TableViews

Adding a tableview to the storyboard

1/ In the storyboard, add a tableView element

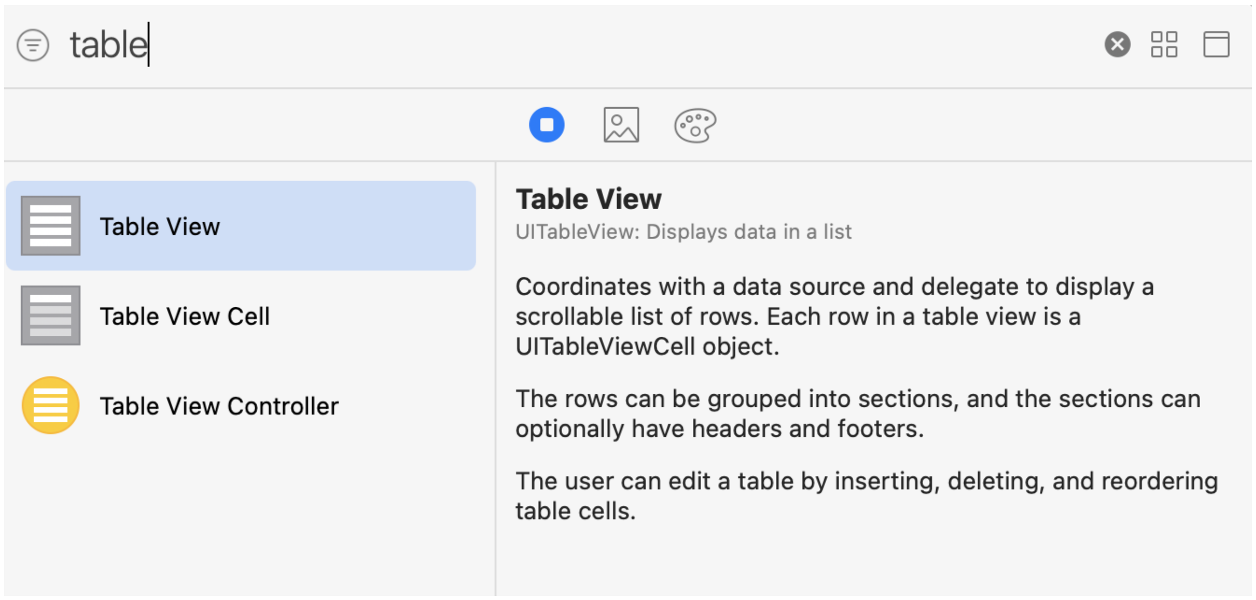


Table View

UITableView: Displays data in a list

Coordinates with a data source and delegate to display a scrollable list of rows. Each row in a table view is a UITableViewCell object.

The rows can be grouped into sections, and the sections can optionally have headers and footers.

The user can edit a table by inserting, deleting, and reordering table cells.

