API Collection

Data Formatting

Convert numbers, dates, measurements, and other values to and from locale-aware string representations.

Overview

Foundation supports two approaches for data formatting:

- When working in Swift, use `formatted` methods directly on the types you want to format, optionally using `FormatStyle` and its subtypes to customize formatter output. This approach supports dates, integers, floating-point numbers, measurements, sequences, and person name components. Foundation caches identically-configured formatter instances internally, allowing you to focus on your app's formatting needs.
- In Objective-C, create instances of `Formatter` and its subtypes, and use the `string(for:)` method to convert objects to formatted strings.

Topics

Data formatting in Swift

{} Language Introspector

Converts data into human-readable text using formatters and locales.

`protocol FormatStyle`

A type that converts a given data type into a representation in another type, such as a string.

`struct IntegerFormatStyle`

A structure that converts between integer values and their textual representations.

`struct FloatingPointFormatStyle`

A structure that converts between floating-point values and their textual representations.

`struct FormatStyle`

A structure that converts between decimal values and their textual representations.

`struct ListFormatStyle`

A type that formats lists of items with a separator and conjunction appropriate for a given locale.

`struct StringStyle`

`struct FormatStyle`

A structure that converts between URL instances and their textual representations.

`struct FormatStyleCapitalizationContext`

The capitalization formatting context used when formatting dates and times.

Format Style Configurations

Behaviors for traits like numeric precision, rounding, and scale, used for formatting and parsing numeric values.

Data parsing in Swift

`protocol ParseableFormatStyle`

A type that can convert a given input data type into a representation in an output type.

`protocol ParseStrategy`

A type that parses an input representation, such as a formatted string, into a provided data type.

`struct IntegerParseStrategy`

A parse strategy for creating integer values from formatted strings.

`struct FloatingPointParseStrategy`

A parse strategy for creating floating-point values from formatted strings.

`struct ParseStrategy`

A parse strategy for creating decimal values from formatted strings.

Numbers and currency

`class` `NumberFormatter`

A formatter that converts between numeric values and their textual representations.

Names

`class` `PersonNameComponentsFormatter`

A formatter that provides localized representations of the components of a person's name.

`struct` `PersonNameComponents`

The separate parts of a person's name, allowing locale-aware formatting.

Dates and times

`class` `DateFormatter`

A formatter that converts between dates and their textual representations.

`class` `DateComponentsFormatter`

A formatter that creates string representations of quantities of time.

`class` `RelativeDateTimeFormatter`

A formatter that creates locale-aware string representations of a relative date or time.

`class` `DateIntervalFormatter`

A formatter that creates string representations of time intervals.

`class` `ISO8601DateFormatter`

A formatter that converts between dates and their ISO 8601 string representations.

Data sizes

`class` `ByteCountFormatter`

A formatter that converts a byte count value into a localized description that is formatted with the appropriate byte modifier (KB, MB, GB and so on).

Measurements

`class` `MeasurementFormatter`

A formatter that provides localized representations of units and measurements.

Lists

`class` `ListFormatter`

An object that provides locale-correct formatting of a list of items using the appropriate separator and conjunction.

Internationalization

`struct` `Locale`

Information about linguistic, cultural, and technological conventions for use in formatting data for presentation.

Custom formatters

`class` `Formatter`

An abstract class that declares an interface for objects that create, interpret, and validate the textual representation of values.

Automatic grammar agreement

`enum` `InflectionRule`

A rule that affects how an attributed string performs automatic grammatical agreement.

`struct` `Morphology`

A description of the grammatical properties of a string.

`struct` `TermOfAddress`

The type for representing grammatical gender in localized text.

`enum` `InflectionConcept`

An inflection method to use when localizing text.

`struct` `Pronoun`

A custom pronoun for referring to a third person.

Deprecated

`class` `LengthFormatter`

A formatter that provides localized descriptions of linear distances, such as length and height measurements.

`class` `MassFormatter`

A formatter that provides localized descriptions of mass and weight values.

`class` `EnergyFormatter`

A formatter that provides localized descriptions of energy values.

See Also

Fundamentals

Numbers, Data, and Basic Values

Work with primitive values and other fundamental types used throughout Cocoa.

Strings and Text

Create and process strings of Unicode characters, use regular expressions to find patterns, and perform natural language analysis of text.

Collections

Use arrays, dictionaries, sets, and specialized collections to store and iterate groups of objects or values.

Dates and Times

Compare dates and times, and perform calendar and time zone calculations.

Units and Measurement

Label numeric quantities with physical dimensions to allow locale-aware formatting and conversion between related units.

Filters and Sorting

Use predicates, expressions, and sort descriptors to examine elements in collections and other services.