NSDateFormatter Class Reference

Data Management: Strings, Text, & Fonts

2009-04-26

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSDateFormatter Class Reference

Inherits from	NSFormatter : NSObject
Conforms to	NSCoding (NSFormatter) NSCopying (NSFormatter) NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in iOS 2.0 and later.
Companion guide	Data Formatting Guide
Declared in	NSDateFormatter.h

Overview

Instances of `NSDateFormatter` create string representations of `NSDate` (and `NSCalendarDate`) objects, and convert textual representations of dates and times into `NSDate` objects. You can express the representation of dates and times flexibly using pre-set format styles or custom format strings.

In general, you are encouraged to use format styles (see `timeStyle` (page 41), `dateStyle` (page 17), and `NSDateFormatterStyle` (page 45)) rather than using custom format strings, since the format for a given style reflects a user's preferences. Format styles also reflect the locale setting.

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
[dateFormatter setTimeStyle:NSDateFormatterNoStyle];
[dateFormatter setDateStyle:NSDateFormatterMediumStyle];

NSDate *date = [NSDate dateWithTimeIntervalSinceReferenceDate:118800];

NSLocale *usLocale = [[NSLocale alloc] initWithLocaleIdentifier:@"en_US"];
[dateFormatter setLocale:usLocale];

NSLog(@"Date for locale %@: %@",
      [[dateFormatter locale] localeIdentifier], [dateFormatter
stringFromDate:date]);
// Output:
// Date for locale en_US: Jan 2, 2001

NSLocale *frLocale = [[NSLocale alloc] initWithLocaleIdentifier:@"fr_FR"];
[dateFormatter setLocale:frLocale];
NSLog(@"Date for locale %@: %@",
      [[dateFormatter locale] localeIdentifier], [dateFormatter
stringFromDate:date]);
// Output:
```

```
// Date for locale fr_FR: 2 janv. 2001
```

Formatter Behaviors and OS Versions

With Mac OS X v10.4 and later, `NSDateFormatter` has two modes of operation (or behaviors). See *Data Formatting Guide* for a full description of the old and new behaviors.

iOS Note: iOS supports only the 10.4+ behavior. 10.0-style methods and format strings are not available on iOS.

By default, on Mac OS X v10.4 instances of `NSDateFormatter` have the same behavior as they did on Mac OS X versions 10.0 to 10.3. On Mac OS X v10.5 and later, `NSDateFormatter` defaults to the 10.4+ behavior.

If you initialize a formatter using `initWithDateFormat:allowNaturalLanguage:`, you are (for backwards compatibility reasons) creating an "old-style" date formatter. To use the new behavior, you initialize the formatter with `init` (page 47). If necessary, you can set the default class behavior using `setDefaultFormatterBehavior:` (page 15)), you can set the behavior for an instance using `setFormatterBehavior:` (page 26) message with the argument `NSDateFormatterBehavior10_4`.

By default, the 10.4-style formatter returns `NSDate` objects (prior to Mac OS X v10.4, date formatters returned `NSCalendarDate` objects). You can change this behavior using `setGeneratesCalendarDates:` (page 26), although this is strongly discouraged (as `NSCalendarDate` is deprecated on Mac OS X v10.6 and later).

Tasks

Initializing a Date Formatter

- `init` (page 47) Available in iOS 2.0 through iOS 3.2
  Initializes and returns an `NSDateFormatter` instance.

Managing Behavior

- `formatterBehavior` (page 19)
  Returns the formatter behavior for the receiver.
- `setFormatterBehavior:` (page 26)
  Sets the formatter behavior for the receiver.
+ `defaultFormatterBehavior` (page 14)
  Returns the default formatting behavior for instances of the class.
+ `setDefaultFormatterBehavior:` (page 15)
  Sets the default formatting behavior for instances of the class.
- `generatesCalendarDates` (page 19)
  Returns a Boolean value that indicates whether the receiver generates calendar dates.

- `setGeneratesCalendarDates:` (page 26)
  Sets whether the receiver generates calendar dates.
- `isLenient` (page 20)
  Returns a Boolean value that indicates whether the receiver uses heuristics when parsing a string.
- `setLenient:` (page 27)
  Sets whether the receiver uses heuristics when parsing a string.
- `doesRelativeDateFormatting` (page 18)
  Returns a Boolean value that indicates whether the receiver uses phrases such as “today” and “tomorrow” for the date component.
- `setDoesRelativeDateFormatting:` (page 24)
  Specifies whether the receiver uses phrases such as “today” and “tomorrow” for the date component.

Converting Objects

- `dateFromString:` (page 17)
  Returns a date representation of a given string interpreted using the receiver’s current settings.
- `stringFromDate:` (page 41)
  Returns a string representation of a given date formatted using the receiver’s current settings.
+ `localizedStringFromDate:dateStyle:timeStyle:` (page 14)
  Returns string representation of a given date formatted for the current locale using the specified date and time styles.
- `getObjectValue:forString:range:error:` (page 19)
  Returns by reference a date representation of a given string and the range of the string used, and returns a Boolean value that indicates whether the string could be parsed.

Managing Formats and Styles

- `dateFormat` (page 16)
  Returns the date format string used by the receiver.
- `setDateFormat:` (page 23)
  Sets the date format for the receiver.
- `dateStyle` (page 17)
  Returns the date style of the receiver.
- `setDateStyle:` (page 24)
  Sets the date style of the receiver.
- `timeStyle` (page 41)
  Returns the time style of the receiver.
- `setTimeStyle:` (page 33)
  Sets the time style of the receiver.
+ `dateFormatFromTemplate:options:locale:` (page 13)
  Returns a localized date format string representing the given date format components arranged appropriately for the specified locale.

Managing Attributes

- `calendar` (page 16)
  Returns the calendar for the receiver.
- `setCalendar:` (page 23)
  Sets the calendar for the receiver.
- `defaultDate` (page 17)
  Returns the default date for the receiver.
- `setDefaultDate:` (page 24)
  Sets the default date for the receiver.
- `locale` (page 21)
  Returns the locale for the receiver.
- `setLocale:` (page 27)
  Sets the locale for the receiver.
- `timeZone` (page 41)
  Returns the time zone for the receiver.
- `setTimeZone:` (page 34)
  Sets the time zone for the receiver.
- `twoDigitStartDate` (page 42)
  Returns the earliest date that can be denoted by a two-digit year specifier.
- `setTwoDigitStartDate:` (page 34)
  Sets the two-digit start date for the receiver.
- `gregorianStartDate` (page 20)
  Returns the start date of the Gregorian calendar for the receiver.
- `setGregorianStartDate:` (page 26)
  Sets the start date of the Gregorian calendar for the receiver.

Managing AM and PM Symbols

- `AMSymbol` (page 15)
  Returns the AM symbol for the receiver.
- `setAMSymbol:` (page 23)
  Sets the AM symbol for the receiver.
- `PMSymbol` (page 22)
  Returns the PM symbol for the receiver.
- `setPMSymbol:` (page 28)
  Sets the PM symbol for the receiver.

Managing Weekday Symbols

- `weekdaySymbols` (page 44)
  Returns the array of weekday symbols for the receiver.

- `setWeekdaySymbols:` (page 36)
  Sets the weekday symbols for the receiver.
- `shortWeekdaySymbols` (page 39)
  Returns the array of short weekday symbols for the receiver.
- `setShortWeekdaySymbols:` (page 31)
  Sets the short weekday symbols for the receiver.
- `veryShortWeekdaySymbols` (page 44)
  Returns the array of very short weekday symbols for the receiver.
- `setVeryShortWeekdaySymbols:` (page 36)
  Sets the vert short weekday symbols for the receiver
- `standaloneWeekdaySymbols` (page 40)
  Returns the array of standalone weekday symbols for the receiver.
- `setStandaloneWeekdaySymbols:` (page 33)
  Sets the standalone weekday symbols for the receiver.
- `shortStandaloneWeekdaySymbols` (page 38)
  Returns the array of short standalone weekday symbols for the receiver.
- `setShortStandaloneWeekdaySymbols:` (page 31)
  Sets the short standalone weekday symbols for the receiver.
- `veryShortStandaloneWeekdaySymbols` (page 43)
  Returns the array of very short standalone weekday symbols for the receiver.
- `setVeryShortStandaloneWeekdaySymbols:` (page 35)
  Sets the very short standalone weekday symbols for the receiver.

Managing Month Symbols

- `monthSymbols` (page 21)
  Returns the month symbols for the receiver.
- `setMonthSymbols:` (page 28)
  Sets the month symbols for the receiver.
- `shortMonthSymbols` (page 37)
  Returns the array of short month symbols for the receiver.
- `setShortMonthSymbols:` (page 29)
  Sets the short month symbols for the receiver.
- `veryShortMonthSymbols` (page 42)
  Returns the very short month symbols for the receiver.
- `setVeryShortMonthSymbols:` (page 34)
  Sets the very short month symbols for the receiver.
- `standaloneMonthSymbols` (page 39)
  Returns the standalone month symbols for the receiver.
- `setStandaloneMonthSymbols:` (page 32)
  Sets the standalone month symbols for the receiver.
- `shortStandaloneMonthSymbols` (page 38)
  Returns the short standalone month symbols for the receiver.

- `setShortStandaloneMonthSymbols:` (page 30)
  Sets the short standalone month symbols for the receiver.
- `veryShortStandaloneMonthSymbols` (page 43)
  Returns the very short month symbols for the receiver.
- `setVeryShortStandaloneMonthSymbols:` (page 35)
  Sets the very short standalone month symbols for the receiver.

Managing Quarter Symbols

- `quarterSymbols` (page 22)
  Returns the quarter symbols for the receiver.
- `setQuarterSymbols:` (page 29)
  Sets the quarter symbols for the receiver.
- `shortQuarterSymbols` (page 37)
  Returns the short quarter symbols for the receiver.
- `setShortQuarterSymbols:` (page 30)
  Sets the short quarter symbols for the receiver.
- `standaloneQuarterSymbols` (page 40)
  Returns the standalone quarter symbols for the receiver.
- `setStandaloneQuarterSymbols:` (page 32)
  Sets the standalone quarter symbols for the receiver.
- `shortStandaloneQuarterSymbols` (page 38)
  Returns the short standalone quarter symbols for the receiver.
- `setShortStandaloneQuarterSymbols:` (page 31)
  Sets the short standalone quarter symbols for the receiver.

Managing Era Symbols

- `eraSymbols` (page 18)
  Returns the era symbols for the receiver.
- `setEraSymbols:` (page 25)
  Sets the era symbols for the receiver.
- `longEraSymbols` (page 21)
  Returns the long era symbols for the receiver
- `setLongEraSymbols:` (page 27)
  Sets the long era symbols for the receiver.

Class Methods

dateFormatFromTemplate:options:locale:

Returns a localized date format string representing the given date format components arranged appropriately for the specified locale.

