Solutions to Common Date and Time Challenges

Session 227

Chris Kane
Cocoa Frameworks

Introduction

- Brief introduction to calendar APIs
- Cover several common tasks
  - Examples are going to use new methods in OS X
- Testing

Calendrical APIs

- NSDate
- NSDateComponents
- NSCalendar
- NSTimeZone

NSDate

NSDate

- Simple value object

NSDate

- Simple value object
- Stores floating-point number of seconds since our reference date

NSDate

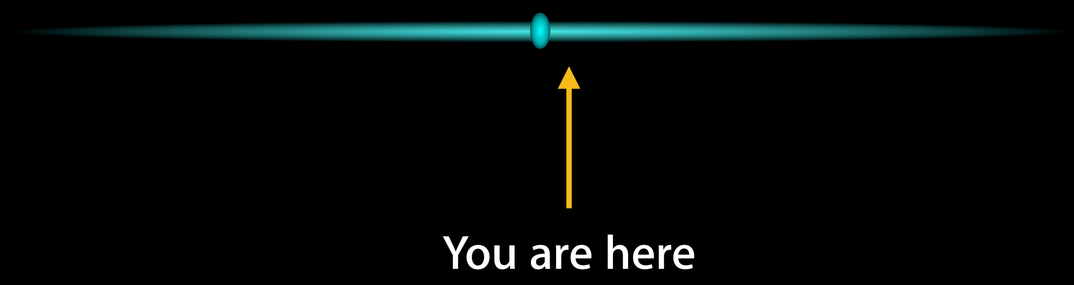
- Simple value object
- Stores floating-point number of seconds since our reference date

NSDate

- Simple value object
- Stores floating-point number of seconds since our reference date

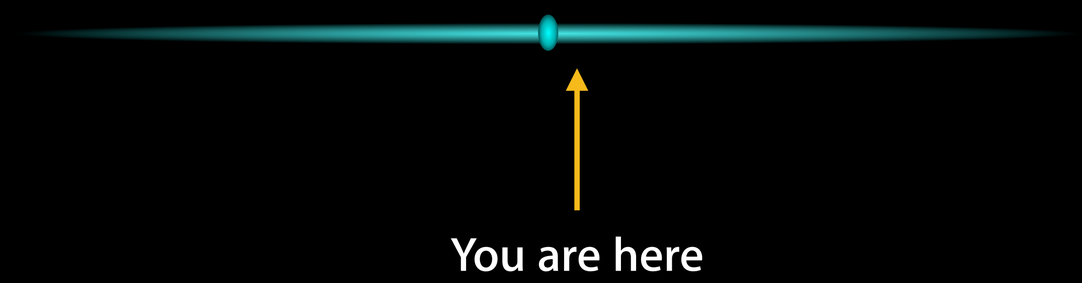
NSDate

- Simple value object
- Stores floating-point number of seconds since our reference date



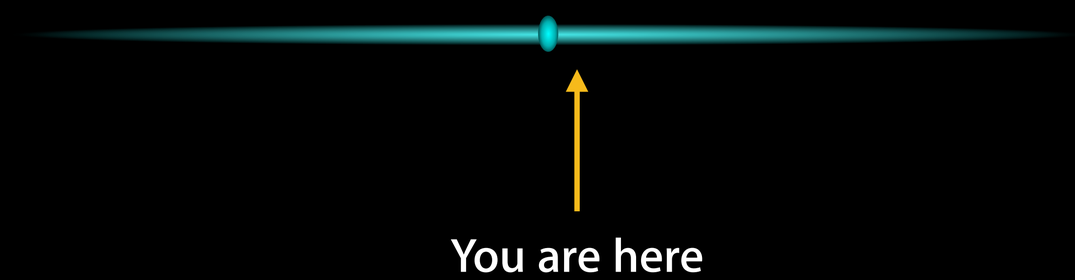
NSDate

- Simple value object
- Stores floating-point number of seconds since our reference date
- Represents both time and date



NSDate

- Simple value object
- Stores floating-point number of seconds since our reference date
- Represents both time and date

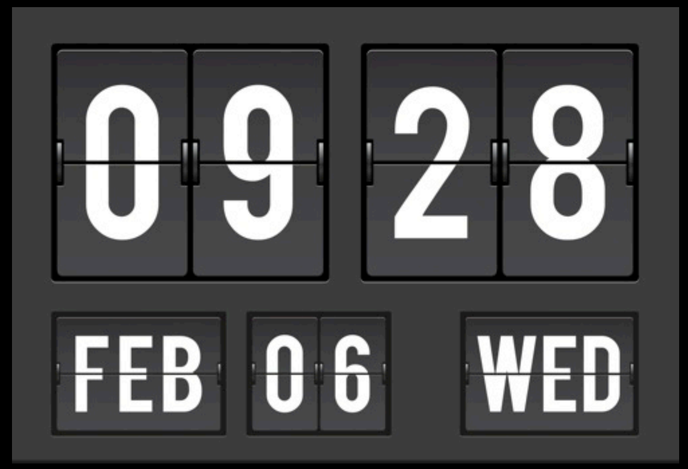



```
now = [NSDate date]
```

NSDateComponents

NSDateComponents

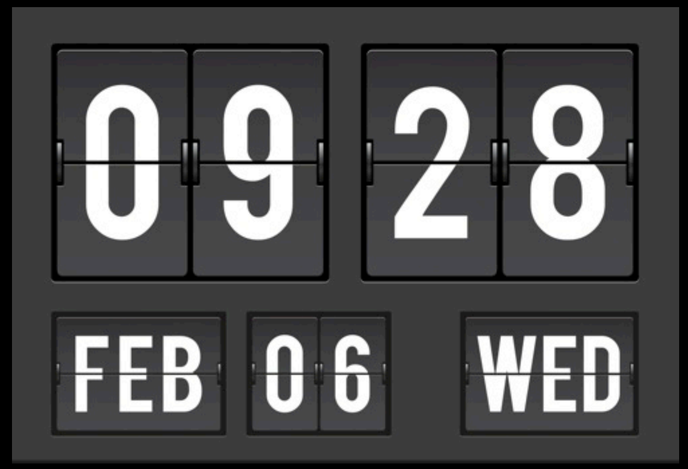
- A simple model object which stores calendar components



NSDateComponents

- A simple model object which stores calendar components

```
dateComponents.month = 6
dateComponents.day = 14
```

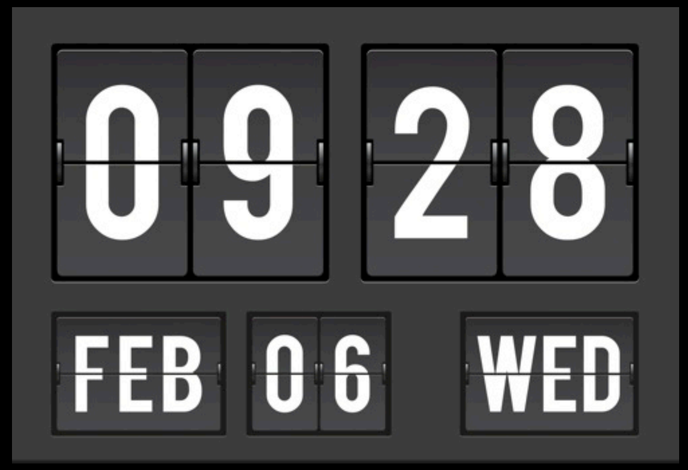


NSDateComponents

- A simple model object which stores calendar components

```
dateComponents.month = 6
dateComponents.day = 14
```

- Default value for each component is “unspecified”



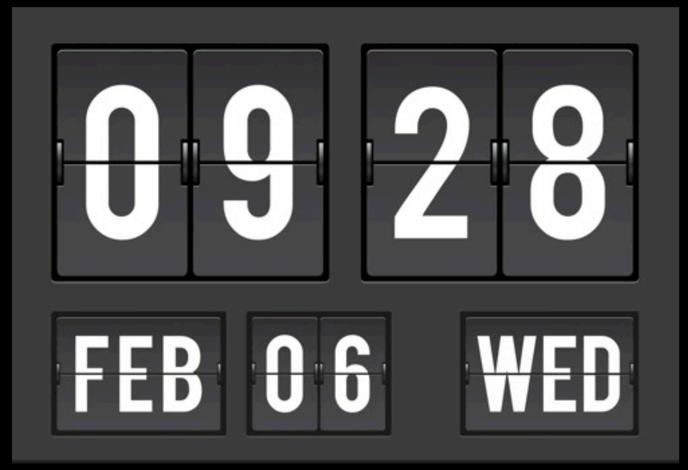
NSDateComponents

- A simple model object which stores calendar components

```
dateComponents.month = 6
dateComponents.day = 14
```

- Default value for each component is “unspecified”

```
NSDateComponentsUnspecified
```



NSCalendar

NSCalendar

- Represents many world calendars

NSCalendar

- Represents many world calendars
- Knows how to convert between NSDates and calendar components

NSCalendar

- Represents many world calendars
- Knows how to convert between NSDates and calendar components
- Contains calendar calculation APIs

NSCalendar

- Represents many world calendars
- Knows how to convert between NSDates and calendar components
- Contains calendar calculation APIs
- Several properties control calculation parameters

NSCalendar

- Represents many world calendars
- Knows how to convert between NSDates and calendar components
- Contains calendar calculation APIs
- Several properties control calculation parameters

```
cal = [NSCalendar autoupdatingCalendar]
```

NSCalendar

June 14 August 26

NSCalendar

NSCalendar

M	T	W	T	F	S	S
June	2013				1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

You are here

NSCalendar

M	T	W	T	F	S	S
June	2013				1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

When did the week with that date start, and how long is it?

