Advanced Text Handling for iPhone OS

Douglas Davidson
Natural Languages Group

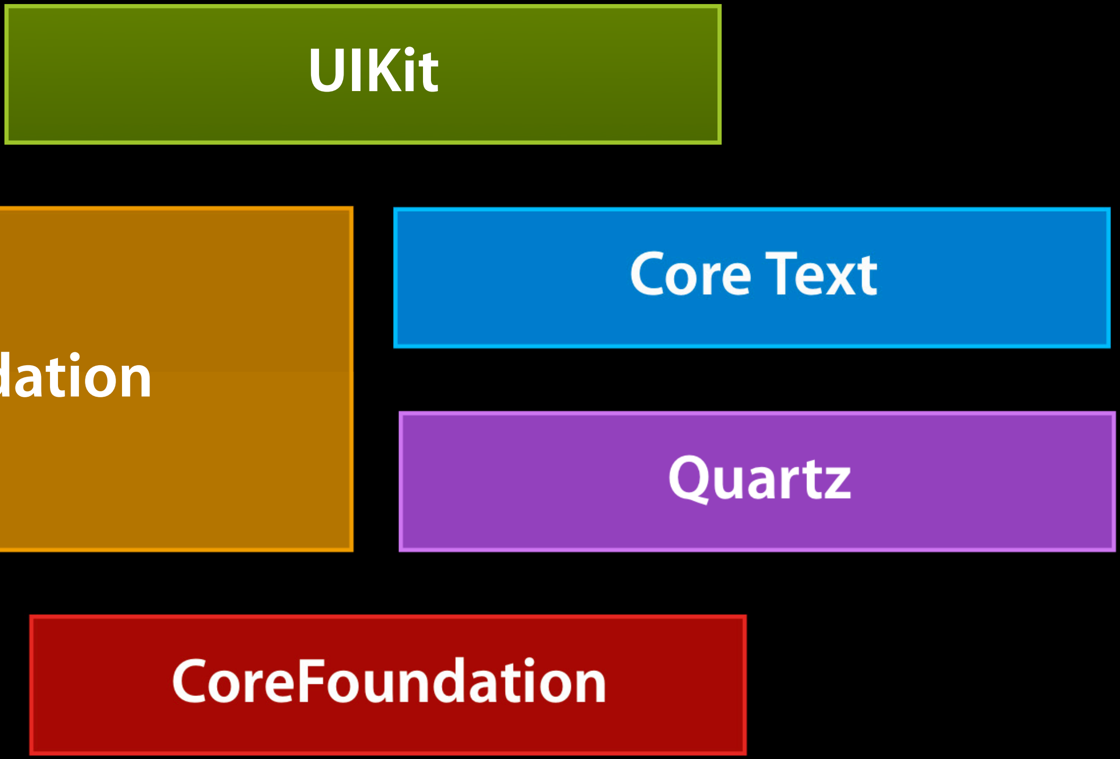
Introduction

- Basic text handling: use standard views and controls
- Some applications need more:
  - Detailed examination of text content
  - Specific fonts and font features
  - Customized text measuring and drawing

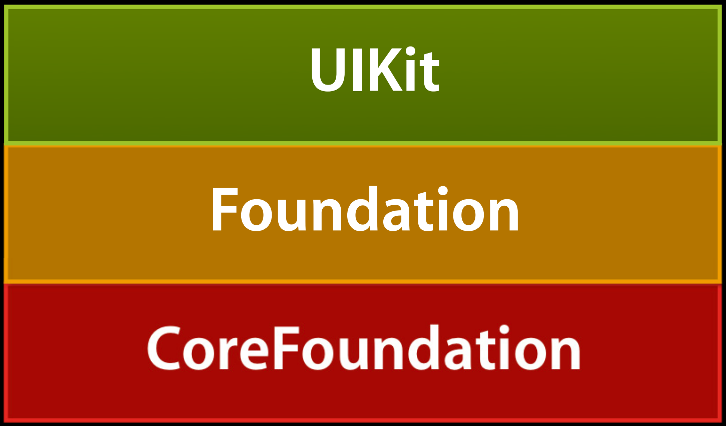
What You’ll Learn

- String objects and their components
- Iteration, matching, searching
- Regular expressions, Data Detectors, and spellchecking
- Font handling
- Text layout and drawing

