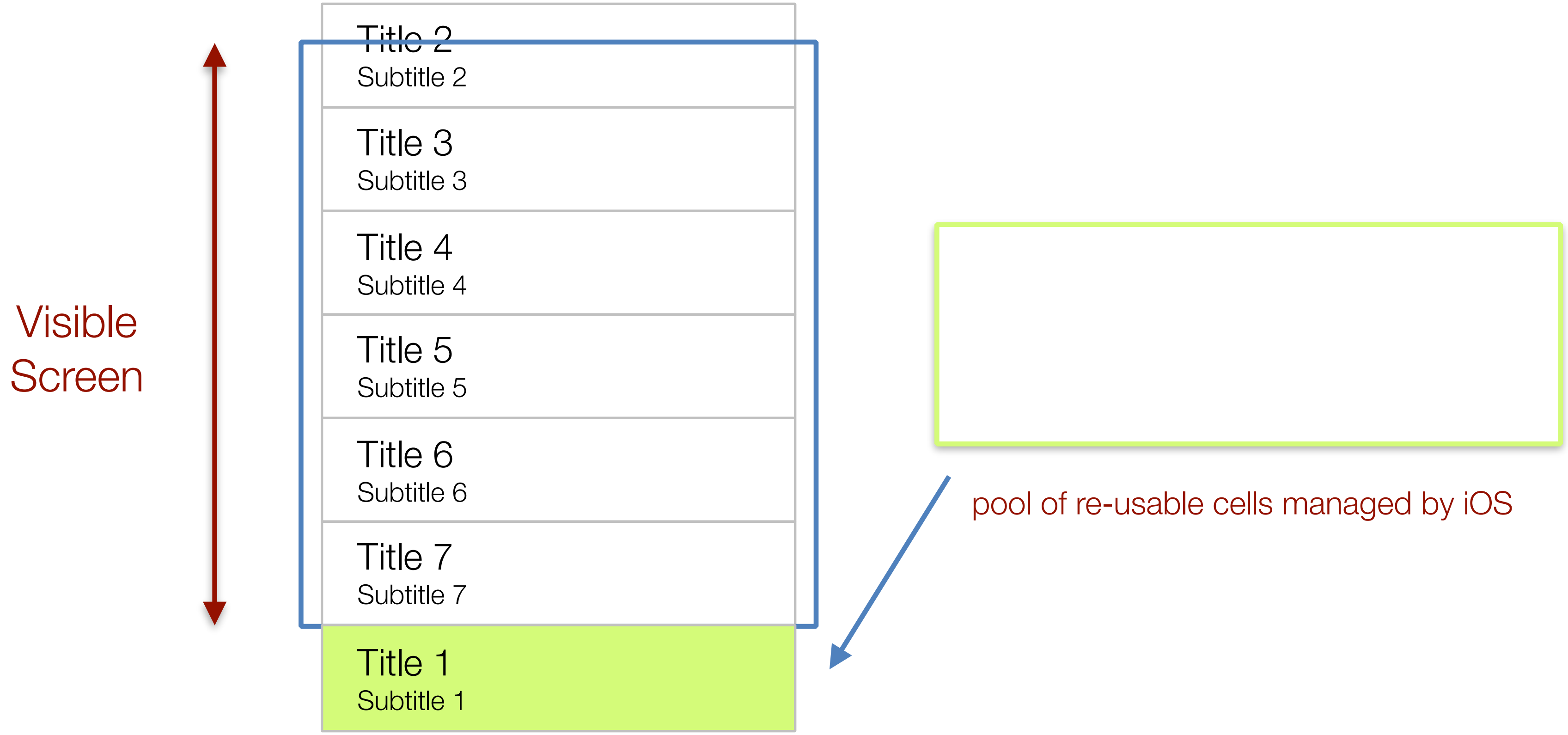
UITableViewCells - What Happens



pool of re-usable cells managed by iOS

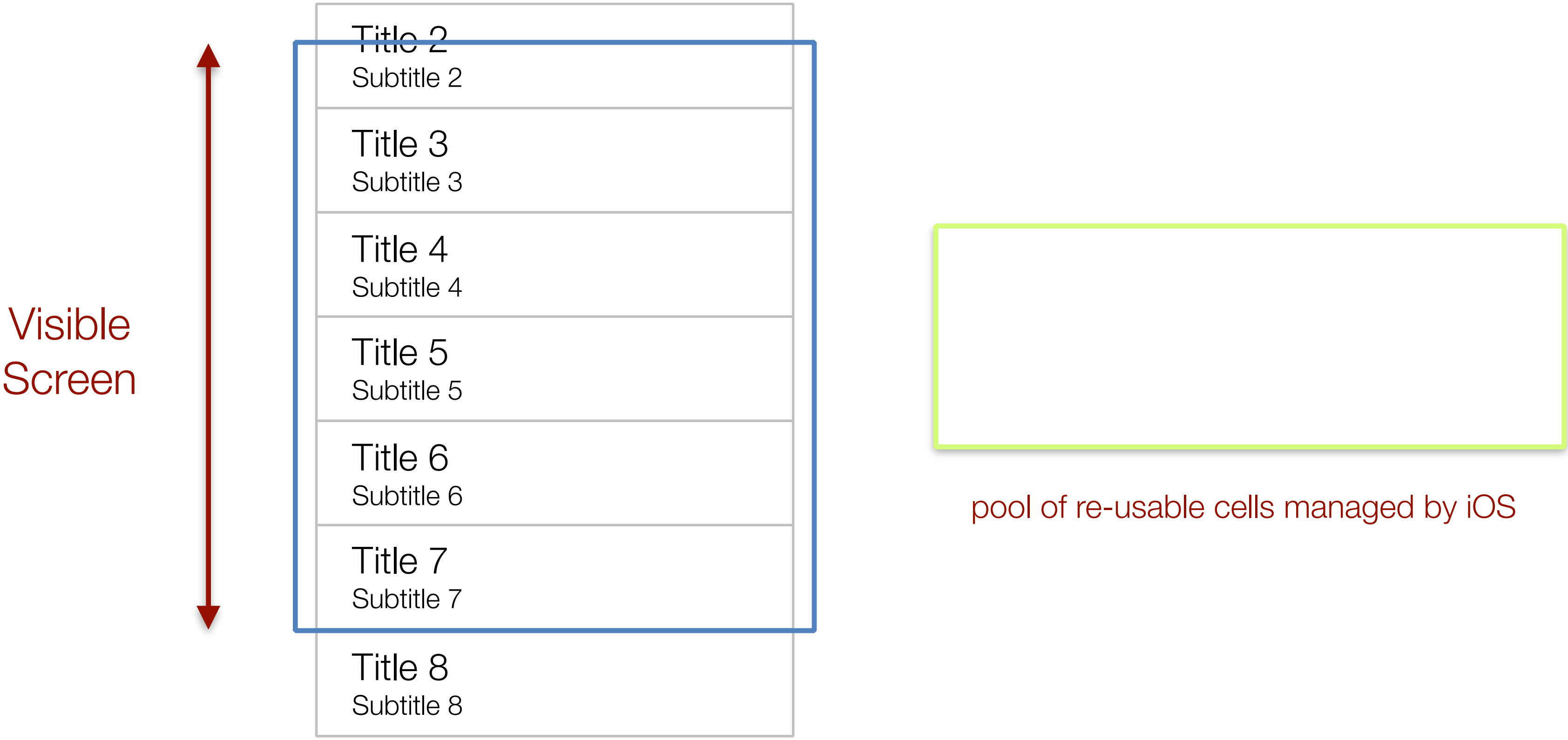
