Understanding Foundation

Tony Parker
Software Engineer, Cocoa Frameworks

What You'll Learn

- What the Foundation framework is
- How Foundation fits in with your application
- Most important Foundation features

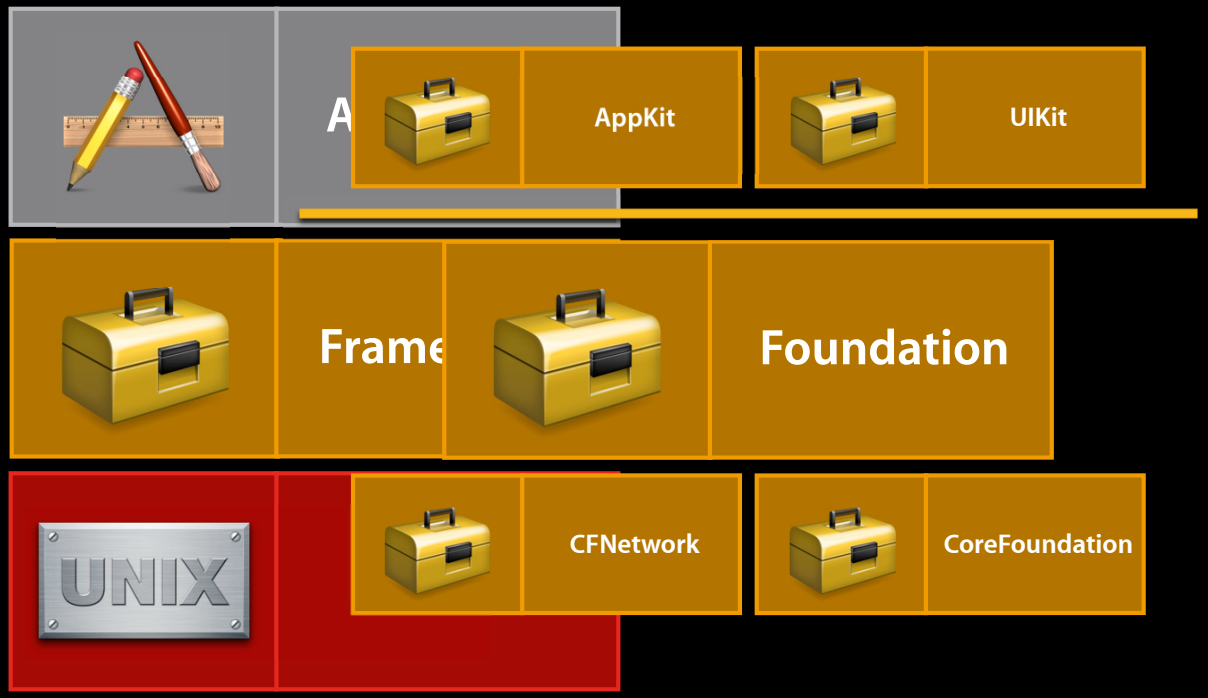
Prerequisites

- Worked on an application
- Familiarity with basic Foundation concepts

What Is Foundation?

- Provides building block classes
  - Fundamental types used by all applications
  - Assembled by higher level software
- Introduces consistent conventions
- Raises lower-level concepts

Where Is Foundation?



Building Blocks

Building Blocks

- Collections
- Strings
- Dates and times
- Persistence and archiving
- Files and URLs
- Bundles
- Operation queues

Collections

Collections

- A place to put your objects
- Most commonly used features:
  - Iterating
  - Sorting
  - Filtering

Picking a Collection

Collection	Prime Directive	Example
NSArray NSMutableArray	Preserves order	Result of 50 meter dash
NSDictionary NSMutableDictionary	Maps key objects to value objects	Phone book
NSSet NSMutableSet	Unique, unordered objects	Suggestion box

Collection Iteration

- `for` loop
- Using NSEnumerator with `while` loop
- Fast enumeration with `for…in`
- Blocks

Iteration Using a For Loop

```
NSUInteger count = [array count];
```

Iteration Using a For Loop

```
NSUInteger count = [array count];
for (NSUInteger i = 0; i < count; i++) {


}
```

Iteration Using a For Loop

```
NSUInteger count = [array count];
for (NSUInteger i = 0; i < count; i++) {
    id value = [array objectAtIndex:i];
    …
}
```

Iteration Using a For Loop

```
NSUInteger count = [array count];
for (NSUInteger i = 0; i < count; i++) {
    id value = [array objectAtIndex:i];
    …
}

NSArray *values = [dictionaryOrSet allObjects];
count = [values count];
```

Iteration Using a For Loop

```
NSUInteger count = [array count];
for (NSUInteger i = 0; i < count; i++) {
    id value = [array objectAtIndex:i];
    …
}

NSArray *values = [dictionaryOrSet allObjects];
count = [values count];
for (NSUInteger i = 0; i < count; i++) {
    id value = [values objectAtIndex:i];
    …
}
```

Iteration Using NSEnumerator

```
NSEnumerator *e = [arrayOrSet objectEnumerator];
while (id object = [e nextObject]) {

    …
}
```

Iteration Using NSEnumerator

```
NSEnumerator *e = [arrayOrSet objectEnumerator];
while (id object = [e nextObject]) {
    // extra step required for index
    …
}
```

Iteration Using NSEnumerator

```
NSEnumerator *e = [arrayOrSet objectEnumerator];
while (id object = [e nextObject]) {
    // extra step required for index
    …
}
```

```
NSEnumerator *e = [dictionary keyEnumerator];
while (id key = [e nextObject]) {


    …
}
```

Iteration Using NSEnumerator

```
NSEnumerator *e = [arrayOrSet objectEnumerator];
while (id object = [e nextObject]) {
    // extra step required for index
    …
}


NSEnumerator *e = [dictionary keyEnumerator];
while (id key = [e nextObject]) {
    id value = [e objectForKey:key];

    …
}
```

Iteration Using NSEnumerator

```
NSEnumerator *e = [arrayOrSet objectEnumerator];
while (id object = [e nextObject]) {
    // extra step required for index
    …
}


NSEnumerator *e = [dictionary keyEnumerator];
while (id key = [e nextObject]) {
    id value = [e objectForKey:key];
    // or, use objectEnumerator and iterate objects directly
    …
}
```

Fast Enumeration

```
for (id object in arrayOrSet) {
    // no easy access to index for array here
    …
}
```