```
+ (NSString *)dateFormatFromTemplate:(NSString
    *)template options:(NSUInteger)opts locale:(NSLocale *)locale
```

Parameters

template

A string containing date format patterns (such as “MM” or “h”).

For full details, see Unicode Technical Standard #35.

opts

No options are currently defined—pass `0`.

locale

The locale for which the template is required.

Return Value

A localized date format string representing the date format components given in *template*, arranged appropriately for the locale specified by *locale*.

The returned string may not contain exactly those components given in *template*, but may—for example—have locale-specific adjustments applied.

Discussion

Different locales have different conventions for the ordering of date components. You use this method to get an appropriate format string for a given set of components for a specified locale (typically you use the current locale—see `currentLocale`).

The following example shows the difference between the date formats for British and American English:

```
NSLocale *usLocale = [[NSLocale alloc] initWithLocaleIdentifier:@"en_US"];
NSLocale *gbLocale = [[NSLocale alloc] initWithLocaleIdentifier:@"en_GB"];

NSString *dateFormat;
NSString *dateComponents = @"yMMMMd";

dateFormat = [NSDateFormatter dateFormatFromTemplate:dateComponents options:0
locale:usLocale];
NSLog(@"Date format for %@: %@",
    [usLocale displayNameForKey:NSLocaleIdentifier value:[usLocale
localeIdentifier]], dateFormat);

dateFormat = [NSDateFormatter dateFormatFromTemplate:dateComponents options:0
locale:gbLocale];
NSLog(@"Date format for %@: %@",
    [gbLocale displayNameForKey:NSLocaleIdentifier value:[gbLocale
localeIdentifier]], dateFormat);

// Output:
// Date format for English (United States): MMMM d, y
```

```
// Date format for English (United Kingdom): d MMMM y
```

Availability
Available in iOS 4.0 and later.

Declared In
NSDateFormatter.h

defaultFormatterBehavior

Returns the default formatting behavior for instances of the class.