As we slide up, the 8th cell
is about to appear

UITableViewCells - What Happens



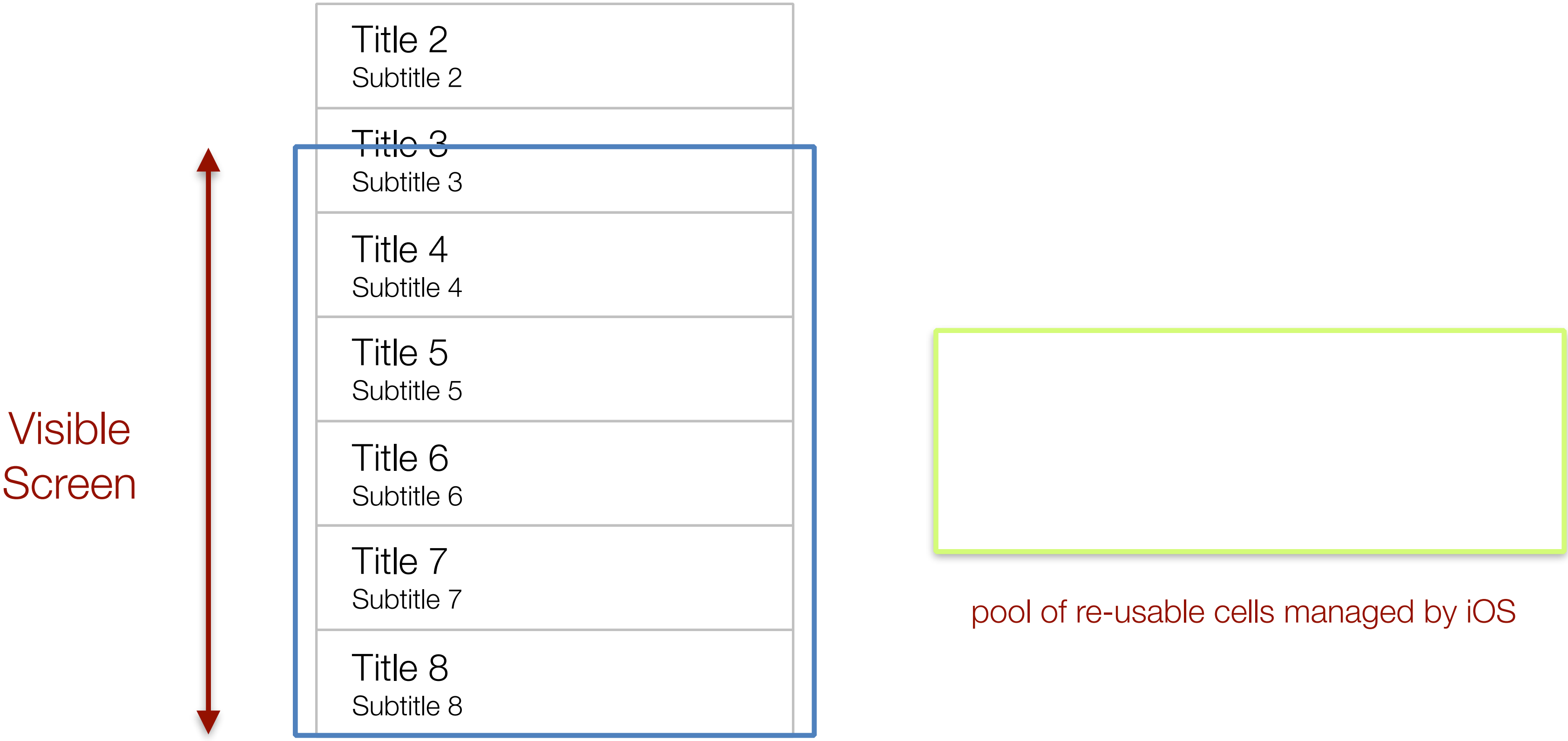
We dequeue this cell
from IOS’s reusable cells