Fast Enumeration

```
for (id object in arrayOrSet) {
    // no easy access to index for array here
    …
}

for (id key in dictionary) {
    // requires extra step to get object
    id value = [dictionary objectForKey:key];
    …
}
```

Iteration Using Blocks


```
[array enumerateObjectsUsingBlock:
   ^(id object, NSUInteger index, BOOL *stop) {
      // your code here
}];
```

Iteration Using Blocks


```
[array enumerateObjectsUsingBlock:
   ^(id object, NSUInteger index, BOOL *stop) {
      // your code here
}];

[dictionary enumerateKeysAndObjectsUsingBlock:
   ^(id key, id object, BOOL *stop) {
      // your code here
}];
```

Iteration Using Blocks


```
[array enumerateObjectsUsingBlock:
   ^(id object, NSUInteger index, BOOL *stop) {
      // your code here
}];

[dictionary enumerateKeysAndObjectsUsingBlock:
   ^(id key, id object, BOOL *stop) {
      // your code here
}];

[set enumerateObjectsUsingBlock:
   ^(id object, BOOL *stop) {
      // your code here
}];
```

Iteration Summary

Minimum OS	Recommended Iteration Method
Mac OS X 10.0	`for` loop NSEnumerator
Mac OS X 10.5 iPhone OS 2	`for` loop Fast enumeration
Mac OS X 10.6 iOS 4	Fast enumeration Block enumeration

Sorting an NSArray

- Determine sort order using:
  - C function
  - Objective-C method
  - NSSortDescriptor
  - Blocks
- For immutable or mutable arrays, return a new object
- For mutable arrays, method to sort in place

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
```

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
[names sortUsingComparator:^(id left, id right) {









}];
```

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
[names sortUsingComparator:^(id left, id right) {
    NSComparisonResult result;









    return result;
}];
```

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
[names sortUsingComparator:^(id left, id right) {
    NSComparisonResult result;
    NSUInteger lLen = [left length], rLen = [right length];









    return result;
}];
```

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
[names sortUsingComparator:^(id left, id right) {
    NSComparisonResult result;
    NSUInteger lLen = [left length], rLen = [right length];
    if (lLen < rLen) {
        result = NSOrderedAscending;
    }



    return result;
}];
```

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
[names sortUsingComparator:^(id left, id right) {
    NSComparisonResult result;
    NSUInteger lLen = [left length], rLen = [right length];
    if (lLen < rLen) {
        result = NSOrderedAscending;
    } else if (lLen > rLen) {
        result = NSOrderedDescending;
    }


    return result;
}];
```

Sorting Example


```
NSMutableArray *names = …; // array of NSString objects
[names sortUsingComparator:^(id left, id right) {
    NSComparisonResult result;
    NSUInteger lLen = [left length], rLen = [right length];
    if (lLen < rLen) {
        result = NSOrderedAscending;
    } else if (lLen > rLen) {
        result = NSOrderedDescending;
    } else {
        result = NSOrderedSame;
    }
    return result;
}];
```

Filtering

- Mutating a collection while enumerating will cause an exception
- To filter a collection:
  - Mutate a copy
  - Gather changes on the side, then apply

Filtering Example

```
NSMutableArray *files = …; // array of NSString objects
NSIndexSet *toRemove = …
```

Filtering Example

```
NSMutableArray *files = …; // array of NSString objects
NSIndexSet *toRemove =
    [files indexesOfObjectsPassingTest:
        ^(id obj, NSUInteger idx, BOOL *stop) {


    }];
```

Filtering Example

```
NSMutableArray *files = …; // array of NSString objects
NSIndexSet *toRemove =
    [files indexesOfObjectsPassingTest:
        ^(id obj, NSUInteger idx, BOOL *stop) {
            if ([obj hasPrefix:@"."]) return YES;
            return NO;
    }];
```

Filtering Example

```
NSMutableArray *files = …; // array of NSString objects
NSIndexSet *toRemove =
    [files indexesOfObjectsPassingTest:
        ^(id obj, NSUInteger idx, BOOL *stop) {
            if ([obj hasPrefix:@"."]) return YES;
            return NO;
    }];

[files removeObjectsAtIndexes:toRemove];
```

More Collection Features

- Searching
- Apply selector to each item
- NSArray: Slicing and concatenation
- NSSet: Intersection, union, set subtraction

Strings

Strings

- Object container for most text on the system
- Array of Unicode characters
- Treat as opaque container
- Common uses:
  - Comparison
  - Searching
  - Converting encodings

Comparing Strings

- Comparison methods:

```
-(NSComparisonResult)compare:(NSString *)string;

-(NSComparisonResult)localizedCompare:(NSString *)string;
```


```
-(NSComparisonResult)localizedStandardCompare:(NSString *)string;

-(NSComparisonResult)compare:(NSString *)string
                     options:(NSStringCompareOptions)mask
                       range:(NSRange)compareRange
                      locale:(id)locale;
```

Comparison Example

```
NSString *str1 = @"string A";
NSString *str2 = @"string B";
```

Comparison Example

```
NSString *str1 = @"string A";
NSString *str2 = @"string B";
NSComparisonResult result = [str1 compare:str2];
```

Comparison Example

```
NSString *str1 = @"string A";
NSString *str2 = @"string B";
NSComparisonResult result = [str1 compare:str2];
// result is: NSOrderedAscending
```

Comparison Example

```
NSString *str1 = @"string A";
NSString *str2 = @"string B";
NSComparisonResult result = [str1 compare:str2];
// result is: NSOrderedAscending


NSArray *strings =
    [NSArray arrayWithObjects:@"Larry", @"Curly", @"Moe", nil];
```

Comparison Example

```
NSString *str1 = @"string A";
NSString *str2 = @"string B";
NSComparisonResult result = [str1 compare:str2];
// result is: NSOrderedAscending


NSArray *strings =
    [NSArray arrayWithObjects:@"Larry", @"Curly", @"Moe", nil];
NSArray *sortedStrings = [strings sortedArrayUsingSelector:
    @selector(localizedCompare:)];
// result is: "Curly", "Larry", "Moe"
```

Searching Strings

- Searching methods:

```
-(NSRange)rangeOfString:(NSString *)aString;

-(NSRange)rangeOfString:(NSString *)aString
                options:(NSStringCompareOptions)mask
                  range:(NSRange)searchRange
                 locale:(NSLocale *)locale;
```

