NSAttributedString Class Cluster

Class Cluster Description

NSAttributedString objects manage character strings and associated sets of attributes (for example, font and kerning) that apply to individual characters or ranges of characters in the string. An association of characters and their attributes is called an *attributed string*. The cluster's two public classes, NSAttributedString and NSMutableAttributedString, declare the programmatic interface for read-only attributed strings and modifiable attributed strings, respectively. The Foundation Kit defines only the basic functionality for attributed strings; the remainder of the classes' interfaces is actually defined by the Application Kit. The Application Kit also uses a subclass of NSMutableAttributedString, called NSTextStorage, to provide the storage for NeXT's extended text-handling system.

Note: NSAttributedString is *not* a subclass of NSString. It contains a string object to which it applies attributes. This protects users of attributed strings from ambiguities caused by the semantic differences between simple and attributed string. For example, equality can't be simply defined between an NSString and an attributed string.

Because of the nature of class clusters, attributed string objects are not actual instances of the NSAttributedString or NSMutableAttributedString classes, but are instances of one of their private concrete subclasses. Although an attributed string object's class is private, its interface is public, as declared by these abstract superclasses, NSAttributedString and NSMutableAttributedString. The attributed string classes adopt the NSCopying and NSMutableCopying protocols, making it convenient to convert an attributed string from one type to the other.

You create an NSAttributedString object by using one of **initWithString:**, **initWithString:attributes:**, or **initWithAttributedString:**. These methods initialize an attributed string with data you provide. The Application Kit also provides methods for creating attributed strings from RTF data.

An attributed string provides basic access to its contents with the **string** and **attributesAtIndex:effectiveRange:** methods, which yield characters and attributes, respectively. These two primitive methods are used by the other access methods to retrieve attributes individually by name, by functional group (font or ruler attributes, for example), and so on.

The attribute values assigned to an attributed string become the property of that string, and shouldn't be modified in any way by other objects. Doing so can render inconsistent the attributed string's internal state. Always use NSMutableAttributedString's **setAttributes:range:** and related methods to change attribute values.

For more information on using attributed strings, see the Application Kit's specification for this class cluster.

NSAttributedString

Inherits From: NSObject

Conforms To: NSCoding
NSCopying
NSMutableCopying
NSObject (NSObject)

Declared In: Foundation/NSAttributedString.h

Class Description

The NSAttributedString class declares the programmatic interface to an object that manages an immutable attributed string. NSAttributedString's two primitive methods provide the basis for all the other methods in the class. The primitive **string** method returns the NSString object to which attributes are applied. The primitive **attributesAtIndex:effectiveRange:** method returns an NSDictionary containing the attribute names and values for characters around a specified index.

Adopted Protocols

NSCoding	– encodeWithCoder: – initWithCoder:
NSCopying	– copyWithZone:
NSMutableCopying	– mutableCopyWithZone:

Method Types

Creating an NSAttributedString	– initWithString: – initWithAttributedString: – initWithString:attributes:
Retrieving character information	– string – length
Retrieving attribute information	– attributesAtIndex:effectiveRange: – attributesAtIndex:longestEffectiveRange:inRange: – attribute:atIndex:effectiveRange: – attribute:atIndex:longestEffectiveRange:inRange:

Comparing attributed strings	– isEqualToAttributedString:
Extracting a substring	– attributedSubstringFromRange:

Instance Methods

NS+ attribute:atIndex:effectiveRange:

– (id)**attribute:**(NSString *)*attributeName*
atIndex:(unsigned int)*index*
effectiveRange:(NSRange *)*aRange*

Returns the value for the attribute named *attributeName* of the character at *index*, or **nil** if there is no such attribute. If the named attribute exists at *index* and *aRange* is non-NULL, it's filled with a range over which the named attribute's value applies. If the named attribute doesn't exist at *index* and *aRange* is non-NULL, *aRange* is filled instead with the range over which the attribute doesn't exist. This range isn't necessarily the maximum range covered by *attributeName*, and its extent is implementation-dependent. If you need the maximum range, use **attribute:atIndex:longestEffectiveRange:inRange:**.

Raises an NSRangeException if *index* lies beyond the end of the receiver's characters.

See also: – **attributesAtIndex:effectiveRange:**

NS+ attribute:atIndex:longestEffectiveRange:inRange:

– (id)**attribute:**(NSString *)*attributeName*
atIndex:(unsigned int)*index*
longestEffectiveRange:(NSRange *)*aRange*
inRange:(NSRange)*rangeLimit*

Returns the value for the attribute named *attributeName* of the character at *index*, or **nil** if there is no such attribute. If the named attribute exists at *index* and *aRange* is non-NULL, it's filled with the full range over which the value of the named attribute is the same as that at *index*, clipped to *rangeLimit*. If the named attribute doesn't exist at *index* and *aRange* is non-NULL, *aRange* is filled instead with the full range over which the attribute doesn't exist, clipped to *rangeLimit*.

