You Deserve Nice Things

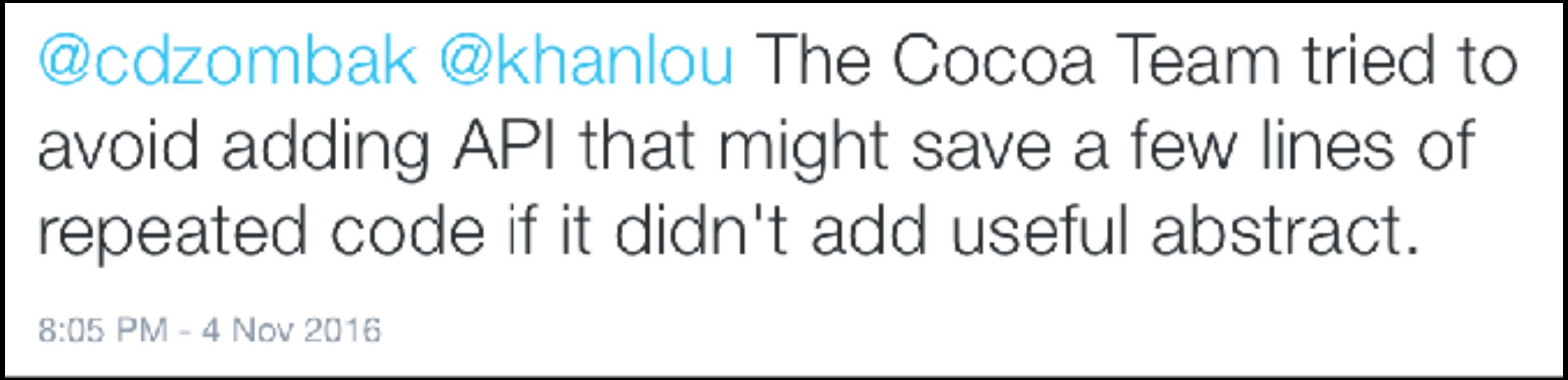
Soroush Khanlou

Pragma 2017

```
@interface NSString (NSStringExtensionMethods)

- (BOOL)containsString:(NSString *)str NS_AVAILABLE(10_10, 8_0);

@end
```

```
@interface NSString (NSStringExtensionMethods)

- (NSRange)rangeOfString:(NSString *)searchString;

@end
```