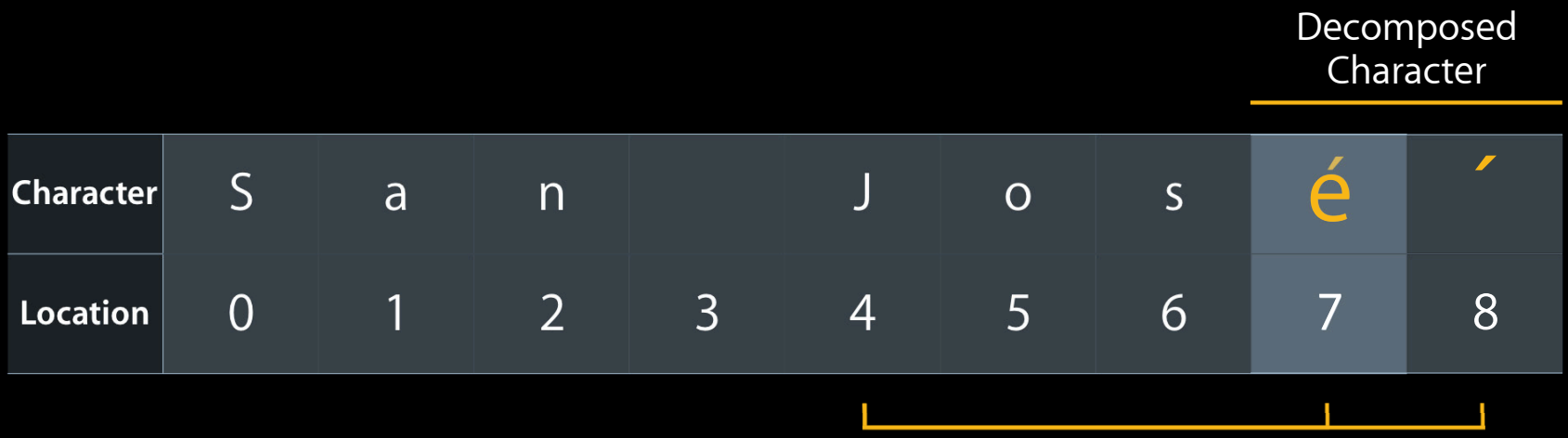
- NSRange has a location and length

Searching Strings

Character	S	a	n		J	o	s	e
Location	0	1	2	3	4	5	6	7

```
NSRange found = [str rangeOfString:@"Jose"];
// found.location = 4, found.length = 4
```

Searching Strings




```
NSRange found = [str rangeOfString:@"José"];
// found.location = 4, found.length = 4
```

Searching Strings

```
found = [str rangeOfString:@"missing"];
// found.length = 0
```

Searching Strings

```
found = [str rangeOfString:@"missing"];
// found.length = 0

str = @"Going going gone!";
found = [str rangeOfString:@"go(\\w*)"
                   options:NSRegularExpressionSearch
                     range:NSMakeRange(0, [str length])];
// found.location = 6, found.length = 5
```

Advanced Text Handling for iPhone OS	Nob Hill Tuesday 4:30PM

String Encodings

- An encoding is a map of numbers to characters
- For example, in NSASCIIStringEncoding:

Value	Character
65	A
97	a
98	b
126	~

Converting Encodings

```
NSData *data = …; // From file or network, perhaps
NSString *inString =
    [[NSString alloc] initWithData:data
                          encoding:NSUTF8StringEncoding];
```

Converting Encodings

```
NSData *data = …; // From file or network, perhaps
NSString *inString =
    [[NSString alloc] initWithData:data
                          encoding:NSUTF8StringEncoding];

NSString *outString = @"For Windows";
NSData *converted =
    [outString dataUsingEncoding:NSUTF16StringEncoding];
```

Converting Encodings

```
NSData *data = …; // From file or network, perhaps
NSString *inString =
    [[NSString alloc] initWithData:data
                          encoding:NSUTF8StringEncoding];

NSString *outString = @"For Windows";
NSData *converted =
    [outString dataUsingEncoding:NSUTF16StringEncoding];

const char *fileName = [outString fileSystemRepresentation];
```

More String Features

- printf-style formatting
- Enumeration of substrings, lines, and paragraphs
- Replacement of substrings
- Path completion
- Mutability

Dates and Times

Date and Time Classes

- Encapsulate complexity of date and time calculations
- Automatically handle user preferences
- Common uses:
  - Represent a date in a calendar
  - Find amount of time between two dates
  - Format a date or number correctly for user in any locale

Why Use Foundation Date and Time Classes?

- Display time 12 hours from now

Date	Expected	Correct	Why a difference?
November 5, 2006 10:00 PM	November 6, 2006 10:00 AM	November 6, 2006 10:00 AM	None—it worked!
November 4, 2007 10:00 PM	November 5, 2007 10:00 AM	November 5, 2007 9:00 AM	Daylight Saving Time ended
November 4, 2007 10:00 PM	November 5, 2007 9:00 AM	November 5, 2007 10:00 AM	User lives in Phoenix

Date and Time Classes

NSDate

- An invariant point in time
- Calendar and time zone independent
- Time is measured as seconds since a reference point, a double

Date and Time Classes

NSCalendar

- Defines beginning, length, and divisions of a year
- Example: Gregorian calendar, Hebrew calendar

Date and Time Classes

NSDateComponents

- Simple object structure
- Contains storage for year, month, day, etc.
- Exact meaning of properties depends on calendar

Date Example

Calculate the number of shopping days between US Thanksgiving and Christmas

- Christmas is on the same date every year, December 25
- Thanksgiving is the 4th Thursday in November
- Shopping season starts the day after Thanksgiving and ends Christmas Eve

Find Christmas

```
NSCalendar *cal = [[NSCalendar alloc]
  initWithCalendarIdentifier:NSGregorianCalendar];
```

Find Christmas

```
NSCalendar *cal = [[NSCalendar alloc]
  initWithCalendarIdentifier:NSGregorianCalendar];

NSDateComponents *components = [[NSDateComponents alloc] init];
[components setYear:2010];
[components setMonth:12];
[components setDay:25];
```

Find Christmas

```
NSCalendar *cal = [[NSCalendar alloc]
  initWithCalendarIdentifier:NSGregorianCalendar];

NSDateComponents *components = [[NSDateComponents alloc] init];
[components setYear:2010];
[components setMonth:12];
[components setDay:25];

NSDate *christmas = [cal dateFromComponents:components];
```

Find Thanksgiving

```
NSDateComponents *tComps = [[NSDateComponents alloc] init];
[tComps setYear:2010];
[tComps setMonth:11];
[tComps setWeekday:5]; // 5 is Thursday in NSGregorianCalendar
[tComps setWeekdayOrdinal:4];
```

