

with *Swift*

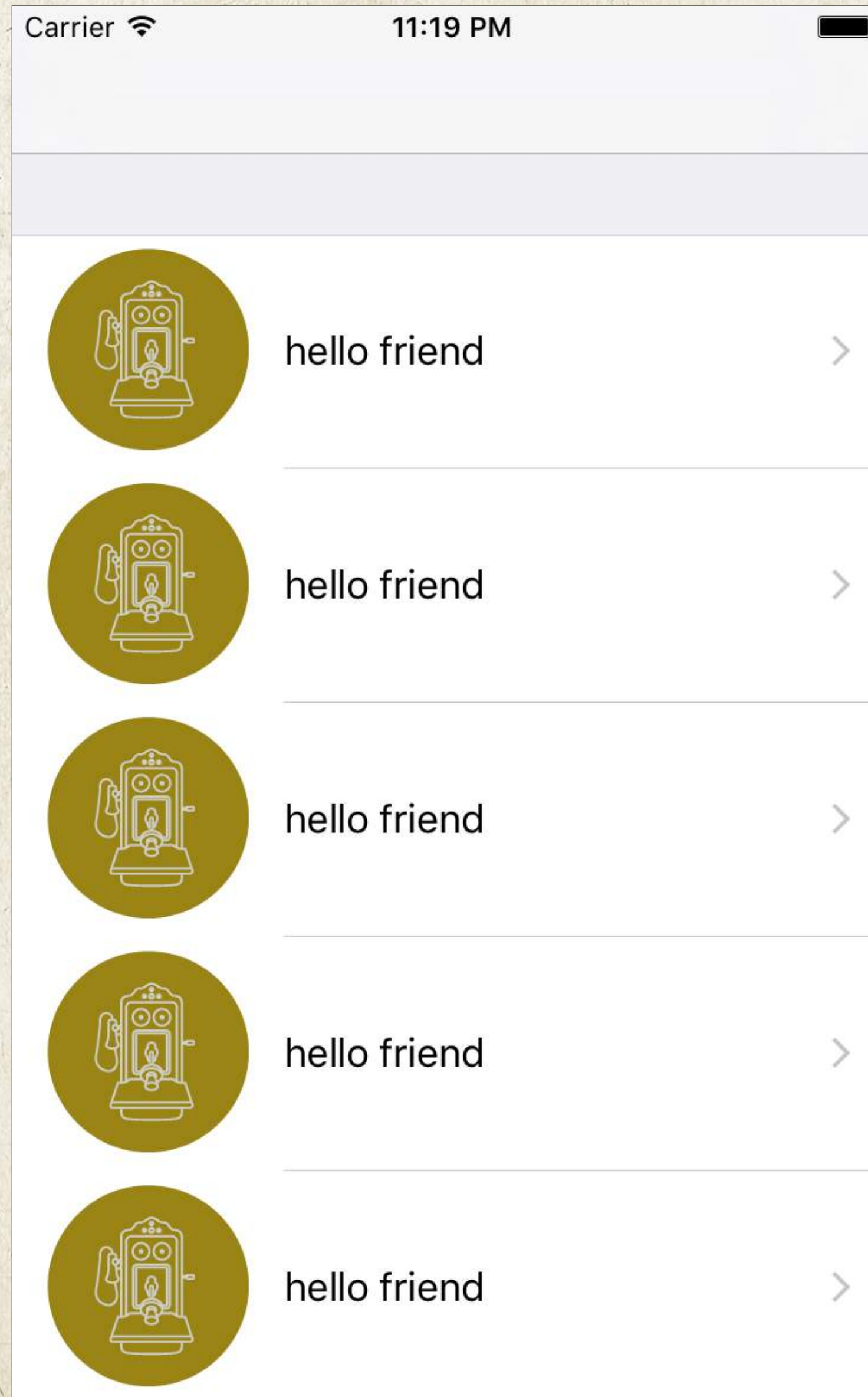
Level 5

Navigation

Section 2 - Displaying Dynamic Table Cell Data



Problem: Cell Text Is Hard-coded



Steps to Dynamic Data in Cells

Steps in ProductsTableViewController

1. Store multiple names in an Array instead of a String
2. Update the Table View required methods to use data from that Array



