

Meet The Overflow, a newsletter by developers, for developers. Fascinating questions, illuminating answers, and entertaining links from around the web. [Learn more](#)

Find out if Character in String is emoji?

Asked 4 years, 3 months ago Active 23 days ago Viewed 31k times



I need to find out whether a character in a string is an emoji.

64

For example, I have this character:



```
let string = "👍"  
let character = Array(string)[0]
```



38

I need to find out if that character is an emoji.

ios

string

swift

character

emoji

edited Jun 10 '15 at 13:09



ABakerSmith

19.9k 7 60 71

asked Jun 10 '15 at 13:03



Andrew

3,461 8 30 66

I am curious: why do you need that information? – [Martin R](#) Jun 10 '15 at 13:04

@EricD.: There are *many* Unicode characters which take more than one UTF-8 code point (e.g. "€" = E2 82 AC) or more than one UTF-16 code point (e.g. "👍" = D834 DD1E). – [Martin R](#) Jun 10 '15 at 13:30 ✎

Hope you will got an idea from this obj-c version of code stackoverflow.com/questions/19886642/... – [Ashish Kakkad](#) Jun 10 '15 at 13:40

Strings have their indexing which is a preferred way of using them. To get a particular character (or grapheme cluster rather) you could: `let character = string[string.index(after: string.startIndex)]` or `let secondCharacter = string[string.index(string.startIndex, offsetBy: 1)]` – [Paul B](#) Sep 12 at 12:32

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

153

What I stumbled upon is the difference between characters, unicode scalars and glyphs.

For example, the glyph 🧑🏿🧑🏿🧑🏿 consists of 7 unicode scalars:

- Four emoji characters: 🧑🏿🧑🏿🧑🏿🧑🏿
- In between each emoji is a special character, which works like character glue; see [the specs for more info](#)

Another example, the glyph 🇳🇪 consists of 2 unicode scalars:

- The regular emoji: 🇳🇪
- A skin tone modifier: 🏿

So when rendering the characters, the resulting glyphs really matter.

What I was looking for was a way to detect if a string is exactly and only one emoji. So I could render it larger than normal text (like Messages does on iOS10 and WhatsApp does nowadays). As described above, character count is really of no use. (The 'glue character' is also not considered an emoji).

What you can do is use CoreText to help you break down the string into glyphs and count them. Furthermore, I would move part of the extension proposed by Arnold and Sebastian Lopez to a separate extension of `UnicodeScalar`.

It leaves you with the following result:

```
import Foundation

extension UnicodeScalar {
    /// Note: This method is part of Swift 5, so you can omit this.
    /// See: https://developer.apple.com/documentation/swift/unicode/scalar
    var isEmoji: Bool {
        switch value {
            case 0x1F600...0x1F64F, // Emoticons
                 0x1F300...0x1F5FF, // Misc Symbols and Pictographs
                 0x1F680...0x1F6FF, // Transport and Map
                 0x1F1E6...0x1F1FF, // Regional country flags
                 0x2600...0x26FF, // Misc symbols
                 0x2700...0x27BF, // Dingbats
                 0xE0020...0xE007F, // Tags
                 0xFE00...0xFE0F, // Variation Selectors
                 0x1F900...0x1F9FF: // Supplemental Symbols and Pictographs
            default:
                return false
        }
    }
}
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

        return true

        default: return false
    }
}

var isZeroWidthJoiner: Bool {
    return value == 8205
}

}

extension String {
    // Not needed anymore in swift 4.2 and later, using `.count` will give you the
    // correct result
    var glyphCount: Int {
        let richText = NSAttributedString(string: self)
        let line = CTLineCreateWithAttributedString(richText)
        return CTLineGetGlyphCount(line)
    }

    var isSingleEmoji: Bool {
        return glyphCount == 1 && containsEmoji
    }

    var containsEmoji: Bool {
        return unicodeScalars.contains { $0.isEmoji }
    }

    var containsOnlyEmoji: Bool {
        return !isEmpty
            && !unicodeScalars.contains(where: {
                !$0.isEmoji && !$0.isZeroWidthJoiner
            })
    }

    // The next tricks are mostly to demonstrate how tricky it can be to determine
    // emoji's
    // If anyone has suggestions how to improve this, please let me know
    var emojiString: String {
        return emojiScalars.map { String($0) }.reduce("", +)
    }

    var emojis: [String] {
        var scalars: [[UnicodeScalar]] = []
        var currentScalarSet: [UnicodeScalar] = []
        var previousScalar: UnicodeScalar?
    }