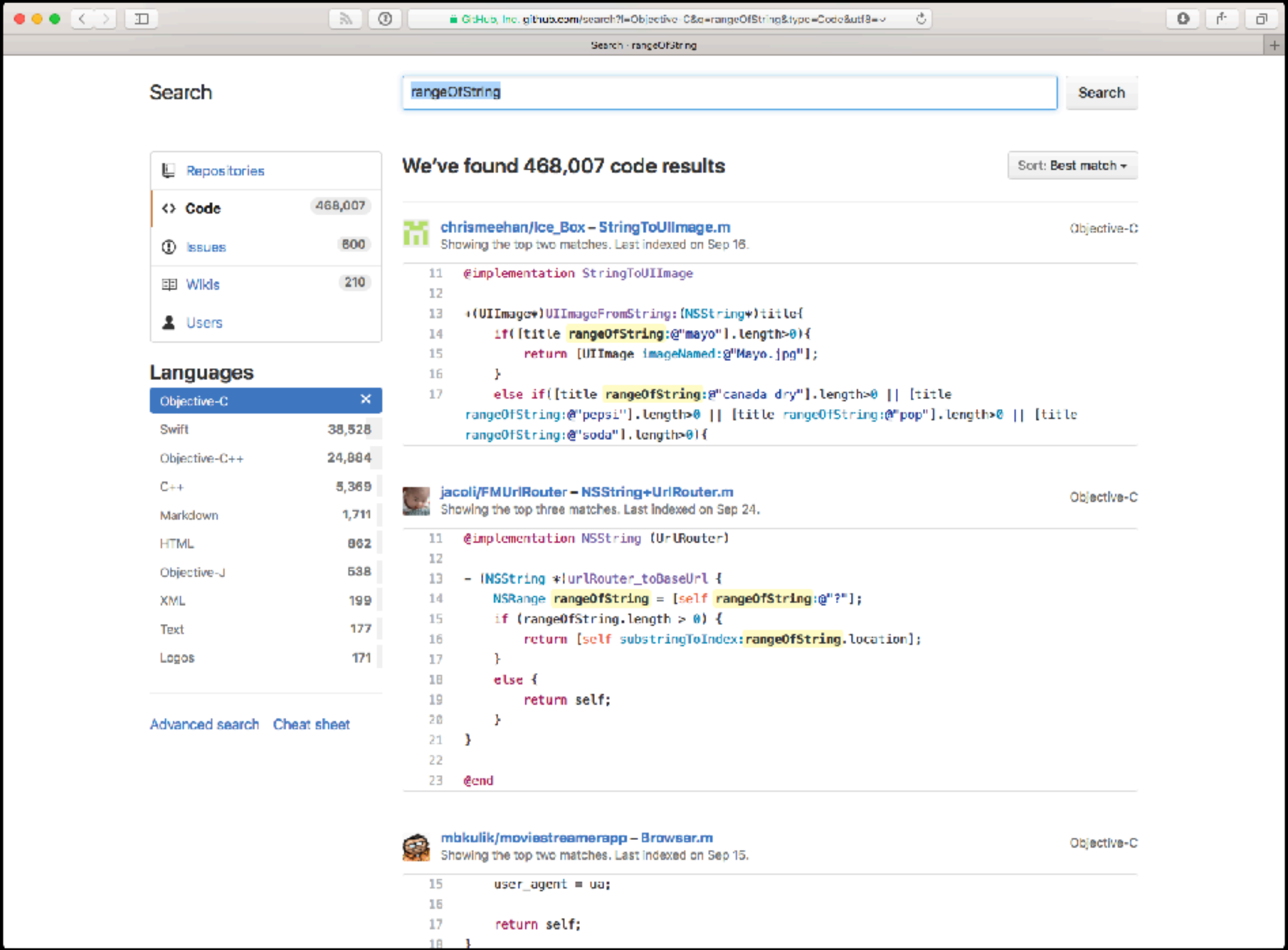
```
if ([haystack rangeOfString:needle].location != NSNotFound) {
    // The string was found
}
```


@cdzombak @khanlou The Cocoa Team tried to avoid adding API that might save a few lines of repeated code if it didn't add useful abstract.

8:05 PM - 4 Nov 2016

– Anonymous

```
@implementation NSString (Contains)

- (BOOL)contains:(NSString *)string {
    return [self rangeOfString:string].location != NSNotFound
}

@end
```


GitHub, Inc. github.com/search?l=Objective-C&q=rangeOfString&type=Code&utf8=✓
Search · rangeOfString

Search

rangeOfString

Repositories	
Code	468,007
Issues	800
Wikis	210
Users	

Languages

Objective-C ✕	
Swift	38,528
Objective-C++	24,884
C++	5,369
Markdown	1,711
HTML	862
Objective-J	538
XML	199
Text	177
Logos	171

Advanced search Cheat sheet

We've found 468,007 code results

chrismeehan/Ice_Box – StringToUIImage.m Objective-C
Showing the top two matches. Last indexed on Sep 18.

```
@implementation StringToUIImage

+(UIImage*)UIImageFromString:(NSString*)title{
    if([title rangeOfString:@"mayo"].length>0){
        return [UIImage imageNamed:@"Mayo.jpg"];
    }
    else if([title rangeOfString:@"canada dry"].length>0 || [title
rangeOfString:@"pepsi"].length>0 || [title rangeOfString:@"pop"].length>0 || [title
rangeOfString:@"soda"].length>0){
```

