

Introduction to Attributed Strings for iOS

Drawing strings with dramatic expression

Session 222

Aki “I 🐉 Unicode” Inoue

Cocoa Engineer

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

Multi-style String Drawing in UIKit

Multi-style String Drawing in UIKit

You can take it off the wishlist now!

NSAttributedString in UIKit

Agenda

- Attributed string essentials
- Drawing with basic attributes
- UIKit adoption

Agenda

- Attributed string essentials
- Drawing with basic attributes
- UIKit adoption

Agenda

- Attributed string essentials
- Drawing with basic attributes
- UIKit adoption

Agenda

- Attributed string essentials
- Drawing with basic attributes
- UIKit adoption

Attributed String Essentials

Displaying Strings

Using `UIStringDrawing`

I'm a string

Displaying Strings

Using `UIStringDrawing`

@`"I'm a string"`

`NSString`



Displaying Strings

Using `UIStringDrawing`

I'm a string

Displaying Strings

Using `UIStringDrawing`

I'm a string



Helvetica Neue

Displaying Strings

Using `UIStringDrawing`

I'm a string



Helvetica Neue Bold

Displaying Strings

Using `UIStringDrawing`

I'm a string

Helvetica Neue Bold

Red

What Is an Attributed String ?

Associating attributes to characters

I'm an attributed string

What Is an Attributed String ?

Associating attributes to characters

I'm an attributed string

Helvetica Neue

A red rectangular box containing the text "Helvetica Neue" is positioned below the text "I'm an attributed string". A thin red vertical line with an upward-pointing arrowhead connects the top of the box to the center of the text above it.

What Is an Attributed String ?

Associating attributes to characters

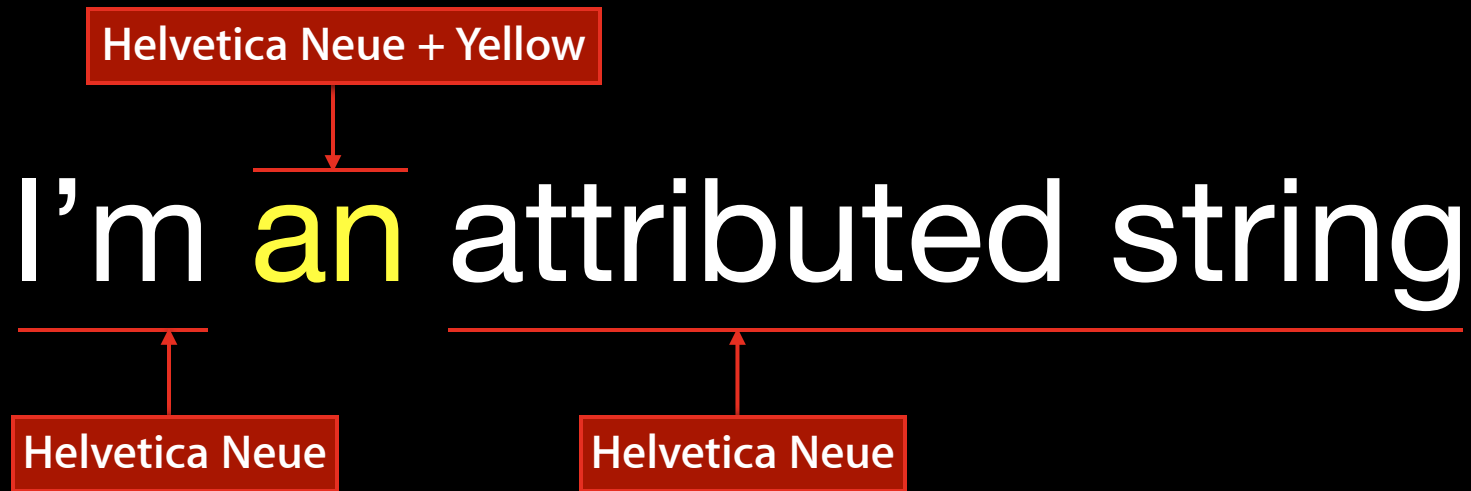
I'm an attributed string

Helvetica Neue

A diagram illustrating an attributed string. A red horizontal line is drawn under the text "I'm an attributed string". From the center of this line, a red arrow points upwards to a red rectangular box containing the text "Helvetica Neue". This indicates that the font attribute "Helvetica Neue" is applied to the characters of the string above it.

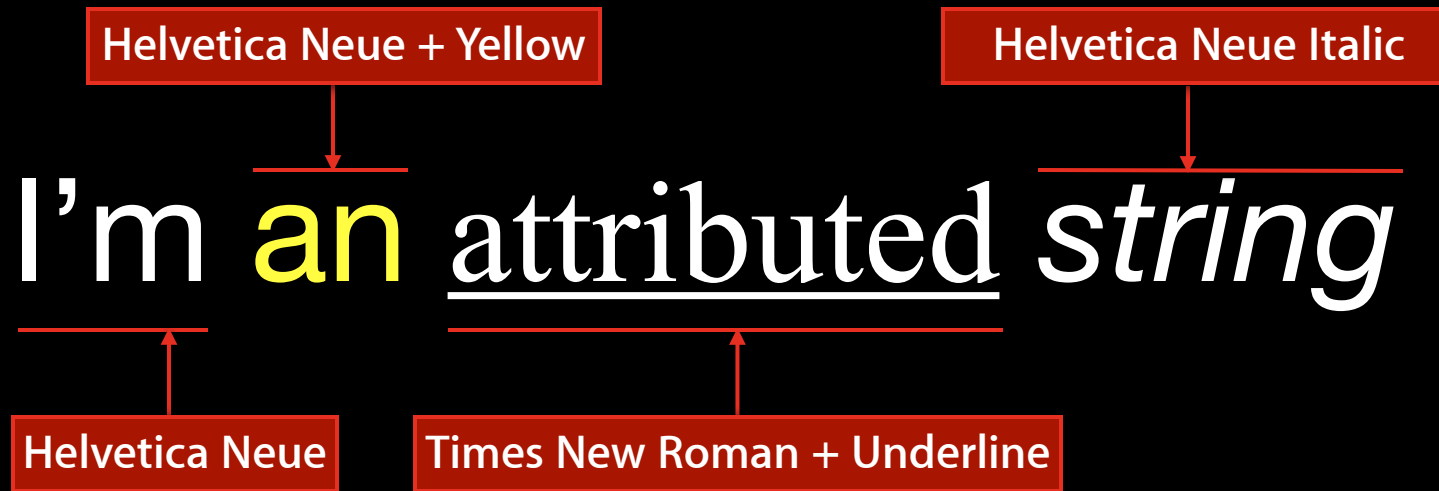
What Is an Attributed String ?

