tInitializes a new, default HashTable
tKeyDesc:(const char *)aKeyDescInitializes a new HashTable
tKeyDesc:(const char *)aKeyDescInitializes a new HashTable
valueDesc:(const char *)aValueDesc
tKeyDesc:(const char *)aKeyDescInitializes a new HashTable
valueDesc:(const char *)aValueDesc
capacity:(unsigned int)aCapacity
eDeallocates the HashTable
eObjectsDeallocates the HashTable's objects
eKeys:(void (*)(void *))keyFuncConditionally frees the HashTable's associations
values:(void (*)(void *))valueFunc
ptyEmpties the HashTable but retains its capacity

pyFromZone:(NXZone *) zoneReturns an empty copy of the HashTable

signed int)countReturns the number of objects in the table
OOL)isKey:(const void *)aKeyIndicates whether aKey is in the table
id *)valueForKey:(const void *)aKeyReturns the value mapped to aKey
id *)insertKey:(const void *)aKeyAdds or updates akey/avalue pair
value:(void *)aValue
id *)removeKey:(const void *)aKeyRemoves akey/avalue pair

XHashState)initStateBegins process of iteration through the HashTable
OOL)nextState:(NXHashState *)aStateMoves to the next entry in the HashTable
key:(const void **)aKey
value:(void **)aValue

d:(NXTypedStream *)streamRead the HashTable from the typed stream stream
te:(NXTypedStream *)streamWrites the HashTable to the typed stream stream

tInitializes the new List object
tCount:(unsigned int)numSlotsInitializes the new List to hold at least numSlots objects

moveObjectAt:(unsigned int)indexRemoves the object located at index
moveLastObjectRemoves the object at the end of the List
placeObjectAt:(unsigned int)index
with:newObjectPuts newObject in place of the object at index
ectAt:(unsigned int)indexReturns the object at index
tObjectReturns the object at the end of the List
signed int)countReturns the number of objects in the List

dObject:anObjectAdds anObject at the end of the List
dObjectIfAbsent:anObjectAdds anObject to the List, if it's not already in the List
moveObject:anObjectRemoves first occurrence of anObject from the List
placeObject:anObject with:newObjectPuts newObject in the List in place of anObject
signed int)indexOf:anObjectReturns the index of anObject

OOL)isEqual:anObjectReturns whether the two Lists have the same contents
pendList:(List *)otherListAdds the objects in otherList to the receiving List

ptyEmpties the List of its contents, but doesn't free the objects
eObjectsDeallocates all the objects in the List

keObjectsPerform:(SEL)aSelectorSends an aSelector message to each object in the List
keObjectsPerform:(SEL)aSelectorSends aSelector message with an argument to each object
with:anObject

signed int)capacityReturns the number of objects the List can store
AvailableCapacity:(unsigned int)numSlotsSets the capacity of the List to at least numSlots objects

te:(NXTypedStream *)streamWrites the List to the typed stream stream
d:(NXTypedStream *)streamReads the List from the typed stream stream

tForDirectory:(const char *)fullPathInitializes a new object for the fullPath directory

ssNamed:(const char *)classNameReturns the class object for the classname class

OOL)getPath:(char *)pathProvides the full path to the filename resource
forResource:(const char *)filename
ofType:(const char *)extension
BOOL)getPath:(char *)pathProvides the full path to the filename resource
forResource:(const char *)filename
ofType:(const char *)extension
inDirectory:(const char *)directory
withVersion:(int)version

nst char *)directoryReturns the full pathname of the NXBundle's directory

Version:(int)versionSets the version that resources must match
t)versionReturns the version that resources must match

tInitializes a new NXStringTable
eDeallocates the NXStringTable

nst char *)valueForStringKey:(const char *)aString
Returns the value that corresponds to aString

dFromFile:(const char *)fileNameReads the keys and values from fileName
teToFile:(const char *)fileNameWrites the keys and values to fileName
dFromStream:(NXStream *)streamReads the keys and values from stream
teToStream:(NXStream *)streamWrites the keys and values to stream

tInitializes the Storage object

dElement:(void *)anElementAdds anElement at the end of the Storage array
ertElement:(void *)anElement
at:(unsigned int)indexPuts anElement in the Storage array at index
noveElementAt:(unsigned int)indexRemoves the element located at index
noveLastElementRemoves the last element
laceElementAt:(unsigned int)index
with:(void *)anElementReplaces the element at index with anElement
ptyEmpties the Storage object but retains its capacity
id *)elementAt:(unsigned int)indexReturns a pointer to the element at index

OOL)isEqual:anObjectReturns whether two Storage objects are the same

signed int)countReturns the number of elements currently stored
nst char *)descriptionReturns the encoding for the type of elements stored
AvailableCapacity:(unsigned int)numSlotsSets the capacity of the Storage array to at least numSlots
NumSlots:(unsigned int)numSlotsSets the number of elements stored to numSlots elements

d:(NXTypedStream *)streamReads the Storage object from the typed stream stream
te:(NXTypedStream *)streamWrites the Storage object to the typed stream stream