Développement mobile sous iOS

R. Bergoin - H. Peytavin

Objectifs

- Comprendre les contraintes «mobiles»
- Apprendre l'Objective-C
- Savoir lire une documentation

Plan du cours

- Composition du Kit de développement
- iOS
- Objective-C

Plan du cours

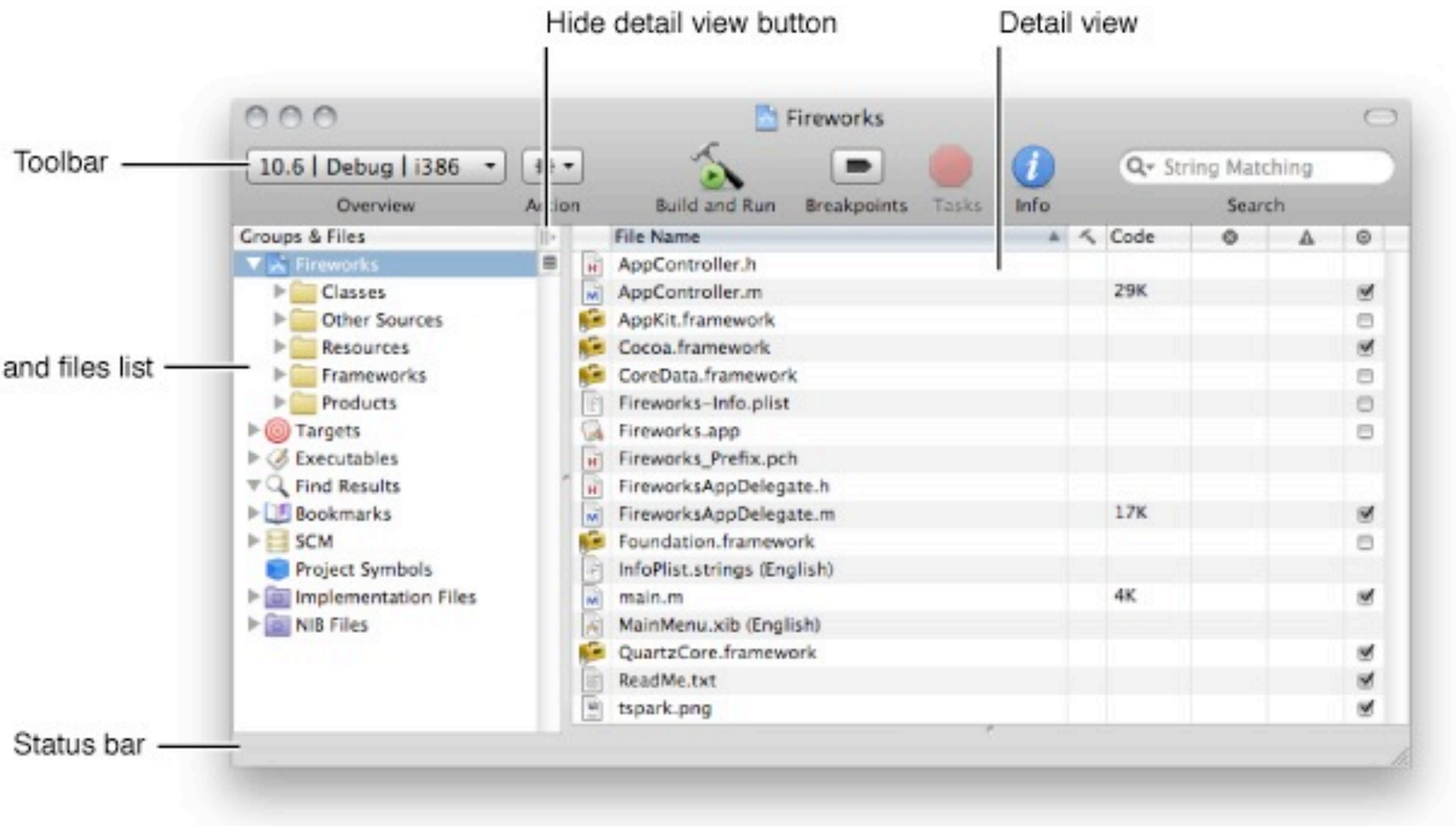
- Composition du Kit de développement
- iOS
- Objective-C

iOS SDK

- Un SDK complet :
  - Xcode (IDE)
  - Interface Builder
  - Instruments
  - iOS Simulator
  - iOS Developer Library

