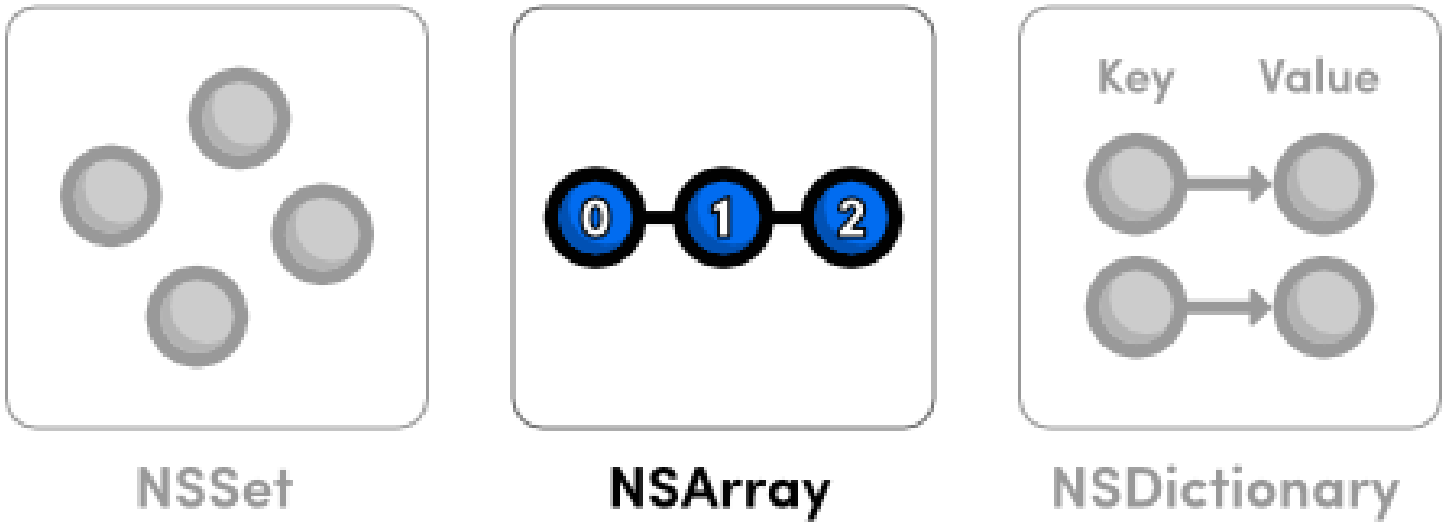
Introduction to Objective-C (Collections & Blocks)

Sisoft Technologies Pvt Ltd
SRC E7, Shipra Riviera Bazar, Gyan Khand-3, Indirapuram, Ghaziabad
Website: www.sisoft.in Email:info@sisoft.in
Phone: +91-9999-283-283

Collection Classes



The basic collection classes of the Foundation Framework

NSArray – Immutable

- Declaration
  - With arrayWithObjects: class method
    NSArray *thearray = [NSArray arrayWithObjects:o1,o2,o3,o4, nil];
  - With literals using the @[] syntax.
    NSArray *ncr = @[@"Delhi",@"Noida", @"Ghaziabad",@"Noida", @"Gurgoan"];
- get element
  - [thearray objectAtIndex:0]; //element at index 0

Example

```
NSDate *now = [NSDate date];
NSDate *tomorrow = [now dateByAddingTImeInterval:24.0*60.0*60.0]; //add a day
NSDate *yesterday = [now dateByAddingTimeInterval:-24.0*60.0*60.0]; //minus a day

//array of Dates
NSArray  *dateList  = [NSArray arrayWithObjects:now, tomorrow, yesterday];
```

NSArray – cycle through with for loop


```
//get elements in array

NSDate *first = [dateList objectAtIndex:0];
```

Methods are:
count = gets number of items in array
objectAtIndex:i = returns element i of array (starting from 0)

```
NSInteger dateCount = [dateList count];
for(int i=0; i<dateCount; i++)
  {   NSDAte *d = [dateList objectAtIndex:i];
      NSLog(@" Date is %@", d);
  }
```

NSArray – cycle through with for loop

OPTION 2

```
For(NSDate *d in dateList)
  {
    NSLog(@” Date is %@”, d);
  }
```

This is a “for in” loop --- convenient

NSMutableArray – Mutable

- The NSMutableArray class lets you dynamically add or remove items from arbitrary locations in the collection.
- Literal arrays are always immutable, so the easiest way to create mutable arrays is still through the arrayWithObjects: method.

```
NSMutableArray   *thearray = [NSArray arrayWithObjects:o1,o2,nil];


//get element
[thearray objectAtIndex:0];   //element at index 0
```

Example

```
//array of Dates
NSMutableArray  *dateList  = [NSMutableArray array];

//set elements
[dateList addObject:now];
[dateList addObject:tomorrow];
[dateList addObject:yesterday];
```

NSMutableArray – adding element at Index location

[arrayName insertObject:obj atIndex:i]

Example -- put in at beginning of array

[dateList insertObject:yesterday atIndex:0]

NSMutableArray – removing an element

[arrayName removeObjectAtIndex:i]