You are here

NSTimeZone

NSTimeZone

- Represents time zone regions

NSTimeZone

- Represents time zone regions
- Knows about the local offset from Universal Time and when that changes

NSTimeZone

- Represents time zone regions
- Knows about the local offset from Universal Time and when that changes

```
tz = [NSTimeZone localTimeZone]
```

Common Operations

Midnight

Calculate “Midnight”

Calculate “Midnight”

- Why do people want midnight?

Calculate “Midnight”

- Why do people want midnight?
  - Using it as a default “don’t care” time

Calculate “Midnight”

- Why do people want midnight?
  - Using it as a default “don’t care” time
    - Use noon instead

Calculate “Midnight”

- Why do people want midnight?
  - Using it as a default “don’t care” time
    - Use noon instead ✓
  - Want to know when the day changes

Calculate "Midnight"

- Why do people want midnight?
  - Using it as a default "don't care" time
    - Use noon instead ✓
  - Want to know when the day changes
- Midnight can be troublesome

Calculate “Midnight”

- Why do people want midnight?
  - Using it as a default “don’t care” time
    - Use noon instead ✓
  - Want to know when the day changes
- Midnight can be troublesome
  - May not exist or there may be two

Calculate "Midnight"

- Why do people want midnight?
  - Using it as a default "don't care" time
    - Use noon instead ✓
  - Want to know when the day changes
- Midnight can be troublesome
  - May not exist or there may be two
- Think in terms of "start of a day"

Calculate Start of a Day

Calculate Start of a Day

- To calculate today's start

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

```
start = [cal startOfDayForDate:[NSDate date]];
```

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

```
start = [cal startOfDayForDate:[NSDate date]];
sometimeTomorrow = [cal dateByAddingUnit:NSCalendarUnitDay
```

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

```
start = [cal startOfDayForDate:[NSDate date]];
sometimeTomorrow = [cal dateByAddingUnit:NSCalendarUnitDay
                                   value:+1
```

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

```
start = [cal startOfDayForDate:[NSDate date]];
sometimeTomorrow = [cal dateByAddingUnit:NSCalendarUnitDay
                                   value:+1
                                  toDate:start
```

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

```
start = [cal startOfDayForDate:[NSDate date]];
sometimeTomorrow = [cal dateByAddingUnit:NSCalendarUnitDay
                                   value:+1
                                  toDate:start
                                 options:0];
```

Calculate Start of a Day

- To calculate today's start

```
start = [cal startOfDayForDate:[NSDate date]];
```

- For tomorrow's start, need more

```
start = [cal startOfDayForDate:[NSDate date]];
sometimeTomorrow = [cal dateByAddingUnit:NSCalendarUnitDay
                                   value:+1
                                  toDate:start
                                 options:0];
start = [cal startOfDayForDate:sometimeTomorrow];
```

Reacting to the Change of Day

Reacting to the Change of Day

- You may want to run some code when the day changes

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

Reacting to the Change of Day

- You may want to run some code when the day changes

  `NSCalendarDayChangedNotification`

- Example

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
```

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
observer = [noteCenter addObserverForName:NSCalendarDayChangedNotification
```

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
observer = [noteCenter addObserverForName:NSCalendarDayChangedNotification
                                   object:nil
```

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
observer = [noteCenter addObserverForName:NSCalendarDayChangedNotification
                                   object:nil
                                    queue:nil
```

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
observer = [noteCenter addObserverForName:NSCalendarDayChangedNotification
                                   object:nil
                                    queue:nil
                               usingBlock:^(NSNotification *note) {
```

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
observer = [noteCenter addObserverForName:NSCalendarDayChangedNotification
                                   object:nil
                                    queue:nil
                               usingBlock:^(NSNotification *note) {
                    // your code here
```

Reacting to the Change of Day

- You may want to run some code when the day changes

```
NSCalendarDayChangedNotification
```

- Example

```
noteCenter = [NSNotificationCenter defaultCenter];
observer = [noteCenter addObserverForName:NSCalendarDayChangedNotification
                                   object:nil
                                    queue:nil
                               usingBlock:^(NSNotification *note) {
                    // your code here
                }];
```

Specific Times

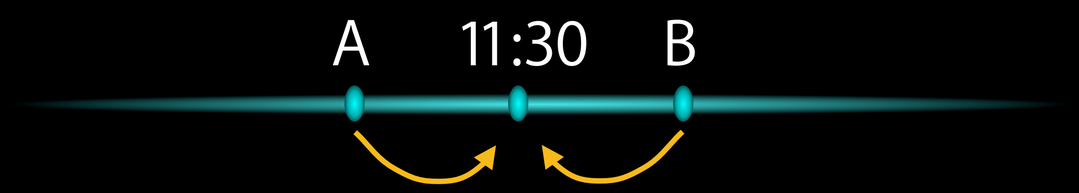
Setting a Date to a Specific Time

Setting a Date to a Specific Time

- May want a specific time in a day

Setting a Date to a Specific Time

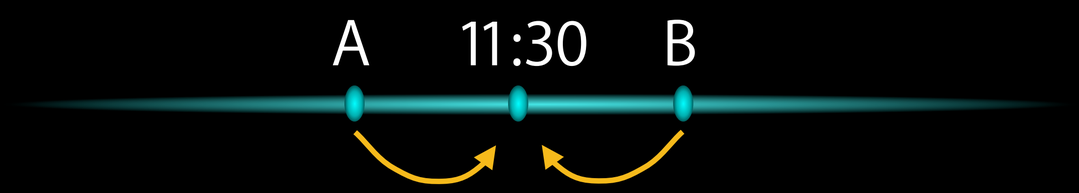
- May want a specific time in a day
- Calculate 11:30 today



Setting a Date to a Specific Time

- May want a specific time in a day
- Calculate 11:30 today

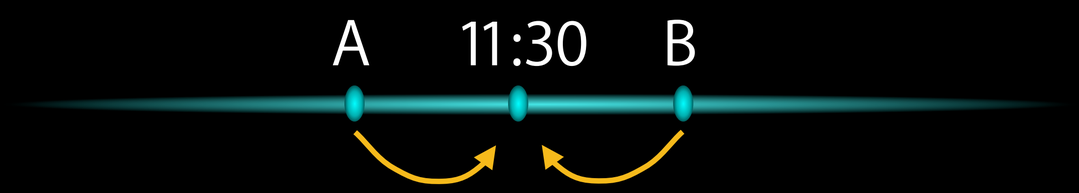
```
date = [NSDate date];
```



Setting a Date to a Specific Time

- May want a specific time in a day
- Calculate 11:30 today

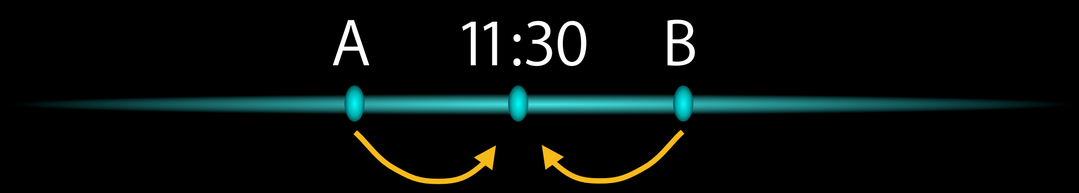
```
date = [NSDate date];
date = [cal dateBySettingHour:11
```



Setting a Date to a Specific Time

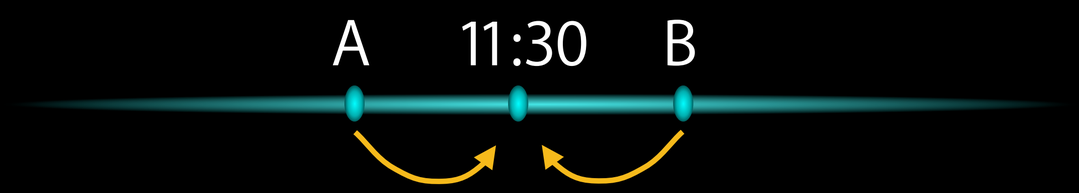
- May want a specific time in a day
- Calculate 11:30 today

```
date = [NSDate date];
date = [cal dateBySettingHour:11
            minute:30
```