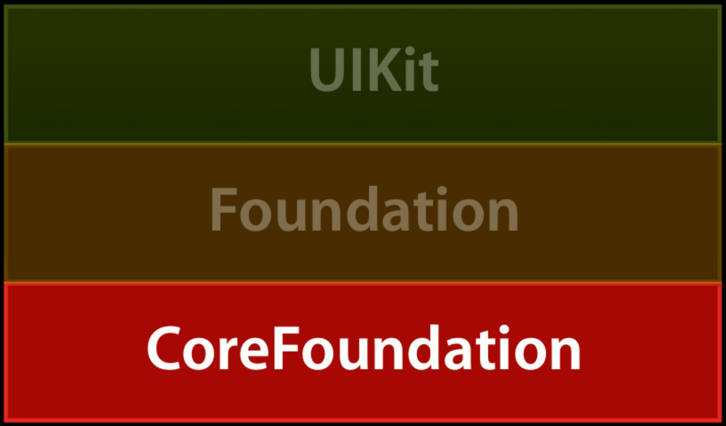
Text Architecture

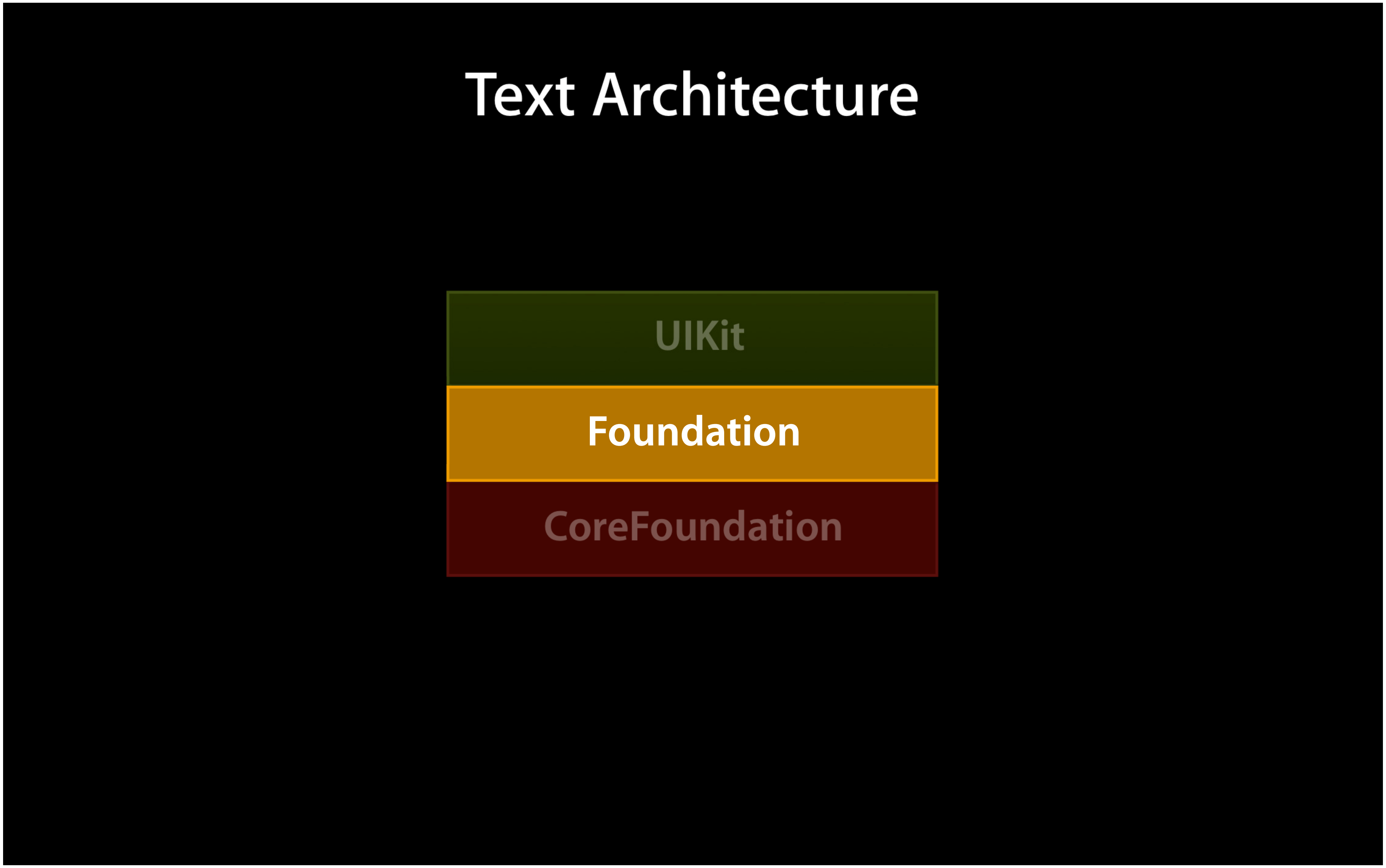


Text Architecture

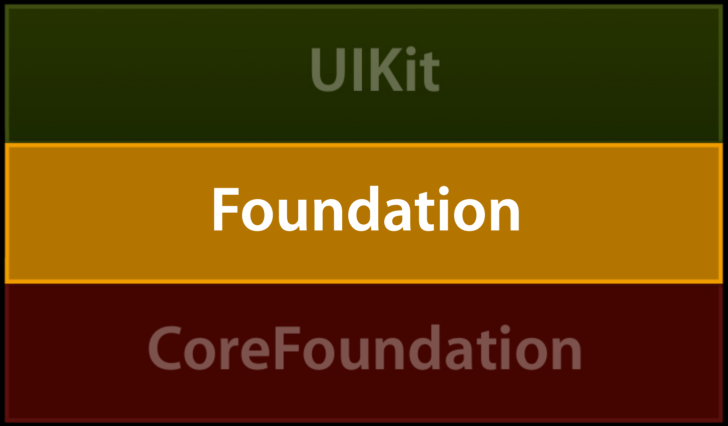


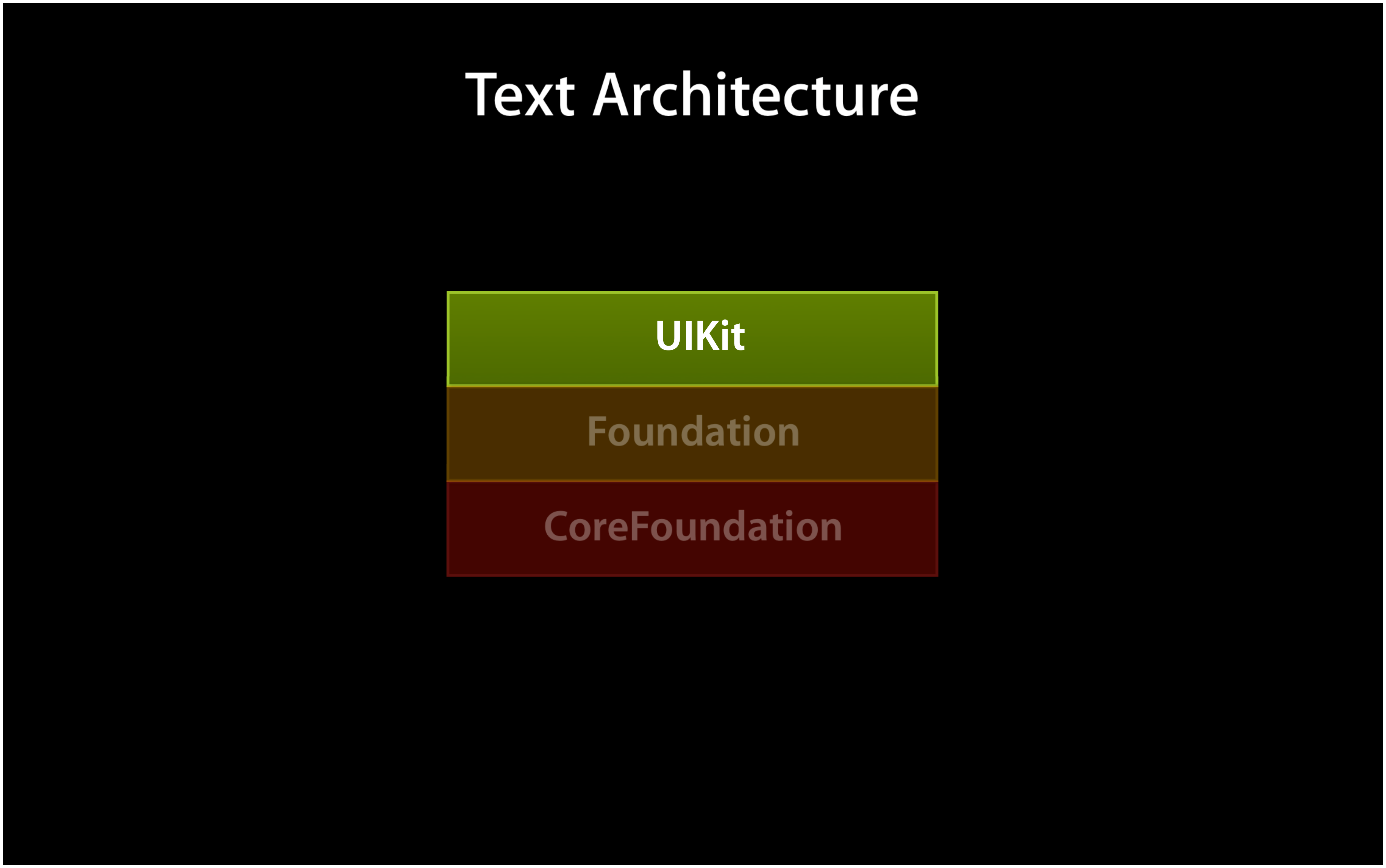
Text Architecture



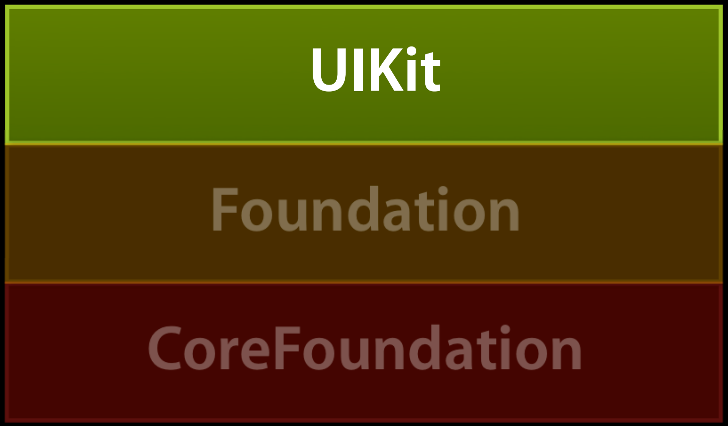


Text Architecture





Text Architecture



String Objects

NSString

- The fundamental string object on Mac OS X and iPhone OS
- Full-fledged string manipulation API
- Simple yet powerful
- Encapsulates Unicode™

String Classes

NSString

String Classes

NSString

CFString

String Classes

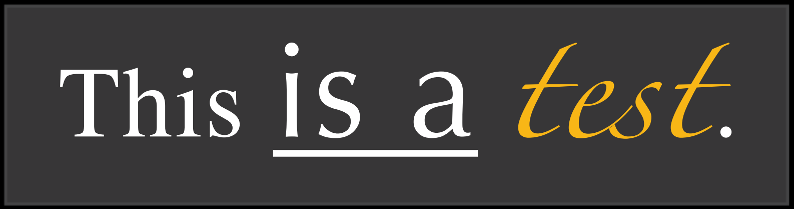
NSString

NSMutableString

CFString

CFMutableString

Attributed Strings


This is a test.

Times 48
White

Optima 64
White
Underlined

Zapfino 64
Yellow

Attributed String Classes

NSAttributedString

NSMutableAttributedString

CFAttributedString

CFMutableAttributedString

What’s in a String?

- Bytes?
- Characters?
- Code points?

Grapheme Clusters

- The smallest processing unit for NSString
- Queried with -rangeOfComposedCharacterSequenceAtIndex:
- Composition
  - é = e + ´
- Surrogate pair
  - 𠀋 U+2000B = 0xD840 + 0xDC0B

Words

- Appropriate processing unit for most transformation tasks
  - Letter-case mapping
  - Spell checking
- Whitespace is not necessarily the only way to break “words”
  - 正しい日本語です = 正しい + 日本語 + です
  - ภาษาไทย = ภาษา + ไทย

Paragraphs

- The maximum processing unit for all Unicode processing tasks
- Especially important for bidirectional languages like Arabic and Hebrew
