数字对象

数字对象

- Foundation框架简介
- NSInteger
- NSNumber和基本数据类型互相转换

数字对象

Foundation框架简介

- 框架是由许多类、方法、函数、文档按照一定的逻辑组织起来的集合
- Foundation框架包括数字、字符串、对象集合、日期和时间、内存和文件管理、对象存储等

数字对象

NSInteger

- 32位上等于int
- 64位上等于long

数字对象

NSNumber和基本数据类型互相转换

- NSInteger
- int、short、long、long long
- float、double
- BOOL

字符串

字符串

- 初始化
- 常用操作
- NSMutableString

字符串

- 初始化

```
NSString *str1 = @"你好"; //字面量初始化
NSString *str2 = [NSString stringWithFormat:"@%@%i岁",@"小明",18];
//format拼接字符串
```

字符串

- 长度

```
long len = str1.length;
NSLog(@"字符串str1长度为%li",len);
```

字符串

- 转换成数值类型

```
NSLog(@"%lli",[@"12345" longLongValue]);
NSLog(@"%f",[@"12345.5555" doubleValue]);
```

字符串

NSRange

- NSRange结构体不是对象不需要加*
- loc表示范围起点索引从0开始,len表示范围长度

```
NSRange range = NSMakeRange(5, 3);
NSLog(@"%@",NSStringFromRange(range));
```

字符串

- 子串查找

```
NSRange range = [@"abcdefg" rangeOfString:@"cde"];
NSLog(@"%@",NSStringFromRange(range));
NSLog(@"%@",[@"abcdefg" substringFromIndex:2]);
NSLog(@"%@",[@"abcdefg" substringToIndex:3]);
NSLog(@"%@",[@"abcdefg" substringWithRange:range]);
```

字符串

- 是否包含子串

```
NSLog(@"%i",[@"abcdefg" hasPrefix:@"abc"]); //是否以指定子串开头
NSLog(@"%i",[@"abcdefg" hasSuffix:@"efg"]); //是否以指定子串结尾
NSLog(@"%i",[@"abcdefg" containsString:@"cde"]); //是否包含子串，谨慎使用,只能在iOS8上使用
```

字符串

- 比较字符串

```
NSLog(@"%i",[@"abc" isEqualToString:@"ABC"]);
NSLog(@"%i",[@"abc" compare:@"abd"] == NSOrderedAscending);
```

字符串

- 替换字符串

```
NSLog(@"%@",[@"abcabc" stringByReplacingOccurrencesOfString:@"b"
  withString:@"B"]);
NSLog(@"%@",[@"abcabc" stringByReplacingCharactersInRange:NSMakeRange(0,
  3) withString:@"ABC"]);
```

字符串

- 追加子串(返回新的字符串)

```
NSLog(@"%@",[@"abc" stringByAppendingString:@"efg"]);
NSLog(@"%@",[@"abc" stringByAppendingFormat:@"%iefg",2]);
```

字符串

- 拆分合并

```
NSArray *array = [@"abcdef" componentsSeparatedByString:@"c"];
NSLog(@"%@",[array componentsJoinedByString:@"c"]);
```

字符串

NSMutableString

- NSString内容不可变,NSMutableString内容可变
- 可以对字符串内容进行append、insert、replace和delete操作

数组

数组

- 定义
- 初始化
- 常用操作
- NSMutableArray

数组

定义

- 存储有序对象的集合
- 可以存储相同的对象
- 它只能保存对象，基本数据类型需要转换成NSNumber对象才能存储

数组

- 初始化

```
NSArray *array = [NSArray array]; //空数组
array = [NSArray arrayWithObjects:@"小明",@(18),@(YES),nil];
array = @[@"小王",@(16),@(NO)]; //使用快捷初始化方式
```

数组

- 元素个数

```
NSLog(@"array有%li个元素",array.count);
```

数组

- 访问元素

```
int index = 5;
if (index < array.count) { //判断索引是否小于数组的元素，访问超过范围的索引会
  crash
    NSLog(@"%@",[array objectAtIndex:index]);
    NSLog(@"%@",array[index]); //如果是数值类型的，取出的为NSNumber，调用相应
  的方法转换成基本数据类型
}
```

