# NSString

## 创建字符串。

NSString \*astring = @"This is a String!";

## 使用变量初始化

 NSString \*name = @"Ivan!";

NSString \*astring = [[NSString stringWithFormat:@”My name is %@!”,name]];

NSLog(@"astring:%@",astring);

[astring release];

## 判断是否包含某字符串

### 检查字符串是否以另一个字符串开头- (BOOL) hasPrefix: (NSString \*) aString;

NSString \*String1 = @"NSStringInformation.txt";

[String1 hasPrefix:@"NSString"] == 1 ? NSLog(@"YES") : NSLog(@"NO");

[String1 hasSuffix:@".txt"] == 1 ? NSLog(@"YES") : NSLog(@"NO");

### 是否包含其它字符

NSString \*astring = [[NSString alloc] initWithString:@"This is a String!"];

Boolean contains = [astring rangeOfString:@”This”].length>0;

## 从文件读取字符串:initWithContentsOfFile方法

NSString \*path = @"astring.text";

NSString \*astring = [[NSString alloc] initWithContentsOfFile:path];

NSLog(@"astring:%@",astring);

[astring release];

## 写字符串到文件:writeToFile方法

NSString \*astring = [[NSString alloc] initWithString:@"This is a String!"];

NSLog(@"astring:%@",astring);

NSString \*path = @"astring.text";

[astring writeToFile: path atomically: YES];

[astring release];

## 比较两个字符串

### isEqualToString方法

NSString \*astring01 = @"This is a String!";

NSString \*astring02 = @"This is a String!";

BOOL result = [astring01 isEqualToString:astring02];

NSLog(@"result:%d",result);

### compare方法(comparer返回的三种值)

NSString \*astring01 = @"This is a String!";

NSString \*astring02 = @"This is a String!";

BOOL result = [astring01 compare:astring02] = = NSOrderedSame;

NSLog(@"result:%d",result);

### NSOrderedSame判断两者内容是否相同

NSString \*astring01 = @"This is a String!";

NSString \*astring02 = @"this is a String!";

BOOL result = [astring01 compare:astring02] = = NSOrderedAscending;

NSLog(@"result:%d",result);

### 不考虑大小写比较字符串1

NSString \*astring01 = @"this is a String!";

NSString \*astring02 = @"This is a String!";

BOOL result = [astring01 caseInsensitiveCompare:astring02] = = NSOrderedSame;

NSLog(@"result:%d",result);

## 改变字符串的大小写

NSString \*string1 = @"A String";

NSString \*string2 = @"String";

NSLog(@"string1:%@",[string1 uppercaseString]);//大写

NSLog(@"string2:%@",[string2 lowercaseString]);//小写

NSLog(@"string2:%@",[string2 capitalizedString]);//首字母大写

## 去除两端空格

NSString \*temp = [textField.text stringByTrimmingCharactersInSet:[NSCharacterSet whitespaceCharacterSet]];

## 去除两端空格和回车

NSString \*text = [temp stringByTrimmingCharactersInSet:[NSCharacterSet whitespaceAndNewlineCharacterSet ]];

## 在串中搜索子串

NSString \*string1 = @"This is a string";

NSString \*string2 = @"string";

NSRange range = [string1 rangeOfString:string2];

int location = range.location;

int lenght = range.length;

NSString \*astring = [[NSString alloc] initWithString:[NSString stringWithFormat:@"Location:%i,Leight:%i",location,leight]];

NSLog(@"astring:%@",astring);

[astring release];

## 替换字符串

NSString \*astring01 = @"hello 中国";

NSString \* new = [astring01 stringByReplacingOccurrencesOfString:@”中国” withString:@"达内"];

NSLog(new);

## 分割字符串成数组

NSString \*s = @"a b d e f";

NSArray \*arr = [s componentsSeparatedByString:@" "];

NSLog(@"count = %d",[arr count]);

## 字符串数组拼接成字符串

NSArray \*pathArray = [NSArray arrayWithObjects:@"here",

@"be", @"dragons", nil];

NSLog(@"%@",[pathArray componentsJoinedByString:@""]);

## 抽取子串

### -substringToIndex: 从字符串的开头一直截取到指定的位置，但不包括该位置的字符

NSString \*string1 = @"This is a string";

NSString \*string2 = [string1 substringToIndex:3];

NSLog(@"string2:%@",string2);

### -substringFromIndex: 以指定位置开始（包括指定位置的字符），并包括之后的全部字符

NSString \*string1 = @"This is a string";

NSString \*string2 = [string1 substringFromIndex:3];

NSLog(@"string2:%@",string2);

### -substringWithRange: //按照所给出的位置，长度，任意地从字符串中截取子串

NSString \*string1 = @"This is a string";

