CS275
Fall 2020

iOS: Editing `UITableView`

BNRG CHAPTER 11

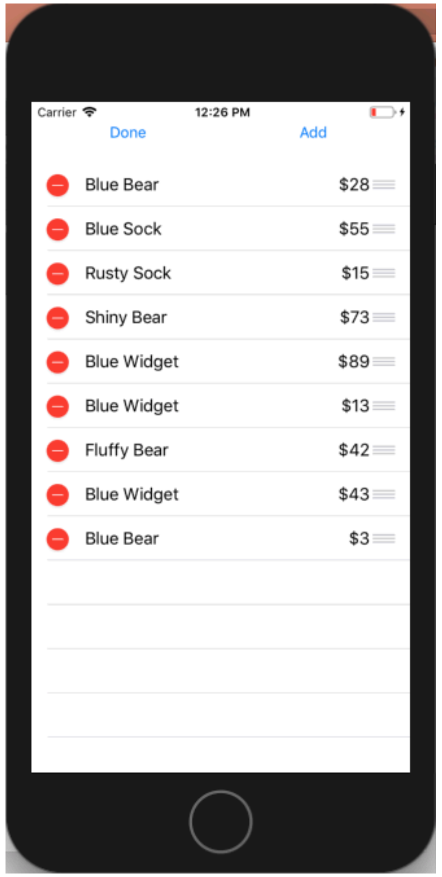
Topics

- Adding, editing, and deleting entries in a `UITableView`

`UITableView`: Editing Mode

`UITableView` has an internal state that reflects whether it is in editing mode

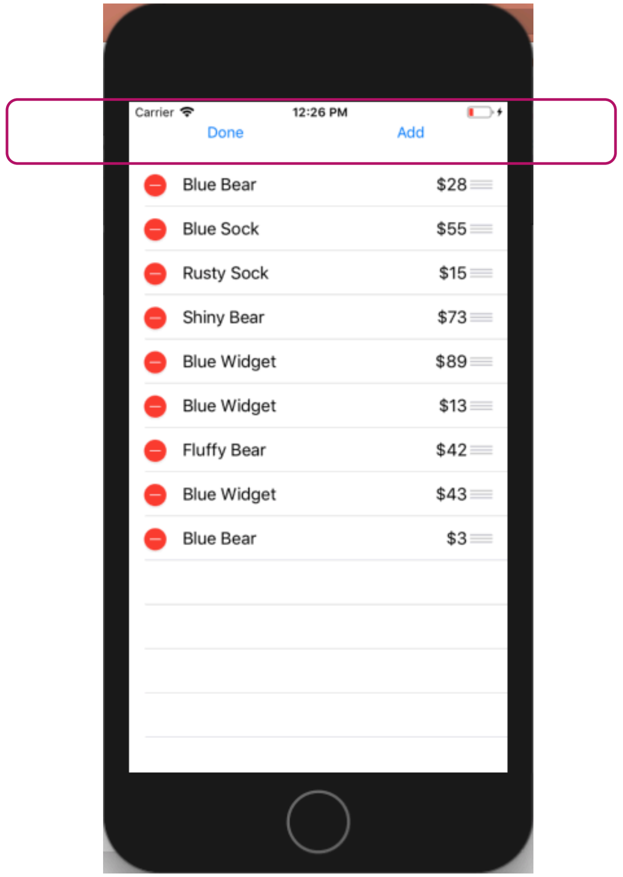
- when it is, then the rows of the table can be manipulated by the user
- but note that editing mode does allow the user to edit the **content** of a row



Blue Bear	$28
Blue Sock	$55
Rusty Sock	$15
Shiny Bear	$73
Blue Widget	$89
Blue Widget	$13
Fluffy Bear	$42
Blue Widget	$43
Blue Bear	$3

`UITableView`: Editing Mode

- But in order to put the `UITableView` into editing mode, we need to give the user a way to tell the UI "I want to edit the table view"
- A useful UI feature: a table header having two buttons to control the editing property
- The table header view can be any view
  - in other words, we can build a view and then add it to the `UITableView`



Blue Bear	$28
Blue Sock	$55
Rusty Sock	$15
Shiny Bear	$73
Blue Widget	$89
Blue Widget	$13
Fluffy Bear	$42
Blue Widget	$43
Blue Bear	$3

Editing Mode

To cause the table view to enter editing mode: call this function on `UITableView`