Setting a Date to a Specific Time

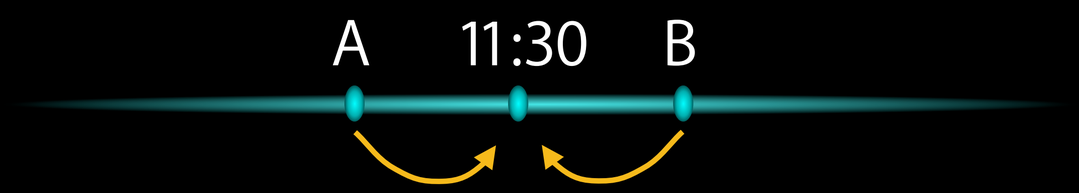
- May want a specific time in a day
- Calculate 11:30 today




```
date = [NSDate date];
date = [cal dateBySettingHour:11
            minute:30
            second:0
```

Setting a Date to a Specific Time

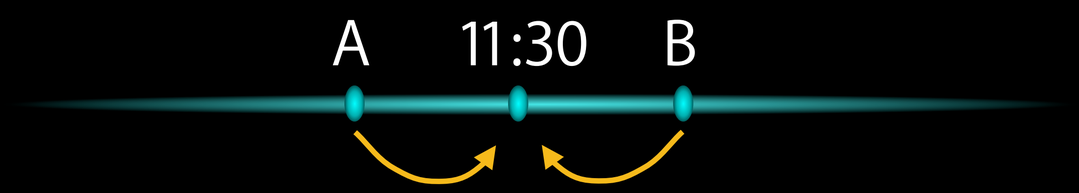
- May want a specific time in a day
- Calculate 11:30 today




```
date = [NSDate date];
date = [cal dateBySettingHour:11
            minute:30
            second:0
            toDate:date
```

Setting a Date to a Specific Time

- May want a specific time in a day
- Calculate 11:30 today




```
date = [NSDate date];
date = [cal dateBySettingHour:11
 minute:30
 second:0
 toDate:date
options:0];
```

Is This Today?

Is This Date Today?

Is This Date Today?

- Is this date object I have in today?

Is This Date Today?

- Is this date object I have in today?

```
BOOL isToday = [cal isDateInToday:date];
```

Is This Date Today?

- Is this date object I have in today?

```
BOOL isToday = [cal isDateInToday:date];
BOOL isYesterday = [cal isDateInYesterday:date];
```

Is This Date Today?

- Is this date object I have in today?

```
BOOL isToday = [cal isDateInToday:date];
BOOL isYesterday = [cal isDateInYesterday:date];
BOOL isTomorrow = [cal isDateInTomorrow:date];
```

Comparing Dates

Granular Comparison

Granular Comparison

- NSDate -compare: method is very literal

Granular Comparison

- NSDate -compare: method is very literal
- “Is this date today?” is a special case of asking if two dates are on the same day

Granular Comparison

- NSDate -compare: method is very literal
- "Is this date today?" is a special case of asking if two dates are on the same day

```
NSComparisonResult result = [cal compareDate:date
```

Granular Comparison

- NSDate -compare: method is very literal
- "Is this date today?" is a special case of asking if two dates are on the same day

```
NSComparisonResult result = [cal compareDate:date
                                      toDate:otherDate
```

Granular Comparison

- NSDate -compare: method is very literal
- "Is this date today?" is a special case of asking if two dates are on the same day

```
NSComparisonResult result = [cal compareDate:date
                                      toDate:otherDate
                           toUnitGranularity:NSCalendarUnitDay];
```

Granular Comparison

- NSDate -compare: method is very literal
- "Is this date today?" is a special case of asking if two dates are on the same day

```
NSComparisonResult result = [cal compareDate:date
                                      toDate:otherDate
                           toUnitGranularity:NSCalendarUnitDay];
BOOL sameDay = (result == NSOrderedSame);
```

Granular Comparison

- NSDate -compare: method is very literal
- "Is this date today?" is a special case of asking if two dates are on the same day

```
NSComparisonResult result = [cal compareDate:date
                                      toDate:otherDate
                           toUnitGranularity:NSCalendarUnitDay];
BOOL sameDay = (result == NSOrderedSame);
```

- Not the same as asking if two dates are within some amount of one another

Timeless Dates

Timeless Dates

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}
- Use an NSDateComponents

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}
- Use an NSDateComponents
- Or create your own simple model object

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}
- Use an NSDateComponents
- Or create your own simple model object
  - Remember to include a property for calendar

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}
- Use an NSDateComponents
- Or create your own simple model object
  - Remember to include a property for calendar
  - Or define the calendar to a fixed value

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}
- Use an NSDateComponents
- Or create your own simple model object
  - Remember to include a property for calendar
  - Or define the calendar to a fixed value
    - NOT the user's calendar

Timeless Dates

- NSDate is more than just an {Era, Year, Month, Day}
- Use an NSDateComponents
- Or create your own simple model object
  - Remember to include a property for calendar
  - Or define the calendar to a fixed value
    - NOT the user's calendar
- Same discussion applies to dateless times

Finding the Next Matching Date

Finding the Next…

Finding the Next…

- Calculate the next date matching a set of components

Finding the Next…

- Calculate the next date matching a set of components
  - Next 10:00

Finding the Next…

- Calculate the next date matching a set of components
  - Next 10:00
  - Next Wednesday

Finding the Next…

- Calculate the next date matching a set of components
  - Next 10:00
  - Next Wednesday
  - Next Wednesday at 10:00

Finding the Next…

- Calculate the next date matching a set of components
  - Next 10:00
  - Next Wednesday
  - Next Wednesday at 10:00

```
- (NSDate *)nextDateAfterDate:(NSDate *)date
          matchingComponents:(NSDateComponents *)comps
                     options:(NSCalendarOptions)options;
```

Finding the Next…

```
- (NSDate *)nextDateAfterDate:(NSDate *)date
          matchingComponents:(NSDateComponents *)comps
                     options:(NSCalendarOptions)options;
```

Finding the Next…

```
- (NSDate *)nextDateAfterDate:(NSDate *)date
          matchingComponents:(NSDateComponents *)comps
                     options:(NSCalendarOptions)options;
```

Finding the Next…

```
- (NSDate *)nextDateAfterDate:(NSDate *)date
          matchingComponents:(NSDateComponents *)comps
                     options:(NSCalendarOptions)options;
```

Finding the Next…

```
- (NSDate *)nextDateAfterDate:(NSDate *)date
          matchingComponents:(NSDateComponents *)comps
                     options:(NSCalendarOptions)options;
```

Finding the Next Wednesday at 10:00

Finding the Next Wednesday at 10:00

```
NSDateComponents *dateComponents = [NSDateComponents new];
```

Finding the Next Wednesday at 10:00

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.weekday = 4;
```

Finding the Next Wednesday at 10:00

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.weekday = 4;
dateComponents.hour = 10;
```

Finding the Next Wednesday at 10:00

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.weekday = 4;
dateComponents.hour = 10;
date = [cal nextDateAfterDate:[NSDate date]
```

Finding the Next Wednesday at 10:00

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.weekday = 4;
dateComponents.hour = 10;
date = [cal nextDateAfterDate:[NSDate date]
            matchingComponents:dateComponents
```

Finding the Next Wednesday at 10:00

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.weekday = 4;
dateComponents.hour = 10;
date = [cal nextDateAfterDate:[NSDate date]
           matchingComponents:dateComponents
                      options:<options>];
```

Matching Components

```
dateComponents.weekday = 4;
dateComponents.hour = 10;
```

Matching Components

```
dateComponents.year = ?
dateComponents.month = ?
dateComponents.weekOfYear = ?
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = ?
dateComponents.second = ?
```

Matching Components

```
dateComponents.year = NSDateComponentsUnspecified;
dateComponents.month = NSDateComponentsUnspecified;
dateComponents.weekOfYear = NSDateComponentsUnspecified;
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = NSDateComponentsUnspecified;
dateComponents.second = NSDateComponentsUnspecified;
```

Matching Components

```
dateComponents.weekday = 4;
dateComponents.hour = 10;
```

Matching Components

```
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = 0;
dateComponents.second = 0;
```

Matching Components

```
dateComponents.weekOfYear = ... match AfterDate’s value or next ...
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = 0;
dateComponents.second = 0;
```

Matching Components

```
dateComponents.weekOfYear = ... match AfterDate's value or next ...
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = 0;
dateComponents.second = 0;
```

M	T	W	T	F	S	S
June	2013				1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Matching Components

```
dateComponents.weekOfYear = ... match AfterDate's value or next ...
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = 0;
dateComponents.second = 0;
```

M	T	W	T	F	S	S
June	2013				1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Matching Components

```
dateComponents.year = ... match AfterDate's value or next ...
dateComponents.month = ... match AfterDate's value or next ...
dateComponents.weekOfYear = ... match AfterDate's value or next ...
dateComponents.weekday = 4;
dateComponents.hour = 10;
dateComponents.minute = 0;
dateComponents.second = 0;
```

M	T	W	T	F	S	S
June	2013				1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Finding Tomorrow's "Midnight"

Finding Tomorrow's "Midnight"

