```swift
//
//  MasterViewController.swift
//  TwitterSearches
//
//  Created by David Hincapie on 11/10/15.
//  Copyright © 2015 David Hincapie. All rights reserved.
//

import UIKit

class MasterViewController: UITableViewController,
ModelDelegate {

    // DetailViewController contains UIWebView to display search results
    var detailViewController: DetailViewController? = nil

    var model: Model! = nil // manages the app's data
    let twitterSearchURL = "https://mobile.twitter.com/search/?q="

    // conform to ModelDelegate protocol; updates view when model changes
    func modelDataChanged() {
        tableView.reloadData() // reload the UITableView
    }

    // configure popover for UITableView on iPad
    override func awakeFromNib() {
        super.awakeFromNib()
        if UIDevice.currentDevice().userInterfaceIdiom == .Pad {
            self.clearsSelectionOnViewWillAppear = false
            self.preferredContentSize =
                CGSize(width: 320.0, height: 600.0)
        }
    }

    // called after the view loads for further UI configuration
    override func viewDidLoad() {
        super.viewDidLoad()

        // set up left and right UIBarButtonItems
        self.navigationItem.leftBarButtonItem = self.editButtonItem()
        let addButton = UIBarButtonItem(barButtonSystemItem: .Add,
            target: self, action: "addButtonPressed:")
        self.navigationItem.rightBarButtonItem = addButton

        if let split = self.splitViewController {
            let controllers = split.viewControllers
            self.detailViewController =
                controllers[controllers.count-1] as? DetailViewController
        }

        model = Model(delegate: self) // create the Model
        model.synchronize() // tell model to sync its data
    }

    // displays a UIAlertController to obtain new search from user
    func addButtonPressed(sender: AnyObject) {
        displayAddEditSearchAlert(isNew: true, index: nil)
    }
```

```
// handles long press for editing or sharing a search
func tableViewCellLongPressed(
    sender: UILongPressGestureRecognizer) {
        if sender.state == UIGestureRecognizerState.Began &&
            !tableView.editing {
                let cell = sender.view as! UITableViewCell // get cell

                if let indexPath = tableView.indexPathForCell(cell) {
                    displayLongPressOptions(indexPath.row)
                }
        }
}

// displays the edit/share options
func displayLongPressOptions(row: Int) {
    // create UIAlertController for user input
    let alertController = UIAlertController(title: "Options",
        message: "Edit or Share your search",
        preferredStyle: UIAlertControllerStyle.Alert)

    // create Cancel action
    let cancelAction = UIAlertAction(title: "Cancel",
        style: UIAlertActionStyle.Cancel, handler: nil)
    alertController.addAction(cancelAction)

    let editAction = UIAlertAction(title: "Edit",
        style: UIAlertActionStyle.Default,
        handler: {(action) in
            self.displayAddEditSearchAlert(isNew: false, index: row)})
    alertController.addAction(editAction)

    let shareAction = UIAlertAction(title: "Share",
        style: UIAlertActionStyle.Default,
        handler: {(action) in self.shareSearch(row)})
    alertController.addAction(shareAction)
    presentViewController(alertController, animated: true,
        completion: nil)
}

// displays add/edit dialog
func displayAddEditSearchAlert(isNew  isNew: Bool, index: Int?) {
    // create UIAlertController for user input
    let alertController = UIAlertController(
        title: isNew ? "Add Search" : "Edit Search",
        message: isNew ? "" : "Modify your query",
        preferredStyle: UIAlertControllerStyle.Alert)

    // create UITextFields in which user can enter a new search
    alertController.addTextFieldWithConfigurationHandler(
        {(textField) in
            if isNew {
                textField.placeholder = "Enter Twitter search query"
            } else {
                textField.text = self.model.queryForTagAtIndex(index!)
            }
    })

    alertController.addTextFieldWithConfigurationHandler(
        {(textField) in
```

```
if isNew {
                textField.placeholder = "Enter Twitter search query"
            } else {