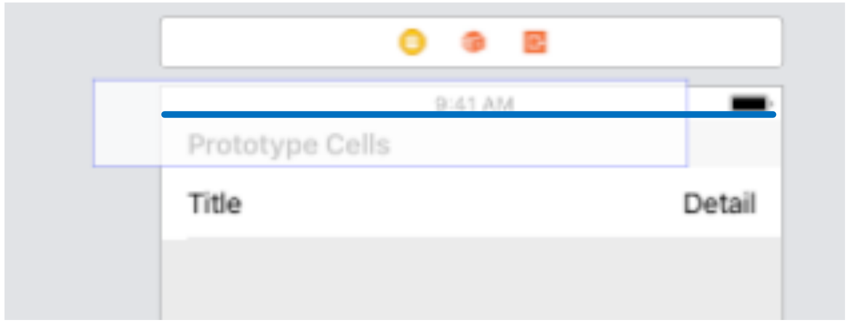
```
setEditing(true, animated: true)
```

So the basic structure for the GUI will be:

- we'll have an Edit button that calls this with `true`
- and a Done button that calls this with `false`

Adding a Header to a `UITableView`

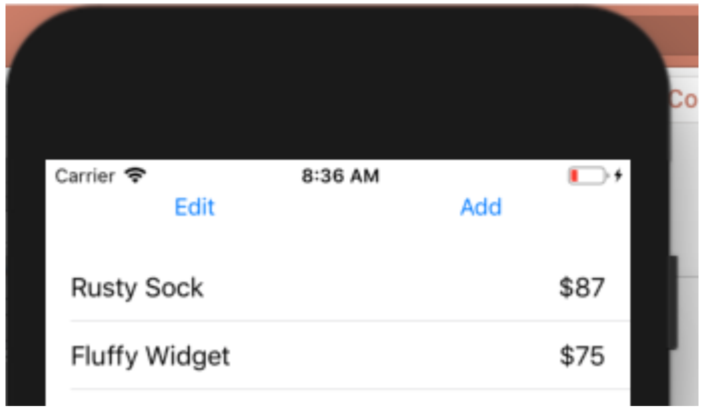
This is a a little tricky – when you get the new view in the right place, a horizontal blue line will appear

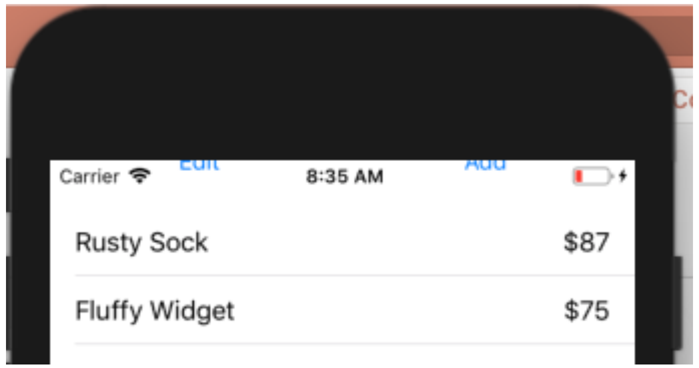


`UITableView` Header

The steps that the book describes will cause the header to scroll with the table entries

- this might or might not seem OK
- we would have to redesign the GUI to change this