Arrays Can Store Multiple Values of a Single Type

ProductsTableViewController.swift

```
import UIKit
```

```
class ProductsTableViewController: UITableViewController {
```

```
    var productNames: [String]?
```

← This array is also optional

```
    ...  
}
```

↑ This array can only contain strings

Setting the Array Values in viewDidLoad

ProductsTableViewController.swift

```
import UIKit

class ProductsTableViewController: UITableViewController {

    var productNames: [String]?

    override func viewDidLoad() {
        super.viewDidLoad()

        productNames = ["1907 Wall Set", "1921 Dial Phone",
                        "1937 Desk Set", "1984 Motorola Portable"]

    }
    ...
}
```

we've set the array to have 4 values

Plan of Attack for Updating the Cells

We have to update these 2 required table view functions to use data from the array instead of just hard-coded strings.

ProductsTableViewController.swift

```
import UIKit
```

```
class ProductsTableViewController: UITableViewController {
```

```
    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int  
    { ... }
```

↑ update this function to return the number of items in the array

```
    override func tableView(tableView: UITableView,  
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell  
    { ... }  
    ...  
}
```

↑ update this function to use the one of the names in the array for each cell

A Problem Updating the Number of Rows

ProductsTableViewController.swift

```
import UIKit

class ProductsTableViewController: UITableViewController {

    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return 5
        return productNames?.count
    }

    override func tableView(tableView: UITableView,
        cellForRowAtIndex indexPath: NSIndexPath) -> UITableViewCell
    { ... }
    ...
}
```

count returns the number of items in the array

we get this error

Value of optional type 'Int?' not unwrapped; did you mean to use '!' or '?'?

The Problem is We're Returning an Optional Int

ProductsTableViewController.swift

```
import UIKit
```

```
class ProductsTableViewController: UITableViewController {
```

```
    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int  
    {  
        return 5  
        return productNames?.count  
    }
```

If productNames doesn't exist,
we'd be returning an optional Int

```
    override func tableView(tableView: UITableView,  
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell  
    { ... }  
    ...  
}
```

This function wants us
to return a regular Int,
not an optional one

Value of optional type 'Int?' not unwrapped; did you mean to use '!' or '?'?

Fixing the “Type not unwrapped” Error With if let


ProductsTableViewController.swift

```
import UIKit

class ProductsTableViewController: UITableViewController {

    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return 5
        if let pNames = productNames {
            return pNames.count
        }
    }

    override func tableView(tableView: UITableView,
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell
    { ... }
    ...
}
```