                textField.text = self.model.queryForTagAtIndex(index!)
            }
    })

    alertController.addTextFieldWithConfigurationHandler(
        {(textField) in
            if isNew {
                textField.placeholder = "Tag your query"
            } else {
                textField.text = self.model.tagAtIndex(index!)
                textField.enabled = false
                textField.textColor = UIColor.lightGrayColor()
            }
    })

    // create Cancel action
    let cancelAction = UIAlertAction(title: "Cancel",
        style: UIAlertActionStyle.Cancel, handler: nil)
    alertController.addAction(cancelAction)

    let saveAction = UIAlertAction(title: "Save",
        style: UIAlertActionStyle.Default,
        handler: {(action) in
            let query =
            (alertController.textFields![0] ).text
            let tag =
            (alertController.textFields![1] ).text

            // ensure query and tag are not empty
            if !query!.isEmpty && !tag!.isEmpty {
                self.model.saveQuery(
                    query!, forTag: tag!, syncToCloud: true)

                if isNew {
                    let indexPath =
                    NSIndexPath(forRow: 0, inSection: 0)
                    self.tableView.insertRowsAtIndexPaths([indexPath],
                        withRowAnimation: .Automatic)
                }
            }
    })
    alertController.addAction(saveAction)

    presentViewController(alertController, animated: true,
        completion: nil)
}

// displays share sheet
func shareSearch(index: Int) {
    let message = "Check out the results of this Twitter search"
    let urlString = twitterSearchURL +
        urlEncodeString(model.queryForTagAtIndex(index)!)
    let itemsToShare = [message, urlString]

    // create UIActivityViewController so user can share search
    let activityViewController = UIActivityViewController(
        activityItems: itemsToShare, applicationActivities: nil)
```

```
presentViewController(activityViewController,
            animated: true, completion: nil)
    }

    // called when app is about to seque from
    // MainViewController to DetailViewController
    override func prepareForSegue(segue: UIStoryboardSegue,
        sender: AnyObject?) {

            if segue.identifier == "showDetail" {
                if let indexPath = self.tableView.indexPathForSelectedRow {
                    let controller = (segue.destinationViewController as!
                        UINavigationController).topViewController as!
                    DetailViewController

                    // get query String
                    let query =
                    String(model.queryForTagAtIndex(indexPath.row)!)

                    // create NSURL to perform Twitter Search
                    controller.detailItem = NSURL(string: twitterSearchURL +
                        urlEncodeString(query))
                    controller.navigationItem.leftBarButtonItem =
                        self.splitViewController?.displayModeButtonItem()
                    controller.navigationItem.leftItemsSupplementBackButton =
                    true
                }
            }
    }

    // returns a URL encoded version of the query String
    func urlEncodeString(string: String) -> String {
        return string.stringByAddingPercentEncodingWithAllowedCharacters(
            NSCharacterSet.URLQueryAllowedCharacterSet())!
    }


    // callback that returns total number of sections in UITableView
    override func numberOfSectionsInTableView(
        tableView: UITableView) -> Int {
            return 1
    }

    // callback that returns number of rows in the UITableView
    override func tableView(tableView: UITableView,
        numberOfRowsInSection section: Int) -> Int {
            return model.count
    }

    // callback that returns a configured cell for the given NSIndexPath
    override func tableView(tableView: UITableView,
        cellForRowAtIndexPath indexPath: NSIndexPath) ->
        UITableViewCell {

            // get cell
            let cell = tableView.dequeueReusableCellWithIdentifier(
                "Cell", forIndexPath: indexPath)

            // set cell label's text to the tag at the specified index
            cell.textLabel?.text = model.tagAtIndex(indexPath.row)
```