```
+ (NSDateFormatterBehavior)defaultFormatterBehavior
```

Return Value
The default formatting behavior for instances of the class. For possible values, see NSDateFormatterBehavior (page 46).

Discussion
The default is NSDateFormatterBehavior10_0.

Availability
Available in iOS 2.0 and later.

See Also
+ setDefaultFormatterBehavior: (page 15).
- formatterBehavior (page 19)
- setFormatterBehavior: (page 26)

Declared In
NSDateFormatter.h

localizedStringFromDate:dateStyle:timeStyle:

Returns string representation of a given date formatted for the current locale using the specified date and time styles.

```
+ (NSString *)localizedStringFromDate:(NSDate
    *)date dateStyle:(NSDateFormatterStyle)dateStyle timeStyle:(NSDateFormatterStyle)timeStyle
```

Parameters

date
A date.

dateStyle
A format style for the date. For possible values, see NSDateFormatterStyle (page 45).

timeStyle
A format style for the time. For possible values, see NSDateFormatterStyle (page 45).

Return Value
A localized string representation of *date* using the specified date and time styles

Discussion
This method uses a date formatter configured with the current default settings. The returned string is the same as if you configured and used a date formatter as shown in the following example:

```
NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
[formatter setFormatterBehavior:NSDateFormatterBehavior10_4];
[formatter setDateStyle:dateStyle];
[formatter setTimeStyle:timeStyle];
NSString *result = [formatter stringForObjectValue:date];
```

Availability
Available in iOS 4.0 and later.

See Also
- `stringFromDate:` (page 41)

Declared In
`NSDateFormatter.h`

setDefaultFormatterBehavior:

Sets the default formatting behavior for instances of the class.

```
+ (void)setDefaultFormatterBehavior:(NSDateFormatterBehavior)behavior
```

Parameters

behavior

The default formatting behavior for instances of the class. For possible values, see NSDateFormatterBehavior (page 46).

Availability
Available in iOS 2.0 and later.

See Also
- `+ defaultFormatterBehavior` (page 14)
- `- formatterBehavior` (page 19)
- `- setFormatterBehavior:` (page 26)

Declared In
`NSDateFormatter.h`

Instance Methods

AMSymbol

Returns the AM symbol for the receiver.

```
- (NSString *)AMSymbol
```

Return Value
The AM symbol for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setAMSymbol:` (page 23)
- `PMSymbol` (page 22)
- `setPMSymbol:` (page 28)

Declared In
`NSDateFormatter.h`

calendar

Returns the calendar for the receiver.

```
- (NSCalendar *)calendar
```

Return Value
The calendar for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setCalendar:` (page 23)

Declared In
`NSDateFormatter.h`

dateFormat

Returns the date format string used by the receiver.

```
- (NSString *)dateFormat
```

Return Value
The date format string used by the receiver.

Discussion
See Date Format String Syntax (Mac OS X Versions 10.0 to 10.3) for a list of the conversion specifiers permitted in date format strings.

Availability
Available in iOS 2.0 and later.

See Also
- `setDateFormat:` (page 23)

Declared In
`NSDateFormatter.h`

dateFromString:

Returns a date representation of a given string interpreted using the receiver's current settings.

```
- (NSDate *)dateFromString:(NSString *)string
```

Parameters

string

The string to parse.

Return Value

A date representation of *string* interpreted using the receiver's current settings.

Availability

Available in iOS 2.0 and later.

See Also

- getObjectValue:forString:range:error: (page 19)
- stringFromDate: (page 41)

Declared In

NSDateFormatter.h

dateStyle

Returns the date style of the receiver.

```
- (NSDateFormatterStyle)dateStyle
```

Return Value

The date style of the receiver. For possible values, see NSDateFormatterStyle (page 45).

Availability

Available in iOS 2.0 and later.

See Also

- setDateStyle: (page 24)

Declared In

NSDateFormatter.h

defaultDate

Returns the default date for the receiver.

```
- (NSDate *)defaultDate
```

Return Value

The default date for the receiver.

Discussion

The default default date is nil.

Availability
Available in iOS 2.0 and later.

See Also
- `setDefaultDate:` (page 24)

Declared In
`NSDateFormatter.h`

doesRelativeDateFormatting

Returns a Boolean value that indicates whether the receiver uses phrases such as “today” and “tomorrow” for the date component.

```
- (BOOL)doesRelativeDateFormatting
```

Return Value
`YES` if the receiver uses relative date formatting, otherwise `NO`.

Discussion
For a full discussion, see `setDoesRelativeDateFormatting:` (page 24).

Availability
Available in iOS 4.0 and later.

See Also
- `setDoesRelativeDateFormatting:` (page 24)

Declared In
`NSDateFormatter.h`

eraSymbols

Returns the era symbols for the receiver.

```
- (NSArray *)eraSymbols
```

Return Value
An array containing `NSString` objects representing the era symbols for the receiver (for example, {“B.C.E.”, “C.E.”}).

Availability
Available in iOS 2.0 and later.

See Also
- `setEraSymbols:` (page 25)
- `longEraSymbols` (page 21)

Declared In
`NSDateFormatter.h`

formatterBehavior

Returns the formatter behavior for the receiver.

```
- (NSDateFormatterBehavior)formatterBehavior
```

Return Value
The formatter behavior for the receiver. For possible values, see `NSDateFormatterBehavior` (page 46).

Availability
Available in iOS 2.0 and later.

See Also
`+ defaultFormatterBehavior` (page 14).
`+ setDefaultFormatterBehavior:` (page 15)
`- setFormatterBehavior:` (page 26)

Declared In
`NSDateFormatter.h`

generatesCalendarDates

Returns a Boolean value that indicates whether the receiver generates calendar dates.

```
- (BOOL)generatesCalendarDates
```

Return Value
`YES` if the receiver generates calendar dates, otherwise `NO`.

Availability
Available in iOS 2.0 and later.

See Also
`- setGeneratesCalendarDates:` (page 26)

Declared In
`NSDateFormatter.h`

getObjectValue:forString:range:error:

Returns by reference a date representation of a given string and the range of the string used, and returns a Boolean value that indicates whether the string could be parsed.