```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

    if (scalar.isZeroWidthJoiner) {
        scalars.append(currentScalarSet)
        currentScalarSet = []
    }
    currentScalarSet.append(scalar)

    previousScalar = scalar
}

scalars.append(currentScalarSet)

return scalars.map { $0.map { String($0) }.reduce("", +) }
}

fileprivate var emojiScalars: [UnicodeScalar] {
    var chars: [UnicodeScalar] = []
    var previous: UnicodeScalar?
    for cur in unicodeScalars {
        if let previous = previous, previous.isZeroWidthJoiner, cur.isEmoji {
            chars.append(previous)
            chars.append(cur)

        } else if cur.isEmoji {
            chars.append(cur)
        }

        previous = cur
    }

    return chars
}

```

Which will give you the following results:

```

"👉".isSingleEmoji // true
"👤".isSingleEmoji // true
"👤👤".isSingleEmoji // true
"👤👤".containsOnlyEmoji // true
"Hello 👤👤".containsOnlyEmoji // false
"Hello 👤👤".containsEmoji // true
"👤👤 Héllö 👤👤".emojiString // "👤👤👤👤"
"👤👤".glyphCount // 1
"👤👤".characters.count // 4, Will return '1' in Swift 4.2 so previous method not needed

```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

128103]
"👋 Héllø 🇩🇪".emojis // ["👋", "🇩🇪"]

"👋👋👋".isSingleEmoji // false
"👋👋👋".containsOnlyEmoji // true
"👋👋👋".glyphCount // 3
"👋👋👋".characters.count // 8, Will return '3' in Swift 4.2 so previous method not
needed anymore

```

edited Jun 10 at 16:58

answered Sep 10 '16 at 12:12



Kevin R

4,838 3 28 40

-
- 4 This is by far the best and most correct answer here. Thank you! One small note, your examples don't match the code (you renamed containsOnlyEmoki to containsEmoji in the snippet - I presume because it's more correct, in my testing it returned true for strings with mixed characters). – [Tim Bull](#) Sep 29 '16 at 22:00
-
- 2 My bad, I changed around some code, guess I messed up. I updated the example – [Kevin R](#) Oct 1 '16 at 10:36
-
- 2 @Andrew: Sure, I added another method to the example to demonstrate this :). – [Kevin R](#) Oct 14 '16 at 15:38
-
- 2 @Andrew this is where it gets really messy. I added an example how to do that. The problem is I have assume to know how CoreText will render the the glyphs by simply checking the characters. If anyone has suggestions for a cleaner method please let me know. – [Kevin R](#) Oct 16 '16 at 6:56 ✎
-
- 3 @Andrew Thanks for pointing that out, I changed the way containsOnlyEmoji checks. I also updated the example to Swift 3.0. – [Kevin R](#) Oct 29 '16 at 8:41
-

The simplest, cleanest, and *swiftest* way to accomplish this is to simply check the Unicode code points for each character in the string against known emoji and dingbats ranges, like so:

42

```

extension String {
    var containsEmoji: Bool {
        for scalar in unicodeScalars {
            switch scalar.value {
            case 0x1F600...0x1F64F, // Emoticons
                 0x1F300...0x1F5FF, // Misc Symbols and Pictographs
                 0x1F5E0...0x1F5FF, // Transport and Map
            default:
                continue
            }
        }
        return false
    }
}

```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```

    0x1F900...0x1F9FF, // Supplemental Symbols and Pictographs
    0x1F1E6...0x1F1FF: // Flags
    return true

    default:
        continue
    }
}
return false
}
}

```

edited Oct 27 '17 at 23:03



Atomic Avocado -
magpali
23 6

answered Mar 28 '16 at 8:29



Arnold
1,363 21 43

- 5 A code example like this is way better than suggesting to include a third party library dependency. Shardul's answer is unwise advice to follow—always write your own code. – [thefaj](#) Mar 31 '16 at 23:24

This is great, thank you for commenting what the cases pertain to – [Shawn Throop](#) Apr 29 '16 at 9:20

- 1 Like so much your code, I implemented it in an answer [here](#). A thing I noticed is that it miss some emoji, maybe because they are not part of the categories you listed, for example this one: Robot Face emoji 🤖 – [Cue](#) Jun 13 '16 at 19:24 ✎

- 1 @Tel I guess it would be the range 0x1F900...0x1F9FF (per Wikipedia). Not sure all of the range should be considered emoji. – [Frizlab](#) Aug 14 '16 at 13:53 ✎

@ Frizlab Thank you, it works. – [Cue](#) Aug 14 '16 at 20:06

8