Example

[dateList removeObjectAtIndex:0] //get rid of 1st element

NSSet

An NSSet object represents a static, unordered collection of distinct objects

```
NSSet *ncr = [NSSet setWithObjects:
    @”Delhi”,@”Noida”, @”Ghaziabad”, nil ] ;
NSLog(@”%@”, ncr) ;
NSArray *ncr = @[@”Delhi”,@”Noida”,
    @”Ghaziabad”,@”Noida”, @”Gurgoan”];
NSSet uniqueNcr = [NSSet setWithArray: ncr] ;
NSLog(@”%@”, uniqueNcr) ;
```

NSSet

Fast Enumeration

```
For (id item in ncr)
{
    NSLog(@”%@”, item) ;
}
```

NSDictionary

- An NSDictionary stores objects against given keys, which can then be used for retrieval
- Immutable hash table. Look up objects using a key to get a value.

```
+ (id)dictionaryWithObjects:(NSArray *)values forKeys:(NSArray *)keys;
+ (id)dictionaryWithObjectsAndKeys:(id)firstObject, ...;
```

- Creation example:

```
NSDictionary *base = [NSDictionary dictionaryWithObjectsAndKeys:
                        [NSNumber numberWithInt:2], @“binary”,
                        [NSNumber numberWithInt:16], @“hexadecimal”, nil];
NSDictionary *ncr = @{
  @”Delhi”: [NSNumber numberWithInt: 1],
  @”Gurgoan”: [NSNumber numberWithInt:2],
  @”Noida” : [NSNumber numberWithInt:3]
} ;
```

NSDictionary

- Methods
  - - (int)count;
  - - (id)objectForKey:(id)key;
  - - (NSArray *)allKeys;
  - - (NSArray *)allValues;

NSMutableDictionary

- Changeable

+ (id)dictionaryWithObjects:(NSArray *)values forKeys:(NSArray *)keys;

+ (id)dictionaryWithObjectsAndKeys:(id)firstObject, ...;

- Creation :

+ (id)dictionary; //creates empty dictionary

- Methods

- (void)setObject:(id)anObject forKey:(id)key;

- (void)removeObjectForKey:(id)key;

- (void)removeAllObjects;

- (void)addEntriesFromDictionary:(NSDictionary *)otherDictionary;

BLOCKS

Blocks

- Blocks are a language-level feature, which allow you to create distinct segments of code that can be passed around to methods or functions as if they were values
- Blocks are Objective-C objects, which means they can be added to collections like NSArray or NSDictionary.
- They also have the ability to capture values from the enclosing scope

Blocks Syntax

- The syntax to define a block literal uses the caret symbol (^), like this:

```
void (^simpleBlock)(void) = ^{
NSLog(@"This is a block");
};
```

- Once you've declared and assigned a block variable, you can use it to invoke the block:

```
simpleBlock();
```

Blocks can also take arguments and return values just like methods and functions.

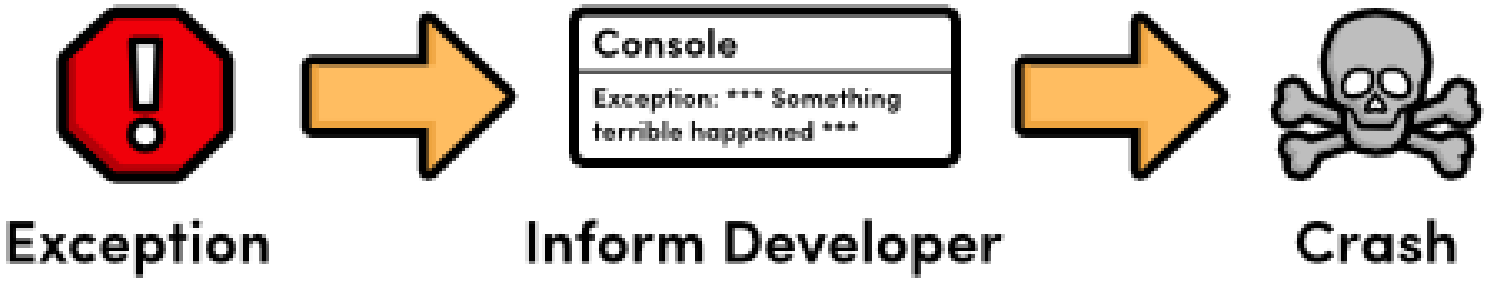
Block Example

```
double (^multiplyTwoValues)(double, double)
   = ^(double firstValue, double secondValue) {
return firstValue * secondValue;
};
double result = multiplyTwoValues(2,4);
NSLog(@"The result is %f", result);
```

Block Storage

- If you need to be able to change the value of a captured variable from within a block, you can use the __block storage type modifier on the original variable declaration.

Exception vs Error



Exception

- Exceptions are represented by the NSException class.
- NSException's three main properties are listed below.

Property	Description
name	An NSString that uniquely identifies the exception.
Reason	An NSString that contains a human-readable description of the exception.
userInfo	An NSDictionary whose key-value pairs contain extra information about the exception. This varies based on the type of exception

Exception - Example

```
@try  { NSString *car = inventory[selectedIndex];
NSLog(@"The selected car is: %@", car);  }
@catch(NSException *theException)
{    NSLog(@"An exception occurred: %@",
   theException.name);
NSLog(@"Here are some details: %@",
   theException.reason); }
@finally { NSLog(@"Executing finally block"); }
```