

Events and Reminders in Event Kit

Session 304

Jeffrey Harris, Matt Lanter
Interpersonal Apps

Aaron Thompson, Scott Adler
iOS Apps & Frameworks

These are confidential sessions—please refrain from streaming, blogging, or taking pictures

Agenda

Agenda

- Calendar events

Agenda

- Calendar events
- Reminders

Agenda

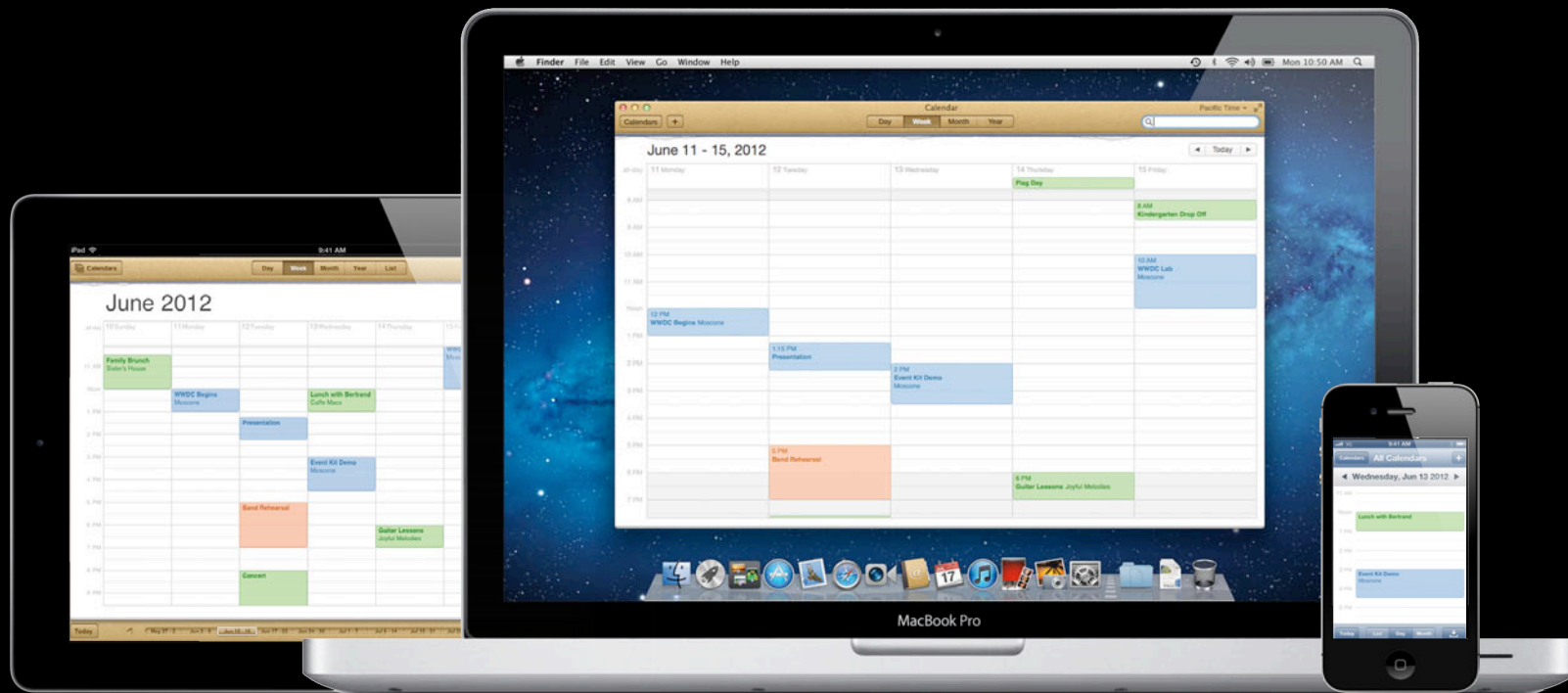
- Calendar events
- Reminders
- Event Kit data isolation

Agenda

- Calendar events
- Reminders
- Event Kit data isolation
- Calendar Store on OS X

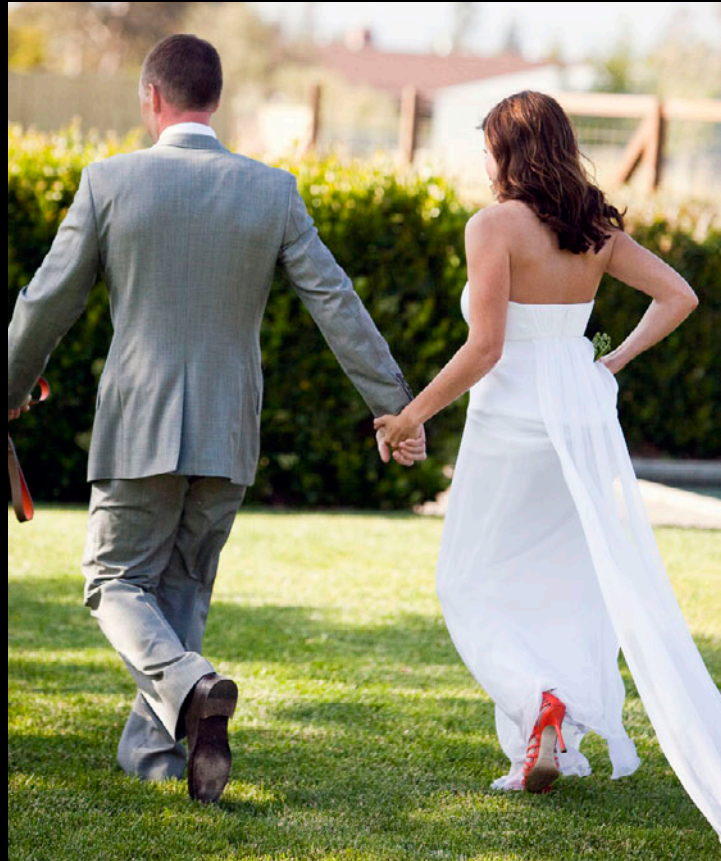
Working with Calendar Events

Calendar Events in Event Kit

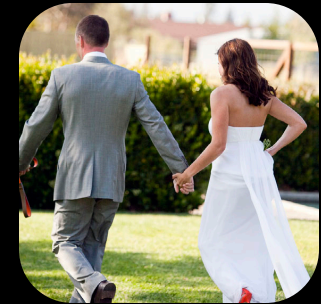


Calendar Events Are Important

Calendar Events Are Important

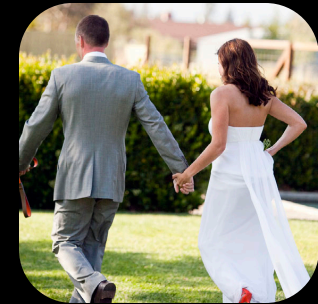
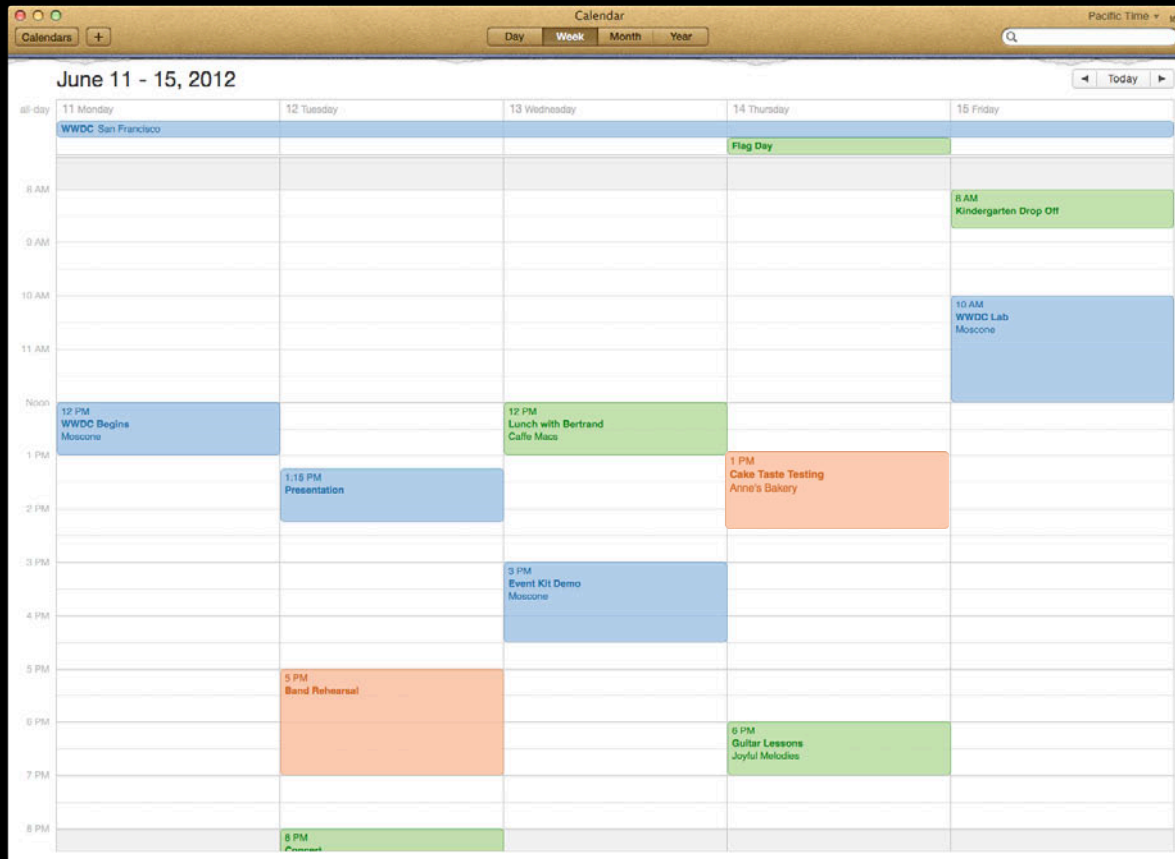