```

extension String {
    func containsEmoji() -> Bool {
        for scalar in unicodeScalars {
            switch scalar.value {
            case 0x3030, 0x00AE, 0x00A9, // Special Characters
                 0x1D000...0x1F77F,    // Emoticons
                 0x2100...0x27BF,      // Misc symbols and Dingbats
                 0xFE00...0xFE0F,      // Variation Selectors
                 0x1F900...0x1F9FF:    // Supplemental Symbols and Pictographs
            }
        }
    }
}

```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```

    }
}
return false
}
}

```

This is my fix, with updated ranges.

answered Aug 18 '16 at 20:25



Sebastian Lopez

81 1 1

Swift 5.0

5 ... introduced a new way of checking exactly this!

You have to break your `String` into its [Scalars](#). Each `Scalar` has a [Property](#) value which supports the [isEmoji](#) value!

Actually you can even check if the Scalar is a Emoji modifier or more. Check out Apple's documentation:

<https://developer.apple.com/documentation/swift/unicode/scalar/properties>

You may want to consider checking for `isEmojiPresentation` instead of `isEmoji`, because Apple states the following for `isEmoji`:

This property is true for scalars that are rendered as emoji by default and also for scalars that have a non-default emoji rendering when followed by U+FE0F VARIATION SELECTOR-16. This includes some scalars that are not typically considered to be emoji.

This way actually splits up Emoji's into all the modifiers, but it is way simpler to handle. And as Swift now counts Emoji's with modifiers (e.g.: 🧑🏽, 🏴) as 1 you can do all kind of stuff.

```

var string = "🧑🏽 test"

for scalar in string.unicodeScalars {
    let isEmoji = scalar.properties.isEmoji
}

```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```
// 🍌
// false
// t false
// e false
// s false
// t false
```

[NSHipster](#) points out an interesting way to get all Emoji's:

```
import Foundation


var emoji = CharacterSet()

for codePoint in 0x0000...0x1F0000 {
    guard let scalarValue = Unicode.Scalar(codePoint) else {
        continue
    }

    // Implemented in Swift 5 (SE-0221)
    // https://github.com/apple/swift-evolution/blob/master/proposals/0221-character-
    // properties.md
    if scalarValue.properties.isEmoji {
        emoji.insert(scalarValue)
    }
}
```

answered Jun 5 at 23:42

 [alexkaessner](#)
672 6 19

Great answer, thanks. It's worth mentioning that your min sdk must be 10.2 to use this part of Swift 5. Also in order to check if a string was only made up of emojis I had to check if it had one of these properties: `scalar.properties.isEmoji` `scalar.properties.isEmojiPresentation` `scalar.properties.isEmojiModifier` `scalar.properties.isEmojiModifierBase` `scalar.properties.isJoinControl` `scalar.properties.isVariationSelector` – [Springham](#) Jul 16 at 9:26 

2 Beware, integers 0-9 are considered emojis. So `"6".unicodeScalars.first!.properties.isEmoji` will evaluate as `true` – [Miniroom](#) Aug 6 at 20:01 

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

should still work though.

```
let str: NSString = "hello🐼"

@objc protocol NSStringPrivate {
    func _containsEmoji() -> ObjCBool
}

let strPrivate = unsafeBitCast(str, to: NSStringPrivate.self)
strPrivate._containsEmoji() // true
str.value(forKey: "_containsEmoji") // 1

let swiftStr = "hello🐼"
(swiftStr as AnyObject).value(forKey: "_containsEmoji") // 1
```

Swift 2.x:

I recently discovered a private API on `NSString` which exposes functionality for detecting if a string contains an Emoji character:

```
let str: NSString = "hello🐼"
```

With an objc protocol and `unsafeBitCast` :

```
@objc protocol NSStringPrivate {
    func cnui_containsEmojiCharacters() -> ObjCBool
    func _containsEmoji() -> ObjCBool
}

let strPrivate = unsafeBitCast(str, NSStringPrivate.self)
strPrivate.cnui_containsEmojiCharacters() // true
strPrivate._containsEmoji() // true
```

With `valueForKey` :

```
str.valueForKey("cnui_containsEmojiCharacters") // 1
str.valueForKey("_containsEmoji") // 1
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```
let str = "hello🐼"
```

