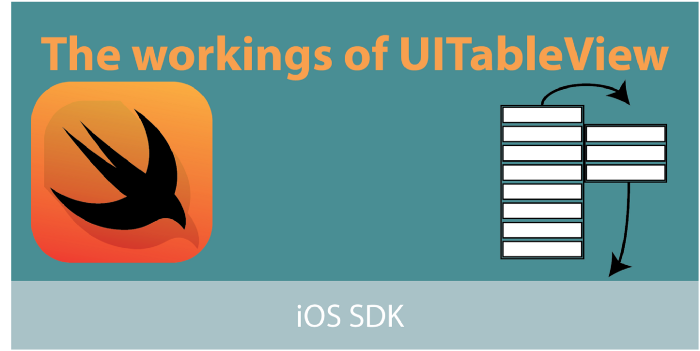
The hidden workings of UITableView

[](https://medium.com/@stevenpcurtis.sc?source=post_page-----73feb60a13c3----------------------)

[Steven Curtis](https://medium.com/@stevenpcurtis.sc?source=post_page-----73feb60a13c3----------------------)

[May 27](https://medium.com/@stevenpcurtis.sc/the-hidden-workings-of-uitableview-73feb60a13c3?source=post_page-----73feb60a13c3----------------------) · 3 min read





UITableViewCell has some interesting workings underneath the hood, giving a fantastic opportunity to think about how they work and some of the internal workings behind some of the basic iOS features.rerequisites:

* Some basic understanding of Xcode, Swift and creating mobile Apps.

**Terminology**

UITableViewController: A view controller that specializes in managing a table view

UITableView: A view that presents data using rows arranging in a single column

UITableViewDataSource: The methods adopted by the object you use to manage data and provide cells for a table view

UITableViewCell: A visual representatation of a single row in a table view

**The problem**

If you naively created a tableview and did not reuse cells, you would waste memory and potentially crash your all — not great for your users!

When you implement the tableview with the traditional **dequeueReusableCell(withIdentifier:for:)** you can encounter further problems because the cells are reused.

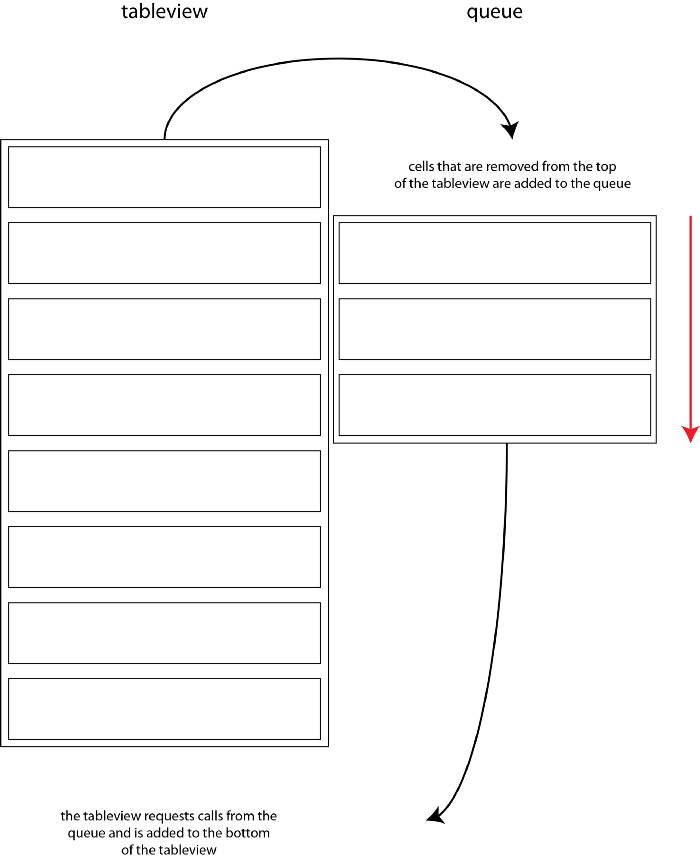
These are covered (and problems solved) in the practical examples section below. However we first run through the theory

**The theory**

If you have a tableview with 100,000 rows they would all need to be stored in memory (at the same time). In the practical example, I found this took nearly 1GB of memory. However, it does not end there since a table may have more rows, and you would need to make sure that your App is not closed due to using too much memory.

