

# Objective-C



# Objective-C

Not as ugly as it looks!

- Dynamic, untyped language
  - Has a lot in common with Smalltalk and Java
- Simple superset of C
  - No new keywords
- Method dispatch done at runtime

# Objective-C

## What does a class look like?

- Each object is defined in terms of its interface and implementation
- Interfaces are in .h files, implementations are in .m files.

# Objective-C

## An example interface

```
@interface DotView : NSView {
    NSPoint center;
    NSColor *color;
    float radius;
}

- (id)initWithFrame:(NSRect)frame;
- (void)dealloc;

- (void)setX: (int)x Y: (int) y;
+ (void)initialize;
@end
```

# Objective-C

## An example interface

Class name

```
@interface DotView : NSView {
    NSPoint center;
    NSColor *color;
    float radius;
}

- (id)initWithFrame:(NSRect)frame;
- (void)dealloc;

- (void)setX: (int)x Y: (int) y;
+ (void)initialize;
@end
```

# Objective-C

## An example interface

Superclass

```
@interface DotView : NSView {  
    NSPoint center;  
    NSColor *color;  
    float radius;  
}  
  
- (id)initWithFrame:(NSRect)frame;  
- (void)dealloc;  
  
- (void)setX: (int)x Y: (int) y;  
+ (void)initialize;  
@end
```

# Objective-C

## An example interface

```
@interface DotView : NSView {  
    NSPoint center;  
    NSColor *color;    Instance  
    float radius;      Variables  
}  
  
- (id)initWithFrame:(NSRect)frame;  
- (void)dealloc;  
  
- (void)setX: (int)x Y: (int) y;  
+ (void)initialize;  
@end
```

# Objective-C

## An example interface

```
@interface DotView : NSView {  
    NSPoint center;  
    NSColor *color;  
    float radius;  
}
```

```
- (id)initWithFrame:(NSRect)frame;  
- (void)dealloc;  
  
- (void)setX: (int)x Y: (int) y;  
+ (void)initialize;  
@end
```

Instance  
Methods

# Objective-C

## An example interface

```
@interface DotView : NSView {
    NSPoint center;
    NSColor *color;
    float radius;
}

- (id)initWithFrame:(NSRect)frame;
- (void)dealloc;

- (void)setX: (int)x Y: (int) y;
+ (void)initialize;      Static
                        Method
@end
```

# Objective-C

## An example implementation

@implementation DotView

```
- (id)initWithFrame:(NSRect)frame {  
    // Implementation goes here  
}  
  
- (void)dealloc {  
    // More implementation  
}  
  
+ (void)initialize {  
    // Static method  
    // No instance variables!  
}
```

# Objective-C

## Sending messages to objects

- With no arguments

```
[color release];
```

- With one argument

```
[self initWithFrame:theRect]
```

- With two arguments

```
[view convertPoint:eventLocation fromView:  
nil];
```

# Objective-C

## Accessing member variables

- Accessing a variable from within the class

```
radius = 2.0;
```

- Accessing a member variable from another object

```
DotView *otherView;
```

```
float myRadius = otherView->radius;
```

# Objective-C

## The id type

- Default variable type for object is 'id'
- Conceptually close to 'void \*'
- All objects can refer to themselves by  
[self ...]
- All objects can refer to superclass by  
[super ...]

# Objective-C

## Memory management

- Objective-C is not garbage collected - you must manage object lifetimes
- Objects use a reference counting mechanism

Method	Description
-retain	increases the reference count of an object by 1
-release	decreases the reference count of an object by 1
-autorelease	decreases the reference count of an object by 1 at some stage in the future
-alloc	allocates memory for an object, and returns it with retain count of 1
-copy	makes a copy of an object, and returns it with retain count of 1

# Where To Go From Here

## Resources

- <http://developer.apple.com/cocoa/>
- <http://www.bignerdranch.com/products/cocoa1.shtml>
  - Aaron Hillegass, Cocoa Programming for Mac OS X, 1e

# Where To Go From Here

## Resources

- <http://www.stepwise.com>
  - Lots of great articles on Cocoa development
  - Memory management information
- XCode documentation window