Associating attributes to characters



What Is an Attributed String ?

Associating attributes to characters



What Can You Do with It?

Multi-style text

NASDAQ	2,844.72	+ 66.61
S&P 500	1,315.13	+ 29.63
AAPL	571.46	+ 8.63

Wall Street climbs two percent on talk of Spain solution Stocks jumped on Wednesday, giving...
Reuters - 6/6/12 at 2:14 PM

Stocks Soar More Than 2%, Finish at Session Highs Expectations that more stimulus is on th...
TheStreet - 6/6/12 at 2:02 PM

An Apple Engineer Abducted by An Alien Spacecraft A spacecraft landed at the Cupert...
TheNews - 6/13/12 at 3:18 PM

What Can You Do with It?

Multi-style text

NASDAQ	2,844.72	+ 66.61
S&P 500	1,315.13	+ 29.63
AAPL	571.46	+ 8.63

An Apple Engineer Abducted by An Alien
Spacecraft A spacecraft landed at the Cupert...
TheNews - 6/13/12 at 3:18 PM

An Apple Engineer Abducted by An Alien
Spacecraft A spacecraft landed at the Cuperti...
TheNews - 6/13/12 at 3:18 PM

What Can You Do with It?

Multi-style text

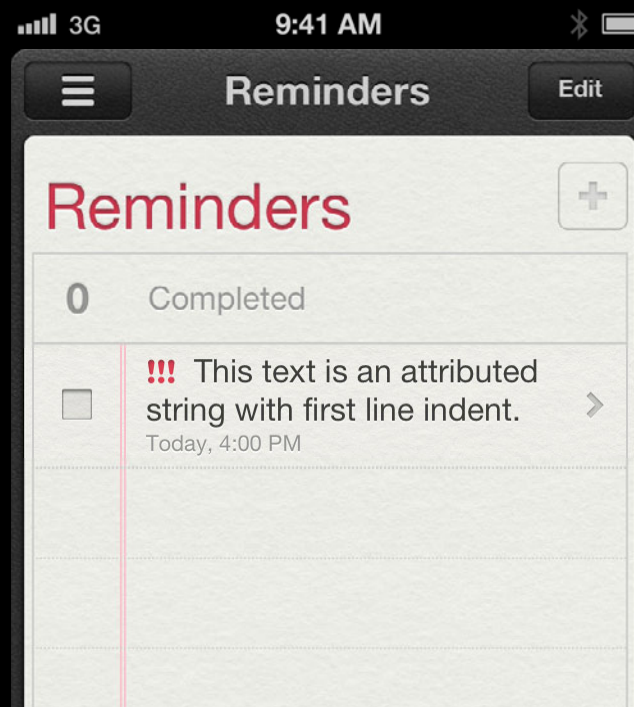
NASDAQ	2,844.72	+ 66.61
S&P 500	1,315.13	+ 29.63
AAPL	571.46	+ 8.63

**An Apple Engineer Abducted by An Alien
Spacecraft** A spacecraft landed at the Cupert...
TheNews - 6/13/12 at 3:18 PM

**An Apple Engineer Abducted by An Alien
Spacecraft** A spacecraft landed at the Cuperti...
TheNews - 6/13/12 at 3:18 PM

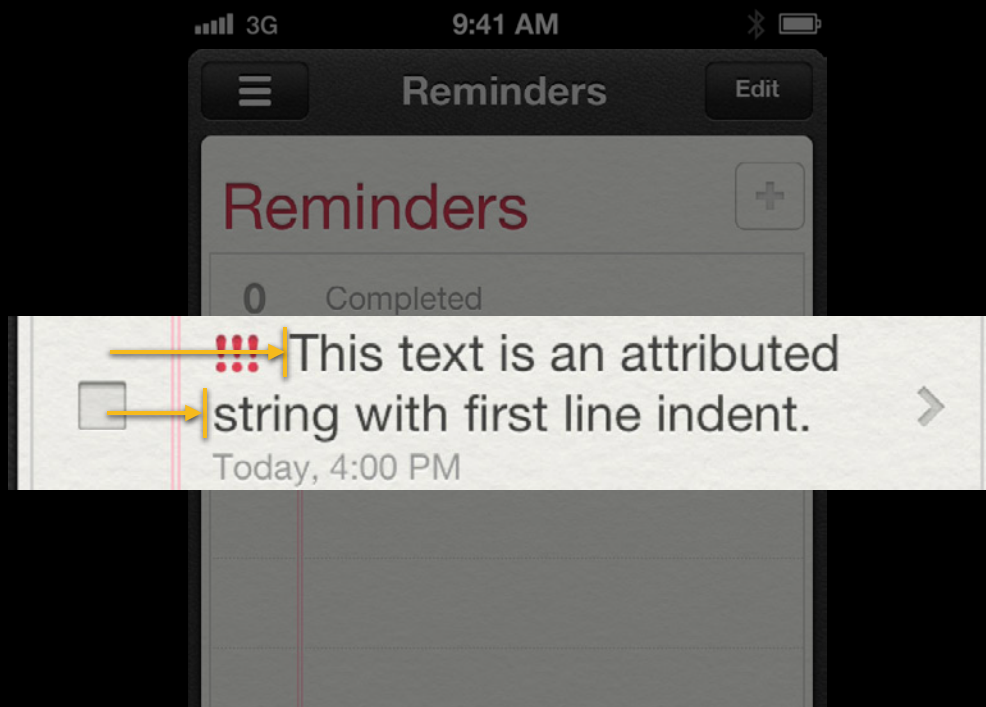
What Can You Do with It?

More expressive layout options



What Can You Do with It?

More expressive layout options



What Can You Do with It?

Graphics effects as text styles

iPad

9:41 AM



Graphics effects right at your
fingertips.