Find Thanksgiving

```
NSDateComponents *tComps = [[NSDateComponents alloc] init];
[tComps setYear:2010];
[tComps setMonth:11];
[tComps setWeekday:5]; // 5 is Thursday in NSGregorianCalendar
[tComps setWeekdayOrdinal:4];

NSDate *thanksgiving = [cal dateFromComponents:tComps];
```

Adding or Subtracting Date Components

```
NSDateComponents *toAdd = [[NSDateComponents alloc] init];
[toAdd setDay:1];
```

Adding or Subtracting Date Components

```
NSDateComponents *toAdd = [[NSDateComponents alloc] init];
[toAdd setDay:1];

NSDate *blackFri = [cal dateByAddingComponents:toAdd
                                        toDate:thanksgiving
                                       options:0];
```

Time Between Two Dates

```
NSDateComponents *diff =
    [cal components:NSDayCalendarUnit
           fromDate:blackFri
             toDate:christmas
            options:0];
```

Time Between Two Dates

```
NSDateComponents *diff =
    [cal components:NSDayCalendarUnit
           fromDate:blackFri
             toDate:christmas
            options:0];

NSInteger days = [diff day];
```

Time Between Two Dates

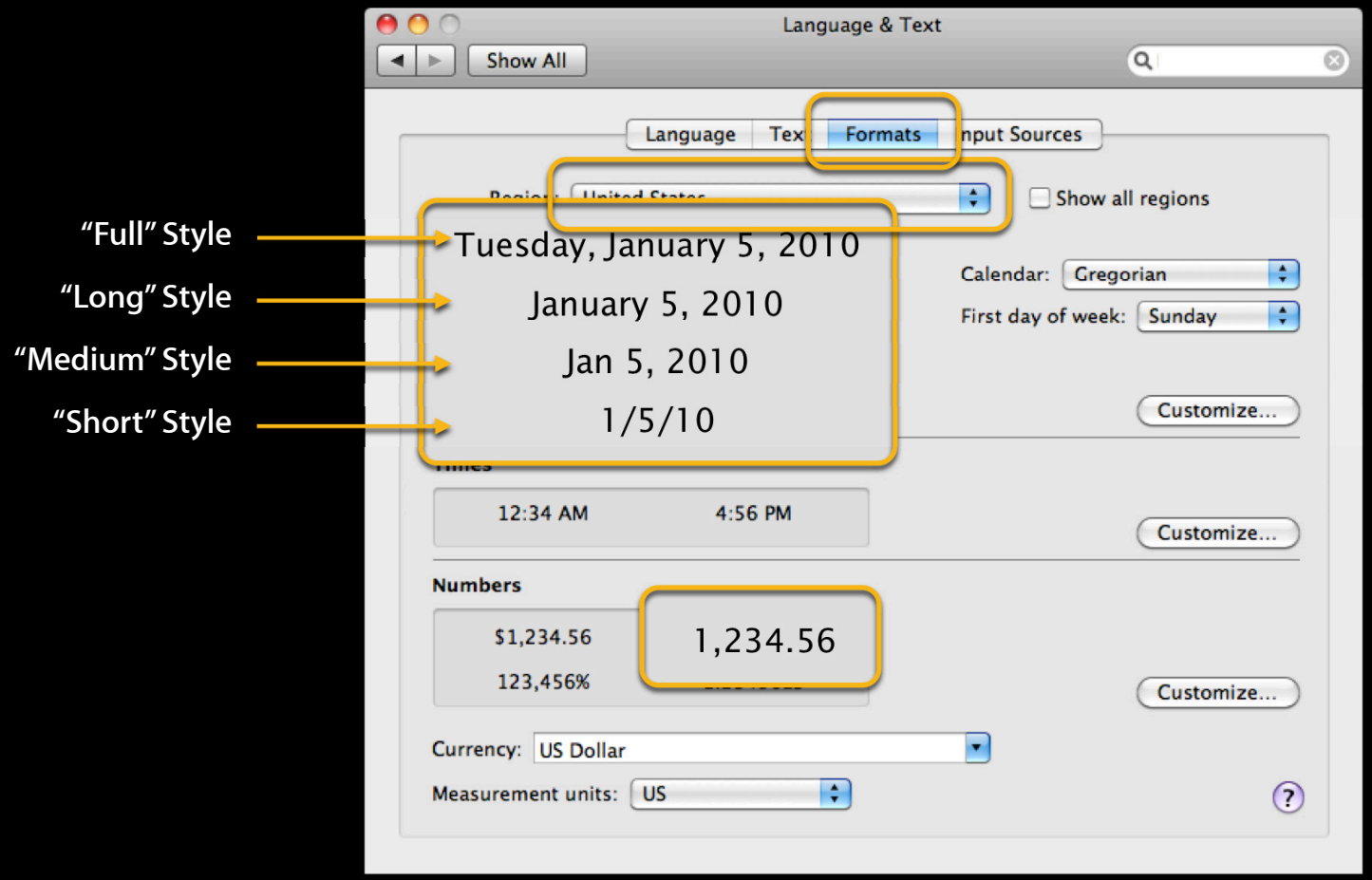
```
NSDateComponents *diff =
    [cal components:NSDayCalendarUnit
           fromDate:blackFri
             toDate:christmas
            options:0];

NSInteger days = [diff day];
// days == 29
```