- Each paragraph has an overall text flow direction
- Queried via -getParagraphStart:end:contentsEnd:forRange:

String Iteration

iOS 4

```
__block NSUInteger count = 0;

[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByWords
  usingBlock:^(NSString *word,
               NSRange wordRange,
               NSRange enclosingRange,
               BOOL *stop){

    // do something for each word

    if (++count >= 100) *stop = YES;

}];
```

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Iteration Types

- By clusters
- By words
- By sentences
- By lines
- By paragraphs

Matching

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSAnchoredSearch |
                 NSCaseInsensitiveSearch |
                 NSDiacriticInsensitiveSearch |
                 NSWidthInsensitiveSearch
          range:wordRange locale:locale];

if (matchRange.length == wordRange.length) {
  // the word matches
}
```

Searching

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSCaseInsensitiveSearch |
                 NSDiacriticInsensitiveSearch |
                 NSWidthInsensitiveSearch
          range:NSMakeRange(0, [string length])
          locale:locale];

if (matchRange.length > 0) {
  // found a match
}
```

Regular Expressions

Regular Expression Search

3.2

```
NSRange foundRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
          options:NSRegularExpressionSearch |
                  NSCaseInsensitiveSearch
          range:searchRange];

if (matchRange.length > 0) {
  // found a match
}
```

