LLVM 컴파일러 4.0에서 부터 새롭게 추가된 리터럴은 크게 세 가지로, NSNumber Literals, Colloection Literals, Object Subscripting 이다.

NSNumber Literals

Foundation 프레임웍의 NSNumber 클래스는 스칼라값을 감싸는 객체이다. 객체에 들어갈 수 있는 값은 C에서는 '수'로 취급할 수 있던 char, short, int, long, long long 등의 정수들과 Float, double과 같은 실수들 그리고 BOOL, bool 값등이다. 이렇게 숫자값을 객체에 감싼 것을 Boxed values라고도 한다더라.
정리하자면 숫자값 앞에 @를 붙이기만하면 NSNumber 객체로 둘러싼 객체를 바로 얻을 수 있다는 것이다.
NSNumber 리터럴은 단순히 숫자값으로 객체를 만드는 것을 넘어서 C수식을 계산하여 그 결과를 사용할 수 있다. 이때는 @({수식})의 형태로 수식을 괄호로 둘러싸주고 그 앞에 @을 붙이면 된다.

또한 enum 타입에 대해서도 만들 수 있다.
typedef enum { Red, Green, Blue } Color; NSNumber *favoriteColor = @(Green);

typedef enum { Red, Green, Blue } Color; NSNumber *favoriteColor = @(Green);

NSNumber 뿐만 아니라 NSStirng도 동일한 방식으로 C계산식의 결과를 리터럴을 통해 NSString 객체로 만들 수 있다 .

```
NSString *path = @(getenv("PATH"));
// = [NSString stringWithUTF8String:getenv("PATH")];
```

컨테이너 리터럴

Objective-C는 고정 배열이나 사전을 만들기위한 표현식 문법을 추가로 제공한다.

```
NSArray *array = [ @"Hello", NSApp, [NSNumber numberWithInt:42] ];
```

배열은 @[ ]를 통해서 만든다. 각 원소는 ","를 통해서 구분되며, 마지막에 nil을 넣어줄 필요는 없다. 사전은 아래와 같이 만들 수 있다.

```
NSDictionary *dictionary = @{
   @"name" : NSUserName(),
      @"date" : [NSDate date],
      @"processInfo" : [NSProcessInfo processInfo]
};
```

사전은 기존 메시지를 통한 방식과 달리 **키 : 값의 구성**으로 이루어지며, 마지막에 nil은 붙지 않는다.

Examples

The following program illustrates the rules for NSNumber literals:

```
void main(int argc, const char *argv[]) {
  // character literals.
  NSNumber *theLetterZ = @'Z';          // equivalent to [NSNumber numberWithChar:'Z']

  // integral literals.
  NSNumber *fortyTwo = @42;             // equivalent to [NSNumber numberWithInt:42]
  NSNumber *fortyTwoUnsigned = @42U;    // equivalent to [NSNumber numberWithUnsignedInt:42U]
  NSNumber *fortyTwoLong = @42L;        // equivalent to [NSNumber numberWithLong:42L]
  NSNumber *fortyTwoLongLong = @42LL;   // equivalent to [NSNumber numberWithLongLong:42LL]

  // floating point literals.
  NSNumber *piFloat = @3.141592654F;    // equivalent to [NSNumber numberWithFloat:3.141592654F]
  NSNumber *piDouble = @3.1415926535;   // equivalent to [NSNumber numberWithDouble:3.1415926535]

  // BOOL literals.
  NSNumber *yesNumber = @YES;           // equivalent to [NSNumber numberWithBool:YES]
  NSNumber *noNumber = @NO;             // equivalent to [NSNumber numberWithBool:NO]

#ifdef __cplusplus
  NSNumber *trueNumber = @true;         // equivalent to [NSNumber numberWithBool:(BOOL)true]
  NSNumber *falseNumber = @false;       // equivalent to [NSNumber numberWithBool:(BOOL)false]
#endif
}
```