If the array exists, create a non-optional version of it

Get the count of the non-optional version of the array

Return Zero Rows If the Array Doesn't Exist

ProductsTableViewController.swift

```
import UIKit

class ProductsTableViewController: UITableViewController {

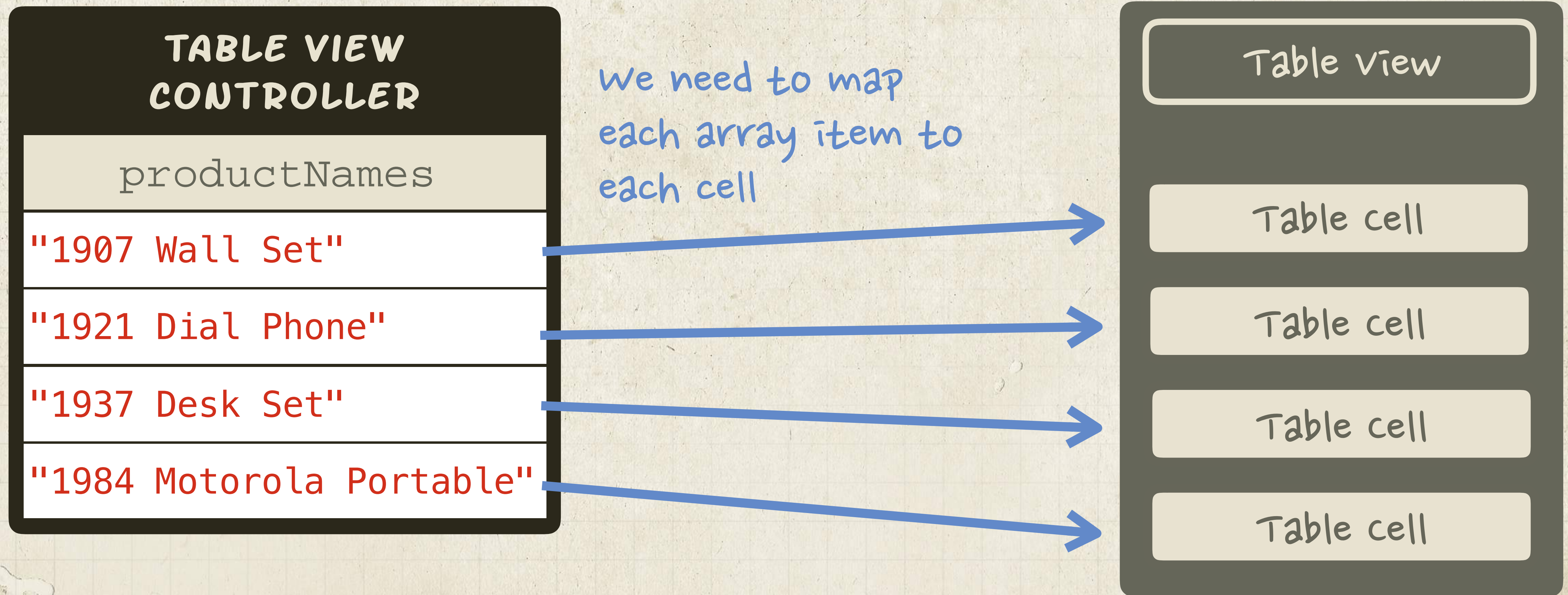
    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return 5
        if let pNames = productNames {
            return pNames.count
        }
        return 0
    }

    override func tableView(tableView: UITableView,
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell
    { ... }
    ...
}
```

← If productNames doesn't exist, we still need to return zero rows

Mapping Array Values to Table Cells

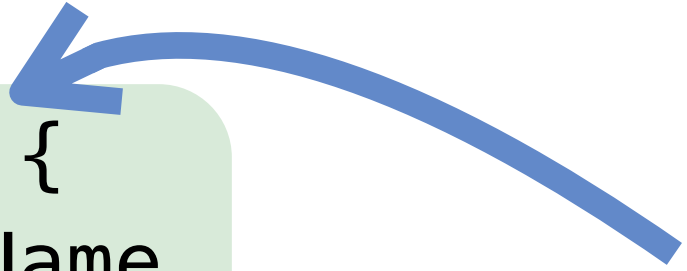
Each item in the array will correspond to each row in the table.



Unwrap the Optional productName Variable

ProductsTableViewController.swift

```
class ProductsTableViewController: UITableViewController {  
  
    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int  
    { ... }  
  
    override func tableView(tableView: UITableView,  
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell  
    {  
        let cell = tableView.dequeueReusableCellWithIdentifier("ProductCell", forIndexPath: indexPath)  
  
        let productName = // set the product name  
  
        if let pName = productName {  
            cell.textLabel?.text = pName  
        }  
        return cell  
    }  
    ...  
}
```



The product name will be an optional, so we need to unwrap it before setting the label to that text

Reading Values From an Array

Each value in the array can be accessed by typing a number between square brackets after the variable name.

TABLE VIEW CONTROLLER

productNames

Index 0

"1907 Wall Set"

Index 1

"1921 Dial Phone"

Index 2

"1937 Desk Set"

Index 3

"1984 Motorola Portable"

```
print(productNames[0])
```



"1907 Wall Set"

```
print(productNames[3])
```

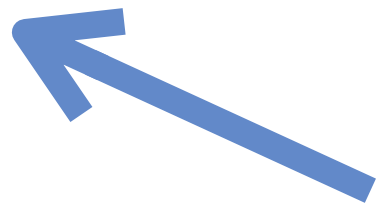


"1984 Motorola Portable"