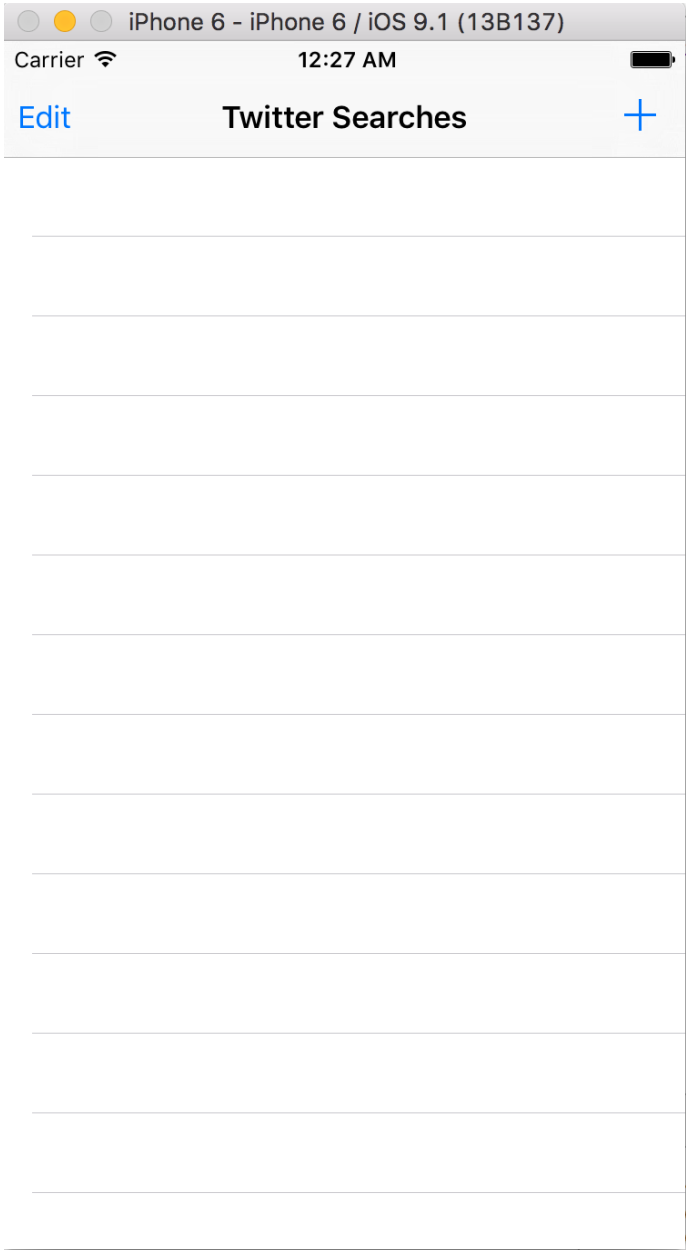
```
        // set up long press guesture recognizer
        let longPressGestureRecognizer = UILongPressGestureRecognizer(
            target: self, action: "tableViewCellLongPressed:")
        longPressGestureRecognizer.minimumPressDuration = 0.5
        cell.addGestureRecognizer(longPressGestureRecognizer)

        return cell
    }

    // callback that returns whether a cell is editable
    override func tableView(tableView: UITableView,
        canEditRowAtIndexPath indexPath: NSIndexPath) -> Bool {
            return true // all cells are editable
    }

    // callback that deletes a row from the UITableView
    override func tableView(tableView: UITableView,
        commitEditingStyle editingStyle: UITableViewCellEditingStyle,
        forRowAtIndexPath indexPath: NSIndexPath) {
            if editingStyle == .Delete {
                model.deleteSearchAtIndex(indexPath.row)

                // remove UITableView row
                tableView.deleteRowsAtIndexPaths(
                    [indexPath], withRowAnimation: .Fade)
            }
    }

    // callback that returns whether cells can be moved
    override func tableView(tableView: UITableView,
        canMoveRowAtIndexPath indexPath: NSIndexPath) -> Bool {
            return true
    }

    // callback that reorders keys when user moves them in the table
    override func tableView(tableView: UITableView,
        moveRowAtIndexPath sourceIndexPath: NSIndexPath,
        toIndexPath destinationIndexPath: NSIndexPath) {
            // tell model to reorder tags based on UITableView order
            model.moveTagAtIndex(sourceIndexPath.row,
                toDestinationIndex: destinationIndexPath.row)
    }

}
```