2/ In the storyboard, add a table view cell to your tableview

- Drag the cell onto the tableview in your storboard

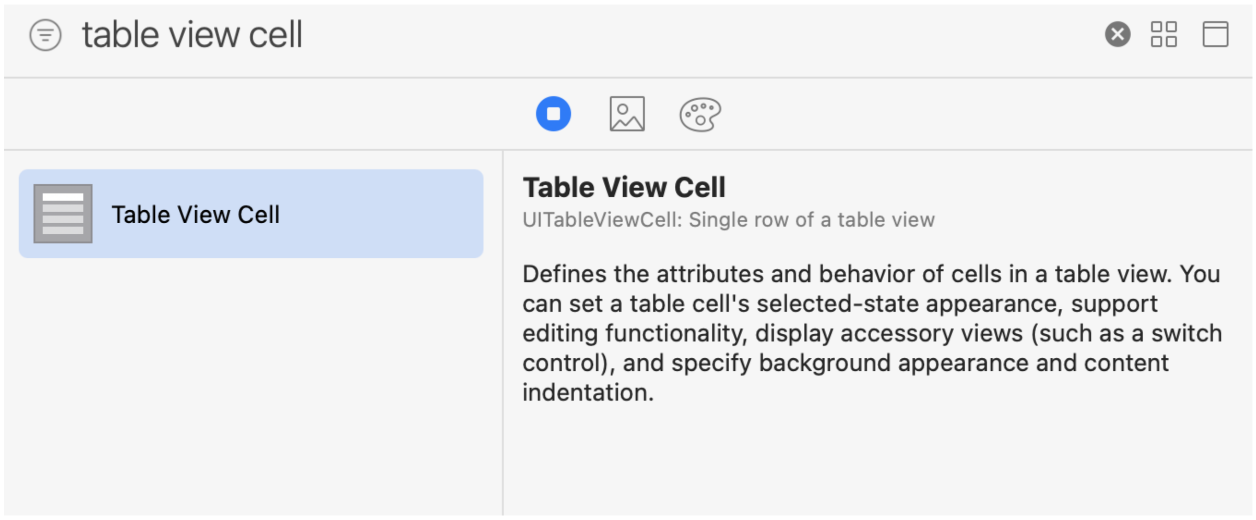


Table View Cell

UITableViewCell: Single row of a table view

Defines the attributes and behavior of cells in a table view. You can set a table cell's selected-state appearance, support editing functionality, display accessory views (such as a switch control), and specify background appearance and content indentation.

Expected result

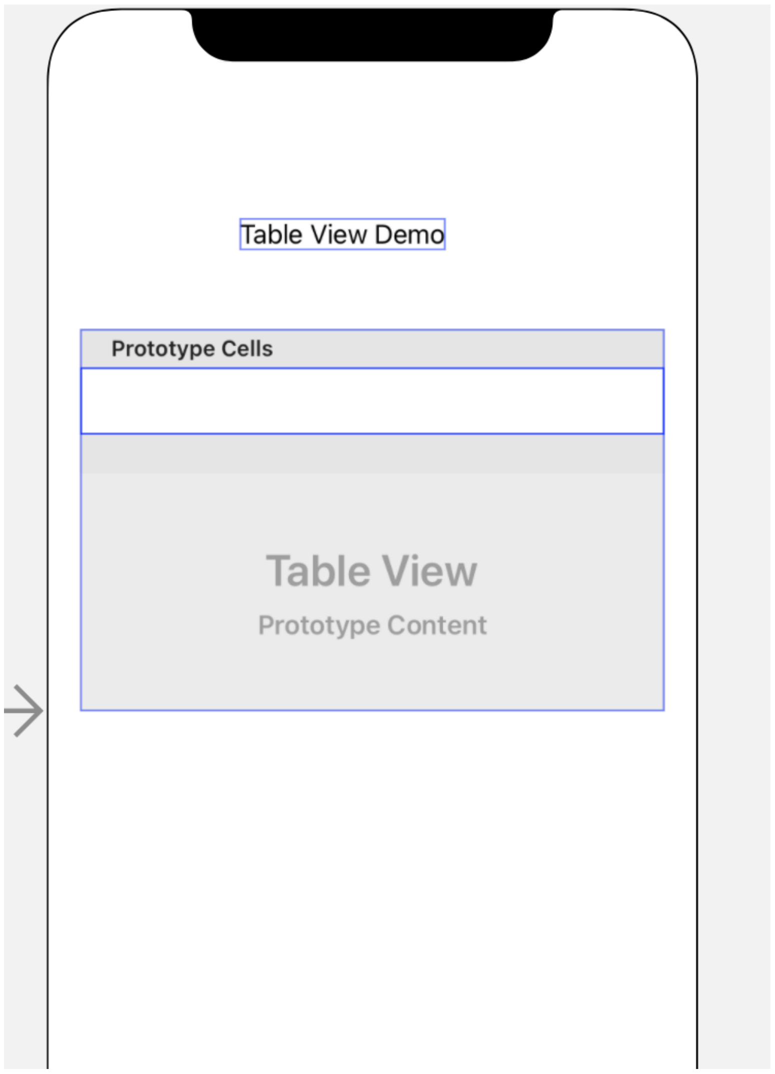


Table View Demo
Prototype Cells
Table View
Prototype Content

3/ Give your table view cell row an identifier

- Click on the table view row
- In the attributes inspector, go to Identifier
- Give the row an identifier (any name you want)

In the example below, I'm using the identifier of "myCell"