Calendar Events Are Important



AwesomeWedding

Calendar Events Are Important



AwesomeWedding

Calendar Events Are Important



Calendar Events Are Important

- Increase user interaction



Calendar Events Are Important

- Increase user interaction
- Link back to your application

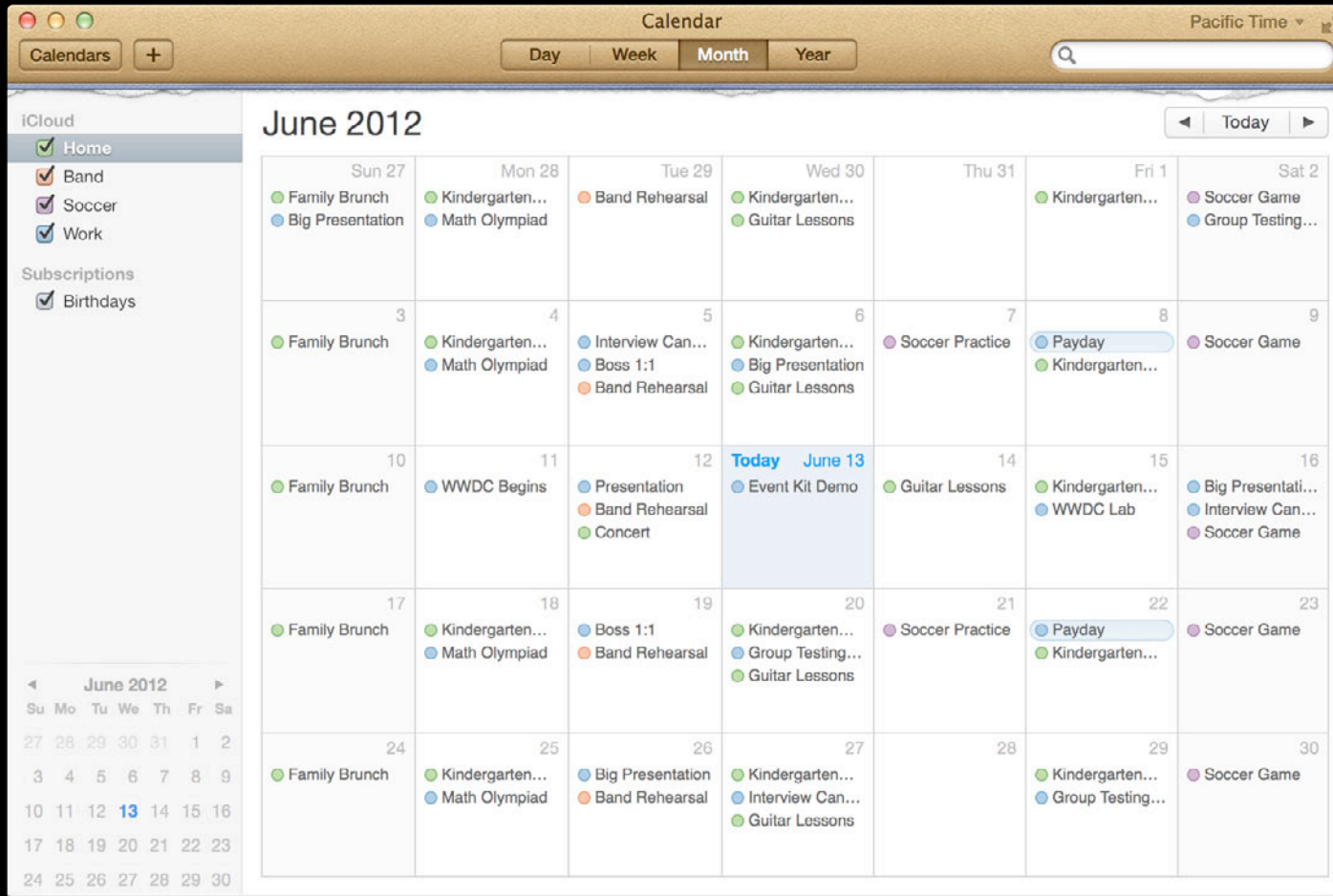


Calendar Events Are Important

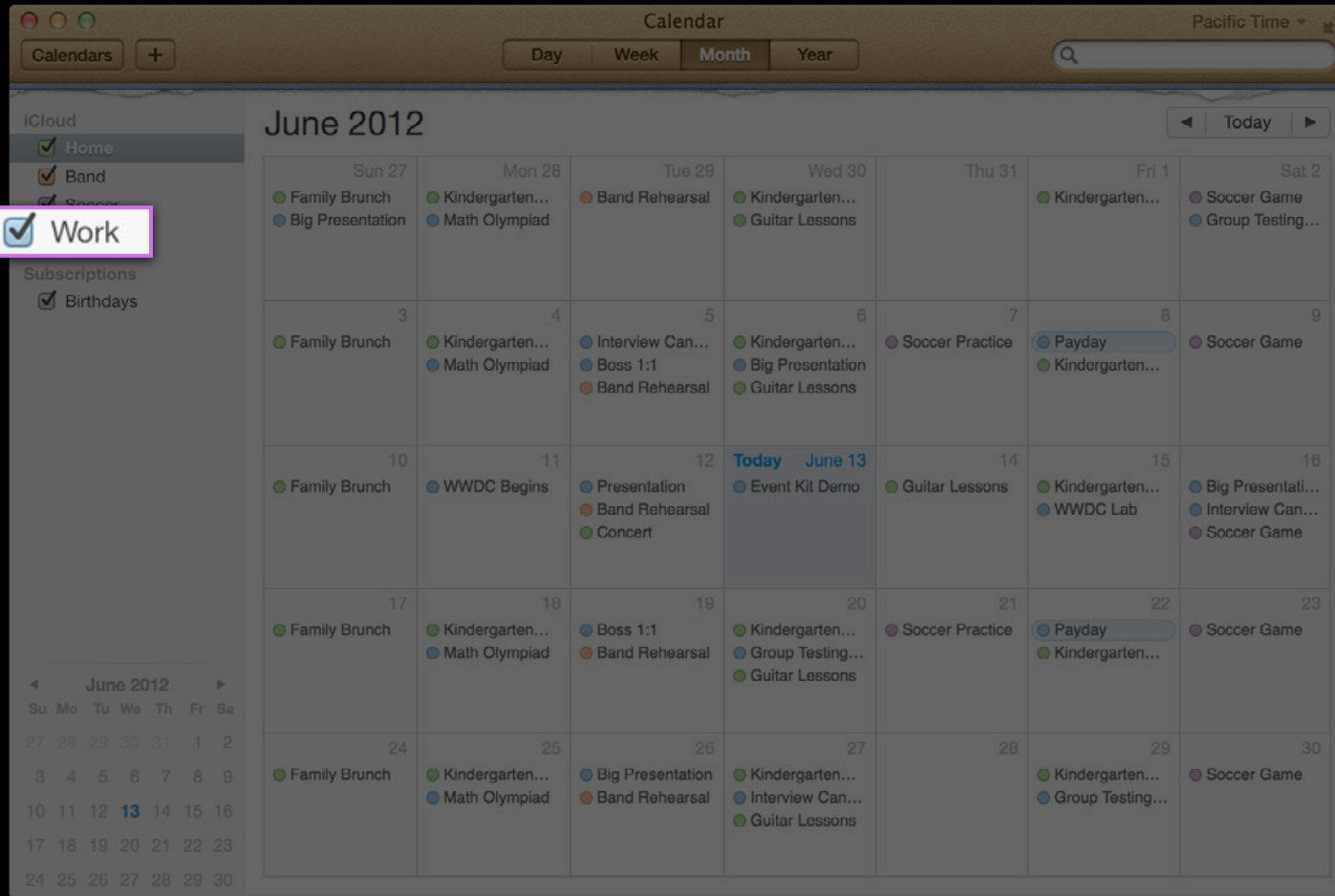
- Increase user interaction
- Link back to your application
- Add alarms



Event Kit Elements



Event Kit Elements



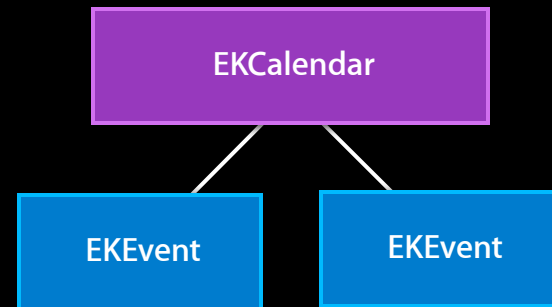
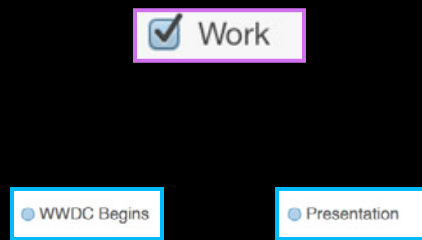
Event Kit Elements

Work

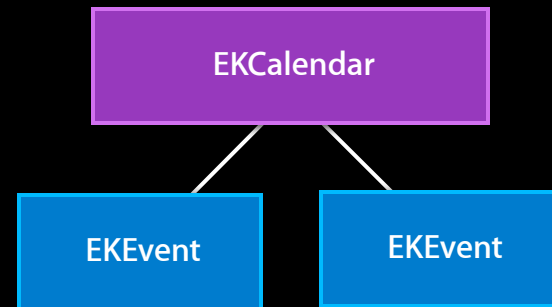
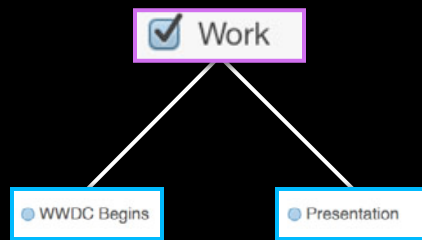
WWDC Begins