数组

- 追加(返回新的数组)

```
array = [array arrayByAddingObject:@(3)];
array = [array arrayByAddingObjectsFromArray:@[@(2),@(1)]];
```

数组

- 是否包含指定对象

//NSNumber根据数值大小判断,NSString等默认根据内存地址判断，最终调用isEqual方法去判断

```
NSLog(@"%d",[array containsObject:@(2)]);
NSLog(@"%d",[array containsObject:@"小明"]);
```

数组

遍历数组

- 普通for循环
- for...in
- enumerateObjectsUsingBlock

数组

NSMutableArray

- NSArray数组元素不可变, NSMutableArray数组元素可变
- 可以对数组元素进行add、insert、replace和remove操作

字典

字典

- 定义
- 初始化
- 常用操作
- NSMutableDictionary

字典

定义

- 存储无序键值对的集合
- 键不能相同
- 它只能保存对象，基本数据类型需要转换成NSNumber对象才能存储

字典

- 初始化

```
NSDictionary *dictionary = [NSDictionary dictionary];
dictionary = [NSDictionary dictionaryWithObjectsAndKeys:@"小明
  ",@"name",@(18),@"age", nil];
dictionary = @{@"name":@"小明",@"age":@(18)};
```

字典

- 元素个数

```
NSLog(@"字典元素个数为%li",dictionary.count);
```

字典

- 访问元素

```
NSString *name = [dictionary objectForKey:@"name"];
int age = [dictionary[@"age"] intValue];
NSLog(@"%@的年龄为%i",name,age);
```

字典

- 遍历

```
for (NSObject *object in [dictionary keyEnumerator) { //遍历key
}
for (NSObject *object in [dictionary objectEnumerator]) { //遍历value
}
[dictionary enumerateKeysAndObjectsUsingBlock:^(id  _Nonnull key, id
  _Nonnull obj, BOOL * _Nonnull stop) {
}];
```

字典

NSMutableDictionary

- NSDictionary元素不可变, NSMutableDictionary元素可变
- 可以对元素进行append、insert、replace和delete操作

日期和时间

日期和时间

- NSDate
- NSCalendar

日期和时间

定义

- NSDate表示一个具体的时间点
- NSCalendar用于处理时间相关问题。比如比较时间前后、计算日期所在的周等。

日期和时间

- NSDate初始化

```
//创建NSDate对象
NSDate *now = [NSDate date];
NSDate *before = [[NSDate date]dateByAddingTimeInterval:-1000000];
//创建日历对象
NSCalendar *calendar = [NSCalendar currentCalendar];
```

日期和时间

- 获取当前系统的时间戳

```
NSLog(@"距离1970年%f毫秒",[[NSDate date]timeIntervalSince1970]);
NSLog(@"now - before = %f毫秒",[now timeIntervalSinceDate:before]);
```

日期和时间

- 格式化日期

```
//NSDate -> NSString
NSDateFormatter *formatter = [[NSDateFormatter alloc]init];
formatter.dateFormat = @"yyyy-MM-dd HH:mm:ss"; //年-月-日 时：分：秒
NSString *nowStr = [formatter stringFromDate:now];
NSLog(@"当前格式化时间为:%@",nowStr);
//NSString -> NSDate
NSDate *date = [formatter dateFromString:nowStr];
NSLog(@"%@",date);
```

日期和时间

- 时间戳转换为NSDate

日期和时间

- 使用NSCalendar获取年月日时分秒

日期和时间

- 使用NSCalendar比较两个日期的差距

归档和解档

归档和解档

- 定义
- 单个对象归档和解档
- 多个对象
- 自定义对象

归档和解档

定义

- 归档就是将内存中的对象写入一个文件中
- 解档就是读取文件转换为内存中对象

归档和解档

Foundation对象归档

```
NSArray *array = @[@"张三",@"李四",@"王五"];
NSString *file = [NSHomeDirectory()
stringByAppendingPathComponent:@"array.src"];
BOOL success = [NSKeyedArchiver archiveRootObject:array toFile:file];
if (success) {
    NSLog(@"保存成功");
}
```