```
- (BOOL)getObjectValue:(out id *)obj forString:(NSString *)string range:(inout
    NSRange *)rangep error:(out NSError **)error
```

Parameters
obj
If the receiver is able to parse *string*, upon return contains a date representation of *string*.
string
The string to parse.

rangep

If the receiver is able to parse *string*, upon return contains the range of *string* used to create the date.

error

If the receiver is unable to create a date by parsing *string*, upon return contains an NSError object that describes the problem.

Return Value
YES if the receiver can create a date by parsing *string*, otherwise NO.

Availability
Available in iOS 2.0 and later.

See Also
- dateFromString: (page 17)
- stringForObjectValue:

Declared In
NSDateFormatter.h

gregorianStartDate

Returns the start date of the Gregorian calendar for the receiver.

```
- (NSDate *)gregorianStartDate
```

Return Value
The start date of the Gregorian calendar for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- setGregorianStartDate: (page 26)

Declared In
NSDateFormatter.h

isLenient

Returns a Boolean value that indicates whether the receiver uses heuristics when parsing a string.

```
- (BOOL)isLenient
```

Return Value
YES if the receiver has been set to use heuristics when parsing a string to guess at the date which is intended, otherwise NO.

Availability
Available in iOS 2.0 and later.

See Also
- setLenient: (page 27)

Declared In
NSDateFormatter.h

locale

Returns the locale for the receiver.

```
- (NSLocale *)locale
```

Return Value
The locale for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- setLocale: (page 27)

Declared In
NSDateFormatter.h

longEraSymbols

Returns the long era symbols for the receiver

```
- (NSArray *)longEraSymbols
```

Return Value
An array containing NSString objects representing the era symbols for the receiver (for example, {"Before Common Era", "Common Era"}).

Availability
Available in iOS 2.0 and later.

See Also
- setLongEraSymbols: (page 27)
- eraSymbols (page 18)

Declared In
NSDateFormatter.h

monthSymbols

Returns the month symbols for the receiver.

```
- (NSArray *)monthSymbols
```

Return Value
An array of NSString objects that specify the month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setMonthSymbols:` (page 28)
- `shortMonthSymbols` (page 37)
- `veryShortMonthSymbols` (page 42)
- `standaloneMonthSymbols` (page 39)
- `shortStandaloneMonthSymbols` (page 38)
- `veryShortStandaloneMonthSymbols` (page 43)

Declared In
`NSDateFormatter.h`

PMSymbol

Returns the PM symbol for the receiver.

```
- (NSString *)PMSymbol
```

Return Value
The PM symbol for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setPMSymbol:` (page 28)
- `AMSymbol` (page 15)
- `setAMSymbol:` (page 23)

Declared In
`NSDateFormatter.h`

quarterSymbols

Returns the quarter symbols for the receiver.

```
- (NSArray *)quarterSymbols
```

Return Value
An array containing `NSString` objects representing the quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setQuarterSymbols:` (page 29)
- `shortQuarterSymbols` (page 37)
- `standaloneQuarterSymbols` (page 40)
- `shortStandaloneQuarterSymbols` (page 38)

Declared In
`NSDateFormatter.h`

setAMSymbol:

Sets the AM symbol for the receiver.

```
- (void)setAMSymbol:(NSString *)string
```

Parameters

string

The AM symbol for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- AMSymbol (page 15)
- PMSymbol (page 22)
- setPMSymbol: (page 28)

Declared In
NSDateFormatter.h

setCalendar:

Sets the calendar for the receiver.

```
- (void)setCalendar:(NSCalendar *)calendar
```

Parameters

calendar

The calendar for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- calendar (page 16)

Declared In
NSDateFormatter.h

setDateFormat:

Sets the date format for the receiver.

```
- (void)setDateFormat:(NSString *)string
```

Parameters

string

The date format for the receiver. See *Data Formatting Guide* for a list of the conversion specifiers permitted in date format strings.

Availability
Available in iOS 2.0 and later.

See Also
- `dateFormat` (page 16).

Declared In
`NSDateFormatter.h`

setDateStyle:

Sets the date style of the receiver.

```
- (void)setDateStyle:(NSDateFormatterStyle)style
```

Parameters

style

The date style of the receiver. For possible values, see NSDateFormatterStyle (page 45).

Availability
Available in iOS 2.0 and later.

See Also
- `dateStyle` (page 17).

Declared In
`NSDateFormatter.h`

setDefaultDate:

Sets the default date for the receiver.

```
- (void)setDefaultDate:(NSDate *)date
```

Parameters

date

The default date for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `defaultDate` (page 17)

Declared In
`NSDateFormatter.h`

setDoesRelativeDateFormatting:

Specifies whether the receiver uses phrases such as “today” and “tomorrow” for the date component.

```
- (void)setDoesRelativeDateFormatting:(BOOL)b
```

Parameters

b

`YES` to specify that the receiver should use relative date formatting, otherwise `NO`.

Discussion

If a date formatter uses relative date formatting, where possible it replaces the date component of its output with a phrase—such as “today” or “tomorrow”—that indicates a relative date. The available phrases depend on the locale for the date formatter; whereas, for dates in the future, English may only allow “tomorrow,” French may allow “the day after the day after tomorrow,” as illustrated in the following example.

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
[dateFormatter setTimeStyle:NSDateFormatterNoStyle];
[dateFormatter setDateStyle:NSDateFormatterMediumStyle];
NSLocale *frLocale = [[NSLocale alloc] initWithLocaleIdentifier:@"fr_FR"];
[dateFormatter setLocale:frLocale];

[dateFormatter setDoesRelativeDateFormatting:YES];

NSDate *date = [NSDate dateWithTimeIntervalSinceNow:60*60*24*3];
NSString *dateString = [dateFormatter stringFromDate:date];

NSLog(@"dateString: %@", dateString);
// Output
// dateString: après-après-demain
```

Availability

Available in iOS 4.0 and later.

See Also

`- doesRelativeDateFormatting` (page 18)

Declared In

`NSDateFormatter.h`

setEraSymbols:

Sets the era symbols for the receiver.

```
- (void)setEraSymbols:(NSArray *)array
```

Parameters

array

An array containing `NSString` objects representing the era symbols for the receiver (for example, {“B.C.E.”, “C.E.”}).

Availability

Available in iOS 2.0 and later.

See Also

`- eraSymbols` (page 18)

`- longEraSymbols` (page 21)

Declared In

`NSDateFormatter.h`

setFormatterBehavior:

Sets the formatter behavior for the receiver.

```
- (void)setFormatterBehavior:(NSDateFormatterBehavior)behavior
```

Parameters

behavior

The formatter behavior for the receiver. For possible values, see `NSDateFormatterBehavior` (page 46).