jacoli/FMUrlRouter – NSString+UrlRouter.m Objective-C
Showing the top three matches. Last indexed on Sep 24.

```
@implementation NSString (UrlRouter)

- (NSString *)urlRouter_toBaseUrl {
    NSRange rangeOfString = [self rangeOfString:@"?"];
    if (rangeOfString.length > 0) {
        return [self substringToIndex:rangeOfString.location];
    }
    else {
        return self;
    }
}

@end
```

mbkulik/moviestreamerapp – Browser.m Objective-C
Showing the top two matches. Last indexed on Sep 15.

```
    user_agent = ua;

    return self;
}
```

```
@interface NSString (NSStringExtensionMethods)

- (BOOL)containsString:(NSString *)str NS_AVAILABLE(10_10, 8_0);

@end
```

```swift
@available(iOS 8.0, *)
open func contains(_ str: String) -> Bool
```

```
extension Sequence {
    func first(_ predicate: (Generator.Element) -> Bool) ->
Generator.Element? {
        for element in self {
            if try predicate(element) {
                return element
            }
        }
        return nil
    }
}


[1, 2, 3, -1, -2, -3].first({ int in int < 0 }) // => -1
```

```
extension Sequence {
    public func first(where predicate: (Element) throws -> Bool)
rethrows -> Element?
}
```

RULES

- Does it increase expressivity?

```
let a = [1, 2, 3, 4]
let pairs = ???                              // [(1, 2), (2, 3), (3, 4)]
```

```swift
let a = [1, 2, 3, 4]
let pairs = zip(a, a.dropFirst()) // [(1, 2), (2, 3), (3, 4)]
```

```swift
extension Sequence {

    func eachPair() -> AnySequence<(Element, Element)> {
        return AnySequence(zip(self, self.dropFirst()))
    }

}
```

```swift
extension Sequence {

    func eachPair() -> AnySequence<(Element, Element)> {
        return AnySequence(zip(self, self.dropFirst()))
    }

}

[1, 2, 3, 4].eachPair() // [(1, 2), (2, 3), (3, 4)]
```

```
let a = [1, 2, 3, 4]

a.reduce(0, +)
```

```swift
let a = [1, 2, 3, 4]

a.reduce(0, +) // => 10
```

```swift
extension Sequence where Element: Numeric {
    var sum: Element {
        return self.reduce(0, +)
    }
}
```

```swift
extension Sequence where Element: Numeric {
    var sum: Element {
        return self.reduce(0, +)
    }
}

extension Array where Element: BinaryFloatingPoint {
    var average: Element {
        return self.reduce(0, +) / Element(self.count)
    }
}
```

RULES

- Does it increase expressivity?

RULES

- Does it increase expressivity?
- Does it decrease noise?

```swift
let rowCount: Int = 5
let height: CGFloat = 20
let rowCount: Int = 5
let height: CGFloat = 20

let result = height * rowCount
let rowCount: Int = 5
let height: CGFloat = 20

let result = height * rowCount
```