NSAttributedString

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer) range;

NSAttributedString

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer) range;

NSAttributedString

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;
- Instantiating with a string and attributes dictionary
 - (id)initWithString:(NSString *)string attributes:(NSDictionary *)attrs;

NSAttributedString

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;
- Instantiating with a string and attributes dictionary
 - (id)initWithString:(NSString *)string attributes:(NSDictionary *)attrs;

Instantiating an Attributed String

Specifying a font



- Using NSFontAttributeName

```
NSAttributedString *string = [[NSAttributedString alloc]
    initWithString:@"Hello World!!"
    attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];
```

Instantiating an Attributed String

Specifying a font



- Using NSFontAttributeName

```
NSAttributedString *string = [[NSAttributedString alloc]
    initWithString:@"Hello World!!"
    attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];
```

Hello World!!

NSAttributedString

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;
- Instantiating with a string and attributes dictionary
 - (id)initWithString:(NSString *)string attributes:(NSDictionary *)attrs;

NSAttributedString

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;
- Instantiating with a string and attributes dictionary
 - (id)initWithString:(NSString *)string attributes:(NSDictionary *)attrs;
- No attribute for an empty string

NSMutableAttributedString

- Changing attributes
- Modifying string contents

NSMutableAttributedString

- Changing attributes
- Modifying string contents

NSMutableAttributedString

Changing attributes

- Mutable methods for attributes

- `(void)setAttributes:(NSDictionary *)attrs range:(NSRange)range;`

NSMutableAttributedString

Changing attributes

- Mutable methods for attributes

- `(void)setAttributes:(NSDictionary *)attrs range:(NSRange)range;`
- `(void)addAttributes:(NSDictionary *)attrs range:(NSRange)range;`
- `(void)addAttribute:(NSString *)key value:(id)value range:(NSRange)range;`
- `(void)removeAttribute:(NSString *)key range:(NSRange)range;`

NSMutableAttributedString

Changing attributes

- Mutable methods for attributes

- `(void)setAttributes:(NSDictionary *)attrs range:(NSRange)range;`
- `(void)addAttributes:(NSDictionary *)attrs range:(NSRange)range;`
- `(void)addAttribute:(NSString *)key value:(id)value range:(NSRange)range;`
- `(void)removeAttribute:(NSString *)key range:(NSRange)range;`

- Modifying existing attributes

- Creating multi-style string

Creating Multi-style String

Emphasizing a range



- Using NSForegroundColorAttributeName

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]
    initWithString:@"Dentist at 9:15am."
    attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];

[string addAttribute:NSForegroundColorAttributeName
    value:[UIColor redColor]
    range:NSMakeRange(11, 6)]; // Change the time, 9:15am, to red
```


Creating Multi-style String

Emphasizing a range



- Using NSForegroundColorAttributeName

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]
initWithString:@"Dentist at 9:15am."
attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];
```

```
[string addAttribute:NSForegroundColorAttributeName
value:[UIColor redColor]
range:NSMakeRange(11, 6)]; // Change the time, 9:15am, to red
```

Dentist at 9:15am.

Creating Multi-style String

Emphasizing a range



- Using NSForegroundColorAttributeName

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]
initWithString:@"Dentist at 9:15am."
attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];
```

```
[string addAttribute:NSForegroundColorAttributeName
value:[UIColor redColor]
range:NSMakeRange(11, 6)]; // Change the time, 9:15am, to red
```

Dentist at 9:15am.

Creating Multi-style String

Emphasizing a range



- Using NSForegroundColorAttributeName

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]
initWithString:@"Dentist at 9:15am."
attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];
```

```
[string addAttribute:NSForegroundColorAttributeName
value:[UIColor redColor]
range:NSMakeRange(11, 6)]; // Change the time, 9:15am, to red
```

Dentist at 9:15am.

NSMutableAttributedString

- Changing attributes
- **Modifying string contents**

NSMutableAttributedString

Modifying string contents

- Mutable methods for string contents
 - `(void)replaceCharactersInRange:(NSRange)range withString:(NSString *)str;`

NSMutableAttributedString

Modifying string contents

- Mutable methods for string contents
 - (void)replaceCharactersInRange:(NSRange)range withString:(NSString *)str;
 - (void)replaceCharactersInRange:(NSRange)range
withAttributedString:(NSAttributedString *)attributedString;
 - (void)insertAttributedString:(NSAttributedString *)attributedString
atIndex:(NSUInteger)location;
 - (void)appendAttributedString:(NSAttributedString *)attributedString;
 - (void)deleteCharactersInRange:(NSRange)range;

NSMutableAttributedString

Modifying string contents

- Mutable methods for string contents

- (void)replaceCharactersInRange:(NSRange)range withString:(NSString *)str;
- (void)replaceCharactersInRange:(NSRange)range
withAttributedString:(NSAttributedString *)attributedString;
- (void)insertAttributedString:(NSAttributedString *)attributedString
atIndex:(NSUInteger)location;
- (void)appendAttributedString:(NSAttributedString *)attributedString;
- (void)deleteCharactersInRange:(NSRange)range;

NSMutableAttributedString

Modifying string contents

- Mutable methods for string contents

- `(void)replaceCharactersInRange:(NSRange)range withString:(NSString *)str;`
- `(void)replaceCharactersInRange:(NSRange)range
withAttributedString:(NSAttributedString *)attributedString;`
- `(void)insertAttributedString:(NSAttributedString *)attributedString
atIndex:(NSUInteger)location;`
- `(void)appendAttributedString:(NSAttributedString *)attributedString;`
- `(void)deleteCharactersInRange:(NSRange)range;`

Modifying String Contents

- Moving the appointment time 1 hour early

```
NSMutableAttributedString *string; // a mutable string
```

```
[string replaceCharactersInRange:NSMakeRange(11, 6) withString:@"8:15am"];
```

Modifying String Contents

- Moving the appointment time 1 hour early

```
NSMutableAttributedString *string; // a mutable string
```

```
[string replaceCharactersInRange:NSMakeRange(11, 6) withString:@"8:15am"];
```

Dentist at 9:15am.

Modifying String Contents

- Moving the appointment time 1 hour early

```
NSMutableAttributedString *string; // a mutable string
```

```
[string replaceCharactersInRange:NSMakeRange(11, 6) withString:@"8:15am"];
```

Dentist at 8:15am.

Modifying String Contents

- Moving the appointment time 1 hour early

```
NSMutableAttributedString *string; // a mutable string
```

```
[string replaceCharactersInRange:NSMakeRange(11, 6) withString:@"8:15am"];
```

Dentist at 8:15am.