UITableViewCells - What Happens



And we set its contents
to what the user expects to see in
the 8th cell

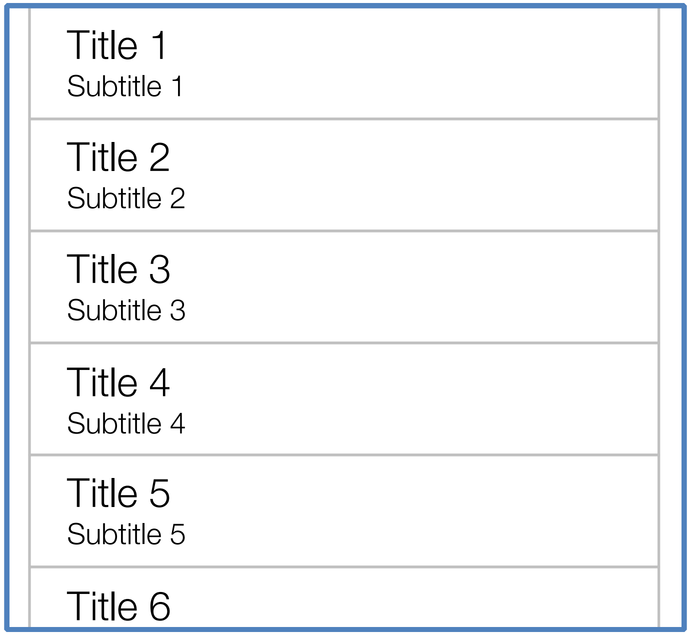
UITableViewCells - What Happens



pool of re-usable cells managed by iOS

Then the 8th cell scrolls into view

UITableViewCells - What You See



Title 1

Subtitle 1

Title 2

Subtitle 2

Title 3

Subtitle 3

Title 4

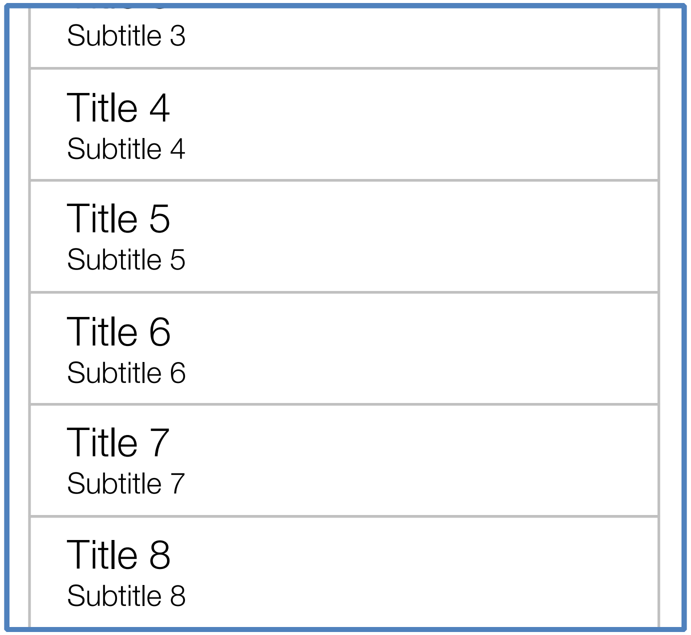
Subtitle 4

Title 5

Subtitle 5

Title 6

UITableViewCells - What You See



Subtitle 3

Title 4
Subtitle 4

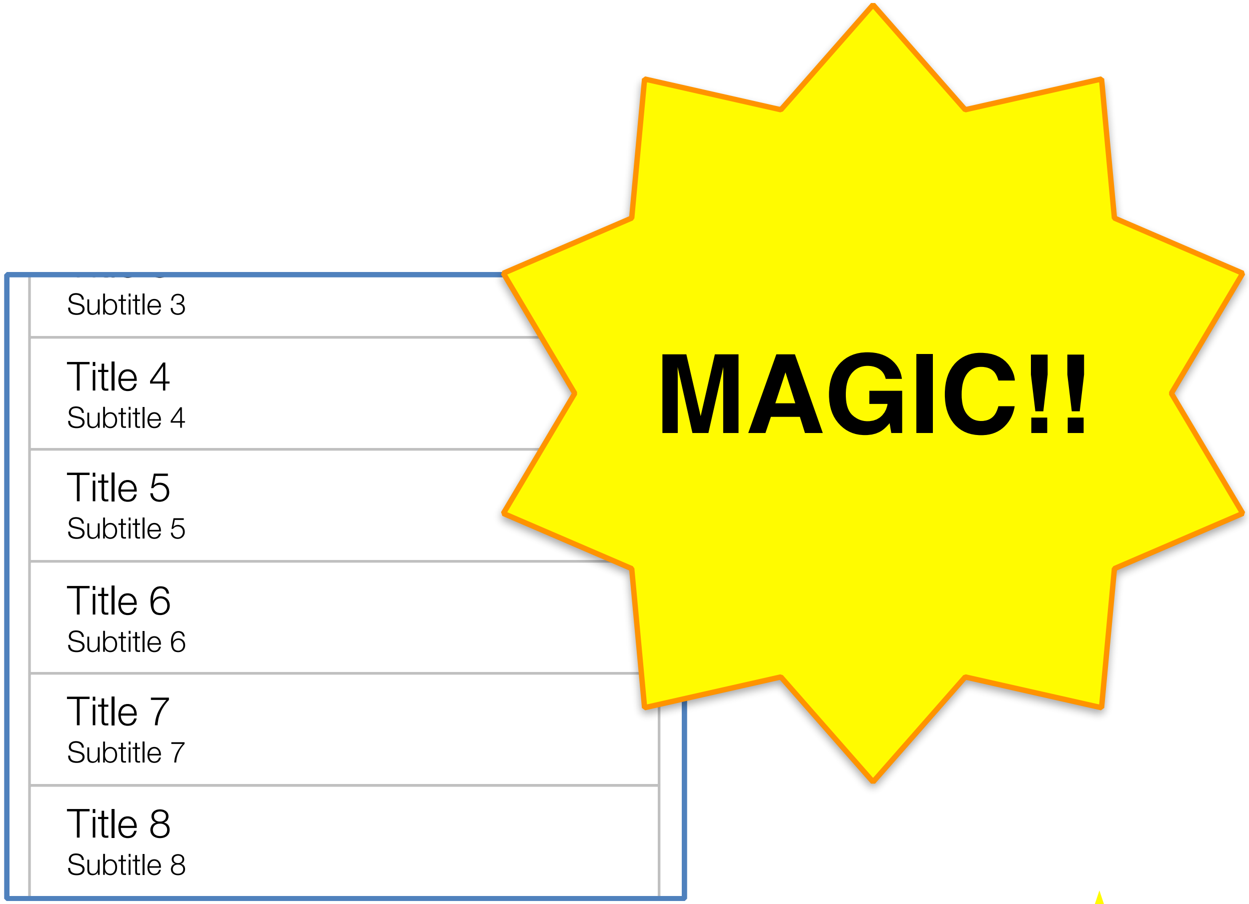
Title 5
Subtitle 5

Title 6
Subtitle 6

Title 7
Subtitle 7

Title 8
Subtitle 8

UITableViewCells - What You See



Subtitle 3

Title 4
Subtitle 4

Title 5
Subtitle 5

Title 6
Subtitle 6

Title 7
Subtitle 7

Title 8
Subtitle 8

UITableViewDataSource Protocol

- Other optional methods that allow you to:
  - configure multiple sections,
  - provide headers and/or footers,
  - support removing and reordering rows in the table

UITableViewDataSource Protocol

- More on UITableViewDataSource:
  - https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITableViewDataSource_Protocol/