As a solution you use a queue of cells. When you scroll through a table





The iOS reusable cell queue mechanism

When the cell is scrolled outside the visible area of the tableview, it is placed within the queue. When cells are taken from the queue it is sent to the cellForRowAt: method and the appropriate data is (re)entered into the cell.

Some we might be reentering data into a used cell Apple recommend that we reset attibutes of the cell that are not related content through **prepareForReuse** and **tableView(\_:cellForRowAt:)** should reset all of the content when reusing a cell.

**Practical examples**

The git link implements the following code: <https://github.com/stevencurtis/HiddenWorkingsUITableview>

**The poor implementation**

The poorest implementation of a tableview does not reuse cells at all:

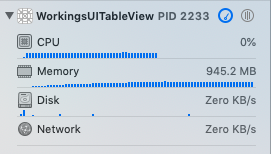
|  |  |
| --- | --- |
|  | class PoorImplementationViewController: UIViewController { |
|  |  |
|  | @IBOutlet weak var tableView: UITableView! |
|  |  |
|  | var cells: [UITableViewCell] = [] |
|  |  |
|  | override func viewDidLoad() { |
|  | super.viewDidLoad() |
|  |  |
|  | for i in 0...100000 { |
|  | let cell = UITableViewCell() |
|  | cell.textLabel?.text = i.description |
|  | cell.imageView?.image = UIImage(named: "placeholder.png") |
|  | cells.append(cell) |
|  | } |
|  | tableView.dataSource = self |
|  | tableView.delegate = self |
|  | } |
|  |  |
|  | } |
|  |  |
|  | extension PoorImplementationViewController: UITableViewDelegate { |
|  |  |
|  | } |
|  |  |
|  | extension PoorImplementationViewController: UITableViewDataSource { |
|  | func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int { |
|  | return cells.count |
|  | } |
|  |  |
|  | func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell { |
|  | let cell = cells[indexPath.row] |
|  | return cell |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/stevencurtis/440d2cea99aba4ac600f96e0bfb49bc3/raw/a813dd4364f89981e2a72cb043d79fbe297d605b/PoorImplementationViewController)

[**PoorImplementationViewController**](https://gist.github.com/stevencurtis/440d2cea99aba4ac600f96e0bfb49bc3#file-poorimplementationviewcontroller) hosted with ❤ by [**GitHub**](https://github.com/)

and, spectacularly, the code uses 1GB of memory.





This is obviously untenable as the the App may be closed due to the large memory footprint of this implementation.

**A reuse implementation**

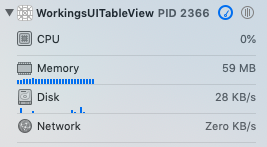
|  |  |
| --- | --- |
|  | // |
|  | // ReuseViewController.swift |
|  | // WorkingsUITableView |
|  | // |
|  | // Created by Steven Curtis on 25/05/2019. |
|  | // Copyright © 2019 Steven Curtis. All rights reserved. |
|  | // |
|  |  |
|  | import UIKit |
|  |  |
|  | class ReuseViewController: UIViewController { |
|  |  |
|  | @IBOutlet weak var tableView: UITableView! |
|  |  |
|  | var data : [String] = [] |
|  |  |
|  | var task: URLSessionDownloadTask! |
|  | var session: URLSession! |
|  |  |
|  | override func viewDidLoad() { |
|  | super.viewDidLoad() |
|  |  |
|  | for i in 0...100000 { |
|  | data.append(i.description) |
|  | } |
|  |  |
|  | tableView.dataSource = self |
|  | tableView.delegate = self |
|  | tableView.register(ReusableTableViewCell.self, forCellReuseIdentifier: "ReusableTableViewCell") |
|  |  |
|  | } |
|  | } |
|  |  |
|  | extension ReuseViewController: UITableViewDelegate { |
|  |  |
|  | } |
|  |  |
|  | extension ReuseViewController: UITableViewDataSource { |
|  | func tableView(\_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int { |
|  | return data.count |
|  | } |
|  |  |
|  | func tableView(\_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell { |
|  | let cell = tableView.dequeueReusableCell(withIdentifier: "ReusableTableViewCell", for: indexPath) |
|  | cell.imageView?.image = UIImage(named: "placeholder.png") |
|  | cell.textLabel?.text = data[indexPath.row] |
|  | return cell |
|  | } |
|  |  |
|  |  |
|  | } |