Modifying String Contents

- Moving the appointment time 1 hour early

```
NSMutableAttributedString *string; // a mutable string
```

```
[string replaceCharactersInRange:NSMakeRange(11, 6) withString:@"8:15am"];
```

Dentist at 8:15am.

Modifying String Contents

- Moving the appointment time 1 hour early

```
NSMutableAttributedString *string; // a mutable string
```

```
[string replaceCharactersInRange:NSMakeRange(11, 6) withString:@"8:15am"];
```

Dentist at 8:15am.

Attributes Preserved

Attributes Preserved

- Attributes are preserved during string content replacement

Attributes Preserved

- Attributes are preserved during string content replacement
- Three rules to remember

Attributes Preserved

- Attributes are preserved during string content replacement
- Three rules to remember
 1. When replacing, the new characters inherit attributes from the **first character** of the range being replaced

Attributes Preserved

- Attributes are preserved during string content replacement
- Three rules to remember
 1. When replacing, the new characters inherit attributes from the first character of the range being replaced
 2. When inserting, the incoming string inherit from **the previous character**

Attributes Preserved

- Attributes are preserved during string content replacement
- Three rules to remember
 1. When replacing, the new characters inherit attributes from the first character of the range being replaced
 2. When inserting, the incoming string inherit from the previous character
 3. When inserting at the beginning, the attributes at **index 0** are used

Attributes Preserved

- Attributes are preserved during string content replacement
- Three rules to remember
 1. When replacing, the new characters inherit attributes from the first character of the range being replaced
 2. When inserting, the incoming string inherit from the previous character
 3. When inserting at the beginning, the attributes at index 0 are used

NOTE: Remember the “no character, no attributes” rule

Modifying String Contents

Replacement rule samples

- Insertion

```
[string replaceCharactersInRange:NSMakeRange(3, 0) withString:@"ABC"];
```

abcdefghi

Modifying String Contents

Replacement rule samples

- Insertion

```
[string replaceCharactersInRange:NSMakeRange(3, 0) withString:@"ABC"];
```

ABC
↓
abcdefghi

Modifying String Contents

Replacement rule samples

- Insertion

```
[string replaceCharactersInRange:NSMakeRange(3, 0) withString:@"ABC"];
```

abcABCdefghi

Modifying String Contents

Replacement rule samples

- Insertion at the beginning

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

abcdefghi

Modifying String Contents

Replacement rule samples

- Insertion at the beginning

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

ABC
↓
abcdefghi

Modifying String Contents

Replacement rule samples

- Insertion at the beginning

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

ABCabcdefghi

Modifying String Contents

Replacement rule samples

- Cross-run replacement

```
[string replaceCharactersInRange:NSMakeRange(2, 3) withString:@"ABC"];
```

abcdefghi

Modifying String Contents

Replacement rule samples

- Cross-run replacement

```
[string replaceCharactersInRange:NSMakeRange(2, 3) withString:@"ABC"];
```

ABC
↓
abcdefghi

Modifying String Contents

Replacement rule samples

- Cross-run replacement

```
[string replaceCharactersInRange:NSMakeRange(2, 3) withString:@"ABC"];
```

abABCfghi

Modifying String Contents

Replacement rule samples

- No character, no attributes

```
[string deleteCharactersInRange:NSMakeRange(0, 9)];
```

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

abcdefghi

Modifying String Contents

Replacement rule samples

- No character, no attributes

```
[string deleteCharactersInRange:NSMakeRange(0, 9)];
```

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

abcdefghi

Modifying String Contents

Replacement rule samples

- No character, no attributes

```
[string deleteCharactersInRange:NSMakeRange(0, 9)];
```

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

Modifying String Contents

Replacement rule samples

- No character, no attributes

```
[string deleteCharactersInRange:NSMakeRange(0, 9)];
```

```
[string replaceCharactersInRange:NSMakeRange(0, 0) withString:@"ABC"];
```

ABC

Querying Attributes

- An attribute dictionary per character
 - (NSString *)string
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

Querying Attributes

- An attribute dictionary per character

- (NSString *)string

- (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

Querying Attributes

- An attribute dictionary per character

- (NSString *)string

- (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

Querying Attributes

- An attribute dictionary per character

- (NSString *)string

- (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

Not guaranteed to return the maximum range

Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

abcdefghi

Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```

abcdefghi

Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```

abc**de**fg*hi*



Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```

abc**de**fg*hi*



Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```

abcdefghi



Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```

The diagram shows the string "abcdefghi" where each character is a different color: 'a' is red, 'b' is orange, 'c' is yellow, 'd' is green, 'e' is blue, 'f' is purple, 'g' is pink, 'h' is light blue, and 'i' is light green. A yellow arrow points down to the character 'd'. A label "{3, 3}" is positioned above the arrow, with a thin yellow arrow pointing from the label to the character 'd'. Vertical lines are drawn between 'c' and 'd', and between 'f' and 'g'.

Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```



Querying Attributes

- An attribute dictionary per character
 - (NSDictionary *)attributesAtIndex:(NSUInteger)location
effectiveRange:(NSRangePointer)range;

```
NSRange range;  
NSDictionary *dict = [string attributesAtIndex:4 effectiveRange:&range];
```

abcdefghi

{4, 1}

Getting Contiguous Run Range

Using longest effective range methods

- Variants of attribute query methods

- (NSDictionary *)attributesAtIndex:(NSUInteger)location
longestEffectiveRange:(NSRangePointer)range
inRange:(NSRange)maximumRange;

- (id)attribute:(NSString)attribute
atIndex:(NSUInteger)location
longestEffectiveRange:(NSRangePointer)range
inRange:(NSRange)maximumRange;

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes

abcdefghi

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes

abcde*fg*hi

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes

abc**de***fg**hi*

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes



*Helvetica Neue
Red*

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes

abcde*fghi*



Helvetica Neue
Blue

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes

abcde*fg*ghi

Helvetica Neue Italic
Blue

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for attributes

abcde*fg*ghi



Helvetica Neue Italic
Green

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for NSAttributedString

abcde*fg*ghi

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for NSFontAttributeName



The image displays the string "abcdefghi" on a black background. The characters are grouped into three contiguous runs: "abc" in red, "def" in blue, and "ghi" in green. A yellow bracket is drawn under the "abc" run. An arrow points from the text "Helvetica Neue" below to the character "c" within the "abc" run.