Presentation

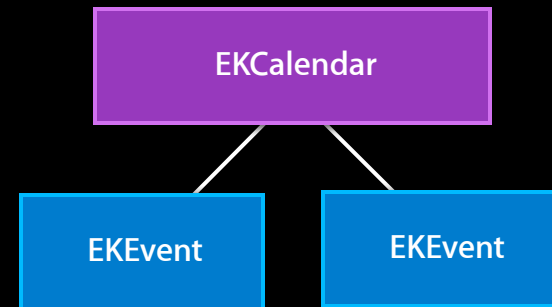
Event Kit Elements



Event Kit Elements



Event Kit Elements



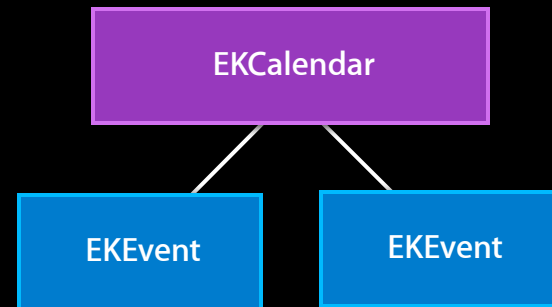
Event Kit Elements



Calendar Events



Reminders



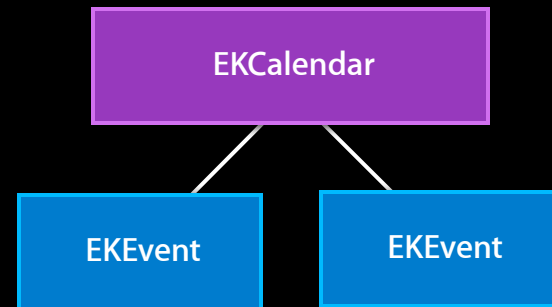
Event Kit Elements



Calendar Events



Reminders



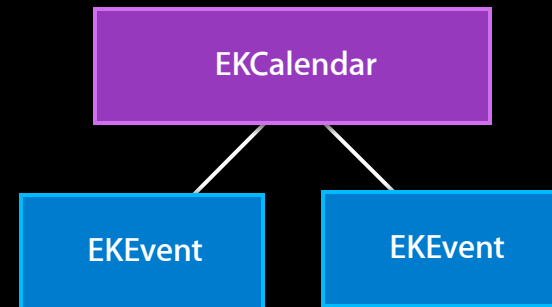
Event Kit Elements



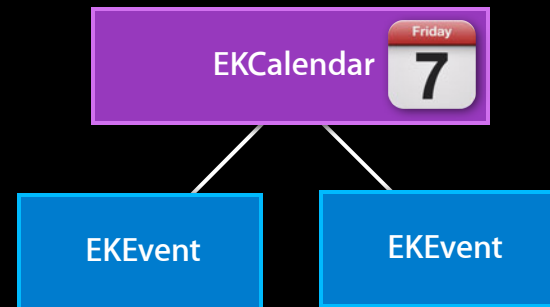
Calendar Events



Reminders

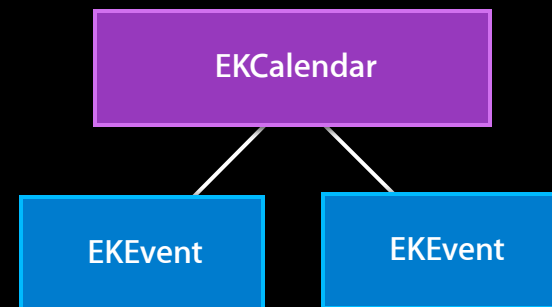


Event Kit Elements



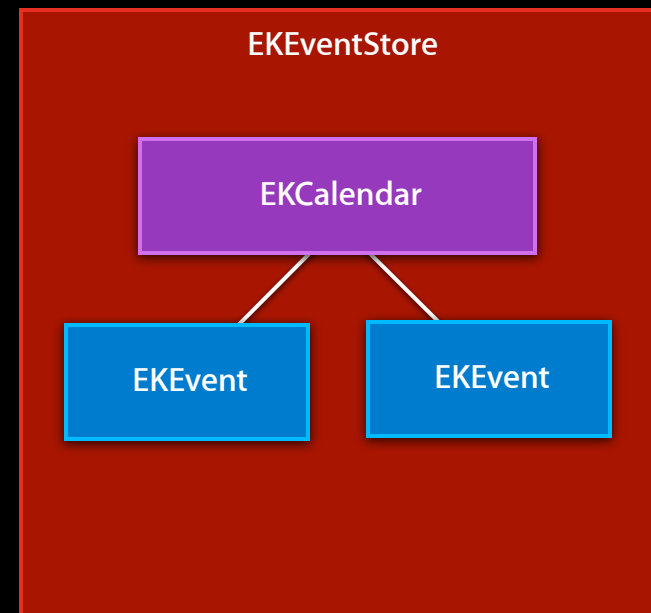
Accessing a User's Calendar Data

EKEventStore



Accessing a User's Calendar Data

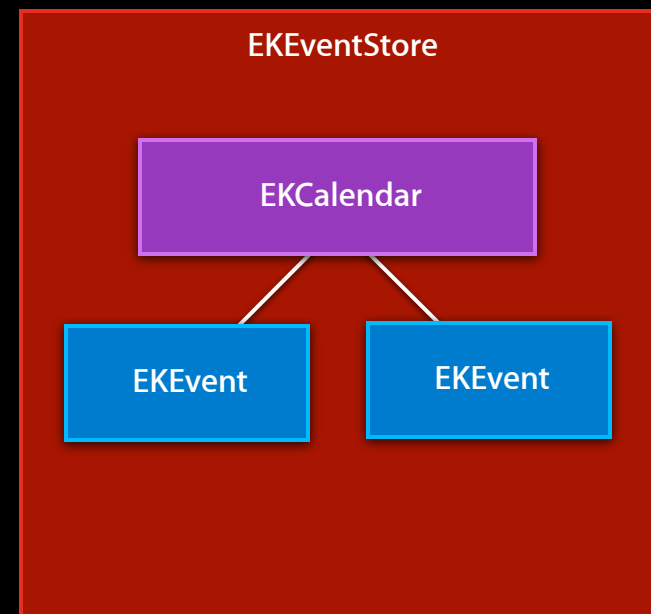
EKEventStore



Accessing a User's Calendar Data

EKEventStore

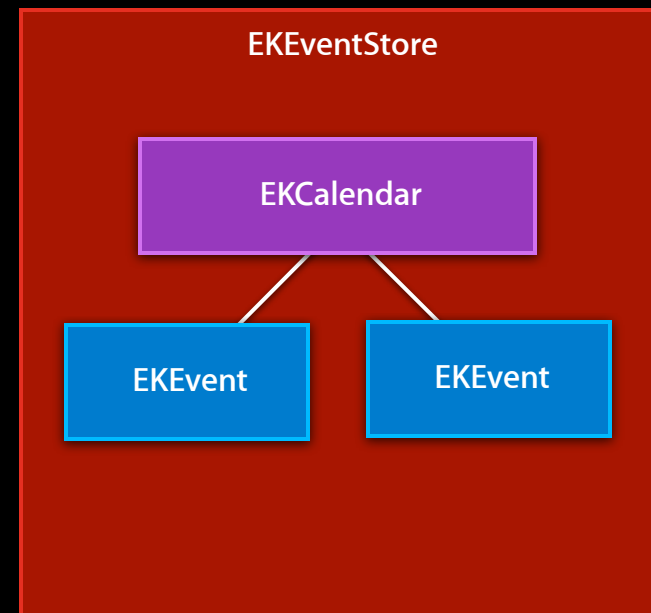
- Connection to calendar data persistence



Accessing a User's Calendar Data

EKEventStore

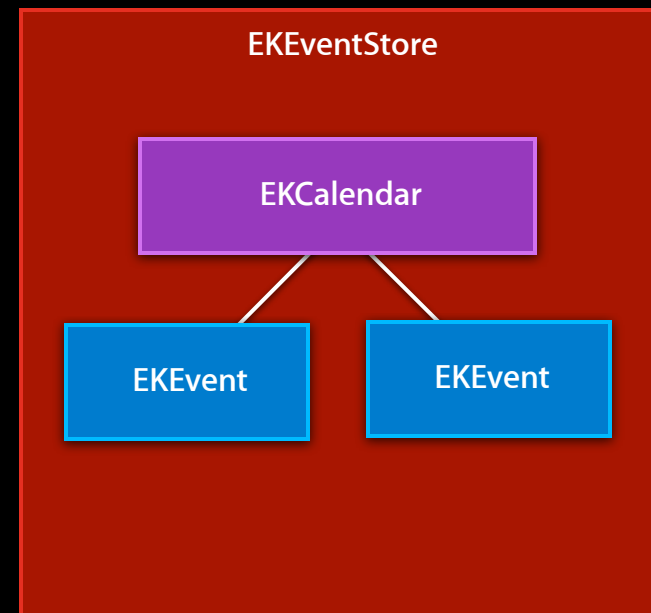
- Connection to calendar data persistence
- Should be long lived



Accessing a User's Calendar Data

EKEventStore

- Connection to calendar data persistence
- Should be long lived
- Specify events or reminders



Accessing a User's Calendar Data

EKEventStore

- Connection to calendar data persistence
- Should be long lived
- Specify events or reminders



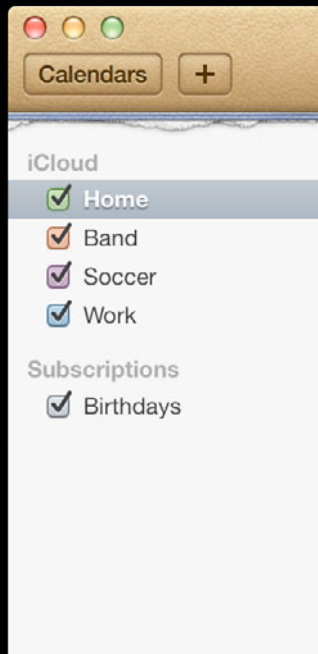
EKCalendar

EKCalendar

Calendars

EKCalendar

OS X



EKCalendar

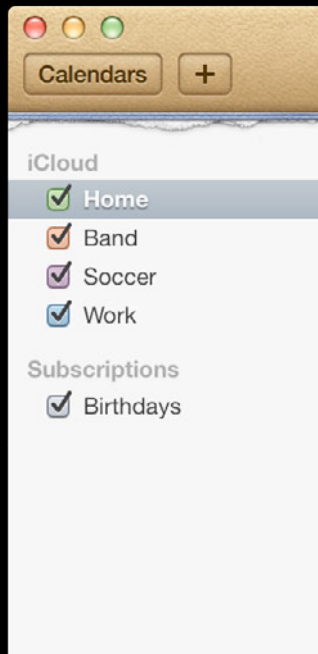
iOS



Calendars

EKCalendar

OS X



EKCalendar

source
title
color
allowedEntityTypes

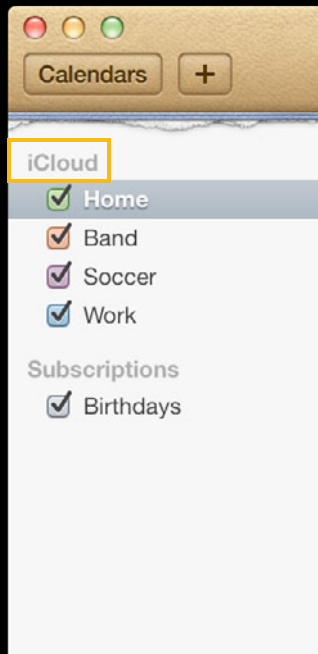
iOS



Calendars

EKCalendar

OS X



EKCalendar

```
source
title
color
allowedEntityTypes
```

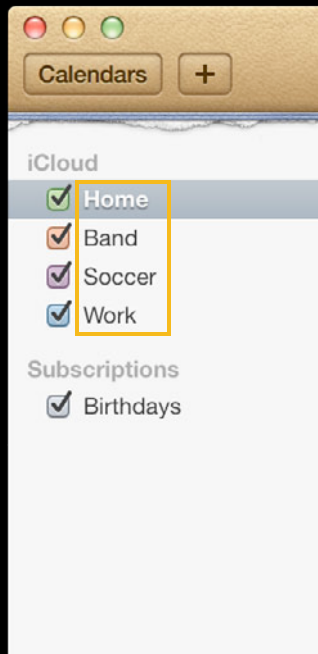
iOS



Calendars

EKCalendar

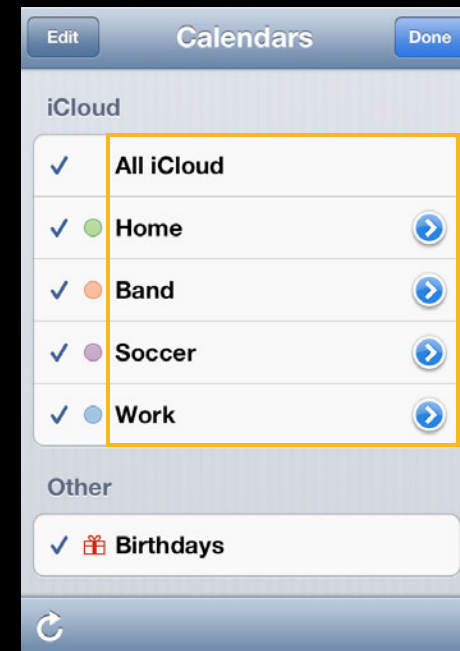
OS X



EKCalendar

```
source  
title  
color  
allowedEntityTypes
```

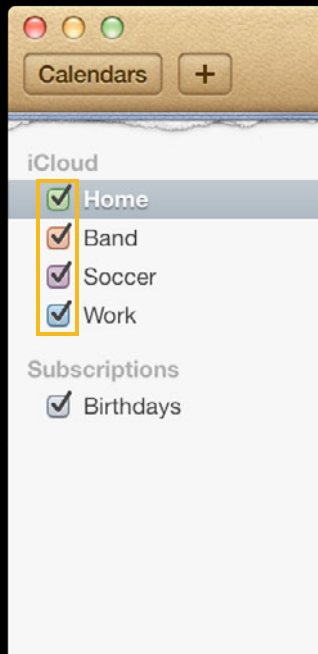
iOS



Calendars

EKCalendar

OS X



EKCalendar

```
source  
title  
color  
allowedEntityTypes
```

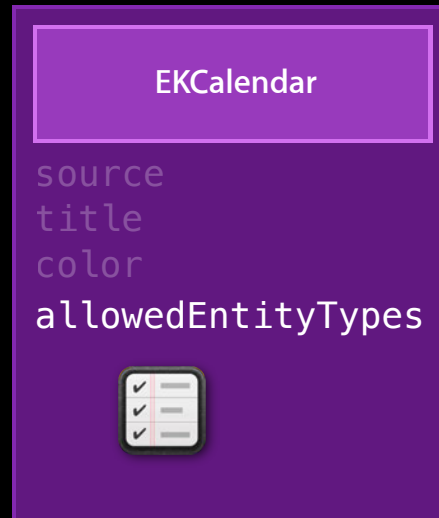
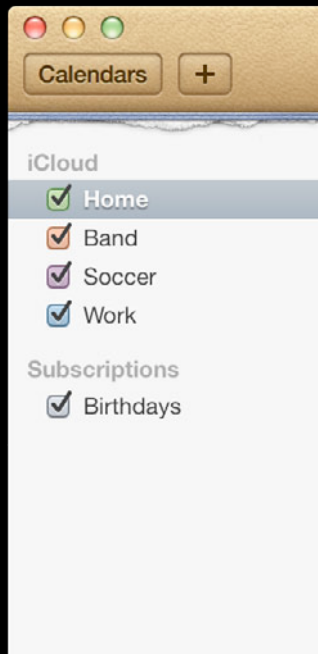
iOS



Calendars

EKCalendar

OS X



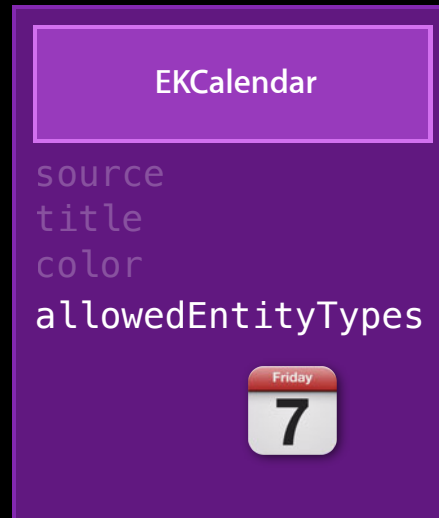
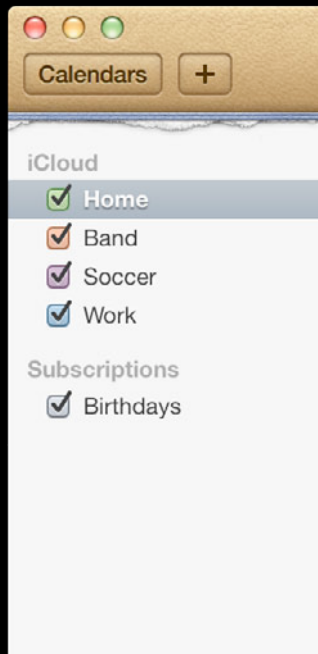
iOS