```
NSDateComponents *dateComponents = [NSDateComponents new];
```

Finding Tomorrow's "Midnight"

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.hour = 0;
```

Finding Tomorrow's "Midnight"

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.hour = 0;
date = [cal nextDateAfterDate:[NSDate date]
```

Finding Tomorrow's "Midnight"

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.hour = 0;
date = [cal nextDateAfterDate:[NSDate date]
           matchingComponents:dateComponents
```

Finding Tomorrow's "Midnight"

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.hour = 0;
date = [cal nextDateAfterDate:[NSDate date]
           matchingComponents:dateComponents
                      options:<options>];
```

No Possible Result

No Possible Result

```
NSDateComponents *dateComponents = [NSDateComponents new];
```

No Possible Result

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.day = 50;
```

No Possible Result

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.day = 50;
date = [cal nextDateAfterDate:[NSDate date]
```

No Possible Result

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.day = 50;
date = [cal nextDateAfterDate:[NSDate date]
           matchingComponents:dateComponents
```

No Possible Result

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.day = 50;
date = [cal nextDateAfterDate:[NSDate date]
           matchingComponents:dateComponents
                      options:<options>];
```

No Possible Result

```
NSDateComponents *dateComponents = [NSDateComponents new];
dateComponents.day = 50;
date = [cal nextDateAfterDate:[NSDate date]
           matchingComponents:dateComponents
                      options:<options>];
// date is nil
```

Find Next…Options

Find Next…Options

- `NSCalendarMatchStrictly`

Find Next…Options

- `NSCalendarMatchStrictly`
- `NSCalendarSearchBackwards`

Find Next…Options

- `NSCalendarMatchStrictly`
- `NSCalendarSearchBackwards`
- Options for missing matches