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for NSFontAttributeName

abcdefghi

Helvetica Neue Italic

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for `NSForegroundColorAttributeName`

abc**def***ghi*

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for `NSForegroundColorAttributeName`



The string "abcdefghi" is displayed with three contiguous color ranges highlighted by yellow brackets and arrows. The first range, "abc", is red. The second range, "def", is blue. The third range, "ghi", is green. An arrow points from the word "Red" below to the "abc" range.

Red

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for NSForegroundColorAttributeName

abcdefghi

Blue

The image illustrates the concept of contiguous run ranges for the string "abcdefghi". The characters are grouped into three color-coded runs: "abc" in red, "def" in blue, and "ghi" in green. A yellow bracket is drawn under the blue run "def", with an arrow pointing to the word "Blue" below it, indicating that this is the longest contiguous range of characters sharing the same foreground color.

Getting Contiguous Run Range

Using longest effective range methods

- Getting the contiguous range for NSForegroundColorAttributeName

abcde*fg*ghi



The image shows the text "abcde" in a red font, "fg" in a blue font, and "ghi" in a green font. A yellow bracket is drawn under the "fg" characters, with a vertical arrow pointing up to the word "Green" written below it. This illustrates that "fg" is a contiguous range of the blue color.

Querying Attributes

Querying Attributes

- Enumeration methods

- enumerateAttributesInRange:options:usingBlock:

- enumerateAttribute:inRange:options:usingBlock:

Querying Attributes

- Enumeration methods
 - `enumerateAttributesInRange:options:usingBlock:`
 - `enumerateAttribute:inRange:options:usingBlock:`
- Changing attributes based on existing attributes

Querying Attributes

- Enumeration methods

 - `enumerateAttributesInRange:options:usingBlock:`

 - `enumerateAttribute:inRange:options:usingBlock:`

- Changing attributes based on existing attributes

- Modifying string contents while preserving the attributes

Querying Attributes

- Enumeration methods

- enumerateAttributesInRange:options:usingBlock:
 - enumerateAttribute:inRange:options:usingBlock:

- Changing attributes based on existing attributes

- **Modifying string contents while preserving the attributes**

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
    NSString *string = [[attrString string] substringWithRange:range];

    [attrString replaceCharactersInRange:range
                      withString:[string uppercaseStringWithLocale:locale]];
}];
```

*abc*defghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;  
NSLocale *locale = [NSLocale currentLocale];  
  
[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])  
options:0  
usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {  
    NSString *string = [[attrString string] substringWithRange:range];  
  
    [attrString replaceCharactersInRange:range  
withString:[string uppercaseStringWithLocale:locale]];  
}];
```

abcdefghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;  
NSLocale *locale = [NSLocale currentLocale];  
  
[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])  
options:0  
usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {  
    NSString *string = [[attrString string] substringWithRange:range];  
  
    [attrString replaceCharactersInRange:range  
withString:[string uppercaseStringWithLocale:locale]];  
};
```

*abc*defghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;  
NSLocale *locale = [NSLocale currentLocale];  
  
[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])  
  options:0  
  usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {  
    NSString *string = [[attrString string] substringWithRange:range];  
  
    [attrString replaceCharactersInRange:range  
                  withString:[string uppercaseStringWithLocale:locale]];  
  }];
```

abc**def**ghi

Enumerating for Modification

Uppercase while preserving attributes



```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
     NSString *string = [[attrString string] substringWithRange:range];

     [attrString replaceCharactersInRange:range
      withString:[string uppercaseStringWithLocale:locale]];
}];
```

*abc*defghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
    NSString *string = [[attrString string] substringWithRange:range];

    [attrString replaceCharactersInRange:range
                      withString:[string uppercaseStringWithLocale:locale]];
}];
```

*abc*defghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
     NSString *string = [[attrString string] substringWithRange:range];

     [attrString replaceCharactersInRange:range
      withString:[string uppercaseStringWithLocale:locale]];
}];
```

ABCdefghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
    NSString *string = [[attrString string] substringWithRange:range];

    [attrString replaceCharactersInRange:range
                      withString:[string uppercaseStringWithLocale:locale]];
}];
```

ABCDEFghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
     NSString *string = [[attrString string] substringWithRange:range];

     [attrString replaceCharactersInRange:range
      withString:[string uppercaseStringWithLocale:locale]];
}];
```

ABCDEFghi

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
    NSString *string = [[attrString string] substringWithRange:range];

    [attrString replaceCharactersInRange:range
                      withString:[string uppercaseStringWithLocale:locale]];
}];
```

ABCDEFGHI

Enumerating for Modification

Uppercase while preserving attributes

```
NSMutableAttributedString *attrString;
NSLocale *locale = [NSLocale currentLocale];

[attrString enumerateAttributesInRange:NSMakeRange(0, [attrString length])
 options:0
 usingBlock:^(NSDictionary *attrs, NSRange range, BOOL *stop) {
    NSString *string = [[attrString string] substringWithRange:range];

    [attrString replaceCharactersInRange:range
                      withString:[string uppercaseStringWithLocale:locale]];
}];
```

ABCDEFGHI

Drawing Attributed Strings

Drawing an Attributed String

```
NSAttributedString *string = [[NSAttributedString alloc]
    initWithString:@"Hello World!!"
    attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];
```

Drawing an Attributed String

```
NSAttributedString *string = [[NSAttributedString alloc]
    initWithString:@"Hello World!!"
    attributes:@{ NSFontAttributeName : [UIFont systemFontOfSize:12.0f ]}];