Calendars

EKCalendar

OS X



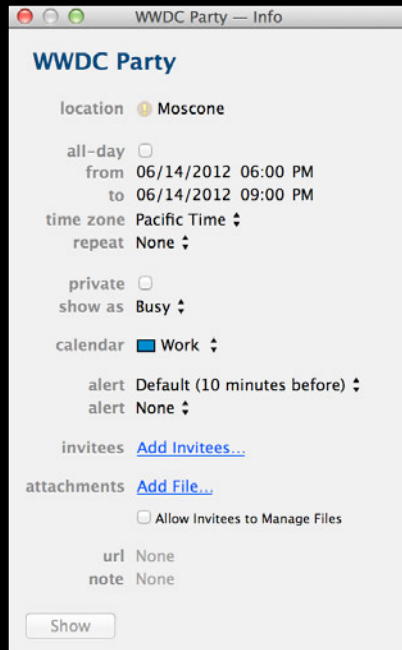
iOS



Calendar Events

EKEvent

OS X



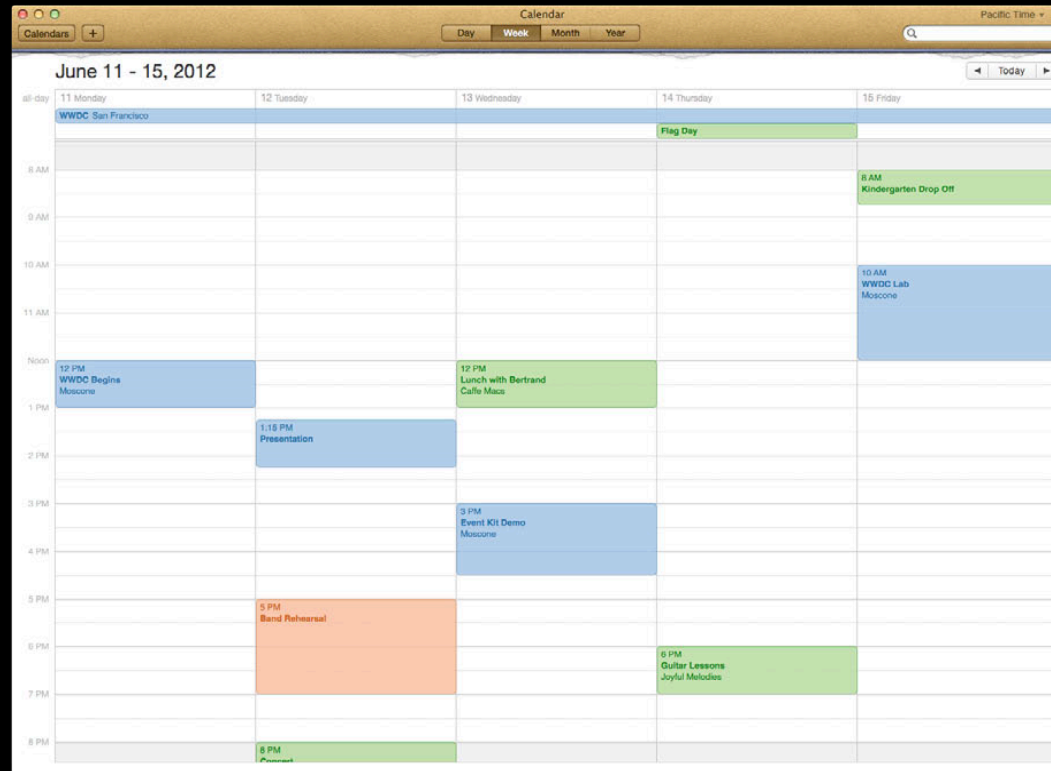
EKEvent

title
location
allDay
startDate
endDate
timeZone
calendar
alarms

iOS



Fetching Events for a Week



Fetching Events for a Week

```
// Fetch and print all Events this week
EKEventStore *store = [[EKEventStore alloc]
                       initWithAccessToEntityTypes:EKEntityTypeMaskEvent];
```

Fetching Events for a Week

```
// Fetch and print all Events this week
EKEventStore *store = [[EKEventStore alloc]
                       initWithAccessToEntityTypes:EKEntityTypeMaskEvent];
```

Fetching Events for a Week

```
// Fetch and print all Events this week
EKEventStore *store = [[EKEventStore alloc]
                       initWithAccessToEntityTypes:EKEntityTypeMaskEvent];
NSArray *eventCalendars = [store calendarsForEntityType:EKEntityTypeEvent];
```

Fetching Events for a Week

```
// Fetch and print all Events this week
EKEventStore *store = [[EKEventStore alloc]
                       initWithAccessToEntityTypes:EKEntityTypeMaskEvent];
NSArray *eventCalendars = [store calendarsForEntityType:EKEntityTypeEvent];
```


Fetching Events for a Week

```
// Fetch and print all Events this week
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskEvent];
NSArray *eventCalendars = [store calendarsForEntityType:EKEntityTypeEvent];
NSDate *monday, *sunday;
... // determine monday and sunday for the current week
NSPredicate *predicate;
predicate = [store predicateForEventsWithStartDate:monday
                                     endDate:sunday
                                     calendars:eventCalendars];
NSArray *events = [store eventsMatchingPredicate:predicate];
NSLog(@"Events this week:");
for (EKEvent *event in events) {
    NSLog(@"%@ - starts at %@", event.title, event.startDate);
}
```

Fetching Events for a Week

```
// Fetch and print all Events this week
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskEvent];
NSArray *eventCalendars = [store calendarsForEntityType:EKEntityTypeEvent];
NSDate *monday, *sunday;
... // determine monday and sunday for the current week
NSPredicate *predicate;
predicate = [store predicateForEventsWithStartDate:monday
                                                endDate:sunday
                                                calendars:eventCalendars];

NSArray *events = [store eventsMatchingPredicate:predicate];
NSLog(@"Events this week:");
for (EKEvent *event in events) {
    NSLog(@"%@ - starts at %@", event.title, event.startDate);
}
```

