Getting an NSScanner's String	string
Configuring an NSScanner	setScanLocation:
	scanLocation
	setCaseSensitive:
	caseSensitive
	setCharactersToBeSkipped:
	charactersToBeSkipped
Scanning a String	scanInt:
	scanLongLong:
	scanFloat:
	scanDouble:
	scanString:intoString:
	scanCharactersFromSet:intoString:
	scanUpToString:intoString:
	scanUpToCharactersFromSet:intoString:
	isAtEnd

initWithString:

(BOOL)caseSensitive

Returns YES if the scanner distinguishes case, NO otherwise. NSScanners are by default not case s

setCaseSensitive:

(NSCharacterSet *)charactersToBeSkipped

Returns a character set containing those characters that the scanner ignores when looking for a scan
example, if a scanner ignores spaces and you ask it to scanInt:, it will skip spaces until it finds a de
character. While an element is being scanned, however, no characters are skipped. If you scan for
characters in the set to be skipped (for example, using scanInt: when the set of characters to be skip
digits), the result is undefined.

The default set is the whitespace and newline character set.

setCharactersToBeSkipped:, + whitespaceAndNewlineCharacterSet (NSCharacterSet)

initWithString:(NSString *)aString

Initializes the receiver, a newly allocated NSScanner, to scan aString. Returns self.

(BOOL)scanCharactersFromSet:(NSCharacterSet *)aSet intoString:(NSString **)value

Scans the string as long as characters from aSet are encountered, accumulating characters into a str
reference in value. If any characters are scanned, returns YES otherwise returns NO.

This method may be invoked with nil as value to simply scan past a given set of characters.

scanUpToCharactersFromSet:intoString:

(BOOL)scanDouble:(double *)value

Scans a double into value if possible. Returns YES if a valid floating-point expression was scanne
HUGE_VAL or HUGE_VAL is put in value on overflow, 0.0 on underflow.

(BOOL)scanFloat:(float *)value

Scans a float into value if possible. Returns YES if a valid floating-point expression was scanned,
HUGE_VAL or HUGE_VAL is put in value on overflow, 0.0 on underflow.

(BOOL)scanInt:(int *)value

Scans an int into value if possible. Returns YES if a valid integer expression was scanned, NO oth
INT_MIN is put in value on overflow.

(unsigned)scanLocation

Returns the character index at which the scanner will begin its next scanning operation.

setScanLocation:

(BOOL)scanLongLong:(long long *)value

Scans a long long int into value if possible. Returns YES if a valid integer expression was scanned
LONG_LONG_MAX or LONG_LONG_MIN is put in value on overflow.

(BOOL)scanString:(NSString *)aString intoString:(NSString **)value

Scans for aString, and if a match is found returns (by reference) in value a string object equal to it.
characters at the scan location, returns YES otherwise returns NO.

This method may be invoked with nil as value to simply scan past a given string.

scanCharactersFromSet:intoString:

(BOOL)scanUpToString:(NSString *)aString intoString:(NSString **)value

Scans the string until aString is encountered, accumulating characters encountered into a string tha
reference in value. The receiver's scan location will then be at the beginning of aString (or at the e
scanned if aString isn't found). If any characters are scanned, returns YES otherwise returns NO.

This method may be invoked with NULL as value to simply scan up to a given string.

scanString:intoString:

(void)setCaseSensitive:(BOOL)flag

If flag is YES, the scanner will distinguish case when scanning characters. If flag is NO, it will ig
NSScanners are by default not case sensitive.

caseSensitive

(void)setCharactersToBeSkipped:(NSCharacterSet *)aSet

Sets the scanner to ignore characters from aSet when scanning its string. Such characters are simpl
scanning. For example, if a scanner ignores spaces and you ask it to scanInt:, it will skip spaces un
digit or other character. While an element is being scanned, however, no characters are skipped. If
made of characters in the set to be skipped (for example, using scanInt: when the set of characters
decimal digits), the result is undefined.

The default set is the whitespace and newline character set.

charactersToBeSkipped, + whitespaceAndNewlineCharacterSet (NSCharacterSet)

(void)setScanLocation:(unsigned)index

Sets the location at which the next scan will begin to index. This method is useful for backing up t
You should use scanString:intoString: or scanCharactersFromSet:intoString: to skip ahead past kn
characters, as this allows you to check for errors in a way that setting the scan location ahead does

scanLocation

(NSString *)string

Returns the string object that the scanner was created with.