

[Foundation](#) / [AttributedString](#)

## Structure

# AttributedString

A value type for a string with associated attributes for portions of its text.

iOS 15.0+ | iPadOS 15.0+ | Mac Catalyst 15.0+ | macOS 12.0+ | tvOS 15.0+ | visionOS 1.0+ | watchOS 8.0+

```
@dynamicMemberLookup  
struct AttributedString
```

## Overview

Attributed strings are character strings that have attributes for individual characters or ranges of characters. Attributes provide traits like visual styles for display, accessibility for guided access, and hyperlink data for linking between data sources. Attribute keys provide the name and value type of each attribute. System frameworks like Foundation and SwiftUI define common keys, and you can define your own in custom extensions.

## String Attributes

You can apply an attribute to an entire string, or to a range within the string. The string represents each range with consistent attributes as a *run*.

[AttributedString](#) uses subscripts and dynamic member lookup to simplify working with attributes from your call points. In its most verbose form, you set an attribute by creating an [AttributeContainer](#) and merging it into an existing attributed string, like this:

```
var attributedString = AttributedString("This is a string with empty attributes.")  
var container = AttributeContainer()  
container[AttributeScopes.AppKitAttributes.ForegroundColorAttribute.self] = .red  
attributedString.mergeAttributes(container, mergePolicy: .keepNew)
```

Using the attributed string's `subscript( : )` method, you can omit the explicit use of an `AttributeContainer` and just set the attribute by its type:

```
attributedString[AttributeScopes.AppKitAttributes.ForegroundColorAttribute.self] = .green
```

Because an `AttributedString` supports dynamic member lookup — as described under `Attributes` in [The Swift Programming Language](#) — you can access its subscripts with dot syntax instead. When combined with properties like `foregroundColor` that return the attribute key type, this final form offers a natural way to set an attribute that applies to an entire string:

```
attributedString.foregroundColor = .green
```

This example works because AppKit defines an `AttributeScope`, `AttributeScopes.AppKitAttributes`, in which the property `foregroundColor` returns the type `AttributeScopes.AppKitAttributes.ForegroundColorAttribute`. Because AppKit's attribute scope implements `AttributeDynamicLookup`, the dot syntax resolves to an equivalent subscript expression, allowing `attributedString.foregroundColor` to replace `attributedString[AttributeScopes.AppKitAttributes.ForegroundColorAttribute.self]`.

You can also set an attribute to apply only to part of an attributed string, by applying the attribute to a range, as seen here:

```
var attributedString = AttributedString("The first month of your subscription is free")
guard let range = attributedString.range(of: "free") else {return}
attributedString[range].foregroundColor = .green
```

You can access portions of the string with unique combinations of attributes by iterating over the string's `runs` property.

You can define your own custom attributes by creating types that conform to `AttributedStringKey`, and collecting them in an `AttributeScope`. Custom keys should also extend `AttributeDynamicLookup`, so callers can use dot-syntax to access the attribute.

## Creating Attributed Strings with Markdown

You can create an attributed string by passing a standard `String` or `Data` instance that contains Markdown to initializers like `init(markdown:options:baseURL:)`. The attributed string creates attributes by parsing the markup in the string.

```
do {
    let thankYouString = try AttributedString(
        markdown:"**Thank you!** Please visit our [website](https://example.com)")
} catch {
    print("Couldn't parse the string. \(error.localizedDescription)")
}
```

Localized strings that you load from strings files with initializers like `init(localized:options:table:bundle:locale:comment:)` can also contain Markdown to add styling. In addition, these localized attributed string initializers can apply the `replacementIndex` attribute, which allows you to determine the range of replacement strings, whose order may vary between languages.

By declaring new attributes that conform to `MarkdownDecodableAttributedStringKey`, you can add attributes that you invoke by using Apple's Markdown extension syntax: `^[text](name:value, name:value, ...)`. See the sample code project `doc:building-a-localized-food-ordering-app` for an example of creating custom attributes and using them with Markdown.

Localized attributed strings can also use the extension syntax to indicate parts of the string where the system can apply automatic grammar agreement. See the initializers that take a `localized:` parameter for examples of this extension syntax, as used with automatic grammar agreement.

## Attribute Scopes

The `AttributedString` API defines keys for common uses, such as text styling, semantically marking up formattable types like dates and numbers, and hyperlinking. You can find these in the `AttributeScopes` enumeration, which contains attributes for AppKit, Foundation, SwiftUI, and UIKit.