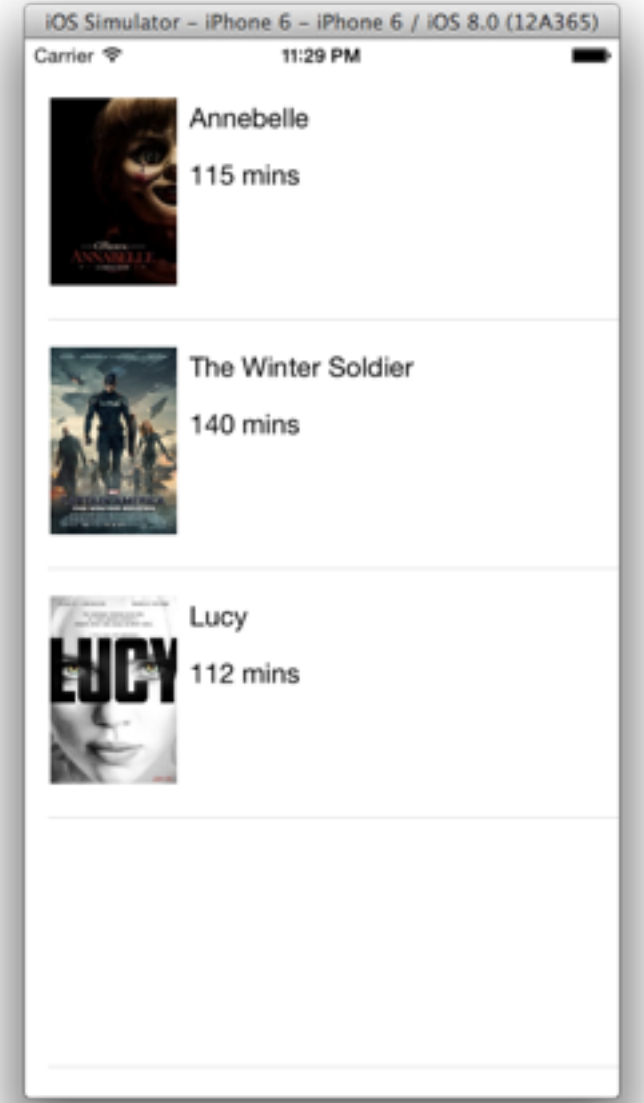
Table View Controller

UITableViewDelegate Protocol

UITableViewDelegate Protocol

- It has no required methods.
- It declares methods that allow the delegate to modify visible aspects of the table view, manage selections, support an accessory view and support edit of individual rows in a table.

Customizing a Table View



iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.0 (12A365)

Carrier 11:29 PM

Annebelle

115 mins

The Winter Soldier

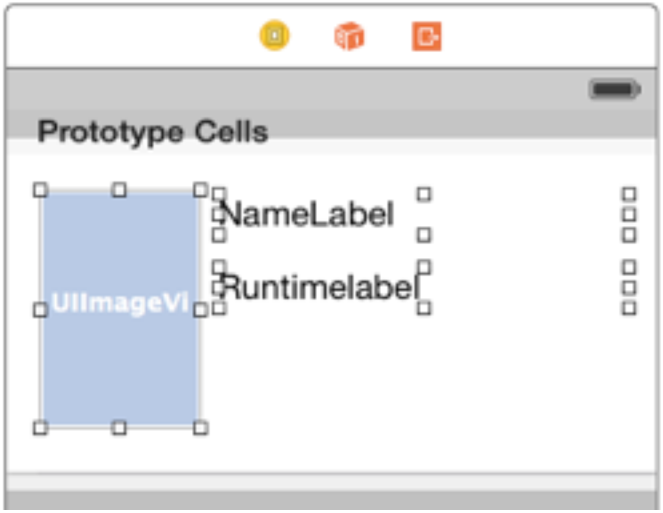
140 mins

Lucy

112 mins

Creating a UITableViewCell

- Create a prototype cell and hook up the views to the IBOutlets.



```
class MovieCell: UITableViewCell {
    @IBOutlet weak var nameLabel : UILabel!
    @IBOutlet weak var runTimeLabel : UILabel!
    @IBOutlet weak var movieImageView: UIImageView!

}
```

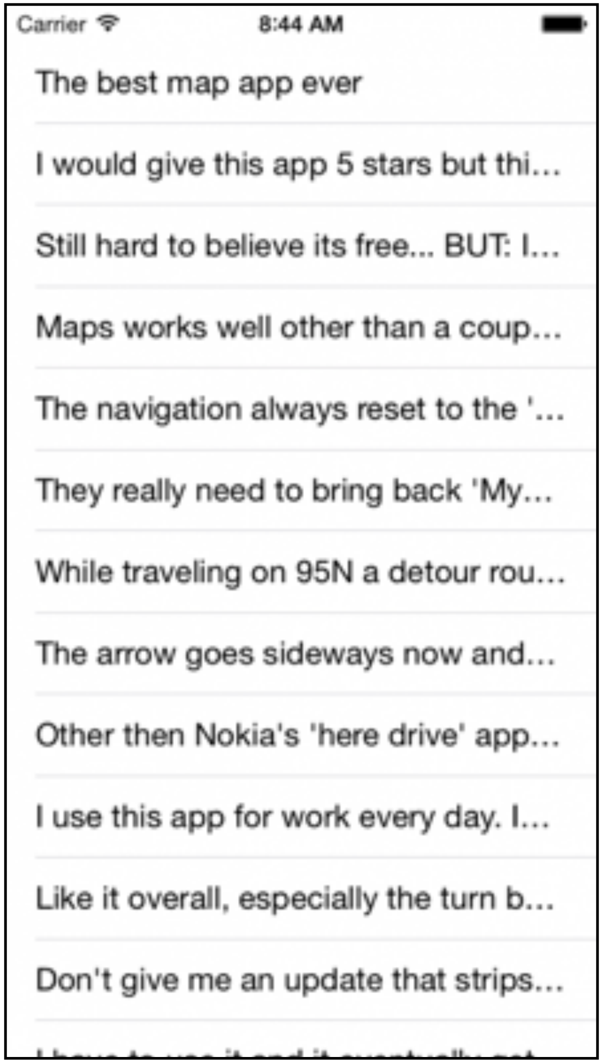
UITableViewDataSource

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
   -> UITableViewCell
{
    // Dequeues a cell marked for re-use if available. If not,
    // iOS will create a new cell for you.
    //
    let cell : UITableViewCell! = tableView.dequeueReusableCell(
        withIdentifier: "cell", for: indexPath)

