Structure

Decimal.FormatStyle

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

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

```
struct FormatStyle
```

Overview

Instances of `Decimal.FormatStyle` create localized, human-readable text from `Decimal` numbers and parse string representations of numbers into instances of `Decimal`.

`Decimal.FormatStyle` includes two nested types, `Decimal.FormatStyle.Percent` and `Decimal.FormatStyle.Currency`, for working with percentages and currencies, respectively. Each format style includes a configuration that determines how it represents numeric values, for things like grouping, displaying signs, and variant presentations like scientific notation. `Decimal.FormatStyle` and `Decimal.FormatStyle.Percent` include a `NumberFormatStyleConfiguration`, and `Decimal.FormatStyle.Currency` includes a `CurrencyFormatStyleConfiguration`. You can customize numeric formatting for a style by adjusting its backing configuration. The system automatically caches unique configurations of a format style to enhance performance.

> **Note**
>
> Foundation provides other format style types for working with the numeric types that the Swift standard library defines. `IntegerFormatStyle` works with types that conform to `BinaryInteger`, and `FloatingPointFormatStyle` works with types that conform to `BinaryFloatingPoint`.

Formatting decimal values

Use the formatted() method to create a string representation of a decimal value using the default Decimal.FormatStyle configuration:

```
let formattedDefault = Decimal(12345.67).formatted()
// formattedDefault is "12,345.67" in en_US locale.
// Other locales may use different separator and grouping behavior.
```

You can specify a format style by providing an argument to the formatted(_:) method. The following example shows the decimal `0.1` represented in each of the available styles in the en_US locale:

```
let number: Decimal = 0.1

let formattedNumber = number.formatted(.number)
// formattedNumber is "0.1"

let formattedPercent = number.formatted(.percent)
// formattedPercent is "10%"

let formattedCurrency = number.formatted(.currency(code: "USD"))
// formattedCurrency is "$0.10"
```

Each style provides methods for updating its numeric configuration, including the number of significant digits, grouping length, and more. You can specify a numeric configuration by calling as many of these methods as you need in any order you choose. The following example shows the same number with default and custom configurations:

```
let exampleNumber: Decimal = 125000.12

let defaultFormatting = exampleNumber.formatted(.number)
// defaultFormatting is "125 000,12" for the "fr_FR" locale
// defaultFormatting is "125,000.12" for the "en_US" locale

let customFormatting = exampleNumber.formatted(
    .number
    .grouping(.never)
    .sign(strategy: .always()))
// customFormatting is "+125000.12"
```

Creating a decimal format style instance

The previous examples use static instances like `number` to create format styles within the call to the `formatted(_:)` method. You can also create a `Decimal.FormatStyle` instance and use it to repeatedly format different values by using the `format(_:)` method, as shown here:

```
let percentFormatStyle = Decimal.FormatStyle.Percent()

percentFormatStyle.format(0.5) // "50%"
percentFormatStyle.format(0.855) // "85.5%"
percentFormatStyle.format(1.0) // "100%"
```

Parsing decimal values

You can use `Decimal.FormatStyle` to parse strings into decimal values. You can define the format style within the type's initializer or pass in a format style created outside the function. The following demonstrates both approaches:

```
let price = try? Decimal("$3,500.63",
                         format: .currency(code: "USD")) // 3500.63

let priceFormatStyle = Decimal.FormatStyle.Currency(code: "USD")
let salePrice = try? Decimal("$731.67",
                             format: priceFormatStyle) // 731.67
```

Matching regular expressions

Along with parsing numeric values in strings, you can use the Swift regular expression domain-specific language to match and capture numeric substrings. The following example defines a currency format style to match and capture a currency value using US dollars and en_US numeric conventions. The rest of the regular expression ignores any characters prior to a `": "` sequence that precedes the currency substring.

```
import RegexBuilder
let source = "Payment due: $49,525.99"
let matcher = Regex {
    OneOrMore(.any)
    ": "
    Capture {
        One(.localizedCurrency(code:Locale.Currency("USD"),
                                    locale:Locale(identifier: "en_US")))
    }
}
let match = source.firstMatch(of: matcher)
let localizedDecimal = match?.1 // 49525.99
```

Topics

Creating a decimal format style

`init(locale: Locale)`

Creates a decimal format style that uses the given locale.

Customizing style behavior

`func decimalSeparator(strategy: Decimal.FormatStyle.Configuration.DecimalSeparatorDisplayStrategy) -> Decimal.FormatStyle`

Modifies the format style to use the specified decimal separator display strategy.

`func grouping(Decimal.FormatStyle.Configuration.Grouping) -> Decimal.FormatStyle`

Modifies the format style to use the specified grouping.

`func notation(Decimal.FormatStyle.Configuration.Notation) -> Decimal.FormatStyle`

Modifies the format style to use the specified notation.

`func precision(Decimal.FormatStyle.Configuration.Precision) -> Decimal.FormatStyle`

Modifies the format style to use the specified precision.

`func rounded(rule: Decimal.FormatStyle.Configuration.RoundingRule, increment: Int?) -> Decimal.FormatStyle`

Modifies the format style to use the specified rounding rule and increment.

`func scale(Double) -> Decimal.FormatStyle`

Modifies the format style to use the specified scale.

`func sign(strategy: Decimal.FormatStyle.Configuration.SignDisplayStrategy) -> Decimal.FormatStyle`

Modifies the format style to use the specified sign display strategy for displaying or omitting sign symbols.

`typealias Configuration`

The type the format style uses for configuration settings.

`enum NumberFormatStyleConfiguration`

Configuration settings for formatting numbers of different types.

Accesssing style locale

`var locale: Locale`

The locale of the format style.

Applying currency styles

`struct Currency`

A format style that converts between decimal currency values and their textual representations.

Applying measurement styles

`struct FormatStyle`

A type that provides localized representations of measurements.

Creating attributed strings

`var attributed: Decimal.FormatStyle.Attributed`

An attributed format style based on the decimal format style.

`struct Attributed`

A format style that converts integers into attributed strings.

Parsing decimals

`struct ParseStrategy`

A parse strategy for creating decimal values from formatted strings.

Supporting types

`struct Currency`

A format style that converts between decimal currency values and their textual representations.

`struct Percent`

A format style that converts between decimal percentage values and their textual representations.

Relationships

Conforms To

```
Copyable
CustomConsumingRegexComponent
Decodable
Encodable
Equatable
FormatStyle
Hashable
ParseableFormatStyle
RegexComponent
Sendable
SendableMetatype
```

See Also

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 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.