Demo

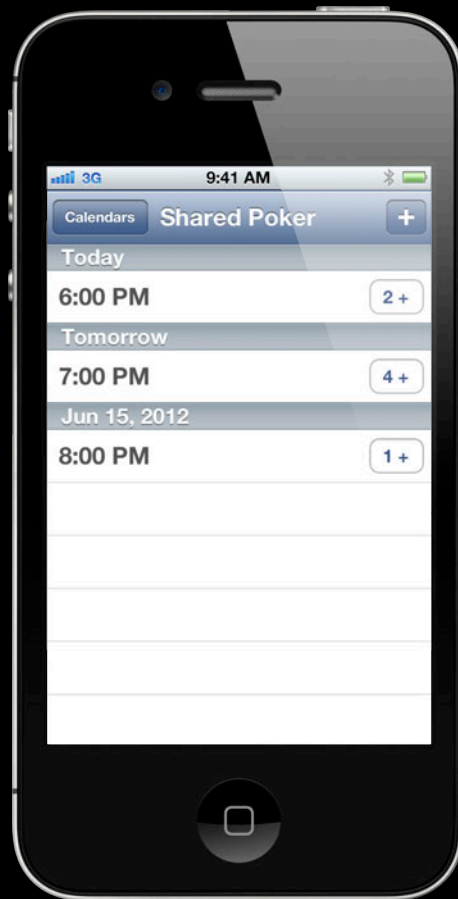
Poker night voting

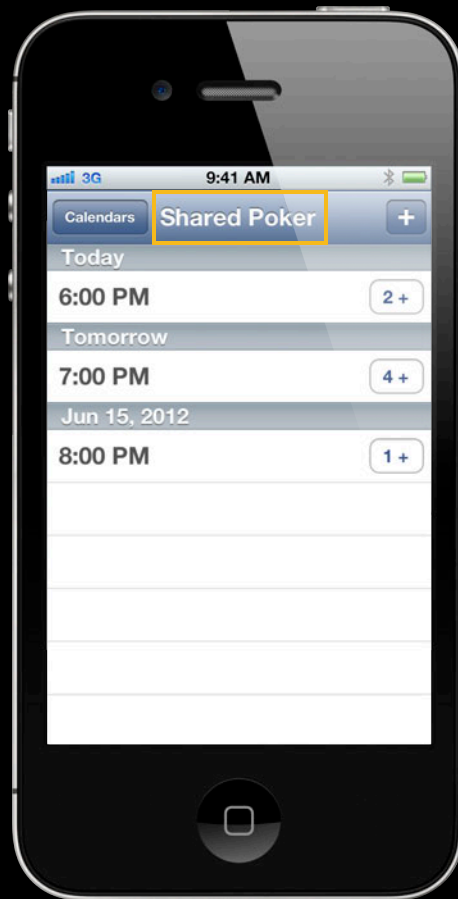
Matt Lanter

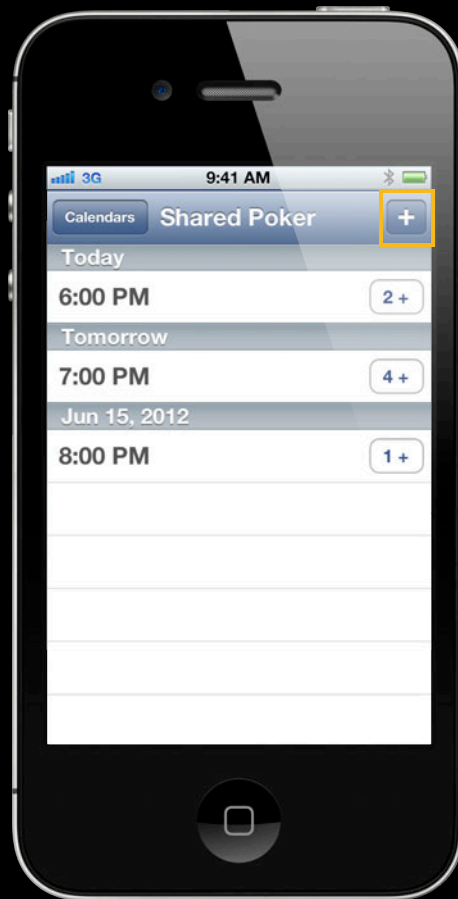
Interpersonal Apps

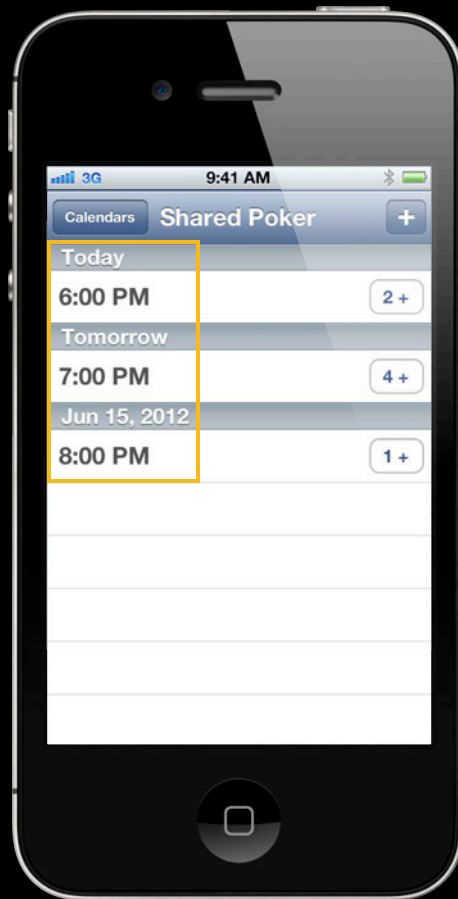


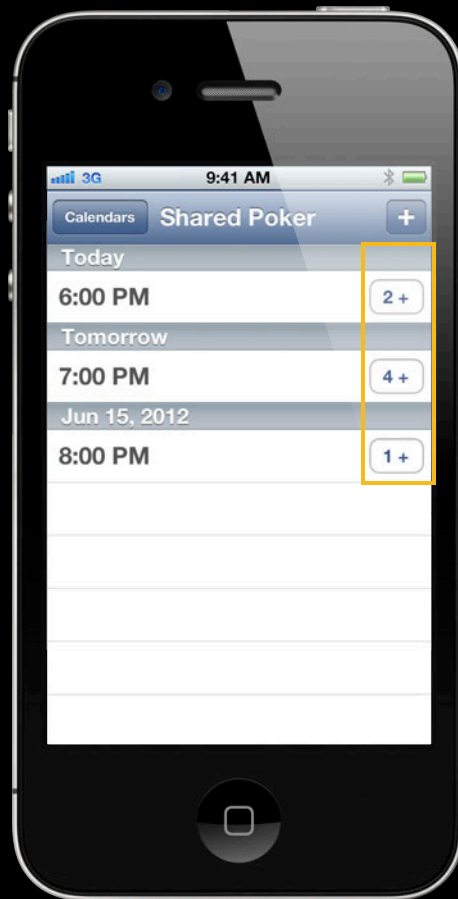












Time	Count
Today	
6:00 PM	2+
Tomorrow	
7:00 PM	4+
Jun 15, 2012	
8:00 PM	1+

Poker Night Voting

Calendar: Shared Poker

Time: 6/13/2012 4:00 PM Add

Time	Votes
Today 6:00 PM	2 +
Tomorrow 7:00 PM	4 +
Jun 15, 2012 8:00 PM	1 +

Model

Poker Night Voting

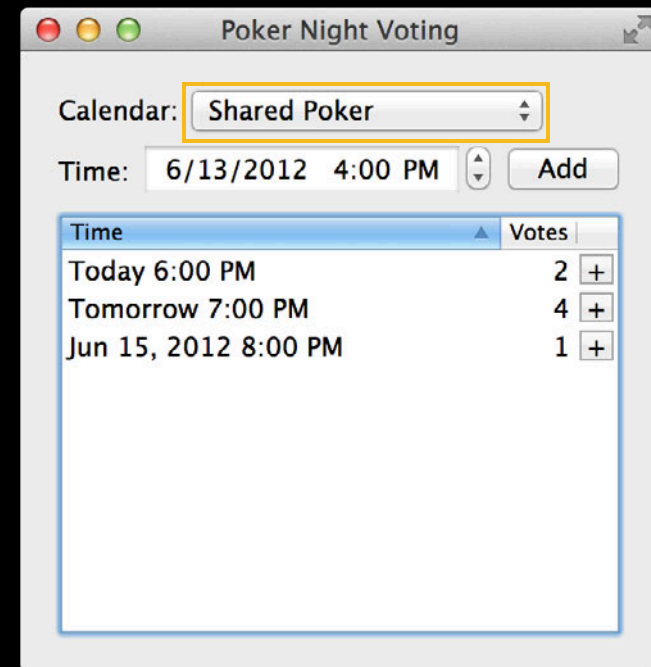
Calendar: Shared Poker

Time: 6/13/2012 4:00 PM Add

Time	Votes
Today 6:00 PM	2 +
Tomorrow 7:00 PM	4 +
Jun 15, 2012 8:00 PM	1 +

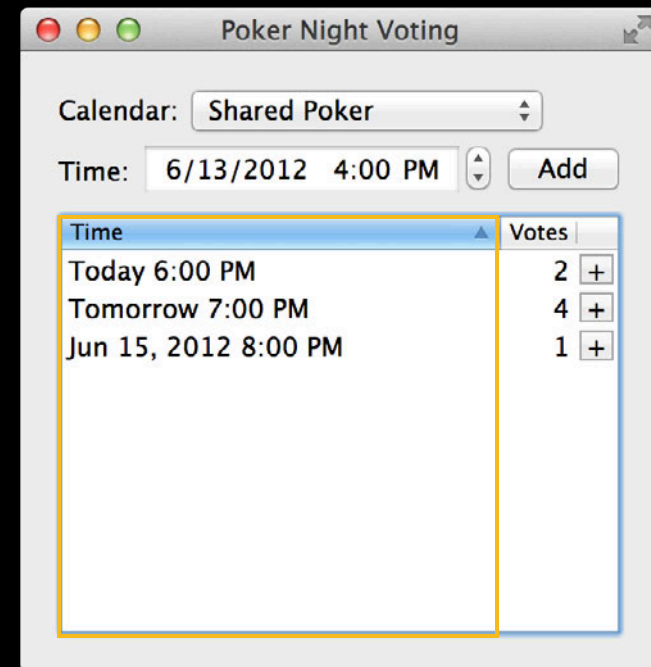
Model

- Fetch calendars



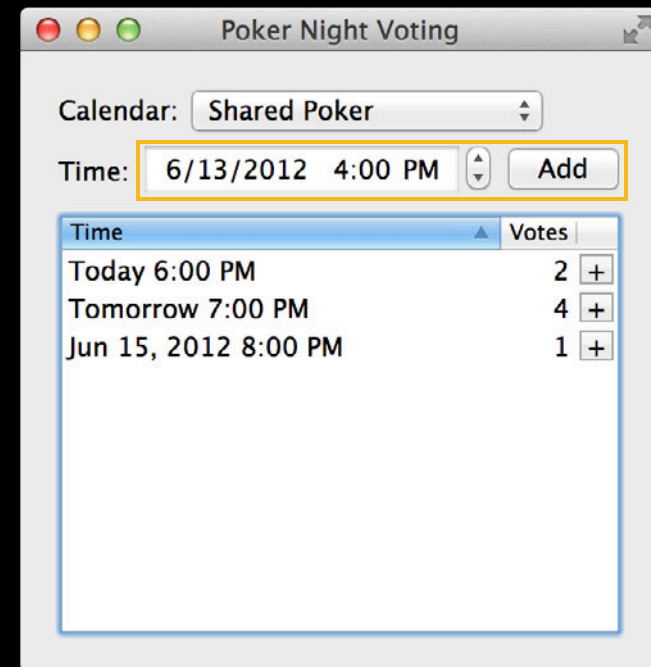
Model

- Fetch calendars
- Fetch calendar events



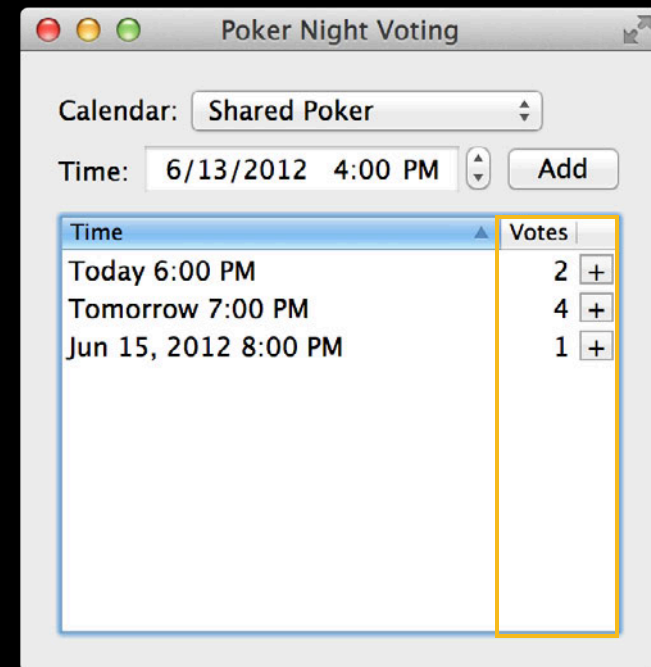
Model

- Fetch calendars
- Fetch calendar events
- Create an event



Model

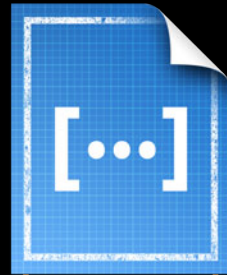
- Fetch calendars
- Fetch calendar events
- Create an event
- Vote on an event



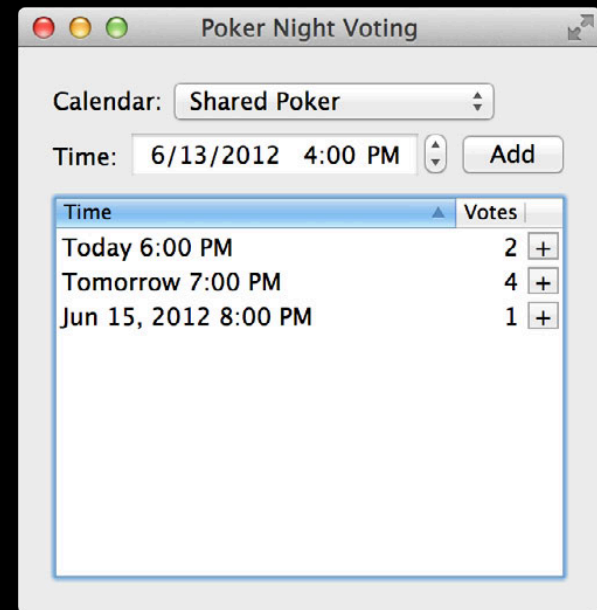
Demo

Poker night voting

Matt Lanter
Interpersonal Apps



Model



Reminders in Event Kit

Aaron Thompson
Gregorian Mixologist

Reminders Everywhere





“Siri, Remind Me to Give My Cat a Bath.”



"Siri, Remind Me to Give My Cat a Bath."



“Siri, Remind Me to Give My Cat a Bath.”



“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]  
                        initWithAccessToEntityTypes:EKEntityMaskReminder];
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityMaskReminder];
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
reminder.title = @"Give my cat a bath"
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
reminder.title = @"Give my cat a bath"
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
reminder.title = @"Give my cat a bath"
reminder.calendar = [store defaultCalendarForNewReminders]
```


“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
reminder.title = @"Give my cat a bath"
reminder.calendar = [store defaultCalendarForNewReminders]
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
reminder.title = @"Give my cat a bath"
reminder.calendar = [store defaultCalendarForNewReminders]
NSError *err = nil;
[store saveReminder:reminder commit:YES error:&err];
```

“Siri, Remind Me to Give My Cat a Bath.”

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeMaskReminder];
EKReminder *reminder = [EKReminder reminderWithEventStore:store];
reminder.title = @"Give my cat a bath"
reminder.calendar = [store defaultCalendarForNewReminders]
NSError *err = nil;
[store saveReminder:reminder commit:YES error:&err];
```

Time-Based Reminders

**“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”**



“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”

Here's your reminder for
tomorrow at 4 pm:

14 Thursday
June 2012

Give my cat a bath
Tomorrow



**“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”**

```
NSDate *alarmDate = [NSDate date]; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
```

“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”

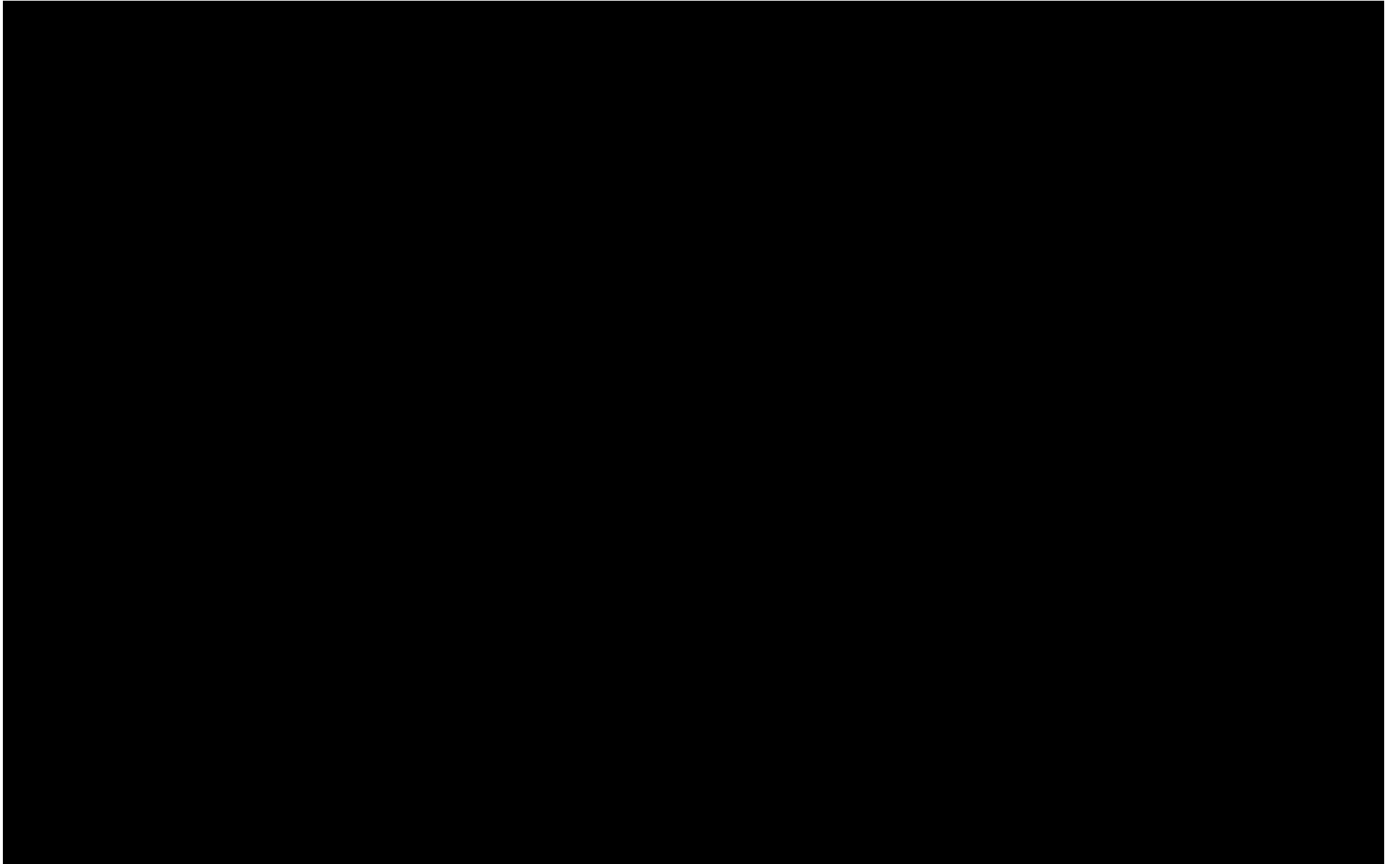
```
NSDate *alarmDate = [NSDate date]; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
```


“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”