```
(str as AnyObject).valueForKey("cnu_containsEmojiCharacters") // 1
(str as AnyObject).valueForKey("_containsEmoji") // 1
```

Methods found in the [NSString header file](#).

edited Dec 2 '16 at 0:27

answered Jul 18 '16 at 20:07



JAL

34.5k

18

132

245

This is what i am looking for, Thanks JAL – user5180348 Sep 6 '16 at 9:06 ✎

Will this be rejected by Apple? – Andrey Chernukha Dec 1 '16 at 20:39

@AndreyChernukha There's always a risk, but I haven't experienced any rejection yet. – JAL Dec 1 '16 at 20:51

Never ever use private APIs. At best, the hurt will only come tomorrow. Or next month. – xaphod Jun 2 '17 at 2:05

You can use this code [example](#) or this [pod](#).

2

To use it in Swift, import the category into the `YourProject_Bridging_Header`

```
#import "NSString+EMOEmoji.h"
```

Then you can check the range for every emoji in your String:

```
let example: NSString = "string🐼with🐼emojis🐼" //string with emojis
let containsEmoji: Bool = example.emo_containsEmoji()

print(containsEmoji)

// Output: ["true"]
```

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



For Swift 3.0.2, the following answer is the simplest one:

2

```
class func stringContainsEmoji (string : NSString) -> Bool
{
    var returnValue: Bool = false

    string.enumerateSubstrings(in: NSRange(0, (string as NSString).length), options:
    NSString.EnumerationOptions.byComposedCharacterSequences) { (substring, substringRange,
    enclosingRange, stop) -> () in

        let objcString:NSString = NSString(string:substring!)
        let hs: unichar = objcString.character(at: 0)
        if 0xd800 <= hs && hs <= 0xdbff
        {
            if objcString.length > 1
            {
                let ls: unichar = objcString.character(at: 1)
                let step1: Int = Int((hs - 0xd800) * 0x400)
                let step2: Int = Int(ls - 0xdc00)
                let uc: Int = Int(step1 + step2 + 0x10000)

                if 0x1d000 <= uc && uc <= 0x1f7ff
                {
                    returnValue = true
                }
            }
        }
        else if objcString.length > 1
        {
            let ls: unichar = objcString.character(at: 1)
            if ls == 0x20e3
            {
                returnValue = true
            }
        }
        else
        {
            if 0x2100 <= hs && hs <= 0x27ff
            {
                returnValue = true
            }
        }
    }
}
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

    {
        returnValue = true
    }

    else if 0x2934 <= hs && hs <= 0x2935
    {
        returnValue = true
    }

    else if 0x3297 <= hs && hs <= 0x3299
    {
        returnValue = true
    }

    else if hs == 0xa9 || hs == 0xae || hs == 0x303d || hs == 0x3030 || hs ==
0x2b55 || hs == 0x2b1c || hs == 0x2b1b || hs == 0x2b50
    {
        returnValue = true
    }
}

return returnValue;
}

```

answered Feb 6 '17 at 6:15



Ankit Goyal

2,695 1 16 22

▲ The absolutely similar answer to those that wrote before me, but with updated set of emoji scalars.

2

```

extension String {
    func isContainEmoji() -> Bool {
        let isContain = unicodeScalars.first(where: { $0.isEmoji }) != nil
        return isContain
    }
}

```

```

extension UnicodeScalar {

```

```

    var isEmoji: Bool {
        switch value {

```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

    0x1F1E6...0x1F1F7,
    0x2600...0x26FF,
    0x2700...0x27BF,
    0xFE00...0xFE0F,
    0x1F900...0x1F9FF,
    65024...65039,
    8400...8447,
    9100...9300,
    127000...127600:
    return true
default:
    return false
}
}
}

```

answered Apr 10 '18 at 8:53



Alex Shoshiashvili

101 4

2 ▲ Future Proof: Manually check the character's pixels; the other solutions will break (and have broken) as new emojis are added.

▼ Note: This is Objective-C (can be converted to Swift)

Over the years these emoji-detecting solutions keep breaking as Apple adds new emojis w/ new methods (like skin-toned emojis built by pre-cursing a character with an additional character), etc.

I finally broke down and just wrote the following method which works for all current emojis and should work for all future emojis.