    // Using the re-used cell, or the newly created
    // cell, we update the text label's text property.
    //
    let movie = movieList[indexPath.row]
    cell.label?.text = "\(movie.MovieName) \(movie.Length)
        mins"

    return cell
}
```

Similar to the code as you would write for a non-custom cell.

Self Sizing Cells

iOS 7 (left) vs iOS 8 (right)



The best map app ever

I would give this app 5 stars but thi...

Still hard to believe its free... BUT: I...

Maps works well other than a coup...

The navigation always reset to the '...

They really need to bring back 'My...

While traveling on 95N a detour rou...

The arrow goes sideways now and...

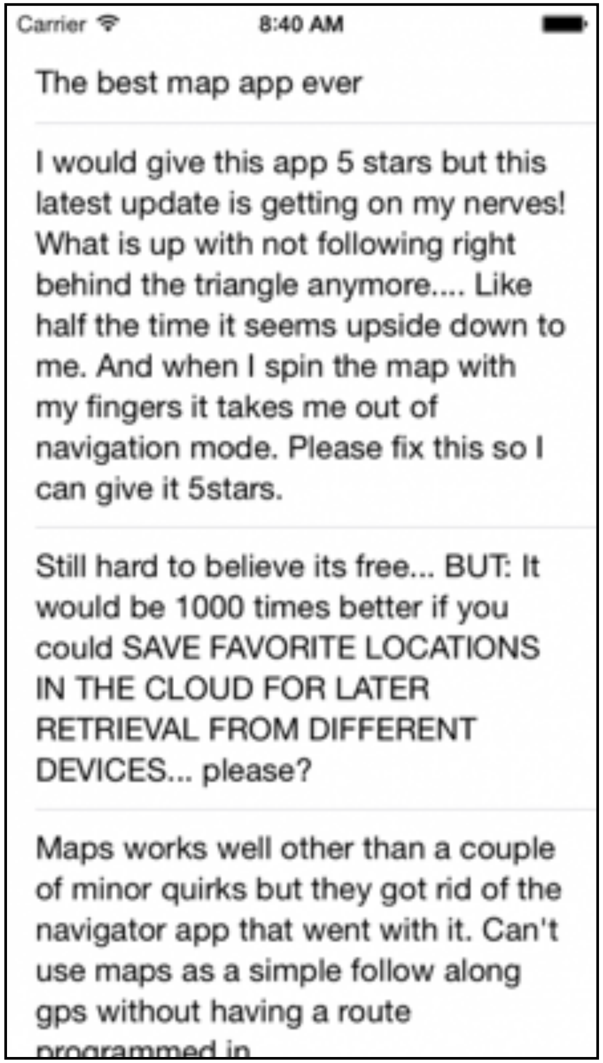
Other then Nokia's 'here drive' app...

I use this app for work every day. I...

Like it overall, especially the turn b...

Don't give me an update that strips...

I have to use it and it eventually got



The best map app ever

I would give this app 5 stars but this latest update is getting on my nerves! What is up with not following right behind the triangle anymore.... Like half the time it seems upside down to me. And when I spin the map with my fingers it takes me out of navigation mode. Please fix this so I can give it 5stars.

Still hard to believe its free... BUT: It would be 1000 times better if you could SAVE FAVORITE LOCATIONS IN THE CLOUD FOR LATER RETRIEVAL FROM DIFFERENT DEVICES... please?

Maps works well other than a couple of minor quirks but they got rid of the navigator app that went with it. Can't use maps as a simple follow along gps without having a route programmed in

Self Sizing Cells

- Self Sizing Cell provides a solution for displaying dynamic content.
  - Define auto layout constraints for your prototype cell
  - Specify the estimatedRowHeight of your table view
  - Set the rowHeight of your table view to UITableViewAutomaticDimension

```
tableView.estimatedRowHeight = 80;
tableView.rowHeight = UITableViewAutomaticDimension
tableView.reloadData()
```

Editing Mode – Insert Row

- UITableView:

```
func insertRows(at: [IndexPath], with: UITableViewRowAnimation)
```

Call this function on a UITableView instance to insert a row into the table at the specified indexPath.

Editing Mode – Insert Row