Availability
Available in iOS 2.0 and later.

See Also
`+ defaultFormatterBehavior` (page 14).
`+ setDefaultFormatterBehavior:` (page 15)
`- formatterBehavior` (page 19)

Declared In
`NSDateFormatter.h`

setGeneratesCalendarDates:

Sets whether the receiver generates calendar dates.

```
- (void)setGeneratesCalendarDates:(BOOL)b
```

Parameters

b

A Boolean value that specifies whether the receiver generates calendar dates.

Availability
Available in iOS 2.0 and later.

See Also
`- generatesCalendarDates` (page 19).

Declared In
`NSDateFormatter.h`

setGregorianStartDate:

Sets the start date of the Gregorian calendar for the receiver.

```
- (void)setGregorianStartDate:(NSDate *)array
```

Parameters

array

The start date of the Gregorian calendar for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- gregorianStartDate (page 20)

Declared In
NSDateFormatter.h

setLenient:

Sets whether the receiver uses heuristics when parsing a string.

```
- (void)setLenient:(BOOL)b
```

Parameters
b
YES to use heuristics when parsing a string to guess at the date which is intended, otherwise NO.

Discussion
If a formatter is set to be lenient, when parsing a string it uses heuristics to guess at the date which is intended. As with any guessing, it may get the result date wrong (that is, a date other than that which was intended).

Availability
Available in iOS 2.0 and later.

See Also
- isLenient (page 20)

Declared In
NSDateFormatter.h

setLocale:

Sets the locale for the receiver.

```
- (void)setLocale:(NSLocale *)locale
```

Parameters
locale
The locale for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- locale (page 21)

Declared In
NSDateFormatter.h

setLongEraSymbols:

Sets the long era symbols for the receiver.

```
- (void)setLongEraSymbols:(NSArray *)array
```

Parameters

array

An array containing `NSString` objects representing the era symbols for the receiver (for example, {“Before Common Era”, “Common Era”}).

Availability
Available in iOS 2.0 and later.

See Also
- `longEraSymbols` (page 21)
- `eraSymbols` (page 18)

Declared In
`NSDateFormatter.h`

setMonthSymbols:

Sets the month symbols for the receiver.

```
- (void)setMonthSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `monthSymbols` (page 21)
- `setShortMonthSymbols:` (page 29)
- `setVeryShortMonthSymbols:` (page 34)
- `setStandaloneMonthSymbols:` (page 32)
- `setShortStandaloneMonthSymbols:` (page 30)
- `setVeryShortStandaloneMonthSymbols:` (page 35)

Declared In
`NSDateFormatter.h`

setPMSymbol:

Sets the PM symbol for the receiver.

```
- (void)setPMSymbol:(NSString *)string
```

Parameters

string

The PM symbol for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `PMSymbol` (page 22)
- `AMSymbol` (page 15)
- `setAMSymbol:` (page 23)

Declared In
`NSDateFormatter.h`

setQuarterSymbols:

Sets the quarter symbols for the receiver.

```
- (void)setQuarterSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `quarterSymbols` (page 22)
- `setShortQuarterSymbols:` (page 30)
- `setStandaloneQuarterSymbols:` (page 32)
- `setShortStandaloneQuarterSymbols:` (page 31)

Declared In
`NSDateFormatter.h`

setShortMonthSymbols:

Sets the short month symbols for the receiver.

```
- (void)setShortMonthSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the short month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `shortMonthSymbols` (page 37)
- `setMonthSymbols:` (page 28)
- `setVeryShortMonthSymbols:` (page 34)
- `setStandaloneMonthSymbols:` (page 32)
- `setShortStandaloneMonthSymbols:` (page 30)
- `setVeryShortStandaloneMonthSymbols:` (page 35)

Declared In
NSDateFormatter.h

setShortQuarterSymbols:

Sets the short quarter symbols for the receiver.

```
- (void)setShortQuarterSymbols:(NSArray *)array
```

Parameters

array

An array of NSString objects that specify the short quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- shortQuarterSymbols (page 37)
- setQuarterSymbols: (page 29)
- setStandaloneQuarterSymbols: (page 32)
- setShortStandaloneQuarterSymbols: (page 31)

Declared In
NSDateFormatter.h

setShortStandaloneMonthSymbols:

Sets the short standalone month symbols for the receiver.

```
- (void)setShortStandaloneMonthSymbols:(NSArray *)array
```

Parameters

array

An array of NSString objects that specify the short standalone month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- shortStandaloneMonthSymbols (page 38)
- setMonthSymbols: (page 28)
- setShortMonthSymbols: (page 29)
- setVeryShortMonthSymbols: (page 34)
- setStandaloneMonthSymbols: (page 32)
- setVeryShortStandaloneMonthSymbols: (page 35)

Declared In
NSDateFormatter.h

setShortStandaloneQuarterSymbols:

Sets the short standalone quarter symbols for the receiver.

```
- (void)setShortStandaloneQuarterSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the short standalone quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `shortStandaloneQuarterSymbols` (page 38)
- `setQuarterSymbols:` (page 29)
- `setShortQuarterSymbols:` (page 30)
- `setStandaloneQuarterSymbols:` (page 32)

Declared In
`NSDateFormatter.h`

setShortStandaloneWeekdaySymbols:

Sets the short standalone weekday symbols for the receiver.

```
- (void)setShortStandaloneWeekdaySymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the short standalone weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `shortStandaloneWeekdaySymbols` (page 38)
- `setWeekdaySymbols:` (page 36)
- `setShortWeekdaySymbols:` (page 31)
- `setVeryShortWeekdaySymbols:` (page 36)
- `setStandaloneWeekdaySymbols:` (page 33)
- `setVeryShortStandaloneWeekdaySymbols:` (page 35)

Declared In
`NSDateFormatter.h`

setShortWeekdaySymbols:

Sets the short weekday symbols for the receiver.

```
- (void)setShortWeekdaySymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the short weekday symbols for the receiver.

Availability

Available in iOS 2.0 and later.