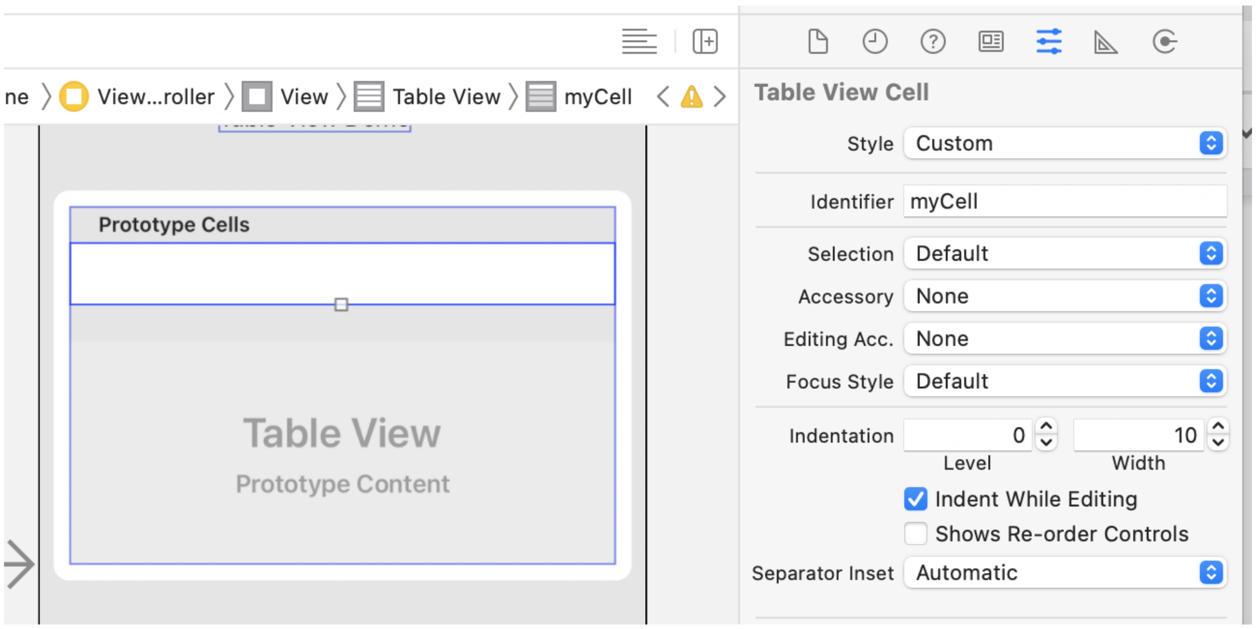


Table View Cell

Style Custom

Identifier myCell

Selection Default

Accessory None

Editing Acc. None

Focus Style Default

Indentation 0 Level 10 Width

☑ Indent While Editing

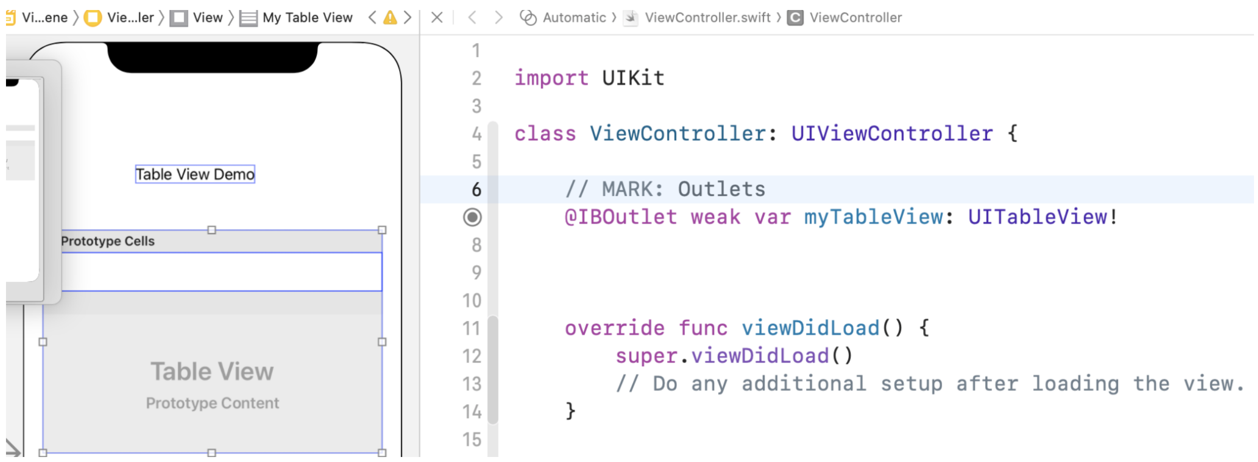
☐ Shows Re-order Controls

Separator Inset Automatic

4/ In your view controller, create an outlet for your TableView

- Ensure that you have *selected* the TABLEVIEW in the storyboard (not the cell)
- Give your outlet a name