归档和解档

Foundation对象解档

```
NSArray *array = [NSKeyedUnarchiver unarchiveObjectWithFile:file];
NSLog(@"%@",array);
```

归档和解档

NSData和NSMutableData

- NSData是以二进制存储信息的，用于把对象信息转换为二进制数据信息
- NSMutableData是NSData的可变形式

归档和解档

多个Foundation对象归档为一个文件

```
NSMutableData *data = [NSMutableData data];
NSKeyedArchiver *archiver = [[NSKeyedArchiver
alloc]initForWritingWithMutableData:data];
[archiver encodeObject:array forKey:@"array"];
[archiver encodeInt:18 forKey:@"age"];
[archiver encodeObject:@"Jack" forKey:@"name"];
[archiver finishEncoding];
[data writeToFile:file atomically:YES];
```

归档和解档

多个Foundation对象解档

```
data = [[NSMutableData alloc]initWithContentsOfFile:file];
NSKeyedUnarchiver *unarchiver = [[NSKeyedUnarchiver
alloc]initForReadingWithData:data];
NSArray *array = [unarchiver decodeObjectForKey:@"array"];
int value = [unarchiver decodeIntForKey:@"age"];
```

归档和解档

自定义对象归档和解档

- 需要实现NSCoding协议
- encodeWithCoder和initWithCoder两个协议方法

文件管理

文件管理

- 文件路径处理
- 常用文件操作
- 获取文件的属性

文件管理

文件路径处理

```
NSString *filePath = [NSHomeDirectory()
stringByAppendingPathComponent:@"file.text"];
NSLog(@"%@",filePath.pathComponents); //返回路径的组成部分
NSLog(@"%@",filePath.lastPathComponent); //返回路径的最后组成部分
NSLog(@"%@",filePath.pathExtension); //返回路径的扩展名
```

文件管理

NSFileManager

- 文件管理类，可以对文件进行操作：创建、赋值、剪切和删除
- [NSFileManager defaultManager]使用单例获取它的对象

文件管理

创建文件

```
BOOL success = [manager createFileAtPath:filePath contents:data
attributes:nil];
if (success) {
    NSLog(@"文件创建成功");
} else {
    NSLog(@"文件创建失败");
}
```

文件管理

创建文件夹

```
NSString *folderPath = [NSHomeDirectory()
stringByAppendingPathComponent:@"folder"];
[manager createDirectoryAtPath:folderPath
withIntermediateDirectories:YES attributes:nil error:nil];
```

文件管理

读取文件

```
NSData *data = [manager contentsAtPath:filePath];
NSString *s = [[NSString alloc]initWithData:data
encoding:NSUTF8StringEncoding];
NSLog(@"%@",s);
```

文件管理

移动、复制和删除文件

- moveItemAtPath
- copyItemAtPath
- removeItemAtPath

文件管理

获取文件属性

```
NSDictionary *fileAttributes = [manager
  attributesOfItemAtPath:targetPath error:nil];
NSLog(@"文件的属性:%@",fileAttributes);
```

KVC和KVO

KVC和KVO

- 定义
- KVC
- KVO

KVC和KVO

定义

- KVC：就是可以暴力的去get/set类的私有属性，同时还有强大的键值路径对更深层次的属性进行操作
- KVO：监听类中属性值变化

KVC和KVO

KVC

- setValue:forKey 设置指定key的属性值，setValue:forKeyPath设置指定路径的属性
- valueForKey 获取指定key的属性值，valueForKeyPath 获取指定keyPath的属性值

KVC和KVO

KVO

- 用来监听类中属性值的变化
- 观察者模式

NSNotification

NSNotification

- 发送广播消息
- 注册监听器
- 处理接收到的消息

NSPredicate

NSPredicate

- 定义
- 使用方法
- 常用过滤条件

NSPredicate

定义

- 谓词
- 从数组中过滤出我们想要的数据

NSPredicate

使用方法

```
NSPredicate *predicate = [NSPredicate predicateWithFormat:@"age<%d",30];
NSArray *results = [array filteredArrayUsingPredicate:predicate];
```

NSPredicate

常用过滤条件

- 逻辑表达式 =、<、>、>=...
- in
- BEGINSWITH
- ENDSWITH
- CONTAINS
- like