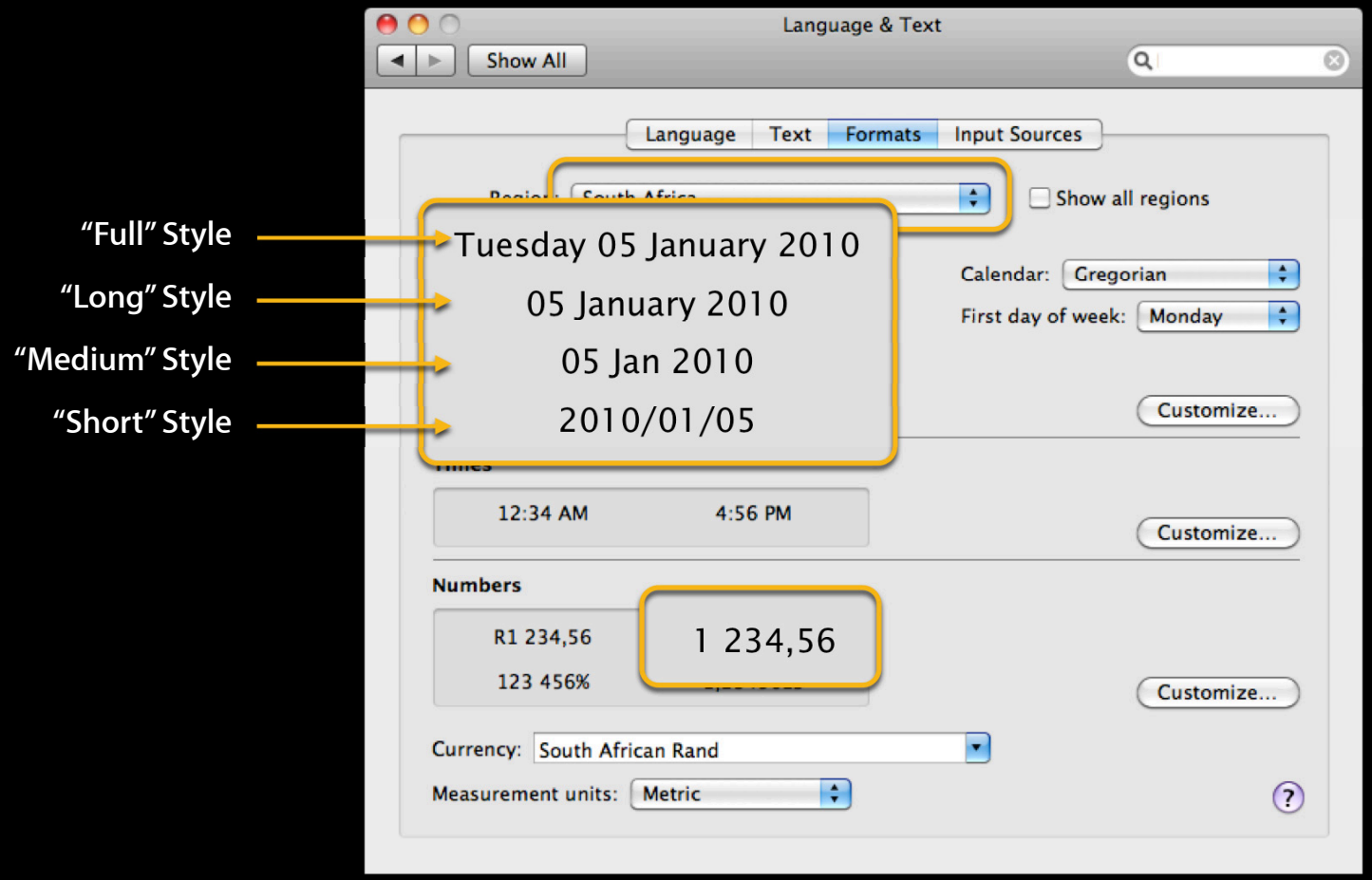
More Date and Time Features

- Find day of the week for a date
- Time zone calculations
- Converting between calendars

NSFormatter

- NSDateFormatter
  - Convert date to or from a string
- NSNumberFormatter
  - Convert number to or from a string
- Each class has a set of pre-defined styles, or use your own


"Full" Style
"Long" Style
"Medium" Style
"Short" Style
Language & Text
Show All
Language
Tex
Formats
nput Sources
Show all regions
Tuesday, January 5, 2010
January 5, 2010
Jan 5, 2010
1/5/10
Calendar: Gregorian
First day of week: Sunday
Customize...
12:34 AM
4:56 PM
Customize...
Numbers
$1,234.56
123,456%
1,234.56
Customize...
Currency: US Dollar
Measurement units: US


"Full" Style
"Long" Style
"Medium" Style
"Short" Style
Language & Text
Show All
Language
Text
Formats
Input Sources
Show all regions
Tuesday 05 January 2010
05 January 2010
05 Jan 2010
2010/01/05
Calendar: Gregorian
First day of week: Monday
Customize...
12:34 AM
4:56 PM
Customize...
Numbers
R1 234,56
123 456%
1 234,56
Customize...
Currency: South African Rand
Measurement units: Metric

Get String from Date

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
```

Get String from Date

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setTimeStyle:NSDateFormatterNoStyle];
[fmt setDateStyle:NSDateFormatterLongStyle];
```

Get String from Date

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setTimeStyle:NSDateFormatterNoStyle];
[fmt setDateStyle:NSDateFormatterLongStyle];

NSLog(@"Thanksgiving is: %@",
    [fmt stringFromDate:thanksgiving]);
```

Get String from Date

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setTimeStyle:NSDateFormatterNoStyle];
[fmt setDateStyle:NSDateFormatterLongStyle];

NSLog(@"Thanksgiving is: %@",
    [fmt stringFromDate:thanksgiving]);

> Thanksgiving is: November 25, 2010
```

Get Date from String

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setDateFormat:@"dd/MM/yyyy HH:mm"];
```

Get Date from String

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setDateFormat:@"dd/MM/yyyy HH:mm"];
[fmt setTimeZone:
    [NSTimeZone timeZoneWithName:@"America/Los_Angeles"]];
[fmt setCalendar:cal];
```

Get Date from String

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setDateFormat:@"dd/MM/yyyy HH:mm"];
[fmt setTimeZone:
    [NSTimeZone timeZoneWithName:@"America/Los_Angeles"]];
[fmt setCalendar:cal];

NSDate *date = [formatter dateFromString:@"10/06/2010 9:30"];
```

Get Date from String

```
NSDateFormatter *fmt = [[NSDateFormatter alloc] init];
[fmt setDateFormat:@"dd/MM/yyyy HH:mm"];
[fmt setTimeZone:
    [NSTimeZone timeZoneWithName:@"America/Los_Angeles"]];
[fmt setCalendar:cal];

NSDate *date = [formatter dateFromString:@"10/06/2010 9:30"];
```

- Format strings conform to Unicode Standard TR35-6

Internationalizing Data on Mac and iPhone	Russian Hill Thursday 10:15AM

Dates and Formatters Summary

- Use NSDate and NSCalendar together for time calculations
- Use formatters when presenting dates and numbers

Persistence and Archiving

Persistence and Archiving

- Store data to disk for later retrieval
  - Property lists
  - User preferences
  - Keyed archives
  - Large, complex data structures

NSPropertyListSerialization

- Stores a small amount of structured data
- Cross-platform and human readable (XML format)
- Property list types:
  - NSArray, NSDictionary
  - NSString
  - NSDate
  - NSNumber (integer, floating point, boolean)
  - NSData

Property List Example