In the below example, my outlet is called myTableView


Vi...ene 〉 Vie...ler 〉 View 〉 My Table View
Automatic 〉 ViewController.swift 〉 ViewController
Table View Demo
Prototype Cells
Table View
Prototype Content

```

import UIKit

class ViewController: UIViewController {

    // MARK: Outlets
    @IBOutlet weak var myTableView: UITableView!


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }

```

```
class ViewController: UIViewController {

   // MARK: Outlets
   @IBOutlet weak var myTableView: UITableView!

   override func viewDidLoad() {
      super.viewDidLoad()

    }


}
```

Configuring the ViewController

1/ Add the TableViewDelegate and TableViewDataSource to your class

- In ViewDidLoad(), Associate the tableview with the datasource and delegate

```
class ViewController: UIViewController, UITableViewDelegate,
UITableViewDataSource {

    // MARK: Outlets
    @IBOutlet weak var myTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // hook up your data source and delegate to your tableview
        myTableView.dataSource = self
        myTableView.delegate = self
    }


}
```

2/ Add your mandatory functions

```
class ViewController: UIViewController, UITableViewDelegate,
UITableViewDataSource {

    // MARK: Outlets
    @IBOutlet weak var myTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // hook up your data source and delegate to your tableview
        myTableView.dataSource = self
        myTableView.delegate = self
    }


    // MARK: Mandatory tableview functions
    func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) → Int {
        // TODO
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) → UITableViewCell {
        // TODO
    }

    func numberOfSections(in tableView: UITableView) → Int {
        // TODO
    }

}
```

3/ Add code to the mandatory functions

- For the :cellForRowAt function, the withIdentifier should be set to the cell row identifier you provided the storyboard