Regular Expression Matches

```
If into in onto of often on and ON.
```

Regular Expression Matches

```
If into in onto of often on and ON.
```

Search and Replace

iOS 4

```
NSString *modifiedString =
   [string stringByReplacingOccurrencesOfString:
     @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
     options:NSRegularExpressionSearch
     range:range];                 // immutable strings


 [mutableString replaceOccurrencesOfString:
     @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
     options:NSRegularExpressionSearch
     range:range];                 // mutable strings
```

Template Replacement

```
If into in onto of often on and ON.
```

Template Replacement

```
If into in onto of often on and ON.
```

$$\downarrow$$

```
fI into ni onto fo often no and NO.
```

NSRegularExpression

iOS 4

```
NSError *error = nil;

NSRegularExpression *regex =
 [NSRegularExpression
  regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
  options:NSRegularExpressionCaseInsensitive
  error:&error];
```

Regular Expression Iteration

```
__block NSUInteger count = 0;

[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    // do something for each match

    if (++count >= 100) *stop = YES;
}];
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Match Objects

- Objects of class NSTextCheckingResult
- @property NSTextCheckingType resultType;
- @property NSRange range;
  - This is the overall range
- - (NSRange)rangeAtIndex:(NSUInteger)idx;
  - These are the ranges of capture groups

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                         [match rangeAtIndex:1];
    NSRange secondHalfRange =
                         [match rangeAtIndex:2];

    // do something with these ranges
}];
```

Additional Methods

- -matchesInString:options:range:
- -numberOfMatchesInString:options:range:
- -firstMatchInString:options:range:
- -rangeOfFirstMatchInString:options:range:

```
NSRange matchRange =
  [regex rangeOfFirstMatchInString:string
           options:0 range:searchRange];

if (matchRange.length > 0) {
  // found a match
}
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings


 [regex replaceMatchesInString:mutableString
    options:0
    range:range
    withTemplate:@"$2$1"];    // mutable strings
```

Data Detectors

NSDataDetector

iOS 4

```
NSError *error = nil;

NSDataDetector *detector =
   [NSDataDetector
     dataDetectorWithTypes:
          (NSTextCheckingTypeLink |
                           NSTextCheckingTypePhoneNumber)
     error:&error];
```

Data Detector Matches

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Data Detector Matches

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Data Detector Types

- NSTextCheckingTypeDate
- NSTextCheckingTypeAddress
- NSTextCheckingTypeLink
- NSTextCheckingTypePhoneNumber
- NSTextCheckingTypeTransitInformation

Data Detector Iteration

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){

    // do something for each match

}];
```

Data Detector Iteration

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Data Detector Iteration

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Data Detector Iteration

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Getting Results

- More NSTextCheckingResult properties:
  - @property NSDate *date;
  - @property NSDictionary *components;
  - @property NSURL *URL;
  - @property NSString *phoneNumber;

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
                  NSMatchingFlags flags, BOOL *stop){

  NSTextCheckingType t = [match resultType];
  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Additional Methods

- -matchesInString:options:range:
- -numberOfMatchesInString:options:range:
- -firstMatchInString:options:range:
- -rangeOfFirstMatchInString:options:range:

```
NSRange matchRange =
  [detector rangeOfFirstMatchInString:string
            options:0 range:searchRange];

if (matchRange.length > 0) {
  // found a match
}
```

Spellchecking

UITextChecker

3.2

```
UITextChecker *checker =
                    [[UITextChecker alloc] init];

NSRange misspelledRange =
  [checker rangeOfMisspelledWordInString:string
    range:range startingAt:range.location
    wrap:NO language:@"en_US"];