NSString \*string2 = [string1 substringWithRange:NSMakeRange(0, 4)];

NSLog(@"string2:%@",string2);

# NSMutableString

## 在已有字符串后面添加字符

//appendString: and appendFormat:

NSMutableString \*String1 = [[NSMutableString alloc] initWithString:@"This is a NSMutableString"];

//[String1 appendString:@", I will be adding some character"];

[String1 appendFormat:[NSString stringWithFormat:@", I will be adding some character"]];

NSLog(@"String1:%@",String1);

\*/

## 在已有字符串中按照所给出范围和长度删除字符

//deleteCharactersInRange:

NSMutableString \*String1 = [[NSMutableString alloc] initWithString:@"This is a NSMutableString"];

[String1 deleteCharactersInRange:NSMakeRange(0, 5)];

NSLog(@"String1:%@",String1);

## 在已有字符串后面在所指定的位置中插入给出的字符串

//-insertString: atIndex:

NSMutableString \*String1 = [[NSMutableString alloc] initWithString:@"This is a NSMutableString"];

[String1 insertString:@"Hi! " atIndex:0];

NSLog(@"String1:%@",String1);

## 将已有的换成其它的字符串

//-setString:

NSMutableString \*String1 = [[NSMutableString alloc] initWithString:@"This is a NSMutableString"];

[String1 setString:@"Hello Word!"];

NSLog(@"String1:%@",String1);

## 按照所给出的范围，和字符串替换的原有的字符

NSMutableString \*String1 = [[NSMutableString alloc] initWithString:@"This is a NSMutableString"];

[String1 replaceCharactersInRange:NSMakeRange(0, 4) withString:@"That"];

NSLog(@"String1:%@",String1);

# NSArray

## 创建数组

NSArray \*array = [[NSArray alloc] initWithObjects:

@"One",@"Two",@"Three",@"Four",nil];

[array release];

NSLog(@"self.dataArray cound:%d",[array count]);

## 获取指定索引处的对象;

NSLog(@"self.dataArray cound 2:%@",[array objectAtIndex:2]);

## 从一个数组拷贝数据到另一数组

//arrayWithArray:

//NSArray \*array1 = [[NSArray alloc] init];

NSMutableArray \*MutableArray = [[NSMutableArray alloc] init];

NSArray \*array = [NSArray arrayWithObjects:

@"a",@"b",@"c",nil];

NSLog(@"array:%@",array);

MutableArray = [NSMutableArray arrayWithArray:array];

NSLog(@"MutableArray:%@",MutableArray);

array1 = [NSArray arrayWithArray:array];

NSLog(@"array1:%@",array1);

## 快速便利数组

//NSMutableArray \*newArray = [[NSMutableArray alloc] init];

NSArray \*oldArray = [NSArray arrayWithObjects:

@"a",@"b",@"c",@"d",@"e",@"f",@"g",@"h",nil];

NSLog(@"oldArray:%@",oldArray);

for(id obj in oldArray)

{

[newArray addObject: obj];

}

//

NSLog(@"newArray:%@", newArray);

[newArray release];

# NSMutableArray

## 在数组末尾添加对象

//- (void) addObject: (id) anObject;

//NSMutableArray \*array = [NSMutableArray arrayWithObjects:

@"One",@"Two",@"Three",nil];

[array addObject:@"Four"];

NSLog(@"array:%@",array);

## 删除数组中指定索引处对象

//-(void) removeObjectAtIndex: (unsigned) index;

//NSMutableArray \*array = [NSMutableArray arrayWithObjects:

@"One",@"Two",@"Three",nil];

[array removeObjectAtIndex:1];

NSLog(@"array:%@",array);

# NSDictionary

## 创建字典

//- (id) initWithObjectsAndKeys;

NSDictionary \*dictionary = [[NSDictionary alloc] initWithObjectsAndKeys:@"One",@"1",@"Two",@"2",@"Three",@"3",nil];

NSString \*string = [dictionary objectForKey:@"One"];

NSLog(@"string:%@",string);

NSLog(@"dictionary:%@",dictionary);

[dictionary release];

# NSMutableDictionary

## 创建

NSMutableDictionary \*dictionary = [NSMutableDictionary dictionary];

## 添加字典

[dictionary setObject:@"One" forKey:@"1"];

[dictionary setObject:@"Two" forKey:@"2"];

[dictionary setObject:@"Three" forKey:@"3"];

[dictionary setObject:@"Four" forKey:@"4"];

NSLog(@"dictionary:%@",dictionary);

## 删除指定的字典

[dictionary removeObjectForKey:@"3"];

NSLog(@"dictionary:%@",dictionary);