Binary operator '*' cannot be applied to operands of type 'CGFloat' and 'Int'

```swift
let rowCount: Int = 5
let height: CGFloat = 20

let result = height * CGFloat(rowCount)
```

```swift
func *(lhs: Int, rhs: CGFloat) -> CGFloat {
    return CGFloat(lhs) * rhs
}

func *(lhs: CGFloat, rhs: Int) -> CGFloat {
    return lhs * CGFloat(rhs)
}
```

```swift
let rowCount: Int = 5
let height: CGFloat = 20

let result = height * rowCount
```

```
// before
![1, 2, 3, -1, -2, -3].contains(where: { !($0 > 0) })
```

```
// before
![1, 2, 3, -1, -2, -3].contains(where: { !($0 > 0) }) // => false
```

```
// before
![1, 2, 3, -1, -2, -3].contains(where: { !($0 > 0) }) // => false



extension Sequence {

    func all(_ predicate: (Element) -> Bool) -> Bool {
        for element in self {
            if !predicate(element) {
                return false
            }
        }
        return true
    }

}



// after
[1, 2, 3, -1, -2, -3].all({ $0 > 0 }) // => false
```

```swift
// before
![1, 2, 3, -1, -2, -3].contains(where: { $0 > 0 })
```

```swift
// before
![1, 2, 3, -1, -2, -3].contains(where: { $0 > 0 }) // => false
```

```swift
// before
![1, 2, 3, -1, -2, -3].contains(where: { $0 > 0 }) // => false



extension Sequence {

    func none(_ predicate: (Element) -> Bool) -> Bool {
        for element in self {
            if predicate(element) {
                return false
            }
        }
        return true
    }

}



// after
[1, 2, 3, -1, -2, -3].none({ $0 > 0 }) // => false
```

```
// before
[1, 2, 3, -1, -2, -3].contains(where: { $0 > 0 }) // => true
```

```swift
// before
[1, 2, 3, -1, -2, -3].contains(where: { $0 > 0 }) // => true



extension Sequence {
    // also in the standard library as `contains`
    func any(_ predicate: (Element) -> Bool) -> Bool {
        for element in self {
            if predicate(element) {
                return true
            }
        }
        return false
    }

}



// after
[1, 2, 3, -1, -2, -3].any({ $0 > 0 }) // => true
```

RULES

- Does it increase expressivity?
- Does it decrease noise?

RULES

- Does it increase expressivity?
- Does it decrease noise?
- Is there an optimization in there?

```
// before
[1, 2, 3, -1, -2, -3].filter({ $0 > 0 }).count // => 3
```

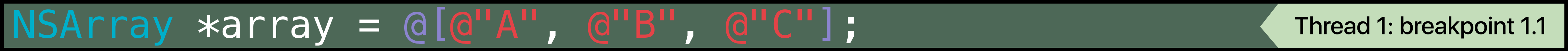
```swift
// before
[1, 2, 3, -1, -2, -3].filter({ $0 > 0 }).count // => 3


extension Sequence {

    func count(where predicate: (Element) -> Bool) -> Int {
        var count = 0
        for element in self {
            if try predicate(element) {
                count += 1
            }
        }
        return count
    }

}
```

```swift
// before
[1, 2, 3, -1, -2, -3].filter({ $0 > 0 }).count // => 3


extension Sequence {

    func count(where predicate: (Element) -> Bool) -> Int {
        var count = 0
        for element in self {
            if try predicate(element) {
                count += 1
            }
        }
        return count
    }

}



// after
[1, 2, 3, -1, -2, -3].count(where: { $0 > 0 }) // => 3
```

```objc
NSArray *array = @[@"A", @"B", @"C"];

NSArray *reversed = [array reversedArray];
```

```
No visible @interface for 'NSArray' declares the selector 'reversedArray'
```

```
NSArray *array = @[@"A", @"B", @"C"];

NSArray *reversed = array.reverseObjectEnumerator.allObjects;
```



```
NSArray *array = @[@"A", @"B", @"C"];
```

Thread 1: breakpoint 1.1

```
NSArray *array = @[@"A", @"B", @"C"];
```

Thread 1: breakpoint 1.1

```
(lldb) po [array reversedArray];
```

```
NSArray *array = @[@"A", @"B", @"C"];                Thread 1: breakpoint 1.1
```

```
(lldb) po [array reversedArray];
<__NSArrayReversed 0x60800003f0a0>(
C,
B,
A
)
```

```swift
public func reversed() -> ReversedRandomAccessCollection<Array<Element>>
```

RULES

- Does it increase expressivity?
- Does it decrease noise?
- Is there an optimization in there?

