Objective-C

Курс лекций и семинаров для студентов,
желающих научиться программировать под iPhone

Осень-Зима 2013

Лекция №4

Типы данных : Стандартные классы Foundation, особенности использования, коллекции объектов.

Автор: Дмитрий Волков, iPhone Developer, Sibers

Sibers®

Что вы узнаете сегодня?

1. Скалярные и объектные типы данных (обзор и сравнение)
2. Литералы
3. Коллекции (обзор, общие правила, isEqual: и hash, Fast enumeration, Массивы, Словари, Множества)

Скалярные и Объектные типы данных

Дизайн языка Objective-C предусматривает разделение объектов на 2 типа:

- ***Объектые*** — типы, описывающие объекты, и обычно наследующиеся от **NSObject**. (***NSString***, ***NSArray***, ***NSNumber*** etc.)
- ***Скалярные*** — типы, предоставляемые языком C. Не являются объектами. (***int***, ***float***, ***double, void**** etc.)

Для скалярных типов во фреймворке Foundation используются собственные именования , объявленные посредством ***typedef***.

Скалярные и объектные типы данных

Foundation использует следующие именования для типов:

- ***typedef long NSIntger;***
- ***typedef unsigned int NSUInteger;***
- ***typedef double NSTimeInterval;***

Так как стандарт C не определяет специализированного типа для логических переменных (Boolean), в Objective-C используется свой тип **BOOL**:

- ***BOOL value = YES / BOOL value = NO;***
- ***typedef signed char BOOL;***
- ***typedef (BOOL)0 NO; typedef (BOOL)1 YES;***

Скалярные и объектные типы данных

Для использования скалярных типов в «объектом мире», Foundation предоставляет классы-обертки:

- ***NSValue*** — базовый класс-контейнер для скалярных или объектных данных. Используется для обертки над скалярными типами, указателями на данные известной длины или указателями на объекты.

- ***NSNumber*** — наследник NSValue. Конкретизирует работу с числовыми типами данных. Предоставляет объектную обертку над числовыми типами данных, а также преобразование от одного числового типа к другому. При преобразовании используются стандартные правила, используемые в C.

Скалярные и объектные типы данных

```
NSRange myRange = {4, 10};
NSValue *theValue = [NSValue valueWithBytes:&myRange
objCType:@encode(NSRange)];

char *myCString = "This is a string.";
NSValue *theValue = [NSValue valueWithBytes:&myCString
objCType:@encode(char **)];

float ten = 10.0;
NSNumber *tenFromFloat = [NSNumber numberWithFloat:ten];
```

Скалярные и объектные типы данных

```
float ten = 10.0;
NSNumber *tenFromFloat = [NSNumber numberWithFloat:ten];

NSMutableArray* array = [NSMutableArray array];

//неправильно
[array addObject:ten];
//правильно
[array addObject:tenFromFloat];
```

Скалярные и объектные типы данных: Ничто и пустота

Для представления несуществующих данных в Objective-C и Foundation используются следующие типы:

- ***NULL*** — наследие C. Нулевой указатель. ***NULL = (void*)0***
- ***Nil*** — нулевой указатель на класс. ***Nil = (Class)0***
- ***nil*** — нулевой указатель на объект. ***nil = (id)0***

- ***NSNull*** — объектная обертка для нулевых данных. Так как коллекции в **Foundation** не допускают хранения ***nil / Nil / NULL***, используется объект-синглтон ***[NSNull null]*** для описания пустых объектов в коллекциях.

Sibers®

Скалярные и объектные типы данных: Ничто и пустота

В отличие от многих языков, в Objective C обращение и посылка сообщений несуществующим объектам (nil / Nil) легальны и валидны.

Сообщение, если оно имеет возвращаемый тип, вернет 0 или nil / Nil.

Важно: на объект ***[NSNull null]*** это правило не распространяется. При посылке ему сообщения, на которое данный объект не может ответить, будет выброшено исключение.