NSArray *guesses =
  [checker guessesForWordRange:misspelledRange
    inString:string language:@"en_US"];
```

Demo

Advanced Text Handling

Core Text

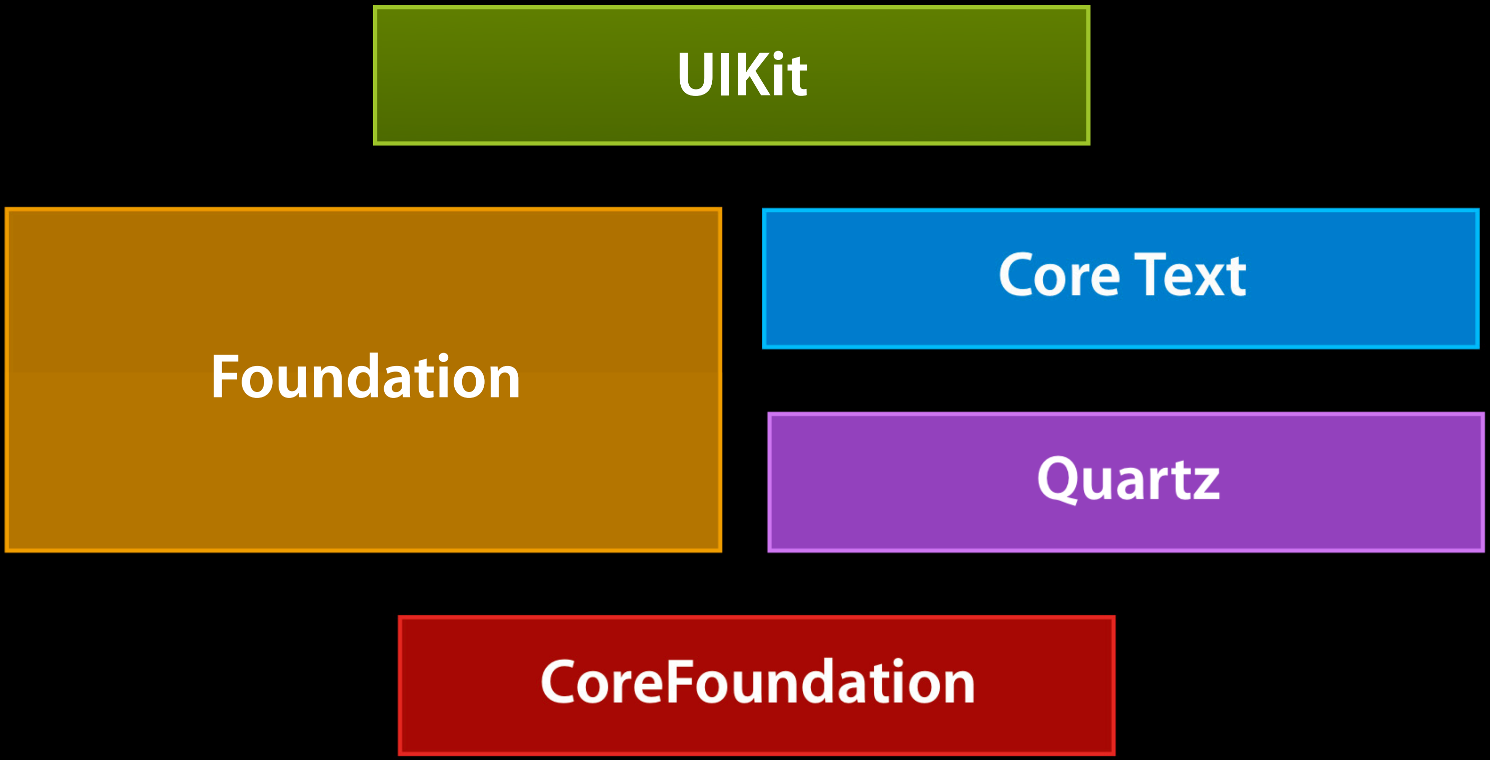
Julio González
Type Engineering Manager

Core Text Overview

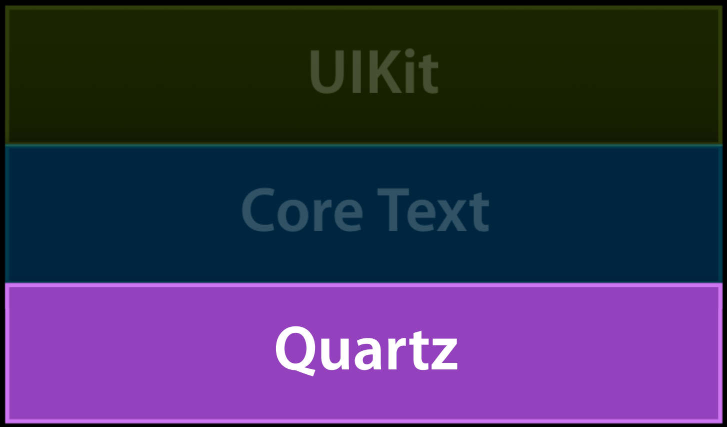
3.2

- Text and font architecture
- Core Text framework
- Principles review
- Differences from OS X

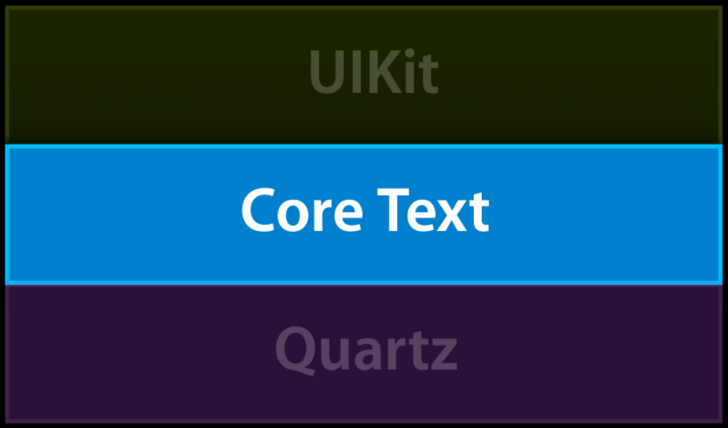
Text Architecture

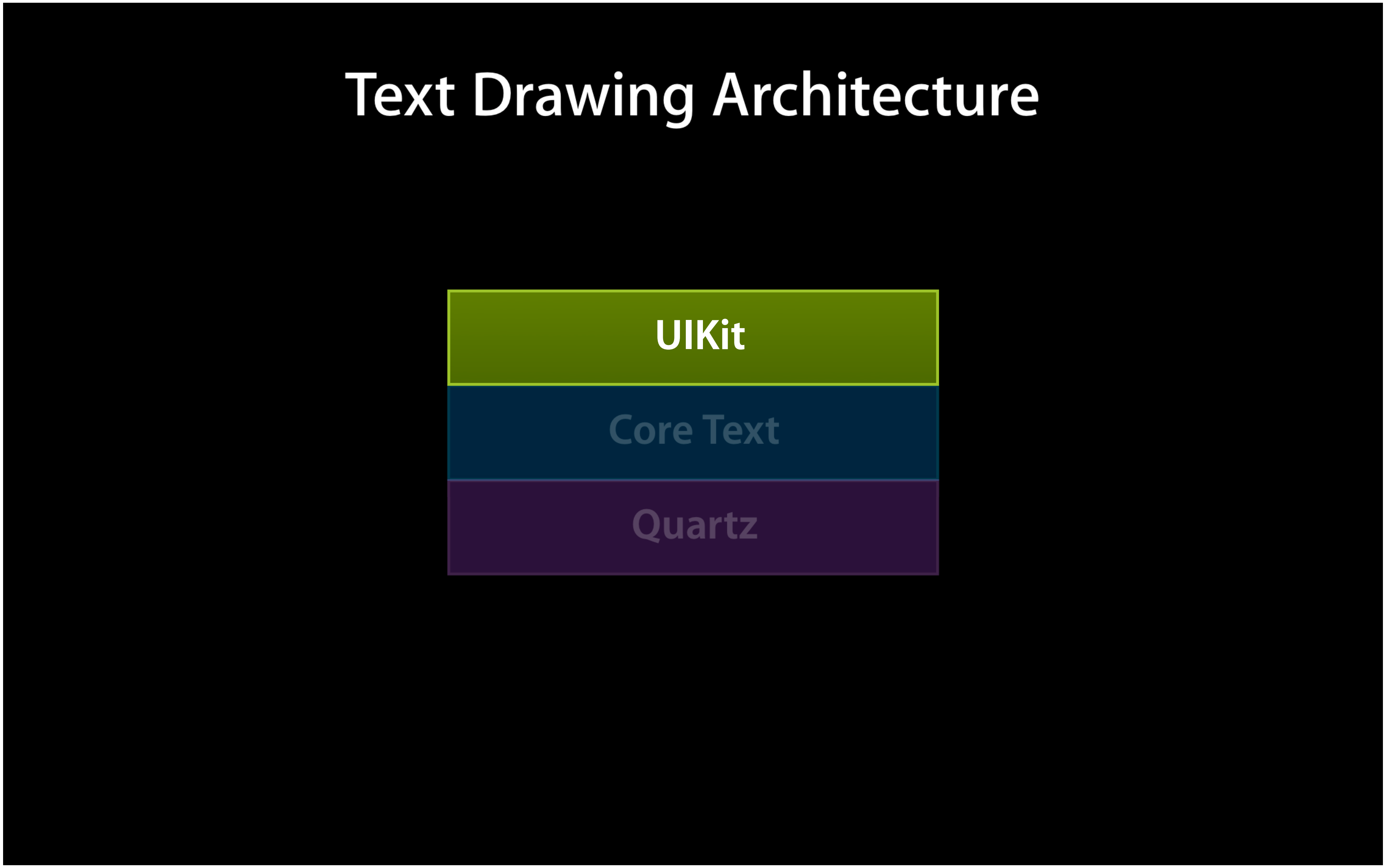


Text Drawing Architecture

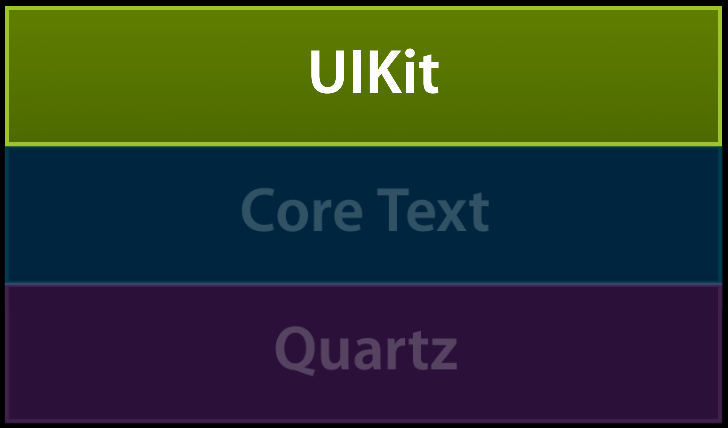


Text Drawing Architecture

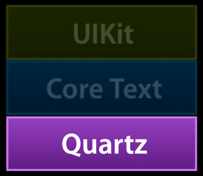




Text Drawing Architecture

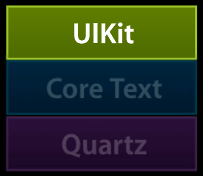


Quartz



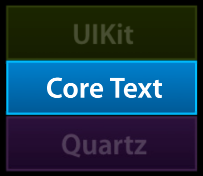
- Renders glyphs
- CGFontRef
- No Unicode™ text support
- No font substitution
- Usage
    - Specialized layout

UIKit



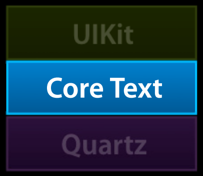
- Simple to use
- UIFont
- Obvious choice for presenting text
  - NSString
  - UILabel
  - UITextView
- Editable text
- Copy/paste support
- Lacks adjustments control

Core Text