Raises an NSRangeException if *index* or any part of *rangeLimit* lies beyond the end of the receiver's characters.

If you don't need the longest effective range, it's far more efficient to use the **attribute:atIndex:effectiveRange:** method to retrieve the attribute value.

See also: – **attributesAtIndex:longestEffectiveRange:inRange:**

NS+ attributedSubstringFromRange:

– (NSAttributedString *)**attributedSubstringFromRange:**(NSRange)*aRange*

Returns an NSAttributedString object consisting of the characters and attributes within *aRange* in the receiver. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

NS+ attributesAtIndex:effectiveRange:

– (NSDictionary *)**attributesAtIndex:**(unsigned int)*index* **effectiveRange:**(NSRange *)*aRange*

Returns the attributes for the character at *index*. If *aRange* is non-NULL it's filled with the range over which the attributes and values apply. This range isn't necessarily the maximum range covered, and its extent is implementation-dependent. If you need the maximum range, use **attributesAtIndex:longestEffectiveRange:inRange:**.

Raises an NSRangeException if *index* lies beyond the end of the receiver's characters.

See also: – **attribute:atIndex:effectiveRange:**

NS+ attributesAtIndex:longestEffectiveRange:inRange:

– (NSDictionary *)**attributesAtIndex:**(unsigned int)*index*
longestEffectiveRange:(NSRange *)*aRange*
inRange:(NSRange)*rangeLimit*

Returns the attributes for the character at *index*. If *aRange* is non-NULL, it's filled with the maximum range over which the attributes and values are the same as those at *index*, clipped to *rangeLimit*.

Raises an NSRangeException if *index* or any part of *rangeLimit* lies beyond the end of the receiver's characters.

If you don't need the range information, it's far more efficient to use the **attributesAtIndex:effectiveRange:** method to retrieve the attribute value.

See also: – **attribute:atIndex:longestEffectiveRange:inRange:**

NS+ initWithAttributedString:

– (id)**initWithAttributedString:**(NSAttributedString *)*attributedString*

Initializes a newly allocated NSAttributedString object with the characters and attributes of *attributedString*. Returns **self**.

See also: – **initWithRTF:documentAttributes:** (NSAttributedString Additions, Application Kit)

NS+ initWithString:

– (id)**initWithString:**(NSString *)*aString*

Initializes a newly allocated NSAttributedString object with the characters of *aString* and with no attribute information. Returns **self**.

See also: – **initWithRTF:documentAttributes:** (NSAttributedString Additions, Application Kit)

NS+ initWithString:attributes:

– (id)**initWithString:**(NSString *)*aString* **attributes:**(NSDictionary *)*attributes*

Initializes a newly allocated NSAttributedString object with the characters of *aString* and the attributes of *attributes*. Returns **self**.

See also: – **initWithRTF:documentAttributes:** (NSAttributedString Additions, Application Kit)

NS+ isEqualToAttributedString:

– (BOOL)**isEqualToAttributedString:**(NSAttributedString *)*otherString*

Returns YES if the receiver is equal to *otherString*. Attributed strings must match in both characters and attributes to be equal.

NS+ length

– (unsigned int)**length**

Returns the length of the receiver's string object.

See also: – **length** (NSString), – **size**

NS+ string

– (NSString *)**string**

Returns the character contents of the receiver as an NSString object. This method doesn't strip out attachment characters; use NSText's **string** method to extract just the linguistically significant characters.

This primitive method must guarantee efficient access to an attributed string's characters; subclasses should implement it to execute in O(1) time.

NSMutableAttributedString

Inherits From:	NSAttributedString : NSObject
Conforms To:	NSCoding (NSAttributedString) NSCopying (NSAttributedString) NSMutableCopying (NSAttributedString) NSObject (NSObject)
Declared In:	Foundation/NSAttributedString.h

Class Description

NSMutableAttributedString declares the programmatic interface to objects that manage mutable attributed strings. You can add and remove characters (raw strings) and attributes separately, or together as attributed strings. When working with the Application Kit, you must also clean up changed attributes using the various **fix...** methods. See the Application Kit's specification for this class cluster for more information on fixing attributes.

NSMutableAttributedString adds two primitive methods to those of NSAttributedString. These primitive methods provide the basis for all the other methods in its class. The primitive **replaceCharactersInRange:withString:** method replaces a range of characters with those from a string, leaving all attribute information outside that range intact. The primitive **setAttributes:range:** method sets attributes and values for a given range of characters, replacing any previous attributes and values for that range.