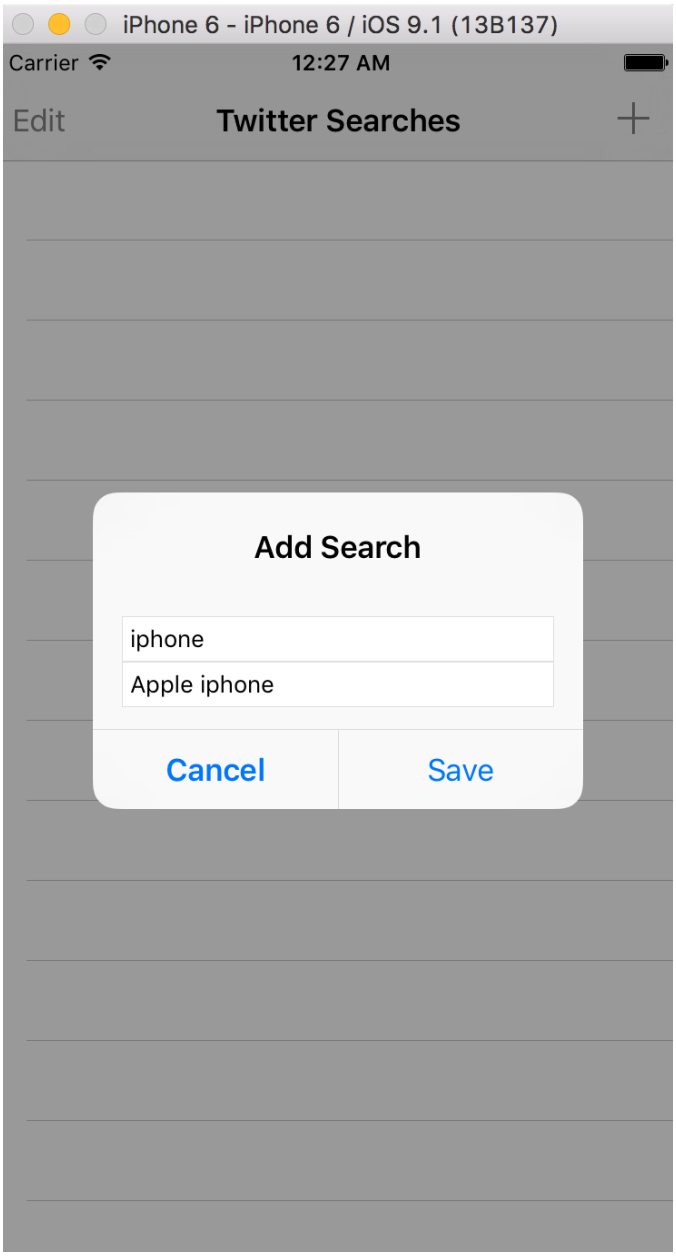
```swift
//
//  DetailViewController.swift
//  TwitterSearches
//
//  Created by David Hincapie on 11/10/15.
//  Copyright © 2015 David Hincapie. All rights reserved.
//

import UIKit

class DetailViewController: UIViewController, UIWebViewDelegate {
    @IBOutlet weak var webView: UIWebView! // displays search results
    var detailItem: NSURL? // URL that will be displayed

    // configure DetailViewController as the webView's delegate
    override func viewDidLoad() {
        super.viewDidLoad()
        webView.delegate = self
    }

    // after view appears, load search results into webview
    override func viewDidAppear(animated: Bool) {
        super.viewDidAppear(animated)

        if let url = self.detailItem {
            webView.loadRequest(NSURLRequest(URL: url))
        }
    }

    // stop page load and hide network activity indicator when
    // returning to MasterViewController
    override func viewWillDisappear(animated: Bool) {
        super.viewWillDisappear(animated)
        UIApplication.sharedApplication()
            . networkActivityIndicatorVisible = false
        webView.stopLoading()
    }

    // when loading starts, show network activity indicator
    func webViewDidStartLoad(webView: UIWebView) {
        UIApplication.sharedApplication()
            . networkActivityIndicatorVisible = true
    }

    // hide network activity indicator when page finishes loading
    func webViewDidFinishLoad(webView: UIWebView) {
        UIApplication.sharedApplication()
            . networkActivityIndicatorVisible = false
    }

    // display static web page if error occurs
    func webView(webView: UIWebView,
        didFailLoadWithError error: NSError?) {
            webView.loadHTMLString(
                "<html><body><p>An error occurred when performing " +
                    "the Twitter search: " + error!.description +
                "</body></html>", baseURL: nil)
    }
}
```