```
NSCalendarMatchNextTime
NSCalendarMatchNextTimePreservingSmallerUnits
NSCalendarMatchPreviousTimePreservingSmallerUnits
```

Find Next…Options

- `NSCalendarMatchStrictly`
- `NSCalendarSearchBackwards`
- Options for missing matches

  `NSCalendarMatchNextTime`
  `NSCalendarMatchNextTimePreservingSmallerUnits`
  `NSCalendarMatchPreviousTimePreservingSmallerUnits`
- Options for multiple matches

  `NSCalendarMatchFirst (default)`
  `NSCalendarMatchLast`

One of These Is Required

- `NSCalendarMatchStrictly`
- `NSCalendarSearchBackwards`
- Options for missing matches

  `NSCalendarMatchNextTime`
  `NSCalendarMatchNextTimePreservingSmallerUnits`
  `NSCalendarMatchPreviousTimePreservingSmallerUnits`
- Options for multiple matches

  `NSCalendarMatchFirst`
  `NSCalendarMatchLast`

Next 02:30

Next 02:30

- In U.S., transition into Daylight Saving Time skips 02:00

Next 02:30

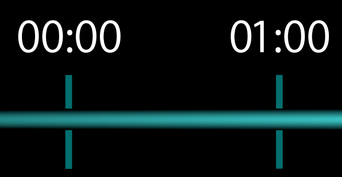
- In U.S., transition into Daylight Saving Time skips 02:00

Next 02:30

- In U.S., transition into Daylight Saving Time skips 02:00

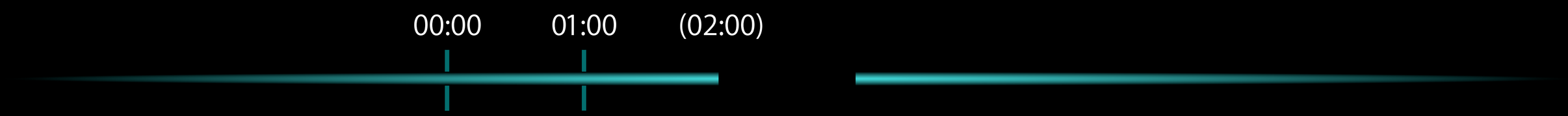
Next 02:30

- In U.S., transition into Daylight Saving Time skips 02:00



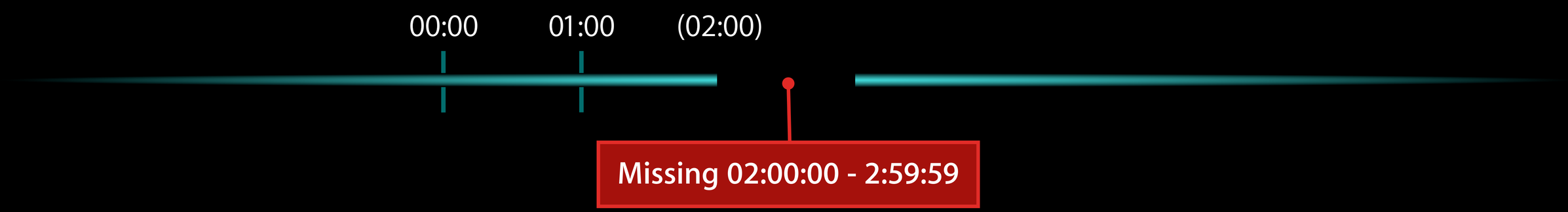
Next 02:30

- In U.S., transition into Daylight Saving Time skips 02:00



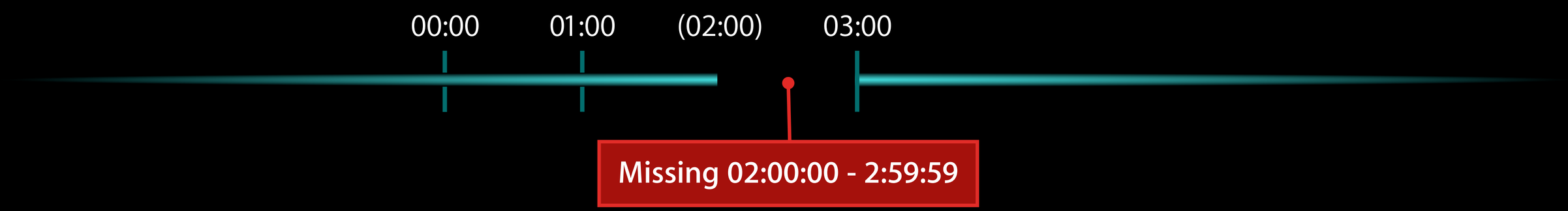
Next 02:30

- In U.S., transition into Daylight Saving Time skips 02:00



Next 02:30

- In U.S., transition into Daylight Saving Time skips 02:00