```
NSDictionary *colors = [NSDictionary
    dictionaryWithObjectsAndKeys:@"Verde", @"Green", @"Rojo",
        @"Red", @"Amarillo", @"Yellow", nil];
```

Property List Example

```
NSDictionary *colors = [NSDictionary
    dictionaryWithObjectsAndKeys:@"Verde", @"Green", @"Rojo",
        @"Red", @"Amarillo", @"Yellow", nil];

NSError *error = nil;
NSData *plist = [NSPropertyListSerialization
    dataWithPropertyList:colors
                  format:NSPropertyListXMLFormat_v1_0
                 options:0
                   error:&error];
```

Property List Example

```
NSDictionary *colors = [NSDictionary
    dictionaryWithObjectsAndKeys:@"Verde", @"Green", @"Rojo",
        @"Red", @"Amarillo", @"Yellow", nil];

NSError *error = nil;
NSData *plist = [NSPropertyListSerialization
    dataWithPropertyList:colors
                  format:NSPropertyListXMLFormat_v1_0
                 options:0
                   error:&error];

if (!plist) [NSApp presentError:error];
```

Property List Example

```
NSData *readData =
    [NSData dataWithContentsOfURL:urlOfFile];
```

Property List Example

```
NSData *readData =
    [NSData dataWithContentsOfURL:urlOfFile];

NSDictionary *newColors = [NSPropertyListSerialization
    propertyListWithData:readData
                 options:0
                  format:nil
                   error:&error];
```

Property List Example

```
NSData *readData =
    [NSData dataWithContentsOfURL:urlOfFile];

NSDictionary *newColors = [NSPropertyListSerialization
    propertyListWithData:readData
                 options:0
                  format:nil
                   error:&error];

if (!newColors) [NSApp presentError:error];
```

NSUserDefaults

- Stores small values representing user preferences
- Organized by domain, which is a group of defaults
- Domains searched in specific order:
  - Arguments to executable (volatile)
  - Application (user home directory)
  - Global (system-wide location)
  - Language (volatile)
  - Registration (volatile)

Set Registration Domain Defaults

```
+ (void)initialize {









}
```

Set Registration Domain Defaults

```
+ (void)initialize {
    NSDictionary *appDefaults = [NSDictionary
      dictionaryWithObjectsAndKeys:
        [NSNumber numberWithBool:YES], @"FrogBlastVentCore",
        @"iPad", @"DefaultDeviceName",
        [NSNumber numberWithInt:3], @"PiValue", nil];




}
```

Set Registration Domain Defaults

```
+ (void)initialize {
    NSDictionary *appDefaults = [NSDictionary
      dictionaryWithObjectsAndKeys:
        [NSNumber numberWithBool:YES], @"FrogBlastVentCore",
        @"iPad", @"DefaultDeviceName",
        [NSNumber numberWithInt:3], @"PiValue", nil];

    NSUserDefaults *defaults =
        [NSUserDefaults standardUserDefaults];


}
```

Set Registration Domain Defaults

```
+ (void)initialize {
    NSDictionary *appDefaults = [NSDictionary
      dictionaryWithObjectsAndKeys:
        [NSNumber numberWithBool:YES], @"FrogBlastVentCore",
        @"iPad", @"DefaultDeviceName",
        [NSNumber numberWithInt:3], @"PiValue", nil];

    NSUserDefaults *defaults =
        [NSUserDefaults standardUserDefaults];

    [defaults registerDefaults:appDefaults];
}
```

Read User Default

```
BOOL doIt = [defaults boolForKey:@"FrogBlastVentCore"];
// doIt == YES
```

Read User Default

```
BOOL doIt = [defaults boolForKey:@"FrogBlastVentCore"];
// doIt == YES

// MyApp.app/Contents/MacOS/MyApp -ConferenceName WWDC
```

Read User Default

```
BOOL doIt = [defaults boolForKey:@"FrogBlastVentCore"];
// doIt == YES

// MyApp.app/Contents/MacOS/MyApp -ConferenceName WWDC
NSString *argPref = [defaults stringForKey:@"ConferenceName"];
// argPref is "WWDC"
```

Set User Default

```
[defaults setBool:NO forKey:@"FrogBlastVentCore"];
```

Set User Default

```
[defaults setBool:NO forKey:@"FrogBlastVentCore"];

// later...
doIt = [defaults boolForKey:@"FrogBlastVentCore"];
// doIt == NO
```

Keyed Archiving

- Stores arbitrary graph of objects
- Enables easy backward and forward compatibility
- Allows for substitutions during encode and decode
- Objects do not need to be property list types
- Object must implement NSCoding protocol

Implementing NSCoding

```
@interface Robot : NSObject <NSCoding>




@end
```

Implementing NSCoding

```
@interface Robot : NSObject <NSCoding> {
    NSString *name;
    Robot *nemesis;
    NSInteger model;
}
@property (copy) NSString *name;
@property (retain) Robot *nemesis;
@property NSInteger model;
@end
```

Implementing NSCoding

```
- (void)encodeWithCoder:(NSCoder *)coder {


}
```

Implementing NSCoding

```
- (void)encodeWithCoder:(NSCoder *)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeObject:nemesis forKey:@"nemesis"];
    [coder encodeInteger:model forKey:@"model"];
}
```

Implementing NSCoding

```
- (void)encodeWithCoder:(NSCoder *)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeObject:nemesis forKey:@"nemesis"];
    [coder encodeInteger:model forKey:@"model"];
}

- (id)initWithCoder:(NSCoder *)coder {




}
```

Implementing NSCoding

```
- (void)encodeWithCoder:(NSCoder *)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeObject:nemesis forKey:@"nemesis"];
    [coder encodeInteger:model forKey:@"model"];
}

- (id)initWithCoder:(NSCoder *)coder {
    self = [super init];



    return self;
}
```

Implementing NSCoding

```
- (void)encodeWithCoder:(NSCoder *)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeObject:nemesis forKey:@"nemesis"];
    [coder encodeInteger:model forKey:@"model"];
}

- (id)initWithCoder:(NSCoder *)coder {
    self = [super init];
    name = [[coder decodeObjectForKey:@"name"] copy];
    nemesis = [[coder decodeObjectForKey:@"nemesis"] retain];
    model = [coder decodeIntegerForKey:@"model"];
    return self;
}
```

Archiving

```
Robot *r1 = [[Robot alloc] init], *r2 = [[Robot alloc] init];
```

Archiving

```
Robot *r1 = [[Robot alloc] init], *r2 = [[Robot alloc] init];
r1.name = @"Bender"; r1.nemesis = r2; r1.model = 22;
r2.name = @"Flexo";  r2.nemesis = r1; r2.model = 22;
```

Archiving

