FCCompareType

Declared In: Foundation/FCCompare.h

```
typedef enum {
            FC_COMPARE_LESS_THAN = -1,
            FC_COMPARE_EQUAL_TO = 0,
            FC_COMPARE_GREATER_THAN = 1,
            FC_COMPARE_NOT_EQUAL_TO = 2,
            FC_COMPARE_CANT_COMPARE = 3
                              } FCCompareType;
```

This enum is used as a return value for comparison routines. The FCOrderedCollection subclasses can sort themselves by comparing objects; the comparison method returns this type. FCCollection and FCString also respond to a compare: method which will return this type. There are also six

primitive comparetype() methods which also return this type, giving the programmer a convenient tool for writing complex class comparison methods.

FC_COMPARE_LESS_THAN, **FC_COMPARE_EQUAL_TO**, and **FC_COMPARE_GREATER_THAN** should be used when the first object in the comparison is less than, equal to, or greater than the second object, respectively.

FC_COMPARE_NOT_EQUAL_TO should be returned when less than and greater than comparisons are meaningless. For instance, if you are comparing two FCCollections, they will either return

FC_COMPARE_EQUAL_TO or **FC_COMPARE_NOT_EQUAL_TO**, since there is no well-defined standard notion of what it means for one collection to be greater than another.

FC_COMPARE_CANT_COMPARE should be returned when two objects are compared that don't have any well-defined standard basis for comparison. For example, if you asked an apple to compare itself to an orange, it should probably return **FC_COMPARE_CANT_COMPARE**. To be more concrete, if you compare two FCCollections that have different **contentClass**es set, the comparison will return **FC_COMPARE_CANT_COMPARE**, since the objects in these two collections are all, by definition, different.

See also: compareChars(), compareFloats(), compareInts(), compareLongs(), compareShorts(), compareStrings(), -€ setSortSelector (FCSortedCollection), -€ sortByCompare: (FCOrderedCollection)

FCLoopState

Declared In: Foundation/FCCollection.h

```
typedef struct {
    NXHashState hashState;
    int intState;
  } FCLoopState;
```

This structure is used by FCCollection and its subclasses to loop through all the elements in the collection. Its fields are private and shouldn't be accessed.

See also: -€ startLoop:, -€ nextObject:, -€ peekNextObject: (FCCollection), FOR_EACH, FOR_EACH_EXCEPT_FIRST, FOR_EACH_SELECTED (FCCollection macros), FOR_EACH_BACKWARDS (FCOrderedCollection macro)