RULES

- Does it increase expressivity?
- Does it decrease noise?
- Is there an optimization in there?
- Does it belong on every instance of this type?

```swift
let requestProperties: [String: Any] = // data to prepare some request

let request: Request? = requestProperties.buildRequest()
let requestProperties: [String: Any] = // data to prepare some request

let request = RequestBuilder(properties: requestProperties).buildRequest()
```

```
@implementation NSDictionary (PushNotifications)

- (NSDictionary *)apsDict {
    return self[@"aps"];
}

@end
```

RULES

- Does it increase expressivity?
- Does it decrease noise?
- Is there an optimization in there?
- Does it belong on every instance of this type?

```
extension Sequence {

    func uniqueElements() -> [Element] {
        return Array(Set(self))
    }

}
```

```
extension Sequence {

    func uniqueElements() -> [Element] {
        return Array(Set(self))
    }

}
```

Generic parameter 'Source' could not be inferred

```swift
extension Sequence {

    func uniqueElements() -> [Element] {
        return Array(Set(self))
    }

}
```

❗ Element does not conform to ‘Hashable’

```swift
extension Sequence where Element: Hashable {

    func uniqueElements() -> [Element] {
        return Array(Set(self))
    }

}
```

```swift
extension Sequence {

    func uniqueElements(by elementsEqual: (Element, Element) -> Bool)
-> [Element] {









    }
}

extension Sequence where Element: Equatable {
    func uniqueElements() -> [Element] {
        return uniqueElements(by: ==)
    }
}
```

```swift
extension Sequence {

    func uniqueElements(by elementsEqual: (Element, Element) -> Bool)
-> [Element] {
        var result: [Element] = []






        return result
    }
}

extension Sequence where Element: Equatable {
    func uniqueElements() -> [Element] {
        return uniqueElements(by: ==)
    }
}
```

```swift
extension Sequence {

    func uniqueElements(by elementsEqual: (Element, Element) -> Bool)
-> [Element] {
        var result: [Element] = []
        for element in self {




        }
        return result
    }
}

extension Sequence where Element: Equatable {
    func uniqueElements() -> [Element] {
        return uniqueElements(by: ==)
    }
}
```

```swift
extension Sequence {

    func uniqueElements(by elementsEqual: (Element, Element) -> Bool)
-> [Element] {
        var result: [Element] = []
        for element in self {
            if !result.contains(where: { resultElement in
elementsEqual(element, resultElement) }) {

            }
        }
        return result
    }
}

extension Sequence where Element: Equatable {
    func uniqueElements() -> [Element] {
        return uniqueElements(by: ==)
    }
}
```