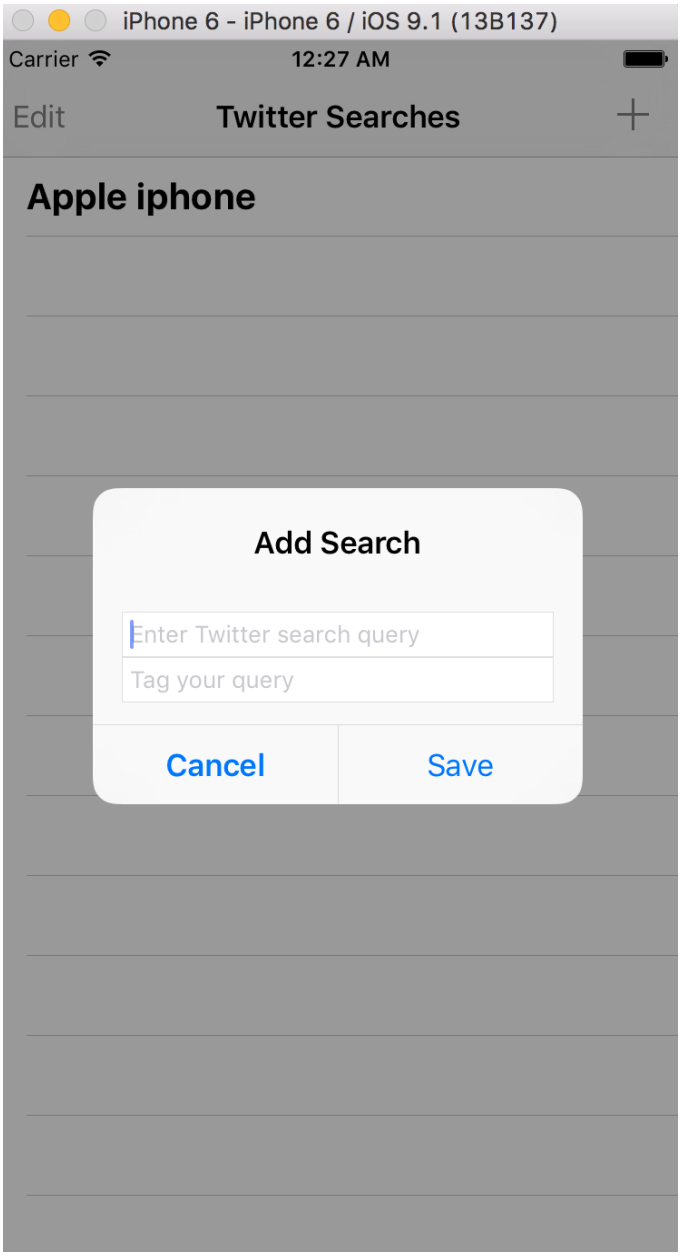
```swift
//
//  Model.swift
//  TwitterSearches
//
//  Created by David Hincapie on 11/13/15.
//  Copyright © 2015 David Hincapie. All rights reserved.
//

import Foundation

// delegate protocol that enables Model to
// notify controller when the data changes
protocol ModelDelegate {
    func modelDataChanged()
}

// manages the saved searches
class Model {
    // keys used for storing app's data in app's NSUserDefaults
    private let pairsKey = "TwitterSearchesKVPairs" // for tag-query pairs
    private let tagsKey = "TwitterSearchesKeyOrder" // for tags

    private var searches: [String: String] = [:] // stores tag-query pairs
    private var tags: [String] = [] // stores tags in user-specified order

    private let delegate: ModelDelegate // delegate is MainViewController

    // initializes the Model
    init(delegate: ModelDelegate) {
        self.delegate = delegate

        // get the NSUserDefaults object for the app
        let userDefaults = NSUserDefaults.standardUserDefaults()

        // get Dictionary of the app's tag-query pairs
        if let pairs = userDefaults.dictionaryForKey(pairsKey) {
            self.searches = pairs as! [String : String]
        }

        // get Array with the app's tag order
        if let tags = userDefaults.arrayForKey(tagsKey) {
            self.tags = tags as! [String]
        }

        // register to iCloud change notifications
        NSNotificationCenter.defaultCenter().addObserver(self,
            selector: "updateSearches:",
            name: NSUbiquitousKeyValueStoreDidChangeExternallyNotification,
            object: NSUbiquitousKeyValueStore.defaultStore())
    }

    // called by view controller to synchronize model after it's created
    func synchronize() {
        NSUbiquitousKeyValueStore.defaultStore().synchronize()
    }

    // returns the tag at the specified index
    func tagAtIndex(index: Int) -> String {
        return tags[index]
    }
```