- Mac OS X font and Unicode layout engine
- Performance and threading
- Font features
- Highly customizable

OS X Differences



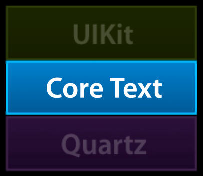
- No font management support
- OpenType font features and shaping disabled
- No vertical glyph support
- Simpler font matching mechanism

Font Types

UIKit
Core Text
Quartz

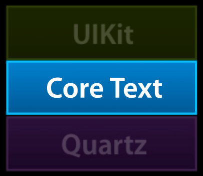
- CTFontRef
  - Postscript names preferred
  - Different from a UIFont
- CTFontDescriptor
  - Attribute matching
  - Persistent storage
- CTFontCollection
  - List of font descriptors

Font API Benefits



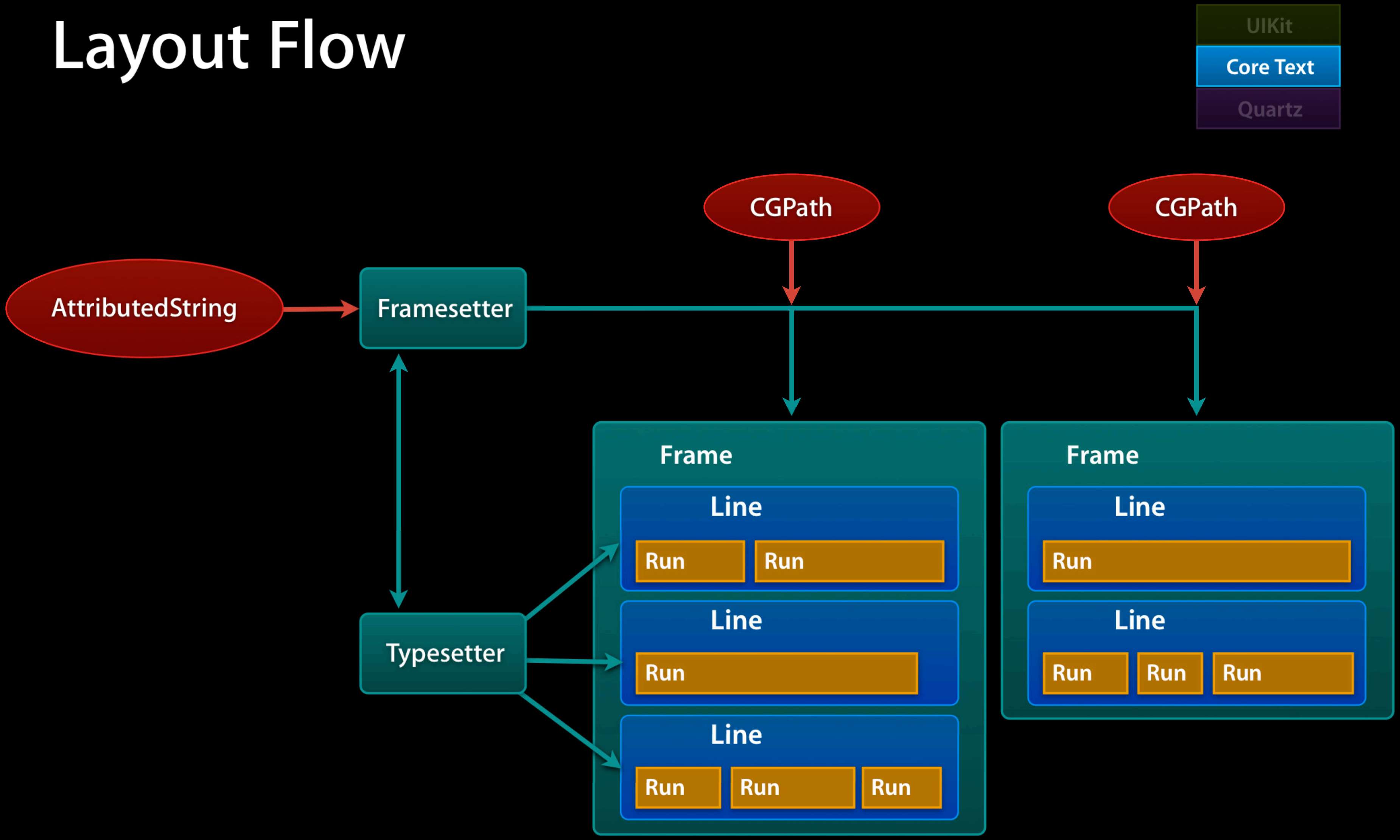
- Access to all font faces
- Better font substitution
- Typographic feature access
- Font abstraction APIs
    - Metrics
    - Character mapping
    - Glyph information

Layout Overview



- Flipped coordinates
- Attributed string
- Framesetter
- Typesetter
- Line
- Glyph run

Layout Flow


UIKit
Core Text
Quartz
CGPath
CGPath
AttributedString
Framesetter
Typesetter
Frame
Line
Run
Run
Line
Run
Line
Run
Run
Run
Frame
Line
Run
Line
Run
Run
Run

Font Creation

UIKit
Core Text
Quartz