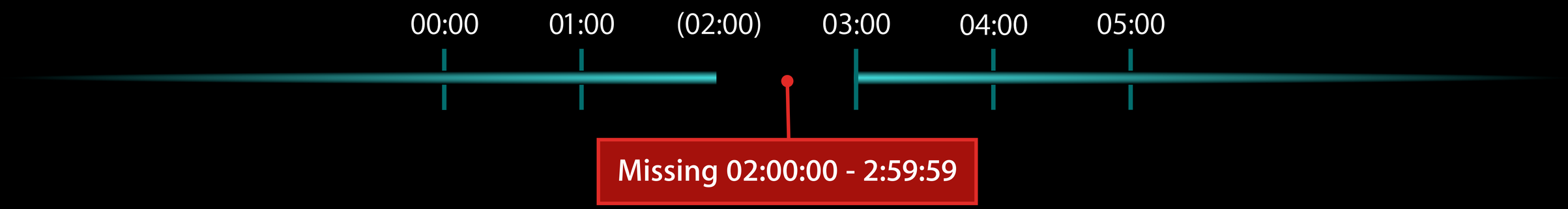


Next 02:30

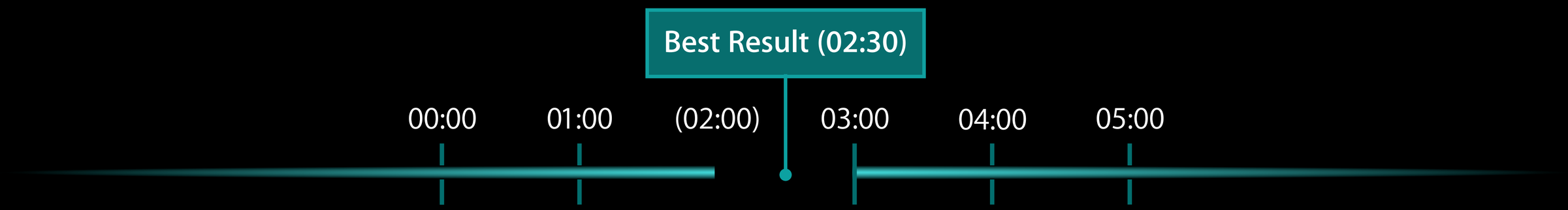
- In U.S., transition into Daylight Saving Time skips 02:00

Next 02:30

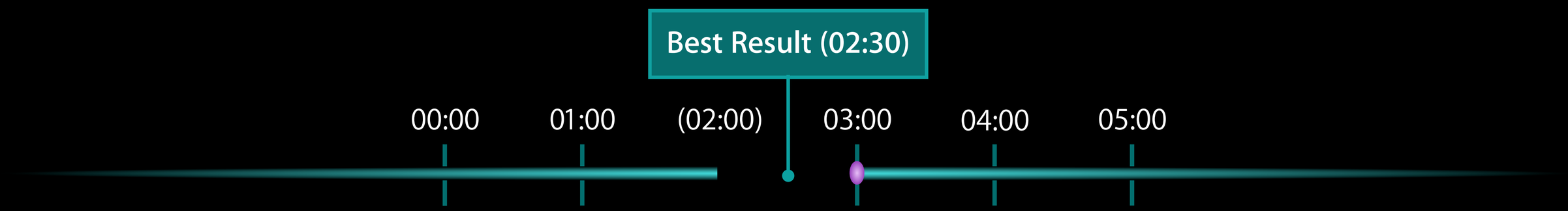
- In U.S., transition into Daylight Saving Time skips 02:00



Options for Missing Results

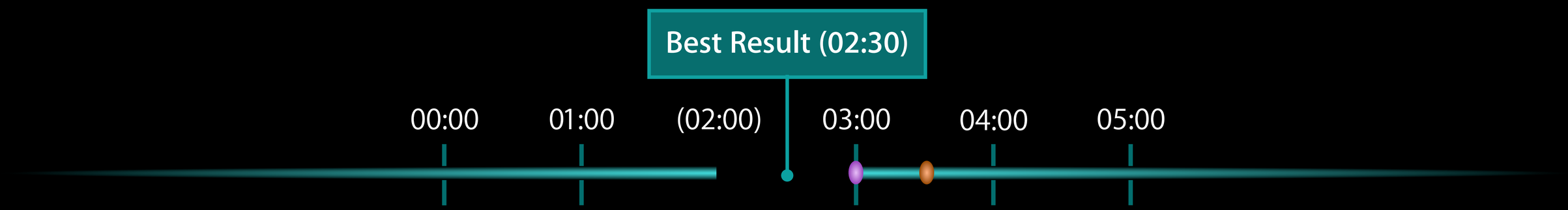


Options for Missing Results



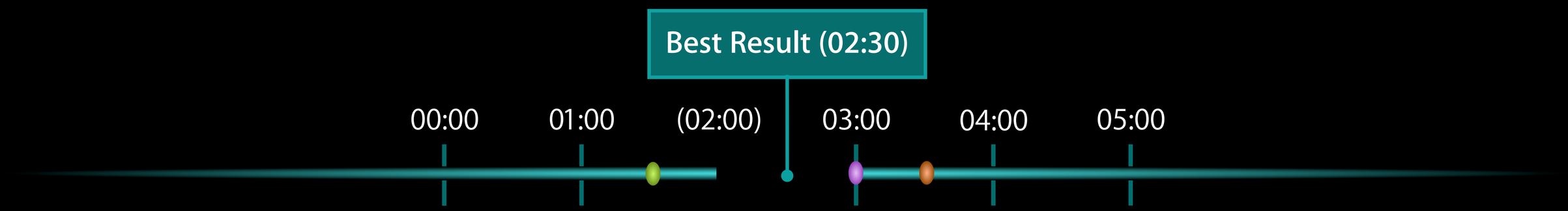
- NSCalendarMatchNextTime

Options for Missing Results



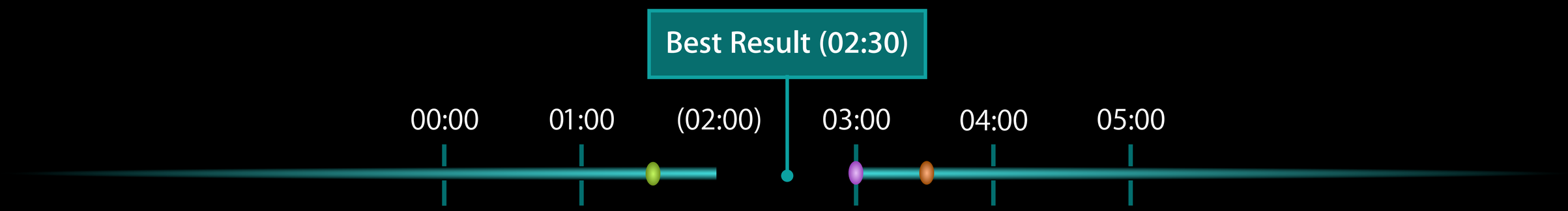
- NSCalendarMatchNextTime
- NSCalendarMatchNextTimePreservingSmallerUnits

Options for Missing Results



- NSCalendarMatchNextTime
- NSCalendarMatchNextTimePreservingSmallerUnits
- NSCalendarMatchPreviousTimePreservingSmallerUnits

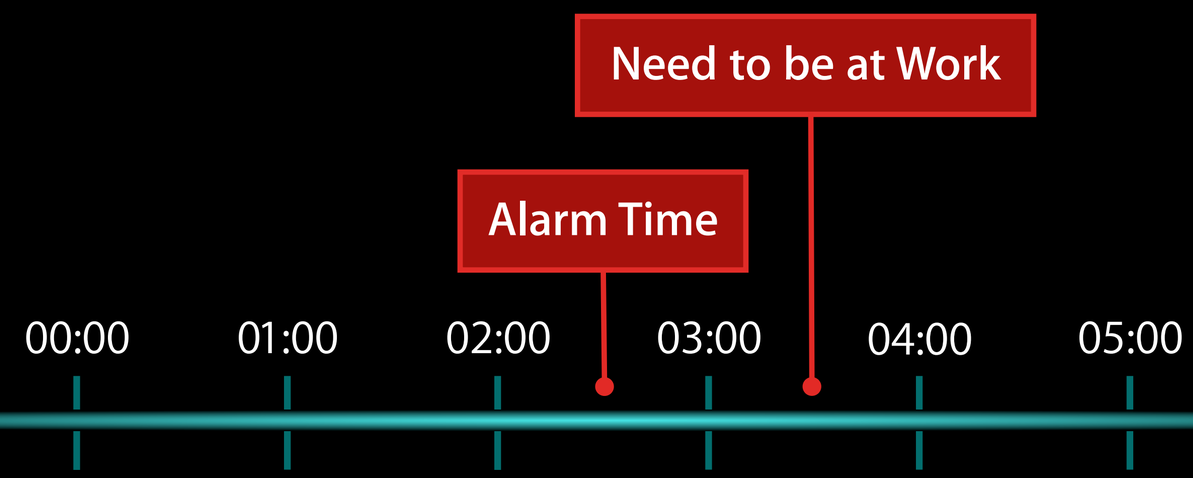
Options for Missing Results



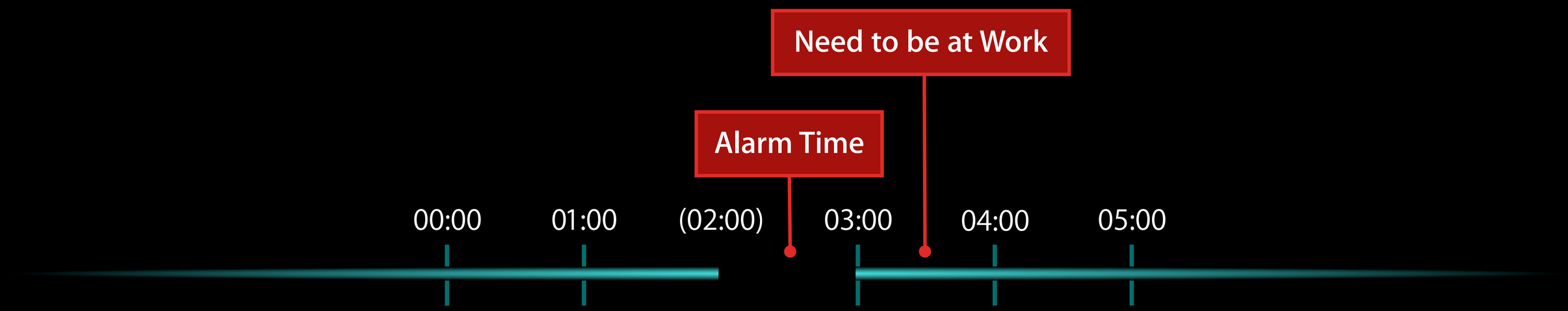
- NSCalendarMatchNextTime
- NSCalendarMatchNextTimePreservingSmallerUnits
- NSCalendarMatchPreviousTimePreservingSmallerUnits
- NSCalendarMatchStrictly – result in next day

What Is the Best Option?

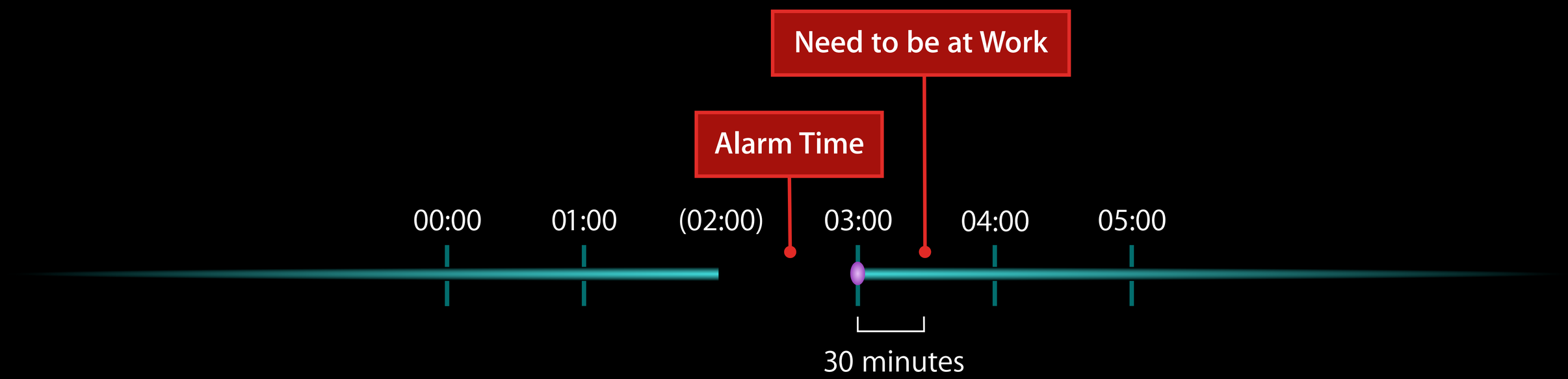
What Is the Best Option?



What Is the Best Option?

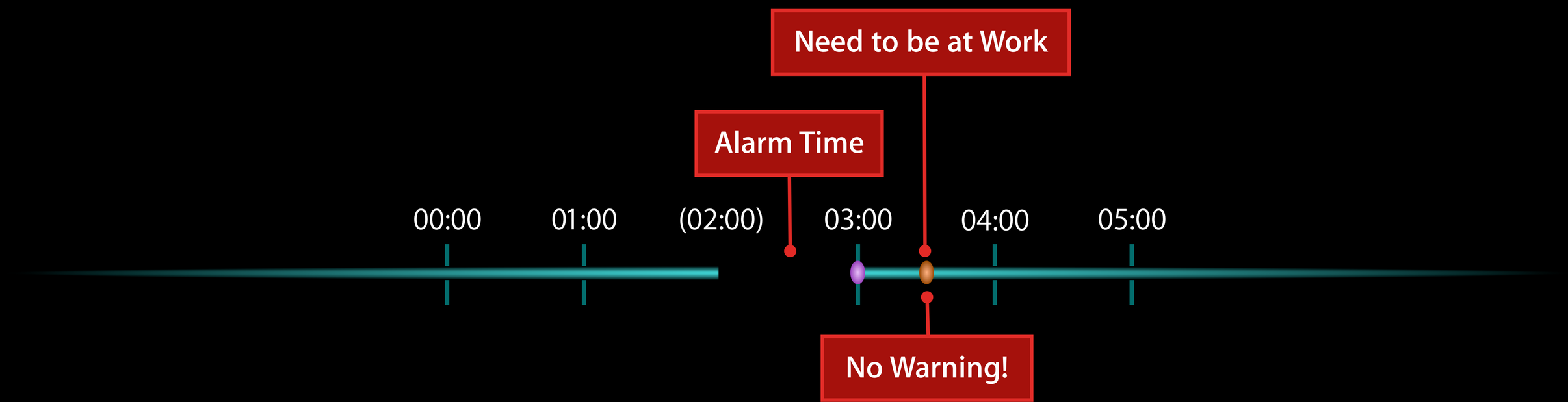


What Is the Best Option?



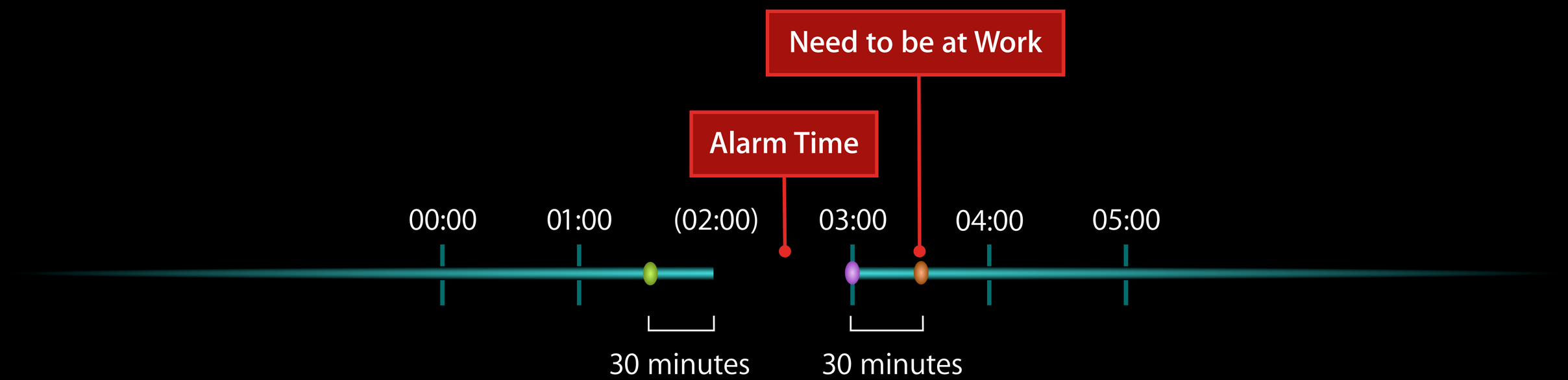
- `NSCalendarMatchNextTime`

What Is the Best Option?



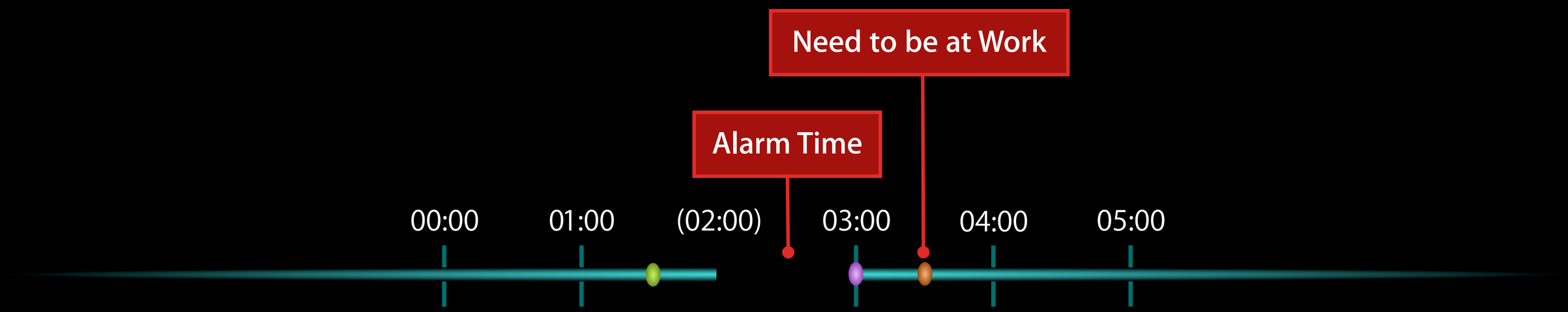
- NSCalendarMatchNextTime
- NSCalendarMatchNextTimePreservingSmallerUnits

What Is the Best Option?