Скалярные и объектные типы данных: Ничто и пустота

id object = nil;

[object doStuff]; //вполне себе легально

Class classObject = Nil;

[classObject doClassStaff];//тоже вполне правильно

NSMutableArray array = [NSMutableArray array];*

[array addObject:object];//выбросит исключение, коллекции не могут содержать nil

[array addObject:[NSNull null]];//правильно

Скалярные и объектные типы данных: Сравнение

- В Objective C используется своя система для определения равенства объектов.
- Класс NSObject объявляет метод ***isEqual:***, который используется для определения равенства объектов.
- Некоторые классы так же объявляют метод ***compare:***, используемый для определения порядка при сравнении объектов.
- По умолчанию, каждый класс сам определяет реализацию данного метода и логику равенства объектов.
- Скалярные типы данных и типы данных C сравниваются оператором ==.

Скалярные и объектные типы данных: Сравнение

По умолчанию, метод ***isEqual:*** в классе NSObject реализован простым сравнением указателей:

```
- (BOOL) isEqual:(id) anotherObject
{
    return self==anotherObject;
}
```

Некоторые типы данных, такие как NSString / NSNumber / NSData и т.д. определяют свои методы для сравнения:

isEqualToString:, isEqualToData:, isEqualToDate: …

Скалярные и объектные типы данных: Сравнение.

```
NSString* string = @”Hello”;
NSString* anotherString = @”Hello”;

[string isEqualToString:anotherString];//returns YES
string == anotherString; //returns NO

NSNumber* number = [NSNumber numberWithInt:10];
NSNumber* anotherNumber = [NSNumber numberWithInt:10];
NSNumber* lesserNumber = [NSNumber numberWithInt:9];

[number isEqualToNumber:anotherNumber];// YES
[number compare:anotherNumber];//NSOrderedSame
[number compare:lesserNumber];//NSOrderedDescending
```

Скалярные и объектные типы данных: Сравнение

```
Int value = 5;
Int anotherValue = 5;

value==anotherValue;//YES

id object = [[NSObject alloc] init];
id anotherObject = object;

object == anotherObject; //YES
[object isEqual:anotherObject];//YES
```

Литералы

Начиная с версии компилятора Clang 3.4, Objective C поддерживает литералы — более компактный стиль создания / доступа к объектам.

NSNumber:

```
NSNumber* number = [NSNumber numberWithInt:10];
NSNumber* number = @10;
NSNumber* number = @(10+20*5);
```