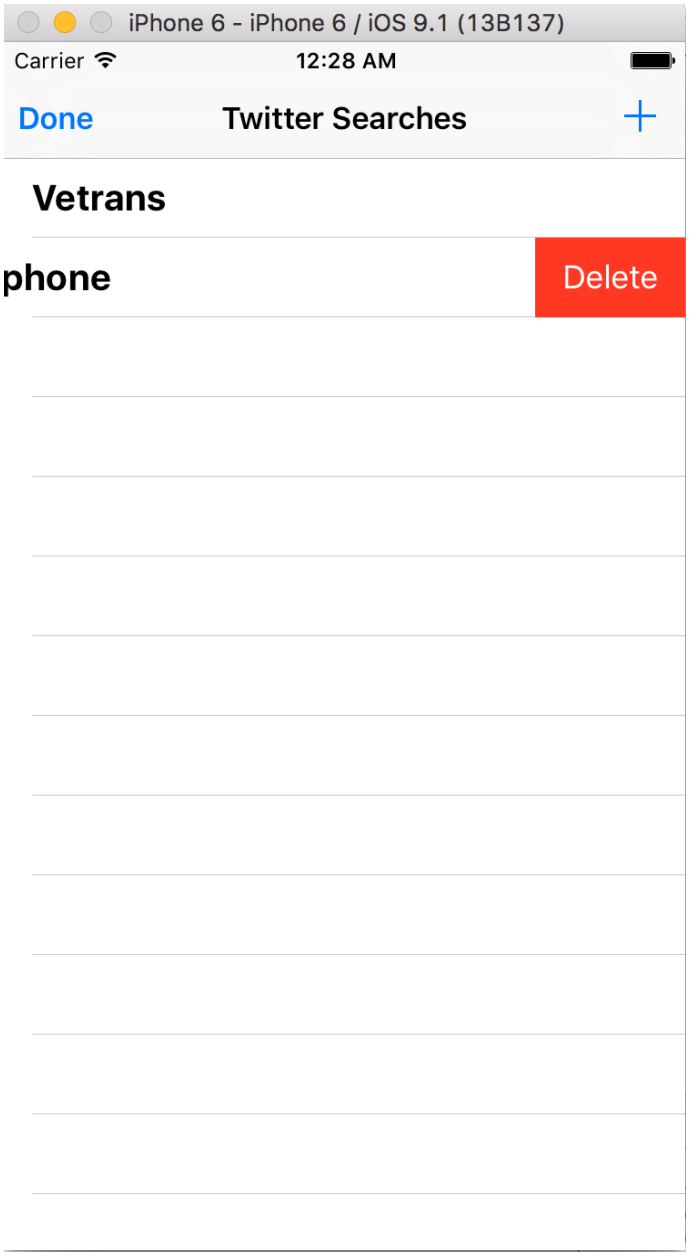
```
// returns the query String for a given tag
func queryForTag(tag: String) -> String? {
    return searches[tag]
}

// returns the query String for the tag at a given index
func queryForTagAtIndex(index: Int) -> String? {
    return searches[tags[index]]
}

// returns the number of tags
var count: Int {
    return tags.count
}

// deletes the tag from tags Array, and the corresponding
// tag-query pair from searches iCloud
func deleteSearchAtIndex(index: Int) {
    searches.removeValueForKey(tags[index])
    let removedTag = tags.removeAtIndex(index)
    updateUserDefaults(updateTags: true, updateSearches: true)

    // remove search from iCloud
    let keyValueStore = NSUbiquitousKeyValueStore.defaultStore()
    keyValueStore.removeObjectForKey(removedTag)
}

// reorders tags Array when user moves tag in controller's UITableView
func moveTagAtIndex(oldIndex: Int, toDestinationIndex newIndex: Int) {
    let temp = tags.removeAtIndex(oldIndex)
    tags.insert(temp, atIndex: newIndex)
    updateUserDefaults(updateTags: true, updateSearches: false)
}

// update user defaults with current searches and tags collections
func updateUserDefaults( updateTags  updateTags: Bool, updateSearches: Bool) {
    let userDefaults = NSUserDefaults.standardUserDefaults()

    if updateTags {
        userDefaults.setObject(tags, forKey: tagsKey)
    }

    if updateSearches {
        userDefaults.setObject(searches, forKey: pairsKey)
    }

    userDefaults.synchronize() // force immediate save to device
}

// update or delete searches when iCloud changes occur
@objc func updateSearches(notification: NSNotification) {
    if let userInfo = notification.userInfo {
        // check reason for change and update accordingly
        if let reason = userInfo[
            NSUbiquitousKeyValueStoreChangeReasonKey] as! NSNumber? {

                // if changes occurred on another device
                if reason.integerValue ==
                    NSUbiquitousKeyValueStoreServerChange ||
```

```
                    reason.integerValue ==
                     NSUbiquitousKeyValueStoreInitialSyncChange {

                         performUpdates(userInfo) // update searches
                 }
             }
         }
     }

     // add, update or delete searches based on iCloud changes
     func performUpdates(userInfo: [NSObject : AnyObject?]) {
         // get changed keys NSArray; convert to [String]
         let changedKeysObject =
         userInfo[NSUbiquitousKeyValueStoreChangedKeysKey]
         let changedKeys = changedKeysObject as! [String]

         // get NSUbiquitousKeyValueStore for updating
         let keyValueStore = NSUbiquitousKeyValueStore.defaultStore()

         // update searches based on iCloud changes
         for key in changedKeys {
             if let query = keyValueStore.stringForKey(key) {
                 saveQuery(query, forTag: key, syncToCloud: false)
             } else {
                 searches.removeValueForKey(key)
                 tags = tags.filter{$0 != key}
                 updateUserDefaults(updateTags: true, updateSearches: true)
             }

             delegate.modelDataChanged() // update the view
         }
     }

     // save a tag-query pair
     func saveQuery(query: String, forTag tag: String,
         syncToCloud sync: Bool) {

             // Dictionary method updateValue returns nil if key is new
             let oldValue = searches.updateValue(query, forKey: tag)

             if oldValue == nil {
                 tags.insert(tag, atIndex: 0) // store search tag
                 updateUserDefaults(updateTags: true, updateSearches: true)
             } else {
                 updateUserDefaults(updateTags: false, updateSearches: true)
             }

             // if sync is true, add tag-query pair to iCloud
             if sync {
                 NSUbiquitousKeyValueStore.defaultStore().setObject(
                     query, forKey: tag)
             }
     }
}
```


