리터럴

‘@’라는 심볼로 X-code 4.4부터 literal이라는 개념이 추가 되었습니다.

NSNumber

여러숫자 타입들(integer,long,double,float,char,BOOL)의 값들을 NSNumber 의 객체형태로 감쌀 수 있도록 해주는 것이 @이다.

예시

```
    #import <Foundation/Foundation.h>
    void main(int argc , const char *argv[]){
/*
NSNumber *charValue = [NSNumber numberWithChar ='Y'];
이렇게 Y라는 글자를  NSNumber클래스의 numberWithChar라는 클래스메서드를 사용해서 값을 객체화후에 넣어줘야 합니다.
*/

NSNumber *charValue =@'Y';
//Y라는 것을 문자로 인식 후에 객체로 바로 감싸서 charValue에 집어넣습니다.


 return 0;
}
```

NSArray

NSArray는 객체형으로 값을 받기때문에 들어가는 것이 객체형으로 만들어 넣어줘야 합니다.

예시

```
#import <Foundation/Foundation.h>
void main(int argc , const char *argv[]){

/*
클래스메서드의 매개변수타입이 객체형으로 들어와야하므로 리터럴을 사용해서 객체형으로 감싸준 뒤 넣어주는 것입니다.
*/
NSArray *persons = [NSArray arrayWithObjects :@"jinho",@"sangyeol",@"junho",nil];

//NSArray 또한 결국 객체를 받아 객체형의 배열을 persons에 집어넣는 것이므로 리터럴로 표현해 줄 수 있습니다.

NSArray *persons = @[@"jinho",@"sangyeol",@"junho"];

 return 0;
}
```

warning
첫번째방법으로한다면 만약 두번째 @"sangyeol"이라는 객체가 nil이라면
nil전까지의 객체만 들어가지만
두번째방법으로하면 예외로 오류를 나타낼 것입니다.

NSDictionary

객체형을 받는 딕셔너리클래스타입 또한 클래스 메서드를 통하여 객체를 생성하고 또한 그에 필요한 매개변수로 객체로 받아야 합니다.
그렇게하면 그 전에 넣어야할 값들을 객체형으로 만들어주거나 메소드안에서 메소드를 호출하여야합니다.

하지만 그건 너무 번거로운 일이고 이 또한 간편하게 리터럴을 사용해서 코드를 간편하게 해줄 수 있습니다.

예시

```
#import <Foundation/Foundation.h>
void main(int argc , const char *argv[]){


NSDictionary* personDic =
 [NSDictionary dictionaryWithObjectsAndKeys:
   @"jinho", @"friend1",@"sangyeol", @"friend2",nil];


//NSArray와 달리 @{} 객체를 만들고 그안에서 @를써서 객체로 다시한번 감싸줄 수 있습니다.
NSDictionary* personDic = @{@"friend1" : @"jinho",
 @"friend2" : @"sangyeol"};

 return 0;
}
```

NSArray와 마찬가지로 @{}하면 마지막에 nil을 붙일 필요가 없으면 중간에 객체가 nil이라면 오류를보냅니다.

이밖에도 우리가 자주사용하는 `NSString` 또한 @""을 통해서 객체로 감싸 문자열을 넣어 줄 수있습니다.