You can define your own attributes by implementing `AttributedStringKey`, and reference them by name by collecting them in an `AttributeScope`.

---

## Topics

### Creating an Attributed String

`init()`

Creates an empty attributed string.

`init(AttributedString)`

Creates an attributed string from an attributed substring.

`init(String, attributes: AttributeContainer)`

Creates an attributed string from a string and an attribute container.

`init(Substring, attributes: AttributeContainer)`

Creates an attributed string from a substring and an attribute container.

`init<S>(S, attributes: AttributeContainer)`

Creates an attributed string from a character sequence and an attribute container.

`struct AttributeContainer`

A container for attribute keys and values.

## Creating a Localized Attributed String

`init(localized: String.LocalizationValue, options: AttributedString.FormattingOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?)`

Creates an attributed string by looking up a localized string from the app's bundle.

`init<S>(localized: String.LocalizationValue, options: AttributedString.FormattingOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?, including: S.Type)`

Creates an attributed string by looking up a localized string from the app's bundle, including an attribute scope.

`init<S>(localized: String.LocalizationValue, options: AttributedString.FormattingOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?, including: KeyPath<AttributeScopes, S.Type>)`

Creates an attributed string by looking up a localized string from the app's bundle, including an attribute scope that a key path identifies.

`struct LocalizationValue`

A reference to a localizable string, with optional string interpolation.

`struct FormattingOptions`

Options that affect the handling of attributes.

`init(localized: LocalizedStringResource)`

Creates a localized attributed string from a localized string resource.

`init<S>(localized: LocalizedStringResource, including: S.Type)`

Creates a localized attributed string from a localized string resource, including an attribute scope.

```
init<S>(localized: LocalizedStringResource, including: KeyPath<AttributeScopes, S.Type>)
```

Creates a localized attributed string from a localized string resource, including an attribute scope that a key path identifies.

```
struct LocalizedStringResource
```

A reference to a localizable string, accessible from another process.

## Creating a Localized Attributed String with a Default Value

```
init(localized: StaticString, defaultValue: String.LocalizationValue, options: AttributedString.FormattingOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?)
```

Creates an attributed string by looking up a localized string from the app's bundle, using a default value if necessary.

```
init<S>(localized: StaticString, defaultValue: String.LocalizationValue, options: AttributedString.FormattingOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?, including: S.Type)
```

Creates an attributed string by looking up a localized string from the app's bundle, including an attribute scope, using a default value if necessary.

```
init<S>(localized: StaticString, defaultValue: String.LocalizationValue, options: AttributedString.FormattingOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?, including: KeyPath<AttributeScopes, S.Type>)
```

Creates an attributed string by looking up a localized string from the app's bundle, including an attribute scope that a key path identifies, using a default value if necessary.

## Creating an Attributed String from Markdown

Use Markdown syntax to initialize an attributed string with text and attributes.

⋮ Instantiating Attributed Strings with Markdown Syntax

Use a Markdown-syntax string to initialize an attributed string with standard or custom attributes.

## Creating an Attributed String from a Reference Type

`init<S>(NSAttributedString, including: S.Type)` throws

Creates a value-type attributed string from a reference type, including an attribute scope.

`init<S>(NSAttributedString, including: KeyPath<AttributeScopes, S.Type>)` throws

Creates a value-type attributed string from a reference type, including an attribute scope that a key path identifies.

`init(NSAttributedString)`

Creates a value-type attributed string from a reference type.

## Creating a Duplicate Attributed String

`init<S, T>(T, including: S.Type)`

Creates an attributed string from another attributed string, including an attribute scope.

`init<S, T>(T, including: KeyPath<AttributeScopes, S.Type>)`

Creates an attributed string from another attributed string, including an attribute scope that a key path identifies.

## Applying and Modifying Attributes

`enum AttributeMergePolicy`

An enumeration of behaviors to apply when merging attributes.

`protocol NSAttributedStringAttributeMutation`

A protocol that defines in-place mutations for attributes in an attributed string.

## Using Defined Attributes

`enum AttributeScopes`

Collections of attributes that system frameworks define.

`enum AttributeDynamicLookup`

A type to support dynamic member lookup of attributes and containers.

`struct ScopedAttributeContainer`

An attribute container that allows dynamic member lookup of its contents within the specified attribute scope.

## Accessing Indices

### ☰ Accessing Indices Within an Attributed String

Access a position within an attributed string, offset from the beginning, or before or after another known position.