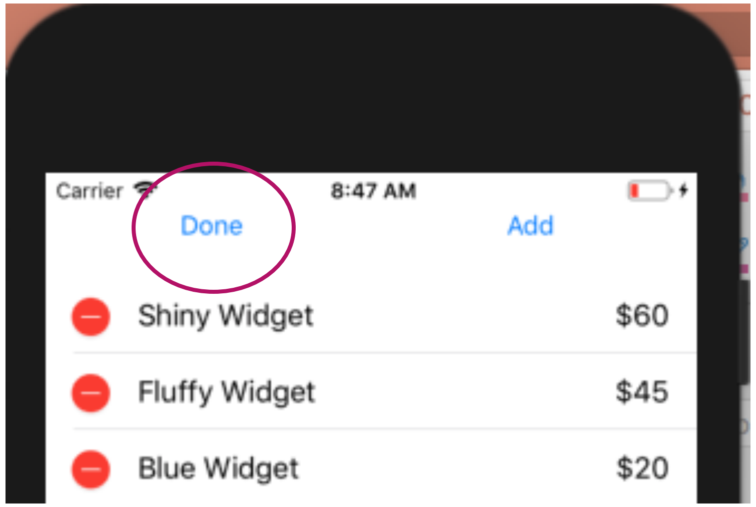




Editing Mode for UITableView

Hook up the Edit/Done button to enable/disable editing of entries

- this changes editing mode on the UIViewController, which then then notifies the UITableView itself



```
@IBAction
func toggleEditingMode(_ sender: UIButton) {
    if isEditing {
        // change the text of the button to inform users of current state
        sender.setTitle("Edit", for: .normal)
        // turn off editing mode
        setEditing(false, animated: true)
    } else {
        //change the text of button to inform users of current state
        sender.setTitle("Done", for: .normal)
        // ender editing mode
        setEditing(true, animated: true)
    }
}
```

Adding a Row

The two common interfaces for adding rows to a table view:

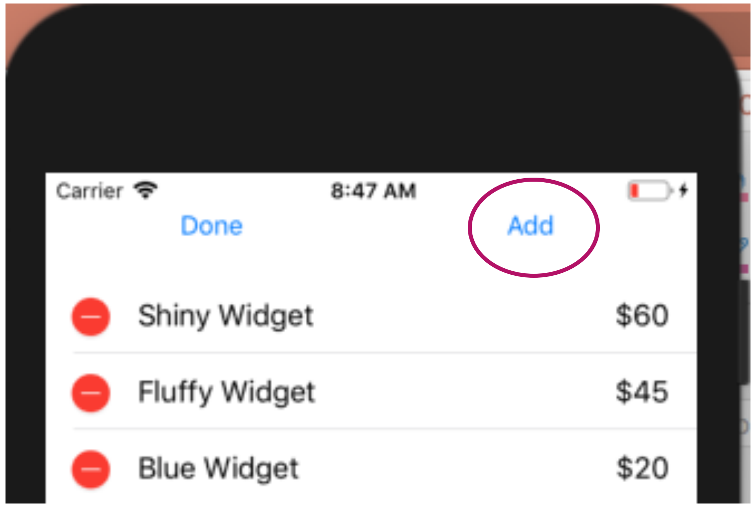
1. a button above the cells of the table view: for adding a new record (e.g., adding a new contact to the list of contacts)
2. a cell with a green plus sign: for adding new information to an existing row in the table

This app uses the first interface

- using `IndexPath`, a struct that holds the location of an item in a nested array (such as a table view with sections)

Editing Mode for `UITableView`

The Add button

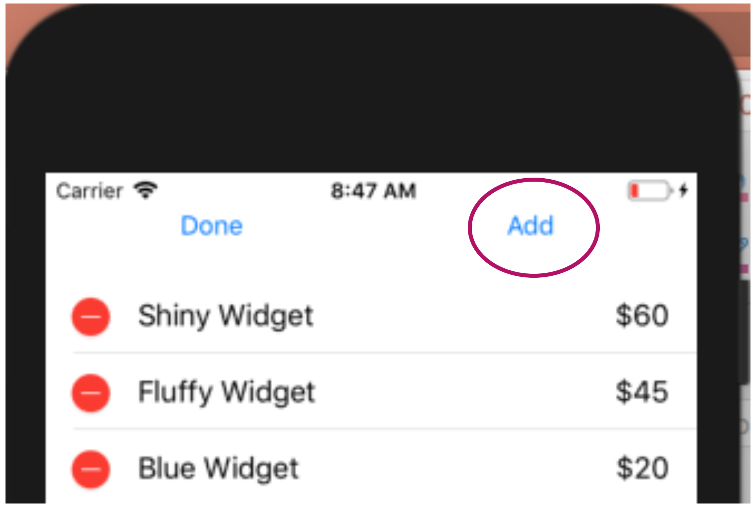


Behavior will be:

- add a new, empty row in the table
- here, the new row will be a randomly generated item

Editing Mode for `UITableView`

The Add button



```
@IBAction
func addNewItem(_ sender: UIButton) {
    // make a new index path for the 0th section, last row
    let lastRow = tableView.numberOfRows(inSection: 0)
    let indexPath = IndexPath(row: lastRow, section: 0)

    // insert this new row into the table
    tableView.insertRows(at: [indexPath], with: .automatic)
}
```