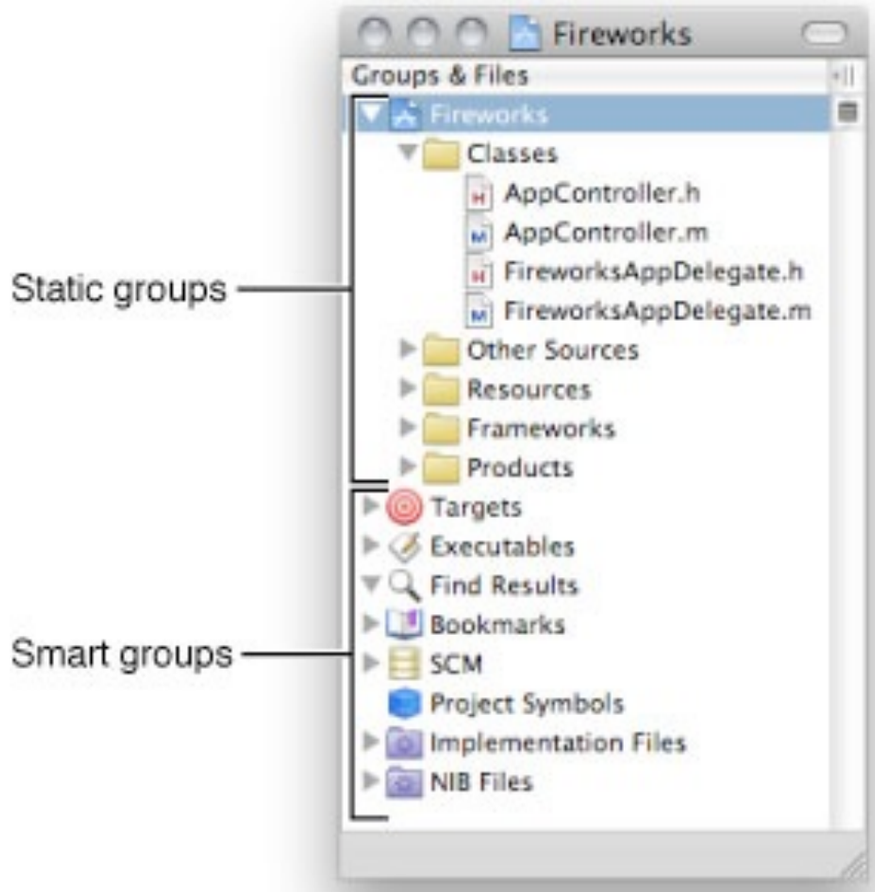
Xcode

Fenêtre principale :