See Also

- `shortWeekdaySymbols` (page 39)
- `setWeekdaySymbols:` (page 36)
- `setVeryShortWeekdaySymbols:` (page 36)
- `setStandaloneWeekdaySymbols:` (page 33)
- `setShortStandaloneWeekdaySymbols:` (page 31)
- `setVeryShortStandaloneWeekdaySymbols:` (page 35)

Declared In

`NSDateFormatter.h`

setStandaloneMonthSymbols:

Sets the standalone month symbols for the receiver.

```
- (void)setStandaloneMonthSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the standalone month symbols for the receiver.

Availability

Available in iOS 2.0 and later.

See Also

- `standaloneMonthSymbols` (page 39)
- `setMonthSymbols:` (page 28)
- `setShortMonthSymbols:` (page 29)
- `setVeryShortMonthSymbols:` (page 34)
- `setShortStandaloneMonthSymbols:` (page 30)
- `setVeryShortStandaloneMonthSymbols:` (page 35)

Declared In

`NSDateFormatter.h`

setStandaloneQuarterSymbols:

Sets the standalone quarter symbols for the receiver.

```
- (void)setStandaloneQuarterSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the standalone quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setStandaloneQuarterSymbols:` (page 32)
- `setQuarterSymbols:` (page 29)
- `setShortQuarterSymbols:` (page 30)
- `setShortStandaloneQuarterSymbols:` (page 31)

Declared In
`NSDateFormatter.h`

setStandaloneWeekdaySymbols:

Sets the standalone weekday symbols for the receiver.

```
- (void)setStandaloneWeekdaySymbols:(NSArray *)array
```

Parameters
array
An array of `NSString` objects that specify the standalone weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `standaloneWeekdaySymbols` (page 40)
- `setWeekdaySymbols:` (page 36)
- `setShortWeekdaySymbols:` (page 31)
- `setVeryShortWeekdaySymbols:` (page 36)
- `setShortStandaloneWeekdaySymbols:` (page 31)
- `setVeryShortStandaloneWeekdaySymbols:` (page 35)

Declared In
`NSDateFormatter.h`

setTimeStyle:

Sets the time style of the receiver.

```
- (void)setTimeStyle:(NSDateFormatterStyle)style
```

Parameters
style
The time style for the receiver. For possible values, see NSDateFormatterStyle (page 45).

Availability
Available in iOS 2.0 and later.

See Also
- `timeStyle` (page 41)

Declared In
NSDateFormatter.h

setTimeZone:

Sets the time zone for the receiver.

```
- (void)setTimeZone:(NSTimeZone *)tz
```

Parameters

tz

The time zone for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- timeZone (page 41)

Declared In
NSDateFormatter.h

setTwoDigitStartDate:

Sets the two-digit start date for the receiver.

```
- (void)setTwoDigitStartDate:(NSDate *)date
```

Parameters

date

The earliest date that can be denoted by a two-digit year specifier.

Availability
Available in iOS 2.0 and later.

See Also
- twoDigitStartDate (page 42)

Declared In
NSDateFormatter.h

setVeryShortMonthSymbols:

Sets the very short month symbols for the receiver.

```
- (void)setVeryShortMonthSymbols:(NSArray *)array
```

Parameters

array

An array of NSString objects that specify the very short month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `veryShortMonthSymbols` (page 42)
- `setMonthSymbols:` (page 28)
- `setShortMonthSymbols:` (page 29)
- `setStandaloneMonthSymbols:` (page 32)
- `setShortStandaloneMonthSymbols:` (page 30)
- `setVeryShortStandaloneMonthSymbols:` (page 35)

Declared In
`NSDateFormatter.h`

setVeryShortStandaloneMonthSymbols:

Sets the very short standalone month symbols for the receiver.

```
- (void)setVeryShortStandaloneMonthSymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the very short standalone month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `veryShortStandaloneMonthSymbols` (page 43)
- `setMonthSymbols:` (page 28)
- `setShortMonthSymbols:` (page 29)
- `setVeryShortMonthSymbols:` (page 34)
- `setStandaloneMonthSymbols:` (page 32)
- `setShortStandaloneMonthSymbols:` (page 30)

Declared In
`NSDateFormatter.h`

setVeryShortStandaloneWeekdaySymbols:

Sets the very short standalone weekday symbols for the receiver.

```
- (void)setVeryShortStandaloneWeekdaySymbols:(NSArray *)array
```

Parameters

array

An array of `NSString` objects that specify the very short standalone weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `veryShortStandaloneWeekdaySymbols` (page 43)
- `setWeekdaySymbols:` (page 36)

- setShortWeekdaySymbols: (page 31)
- setVeryShortWeekdaySymbols: (page 36)
- setStandaloneWeekdaySymbols: (page 33)
- setShortStandaloneWeekdaySymbols: (page 31)

Declared In
NSDateFormatter.h

setVeryShortWeekdaySymbols:

Sets the vert short weekday symbols for the receiver

```
- (void)setVeryShortWeekdaySymbols:(NSArray *)array
```

Parameters

array
An array of NSString objects that specify the very short weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- veryShortWeekdaySymbols (page 44)
- setWeekdaySymbols: (page 36)
- setShortWeekdaySymbols: (page 31)
- setStandaloneWeekdaySymbols: (page 33)
- setShortStandaloneWeekdaySymbols: (page 31)
- setVeryShortStandaloneWeekdaySymbols: (page 35)

Declared In
NSDateFormatter.h

setWeekdaySymbols:

Sets the weekday symbols for the receiver.

```
- (void)setWeekdaySymbols:(NSArray *)array
```

Parameters

array
An array of NSString objects that specify the weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- weekdaySymbols (page 44)
- setShortWeekdaySymbols: (page 31)
- setVeryShortWeekdaySymbols: (page 36)
- setStandaloneWeekdaySymbols: (page 33)
- setShortStandaloneWeekdaySymbols: (page 31)

- setVeryShortStandaloneWeekdaySymbols: (page 35)

Declared In
NSDateFormatter.h

shortMonthSymbols

Returns the array of short month symbols for the receiver.

```
- (NSArray *)shortMonthSymbols
```

Return Value
An array containing NSString objects representing the short month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- setShortMonthSymbols: (page 29)
- monthSymbols (page 21)
- veryShortMonthSymbols (page 42)
- standaloneMonthSymbols (page 39)
- shortStandaloneMonthSymbols (page 38)
- veryShortStandaloneMonthSymbols (page 43)

Declared In
NSDateFormatter.h

shortQuarterSymbols

Returns the short quarter symbols for the receiver.

```
- (NSArray *)shortQuarterSymbols
```

Return Value
An array containing NSString objects representing the short quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- setShortQuarterSymbols: (page 30)
- quarterSymbols (page 22)
- standaloneQuarterSymbols (page 40)
- shortStandaloneQuarterSymbols (page 38)

Declared In
NSDateFormatter.h

shortStandaloneMonthSymbols

Returns the short standalone month symbols for the receiver.

