NSNumber

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

Class at a Glance

Purpose

An NSNumber object serves as an object wrapper for C numeric data items, allowing them to be stored in collections such as NSArray and NSDictionary objects.

Creation

+ numberWith*Type*:	Returns an initialized NSNumber of the specified type.

Commonly Used Methods

– *type*Value	Returns the value of an NSNumber as a specific numeric type.
– compare:	Compares two NSNumbers.

lass Description

NSNumber is a subclass of NSValue that offers a value as any C scalar (numeric) type. It defines a set of methods specifically for setting and accessing the value as a signed or unsigned **char**, **short**, **int**, **long int**, **long long int**, **float**, or **double**, or as a BOOL. It also defines a **compare:** method to determine the ordering of two NSNumber objects.

An NSNumber records the numeric type it’s created with, and uses the C rules for numeric conversion when comparing NSNumbers of different numeric types and when returning values as C numeric types. See any standard C reference for information on type conversion.

Method Types

Creating an NSNumber	+ numberWithBool:
	+ numberWithChar:
	+ numberWithDouble:
	+ numberWithFloat:
	+ numberWithInt:
	+ numberWithLong:
	+ numberWithLongLong:
	+ numberWithShort:
	+ numberWithUnsignedChar:
	+ numberWithUnsignedInt:
	+ numberWithUnsignedLong:
	+ numberWithUnsignedLongLong:
	+ numberWithUnsignedShort:
	– initWithBool:
	– initWithChar:
	– initWithDouble:
	– initWithFloat:
	– initWithInt:
	– initWithLong:
	– initWithLongLong:
	– initWithShort:
	– initWithUnsignedChar:
	– initWithUnsignedInt:
	– initWithUnsignedLong:
	– initWithUnsignedLongLong:
	– initWithUnsignedShort:
Accessing numeric values	– boolValue
	– charValue
	– descriptionWithLocale:
	– doubleValue
	– floatValue
	– intValue
	– longLongValue
	– longValue
	– shortValue
	– stringValue
	– unsignedCharValue
	– unsignedIntValue
	– unsignedLongLongValue
	– unsignedLongValue
	– unsignedShortValue

Comparing NSNumbers	– compare:

lass Methods

numberWithBool:

+ (NSNumber *)**numberWithBool:**(BOOL)*value*

Creates and returns an NSNumber containing *value*, treating it as a BOOL.

numberWithChar:

+ (NSNumber *)**numberWithChar:**(char)*value*

Creates and returns an NSNumber containing *value*, treating it as a signed **char**.

numberWithDouble:

+ (NSNumber *)**numberWithDouble:**(double)*value*

Creates and returns an NSNumber containing *value*, treating it as a **double**.

numberWithFloat:

+ (NSNumber *)**numberWithFloat:**(float)*value*

Creates and returns an NSNumber containing *value*, treating it as a **float**.

numberWithInt:

+ (NSNumber *)**numberWithInt:**(int)*value*

Creates and returns an NSNumber containing *value*, treating it as a signed **int**.

numberWithLong:

+ (NSNumber *)**numberWithLong:**(long int)*value*

Creates and returns an NSNumber containing *value*, treating it as a signed **long int**.

numberWithLongLong:

+ (NSNumber *)**numberWithLongLong:**(long long int)*value*

Creates and returns an NSNumber containing *value*, treating it as a signed **long long int**.

numberWithShort:

+ (NSNumber *)**numberWithShort:**(short int)*value*

Creates and returns an NSNumber containing *value*, treating it as a signed **short int**.

numberWithUnsignedChar:

+ (NSNumber *)**numberWithUnsignedChar:**(unsigned char)*value*

Creates and returns an NSNumber containing *value*, treating it as an **unsigned char**.

numberWithUnsignedInt:

+ (NSNumber *)**numberWithUnsignedInt:**(unsigned int)*value*

Creates and returns an NSNumber containing *value*, treating it as an **unsigned int**.

numberWithUnsignedLong:

+ (NSNumber *)**numberWithUnsignedLong:**(unsigned long int)*value*

Creates and returns an NSNumber containing *value*, treating it as an **unsigned long int**.

numberWithUnsignedLongLong:

+ (NSNumber *)**numberWithUnsignedLongLong:**(unsigned long long int)*value*

Creates and returns an NSNumber containing *value*, treating it as an **unsigned long long int**.

numberWithUnsignedShort:

+ (NSNumber *)**numberWithUnsignedShort:**(unsigned short int)*value*

Creates and returns an NSNumber containing *value*, treating it as an **unsigned short int**.

nstance Methods

boolValue

– (BOOL)**boolValue**

Returns the receiver’s value as a BOOL, converting it as necessary.