```
Robot *r1 = [[Robot alloc] init], *r2 = [[Robot alloc] init];
r1.name = @"Bender"; r1.nemesis = r2; r1.model = 22;
r2.name = @"Flexo";  r2.nemesis = r1; r2.model = 22;

NSData *data = [NSKeyedArchiver archivedDataWithRootObject:r2];
```

Unarchiving

```
Robot *r3 = [NSKeyedUnarchiver unarchiveObjectWithData:data];
```

Unarchiving

```
Robot *r3 = [NSKeyedUnarchiver unarchiveObjectWithData:data];

NSLog(@"Nemesis is: %@", r3.nemesis.name);
```

Unarchiving

```
Robot *r3 = [NSKeyedUnarchiver unarchiveObjectWithData:data];

NSLog(@"Nemesis is: %@", r3.nemesis.name);
> Nemesis is: Bender
```

Core Data

- Stores large, complex graphs of objects
- Features:
  - Support for key-value coding and key-value observing
  - Validation of values and consistency of relationships
  - Change tracking and undo
  - Sophisticated queries
  - Incremental editing

Mastering Core Data	Russian Hill Wednesday 2:00PM

Persistence Cheat Sheet

Data Type	Recommended Persistence Method
User preferences	NSUserDefaults
Small files, cross-platform	NSPropertyListSerialization
Object graph with cycles, non-property list types	NSKeyedArchiver
Large data set, object relational database	Core Data
Specialized data	Custom Format

Files and URLs

Files and URLs

- NSURL is the preferred way to reference files and resources
- URLs are used with the file system through NSFileManager
- URLs are used with a network resource through the URL loading classes

Creating URLs

```
NSURL *file = [NSURL fileURLWithPath:@"/Users/tony/file.txt"];
```

Creating URLs

```
NSURL *file = [NSURL fileURLWithPath:@"/Users/tony/file.txt"];
NSURL *up = [file URLByDeletingLastPathComponent];
```

Creating URLs

```
NSURL *file = [NSURL fileURLWithPath:@"/Users/tony/file.txt"];
NSURL *up = [file URLByDeletingLastPathComponent];
NSURL *file2 = [up URLByAppendingPathComponent:@"file2.txt"];
```

Creating URLs

```
NSURL *file = [NSURL fileURLWithPath:@"/Users/tony/file.txt"];
NSURL *up = [file URLByDeletingLastPathComponent];
NSURL *file2 = [up URLByAppendingPathComponent:@"file2.txt"];

NSURL *aapl = [NSURL URLWithString:@"http://www.apple.com"];
```

More NSURL Features

- File reference URLs
  - Track a file independent of its location on disk
- File system resource properties

```
NSImage *icon = nil;
BOOL res = [fileURL getResourceValue:&icon
                              forKey:NSURLEffectiveIconKey
                               error:&error];
```

  - Localized name
  - File size
  - Creation, access, modification date
  - Label color and more

NSFileManager

- Copy, move, link, or delete files

```
NSFileManager *mgr = [[NSFileManager alloc] init];
BOOL res;
res = [mgr copyItemAtURL:src toURL:dst error:&error];
res = [mgr moveItemAtURL:src toURL:dst error:&error];
res = [mgr linkItemAtURL:src toURL:dst error:&error];
res = [mgr removeItemAtURL:src error:&error];
```

- Enumerate directory contents

```
NSArray *stuff = [mgr contentsOfDirectoryAtURL:dirURL
                    includingPropertiesForKeys:[NSArray array]
                                       options:0
                                         error:&error];
```

More NSFileManager Features

- Find system directories (Applications, Desktop, etc.)
- Delegate methods for detailed control
- Get and set permissions on files

URL Loading System

- NSURLConnection
  - Performs loading of the URL
  - Two modes of operation
    - Asynchronous using run loop and delegation
    - Synchronous for background threads
- NSURLRequest and NSURLResponse
  - Stores data used when loading

Load URL

```
NSURLRequest *req = [NSURLRequest requestWithURL:aapl];
```

Load URL

```
NSURLRequest *req = [NSURLRequest requestWithURL:aapl];
NSURLConnection *conn =
    [[NSURLConnection alloc] initWithRequest:req
                                    delegate:self
                            startImmediately:YES];
```

Load URL

```
NSURLRequest *req = [NSURLRequest requestWithURL:aapl];
NSURLConnection *conn =
    [[NSURLConnection alloc] initWithRequest:req
                                    delegate:self
                            startImmediately:YES];

// continue execution, including running the main run loop
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
 didReceiveResponse:(NSURLResponse *)response {

}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
 didReceiveResponse:(NSURLResponse *)response {
    rxData = [[NSMutableData alloc] init];
}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
 didReceiveResponse:(NSURLResponse *)response {
    rxData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)conn
    didReceiveData:(NSData *)data {

}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
 didReceiveResponse:(NSURLResponse *)response {
    rxData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)conn
    didReceiveData:(NSData *)data {
    [rxData appendData:data];
}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
  didFailWithError:(NSError *)error {


}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
  didFailWithError:(NSError *)error {
    [rxData release]; rxData = nil;
    // error handling
}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
  didFailWithError:(NSError *)error {
    [rxData release]; rxData = nil;
    // error handling
}

- (void)connectionDidFinishLoading:(NSURLConnection *)conn {

}
```

URL Connection Delegate

```
- (void)connection:(NSURLConnection *)conn
  didFailWithError:(NSError *)error {
    [rxData release]; rxData = nil;
    // error handling
}

- (void)connectionDidFinishLoading:(NSURLConnection *)conn {
    // loading finished, use rxData
}
```

More NSURL Networking Features

- Cache management
- Authentication
- Cookies
- Protocol support

Bundles

Bundles

- Group code and resources
- Include code for different platforms and architectures
- Simplify loading localized resources

Example Bundle Layout (Mac OS)

```
MyApplication.app/
    Contents/
        Info.plist
        MacOS/
            MyApplication
        Resources/
            MyApplication.icns
            en.lproj/
                localizedImage.png
                MainMenu.nib
                Localizable.strings
            fr.lproj/
                localizedImage.png
                MainMenu.nib
                Localizable.strings
```

Loading a Bundle Resource

```
// Get application bundle
NSBundle *bundle = [NSBundle mainBundle];
```

Loading a Bundle Resource