Data Inconsistency

After adding the code for the Add button and running:

```
Homepwner[54574:3673462] *** Terminating app due to uncaught exception
'NSInternalInconsistencyException', reason: 'attempt to insert row 5 into
section 0, but there are only 5 rows in section 0
```

Here's the exception (I have an exception breakpoint active)

```
Homepwner[54574:3673462] *** Assertion failure in -[UITableView
_endCellAnimationsWithContext:],
/BuildRoot/Library/Caches/com.apple.xbs/Sources/UIKitCore_Sim/UIKit-
3698.93.8/UITableView.m:1821
```

Data Inconsistency

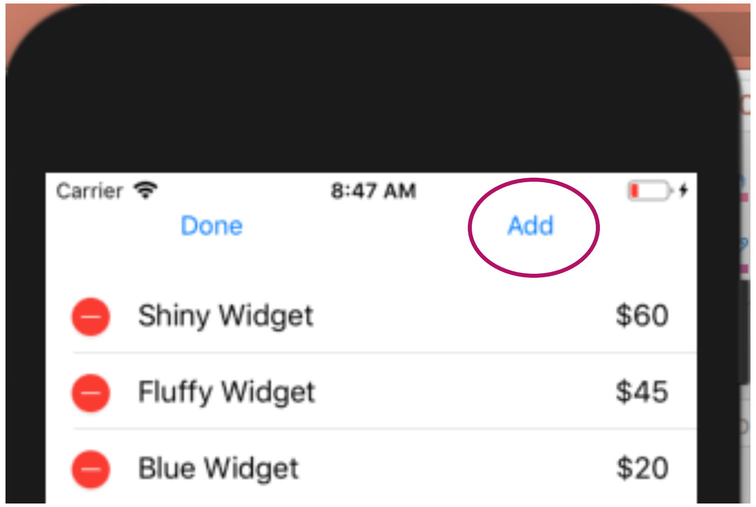
The problem is that the model (`ItemStore`) must stay consistent with the GUI

- the `UITableView` asks the dataSource (the `ItemStore`) how many rows there are
- after we add a row to the GUI, there are still the same number of rows in the `ItemStore`
- so the model is no longer consistent with the view

Solution: when we add a row to the GUI (the table view), we must first add it to the model

Editing Mode for UITableView

The Add button



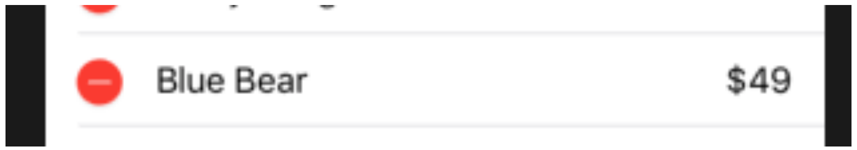
Shiny Widget $60
Fluffy Widget $45
Blue Widget $20

```
@IBAction
func addNewItem(_ sender: UIButton) {
    let newItem = itemStore.createItem()
    // figure out where that item is in the array
    if let index = itemStore.allItems.index(of: newItem) {
        let indexPath = IndexPath(row: index, section: 0)

        // insert this new row into the table
        tableView.insertRows(at: [indexPath], with: .automatic)
    }
}
```

Deleting Rows

Here's the graphical interface for deleting a row, when the table view is in editing mode



this will be `UITableViewCell.EditingStyle.insert` or `UITableViewCellEditingStyle.delete`

The red minus sign will call a method on the table view, which we have to implement

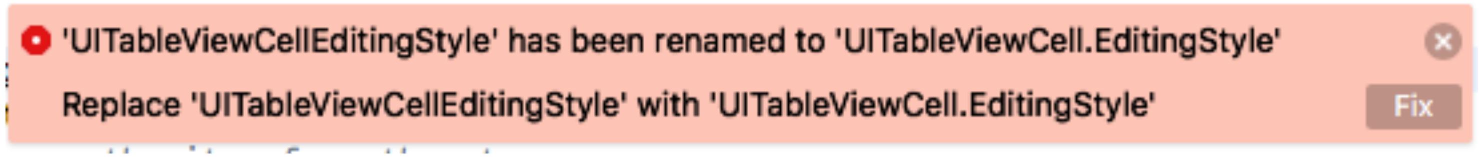
- the method it calls is `tableView(_:commit:forRow:)`
- this method is in the protocol `UITableViewDataSource`
- we must provide an implementation of this method to get the method call