iPhone 6 - iPhone 6 / iOS 9.1 (13B137)
Carrier
12:27 AM
Edit
Twitter Searches


iPhone 6 - iPhone 6 / iOS 9.1 (13B137)
Carrier
12:27 AM
Edit
Twitter Searches
Add Search
iphone
Apple iphone
Cancel
Save


iPhone 6 - iPhone 6 / iOS 9.1 (13B137)
Carrier
12:27 AM
Edit
Twitter Searches
Apple iphone
Add Search
Enter Twitter search query
Tag your query
Cancel
Save


iPhone 6 - iPhone 6 / iOS 9.1 (13B137)
Carrier
12:28 AM
Done
Twitter Searches
Vetrans
phone
Delete

iPhone 6 - iPhone 6 / iOS 9.1 (13B137)

Carrier 12:28 AM

Twitter Searches **Detail**

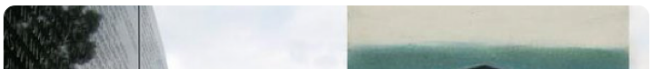
Sign up | Log in

Vetrans Day

Chicago Blackhawks @NHLBlack… Nov 12
The camo warmup jerseys are looking very good.

Very. Good. #VeteransDay

404 963 •••



iPhone 6 - iPhone 6 / iOS 9.1 (13B137)

Carrier 12:28 AM

Twitter Searches **Detail**

Sign up | Log in

iphone

Bring Me The Horizon @bmthofficial Nov 12
BACK IN STOCK:
Umbrella and That's The Spirit iPhone 6/S & 6+ cases -
horizonsupply.co/search?q=iphone
#HorizonSupplyCo

613 1,828 •••