```
CTFontRef sysUIFont = CTFontCreateUIFontForLanguage(
  kCTFontSystemFontType, 24.0, NULL /* language */  );


CTFontRef helveticaBold = CTFontCreateWithName(
  CFSTR("Helvetica-Bold"), 24.0, NULL /* matrix */ );

CTFontRef helveticaItalic = CTFontCreateCopyWithSymbolicTraits(
  helveticaBold, 24.0, NULL /* matrix */,
  kCTFontItalicTrait, kCTFontBoldTrait | kCTFontItalicTrait );
```

Attributed String Creation

UIKit
Core Text
Quartz

```
NSString* unicodeString =
 NSLocalizedString(@"TitleString", @"Window Title");

CGColorRef color = [UIColor blueColor].CGColor;

NSNumber* underline = [NSNumber numberWithInt:
  kCTUnderlineStyleSingle|kCTUnderlinePatternDot];

NSDictionary* attributesDict =
 [NSDictionary dictionaryWithObjectsAndKeys:
   helveticaBold, (NSString*)kCTFontAttributeName,
   color, (NSString*)kCTForegroundColorAttributeName,
   underline, (NSString*)kCTUnderlineStyleAttributeName,
   nil];


NSAttributedString* stringToDraw = [[NSAttributedString alloc]
  initWithString:unicodeString attributes:attributesDict];
```

Attributed String Creation

UIKit
Core Text
Quartz

```
NSString* unicodeString =
 NSLocalizedString(@"TitleString", @"Window Title");
CGColorRef color = [UIColor blueColor].CGColor;

NSNumber* underline = [NSNumber numberWithInt:
  kCTUnderlineStyleSingle|kCTUnderlinePatternDot];

NSDictionary* attributesDict =
 [NSDictionary dictionaryWithObjectsAndKeys:
   helveticaBold, (NSString*)kCTFontAttributeName,
   color, (NSString*)kCTForegroundColorAttributeName,
   underline, (NSString*)kCTUnderlineStyleAttributeName,
   nil];


NSAttributedString* stringToDraw = [[NSAttributedString alloc]
  initWithString:unicodeString attributes:attributesDict];
```

Attributed String Creation

UIKit
Core Text
Quartz

```
NSString* unicodeString =
 NSLocalizedString(@"TitleString", @"Window Title");

CGColorRef color = [UIColor blueColor].CGColor;

NSNumber* underline = [NSNumber numberWithInt:
  kCTUnderlineStyleSingle|kCTUnderlinePatternDot];

NSDictionary* attributesDict =
 [NSDictionary dictionaryWithObjectsAndKeys:
   helveticaBold, (NSString*)kCTFontAttributeName,
   color, (NSString*)kCTForegroundColorAttributeName,
   underline, (NSString*)kCTUnderlineStyleAttributeName,
   nil];


NSAttributedString* stringToDraw = [[NSAttributedString alloc]
  initWithString:unicodeString attributes:attributesDict];
```

Attributed String Creation

UIKit
Core Text
Quartz

```
NSString* unicodeString =
 NSLocalizedString(@"TitleString", @"Window Title");

CGColorRef color = [UIColor blueColor].CGColor;

NSNumber* underline = [NSNumber numberWithInt:
  kCTUnderlineStyleSingle|kCTUnderlinePatternDot];

NSDictionary* attributesDict =
 [NSDictionary dictionaryWithObjectsAndKeys:
   helveticaBold, (NSString*)kCTFontAttributeName,
   color, (NSString*)kCTForegroundColorAttributeName,
   underline, (NSString*)kCTUnderlineStyleAttributeName,
   nil];

NSAttributedString* stringToDraw = [[NSAttributedString alloc]
  initWithString:unicodeString attributes:attributesDict];
```

Attributed String Creation

UIKit
Core Text
Quartz