The solution creates a UILabel with the character and a black background. CG then takes a snapshot of the label and I scan all pixels in the snapshot for any non solid-black pixels. The reason I add the black background is to avoid issues of false-coloring due to [Subpixel Rendering](#)

The solution runs VERY fast on my device, I can check hundreds of characters a second, but it should be noted that this is a CoreGraphics solution and should not be used heavily like you could with a regular text method. Graphics processing is data heavy so

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```

    UILabel *characterRender = [[UILabel alloc] initWithFrame:CGRectMake(0, 0, 1, 1)];
    characterRender.text = character;

    characterRender.font = [UIFont fontWithName:@"AppleColorEmoji" size:12.0f];//Note:
    Size 12 font is likely not crucial for this and the detector will probably still work at
    an even smaller font size, so if you needed to speed this checker up for serious
    performance you may test lowering this to a font size like 6.0
    characterRender.backgroundColor = [UIColor blackColor];//needed to remove subpixel
    rendering colors
    [characterRender sizeToFit];

    CGRect rect = [characterRender bounds];
    UIGraphicsBeginImageContextWithOptions(rect.size, YES, 0.0f);
    CGContextRef contextSnap = UIGraphicsGetCurrentContext();
    [characterRender.layer renderInContext:contextSnap];
    UIImage *capturedImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();

    CGImageRef imageRef = [capturedImage CGImage];
    NSUInteger width = CGImageGetWidth(imageRef);
    NSUInteger height = CGImageGetHeight(imageRef);
    CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
    unsigned char *rawData = (unsigned char*) calloc(height * width * 4, sizeof(unsigned
char));
    NSUInteger bytesPerPixel = 4;//Note: Alpha Channel not really needed, if you need to
    speed this up for serious performance you can refactor this pixel scanner to just RGB
    NSUInteger bytesPerRow = bytesPerPixel * width;
    NSUInteger bitsPerComponent = 8;
    CGContextRef context = CGBitmapContextCreate(rawData, width, height,
                                                bitsPerComponent, bytesPerRow,
colorSpace,
                                                kCGImageAlphaPremultipliedLast |
kCGBitmapByteOrder32Big);
    CGColorSpaceRelease(colorSpace);

    CGContextDrawImage(context, CGRectMake(0, 0, width, height), imageRef);
    CGContextRelease(context);

    BOOL colorPixelFound = NO;

    int x = 0;
    int y = 0;
    while (y < height && !colorPixelFound) {
        while (x < width && !colorPixelFound) {

```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

```

CGFloat green = (CGFloat)rawData[byteIndex+1],
CGFloat blue = (CGFloat)rawData[byteIndex+2];

CGFloat h, s, b, a;
UIColor *c = [UIColor colorWithRed:red green:green blue:blue alpha:1.0f];
[c getHue:&h saturation:&s brightness:&b alpha:&a]; //Note: I wrote this
method years ago, can't remember why I check HSB instead of just checking r,g,b==0; Upon
further review this step might not be needed, but I haven't tested to confirm yet.

b /= 255.0f;

if (b > 0) {
    colorPixelFound = YES;
}

x++;
}
x=0;
y++;
}

return colorPixelFound;
}

```

edited May 16 at 20:12

answered Jan 16 '18 at 20:59



Albert Renshaw

9,797 15 72 161

3 I like your thinking! ;) - Out of the box! – Ramon Aug 15 '18 at 11:56

Why are you doing this to us? #apple #unicodestandard 🙄🙄🙄🙄🙄🙄 – d4Rk Feb 26 at 15:48

I haven't looked at this in a while but I wonder if I have to convert to UIColor then to hsb; it seems I can just check that r,g,b all == 0? If someone tries let me know – Albert Renshaw Feb 26 at 18:24

i like this solution, but won't it break with a character like **ı** ? – Juan Carlos Ospina Gonzalez May 16 at 14:25

1 @JuanCarlosOspinaGonzalez Nope, in emoji that renders as a blue box with a white i. It does bring up a good point though that the UILabel should force the font to be `AppleColorEmoji`, adding that in now as a fail safe, although I think Apple will default it for those anyways – Albert Renshaw May 16 at 20:09

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

on each letter. The problem is `isEmoji` will return true for any character that can be converted into a 2-byte emoji, such as 0-9.

2

We can look at the variable `isEmoji` and also check the for the presence of an emoji modifier to determine if the ambiguous characters will display as an emoji.

This solution should be much more future proof than the regex solutions offered here.