```
NSDate *alarmDate = [NSDate date]; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
```

Date Math

"Bad dates." - Raiders of the Lost Ark



86,400

86,400

60 × 60 × 24

seconds

minutes

hours

86400
60 × 24
seconds × minutes × hours





May be **wrong** twice per year!

Tomorrow at 4 pm

```
NSDate *calendar = [NSDate gregorianCalendar];
```


Tomorrow at 4 pm

```
NSDate *calendar = [NSDate gregorianCalendar];
```

Tomorrow at 4 pm

```
NSCalendar *calendar = [NSCalendar gregorianCalendar];
NSDateComponents *oneDayComponents = [[NSDateComponents alloc] init];
oneDayComponents.day = 1;
```

Tomorrow at 4 pm

```
NSCalendar *calendar = [NSCalendar gregorianCalendar];
```

```
NSDateComponents *oneDayComponents = [[NSDateComponents alloc] init];  
oneDayComponents.day = 1;
```


Tomorrow at 4 pm

```
NSCalendar *calendar = [NSCalendar gregorianCalendar];
NSDateComponents *oneDayComponents = [[NSDateComponents alloc] init];
oneDayComponents.day = 1;
NSDate *tomorrow = [calendar dateByAddingComponents:oneDayComponents
                                             toDate:[NSDate date]
                                             options:nil];

NSUInteger unitFlags = NSEraCalendarUnit | NSYearCalendarUnit |
                      NSMonthCalendarUnit | NSDayCalendarUnit;
NSDateComponents *tomorrowAt4PM = [calendar components:unitFlags
                                             fromDate:tomorrow];

tomorrowAt4PM.hour    = 16;
tomorrowAt4PM.minute = 0;
tomorrowAt4PM.second  = 0;
```

Tomorrow at 4 pm

```
NSCalendar *calendar = [NSCalendar gregorianCalendar];
NSDateComponents *oneDayComponents = [[NSDateComponents alloc] init];
oneDayComponents.day = 1;
NSDate *tomorrow = [calendar dateByAddingComponents:oneDayComponents
                                             toDate:[NSDate date]
                                             options:nil];
```

```
NSUInteger unitFlags = NSEraCalendarUnit | NSYearCalendarUnit |
                       NSMonthCalendarUnit | NSDayCalendarUnit;
NSDateComponents *tomorrowAt4PM = [calendar components:unitFlags
                                             fromDate:tomorrow];

tomorrowAt4PM.hour    = 16;
tomorrowAt4PM.minute  = 0;
tomorrowAt4PM.second  = 0;
```

Tomorrow at 4 pm

```
NSCalendar *calendar = [NSCalendar gregorianCalendar];
NSDateComponents *oneDayComponents = [[NSDateComponents alloc] init];
oneDayComponents.day = 1;
NSDate *tomorrow = [calendar dateByAddingComponents:oneDayComponents
                                             toDate:[NSDate date]
                                             options:nil];

NSUInteger unitFlags = NSEraCalendarUnit | NSYearCalendarUnit |
                      NSMonthCalendarUnit | NSDayCalendarUnit;
NSDateComponents *tomorrowAt4PM = [calendar components:unitFlags
                                             fromDate:tomorrow];

tomorrowAt4PM.hour    = 16;
tomorrowAt4PM.minute = 0;
tomorrowAt4PM.second  = 0;
NSDate *alarmDate = [calendar dateFromComponents:tomorrowAt4PM];
```


Tomorrow at 4 pm

```
NSCalendar *calendar = [NSCalendar gregorianCalendar];
NSDateComponents *oneDayComponents = [[NSDateComponents alloc] init];
oneDayComponents.day = 1;
NSDate *tomorrow = [calendar dateByAddingComponents:oneDayComponents
                                             toDate:[NSDate date]
                                             options:nil];

NSUInteger unitFlags = NSEraCalendarUnit | NSYearCalendarUnit |
                      NSMonthCalendarUnit | NSDayCalendarUnit;
NSDateComponents *tomorrowAt4PM = [calendar components:unitFlags
                                             fromDate:tomorrow];

tomorrowAt4PM.hour    = 16;
tomorrowAt4PM.minute = 0;
tomorrowAt4PM.second  = 0;
NSDate *alarmDate = [calendar dateFromComponents:tomorrowAt4PM];
```

Date Math

"Bad dates." - Raiders of the Lost Ark

**“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”**

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
```

“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
```

**“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”**

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];  
[reminder addAlarm:alarm];
```

“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];  
[reminder addAlarm:alarm];
```

**“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”**

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];  
[reminder addAlarm:alarm];  
reminder.dueDateComponents = ...; // Tomorrow at 4 PM
```

“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM  
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];  
[reminder addAlarm:alarm];  
reminder.dueDateComponents = ...; // Tomorrow at 4 PM
```


**“Siri, Remind Me to Give My
Cat a Bath Tomorrow at 4 pm.”**

```
NSDate *alarmDate = ...; // Tomorrow at 4 PM
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
[reminder addAlarm:alarm];
reminder.dueDateComponents = ...; // Tomorrow at 4 PM
[store saveReminder:reminder commit:YES error:&err];
```

“Siri, Remind Me to Give My Cat a Bath Tomorrow at 4 pm.”

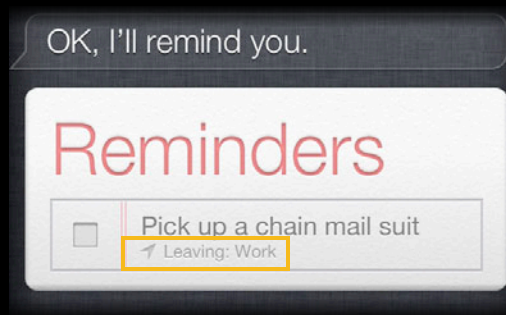
```
NSDate *alarmDate = ...; // Tomorrow at 4 PM
EKAlarm *alarm = [EKAlarm alarmWithAbsoluteDate:alarmDate];
[reminder addAlarm:alarm];
reminder.dueDateComponents = ...; // Tomorrow at 4 PM
[store saveReminder:reminder commit:YES error:&err];
```

Location-Based Reminders

**“Siri, Remind Me to Pick up a
Chain Mail Suit When I Leave Work.”**



“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”



**“Siri, Remind Me to Pick up a
Chain Mail Suit When I Leave Work.”**

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];
```

“Siri, Remind Me to Pick up a
Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];
```


“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;  
  
EKAlarm *alarm = [[EKAlarm alloc] init];
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;
```

```
EKAlarm *alarm = [[EKAlarm alloc] init];
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;  
  
EKAlarm *alarm = [[EKAlarm alloc] init];  
alarm.structuredLocation = location;  
alarm.proximity = EKAlarmProximityLeave;
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;  
  
EKAlarm *alarm = [[EKAlarm alloc] init];  
alarm.structuredLocation = location;  
alarm.proximity = EKAlarmProximityLeave;
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;
location = [EKStructuredLocation locationWithTitle:@"Work"];
CLLocation *geoLocation; // Can obtain from CLGeocoder
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];
location.geoLocation = geoLocation;

EKAlarm *alarm = [[EKAlarm alloc] init];
alarm.structuredLocation = location;
alarm.proximity = EKAlarmProximityLeave;
[reminder addAlarm:alarm];
```


“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;  
  
EKAlarm *alarm = [[EKAlarm alloc] init];  
alarm.structuredLocation = location;  
alarm.proximity = EKAlarmProximityLeave;  
[reminder addAlarm:alarm];
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;
location = [EKStructuredLocation locationWithTitle:@"Work"];
CLLocation *geoLocation; // Can obtain from CLGeocoder
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];
location.geoLocation = geoLocation;

EKAlarm *alarm = [[EKAlarm alloc] init];
alarm.structuredLocation = location;
alarm.proximity = EKAlarmProximityLeave;
[reminder addAlarm:alarm];
[store saveReminder:reminder commit:YES error:&err];
```

“Siri, Remind Me to Pick up a Chain Mail Suit When I Leave Work.”

```
EKStructuredLocation *location;  
location = [EKStructuredLocation locationWithTitle:@"Work"];  
CLLocation *geoLocation; // Can obtain from CLGeocoder  
geoLocation = [[CLLocation alloc] initWithLatitude:37.332 longitude:-122.03];  
location.geoLocation = geoLocation;  
  
EKAlarm *alarm = [[EKAlarm alloc] init];  
alarm.structuredLocation = location;  
alarm.proximity = EKAlarmProximityLeave;  
[reminder addAlarm:alarm];  
[store saveReminder:reminder commit:YES error:&err];
```

Recurring Reminders

**“Siri, Remind Me to Give
My Cat a Bath Every Month.”**



“Siri, Remind Me to Give
My Cat a Bath Every Month.”

OK, I'll start reminding you.

6 Wednesday
June 2012



Give my cat a bath
Tomorrow



“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
    initWithFrequency:EKRecurrenceFrequencyMonthly
    interval:1
    end:nil];
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
    initWithFrequency:EKRecurrenceFrequencyMonthly
    interval:1
    end:nil];
```


“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
                           initWithFrequency:EKRecurrenceFrequencyMonthly
                           interval:1
                           end:nil];
[reminder addRecurrenceRule:rule];
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
    initWithFrequency:EKRecurrenceFrequencyMonthly
    interval:1
    end:nil];
[reminder addRecurrenceRule:rule];
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
                           initWithFrequency:EKRecurrenceFrequencyMonthly
                           interval:1
                           end:nil];

[reminder addRecurrenceRule:rule];
NSDateComponents *componentsForNow = ...; // Current time
reminder.startDateComponents = componentsForNow;
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
                           initWithFrequency:EKRecurrenceFrequencyMonthly
                           interval:1
                           end:nil];

[reminder addRecurrenceRule:rule];
NSDateComponents *componentsForNow = ...; // Current time
reminder.startDateComponents = componentsForNow;
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
                           initWithFrequency:EKRecurrenceFrequencyMonthly
                           interval:1
                           end:nil];

[reminder addRecurrenceRule:rule];
NSDateComponents *componentsForNow = ...; // Current time
reminder.startDateComponents = componentsForNow;
NSError *error = nil;
[store saveReminder:reminder commit:YES error:&error];
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
                           initWithFrequency:EKRecurrenceFrequencyMonthly
                           interval:1
                           end:nil];

[reminder addRecurrenceRule:rule];
NSDateComponents *componentsForNow = ...; // Current time
reminder.startDateComponents = componentsForNow;
NSError *error = nil;
[store saveReminder:reminder commit:YES error:&error];
```

“Siri, Remind Me to Give My Cat a Bath Every Month.”