## Accessing Views into the Attributed String

### `struct CharacterView`

A view into the underlying storage of the attributed string, as Unicode characters.

### `struct UnicodeScalarView`

A view into the underlying storage of the attributed string, as Unicode scalars.

### `struct Runs`

An iterable view into segments of the attributed string, each of which indicates where a run of identical attributes begins or ends.

## Modifying an Attributed String

### `func insert(some AttributedStringProtocol, at: AttributedString.Index)`

Inserts the specified string at a specific index in the attributed string.

### `struct Index`

A type that represents the position of a character or code unit within an attributed string.

### `func removeSubrange(some RangeExpression<AttributedString.Index>)`

Removes a range of characters from the attributed string.

### `func replaceSubrange(some RangeExpression<AttributedString.Index>, with : some AttributedStringProtocol)`

Replaces the contents in a range of the attributed string.

## Transforming Attributes

### `func transformingAttributes<K>(K.Type, (inout AttributedString.Single AttributeTransformer<K>) -> Void) -> AttributedString`

Returns an attributed string by calling a closure that transforms one attribute of a source attributed string.

```
func transformingAttributes<K>(KeyPath<AttributeDynamicLookup, K>, (
inout AttributedString.SingleAttributeTransformer<K>) -> Void) ->
AttributedString
```

Returns an attributed string by calling a closure that transforms one attribute, which a key path identifies, of a source attributed string.

```
func transformingAttributes<K1, K2>(K1.Type, K2.Type, (inout Attributed
String.SingleAttributeTransformer<K1>, inout AttributedString.Single
AttributeTransformer<K2>) -> Void) -> AttributedString
```

Returns an attributed string by calling a closure that transforms two attributes of a source attributed string.

```
func transformingAttributes<K1, K2>(KeyPath<AttributeDynamicLookup, K1
>, KeyPath<AttributeDynamicLookup, K2>, (inout AttributedString.Single
AttributeTransformer<K1>, inout AttributedString.SingleAttribute
Transformer<K2>) -> Void) -> AttributedString
```

Returns an attributed string created by calling a closure that transforms two attributes, which key paths identify, of a source attributed string.

```
func transformingAttributes<K1, K2, K3>(K1.Type, K2.Type, K3.Type, (
inout AttributedString.SingleAttributeTransformer<K1>, inout Attributed
String.SingleAttributeTransformer<K2>, inout AttributedString.Single
AttributeTransformer<K3>) -> Void) -> AttributedString
```

Returns an attributed string by calling a closure that transforms three attributes of a source attributed string.

```
func transformingAttributes<K1, K2, K3>(KeyPath<AttributeDynamicLookup,
K1>, KeyPath<AttributeDynamicLookup, K2>, KeyPath<AttributeDynamic
Lookup, K3>, (inout AttributedString.SingleAttributeTransformer<K1>,
inout AttributedString.SingleAttributeTransformer<K2>, inout Attributed
String.SingleAttributeTransformer<K3>) -> Void) -> AttributedString
```

Returns an attributed string by calling a closure that transforms three attributes, which key paths identify, of a source attributed string.

```
func transformingAttributes<K1, K2, K3, K4>(K1.Type, K2.Type, K3.Type,
K4.Type, (inout AttributedString.SingleAttributeTransformer<K1>, inout
AttributedString.SingleAttributeTransformer<K2>, inout AttributedString
.SingleAttributeTransformer<K3>, inout AttributedString.SingleAttribute
Transformer<K4>) -> Void) -> AttributedString
```

Returns an attributed string by calling a closure that transforms four attributes of a source attributed string.

```
func transformingAttributes<K1, K2, K3, K4>(KeyPath<AttributeDynamic
Lookup, K1>, KeyPath<AttributeDynamicLookup, K2>, KeyPath<Attribute
DynamicLookup, K3>, KeyPath<AttributeDynamicLookup, K4>, (inout
AttributedString.SingleAttributeTransformer<K1>, inout AttributedString
.SingleAttributeTransformer<K2>, inout AttributedString.SingleAttribute
Transformer<K3>, inout AttributedString.SingleAttributeTransformer<K4>)
-> Void) -> AttributedString
```

Returns an attributed string created by calling a closure that transforms four attributes, which key paths identify, of a source attributed string.

```
func transformingAttributes<K1, K2, K3, K4, K5>(K1.Type, K2.Type, K3
.Type, K4.Type, K5.Type, (inout AttributedString.SingleAttribute
Transformer<K1>, inout AttributedString.SingleAttributeTransformer<K2>,
inout AttributedString.SingleAttributeTransformer<K3>, inout Attributed
String.SingleAttributeTransformer<K4>, inout AttributedString.Single
AttributeTransformer<K5>) -> Void) -> AttributedString
```