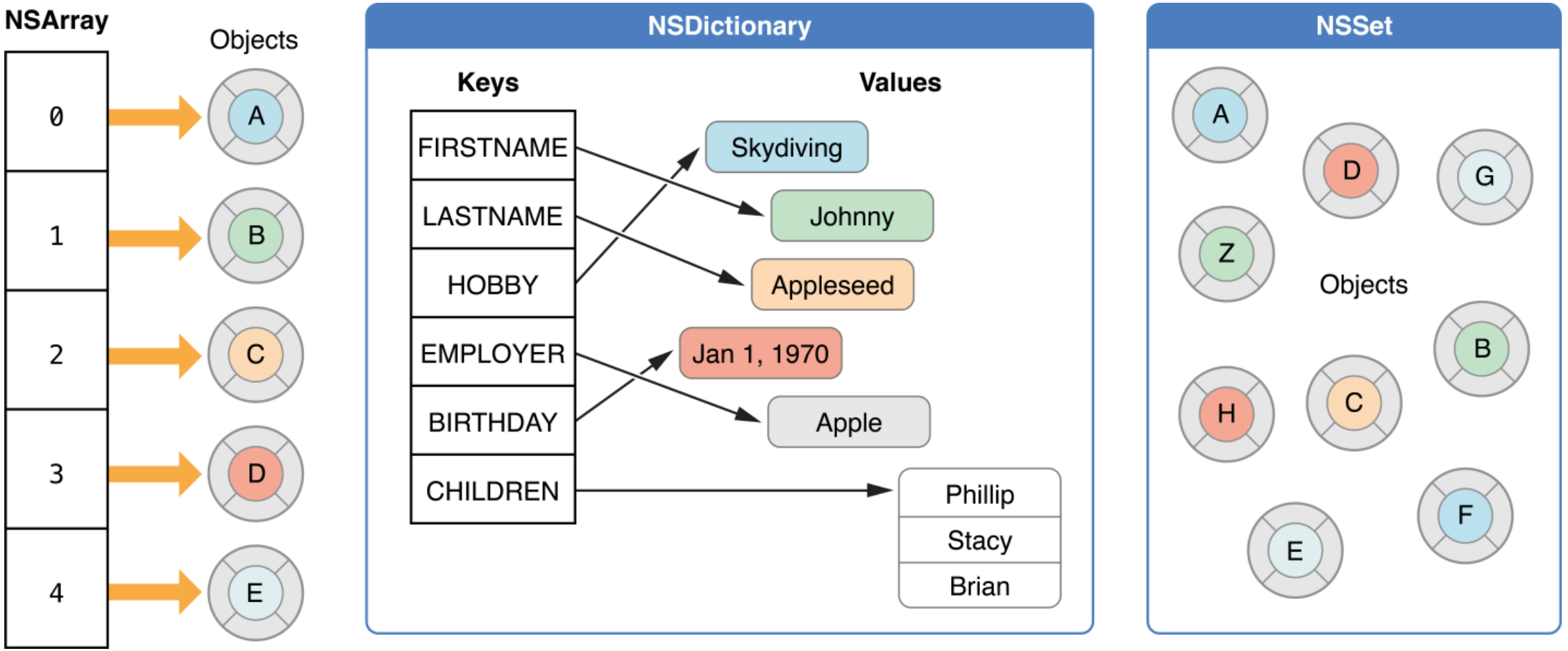
Коллекции

Фреймворк Foundation предоставляет классы-коллекции для управления группами объектов.

Фреймворк предоставляет 3 типа коллекций:

- ***Массивы*** — содержит упорядоченный список объектов
- ***Словари*** — содержит пары ключ-значение
- ***Множества*** — неупорядоченное множество уникальных элементов

Коллекции


NSArray
Objects
0
1
2
3
4
A
B
C
D
E
NSDictionary
Keys
Values
FIRSTNAME
LASTNAME
HOBBY
EMPLOYER
BIRTHDAY
CHILDREN
Skydiving
Johnny
Appleseed
Jan 1, 1970
Apple
Phillip
Stacy
Brian
NSSet
A
D
G
Z
Objects
B
H
C
F
E

Коллекции: Общие правила

- Стандартные коллекции Foundation работают **ТОЛЬКО** с объектными типами (т.е. могут содержать в себе только объекты типа ***id***)
- Коллекции не типизированы — нельзя указать коллекции, что она содержит объекты определенного класса. Все коллекции содержат объекты типа ***id***.
- Коллекции не могут содержать ***nil / Nil / Null***. Для этого используется объект-синглтон ***NSNull***.
- При добавлении объекта в коллекцию ему посылается сообщение ***retain***. При удалении объекта из коллекции, либо при уничтожении коллекции ее объектам посылается сообщение ***release***.

Коллекции: Общие правила

```
NSMutableArray* array = [NSMutableArray array];
id string = @"String";
id nilObject = nil;
[array addObject:string];//OK
[array addObject:nilObject];//WRONG – Runtime exception
[array addObject:5];//WRONG – compiler error
[array addObject:@5];//OK
[array addObject:[NSNull null]];//OK
```

Коллекции: isEqual: и hash

При добавлении в коллекции собственных объектов, ВСЕГДА следует переопределять либо оба метода ***isEqual:*** и ***hash***, либо не переопределять их вообще.

```
- (BOOL) isEqual:(id) anotherObject
{
    return [self.name isEqualToString:anotherObject.name] &&
self.age==anotherObject.age;
}

(NSUInteger) hash
{
    Return self.name.hash ^ self.age;
}
```

Sibers®

Коллекции: Fast enumeration

- Все классы-коллекции реализуют протокол ***NSFastEnumeration***, позволяющий иттерировать контент коллекций намного быстрее.
- Так же, любой объект может реализовать этот протокол и использовать fast enumeration синтаксис для доступа к своим элементам.
- Изменение коллекции по время прохода по ее элементам генерирует исключение

```
NSArray* array = [NSArray array];
for(id object in array)
{
    NSLog(@"%@",object);
}
```

Коллекции: Массивы

Массивы — коллекции с упорядоченным расположением элементов.

Представлены двумя классами:

- ***NSArray*** — неизменяемый массив. Нельзя добавлять / удалять элементы после создания. Сами объекты модифицировать можно.
- ***NSMutableArray*** — наследник NSArray. Изменяемый массив, можно добавлять, удалять, перемещать элементы.

Ничего особенного, старый добрый массив объектов.