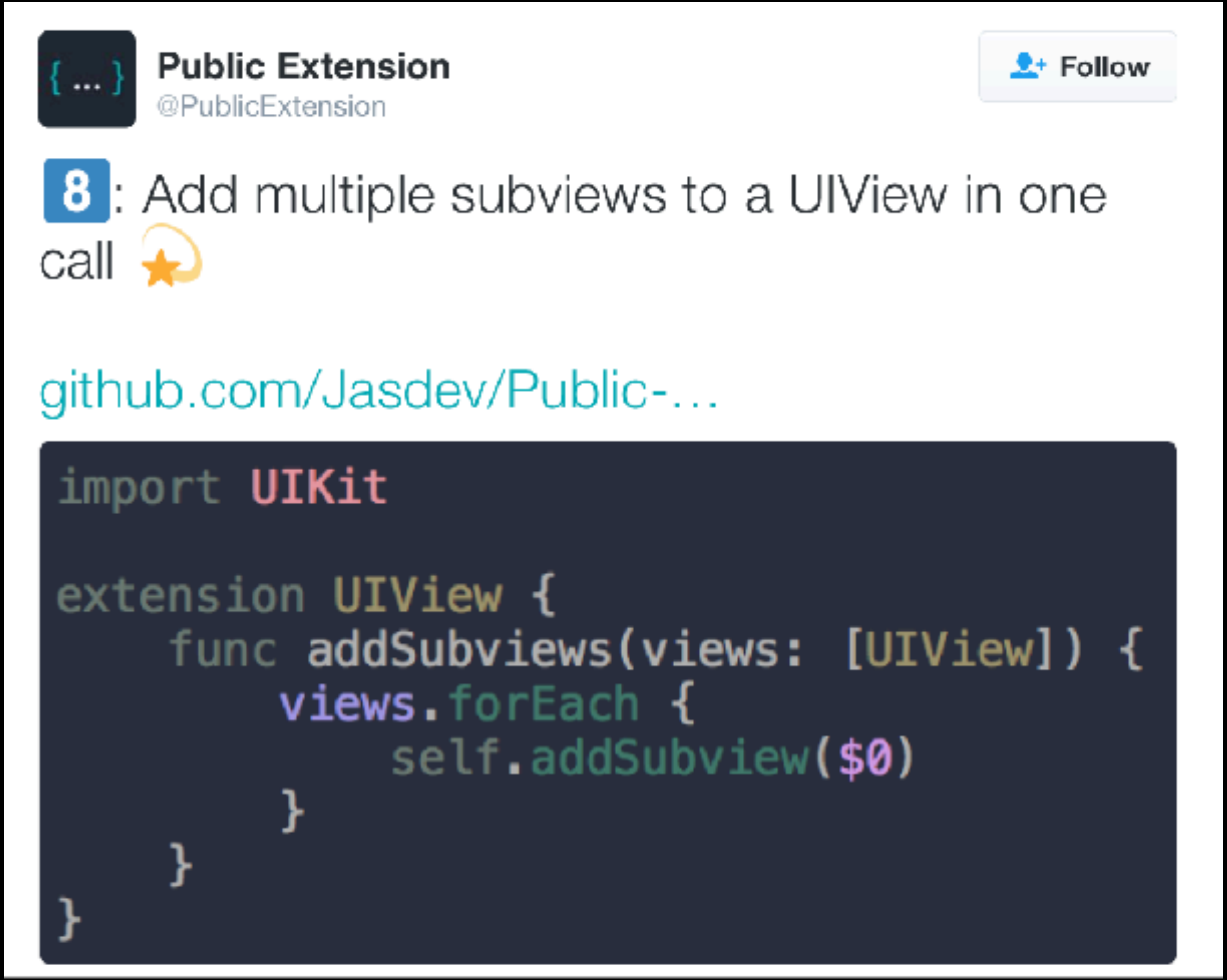
```swift
extension Sequence {

    func uniqueElements(by elementsEqual: (Element, Element) -> Bool)
-> [Element] {
        var result: [Element] = []
        for element in self {
            if !result.contains(where: { resultElement in
elementsEqual(element, resultElement) }) {
                result.append(element)
            }
        }
        return result
    }
}

extension Sequence where Element: Equatable {
    func uniqueElements() -> [Element] {
        return uniqueElements(by: ==)
    }
}
```

```swift
// before
houses.sorted(by: { $0.numberOfResidents < $1.numberOfResidents })
```

```
// before
houses.sorted(by: { $0.numberOfResidents < $1.numberOfResidents })
```

```
extension Sequence {
    func sorted<T: Comparable>(on propertyAccessor: (Element) -> T) ->
[Element] {
        return sorted(by: { propertyAccessor($0) <
propertyAccessor($1) })
    }
}
```

```
// after
houses.sorted(on: { $0.numberOfResidents })
```



8: Add multiple subviews to a UIView in one call 💫

github.com/Jasdev/Public-…

```
import UIKit

extension UIView {
    func addSubviews(views: [UIView]) {
        views.forEach {
            self.addSubview($0)
        }
    }
}
```

@jasdev / @publicextension

```swift
extension Data {
    var hexString: String {
        return self
            .map({ return String(format: "%02hhx", $0) })
            .joined()
    }
}
```