```
func addItem()
{
    let m = Movie(
        name: "Dark Flight",
        description: "Ten years ago, a young air hostess, New,
            miraculously survived a plane crash. New insists that a
            vengeful spirit caused that accident, but her belief
            unsettled those around her and she had to go through
            a psycho-therapy. Now, the woman is ready to fly again.",
        runTime: 105,
        imageName: "darkFlight.jpg")

    movieList.append(m)

    let lastRow = data.index(of: m)
    let indexPath = IndexPath(item: lastRow!, section: 0)
    tableView.insertRows(at: [indexPath], with: .automatic)
}
```

Calling this triggers the TableViewDataSource functions to create the respective cells

Editing Mode – Swipe-to-Delete

- UITableViewDelegate:

```
func tableView(UITableView, commit: UITableViewCellEditingStyle,
forRowAt: IndexPath)
```

Triggered when the user swipes-to-delete. Your application then needs to handle what happen when the user deletes the record in the table.

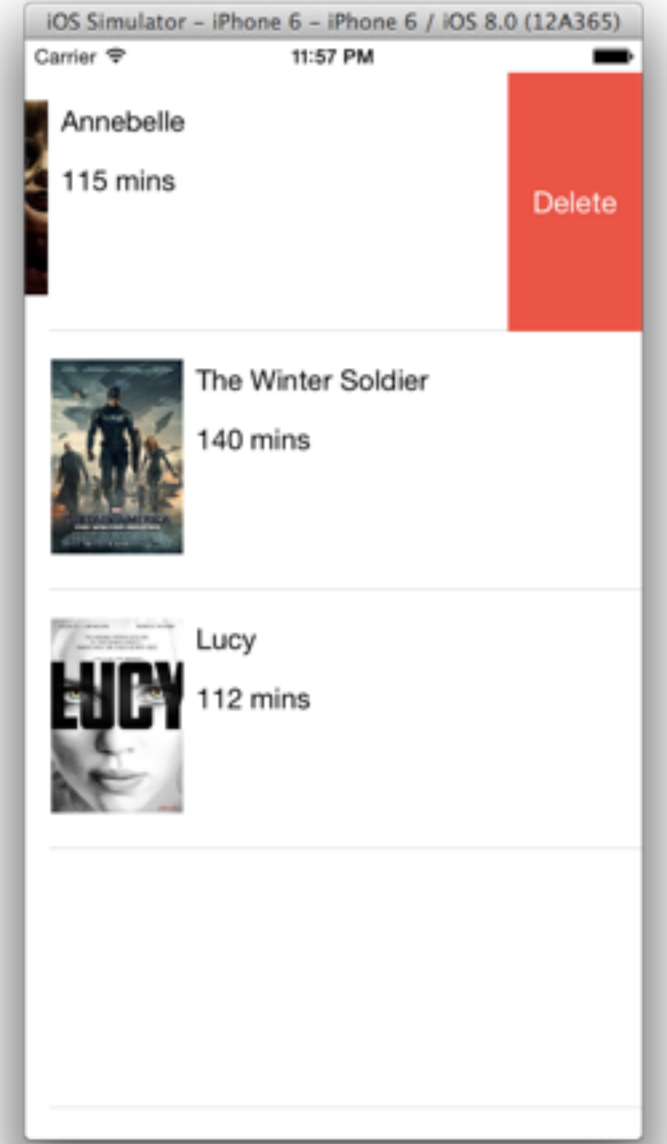
NOTE: You must implement this method if you want swipe-to-delete.

Editing Mode – Swipe-to-Delete

```
func tableView(_ tableView: UITableView,
    commit editingStyle: UITableViewCellEditingStyle,
    forRowAt indexPath: IndexPath)
{
    if editingStyle == .delete
    {
        movieList.remove(at: indexPath.row)
        tableView.deleteRows(at: [indexPath], with: .automatic)
    }

}
```

when you implement this, the rows in the table becomes deletable (with left-swipe gesture)

Editing Mode – Delete Row



iOS Simulator - iPhone 6 - iPhone 6 / iOS 8.0 (12A365)

Annebelle

115 mins

The Winter Soldier

140 mins

Lucy

112 mins

Editing Mode – Switching In/Out

- UITableView:

```
func setEditing(Bool, animated: Bool)
```

Call this function on a UITableView instance to switch the table view in and out of edit mode.

Editing Mode – Re-order Row

- UITableViewDelegate:

```
func tableView(UITableView,
    moveFromRowAt sourceIndexPath: IndexPath,
    to destinationIndexPath: IndexPath)
```

Triggered when the user reorders the row.

NOTE: You must implement this function if you want to display the reorder icons.

Managing the Reordering of Rows

