Objective-C(superset of C)

Added
1. Definition of new classes
2. Class and instance methods
3. Method invocation
4. Declaration of properties
5. Static and dynamic typing
6. Blocks—encapsulated segments of code that can be executed at any time
7. Extensions to the base language such as protocols and categories.

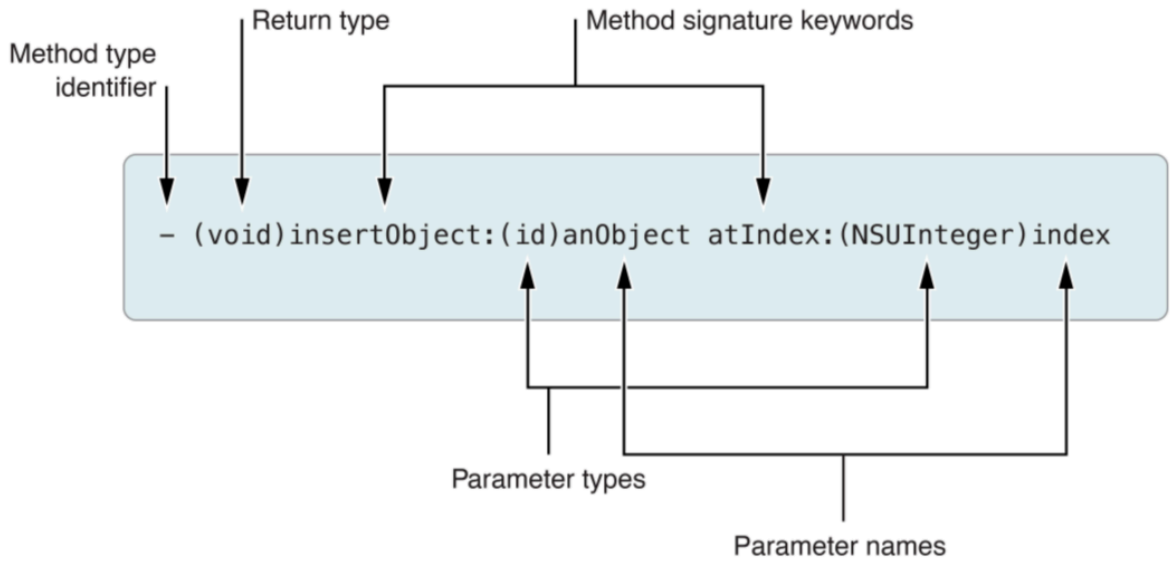
Object
an object is a runtime instance of a class.
1. contains its own in-memory copy of the instance variables declared by its class;
2. contains pointers to the methods of the class.

Dynamic typing & Static typing
Statically typed variables include the class name in the variable type declaration.
Dynamically typed variables use the type id for the object instead.

```
MyClass *myObject1;  // Static typing
id       myObject2;  // Dynamic typing
NSString *userName;  // From Your First iOS App (static typing)
```

Asterisk (*)
In Objective-C, object reference must always be a pointer.
statically typed variables must have asterisk(*).
id type implies a pointer.

Methods
1. instance method
this execution is scoped to a particular instance of the class,
before using it, you need to create an instance of the class.
2. class method
this execution is scoped to the method's class.



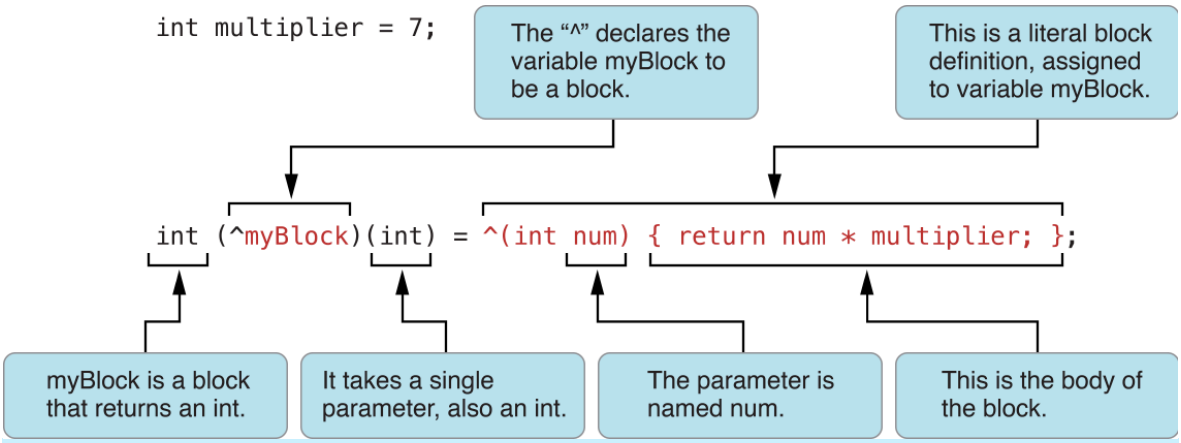
Accessor methods
it GET and SET the state of an object.
The class of an object defines an interface that enables users of its objects to GET and SET the values of encapsulated properties.
You can't use a reference to a dynamically typed object (object of type id) in a dot-notation expression.

Property
a property is some data encapsulated or stored by an object,
like attribute (name or color), or a relationship to one or more objects.

You include property declarations with the method declarations in your class interface.
You declare public properties in the class header files;
You declare private properties in a class extension in the source file(.m).
Properties of controller objects such as delegates and view controllers should typically be private.

```
@property (copy) MyModelObject *theObject;  // Copy the object during assignment.
@property (readonly) NSView *rootView;      // Declare only a getter method.
@property (weak) id delegate;               // Declare delegate as a weak reference
```

Blocks
objects that encapsulate a unit of work (a segment of code).




```
int result = myBlock(4); // result is 28
```

Protocol

it declares methods that can be implemented by any class, even if those classes implementing the protocol don't have a common super class.

```
@interface HelloWorldViewController : UIViewController
<UITextFieldDelegate> {
```

Category

```
@interface NSDate (NSDateCreation)
```

a feature that enables you to extend the interface of a class without having to subclass it.

the methods in the category become part of the class type and are inherited by all the class's subclass.

Class Extension

a category except there is no text between the parentheses.

to declare private properties and methods in implement file(.m).

Define Types

Type	Description and literal
`id`	The dynamic object type. The negative literal for both dynamically and statically typed objects is `nil`.

`Class`	The dynamic class type. Its negative literal is `Nil`.
`SEL`	The data type (`typedef`) of a selector; this data type represents a method signature at runtime. Its negative literal is `NULL`.
`BOOL`	A Boolean type. The literal values are `YES` and `NO`.

Self & Super
a local variable that you can use within a message implementation to refer to the current object.
If you send a message to self, the runtime first looks for the method implementation in the current object's class;
if it can't find the method there, it looks for it in its superclass (and so on).
If you send a message to super, the runtime first looks for the method implementation in the superclass.