```
EKRecurrenceRule *rule = [[EKRecurrenceRule alloc]
                           initWithFrequency:EKRecurrenceFrequencyMonthly
                           interval:1
                           end:nil];

[reminder addRecurrenceRule:rule];
NSDateComponents *componentsForNow = ...; // Current time
reminder.startDateComponents = componentsForNow;
NSError *error = nil;
[store saveReminder:reminder commit:YES error:&error];
```

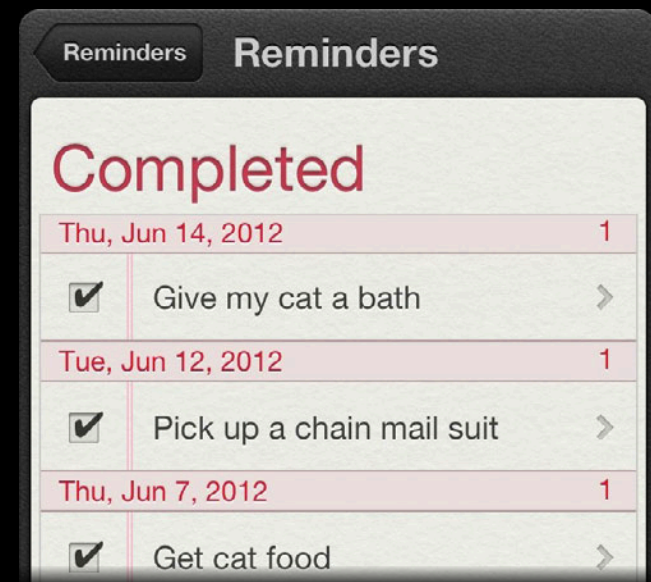
When one is completed, next is generated

Marking Reminders Complete

Marking Reminders Complete

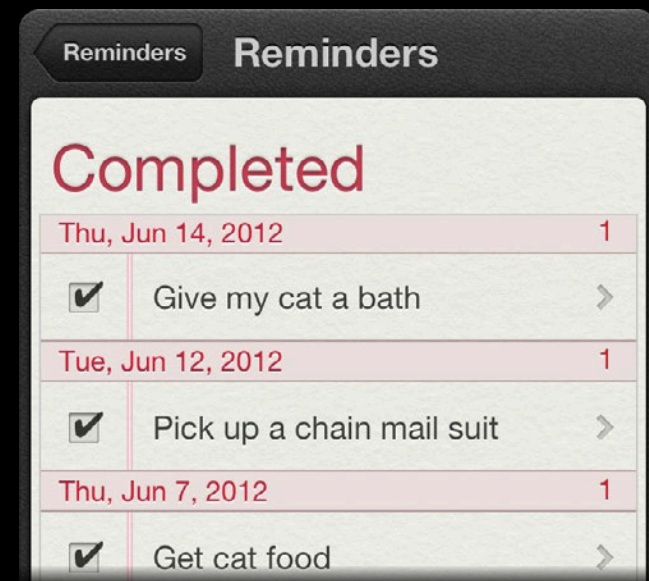


Marking Reminders Complete



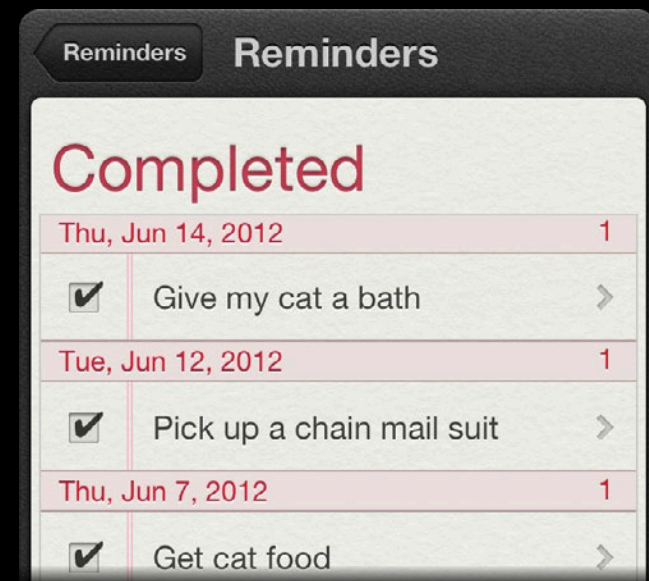
Marking Reminders Complete

```
reminder.completed = YES;  
NSError *error = nil;  
[store saveReminder:reminder  
      commit:YES  
      error:&error];
```



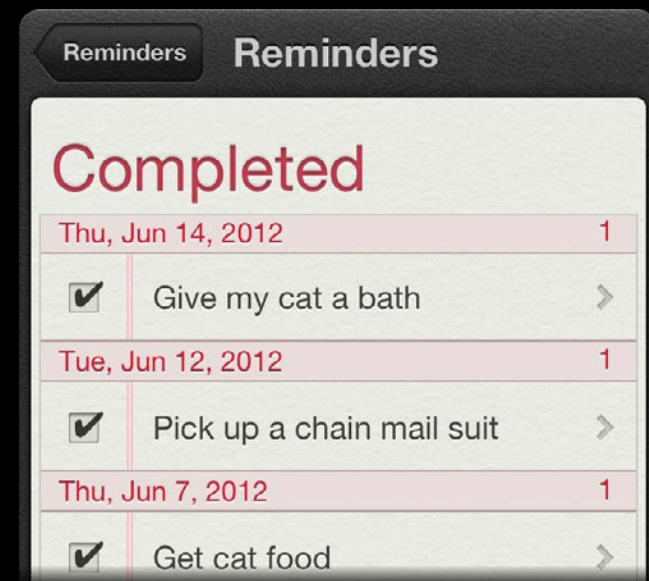
Marking Reminders Complete

```
reminder.completed = YES;  
NSError *error = nil;  
[store saveReminder:reminder  
      commit:YES  
      error:&error];
```



Marking Reminders Complete

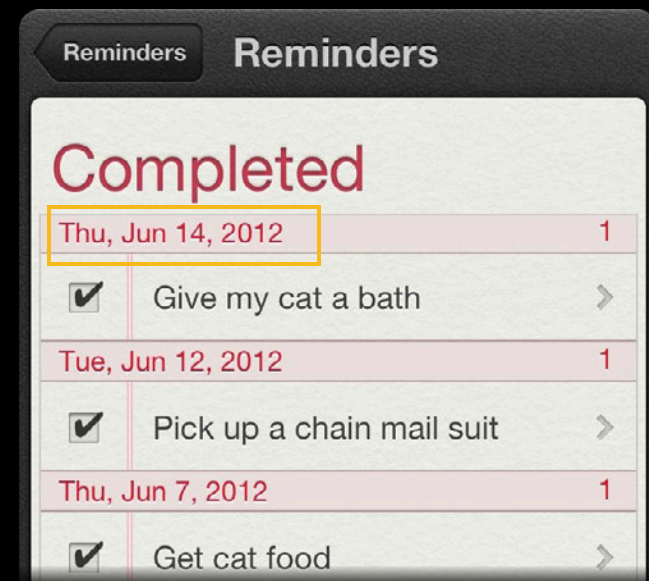
```
reminder.completed = YES;  
NSError *error = nil;  
[store saveReminder:reminder  
      commit:YES  
      error:&error];
```



Setting a Completion Date



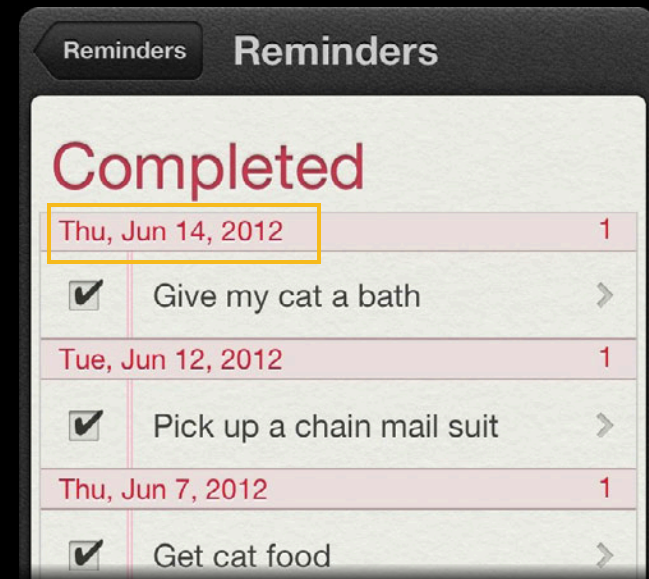
Setting a Completion Date



Setting a Completion Date

- Automatically set to the current time when you set

```
reminder.completed = YES;
```



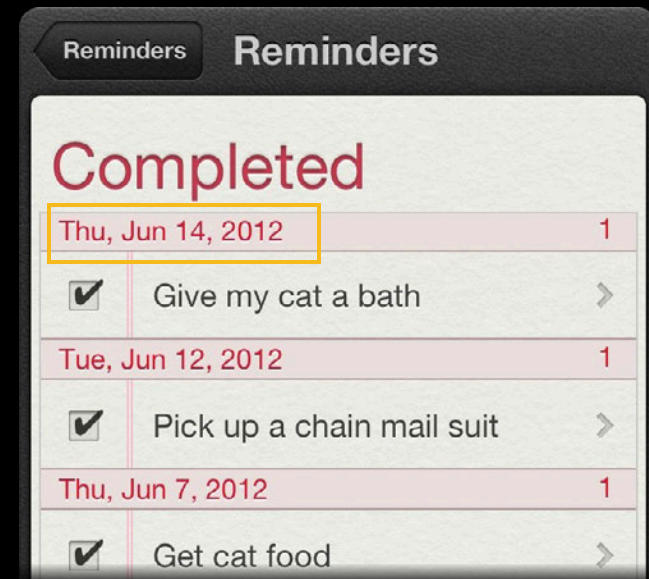
Setting a Completion Date

- Automatically set to the current time when you set