Коллекции: Массивы

```
NSArray* array = [NSArray arrayWithObjects:@”1”, @”2”, nil];
id str1 = array[0];// @”1”
id str2 = [array objectAtIndex:1]; //@”2”

NSMutableArray* mutableArray= [NSMutableArray array];
[mutableArray addObject:@”1”];
[mutableArray addObject:@”2”];
[mutableArray removeObjectAtIndex:0];
[mutableArray removeObject:@”2”];
```

Коллекции: Массивы, литералы

Для работы с массивами так же доступны литералы:

- ***Создание массива:***

```
NSArray* array = [NSArray arrayWithObjects:@”1”, @”2”, @”3”, nil];
NSArray* array = @[@”1”, @”2”, @”3”];
```

- ***Доступ к элементам:***

```
id object = [array objectAtIndex:0];
id object = array[0];
Array[0] = @”3”;
```

Коллекции: Словари

Словари — коллекции, содержащие пары ключ-значение.

Ключ — «индекс», по которому можно получить ***объект -«значение»***.

Представлены двумя классами:

- ***NSDictionary*** — неизменяемый словарь. Нельзя добавлять новые пары ключ-значение после создания. Объекты, извлекаемые по ключу, можно изменять.
- ***NSMutableDictionary*** — изменяемый словарь. Можно добавлять новые пары ключ-значение, а так же удалять значения по ключу.

Коллекции: Словари

ВАЖНО: Словари ***копируют*** свои ключи и посылают ***retain*** сообщения значениям.

```
NSDictionary* dictionary = [NSDictionary dictionaryWithObjectsAndKeys:
@”John”, @”name”, nil];
id name = [dictionary objectForKey:@”name”];//@”John”

NSMutableDictionary* mutableDictionary = [NSMutableDictionary dictionary];
[mutableDictionary setObject:@”John”  forKey:@”name”];
id name = [mutableDictionary objectForKey:@”name”];
[mutableDictionary removeObjectForKey:@”name”];
```

Коллекции: Словари, литералы.

Для работы со словарями так же можно использовать литералы:

```
NSDictionary* dictionary = [NSDictionary dictionaryWithObjectsAndKeys:
@”John”, @”name”, nil];

NSDictionary* dictionary = @{@”name” : @”John”};

id name = [dictionary objectForKey:@”name”];//@”John”

id name = dictionary[@”name”];


NSMutableDictionary* mutableDictionary = @{@”name” : @”John”};

mutableDictionary[@”name”] = @”Steve”;
```

Коллекции: Множества

Множества — неупорядоченные коллекции уникальных объектов. Если объект уже был добавлен во множество, повторное его добавление ни на что не повлияет.

Представлены целым набором классов:

- ***NSSet*** — неизменяемое множество.
- ***NSMutableSet*** —изменяемое множество.
- ***NSCountedSet*** — множество, подсчитывающее, сколько уникальных объектов в него было добавлено.
- ***NSIndexSet / NSMutableIndexSet*** — множество, оптимизированное для работы с индексами (числами, диапазонами чисел)
- ***NSOrderedSet / NSMutableOrderedSet*** — множество, комбинирующее в себе уникальность объектов и их упорядоченность.

Коллекции: Множества

Множества поддерживают основные операции реляционной алгебры, что позволяет легко фильтровать объекты, которые они содержат.

Поддерживаются операции:

- Объединение (***Union***)
- Пересечение (***Intersection***)
- Вычитание (***Minus***)
- Проверка, добавлен ли объект во множество

ВАЖНО: Объекты, содержащиеся во множествах не должны менять свое внутреннее состояние так, что изменится результат функции hash этого объекта.

Множества не поддерживают литералы.

Заключение

1. Foundation framework предоставляет разработчикам удобные коллекции для управления группами объектов.
2. Objective-C разделяет объекты на 2 типа: объектные и примитивные.
3. Коллекции могут работать только с объектными типами. Но Foundation предоставляет отличные абстракции для переноса примитивных типов в объектный мир
4. Коллекции управляют памятью объектов, которые они содержат. Не стоит забывать о соответствующих сообщениях retain / release для объектов, хранимых в коллекциях

В следующей лекции:

Компоненты UI:

UIView

Иерархия UI элементов

Responder chain

Спасибо за внимание!

Жду ваших вопросов.