Method Types

Retrieving character information	– mutableString
Changing characters	– replaceCharactersInRange:withString: – deleteCharactersInRange:
Changing attributes	– setAttributes:range: – addAttribute:value:range: – addAttributes:range: – removeAttribute:range:
Changing characters and attributes	– appendAttributedString: – insertAttributedString:atIndex: – replaceCharactersInRange:withAttributedString: – setAttributedString:

Grouping changes	– beginEditing – endEditing

Instance Methods

NS+ addAttribute:value:range:

– (void)**addAttribute:**(NSString *)*name*
value:(id)*value*
range:(NSRange)*aRange*

Adds an attribute with the given *name* and *value* to the characters in *aRange*. Raises an NSInvalidArgumentException if *name* or *value* is **nil**, and an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **addAttributes:range:**, – **removeAttribute:range:**

NS+ addAttributes:range:

– (void)**addAttributes:**(NSDictionary *)*attributes* **range:**(NSRange)*aRange*

Adds the collection of attributes in *attributes* to the characters in *aRange*. Raises an NSInvalidArgumentException if *attributes* is **nil** and an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **addAttribute:value:range:**, – **removeAttribute:range:**

NS+ appendAttributedString:

– (void)**appendAttributedString:**(NSAttributedString *)*attributedString*

Adds the characters and attributes of *attributedString* to the end of the receiver.

See also: – **insertAttributedString:atIndex:**, + **attributedStringWithAttachment:** (NSAttributedString Additions, Application Kit)

NS+ beginEditing

– (void)**beginEditing**

Overridden by subclasses to buffer or optimize a series of changes to the receiver's characters or attributes, until it receives a matching **endEditing** message, upon which it can consolidate changes and notify any observers that it has changed. You can nest pairs of **beginEditing** and **endEditing** messages.

NS+ deleteCharactersInRange:

– (void)**deleteCharactersInRange:**(NSRange)*aRange*

Deletes the characters in *aRange* along with their associated attributes. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **replaceCharactersInRange:withAttributedString:**,
– **replaceCharactersInRange:withString:**

NS+ endEditing

– (void)**endEditing**

Overridden by subclasses to consolidate changes made since a previous **beginEditing** message and to notify any observers of the changes. NSMutableAttributedString's implementation does nothing. NSTextStorage, for example, overrides this method to invoke **fixAttributesInRange:** and to inform its NSLayoutManagers that they need to re-lay the text.

See also: – **processEditing** (NSTextStorage class of the Application Kit)

NS+ insertAttributedString:atIndex:

– (void)**insertAttributedString:**(NSAttributedString *)*attributedString* **atIndex:**(unsigned int)*index*

Inserts the characters and attributes of *attributedString* into the receiver, so that the new characters and attributes begin at *index* and the existing characters and attributes from *index* to the end are shifted by the length of *attributedString*. Raises an NSRangeException if *index* lies beyond the end of the receiver's characters.

See also: – **appendAttributedString:**, + **attributedStringWithAttachment:** (NSAttributedString Additions, Application Kit)

NS+ mutableString

– (NSMutableString *)**mutableString**

Returns the character contents of the receiver as an NSMutableString object. The receiver tracks changes to this string and keeps its attribute mappings up to date.

See also: – **replaceCharactersInRange:withString:**

NS+ removeAttribute:range:

– (void)**removeAttribute:**(NSString *)*name* **range:**(NSRange)*aRange*

Removes the attribute named *name* from the characters in *aRange*. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **addAttribute:value:range:**, – **addAttributes:range:**

NS+ replaceCharactersInRange:withAttributedString:

– (void)**replaceCharactersInRange:**(NSRange)*aRange*
withAttributedString:(NSAttributedString *)*attributedString*

Replaces the characters and attributes in *aRange* with the characters and attributes of *attributedString*. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **insertAttributedString:atIndex:**

NS+ replaceCharactersInRange:withString:

– (void)**replaceCharactersInRange:**(NSRange)*aRange* **withString:**(NSString *)*aString*

Replaces the characters in *aRange* with the characters of *aString*. The new characters inherit the attributes of the first replaced character from *aRange*. Where the length of *aRange* is zero, the new characters inherit the attributes of the character preceding *aRange* if it has any, otherwise of the character following *aRange*.

Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **deleteCharactersInRange:**

NS+ setAttributedString:

– (void)**setAttributedString:**(NSAttributedString *)*attributedString*

Replaces the receiver's entire contents with the characters and attributes of *attributedString*.

See also: – **appendAttributedString:**

NS+ setAttributes:range:

– (void)**setAttributes:**(NSDictionary *)*attributes* **range:**(NSRange)*aRange*

Sets the attributes for the characters in *aRange* to *attributes*. These new attributes replace any attributes previously associated with the characters in *aRange*. Raises an NSRangeException if any part of *aRange* lies beyond the end of the receiver's characters.

See also: – **addAtributes:range:**, – **removeAttributes:range:**