# NSValue（对任何对象进行包装）

## 将NSRect放入NSArray中

NSMutableArray \*array = [[NSMutableArray alloc] init];

NSValue \*value;

CGRect rect = CGRectMake(0, 0, 320, 480);

value = [NSValue valueWithBytes:&rect objCType:@encode(CGRect)];

[array addObject:value];

NSLog(@"array:%@",array);

## 从Array中提取

value = [array objectAtIndex:0];

[value getValue:&rect];

NSLog(@"value:%@",value);

# NSNumber

**NSNumber**

+ (NSNumber \*)numberWithInt:(int)value;

+ (NSNumber \*)numberWithDouble:(double)value;

- (int)intValue;

- (double)doubleValue;

NSNumber可以将基本数据类型包装起来，形成一个对象，这样就可以给其发送消息，装入NSArray中等等。

NSNumber \* intNumber=[NSNumber numberWithInt：100]；

NSNumber \*floatNumber=[NSNUmber numberWithFloat:100.00];

int i=[intNumber intValue]；

if([intNumber isEqualToNumber:floatNumber]) ....

NSNumber继承NSObject ，可以使用比较 compare： isEqual等消息

# 数学常用方法

## 数学常量：

#define M\_E         2.71828182845904523536028747135266250   // e #define M\_LOG2E     1.44269504088896340735992468100189214   // log 2e #define M\_LOG10E    0.434294481903251827651128918916605082  // log 10e #define M\_LN2       0.693147180559945309417232121458176568  // log e2 #define M\_LN10      2.30258509299404568401799145468436421   // log e10 #define M\_PI        3.14159265358979323846264338327950288   // pi #define M\_PI\_2      1.57079632679489661923132169163975144   // pi/2 #define M\_PI\_4      0.785398163397448309615660845819875721  // pi/4 #define M\_1\_PI      0.318309886183790671537767526745028724  // 1/pi #define M\_2\_PI      0.636619772367581343075535053490057448  // 2/pi #define M\_2\_SQRTPI  1.12837916709551257389615890312154517   // 2/sqrt(pi) #define M\_SQRT2     1.41421356237309504880168872420969808   // sqrt(2) #define M\_SQRT1\_2   0.707106781186547524400844362104849039  // 1/sqrt(2)

## 常用函数：

### 指数运算

 NSLog(@"%.f", pow(3,2) ); //result 9 NSLog(@"%.f", pow(3,3) ); //result 27

### 开平方运算 （计算两点间的距离时用到）

NSLog(@"%.f", sqrt(16) ); //result 4 NSLog(@"%.f", sqrt(81) ); //result 9

### 上舍入

NSLog(@"res: %.f", ceil(3.000000000001)); //result 4 NSLog(@"res: %.f", ceil(3.00)); //result 3

### 下舍入

 NSLog(@"res: %.f", floor(3.000000000001)); //result 3 NSLog(@"res: %.f", floor(3.9999999)); //result 3

### 四舍五入

NSLog(@"res: %.f", round(3.5)); //result 4 NSLog(@"res: %.f", round(3.46)); //result 3 NSLog(@"res: %.f", round(-3.5)); //NB: this one returns -4

### 最小值

NSLog(@"res: %.f", fmin(5,10)); //result 5

### 最大值

NSLog(@"res: %.f", fmax(5,10)); //result 10

|  |
| --- |
| 绝对值 NSLog(@"res: %.f", fabs(10)); //result 10 NSLog(@"res: %.f", fabs(-10)); //result 10 |

# NSDate

## 得到当前的日期

NSDate \*date = [NSDate date];

## 日期之间比较可用以下方法

    - (BOOL)isEqualToDate:(NSDate \*)otherDate;// 与otherDate比较，相同返回YES

    - (NSDate \*)earlierDate:(NSDate \*)anotherDate;// 与anotherDate比较，返回较早的那个日期

    - (NSDate \*)laterDate:(NSDate \*)anotherDate;//与anotherDate比较，返回较晚的那个日期

## 将日期转换成字符串

NSLog(@”date = %@”,[data description]);

## 设置日期显示格式

NSDateFormatter \*formatter =[[[NSDateFormatter alloc] init] autorelease];

[formatter setTimeStyle:NSDateFormatterFullStyle]; //设置几种默认的显示效果

[formatter setDateFormat:@"yyyy-MM-dd HH:mm:ss a"];//设置自定义的显示效果

NSLog([formatter stringFromDate:date]);

# NSData

## NSData－> NSString

NSString \*aString = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];

## NSString－>NSData

NSString \*aString = @"1234abcd";

NSData \*aData = [aString dataUsingEncoding: NSUTF8StringEncoding];