[string drawAtPoint:NSMakePoint(10.0f, 10.0f)]; // rendering at 10,10
```


NSStringDrawing API



- Simple drawing/sizing methods

- (void)drawInRect:(CGRect)rect;
- (void)drawAtPoint:(CGPoint)point;
- (CGSize)size;

- Extended drawing/sizing methods

- (void)drawWithRect:(CGRect)rect
options:(NSStringDrawingOptions)options
context:(NSStringDrawingContext *)context;
- (CGRect)boundingRectWithSize:(CGSize)size
options:(NSStringDrawingOptions)options
context:(NSStringDrawingContext *)context;

NSStringDrawing API

- Simple drawing/sizing methods
 - (void)drawInRect:(CGRect)rect;
 - (void)drawAtPoint:(CGPoint)point;
 - (CGSize)size;

NSStringDrawing API

- Simple drawing/sizing methods
 - (void)drawInRect:(CGRect)rect;

 - (void)drawAtPoint:(CGPoint)point;

 - (CGSize)size;

NSStringDrawing API

- Simple drawing/sizing methods

- (void)drawInRect:(CGRect)rect;

- drawInRect:withFont:lineBreakMode:alignment:

- (void)drawAtPoint:(CGPoint)point;

- (CGSize)size;

NSStringDrawing API

- Simple drawing/sizing methods

- (void)drawInRect:(CGRect)rect;

- (void)drawAtPoint:(CGPoint)point;

- drawAtPoint:withFont:

- (CGSize)size;

NSStringDrawing API

- Simple drawing/sizing methods

- (void)drawInRect:(CGRect)rect;

- (void)drawAtPoint:(CGPoint)point;

- (CGSize)size;

- sizeWithFont:

Available Attributes



- Font
- Text color
- Paragraph style
- Text background color
- Ligature, kerning, and baseline offset
- Underline and strike-through
- Stroke width and color
- Shadow

Available Attributes

- Font
- Text color
- Paragraph style
- Text background color
- Ligature, kerning, and baseline offset
- Underline and strike-through
- Stroke width and color
- Shadow

Available Attributes

- Font `NSFontAttributeName (UIFont)`
- Text color `NSForegroundColorAttributeName (UIColor)`
- Paragraph style `NSParagraphStyleAttributeName (NSParagraphStyle)`

Available Attributes

- Font NSFontAttributeName (UIFont)
- Text color NSForegroundColorAttributeName (UIColor)
- Paragraph style NSParagraphStyleAttributeName (NSParagraphStyle)

A missing attribute implies its default value.

Available Attributes

Attribute default value

- Font `[UIFont systemFontOfSize:[UIFont systemFontOfSize]]`
- Text color `[UIColor blackColor]`
- Paragraph style `[NSParagraphStyle defaultParagraphStyle]`

Missing Attributes

Quartz graphics state independent

```
NSAttributedString *string;

// No attribute specified
string = [[NSAttributedString alloc] initWithString:@"Hello World!!"];

// Leaving CGContext color properties with red color
[[UIColor redColor] set];
UIRectFill(rect);

[string drawAtPoint:point];
```

Missing Attributes

Quartz graphics state independent

```
NSAttributedString *string;
```

```
// No attribute specified  
string = [[NSAttributedString alloc] initWithString:@"Hello World!!"];
```

```
// Leaving CGContext color properties with red color  
[[UIColor redColor] set];  
UIRectFill(rect);
```

```
[string drawAtPoint:point];
```

Missing Attributes

Quartz graphics state independent

```
NSAttributedString *string;
```

```
// No attribute specified  
string = [[NSAttributedString alloc] initWithString:@"Hello World!!"];
```

```
// Leaving CGContext color properties with red color  
[[UIColor redColor] set];  
UIRectFill(rect);
```

```
[string drawAtPoint:point];
```

Missing Attributes

Quartz graphics state independent

```
NSAttributedString *string;

// No attribute specified
string = [[NSAttributedString alloc] initWithString:@"Hello World!!"];

// Leaving CGContext color properties with red color
[[UIColor redColor] set];
UIRectFill(rect);

[string drawAtPoint:point];
```

Missing Attributes

Quartz graphics state independent

```
NSAttributedString *string;

// No attribute specified
string = [[NSAttributedString alloc] initWithString:@"Hello World!!"];

// Leaving CGContext color properties with red color
[[UIColor redColor] set];
UIRectFill(rect);

[string drawAtPoint:point];
```


Missing Attributes

Quartz graphics state independent

```
NSAttributedString *string;

// No attribute specified
string = [[NSAttributedString alloc] initWithString:@"Hello World!!"];

// Leaving CGContext color properties with red color
[[UIColor redColor] set];
UIRectFill(rect);

[string drawAtPoint:point]; // Renders with system font in black
```



Hello World!!

Available Attributes

- Font `NSFontAttributeName (UIFont)`
- Text color `NSForegroundColorAttributeName (UIColor)`
- Paragraph style `NSParagraphStyleAttributeName (NSParagraphStyle)`

Available Attributes

- Font `NSFontAttributeName (UIFont)`
- Text color `NSForegroundColorAttributeName (UIColor)`
- Paragraph style `NSParagraphStyleAttributeName (NSParagraphStyle)`

Layout Text Like Pages Documents

NSParagraphStyle

Layout Text Like Pages Documents

NSParagraphStyle

- Encapsulates various paragraph-wide style attributes

Layout Text Like Pages Documents

NSParagraphStyle

- Encapsulates various paragraph-wide style attributes
- Allows richer word processor like text formatting

Layout Text Like Pages Documents

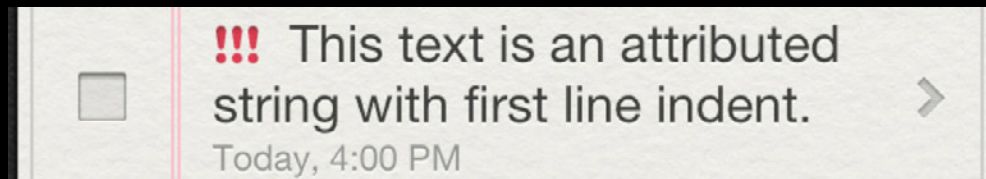
NSParagraphStyle

- Encapsulates various paragraph-wide style attributes
- Allows richer word processor like text formatting
- The paragraph style for the first character is used for the rest of paragraph

Layout Text Like Pages Documents

NSParagraphStyle

- Encapsulates various paragraph-wide style attributes
- Allows richer word processor like text formatting
- The paragraph style for the first character is used for the rest of paragraph



NSParagraphStyle

- Line break mode
- Alignment
- Spacings
- Indentation
- Line height control
- Hyphenation
- Base writing direction

NSParagraphStyle

- Line break mode
- Alignment
- Spacings
- Indentation
- Line height control
- Hyphenation
- Base writing direction

NSParagraphStyle

Line break mode

- Very similar to UILineBreakMode

```
@property NSLineBreakMode lineBreakMode;
```

```
NSMutableParagraphStyle *style = [[NSMutableParagraphStyle alloc] init];
```

```
style.lineBreakMode = NSLineBreakByTruncatingTail;
```

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]  
    initWithString:@"Hello World!!"  
    attributes:@{ NSParagraphStyleAttributeName : style }];
```