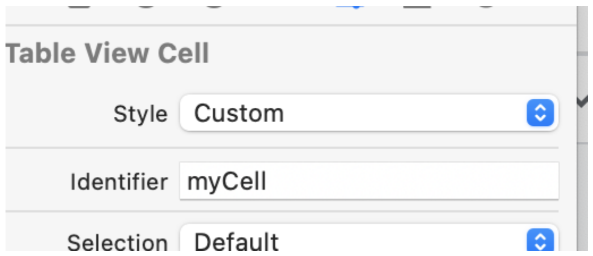


Table View Cell

Style: Custom

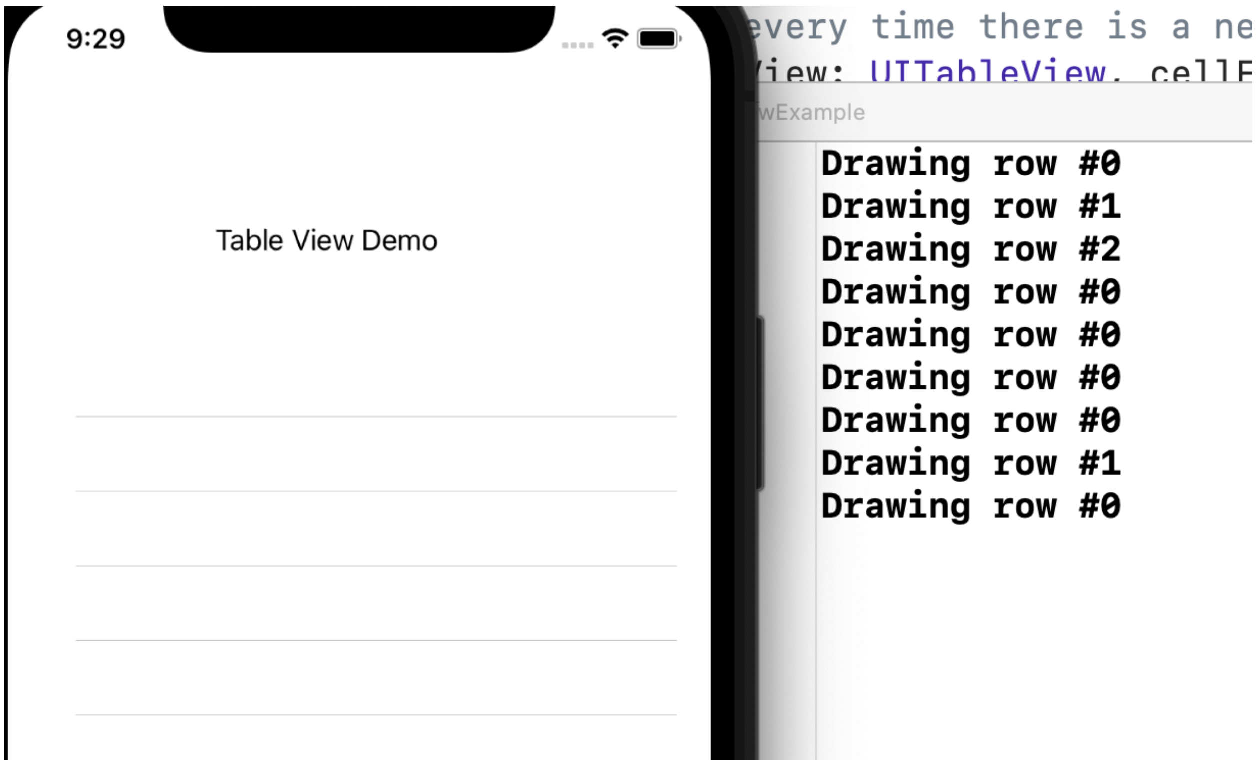
Identifier: myCell

Selection: Default

```
import UIKit

class ViewController: UIViewController, UITableViewDelegate,
UITableViewDataSource {

    // MARK: Outlets
    @IBOutlet weak var myTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()
        // hook up your data source and delegate to your tableview
        myTableView.dataSource = self
        myTableView.delegate = self
    }


    // MARK: Mandatory tableview functions

    // Specifies the number of items per section (number of rows in the
section)
    func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        // usually, this is dynamically calculated => It depends on how many
items you have in your data source
        return 3
    }

    // Controls what DATA is displayed in each row of the tableview
    // This row is called every time there is a new item to draw on the
tableview
    func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {


        // boilerplate code
        let cell = myTableView.dequeueReusableCell(withIdentifier: "myCell", 
for: indexPath)

        print("Drawing row #\(indexPath.row)")

        return cell
    }

    // how many sections do you want in your tableview
    func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

}
```

Expected result:

- You should get an empty tableview
- In the terminal, there will be some output
- As you scroll up and down in the table view, you may see some additional output in the terminal



```
Drawing row #0
Drawing row #1
Drawing row #2
Drawing row #0
Drawing row #0
Drawing row #0
Drawing row #0
Drawing row #1
Drawing row #0
```

Exercise: Update the :cellForRowAt function to display “Hello World”!

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

        // boilerplate code
        let cell = myTableView.dequeueReusableCell(withIdentifier: "myCell", for:
indexPath)

        print("Drawing row #\(indexPath.row)")
        // Add whatever text / content you want to display in the row
        cell.textLabel!.text = "Hello World!"

        return cell
    }
```

Expected result:

- Three rows, each with Hello World!

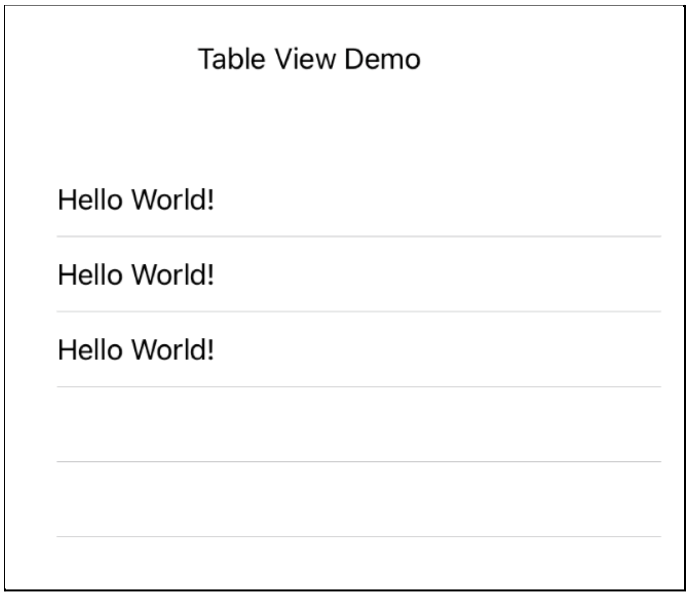


Table View Demo

Hello World!

Hello World!

Hello World!

Adding a Data Source to your Table View

Add a data source to your ViewController so we can display custom text in our tableview

- Array of strings (movies)
- Update the :numberOfRowsInSection function to be the # of items in your datasource
- Update the :cellForRowAt function to use the data in the array instead of our statically coded text

```
import UIKit

class ViewController: UIViewController, UITableViewDelegate,
UITableViewDataSource {

    // MARK: Outlets
    @IBOutlet weak var myTableView: UITableView!

