Structure

Character

A single extended grapheme cluster that approximates a user-perceived character.

Availability

iOS 8.0+

iPadOS 8.0+

macOS 10.10+

Mac Catalyst 13.0+

tvOS 9.0+

watchOS 2.0+

Xcode 6.3+

Technology

Swift Standard Library

On This Page

Declaration

Overview

Topics

Relationships

See Also

Declaration

```
@frozen struct Character
```

Overview

The `Character` type represents a character made up of one or more Unicode scalar values, grouped by a Unicode boundary algorithm. Generally, a `Character` instance matches what the reader of a string will perceive as a single character. Strings are collections of `Character` instances, so the number of visible characters is generally the most natural way to count the length of a string.

```
let greeting = "Hello! 🐥"
print("Length: \(greeting.count)")
// Prints "Length: 8"
```

Because each character in a string can be made up of one or more Unicode scalar values, the number of characters in a string may not match the length of the Unicode scalar value representation or the length of the string in a particular binary representation.

```
print("Unicode scalar value count: \(greeting.unicodeScalars.count)")
// Prints "Unicode scalar value count: 8"

print("UTF-8 representation count: \(greeting.utf8.count)")
// Prints "UTF-8 representation count: 11"
```

Every `Character` instance is composed of one or more Unicode scalar values that are grouped together as an *extended grapheme cluster*. The way these scalar values are grouped is defined by a canonical, localized, or otherwise tailored Unicode segmentation algorithm.

For example, a country's Unicode flag character is made up of two regional indicator scalar values that correspond to that country's ISO 3166-1 alpha-2 code. The alpha-2 code for The United States is "US", so its flag character is made up of the Unicode scalar values `"\u{1F1FA}"` (REGIONAL INDICATOR SYMBOL LETTER U) and `"\u{1F1F8}"` (REGIONAL

INDICATOR SYMBOL LETTER S). When placed next to each other in a string literal, these two scalar values are combined into a single grapheme cluster, represented by a `Character` instance in Swift.

```
let usFlag: Character = "\u{1F1FA}\u{1F1F8}"
print(usFlag)
// Prints "🇺🇸"
```

For more information about the Unicode terms used in this discussion, see the Unicode.org glossary. In particular, this discussion mentions extended grapheme clusters and Unicode scalar values.

Topics

Creating a Character

In addition to creating a character from a single-character string literal, you can also convert a unicode scalar value or single-character string.

`init(String)`

Creates a character from a single-character string.

Writing to a Text Stream

`func write<Target>(to: inout Target)`

Writes the character into the given output stream.

Comparing Characters

`static func == (Character, Character) -> Bool`

Returns a Boolean value indicating whether two values are equal.

`static func != (Character, Character) -> Bool`

Returns a Boolean value indicating whether two values are not equal.

`static func < (Character, Character) -> Bool`

Returns a Boolean value indicating whether the value of the first argument is less than that of the second argument.

`static func <= (Character, Character) -> Bool`

Returns a Boolean value indicating whether the value of the first argument is less than or equal to that of the second argument.

`static func > (Character, Character) -> Bool`

Returns a Boolean value indicating whether the value of the first argument is greater than that of the second argument.

`static func >= (Character, Character) -> Bool`

Returns a Boolean value indicating whether the value of the first argument is greater than or equal to that of the second argument.

Working with a Character's Unicode Values

`init(Unicode.Scalar)`

Creates a character containing the given Unicode scalar value.

`var unicodeScalars: Character.UnicodeScalarView`

`typealias Character.UnicodeScalarView`

`var isASCII: Bool`

A Boolean value indicating whether this is an ASCII character.

`var asciiValue: UInt8?`

The ASCII encoding value of this character, if it is an ASCII character.

Inspecting a Character

`var isLetter: Bool`

A Boolean value indicating whether this character is a letter.

`var isPunctuation: Bool`

A Boolean value indicating whether this character represents punctuation.

`var isNewline: Bool`

A Boolean value indicating whether this character represents a newline.

`var isWhitespace: Bool`

A Boolean value indicating whether this character represents whitespace, including newlines.

`var isSymbol: Bool`

A Boolean value indicating whether this character represents a symbol.

`var isMathSymbol: Bool`

A Boolean value indicating whether this character represents a symbol that naturally appears in mathematical contexts.

`var isCurrencySymbol: Bool`

A Boolean value indicating whether this character represents a currency symbol.

Checking a Character's Case

`var isCased: Bool`

A Boolean value indicating whether this character changes under any form of case conversion.

`var isUppercase: Bool`

A Boolean value indicating whether this character is considered uppercase.

`func uppercased() -> String`

Returns an uppercased version of this character.

`var isLowercase: Bool`

A Boolean value indicating whether this character is considered lowercase.

`func lowercased() -> String`

Returns a lowercased version of this character.

Checking a Character's Numeric Properties

`var isNumber: Bool`

A Boolean value indicating whether this character represents a number.

`var isWholeNumber: Bool`

A Boolean value indicating whether this character represents a whole number.

`var wholeNumberValue: Int?`

The numeric value this character represents, if it represents a whole number.

`var isHexDigit: Bool`

A Boolean value indicating whether this character represents a hexadecimal digit.

`var hexDigitValue: Int?`

The numeric value this character represents, if it is a hexadecimal digit.

Creating a Range Expression

`static func ..< (Character, Character) -> Range<Character>`

Returns a half-open range that contains its lower bound but not its upper bound.

`static func ... (Character, Character) -> ClosedRange<Character>`

Returns a closed range that contains both of its bounds.

`static func ..< (Character) -> PartialRangeUpTo<Character>`

Returns a partial range up to, but not including, its upper bound.

`static func ... (Character) -> PartialRangeThrough<Character>`

Returns a partial range up to, and including, its upper bound.

`static func ... (Character) -> PartialRangeFrom<Character>`

Returns a partial range extending upward from a lower bound.

Describing a Character

`var description: String`

A textual representation of this instance.

`var debugDescription: String`

A textual representation of the character, suitable for debugging.

`var customMirror: Mirror`

A mirror that reflects the `Character` instance.

~~`var customPlaygroundQuickLook: _PlaygroundQuickLook`~~

A custom playground Quick Look for the `Character` instance.

Deprecated

`func hash(into: inout Hasher)`

Hashes the essential components of this value by feeding them into the given hasher.

Infrequently Used Functionality

`init(extendedGraphemeClusterLiteral: Character)`

Creates a character with the specified value.

`init(unicodeScalarLiteral: Character)`

Creates an instance initialized to the given value.

Type Aliases

`typealias Character.ExtendedGraphemeClusterLiteralType`

A type that represents an extended grapheme cluster literal.

`typealias Character.UTF16View`

A view of a character’s contents as a collection of UTF-16 code units. See String.UTF16View for more information

`typealias Character.UTF8View`

A view of a character’s contents as a collection of UTF-8 code units. See String.UTF8View for more information

`typealias Character.UnicodeScalarLiteralType`

A type that represents a Unicode scalar literal.

Instance Properties

`var hashValue: Int`

The hash value.

`var utf16: Character.UTF16View`

A UTF-16 encoding of `self`.

`var utf8: Character.UTF8View`

A UTF-8 encoding of `self`.

Relationships

Conforms To

```
Comparable
CustomDebugStringConvertible
CustomReflectable
CustomStringConvertible
Equatable
ExpressibleByExtendedGraphemeClusterLiteral
Hashable
Sendable
TextOutputStreamable
```

See Also

Strings and Characters

`struct String`

A Unicode string value that is a collection of characters.