Note: The value returned by this method isn’t guaranteed to be one of YES or NO. A zero value always means NO or false, but any nonzero value should be interpreted as YES or true.

charValue

– (char)**charValue**

Returns the receiver’s value as a char, converting it as necessary.

compare:

– (NSComparisonResult)**compare:**(NSNumber *)*aNumber*

Returns NSOrderedAscending if *aNumber*’s value is greater than the receiver’s, NSOrderedSame if they’re equal, and NSOrderedDescending if *aNumber*’s value is less than the receiver’s.

compare: follows the standard C rules for type conversion. For example, if you compare an NSNumber that has an integer value with an NSNumber that has a floating point value, the integer value is converted to a float for comparison.

descriptionWithLocale:

– (NSString *)**descriptionWithLocale:**(NSDictionary *)*aLocale*

Returns an NSString that represents the contents of the receiver. *aLocale* specifies options used for formatting the description; use **nil** if you don’t want the description formatted.

To obtain the string representation, this method invokes NSString’s **initWithFormat:locale:** method, supplying the format based on the type the NSNumber was created with:

Data Type	Format Specification
char	%i
double	%0.16g
float	%0.7g
int	%i
long	%li
long long	%li
short	%hi
unsigned char	%u
unsigned int	%u
unsigned long	%lu
unsigned long long	%lu
unsigned short	%hu

See also: – **stringValue**

doubleValue

– (double)**doubleValue**

Returns the receiver's value as a double, converting it as necessary.

floatValue

– (float)**floatValue**

Returns the receiver's value as a float, converting it as necessary.

initWithBool:

– (id)**initWithBool:**(BOOL)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a BOOL.

initWithChar:

– (id)**initWithChar:**(char)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a signed **char**.

initWithDouble:

– (id)**initWithDouble:**(double)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a **double**.

initWithFloat:

– (id)**initWithFloat:**(float)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a **float**.

initWithInt:

– (id)**initWithInt:**(int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a signed **int**.

initWithLong:

– (id)**initWithLong:**(long int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a signed **long int**.

initWithLongLong:

– (id)**initWithLongLong:**(long long int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a signed **long long int**.

initWithShort:

– (id)**initWithShort:**(short int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as a signed **short int**.

initWithUnsignedChar:

– (id)**initWithUnsignedChar:**(unsigned char)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as an **unsigned char**.

initWithUnsignedInt:

– (id)**initWithUnsignedInt:**(unsigned int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as an **unsigned int**.

initWithUnsignedLong:

– (id)**initWithUnsignedLong:**(unsigned long int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as an **unsigned long int**.

initWithUnsignedLongLong:

– (id)**initWithUnsignedLongLong:**(unsigned long long int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as an **unsigned long long int**.

initWithUnsignedShort:

– (id)**initWithUnsignedShort:**(unsigned short int)*value*

Initializes a newly allocated NSNumber to contain *value*, treated as an **unsigned short int**.

intValue

– (int)**intValue**

Returns the receiver's value as an int, converting it as necessary.

isEqual:

@protocol NSObject
– (BOOL)**isEqual:**(id)*anObject*

Returns YES if the receiver and *anObject* are equal, otherwise returns NO. An NSNumber is equal to *anObject* if they have the same **id**s or if they're both NSNumbers with equivalent values (as determined using the **compare:** method).

longLongValue

– (long long int)**longLongValue**

Returns the receiver’s value as a long long int, converting it as necessary.

longValue

– (long int)**longValue**

Returns the receiver’s value as a long int, converting it as necessary.

shortValue

– (short int)**shortValue**

Returns the receiver’s value as a short int, converting it as necessary.

stringValue

– (NSString *)**stringValue**

Returns the receiver’s value as a human-readable NSString, by invoking **descriptionWithLocale:** where locale is **nil.**

unsignedCharValue

– (unsigned char)**unsignedCharValue**

Returns the receiver’s value as an unsigned char, converting it as necessary.

unsignedIntValue

– (unsigned int)**unsignedIntValue**

Returns the receiver’s value as an unsigned int, converting it as necessary.

unsignedLongLongValue

– (unsigned long long int)**unsignedLongLongValue**

Returns the receiver’s value as an unsigned long long int, converting it as necessary.

unsignedLongValue

– (unsigned long int)**unsignedLongValue**

Returns the receiver’s value as an unsigned long int, converting it as necessary.

unsignedShortValue

– (unsigned short int)**unsignedShortValue**

Returns the receiver’s value as an unsigned short int, converting it as necessary.