Objective C

Elisabeth Childs
Naomi Staley

Intro

- Superset of C
- Influenced by C and Smalltalk
- Compiled
- Rise in popularity, developed by Apple

Naming, Scope and Syntax

- Basic naming, scope are the same as they are in C
- Additions to C Syntax
  - The way functions are called
  - Additional keywords are specified with the @

```
int main(int argc, const char * argv[])
{
    @autoreleasepool {
```

Types

- You can use the basic C types
- Objective C has its own built in types
  - Start with NS
    - NSInteger, NSString, NSArray
  - Type id
- Objects are there own types

```
//create and release automatically
NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
NSString* string = [[[NSString alloc] initWithString: @"Hello"] autorelease];
NSArray pets = [[[NSString alloc] arrayWithObjects: @"dog",@"cat",@"fish"] autorelease];
[pool drain];
}
```

Methods

- Method Declarations
  - **Java**
    - `public void ll_push(Object data)`
  - **Objective C**
    - `-(void) ll_push: (id)data`
    - `-(int) multiply: (int)a bVar:(int)b cVar:(int)c`
    - `-(int) multiply:(int)a :(int)b :(int)c`
- Method Calls
  - **C**
    - `ll_push(LList, n)`
  - **Java**
    - `Llist.ll_push(n)`
  - **Objecive C** – called messages
    - `[LList ll_push: n]`
    - `int m = [l multiply:3 bVar:2 cVar:4] – names`
    - `int m = [l multiply:3 :2 :4] – no names`

Objects

.h file

```
@interface Node : NSObject
{
    id data;
    Node* next;
}
@property id data;
@property Node* next;

- (id)initNodeData: (id)value;

@end
```

- @interface
- @property
- @end

.m file

```
@implementation Node
@synthesize data;
@synthesize next;
- (id)initNodeData: (id)value
{
    self = [super init];
    if (self) {
        [self.data setData:value];
        [self.next setNext:NULL];
    }
    return self;
}

- (id)init
{
    return [self initNodeData: NULL];
}

@end
```

- @implementation
- @synthesize

Creating an Object

```
NSNumber *n = [NSNumber numberWithInt:i];

LinkedList* l = [[LinkedList alloc] init];

[l ll_push: n];
```

- Create a pointer to an object
- Allocate memory for that object
- Initialize the object

Memory

- The user can control Memory management

```
//create and release yourself
NSString* string = [[NSString alloc] init];
string = @"Hello";
[string release];
```

```
//create and release automatically
NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
NSString* string1 = [@"Hello" autorelease];
[pool drain];
```

```
//releases automatically
@autoreleasepool{
    //...
}
```

Compiling and Running

- This is basically the same as in C. What you do is if you have a file called types.m you want to run, in the terminal you write:
- gcc types.m –o types –framework Foundation
- Then ./types to run it.