    // MARK: Data Source For TableView
    var moviesList = ["Shangchi and the Ten Rings", "Spiderman: Far From
Home", "Dune", "Squid Game", "007: No Time to Die"]

    override func viewDidLoad() {
        super.viewDidLoad()
        // hook up your data source and delegate to your tableview
        myTableView.dataSource = self
        myTableView.delegate = self
    }


    // MARK: Mandatory tableview functions

    // Specifies the number of items per section (number of rows in the
section)
    func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        // dyanamically calculated => It depends on how many items you have
in your data source
        return moviesList.count
    }

    // Controls what DATA is displayed in each row of the tableview
    // This row is called every time there is a new item to draw on the
tableview
```

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

        // boilerplate code
        let cell = myTableView.dequeueReusableCell(withIdentifier: "myCell",
for: indexPath)

        // indexPath.row = the position of the row in the tableview
        // that is currently being rendered on the screen
        print("Drawing row #\(indexPath.row)")
        // Add whatever text / content you want to display in the row
        //cell.textLabel!.text = "Hello World!"
        cell.textLabel!.text = moviesList[indexPath.row]

        return cell
    }

    // how many sections do you want in your tableview
    func numberOfSections(in tableView: UITableView) -> Int {
        return 1
    }

}
```

Expected result

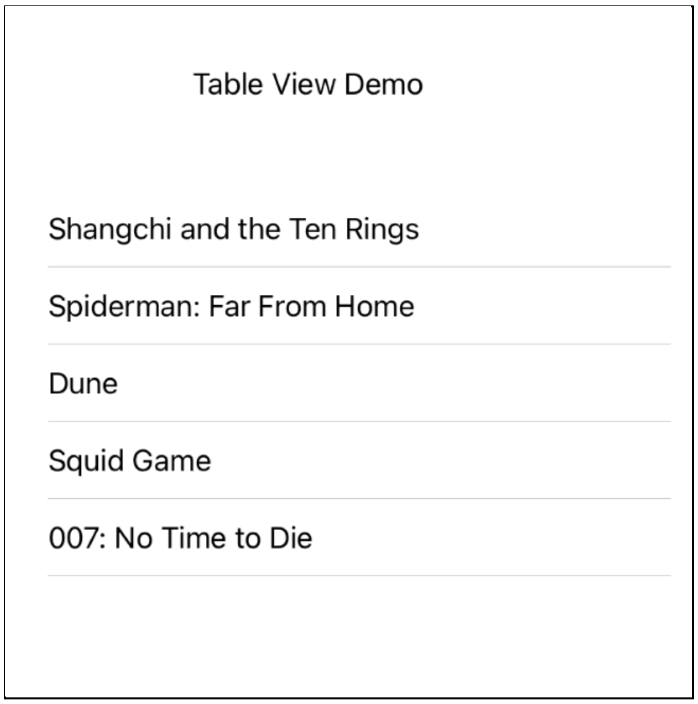


Table View Demo

Shangchi and the Ten Rings

Spiderman: Far From Home

Dune

Squid Game

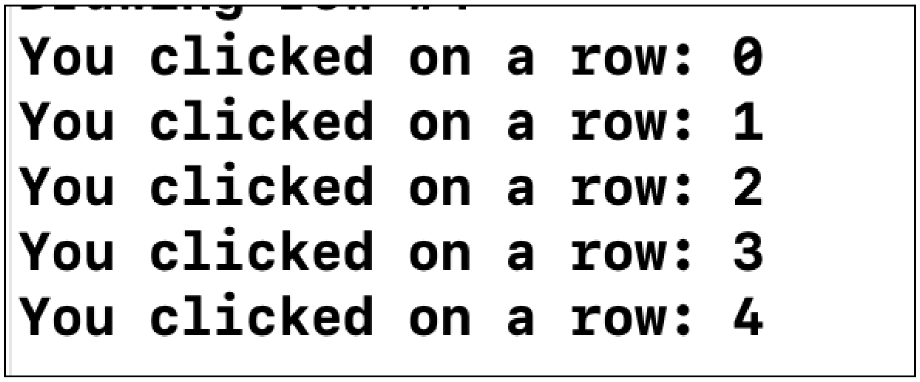
007: No Time to Die

Detect when someone clicks on your tableview row

```
func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath)
{
        // indexPath will return us the row that is currently being clicked on
        print("You clicked on a row: \(indexPath.row)")
    }
```