We must also delete the item from the model (the `ItemStore`)

Update to the `UITableView` Interface

You'll get this error when you put in the code from p. 202

```
'UITableViewCellEditingStyle' has been renamed to
'UITableViewCell.EditingStyle'
```



'UITableViewCellEditingStyle' has been renamed to 'UITableViewCell.EditingStyle'

Replace 'UITableViewCellEditingStyle' with 'UITableViewCell.EditingStyle'

Note: the swipe-to-delete is not active when the table view is in editing mode

Deleting a Row

```
// for deleting a row
override func tableView(_ tableView: UITableView,
                        commit editingStyle: UITableViewCell.EditingStyle,
                        forRowAt indexPath: IndexPath) {
    // if the table view is asking to commit a delete command (this will give
    // us a chance to intercept the delete if we want to)
    if editingStyle == .delete {
        let item = itemStore.allItems[indexPath.row]
        // remove the item from the store
        itemStore.removeItem(item)

        // also remove that row from the table view with an animation
        tableView.deleteRows(at: [indexPath], with: .automatic)
    }
}
```

Moving Rows

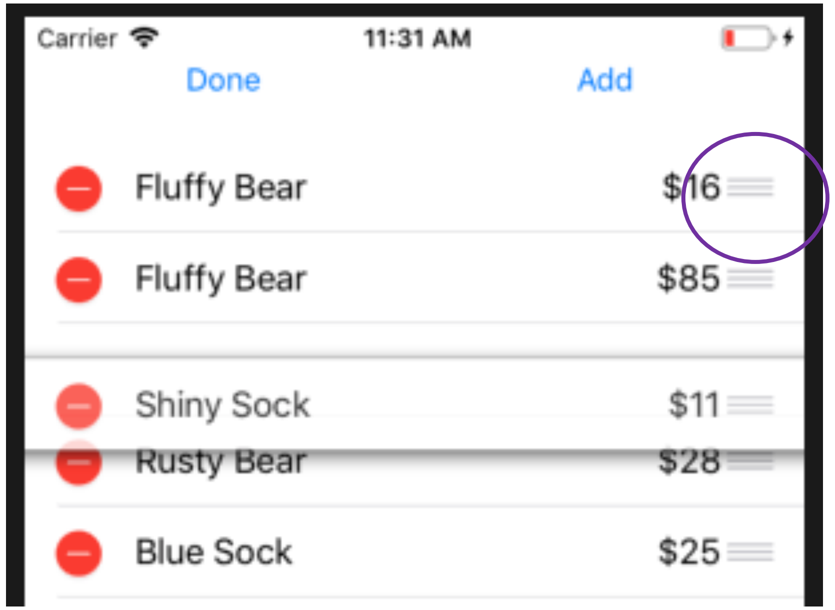
There is a straightforward interface to move a row

- the `UITableView` delegate will get a call to this method, which we must implement:

`tableView(_:moveRowAt:to:)`

We also implement a `moveItem(from:to:)` in the `ItemStore`

- to keep the view and the model consistent



if we implement this method, then the `UITableView` controller puts up the reordering controls whenever the table view enters editing mode

Moving Rows

In order to keep the model (the dataSource, which is the `itemStore`) and the view (the `UITableView`) in sync:

- we have to implement a method on `ItemStore` to rearrange items
- so that when the user drags a row to move it in the UI, we can then also change its position in the `itemStore`

```
func moveItem(from fromIndex: Int, to toIndex: Int) {
    if fromIndex == toIndex {
        return
    }

    // get reference to the object being moved so that we can reinsert it
    let movedItem = allItems[fromIndex]
    // rmove the item from array
    allItems.remove(at: fromIndex)
    // insert it in the array at the new position
    allItems.insert(movedItem, at: toIndex)
}
```

Displaying User Alerts

User alerts

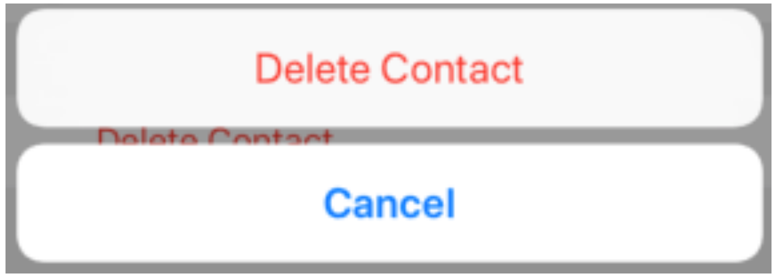
- often used to warn users that an important action is about to happen
- and to give them the opportunity to cancel that action