```
reminder.completed = YES;
```

- You can set it to any date using

```
reminder.completionDate = date;
```



Reminders API Summary

“Remind me how that API works.”



Reminders API Summary

“Remind me how that API works.”

- Time-based reminders use EKAlarm with absolute time



Reminders API Summary

“Remind me how that API works.”

- Time-based reminders use EKAlarm with absolute time
- Location-based reminders use EKAlarm with structured location



Reminders API Summary

“Remind me how that API works.”

- Time-based reminders use EKAlarm with absolute time
- Location-based reminders use EKAlarm with structured location
- Recurring reminders use EKRecurrenceRule



Reminders API Summary

“Remind me how that API works.”





- Time-based reminders use EKAlarm with absolute time
- Location-based reminders use EKAlarm with structured location
- Recurring reminders use EKRecurrenceRule
- Mark as done using completed and completionDate properties





Integrating Reminders

Scott Adler
Person of Interest

Edit **Recipes** **+**

-  **Chocolate Cake**
Chocolate cake with chocolate... >
-  **Crêpes Suzette**
Crêpes flambées with grand ma... >
-  **Gaufres de Liège**
Belgian-style waffles >
-  **Ginger snaps**
Nana's secret recipe >
-  **Macarons**
Macarons français: chocolat, pis... >
-  **Tarte aux Fraises**
Delicious tart >

 **Recipes**

 **Unit Conversion**

Recipes

Crêpes Suzette

Edit



Crêpes Suzette

Crêpes flambées with grand marnier.

Preparation time: 20min

Category

Dessert

Ingredients

Flour

200g

Grand Marnier®

10cl

Sugar

50g



Recipes



Unit Conversion

Demo

Recipes 2: Electric Boogaloo

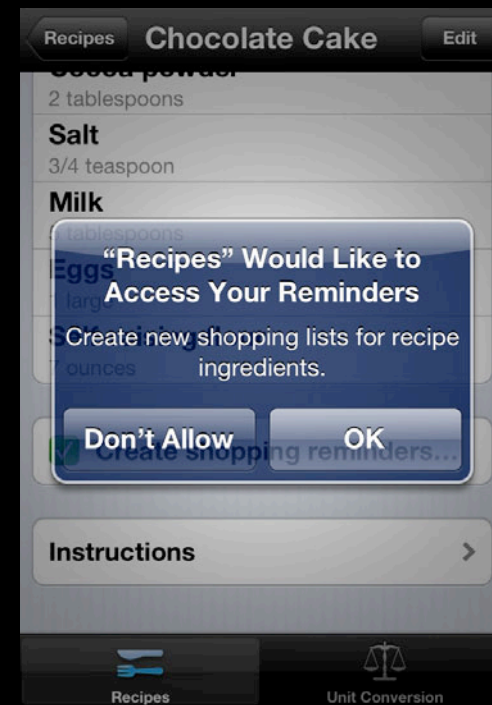
Scott Adler

Person of Interest

Event Kit Data Isolation

Event Kit Data Isolation

Talk to the 



Event Kit Data Isolation

Talk to the 

- User is prompted for access when EKEventStore is instantiated



Event Kit Data Isolation

Talk to the 

- User is prompted for access when EKEventStore is instantiated
- Your code is not blocked when displaying access prompt



Event Kit Data Isolation

Talk to the 

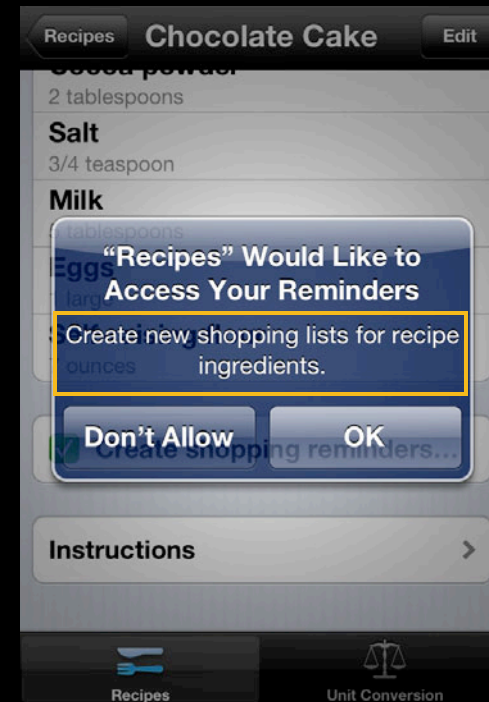
- User is prompted for access when EKEventStore is instantiated
- Your code is not blocked when displaying access prompt
- EKEventStoreChangedNotification fires when authorization changes



Event Kit Data Isolation

Talk to the 🖐️

- User is prompted for access when EKEventStore is instantiated
- Your code is not blocked when displaying access prompt
- EKEventStoreChangedNotification fires when authorization changes
- Usage description stored in Info.plist



Demo

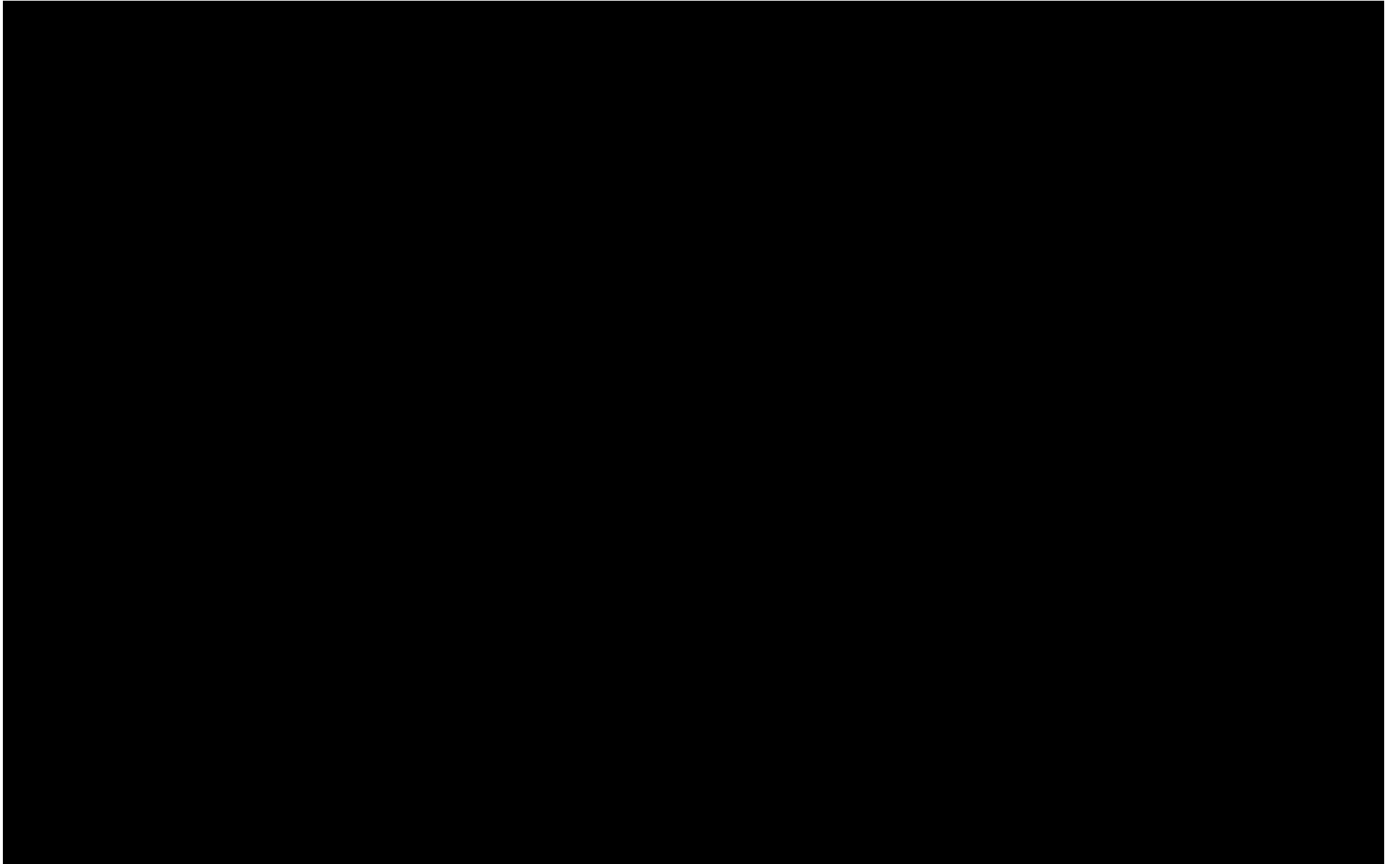
Adapting existing code for privacy

Scott Adler

Person of Interest

Calendar Store on OS X

Jeffrey Harris
Interpersonal Apps



Calendar Store Is Deprecated

Calendar Store and Event Kit Parity

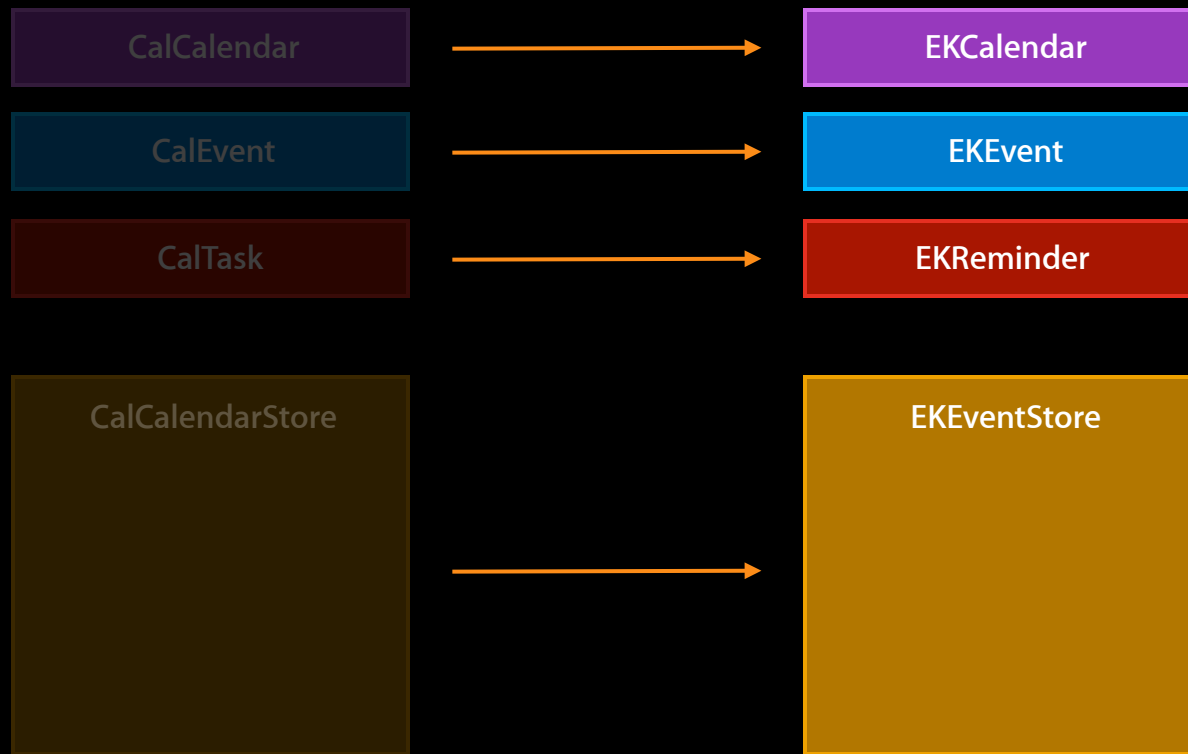
CalCalendar

CalEvent

CalTask

CalCalendarStore

Calendar Store and Event Kit Parity



Fetching Events for a Week

Using EventKit

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeEvent];
NSArray *eventCalendars = [store calendarsForEntityType:EKEntityTypeEvent];
NSDate *monday, *sunday;
... // determine monday and sunday for the current week
NSPredicate *predicate;
predicate = [store predicateForEventsWithStartDate:monday
                        endDate:sunday
                        calendars:eventCalendars];

NSArray *events = [store eventsMatchingPredicate:predicate];
NSLog(@"Events this week:");
for (EKEvent *event in events) {
    NSLog(@"%@ - starts at %@", event.title, event.startDate);
}
```

Fetching Events for a Week

Using CalendarStore

```
CalCalendarStore *store = [CalCalendarStore defaultCalendarStore];

NSArray *eventCalendars = [store calendars];
NSDate *monday, *sunday;
... // determine monday and sunday for the current week
NSPredicate *predicate;
predicate = [store eventPredicateWithStartDate      :monday
                                                endDate:sunday
                                                calendars:eventCalendars];

NSArray *events = [store eventsWithPredicate      :predicate];
NSLog(@"Events this week:");
for (CalEvent *event in events) {
    NSLog(@"%@ - starts at %@", event.title, event.startDate);
}
```


Fetching Events for a Week

Using EventKit

```
EKEventStore *store = [[EKEventStore alloc]
                        initWithAccessToEntityTypes:EKEntityTypeEvent];
NSArray *eventCalendars = [store calendarsForEntityType:EKEntityTypeEvent];
NSDate *monday, *sunday;
... // determine monday and sunday for the current week
NSPredicate *predicate;
predicate = [store predicateForEventsWithStartDate:monday
                        endDate:sunday
                        calendars:eventCalendars];

NSArray *events = [store eventsMatchingPredicate:predicate];
NSLog(@"Events this week:");
for (EKEvent *event in events) {
    NSLog(@"%@ - starts at %@", event.title, event.startDate);
}
```

Event Kit Benefits

Event Kit Benefits

- Works across iOS and OS X

Event Kit Benefits

- Works across iOS and OS X
- Has properties not available in Calendar Store

Event Kit Benefits

- Works across iOS and OS X
- Has properties not available in Calendar Store
 - Time zones for event start

Event Kit Benefits

- Works across iOS and OS X
- Has properties not available in Calendar Store
 - Time zones for event start
 - Types of items allowed in a calendar (events, reminders, or both)

Previous Event Kit Presentation

- Calendar Integration with Event Kit
<https://developer.apple.com/videos/wwdc/2010/?id=136>

Related Sessions

Privacy Support in iOS and OS X

Pacific Heights
Thursday 3:15PM

Staying on Track with Location Services

Nob Hill
Wednesday 2:00PM

Internationalization Tips and Tricks

Marina
Friday 10:15AM

Labs

Event Kit and Reminders Lab

App Services Lab A
Thursday 9:00AM

Core Location Lab

App Services Lab B
Wednesday 3:15PM

Internationalization Lab

App Services Lab A
Friday 11:30AM

Summary

- Events and reminders now available on iOS and OS X
- It's easy!

 WWDC2012

The last 3 slides
after the logo are
intentionally left
blank for all
presentations.

The last 3 slides
after the logo are
intentionally left
blank for all
presentations.

The last 3 slides
after the logo are
intentionally left
blank for all
presentations.