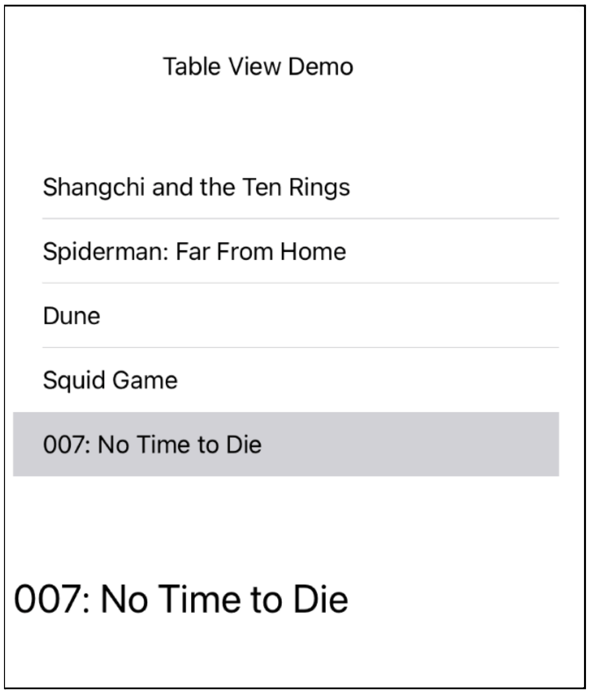
Expected result:

- When you click on a row in the tableview, the console should output the row and that was tapped



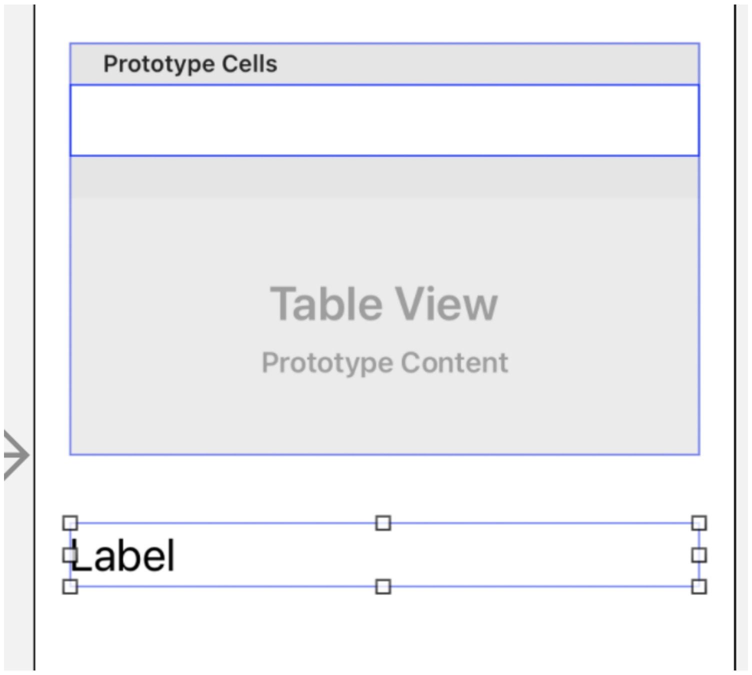
```
You clicked on a row: 0
You clicked on a row: 1
You clicked on a row: 2
You clicked on a row: 3
You clicked on a row: 4
```

Exercise: When person taps on row, update a label in the app



1/ Add a results label

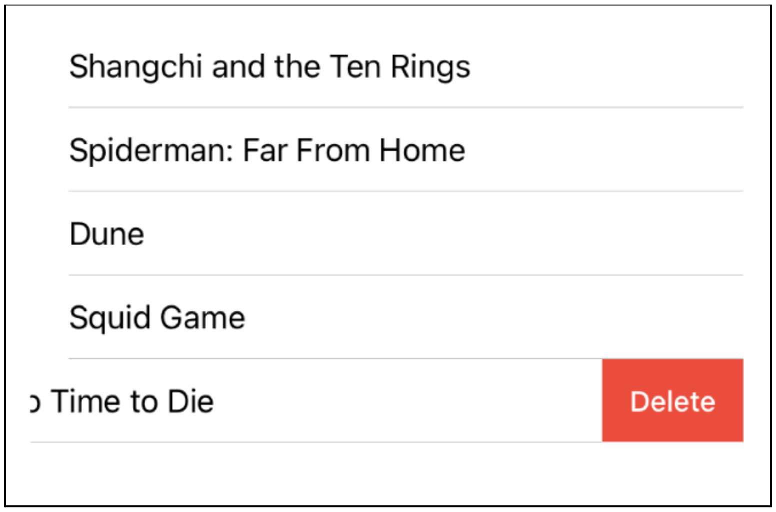
```
    @IBOutlet weak var resultsLabel: UILabel!
```