```
- (NSArray *)shortStandaloneMonthSymbols
```

Return Value
An array of `NSString` objects that specify the short standalone month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setShortStandaloneMonthSymbols:` (page 30)
- `monthSymbols` (page 21)
- `shortMonthSymbols` (page 37)
- `veryShortMonthSymbols` (page 42)
- `standaloneMonthSymbols` (page 39)
- `veryShortStandaloneMonthSymbols` (page 43)

Declared In
`NSDateFormatter.h`

shortStandaloneQuarterSymbols

Returns the short standalone quarter symbols for the receiver.

```
- (NSArray *)shortStandaloneQuarterSymbols
```

Return Value
An array containing `NSString` objects representing the short standalone quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setShortStandaloneQuarterSymbols:` (page 31)
- `quarterSymbols` (page 22)
- `shortQuarterSymbols` (page 37)
- `standaloneQuarterSymbols` (page 40)

Declared In
`NSDateFormatter.h`

shortStandaloneWeekdaySymbols

Returns the array of short standalone weekday symbols for the receiver.

```
- (NSArray *)shortStandaloneWeekdaySymbols
```

Return Value
An array of `NSString` objects that specify the short standalone weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setShortStandaloneWeekdaySymbols:` (page 31)
- `weekdaySymbols` (page 44)
- `shortWeekdaySymbols` (page 39)
- `veryShortWeekdaySymbols` (page 44)
- `standaloneWeekdaySymbols` (page 40)
- `veryShortStandaloneWeekdaySymbols` (page 43)

Declared In
`NSDateFormatter.h`

shortWeekdaySymbols

Returns the array of short weekday symbols for the receiver.

```
- (NSArray *)shortWeekdaySymbols
```

Return Value
An array of `NSString` objects that specify the short weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setShortWeekdaySymbols:` (page 31)
- `weekdaySymbols` (page 44)
- `veryShortWeekdaySymbols` (page 44)
- `standaloneWeekdaySymbols` (page 40)
- `shortStandaloneWeekdaySymbols` (page 38)
- `veryShortStandaloneWeekdaySymbols` (page 43)

Declared In
`NSDateFormatter.h`

standaloneMonthSymbols

Returns the standalone month symbols for the receiver.

```
- (NSArray *)standaloneMonthSymbols
```

Return Value
An array of `NSString` objects that specify the standalone month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `monthSymbols` (page 21)

- `setStandaloneMonthSymbols:` (page 32)
- `shortMonthSymbols` (page 37)
- `veryShortMonthSymbols` (page 42)
- `shortStandaloneMonthSymbols` (page 38)
- `veryShortStandaloneMonthSymbols` (page 43)

Declared In
`NSDateFormatter.h`

standaloneQuarterSymbols

Returns the standalone quarter symbols for the receiver.

```
- (NSArray *)standaloneQuarterSymbols
```

Return Value
An array containing `NSString` objects representing the standalone quarter symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setStandaloneQuarterSymbols:` (page 32)
- `quarterSymbols` (page 22)
- `shortQuarterSymbols` (page 37)
- `shortStandaloneQuarterSymbols` (page 38)

Declared In
`NSDateFormatter.h`

standaloneWeekdaySymbols

Returns the array of standalone weekday symbols for the receiver.

```
- (NSArray *)standaloneWeekdaySymbols
```

Return Value
An array of `NSString` objects that specify the standalone weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setStandaloneWeekdaySymbols:` (page 33)
- `weekdaySymbols` (page 44)
- `shortWeekdaySymbols` (page 39)
- `veryShortWeekdaySymbols` (page 44)
- `shortStandaloneWeekdaySymbols` (page 38)
- `veryShortStandaloneWeekdaySymbols` (page 43)

Declared In
NSDateFormatter.h

stringFromDate:

Returns a string representation of a given date formatted using the receiver's current settings.

```
- (NSString *)stringFromDate:(NSDate *)date
```

Parameters

date

The date to format.

Return Value
A string representation of *date* formatted using the receiver's current settings.

Availability
Available in iOS 2.0 and later.

See Also
- dateFromString: (page 17)
+ localizedStringFromDate:dateStyle:timeStyle: (page 14)

Declared In
NSDateFormatter.h

timeStyle

Returns the time style of the receiver.

```
- (NSDateFormatterStyle)timeStyle
```

Return Value
The time style of the receiver. For possible values, see NSDateFormatterStyle (page 45).

Availability
Available in iOS 2.0 and later.

See Also
- setTimeStyle: (page 33)

Declared In
NSDateFormatter.h

timeZone

Returns the time zone for the receiver.

```
- (NSTimeZone *)timeZone
```

Return Value
The time zone for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setTimeZone:` (page 34)

Declared In
`NSDateFormatter.h`

twoDigitStartDate

Returns the earliest date that can be denoted by a two-digit year specifier.

```
- (NSDate *)twoDigitStartDate
```

Return Value
The earliest date that can be denoted by a two-digit year specifier.

Discussion
If the two-digit start date is set to January 6, 1976, then “January 1, 76” is interpreted as New Year's Day in 2076, whereas “February 14, 76” is interpreted as Valentine's Day in 1976.

The default date is December 31, 1949.

Availability
Available in iOS 2.0 and later.

See Also
- `setTwoDigitStartDate:` (page 34)

Declared In
`NSDateFormatter.h`

veryShortMonthSymbols

Returns the very short month symbols for the receiver.

```
- (NSArray *)veryShortMonthSymbols
```

Return Value
An array of `NSString` objects that specify the very short month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setVeryShortMonthSymbols:` (page 34)
- `monthSymbols` (page 21)
- `shortMonthSymbols` (page 37)
- `standaloneMonthSymbols` (page 39)
- `shortStandaloneMonthSymbols` (page 38)
- `veryShortStandaloneMonthSymbols` (page 43)

Declared In
NSDateFormatter.h

veryShortStandaloneMonthSymbols

Returns the very short month symbols for the receiver.

```
- (NSArray *)veryShortStandaloneMonthSymbols
```

Return Value
An array of NSString objects that specify the very short standalone month symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- setVeryShortStandaloneMonthSymbols: (page 35)
- monthSymbols (page 21)
- shortMonthSymbols (page 37)
- veryShortMonthSymbols (page 42)
- standaloneMonthSymbols (page 39)
- shortStandaloneMonthSymbols (page 38)

Declared In
NSDateFormatter.h

veryShortStandaloneWeekdaySymbols

Returns the array of very short standalone weekday symbols for the receiver.