Returns an attributed string created by calling a closure that transforms five attributes of a source attributed string.

```
func transformingAttributes<K1, K2, K3, K4, K5>(KeyPath<Attribute
DynamicLookup, K1>, KeyPath<AttributeDynamicLookup, K2>, KeyPath<
AttributeDynamicLookup, K3>, KeyPath<AttributeDynamicLookup, K4>, Key
Path<AttributeDynamicLookup, K5>, (inout AttributedString.Single
AttributeTransformer<K1>, inout AttributedString.SingleAttribute
Transformer<K2>, inout AttributedString.SingleAttributeTransformer<K3>,
inout AttributedString.SingleAttributeTransformer<K4>, inout Attributed
String.SingleAttributeTransformer<K5>) -> Void) -> AttributedString
```

Returns an attributed string created by calling a closure that transforms five attributes, which key paths identify, of a source attributed string.

```
struct SingleAttributeTransformer
```

A type that transforms an attribute by altering its range or value, or by replacing it entirely.

## Accessing Whole-String Attributes

```
enum AttributeDynamicLookup
```

A type to support dynamic member lookup of attributes and containers.

```
struct ScopedAttributeContainer
```

An attribute container that allows dynamic member lookup of its contents within the specified attribute scope.

## Combining Attributed Strings

```
func append(some AttributedStringProtocol)
```

Appends a string to the attributed string.

```
static func + (AttributedString, AttributedString) -> AttributedString
```

Concatenates two attributed strings.

```
static func + (AttributedString, some AttributedStringProtocol) ->  
AttributedString
```

Concatenates two attributed strings or substrings.

```
static func += (inout AttributedString, AttributedString)
```

Appends an attributed string to another attributed string.

```
static func += (inout AttributedString, some AttributedStringProtocol)
```

Appends an attributed string or substring to another attributed string.

## Performing Automatic Grammar Agreement

```
func inflected() -> AttributedString
```

Applies automatic grammar agreement inflection rules to the attributed string and returns the result.

## Performing String Interpolation

```
struct InterpolationOptions
```

Options that affect the behavior of string interpolation on the attributed string.

## Encoding and Decoding

```
struct AttributeScopeCodableConfiguration
```

A configuration type for encoding and decoding attributed strings.

```
⋮ Encoding and Decoding Attributed String Keys
```

Protocols adopted by attribute keys to encode or decode data.

## Structures

`struct AdaptiveImageGlyph`

`struct AttributeInvalidationCondition`

`struct LineHeight`

The line height definition of a paragraph.

`struct LocalizationOptions`

Configuration options for the localization of text.

`struct MarkdownSourcePosition`

The position of attributed string text in its original Markdown source string.

`struct UTF16View`

A view of an attributed string's contents as a collection of UTF-16 code units.

`struct UTF8View`

A view of an attributed string's contents as a collection of UTF-8 code units.

## Initializers

`init(DiscontiguousAttributedSubstring)`

Creates an attributed string from a discontiguous attributed substring.

`init(localized: StaticString, defaultValue: String.LocalizationValue, options: AttributedString.LocalizationOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?)`

`init<S>(localized: StaticString, defaultValue: String.LocalizationValue, options: AttributedString.LocalizationOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?, including: S.Type)`

`init<S>(localized: StaticString, defaultValue: String.LocalizationValue, options: AttributedString.LocalizationOptions, table: String?, bundle: Bundle?, locale: Locale?, comment: StaticString?, including: KeyPath<AttributeScopes, S.Type>)`

`init(localized: LocalizedStringResource, options: AttributedString.LocalizationOptions)`

`init<S>(localized: LocalizedStringResource, options: AttributedString.LocalizationOptions, including: S.Type)`

`init<S>(localized: LocalizedStringResource, options: AttributedString.LocalizationOptions, including: KeyPath<AttributeScopes, S.Type>)`

```
init(localized: String.LocalizationValue, options: AttributedString.  
LocalizationOptions, table: String?, bundle: Bundle?, locale: Locale?,  
comment: StaticString?)
```

```
init<S>(localized: String.LocalizationValue, options: AttributedString.  
LocalizationOptions, table: String?, bundle: Bundle?, locale: Locale?,  
comment: StaticString?, including: KeyPath<AttributeScopes, S.Type>)
```

```
init<S>(localized: String.LocalizationValue, options: AttributedString.  
LocalizationOptions, table: String?, bundle: Bundle?, locale: Locale?,  
comment: StaticString?, including: S.Type)
```

```
init(transferable: AttributedStringFormatting.Transferable, in:  
EnvironmentValues) throws
```

Extract an attributed string from SwiftUI's transferable representation in a certain environment.