2/ Update :didSelectRowAt

```
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath) {
        // indexPath will return us the row that is currently being clicked on
        print("You clicked on a row: \(indexPath.row)")
        print("The item selected was: \(moviesList[indexPath.row])")
        // when you click on a row, navigate to another scren
        // when you click on a row, change something on the screen
        resultsLabel.text = moviesList[indexPath.row]
    }
```

Delete



Shangchi and the Ten Rings

Spiderman: Far From Home

Dune

Squid Game

ɔ Time to Die

```
func tableView(_ tableView: UITableView, commit editingStyle:
UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
        // detecting that person wants to delete
        if (editingStyle == UITableViewCell.EditingStyle.delete) {
            print("Person wants to delete the row")
        }
    }
```

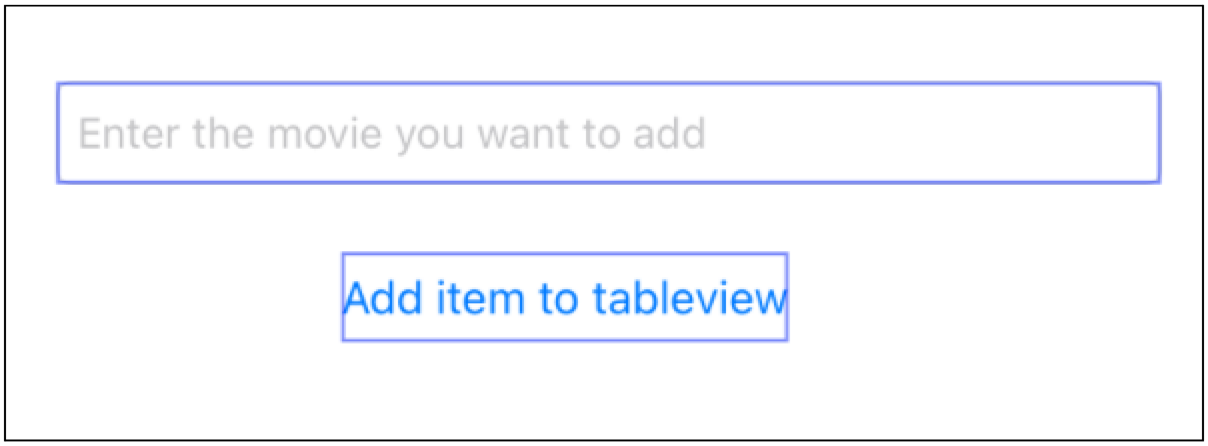
Deleting an item from the tableview

Update logic to delete item from data source and UI

```
   func tableView(_ tableView: UITableView, commit editingStyle:
UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
        // detecting that person wants to delete
        if (editingStyle == UITableViewCell.EditingStyle.delete) {
            print("Person wants to delete the row: \(indexPath.row)")
            // to actually REMOVE the item from the table view

            // You need to know the position of which row was selected

            // 1. Remove the selected item from the data source (array)
            moviesList.remove(at: indexPath.row)
            print("Updated list of movies: \(moviesList)")

            // 2. Remove the selected item from the UI
            myTableView.deleteRows(at: [indexPath], with:
UITableView.RowAnimation.automatic)

        }
    }
```

Dynamically add items to the tableview

1/ In the storyboard, add a textbox and label to accept user input



2/ Add an outlet and action for your textbox and label:

```
@IBOutlet weak var tbMovieName: UITextField!

@IBAction func btnAddPressed(_ sender: Any) {
      // 1. Get the content the user entered in the textbox

      // 2. Add that item to the datasource (moviesList)

      // 3. Update the user interface!!!!
      // 3a. Update the tableview
      // 3b. Clear the textbox and wait for new imput
   }
```

3/ In the button's action function, add code to:

- Add the item to the data source
- Refresh the tableView

```
@IBAction func btnAddPressed(_ sender: Any) {
    // 1. Get the content the user entered in the textbox
    let movieToAdd = tbMovieName.text!

    // 2. Add that item to the datasource (moviesList)
    moviesList.append(movieToAdd)
    print(moviesList)
    // 3. Update the user interface!!!!
    // 3a. Update the tableview
    myTableView.reloadData()

    // 3b. Clear the textbox and wait for new input
    tbMovieName.text = ""
}
```