Structure

Date

A specific point in time, independent of any calendar or time zone.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 8.0+ | macOS 10.10+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
struct Date
```

Mentioned in

Encoding and Decoding Custom Types

Overview

A `Date` value encapsulate a single point in time, independent of any particular calendrical system or time zone. Date values represent a time interval relative to an absolute reference date.

The `Date` structure provides methods for comparing dates, calculating the time interval between two dates, and creating a new date from a time interval relative to another date. Use date values in conjunction with `DateFormatter` instances to create localized representations of dates and times and with `Calendar` instances to perform calendar arithmetic.

`Date` bridges to the `NSDate` class. You can use these interchangeably in code that interacts with Objective-C APIs.

Topics

Creating a Date

`init()`

Creates a date value initialized to the current date and time.

`init(timeIntervalSinceNow: TimeInterval)`

Creates a date value initialized relative to the current date and time by a given number of seconds.

`init(timeInterval: TimeInterval, since: Date)`

Creates a date value initialized relative to another given date by a given number of seconds.

`init(timeIntervalSinceReferenceDate: TimeInterval)`

Creates a date value initialized relative to 00:00:00 UTC on 1 January 2001 by a given number of seconds.

`init(timeIntervalSince1970: TimeInterval)`

Creates a date value initialized relative to 00:00:00 UTC on 1 January 1970 by a given number of seconds.

Retrieving the Current Date

`static var now: Date`

Returns a date instance that represents the current date and time, at the moment of access.

Getting Temporal Boundaries

`static let distantFuture: Date`

A date value representing a date in the distant future.

`static let distantPast: Date`

A date value representing a date in the distant past.

Comparing Dates

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

Returns true if the two `Date` values represent the same point in time.

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

Returns true if the left hand `Date` is later in time than the right hand `Date`.

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

Returns true if the left hand `Date` is earlier in time than the right hand `Date`.

`func compare(Date) -> ComparisonResult`

Compares another date to this one.

Getting Time Intervals

`func timeIntervalSince(Date) -> TimeInterval`

Returns the interval between this date and another given date.

`var timeIntervalSinceNow: TimeInterval`

The time interval between the date value and the current date and time.

`var timeIntervalSinceReferenceDate: TimeInterval`

The interval between the date value and 00:00:00 UTC on 1 January 2001.

`var timeIntervalSince1970: TimeInterval`

The interval between the date value and 00:00:00 UTC on 1 January 1970.

`static var timeIntervalSinceReferenceDate: TimeInterval`

The interval between 00:00:00 UTC on 1 January 2001 and the current date and time.

`static let timeIntervalBetween1970AndReferenceDate: Double`

The number of seconds from 1 January 1970 to the reference date, 1 January 2001.

Adding or Subtracting a Time Interval

`func addTimeInterval(TimeInterval)`

Adds a time interval to this date.

`func addingTimeInterval(TimeInterval) -> Date`

Creates a new date value by adding a time interval to this date.

`static func + (Date, TimeInterval) -> Date`

Returns a date with a specified amount of time added to it.

`static func += (inout Date, TimeInterval)`

Adds a time interval to a date.

`static func - (Date, TimeInterval) -> Date`

Returns a `Date` with a specified amount of time subtracted from it.

`static func -= (inout Date, TimeInterval)`

Subtract a `TimeInterval` from a `Date`.

Formatting a Date

`func formatted() -> String`

Generates a locale-aware string representation of a date using the default date format style.

`func formatted(date: Date.FormatStyle.DateStyle, time: Date.FormatStyle.TimeStyle) -> String`

Generates a locale-aware string representation of a date using specified date and time format styles.

`func formatted<F>(F) -> F.FormatOutput`

Generates a locale-aware string representation of a date using the specified date format style.

`struct FormatStyle`

A structure that creates a locale-appropriate string representation of a date instance and converts strings of dates and times into date instances.

`struct RelativeFormatStyle`

A format style that forms locale-aware string representations of a relative date or time.

`struct IntervalFormatStyle`

A format style that creates string representations of date intervals.

`func ISO8601Format(Date.ISO8601FormatStyle) -> String`

Generates a locale-aware string representation of a date using the ISO 8601 date format.

`struct ISO8601FormatStyle`

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

Describing Dates

`var description: String`

The representation is useful for debugging only. There are a number of options to acquire a formatted string for a date including: date formatters (see NSDateFormatter and Data Formatting Guide), and the `Date` function `description(locale:)`.

`func description(with: Locale?) -> String`

Returns a string representation of the receiver using the given locale.

~~`var customPlaygroundQuickLook: PlaygroundQuickLook`~~

A custom playground Quick Look for the date.

Deprecated

Using Reference Types

`class NSDate`

A representation of a specific point in time, independent of any calendar or time zone.

Structures

`struct AnchoredRelativeFormatStyle`

~~`struct AttributedStyle`~~

A structure that creates a locale-appropriate attributed string representation of a date instance.

Deprecated

`struct ComponentsFormatStyle`

A style for formatting a date interval in terms of specific date components.

`struct FormatString`

`struct HTTPFormatStyle`

Options for generating and parsing string representations of dates following the HTTP date format from RFC 9110 § 5.6.7.

`struct ParseStrategy`

`struct SystemClockDidChangeMessage`

`struct VerbatimFormatStyle`

A style that formats a date with an explicitly-specified style.

Initializers

`init<T>(T.ParseInput, strategy: T) throws`

`init<T, Value>(Value, strategy: T) throws`

Type Aliases

`typealias Specification`

`typealias UnwrappedType`

`typealias ValueType`

Type Properties

`static var defaultResolverSpecification: some ResolverSpecification`

Default Implementations

CustomStringConvertible Implementations

Relationships

Conforms To

```
CKRecordValueProtocol
Comparable
Copyable
CustomDebugStringConvertible
CustomReflectable
CustomStringConvertible
Decodable
Encodable
Equatable
Hashable
Plottable
PrimitivePlottableProtocol
ReferenceConvertible
Sendable
SendableMetatype
Strideable
```

See Also

Date Representations

`struct DateInterval`

The span of time between a specific start date and end date.

`typealias TimeInterval`

A number of seconds.