NSParagraphStyle

Line break mode

- Very similar to UILineBreakMode

```
@property NSLineBreakMode lineBreakMode;
```

```
NSMutableParagraphStyle *style = [[NSMutableParagraphStyle alloc] init];
```

```
style.lineBreakMode = NSLineBreakByTruncatingTail;
```

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]  
initWithString:@"Hello World!!"  
attributes:@{ NSParagraphStyleAttributeName : style }];
```

NSParagraphStyle

Line break mode

- Very similar to UILineBreakMode

```
@property NSLineBreakMode lineBreakMode;
```

```
NSMutableParagraphStyle *style = [[NSMutableParagraphStyle alloc] init];
```

```
style.lineBreakMode = NSLineBreakByTruncatingTail;
```

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]  
    initWithString:@"Hello World!!"  
    attributes:@{ NSParagraphStyleAttributeName : style }];
```

NSParagraphStyle

Line break mode

- Very similar to UILineBreakMode

```
@property NSLineBreakMode lineBreakMode;
```

```
NSMutableParagraphStyle *style = [[NSMutableParagraphStyle alloc] init];
```

```
style.lineBreakMode = NSLineBreakByTruncatingTail;
```

```
NSMutableAttributedString *string = [[NSMutableAttributedString alloc]
initWithString:@"Hello World!!"
attributes:@{ NSParagraphStyleAttributeName : style }];
```

Hello Wor...

NSParagraphStyle

Alignment

- Additional alignments introduced

```
@property NSTextAlignment alignment;
```

`NSTextAlignmentLeft`

!Hello World!!

NSParagraphStyle

Alignment

- Additional alignments introduced

```
@property NSTextAlignment alignment;
```

```
NSTextAlignmentLeft
```

```
NSTextAlignmentCenter
```

|| Hello World!! ||

NSParagraphStyle

Alignment

- Additional alignments introduced

```
@property NSTextAlignment alignment;
```

```
NSTextAlignmentLeft
```

```
NSTextAlignmentCenter
```

```
NSTextAlignmentRight
```

|| Hello World!! ||

NSParagraphStyle

Alignment

- Additional alignments introduced

```
@property NSTextAlignment alignment;
```

```
NSTextAlignmentLeft
```

```
NSTextAlignmentCenter
```

```
NSTextAlignmentRight
```

```
NSTextAlignmentJustified
```

This paragraph shows the new alignment mode, justification.

NSParagraphStyle

Alignment

- Additional alignments introduced

```
@property NSTextAlignment alignment;
```

```
NSTextAlignmentLeft
```

```
NSTextAlignmentCenter
```

```
NSTextAlignmentRight
```

```
NSTextAlignmentJustified
```

```
NSTextAlignmentNatural
```

!Hello World!!

NSParagraphStyle

Alignment

- Additional alignments introduced

```
@property NSTextAlignment alignment;
```

```
NSTextAlignmentLeft
```

```
NSTextAlignmentCenter
```

```
NSTextAlignmentRight
```

```
NSTextAlignmentJustified
```

```
NSTextAlignmentNatural
```

مرحبا العالم!!

Drawing Attributed Strings

Rest of drawing options

- Extended NSStringDrawing methods
- Advanced attributes
- ...and more

Drawing Attributed Strings

Rest of drawing options

- Extended NSStringDrawing methods
- Advanced attributes
- ...and more

Attributed Strings in UIKit

How do I get this stuff in my labels?

Ian Baird
Label Engineer

Attributed Strings in UIKit

UILabel is the nexus of power

Attributed Strings in UIKit

UILabel is the nexus of power

I'm a UILabel with a title.

Attributed Strings in UIKit

UILabel is the nexus of power

I'm a UILabel with an **attributed** title.

Attributed Strings in UIKit

UILabel

- Existing properties

Attributed Strings in UIKit

UILabel

- Existing properties

text

Big red dog.

Attributed Strings in UIKit

UILabel

- Existing properties

text

font

Big red dog.

Attributed Strings in UIKit

UILabel

- Existing properties

text

font

textColor

Big red dog.

Attributed Strings in UIKit

UILabel

- Existing properties

text

font

textColor

textAlignment

Big red dog.

Attributed Strings in UIKit

UILabel

- Existing properties

text

font

textColor

textAlignment

lineBreakMode

Big red
dog.

Attributed Strings in UIKit

UILabel

- Existing properties

text

font

textColor

textAlignment

lineBreakMode

shadowColor

Big red
dog.

Attributed Strings in UIKit

UILabel

- Existing properties

text

font

textColor

textAlignment

lineBreakMode

shadowColor

shadowOffset

Big red
dog.

Attributed Strings in UIKit

UILabel

- New properties:

- `attributedText`

- `minimumScaleFactor`

- `adjustsLetterSpacingToFitWidth`

Attributed Strings in UIKit

UILabel

- New properties:

 - `attributedString`

 - `minimumScaleFactor`

 - `adjustsLetterSpacingToFitWidth`

Big red dog.

Attributed Strings in UIKit

UILabel

- New properties:
attributedString

Big *red* dog.

Attributed Strings in UIKit

UILabel

- New properties:

`attributedString`

`minimumScaleFactor`

Big *red* ...

Attributed Strings in UIKit

UILabel

- New properties:

- `attributedString`

- `minimumScaleFactor`

- `adjustsLetterSpacingToFitWidth`

Big *red* dog.

Attributed Strings in UIKit

UILabel

- Deprecated property

Attributed Strings in UIKit

UILabel

- Deprecated property

`minimumFontSize`



Attributed Strings in UIKit

UILabel

- Deprecated property

`minimumFontSize`

- Instead use

Attributed Strings in UIKit

UILabel



- Deprecated property

`minimumFontSize`

- Instead use

`minimumScaleFactor`

Attributed Strings in UIKit

UILabel

Attributed Strings in UIKit

UILabel

- Style properties apply style to entire attributed string, for example

Big red dog.