What can you steal from Ruby’s Enumerable?

```
let numbers = ["1","2","3","4","5","6","7"]

let chunked = numbers.chunk(size: 3)
  // => [["1", "2", "3"], ["4", "5", "6"], ["7"]]
```

```swift
extension Array {
    func chunk(size: Int) -> [[Element]] {











    }
}
```

```
extension Array {
    func chunk(size: Int) -> [[Element]] {
        let steps = stride(
            from: self.startIndex,
            to: self.endIndex,
            by: size)






    }
}
```

```swift
extension Array {
    func chunk(size: Int) -> [[Element]] {
        let steps = stride(
            from: self.startIndex,
            to: self.endIndex,
            by: size)
        return steps.map({ i -> Array<Element> in





        })
    }
}
```

```swift
extension Array {
    func chunk(size: Int) -> [[Element]] {
        let steps = stride(
            from: self.startIndex,
            to: self.endIndex,
            by: size)
        return steps.map({ i -> Array<Element> in
            let end = self.index(i,
                                 offsetBy: size,
                                 limitedBy: self.endIndex)
                ?? self.endIndex

        })
    }
}
```

```swift
extension Array {
    func chunk(size: Int) -> [[Element]] {
        let steps = stride(
            from: self.startIndex,
            to: self.endIndex,
            by: size)
        return steps.map({ i -> Array<Element> in
            let end = self.index(i,
                                 offsetBy: size,
                                 limitedBy: self.endIndex)
                ?? self.endIndex
            return Array(self[i ..< end])
        })
    }
}
```

UIKit

```swift
extension UIImage {
    var aspectRatio: CGFloat {
        return size.width / size.height
    }
}
```

```swift
func update(with freshData: [Data]) {
    // ...

    // user may be interacting with changing content

    // ...
}
```

```swift
func update(with freshData: [Data]) {

    // ...


    panGestureRecognizer.isEnabled = false
    panGestureRecognizer.isEnabled = true

    // ...

}
```

```swift
func update(with freshData: [Data]) {

    // ...

    // this cancels any in-progress gestures
    panGestureRecognizer.isEnabled = false
    panGestureRecognizer.isEnabled = true

    // ...

}
```

```
extension UIGestureRecognizer {
    func cancel() {
//save old value?
        isEnabled = false
        isEnabled = true
    }
}
```

```swift
func update(with freshData: [Data]) {
    // ...
    panGestureRecognizer.cancel()
    // ...
}
```

```swift
extension UIViewController {
   var isVisible: Bool {
      return self.view.window != nil
   }
}
```

```swift
extension UIViewController {
   var isVisible: Bool {
      return self.isViewLoaded && self.view.window != nil
   }
}
```

THANKS

What’s the vendor’s responsibility here?

- HMMM?

```swift
extension Int {

    func times(_ each: () -> ()) {
        (0..<self).forEach { _ in each() }
    }

}

5.times { print("hello") }
```

```swift
extension Int {

    func timesWithIndex(_ each: (Int) -> ()) {
        (0..<self).forEach({ i in each(i) })
    }

}

5.timesWithIndex { i in print("hello", i) }
```

```swift
extension Array where Element: Equatable {
    func isComposedOfIdenticalSlices(ofLength sliceLength: Int) ->
Bool {
        let chunks = array.chunked(size: sliceLength)
        guard let firstChunk = chunks.first else { return true }
        return chunks.all(predicate: { $0 == firstChunk })
    }
}
```

```
extension URLComponents {
    var pathExtension: String {
        get {
            return NSString(string: self.path).pathExtension
        }
        set {
            let nsStringPath = NSString(string: self.path)
            let nsStringPathWithoutExtension = NSString(string:
nsStringPath.deletingPathExtension)
            if let nsStringWithNewExtension =
nsStringPathWithoutExtension.appendingPathExtension(newValue) {
                self.path = nsStringWithNewExtension
            }
        }
    }
}
```

```swift
public struct NilError: Error { }

extension Optional {
    public func unwrap() throws -> Wrapped {
        guard let result = self else {
            throw NilError()
        }
        return result
    }
}
```