OBJECTIVE-C TYPEDEF

The Objective-C programming language provides a keyword called **typedef**, which you can use to give a type a new name. Following is an example to define a term **BYTE** for one-byte numbers:

```
typedef unsigned char BYTE;
```

After this type definition, the identifier BYTE can be used as an abbreviation for the type **unsigned char, for example:**.

```
BYTE  b1, b2;
```

By convention, uppercase letters are used for these definitions to remind the user that the type name is really a symbolic abbreviation, but you can use lowercase, as follows:

```
typedef unsigned char byte;
```

You can use **typedef** to give a name to user-defined data type as well. For example, you can use typedef with structure to define a new data type and then use that data type to define structure variables directly as follows:

```
#import <Foundation/Foundation.h>

typedef struct Books
{
   NSString *title;
   NSString *author;
   NSString *subject;
   int book_id;

} Book;

int main( )
{

   Book book;
   book.title = @"Objective-C Programming";
   book.author = @"TutorialsPoint";
   book.subject = @"Programming tutorial";
   book.book_id = 100;
   NSLog( @"Book title : %@\n", book.title);
   NSLog( @"Book author : %@\n", book.author);
   NSLog( @"Book subject : %@\n", book.subject);
   NSLog( @"Book Id : %d\n", book.book_id);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
2013-09-12 12:21:53.745 demo[31183] Book title : Objective-C Programming
2013-09-12 12:21:53.745 demo[31183] Book author : TutorialsPoint
2013-09-12 12:21:53.745 demo[31183] Book subject : Programming tutorial
2013-09-12 12:21:53.745 demo[31183] Book Id : 100
```

typedef vs #define

The **#define** is a Objective-C directive, which is also used to define the aliases for various data types similar to **typedef** but with following differences:

- The **typedef** is limited to giving symbolic names to types only whereas **#define** can be used

to define alias for values as well, like you can define 1 as ONE, etc.

- The **typedef** interpretation is performed by the compiler where as **#define** statements are processed by the pre-processor.

Following is a simplest usage of #define:

```
#import <Foundation/Foundation.h>

#define TRUE  1
#define FALSE 0

int main( )
{
   NSLog( @"Value of TRUE : %d\n", TRUE);
   NSLog( @"Value of FALSE : %d\n", FALSE);

   return 0;
}
```

When the above code is compiled and executed, it produces the following result:

```
2013-09-12 12:23:37.993 demo[5160] Value of TRUE : 1
2013-09-12 12:23:37.994 demo[5160] Value of FALSE : 0
```