```
// Get application bundle
NSBundle *bundle = [NSBundle mainBundle];

NSURL *url = [bundle URLForResource:@"localizedImage"
                      withExtension:@"png"];
```

Loading a Bundle Resource

```
// Get application bundle
NSBundle *bundle = [NSBundle mainBundle];

NSURL *url = [bundle URLForResource:@"localizedImage"
                      withExtension:@"png"];
```

en.lproj/localizedImage.png

fr.lproj/localizedImage.png

Bundles vs. Packages

	Bundle	File Package	Bundle + File Package
Primary Class	NSBundle	NSFileWrapper	NSBundle
Use For	Code and Resources	User Documents	Application Code and Resources
Example	.framework	.rtfd	.app

More Bundle Features

- Load other bundles
- Locate components of a bundle
- List all resources of a type or in a subdirectory
- Support for localized strings

Operations and Operation Queues

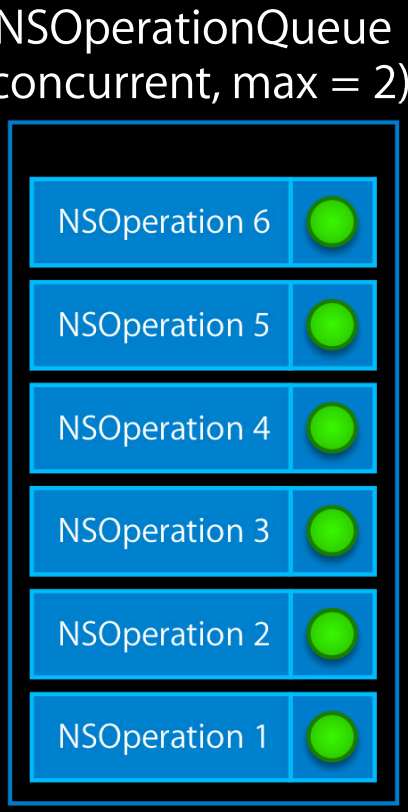
Operations and Operation Queues

- Object-oriented way to describe work
- Simplifies design of concurrent applications

Operations and Operation Queues

- NSOperation
  - Encapsulates a work unit
  - Subclass and override "-main" to define your work
  - Or use one of the Foundation-provided subclasses:
    - NSInvocationOperation
    - NSBlockOperation
- NSOperationQueue
  - Maintains list of operations to perform
  - Runs operations concurrently or serially

How Operation Queues Work



Using an NSOperation Subclass

```
@interface MyOp : NSOperation
@end
```

Using an NSOperation Subclass

```
@interface MyOp : NSOperation
@end
@implementation MyOp
- (void)main {
    // Your work goes here
}
@end
```

Using an NSOperation Subclass

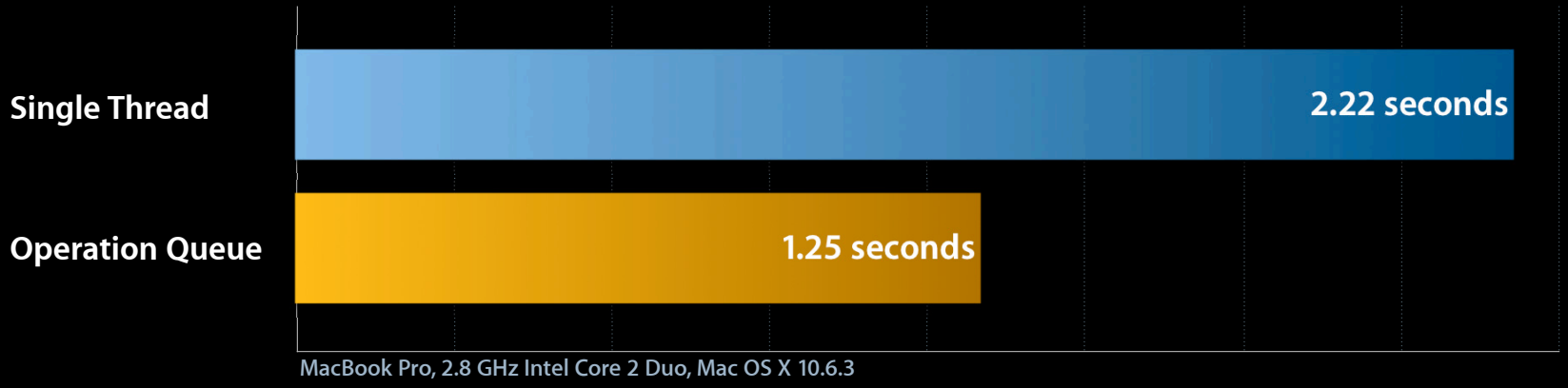
```
@interface MyOp : NSOperation
@end
@implementation MyOp
- (void)main {
    // Your work goes here
}
@end

// elsewhere...
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
```

Using an NSOperation Subclass

```
@interface MyOp : NSOperation
@end
@implementation MyOp
- (void)main {
    // Your work goes here
}
@end

// elsewhere...
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
MyOp *op = [[MyOp alloc] init];
[queue addOperation:op];
```

Operation Queue Benefits

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
for (id img in allImages) {
    [queue addOperationWithBlock:^{
        // Image processing on 'img'
    }];
}
[queue waitUntilAllOperationsAreFinished];
```



Other NSOperation Features

- Dependencies, even across different queues
- Priorities
- Key-value observing compatible properties
- Completion block 

Building Blocks Summary

- Collections
  - Store your objects
- Strings
  - Create, search, and manipulate characters
- Dates and times
  - Easily handle complex locale-dependent dates
- Persistence and archiving
  - Storing your data to disk

Building Blocks Summary

- Files and URLs
  - Efficiently perform file system operations
- Bundles
  - Simplify cross-platform development and localization
- Operation queues
  - Take advantage of multi-core computers with minimal complexity

Learning More

Related Sessions and Labs

API Design for Cocoa and Cocoa Touch	Marina Thursday 4:30PM
What's New in Foundation for iOS 4	Pacific Heights Tuesday 10:15AM
Advanced Objective-C and Garbage Collection Techniques	Pacific Heights Friday 11:30AM
Cocoa Lab	Application Frameworks Lab D Thursday 2:00PM

More Information

Bill Dudney
Frameworks Evangelist
dudney@apple.com

Documentation
Foundation Framework Reference
http://developer.apple.com/mac/library/documentation/Cocoa/Reference/Foundation/ObjC_classic/index.html

Apple Developer Forums
http://devforums.apple.com


WWDC10