```
extension String {
    func containsOnlyEmojis() -> Bool {
        if count == 0 {
            return false
        }
        for character in self {
            if !character.isEmoji {
                return false
            }
        }
        return true
    }

    func containsEmoji() -> Bool {
        for character in self {
            if character.isEmoji {
                return true
            }
        }
        return false
    }
}

extension Character {
    // An emoji can either be a 2 byte unicode character or a normal UTF8 character with
    // an emoji modifier
    // appended as is the case with 3. 0x238C is the first instance of UTF16 emoji that
    // requires no modifier.
    // `isEmoji` will evaluate to true for any character that can be turned into an
    // emoji by adding a modifier
    // such as the digit "3". To avoid this we confirm that any character below 0x238C
    // has an emoji modifier attached
    var isEmoji: Bool {
        guard let scalar = unicodeScalars.first else { return false }
        return scalar.properties.isEmoji && (scalar.value > 0x238C ||
        scalar.value < 0x238C)
```

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Giving us

```
"hey".containsEmoji() //false

"Hello World 😊".containsEmoji() //true
"Hello World 😊".containsOnlyEmojis() //false

"😊".containsEmoji() //true
"😊".containsOnlyEmojis() //true
```

edited Aug 8 at 21:26

answered Aug 6 at 23:20



Miniroom

112 2 10

And what's more is `Character("3").isEmoji // true` while `Character("3").isEmoji // false` – Paul B Sep 12 at 13:01

You can use [NSString-RemoveEmoji](#) like this:

```
1 if string.isIncludingEmoji {
    }
}
```

edited Mar 10 '16 at 2:36

answered Jun 10 '15 at 13:36



clemkoa

993 2 12 27



Shardul

3,043 3 25 46

For Swift 5.0:

0 Accounts for:

- Non-Emoji emoji (e.g. ASCII characters)

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

- Variation selectors

All true:

```
"😊".isPureEmojiString(withMinLength: 1, max: 1) // 1 scalar
"👉".isPureEmojiString(withMinLength: 1, max: 1) // 2 scalars (emoji modifier)
"👉🏿".isPureEmojiString(withMinLength: 1, max: 1) // 2 scalars (variation selector)
"PR".isPureEmojiString(withMinLength: 1, max: 1) // 2 scalars (2x regional indicators)
"👨🏿".isPureEmojiString(withMinLength: 1, max: 1) // 4 scalars (ZWJ + ♀ + variation)
"👨🏿👩🏿".isPureEmojiString(withMinLength: 1, max: 1) // 7 scalars (ZW joiners)
```

```
extension String {

    func isPureEmojiString(withMinLength min: Int, max: Int) -> Bool {

        if count < min || count > max {
            return false
        }
        return isPureEmojiString()
    }

    func isPureEmojiString() -> Bool {

        for scalar in unicodeScalars {

            let prop = scalar.properties

            if prop.isJoinControl || prop.isVariationSelector || prop.isEmojiModifier {
                continue
            }
            else if scalar.properties.isEmoji == false || scalar.isASCII == true {
                return false
            }
        }
        return true
    }
}
```

edited Aug 8 at 14:30

answered Aug 7 at 19:20

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.



-1

i had the same problem and ended up making a `String` and `Character` extensions.

The code is too long to post as it actually lists all emojis (from the official unicode list v5.0) in a `CharacterSet` you can find it here:

<https://github.com/piterwilson/StringEmoji>



Constants

let emojiCharacterSet: CharacterSet

Character set containing all known emoji (as described in official Unicode List 5.0 <http://unicode.org/emoji/charts-5.0/emoji-list.html>)

String

var isEmoji: Bool { get }

Whether or not the `String` instance represents a known single Emoji character

```
print("").isEmoji // false
print("😄".isEmoji) // true
print("😄👉".isEmoji) // false (String is not a single Emoji)
```

var containsEmoji: Bool { get }

Whether or not the `String` instance contains a known Emoji character

```
print("").containsEmoji // false
print("😄".containsEmoji) // true
print("😄👉".containsEmoji) // true
```

var unicodeName: String { get }

Applies a `kCFStringTransformToUnicodeName` - `CFStringTransform` on a copy of the `String`

```
print("á".unicodeName) // \N{LATIN SMALL LETTER A WITH ACUTE}
print("👉".unicodeName) // \N{FACE WITH STUCK-OUT TONGUE AND WINKING EYE}"
```

var niceUnicodeName: String { get }

Returns the result of a `kCFStringTransformToUnicodeName` - `CFStringTransform` with `\N{` prefixes and `}` suffixes removed

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

Character

```
var isEmoji: Bool { get }
```

Whether or not the `Character` instance represents a known Emoji character

```
print("").isEmoji) // false  
print("😊".isEmoji) // true
```

edited Nov 24 '17 at 14:50

answered Nov 21 '17 at 20:33



**Juan Carlos Ospina
Gonzalez**

3,091 3 26 39