[**view raw**](https://gist.github.com/stevencurtis/f53f6928c7a70b7fddbf9babfcd21319/raw/00fb84e1a7d9b8929a109512709eaa9df561a676/ReuseImplementationViewController)

[**ReuseImplementationViewController**](https://gist.github.com/stevencurtis/f53f6928c7a70b7fddbf9babfcd21319#file-reuseimplementationviewcontroller) hosted with ❤ by [**GitHub**](https://github.com/)

This brings the memory usage down greatly.

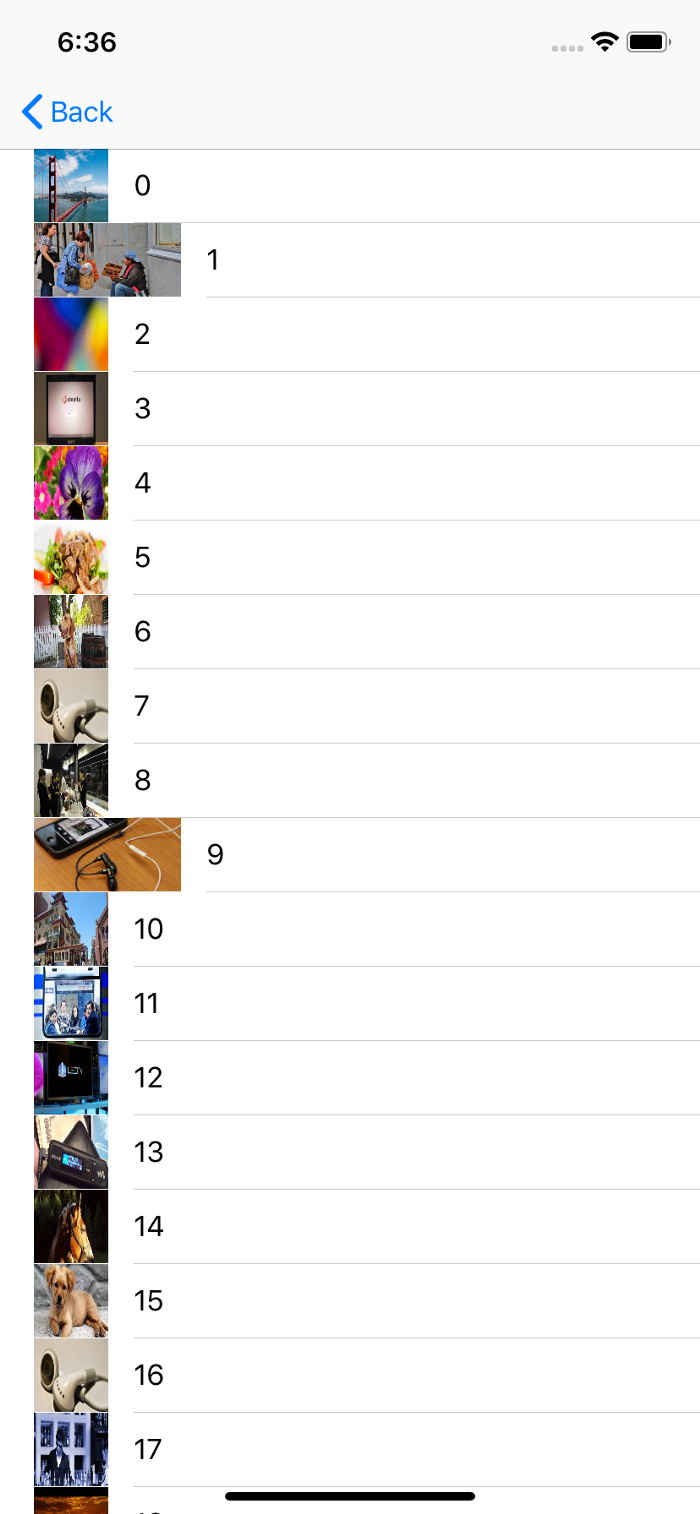




**The problem**

When the images are loaded into reusable cells they can be not only displayed as the wrong size (in an implementation where we download images from an API):





But they can actually display the wrong image as it is the reused image!

To implement this we need to implement a custom cell using a NIB which requires a slightly different registration:

***let*** *cellNib = UINib(nibName: “ReusableTableViewCell”, bundle:* ***nil****)*

*tableView.register(cellNib, forCellReuseIdentifier: “ReusableTableViewCell”)*

we can then download the image from within the cell and, most importantly use **prepareForReuse()**. It can be argued that prepareForReuse should only reset the attributes of the cell that are not content related, but the following code is only given for purposed of this post.

|  |  |
| --- | --- |
|  | class ReusableTableViewCell: UITableViewCell { |
|  | var task: URLSessionDownloadTask! |
|  | var session: URLSession! |
|  |  |
|  | override func awakeFromNib() { |
|  | super.awakeFromNib() |
|  | session = URLSession.shared |
|  | task = URLSessionDownloadTask() |
|  | downloadImage() |
|  | } |
|  |  |
|  | func downloadImage() { |
|  | let url:URL! = URL(string: "https://lorempixel.com/400/200/") |