Attributed Strings in UIKit

UILabel

- Style properties apply style to entire attributed string, for example
font

Big red dog.

font: Helvetica Neue

Attributed Strings in UIKit

UILabel

- Style properties apply style to entire attributed string, for example

font

textColor

Big red dog.

textColor: red

Attributed Strings in UIKit

UIButton

- Existing method

Attributed Strings in UIKit

UIButton

- Existing method

`setTitle:forControlState:`



Touch Me

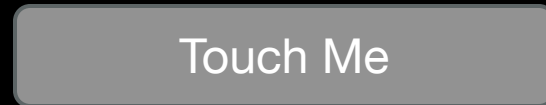
Attributed Strings in UIKit

UIButton

- Existing method

`setTitle:forControlState:`

- New method



Attributed Strings in UIKit

UIButton

- Existing method

```
setTitle:forControlState:
```

- New method

```
setAttributedTitle:forControlState:
```



Attributed Strings in UIKit

UIButton

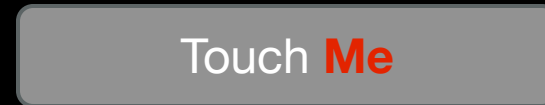
- Existing method

`setTitle:forControlState:`

- New method

`setAttributedTitle:forControlState:`

- Attributed title takes precedence

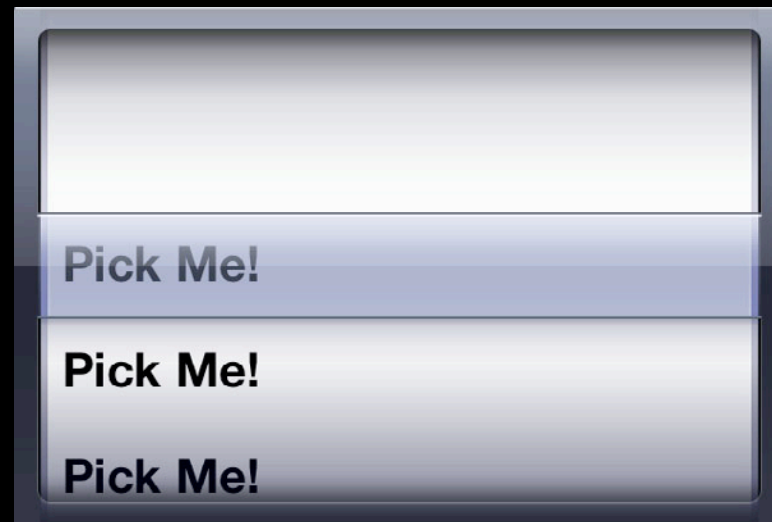


Attributed Strings in UIKit

UIPickerView

Attributed Strings in UIKit

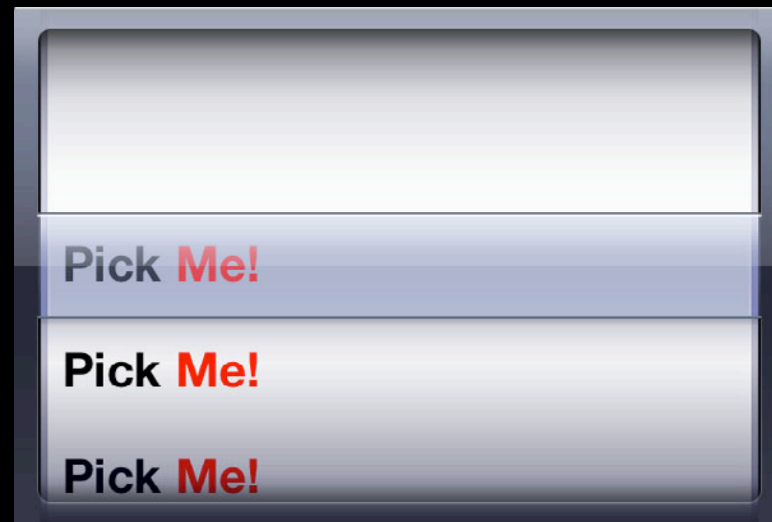
UIPickerView



```
pickerView:titleForRow:forComponent:
```

Attributed Strings in UIKit

UIPickerView



```
pickerView:attributedStringForRow:forComponent:
```

Attributed Strings in UIKit

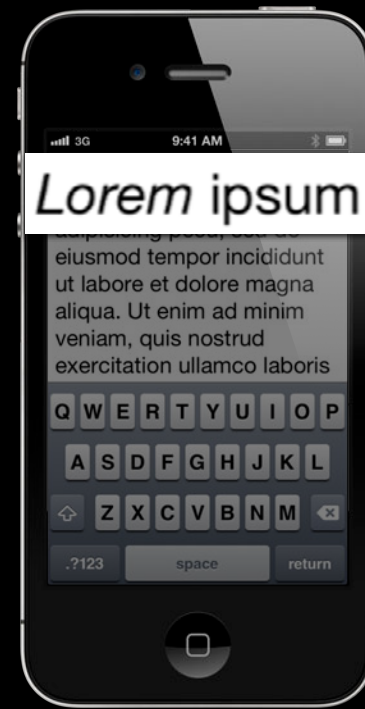
UITextView

- Added properties

Attributed Strings in UIKit

UITextView

- Added properties
`attributedText`



Attributed Strings in UIKit

UITextView

- Added properties

`attributedText`

`allowsEditingTextAttributes`



Attributed Strings in UIKit

UITableView

Attributed Strings in UIKit

UITableView

- No new properties or methods



Attributed Strings in UIKit

UITableView

- No new properties or methods
Just set `attributedString` on the cell
or header view's `textLabel`.



Demo

Johannes Fortmann
UIKit Engineer

Attributed Strings in UIKit

Demo summary

- Drawing text using attributed string

`drawInRect:`

- Enhancing existing labels

`NSMutableAttributedString, addAttribute:value:range:`

- Formatting paragraphs

`NSMutableParagraphStyle`

Summary

Summary

- Attributed string essentials

Summary

- Attributed string essentials
- Drawing and use

Summary

- Attributed string essentials
- Drawing and use
- UIKit adoption

More information

Jake Behrens

UI Frameworks Evangelist

behrens@apple.com

Documentation

UIKit Documentation

<https://developer.apple.com/library/ios>

Apple Developer Forums

<http://devforums.apple.com>

Related Sessions

Advanced Attributed Strings for iOS

Mission
Thursday 10:15am

Keyboard Input in iOS

Russian Hill
Wednesday 2:00PM

Core Text and Fonts

Russian Hill
Wednesday 4:30PM

Labs

Attributed Strings & Text Lab

Essentials Lab A
Thursday 11:30AM

 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.