- `NSCalendarMatchNextTime`
- `NSCalendarMatchNextTimePreservingSmallerUnits`
- `NSCalendarMatchPreviousTimePreservingSmallerUnits`

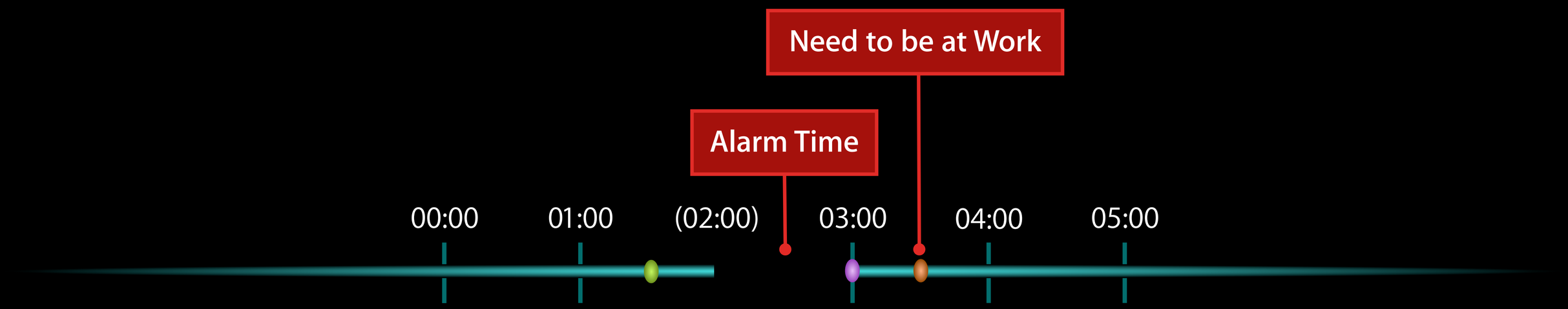
What Is the Best Option?



- NSCalendarMatchNextTime
- NSCalendarMatchNextTimePreservingSmallerUnits
- NSCalendarMatchPreviousTimePreservingSmallerUnits

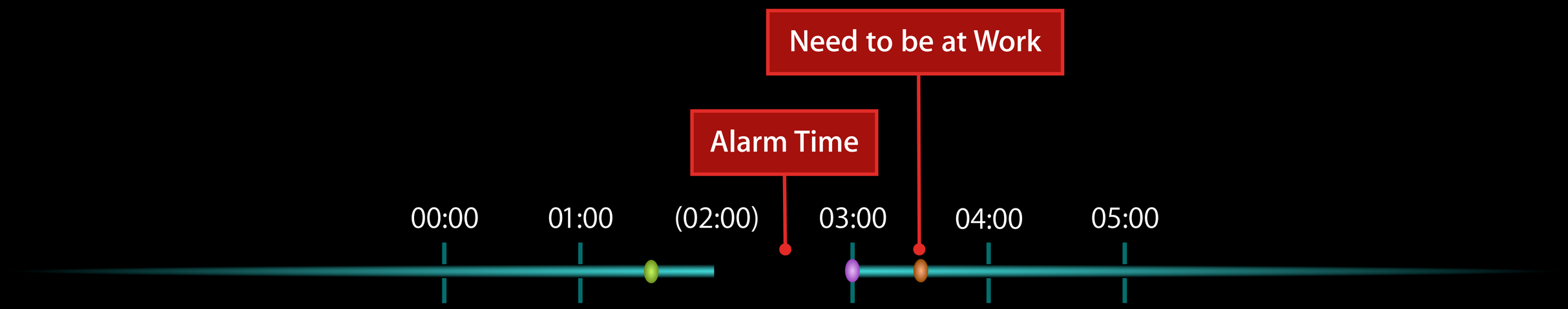
NSCalendarMatchStrictly – alarm next day!

What Is the Best Option?



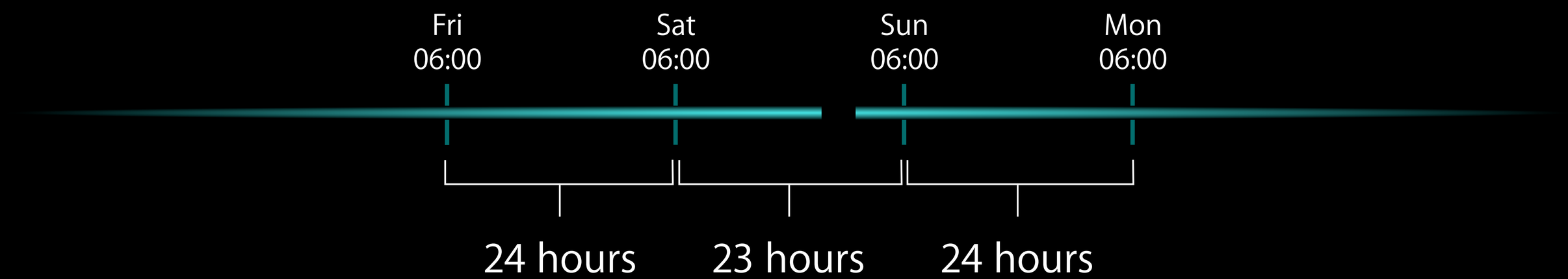
What my friend really wants to specify isn't 2:30

What Is the Best Option?



What my friend really wants to specify isn't 2:30
but 1 hour before 3:30.

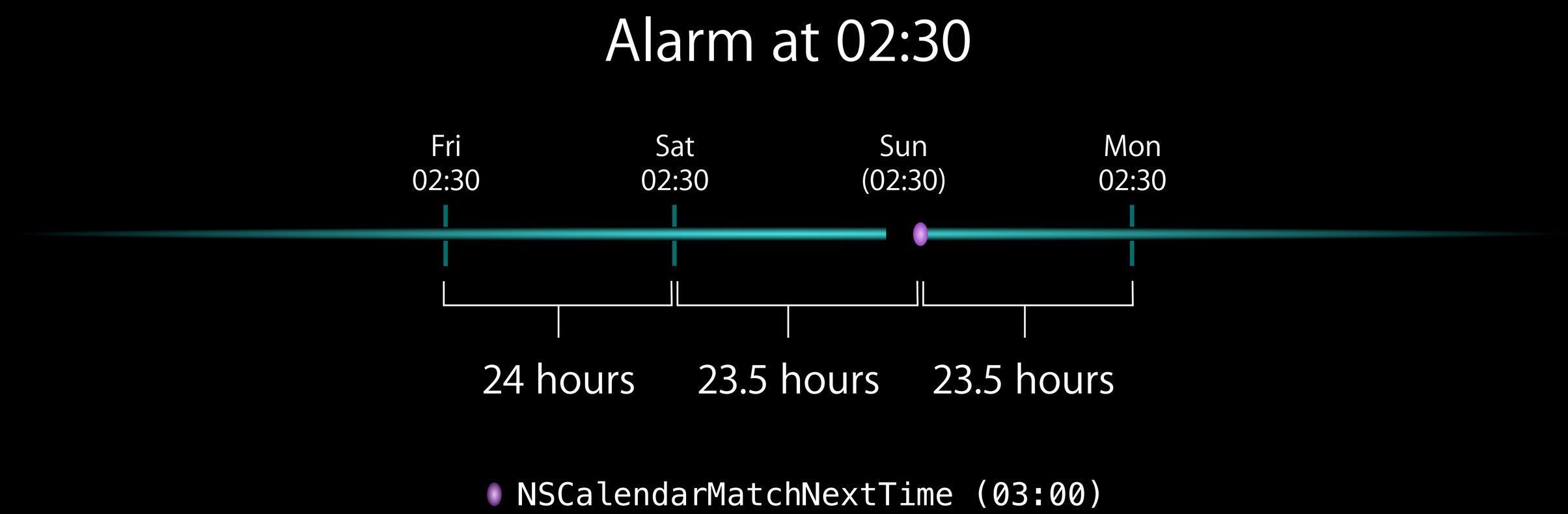
What Is the Best Option? Again



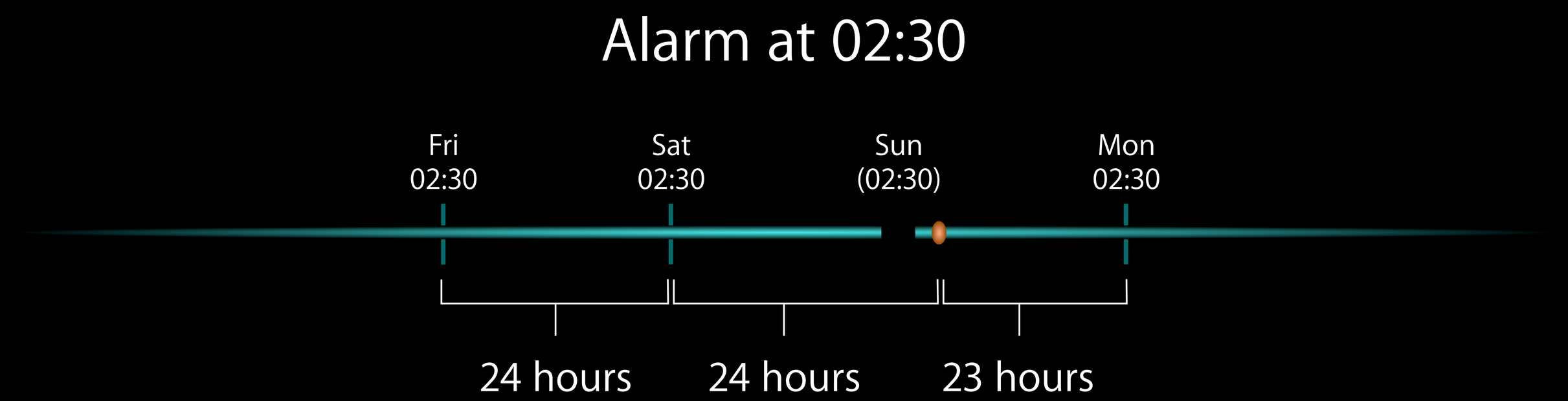
What Is the Best Option? Again

Alarm at 02:30

What Is the Best Option? Again

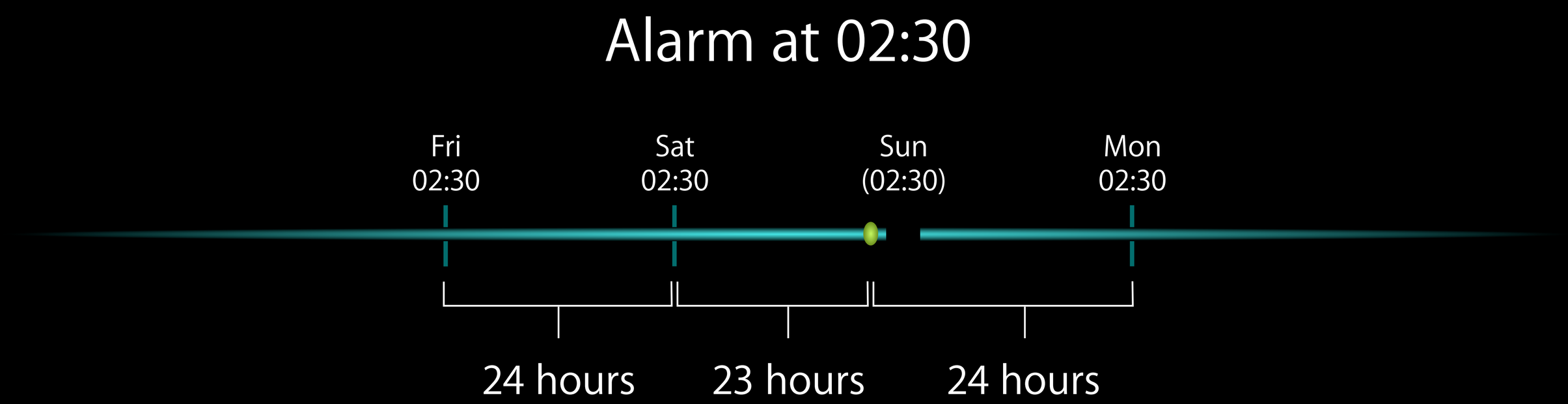


What Is the Best Option? Again



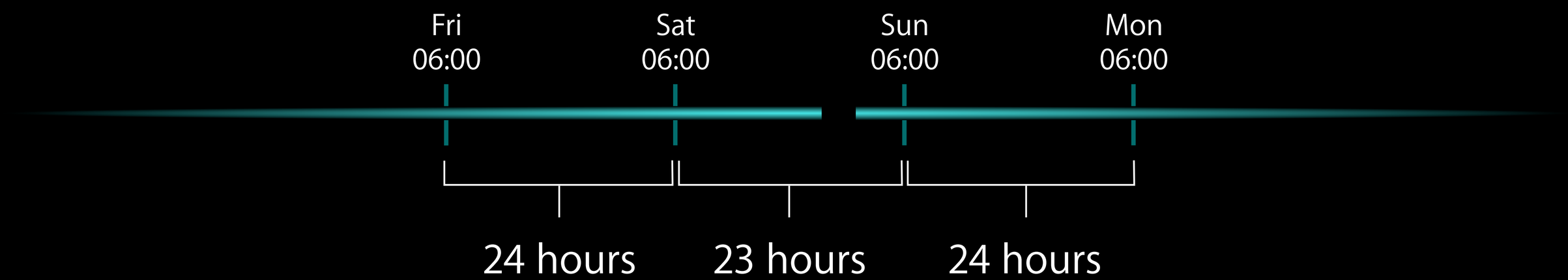
- `NSCalendarMatchNextTimePreservingSmallerUnits (03:30)`

What Is the Best Option? Again

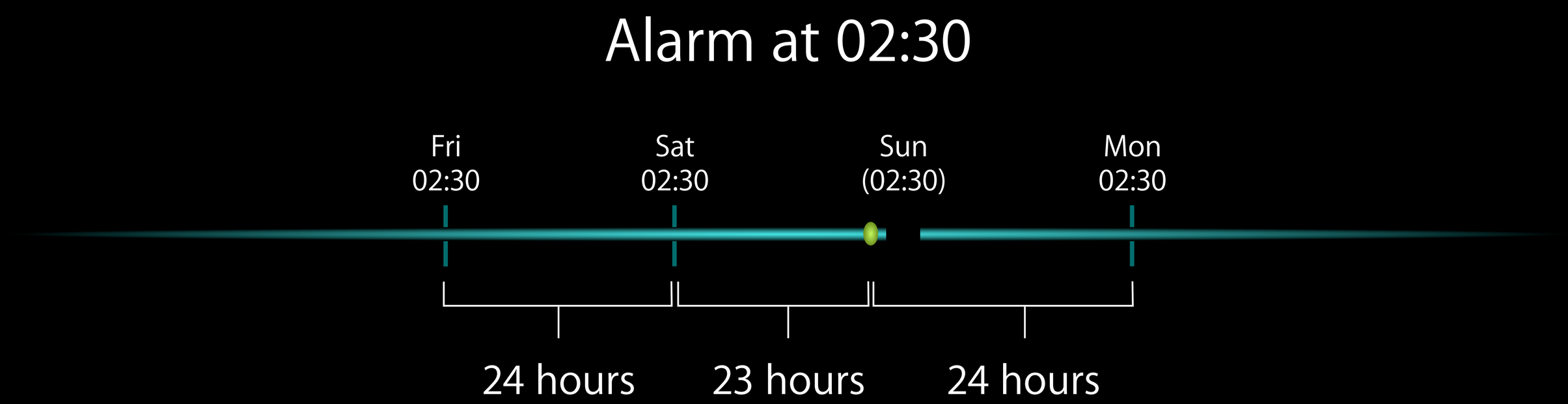


- `NSCalendarMatchPreviousTimePreservingSmallerUnits (01:30)`

What Is the Best Option? Again



What Is the Best Option? Again



- `NSCalendarMatchPreviousTimePreservingSmallerUnits (01:30)`

Enumerating Matches

Enumerating Matches

- If you are going to enumerate matches, use this method

Enumerating Matches

- If you are going to enumerate matches, use this method