|  | task = session.downloadTask(with: url, completionHandler: { (location, response, error) -> Void in |
|  | if let data = try? Data(contentsOf: url){ |
|  | DispatchQueue.main.async(execute: { () -> Void in |
|  | let img:UIImage! = UIImage(data: data) |
|  | self.imageView?.image = img |
|  | }) |
|  | } |
|  | }) |
|  | task.resume() |
|  |  |
|  |  |
|  | self.imageView?.image = UIImage(named: "placeholder.png") |
|  | } |
|  |  |
|  | override func setSelected(\_ selected: Bool, animated: Bool) { |
|  | super.setSelected(selected, animated: animated) |
|  | } |
|  |  |
|  | override func prepareForReuse() { |
|  | self.imageView?.image = UIImage(named: "placeholder.png") |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/stevencurtis/6ef4ec3eeda927f035062a13ea6ee5f2/raw/327d932cba50670f3e1998ac664e7d0ce7e1a521/ReusableTableViewCell)

[**ReusableTableViewCell**](https://gist.github.com/stevencurtis/6ef4ec3eeda927f035062a13ea6ee5f2#file-reusabletableviewcell) hosted with ❤ by [**GitHub**](https://github.com/)

Here is that link to the Git repo:

**[stevencurtis/HiddenWorkingsUITableview](https://github.com/stevencurtis/HiddenWorkingsUITableview?source=post_page-----73feb60a13c3----------------------)**

[HiddenWorkingsUITableview. Contribute to stevencurtis/HiddenWorkingsUITableview development by creating an account on…](https://github.com/stevencurtis/HiddenWorkingsUITableview?source=post_page-----73feb60a13c3----------------------)

[github.com](https://github.com/stevencurtis/HiddenWorkingsUITableview?source=post_page-----73feb60a13c3----------------------)

**Want to get in contact? Have a question? Hit me up on Twitter:**