```
@IBAction func editButtonPressed(sender: AnyObject) {
    if !self.tableView.isEditing
    {
        sender.setTitle("Done", for: .normal)
        tableView.setEditing(true, animated: true)
    }
    else
    {
        sender.setTitle("Edit", for: .normal)
        tableView.setEditing(false, animated: true)
    }
}
```

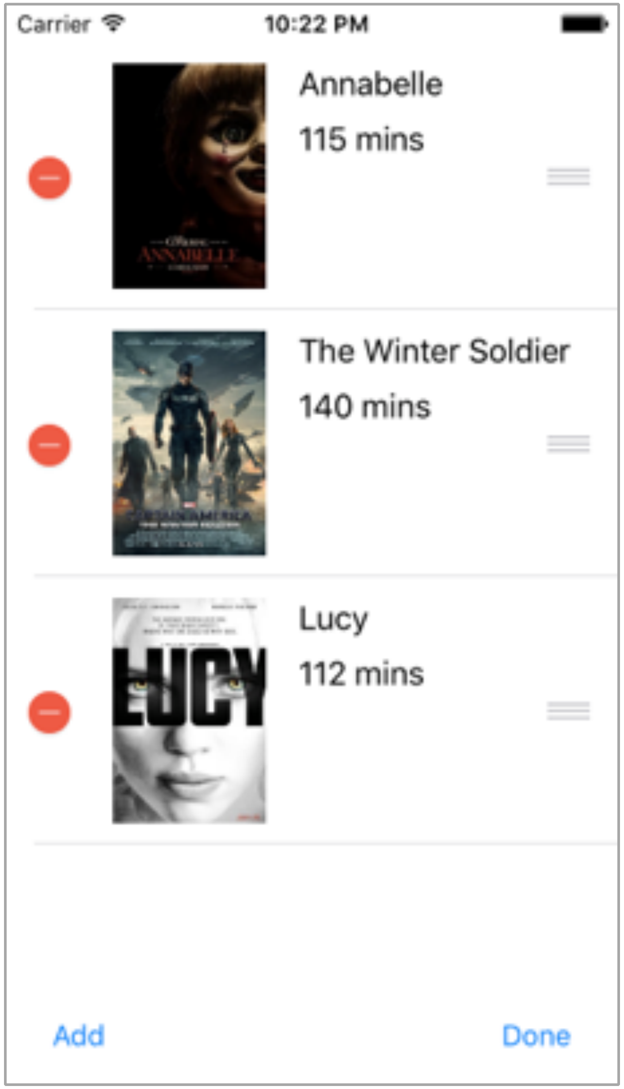
changes the TableView’s mode into edit mode, where the user can delete / reorder rows

```
func tableView(_ tableView: UITableView,
    moveRowAt sourceIndexPath: IndexPath,
    to destinationIndexPath: IndexPath)
{
    let m = data[sourceIndexPath.row]
    data.remove(at: sourceIndexPath.row)
    data.insert(m, at: destinationIndexPath.row)
}
```

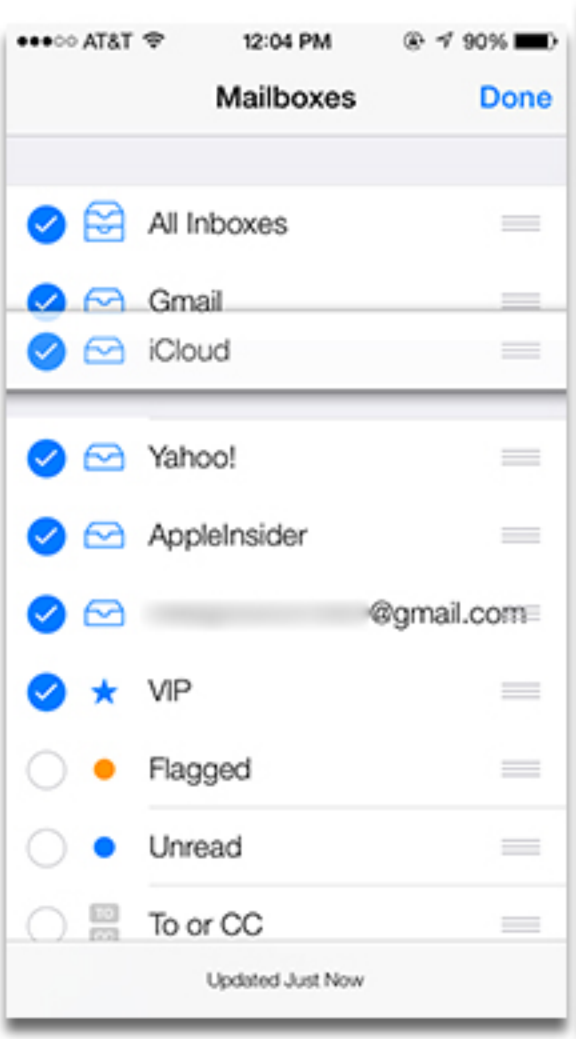
once you implement this method, the row will display a ‘move handle’ on the right side in edit mode.

once the rows are re-ordered, the UITableView-DataSource methods will be called to update the cells

Managing the Reordering of Rows



Managing the Reordering of Rows



UITableViewDelegate Protocol

- More on UITableViewDelegate:
  - https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITableViewDelegate_Protocol/

Summary

- Table View Delegates and Data Source
- Bind data to table view
- Manage table view events
- Understand the difference type of Table View Cell Style
- Create customized table view cell
- Self sizing cells and Dynamic Type in iOS 8
- Edit UITableView – Editing Mode, Adding Rows, Delete Rows, Moving Rows