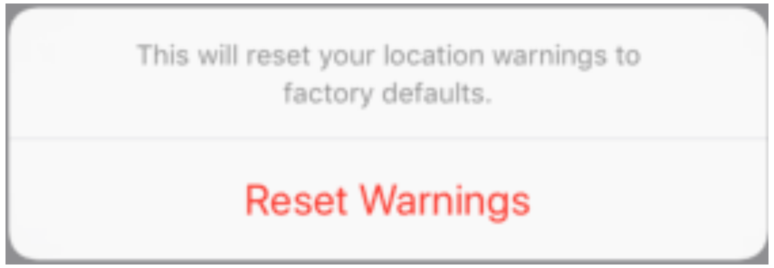
To display an alert, create an instance of `UIAlertController`

Here, we will put up an alert whenever the user asks to delete a row

`UIAlertController` Styles

`UIAlertController` has two styles:




This will reset your location warnings to factory defaults.

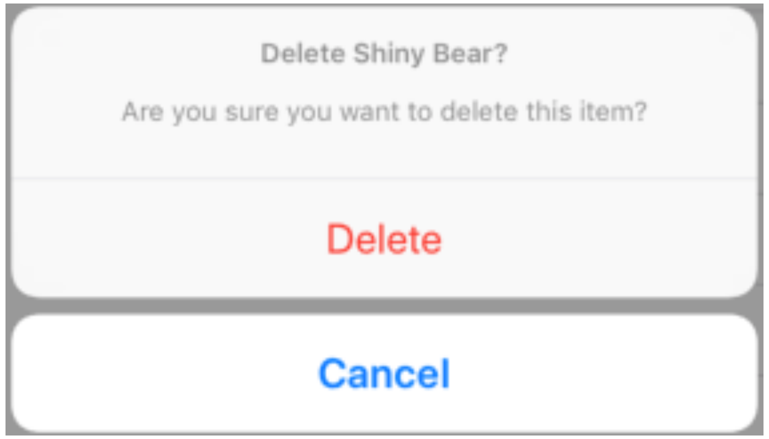
`.actionSheet`: presents the user with a list of actions from which to choose; use this if the action is not super critical

`.alert`: displays critical information to require the user to decide how to proceed

`UIAlertController`

Create and display an alert when the user chooses a row to delete

- the `UIAlertController` will then notify us when the user selects an action
- using a closure that we supply for that action



Delete Shiny Bear?

Are you sure you want to delete this item?

The Implementation

Here's the code:

```
// ask user to confirm
let title = "Delete \(item.name)?"
let message = "Are you sure you want to delete this item?"
let ac = UIAlertController(title: title, message: message, preferredStyle: .actionSheet)
let cancelAction = UIAlertAction(title: "Cancel", style: .cancel, handler: nil)
ac.addAction(cancelAction)
let deleteAction = UIAlertAction(title: "Delete", style: .destructive, handler: { (action) -> Void in
    // remove the item from the store
    self.itemStore.removeItem(item)
    //also remove that row from the table view, with animation
    tableView.deleteRows(at: [indexPath], with: .automatic)
})
ac.addAction(deleteAction)

// put up the controller as a modal view
present(ac, animated: true, completion: nil)
```

no further action needed

will be bright red text

iOS Bug in `UIAlertController`

When a `UIAlertController` is displayed, you might see this:

```
2020-10-15 08:26:44.360702-0400 HomePwnerCh12[10437:28672495] [LayoutConstraints] Unable to simultaneously
satisfy constraints.
Probably at least one of the constraints in the following list is one you don't want.
Try this:
(1) look at each constraint and try to figure out which you don't expect;
(2) find the code that added the unwanted constraint or constraints and fix it.
(
    "<NSLayoutConstraint:0x600002a5e990 UIView:0x7fa2ec61d2f0.width == - 16   (active)>"
)
```

This is evidently a (harmless) bug in iOS

- see https://stackoverflow.com/questions/55372093/uialertcontrollers-actionsheet-gives-constraint-error-on-ios-12-2-12-3