```
- (NSArray *)veryShortStandaloneWeekdaySymbols
```

Return Value
An array of NSString objects that specify the very short standalone weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- setShortStandaloneWeekdaySymbols: (page 31)
- weekdaySymbols (page 44)
- shortWeekdaySymbols (page 39)
- veryShortWeekdaySymbols (page 44)
- standaloneWeekdaySymbols (page 40)
- shortStandaloneWeekdaySymbols (page 38)

Declared In
NSDateFormatter.h

veryShortWeekdaySymbols

Returns the array of very short weekday symbols for the receiver.

```
- (NSArray *)veryShortWeekdaySymbols
```

Return Value
An array of `NSString` objects that specify the very short weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setVeryShortWeekdaySymbols:` (page 36)
- `weekdaySymbols` (page 44)
- `shortWeekdaySymbols` (page 39)
- `standaloneWeekdaySymbols` (page 40)
- `shortStandaloneWeekdaySymbols` (page 38)
- `veryShortStandaloneWeekdaySymbols` (page 43)

Declared In
`NSDateFormatter.h`

weekdaySymbols

Returns the array of weekday symbols for the receiver.

```
- (NSArray *)weekdaySymbols
```

Return Value
An array of `NSString` objects that specify the weekday symbols for the receiver.

Availability
Available in iOS 2.0 and later.

See Also
- `setWeekdaySymbols:` (page 36)
- `shortWeekdaySymbols` (page 39)
- `veryShortWeekdaySymbols` (page 44)
- `standaloneWeekdaySymbols` (page 40)
- `shortStandaloneWeekdaySymbols` (page 38)
- `veryShortStandaloneWeekdaySymbols` (page 43)

Declared In
`NSDateFormatter.h`

Constants

NSDateFormatterStyle

The following constants specify predefined format styles for dates and times.

```
typedef enum {
    NSDateFormatterNoStyle     = kCFDateFormatterNoStyle,
    NSDateFormatterShortStyle  = kCFDateFormatterShortStyle,
    NSDateFormatterMediumStyle = kCFDateFormatterMediumStyle,
    NSDateFormatterLongStyle   = kCFDateFormatterLongStyle,
    NSDateFormatterFullStyle   = kCFDateFormatterFullStyle
} NSDateFormatterStyle;
```

Constants

`NSDateFormatterNoStyle`

Specifies no style.

Equal to `kCFDateFormatterNoStyle`.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

`NSDateFormatterShortStyle`

Specifies a short style, typically numeric only, such as “11/23/37” or “3:30pm”.

Equal to `kCFDateFormatterShortStyle`.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

`NSDateFormatterMediumStyle`

Specifies a medium style, typically with abbreviated text, such as “Nov 23, 1937”.

Equal to `kCFDateFormatterMediumStyle`.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

`NSDateFormatterLongStyle`

Specifies a long style, typically with full text, such as “November 23, 1937” or “3:30:32pm”.

Equal to `kCFDateFormatterLongStyle`.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

`NSDateFormatterFullStyle`

Specifies a full style with complete details, such as “Tuesday, April 12, 1952 AD” or “3:30:42pm PST”.

Equal to `kCFDateFormatterFullStyle`.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

Discussion

The format for these date and time styles is not exact because they depend on the locale, user preference settings, and the operating system version. Do not use these constants if you want an exact format.

Availability

Available in iOS 2.0 and later.

Declared In
NSDateFormatter.h

NSDateFormatterBehavior

Constants that specify the behavior `NSDateFormatter` should exhibit.

```
typedef enum {
    NSDateFormatterBehaviorDefault = 0,
    NSDateFormatterBehavior10_0    = 1000,
    NSDateFormatterBehavior10_4    = 1040,
} NSDateFormatterBehavior;
```

Constants

`NSDateFormatterBehaviorDefault`

Specifies default formatting behavior.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

`NSDateFormatterBehavior10_0`

Specifies formatting behavior equivalent to that in Mac OS X 10.0.

Available in iOS 2.0 through iOS 2.1.

Declared in `NSDateFormatter.h`.

`NSDateFormatterBehavior10_4`

Specifies formatting behavior equivalent for Mac OS X 10.4.

Available in iOS 2.0 and later.

Declared in `NSDateFormatter.h`.

Availability
Available in iOS 2.0 and later.

Declared In
NSDateFormatter.h

Deprecated NSDateFormatter Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Available in iOS 2.0 through iOS 3.2

init

Initializes and returns an `NSDateFormatter` instance. (Available in iOS 2.0 through iOS 3.2.)

```
- (id)init
```

Return Value
An `NSDateFormatter` instance initialized with locale, time zone, calendar, and behavior set to the appropriate default values.

Discussion
There are many new attributes you can get and set on a 10.4-style date formatter, including the locale, time zone, calendar, format string, the two-digit-year cross-over date, the default date which provides unspecified components, and there is also access to the various textual strings, like the month names. You are encouraged, however, not to change individual settings. Instead you should accept the default settings established on initialization and specify the format using `setDateStyle:` (page 24), `setTimeStyle:` (page 33), and appropriate style constants (see `NSDateFormatterStyle` (page 45)—these are styles that the user can configure in the International preferences panel in System Preferences).

Special Considerations

If you want the Mac OS X 10.4 behavior but have not set the class's default behavior to `NSDateFormatterBehavior10_4`, you also need to send the new instance a `setFormatterBehavior:` (page 26) message with the argument `NSDateFormatterBehavior10_4`.

Availability
Available in iOS 2.0 through iOS 3.2.

See Also
- `setDateStyle:` (page 24)
- `setTimeStyle:` (page 33)

Declared In
`NSDateFormatter.h`

 Available in iOS 2.0 through iOS 3.2

Document Revision History

This table describes the changes to *NSDateFormatter Class Reference*.

Date	Notes
2009-04-26	Updated for Mac OS X v10.6.
2008-11-19	Added note about supported behaviors for iOS.
2007-02-08	Clarified the meaning of "lenient."
2007-01-23	Included API introduced in Mac OS X v10.5.
2006-05-23	Clarified configuration of formatter after init.
	Clarified configuration of formatter after init.
	First publication of this content as a separate document.

Document Revision History