Plan of Attack for Updating the Cells

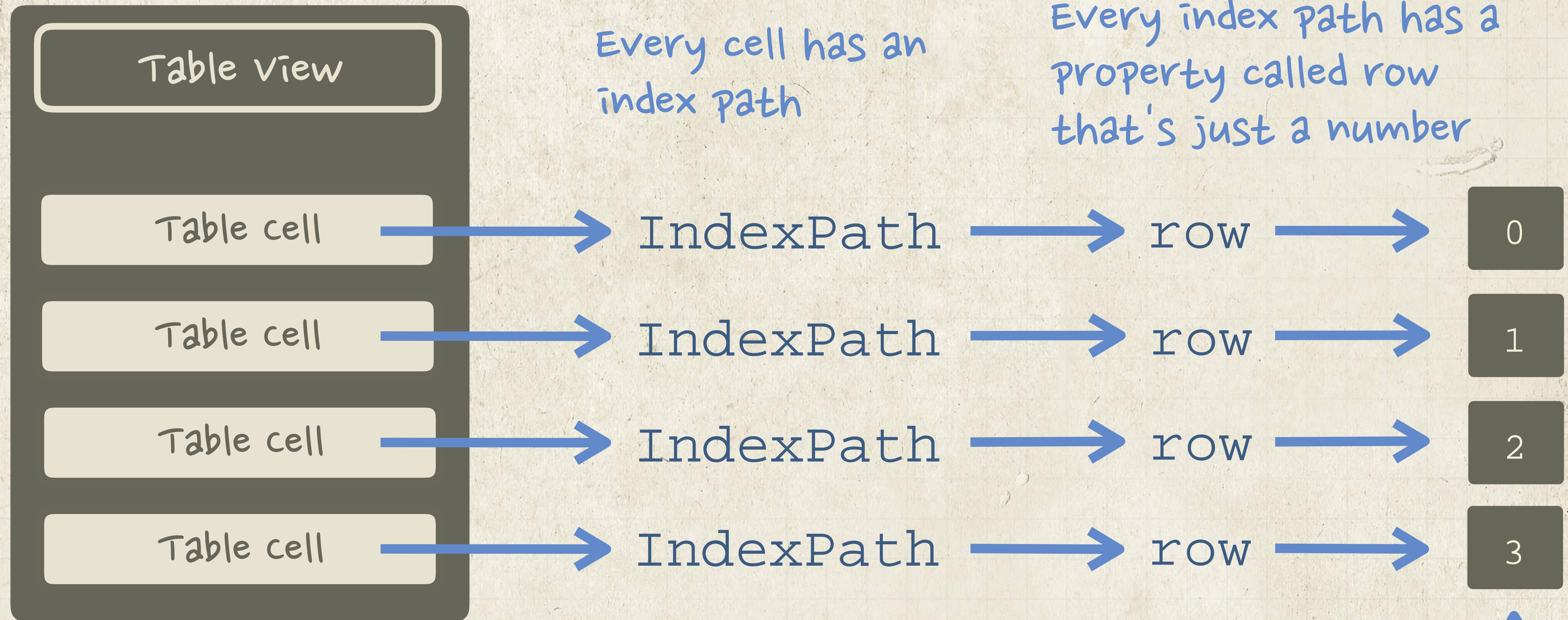
ProductsTableViewController.swift

```
class ProductsTableViewController: UITableViewController {  
  
    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int  
    { ... }  
  
    override func tableView(tableView: UITableView,  
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell  
    {  
        let cell = tableView.dequeueReusableCellWithIdentifier("ProductCell", forIndexPath: indexPath)  
  
        let productName = productNames[0]  
  
        if let pName = productName {  
            cell.textLabel?.text = pName  
        }  
        return cell  
    }  
    ...  
}
```



This will get the first item in the array every time, but we want this number to be different for each cell

Finding a Cell With an Index Path



Every cell has an index path


Every index path has a property called row that's just a number

Rows start at zero, just like array values!

Using the indexPath to Access an Array Item

ProductsTableViewController.swift

```
class ProductsTableViewController: UITableViewController {  
  
    override func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int  
    { ... }  
  
    override func tableView(tableView: UITableView,  
        cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell  
    {  
        let cell = tableView.dequeueReusableCellWithIdentifier("ProductCell", forIndexPath: indexPath)  
  
        let productName = productNames?[indexPath.row]  
  
        if let pName = productName {  
            cell.textLabel?.text = pName  
        }  
        return cell  
    }  
    ...  
}
```



Now we've got a copy of the name that should be displayed in this cell

Demo: Dynamic Cell Text