## Instance Methods

```
func inflected(locale: Locale, userTermOfAddress: TermOfAddress?,  
inflectionConcepts: [InflectionConcept]) -> AttributedString
```

```
func rangeOfAudioTimeRangeAttributes(intersecting: CMTimeRange) ->  
Range<AttributedString.Index>?
```

Returns the range of indices of the receiver that are part of given time range.

```
func removeSubranges(RangeSet<AttributedString.Index>)
```

Removes the elements at the given indices.

```
func replaceSelection(inout AttributedStringSelection, with: some  
AttributedStringProtocol)
```

Replace the selection with new attributed content.

```
func replaceSelection(inout AttributedStringSelection, withCharacters:  
some Collection<Character>)
```

Replace the selection with new content, attributed with the typing attributes.

```
func transform<E>(updating: inout Range<AttributedString.Index>, body:  
(inout AttributedString) throws(E) -> Void) throws(E)
```

Tracks the location of the provided range throughout the mutation closure, updating the provided range to one that represents the same effective locations after the mutation.

```
func transform<E>(updating: inout [Range<AttributedString.Index>], body: (inout AttributedString) throws(E) -> Void) throws(E)
```

Tracks the location of the provided ranges throughout the mutation closure, updating them to new ranges that represent the same effective locations after the mutation.

```
func transform<E>(updating: Range<AttributedString.Index>, body: (inout AttributedString) throws(E) -> Void) throws(E) -> Range<AttributedString.Index>?
```

Tracks the location of the provided range throughout the mutation closure, returning a new, updated range that represents the same effective locations after the mutation.

```
func transform<E>(updating: [Range<AttributedString.Index>], body: (inout AttributedString) throws(E) -> Void) throws(E) -> [Range<AttributedString.Index>]?
```

Tracks the location of the provided ranges throughout the mutation closure, returning a new, updated range that represents the same effective locations after the mutation

```
func transform<E>(updating: inout AttributedTextSelection, body: (inout AttributedString) throws(E) -> Void) throws(E)
```

Tracks the location of the selection throughout the mutation closure, updating the selection so it represents the same effective locations after the mutation.

```
func transformAttributes<E>(in: inout AttributedTextSelection, body: (inout AttributeContainer) throws(E) -> Void) throws(E)
```

Apply a change to the attributes in the entire selection.

## Subscripts

```
subscript(AttributedTextSelection) -> DiscontiguousAttributedString
```

Obtain the discontiguous substring of a selection.

```
subscript(RangeSet<AttributedString.Index>) -> DiscontiguousAttributedString
```

Returns a discontiguous substring of this discontiguous attributed string using a set of ranges to indicate the discontiguous substring bounds.

## Type Aliases

```
typealias Specification
```

```
typealias UnwrappedType
```

typealias ValueType

## Type Properties

static var defaultResolverSpecification: some ResolverSpecification

## Enumerations

enum AttributeRunBoundaries

enum TextAlignment  
The explicit alignment of text within its container.

enum WritingDirection  
The writing direction of a piece of text.

## Default Implementations

⋮ NSAttributedStringProtocol Implementations

---

# Relationships

## Conforms To

- NSAttributedStringAttributeMutation
- NSAttributedStringProtocol
- Copyable
- CustomStringConvertible
- Decodable
- DecodableWithConfiguration
- Encodable
- EncodableWithConfiguration
- Equatable
- ExpressibleByExtendedGraphemeClusterLiteral
- ExpressibleByStringLiteral
- ExpressibleByUnicodeScalarLiteral
- Hashable
- Sendable
- SendableMetatype

## See Also

### Strings with Metadata

`struct AttributedString`

A portion of an attributed string.

☰ `AttributedString Supporting Types`

Types that the attributed string, attributed substring, and helper types extend or conform to, for sharing common functionality.

`class NSAttributedString`

A string of text that manages data, layout, and stylistic information for ranges of characters to support rendering.

`class NSMutableAttributedString`

A mutable string with associated attributes (such as visual style, hyperlinks, or accessibility data) for portions of its text.