```
NSString* unicodeString =
 NSLocalizedString(@"TitleString", @"Window Title");

CGColorRef color = [UIColor blueColor].CGColor;

NSNumber* underline = [NSNumber numberWithInt:
  kCTUnderlineStyleSingle|kCTUnderlinePatternDot];

NSDictionary* attributesDict =
 [NSDictionary dictionaryWithObjectsAndKeys:
   helveticaBold, (NSString*)kCTFontAttributeName,
   color, (NSString*)kCTForegroundColorAttributeName,
   underline, (NSString*)kCTUnderlineStyleAttributeName,
   nil];

NSAttributedString* stringToDraw = [[NSAttributedString alloc]
  initWithString:unicodeString attributes:attributesDict];
```

Drawing Simple Labels

UIKit
Core Text
Quartz

```
// Prepare our view for drawing
CGContextSetTextMatrix(context, CGAffineTransformIdentity);
CGContextTranslateCTM(context, 0, ([self bounds]).size.height );
CGContextScaleCTM(context, 1.0, -1.0);

// Draw the Label
CTLineRef line = CTLineCreateWithAttributedString(stringToDraw);
CGContextSetTextPosition(context, x, y);
CTLineDraw(line, context);
```

Drawing Simple Labels

UIKit
Core Text
Quartz

```
// Prepare our view for drawing
CGContextSetTextMatrix(context, CGAffineTransformIdentity);
CGContextTranslateCTM(context, 0, ([self bounds]).size.height );
CGContextScaleCTM(context, 1.0, -1.0);

// Draw the Label
CTLineRef line = CTLineCreateWithAttributedString(stringToDraw);
CGContextSetTextPosition(context, x, y);
CTLineDraw(line, context);
```

Drawing Simple Labels

UIKit
Core Text
Quartz

```
// Prepare our view for drawing
CGContextSetTextMatrix(context, CGAffineTransformIdentity);
CGContextTranslateCTM(context, 0, ([self bounds]).size.height );
CGContextScaleCTM(context, 1.0, -1.0);
```

```
// Draw the Label
CTLineRef line = CTLineCreateWithAttributedString(stringToDraw);
CGContextSetTextPosition(context, x, y);
CTLineDraw(line, context);
```

Drawing Simple Paragraphs

UIKit
Core Text
Quartz

```
// Prepare our view for drawing

...

// Draw the paragraph

CTFramesetterRef framesetter =
  CTFramesetterCreateWithAttributedString(stringToDraw);

CGMutablePathRef path = CGPathCreateMutable();

CGPathAddRect(path, NULL, viewRect);

CTFrameRef frame =
  CTFramesetterCreateFrame(framesetter, CFRangeMake(0,0),
  path, NULL);

CTFrameDraw(frame, context);
```

Drawing Simple Paragraphs

UIKit
Core Text
Quartz

```
// Prepare our view for drawing

...

// Draw the paragraph

CTFramesetterRef framesetter =
  CTFramesetterCreateWithAttributedString(stringToDraw);

CGMutablePathRef path = CGPathCreateMutable();

CGPathAddRect(path, NULL, viewRect);

CTFrameRef frame =
  CTFramesetterCreateFrame(framesetter, CFRangeMake(0,0),
  path, NULL);

CTFrameDraw(frame, context);
```

Drawing Simple Paragraphs

UIKit
Core Text
Quartz

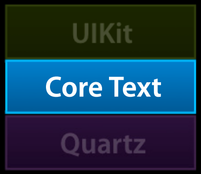
```
// Prepare our view for drawing

...

// Draw the paragraph

CTFramesetterRef framesetter =
  CTFramesetterCreateWithAttributedString(stringToDraw);

CGMutablePathRef path = CGPathCreateMutable();

CGPathAddRect(path, NULL, viewRect);

CTFrameRef frame =
  CTFramesetterCreateFrame(framesetter, CFRangeMake(0,0),
  path, NULL);

CTFrameDraw(frame, context);
```

Drawing Simple Paragraphs

UIKit
Core Text
Quartz

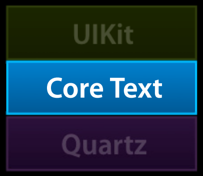
```
// Prepare our view for drawing

...

// Draw the paragraph

CTFramesetterRef framesetter =
  CTFramesetterCreateWithAttributedString(stringToDraw);

CGMutablePathRef path = CGPathCreateMutable();

CGPathAddRect(path, NULL, viewRect);

CTFrameRef frame =
  CTFramesetterCreateFrame(framesetter, CFRangeMake(0,0),
  path, NULL);

CTFrameDraw(frame, context);
```

Other Objects



- ParagraphStyle
- TabStops
- GlyphInfo
- RunDelegate
  - Very low level
  - Only in iPhone OS

Layout API Benefits



- Multiple styles and feature support
- Fine control over text layout
  - Line breaking
  - Paragraph primitives
  - Glyph substitution
- Thread safe

Demo

Summary

UIKit
Core Text
Quartz

- Text analysis based on ranges within NSStrings
- Use NSString, NSRegularExpression, and NSDataDetector APIs
- Core Text bridges the layout gap
- Gives you greater access to fonts
- Remember functionality trade-offs

Related Sessions

What’s New in Foundation for iOS 4	Pacific Heights Tuesday 10:15AM
Advanced Cocoa Text Tips and Tricks	Russian Hill Wednesday 9:00AM
Understanding Foundation	Russian Hill Thursday 9:00AM
Internationalizing Data on Mac and iPhone	Russian Hill Thursday 10:15AM

Labs

Core Text Lab	Application Frameworks Lab D Wednesday 11:30AM
iPhone Text Lab	Application Frameworks Lab C Wednesday 11:30AM
Internationalization Lab	Application Frameworks Lab C Thursday 11:30AM

More Information

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

Documentation
String Programming Guide for Cocoa
http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/Strings/introStrings.html

Core Text Programming Guide
http://developer.apple.com/mac/library/documentation/Carbon/Conceptual/CoreText_Programming/Introduction/Introduction.html

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

Q&A

WWDC10