```
[cal enumerateDatesStartingAfterDate:[NSDate date]
```

Enumerating Matches

- If you are going to enumerate matches, use this method

```
[cal enumerateDatesStartingAfterDate:[NSDate date]
          matchingComponents:dateComponents
                     options:<options>
```

Enumerating Matches

- If you are going to enumerate matches, use this method

```
[cal enumerateDatesStartingAfterDate:[NSDate date]
            matchingComponents:dateComponents
                       options:<options>
                    usingBlock:^(NSDate *date, BOOL exactMatch, BOOL *stop) {
    // your code here



    }];
```

Enumerating Matches

- If you are going to enumerate matches, use this method

```
[cal enumerateDatesStartingAfterDate:[NSDate date]
            matchingComponents:dateComponents
                       options:<options>
                    usingBlock:^(NSDate *date, BOOL exactMatch, BOOL *stop) {
    // your code here

        if (condition) *stop = YES;
    }];
```

Testing

Testing

Testing

- Interesting cases

Testing

- Interesting cases
  - Different locales and calendars

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries
    - Start/end of day, month, year (and era in some calendars)

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries
    - Start/end of day, month, year (and era in some calendars)
- What do you look for?

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries
    - Start/end of day, month, year (and era in some calendars)
- What do you look for?
  - How do you know what you see is correct?

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries
    - Start/end of day, month, year (and era in some calendars)
- What do you look for?
  - How do you know what you see is correct?
  - Find authoritative information

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries
    - Start/end of day, month, year (and era in some calendars)
- What do you look for?
  - How do you know what you see is correct?
  - Find authoritative information
    - Almanacs, books, the internet

Testing

- Interesting cases
  - Different locales and calendars
  - Different time zones
  - Cycle boundaries
    - Start/end of day, month, year (and era in some calendars)
- What do you look for?
  - How do you know what you see is correct?
  - Find authoritative information
    - Almanacs, books, the internet
    - People

Testing

Testing

- How do you test?

Testing

- How do you test?
  - Turn off time syncing

Testing

- How do you test?
  - Turn off time syncing
  - Set clock, time zone, calendar, and locale manually

Testing

- How do you test?
  - Turn off time syncing
  - Set clock, time zone, calendar, and locale manually
- Develop variety in your beta tester pool

Testing

- How do you test?
  - Turn off time syncing
  - Set clock, time zone, calendar, and locale manually
- Develop variety in your beta tester pool
  - People from around the world

Testing

- How do you test?
  - Turn off time syncing
  - Set clock, time zone, calendar, and locale manually
- Develop variety in your beta tester pool
  - People from around the world
  - Direct their testing

More Information

Paul Marcos

Application Services Evangelist
pmarcos@apple.com

Documentation

Date & Time Programming Guide
http://developer.apple.com/documentation/Cocoa/Conceptual/DatesAndTimes/
Reference documentation for the NSCalendar class
http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSCalendar_Class/

Apple Developer Forums

http://devforums.apple.com

WWDC2013