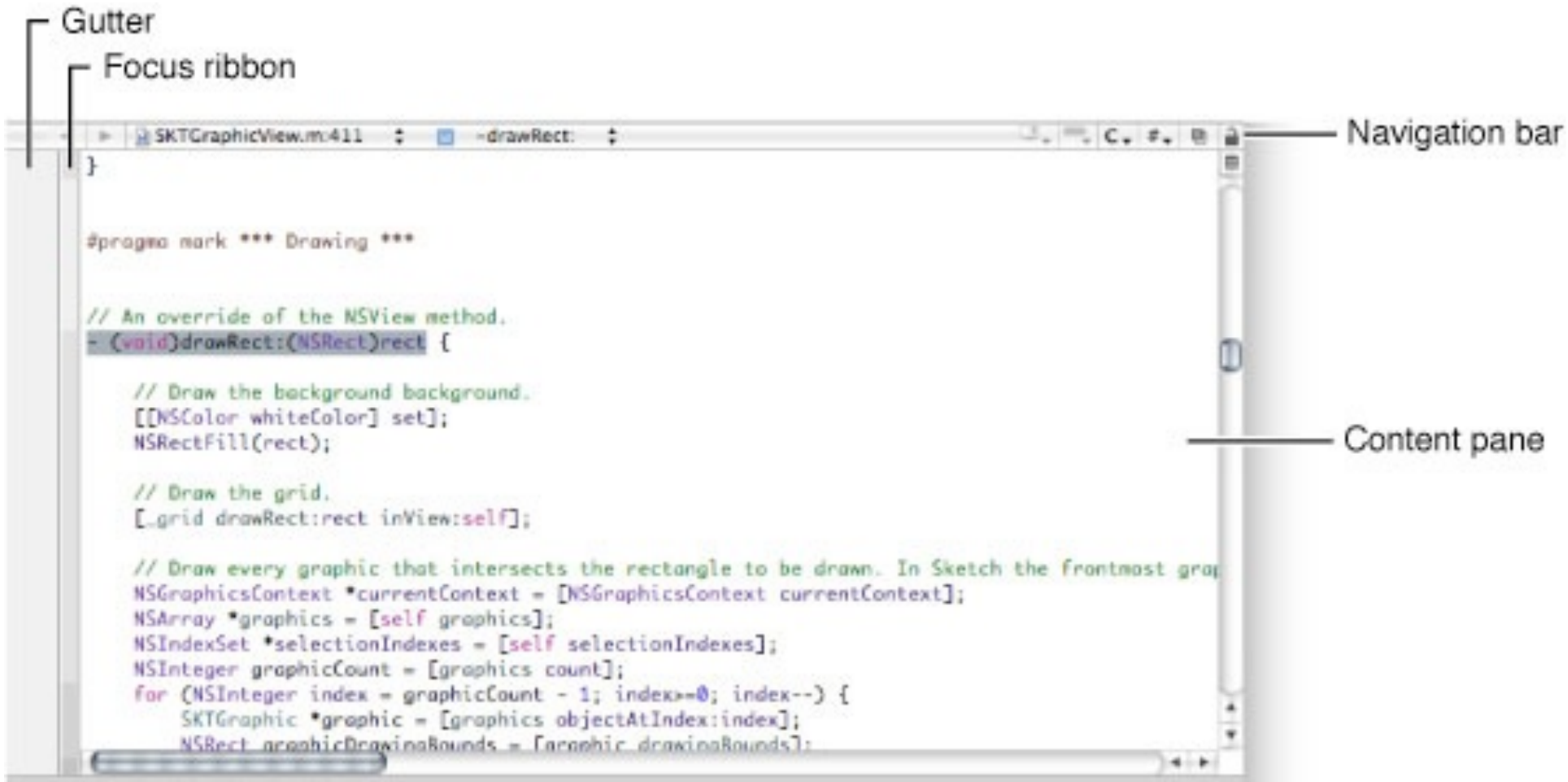
Xcode

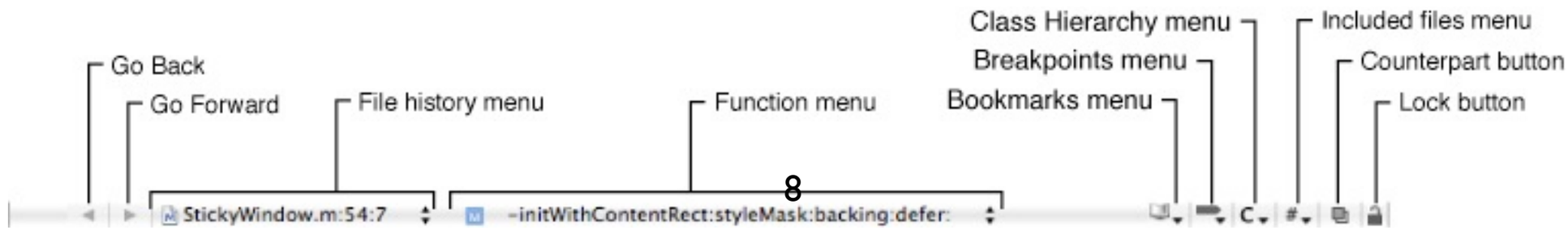
- Gestion de projet :
  - Fichiers sources
  - Frameworks
  - Cibles
  - Software Configuration Management



Xcode

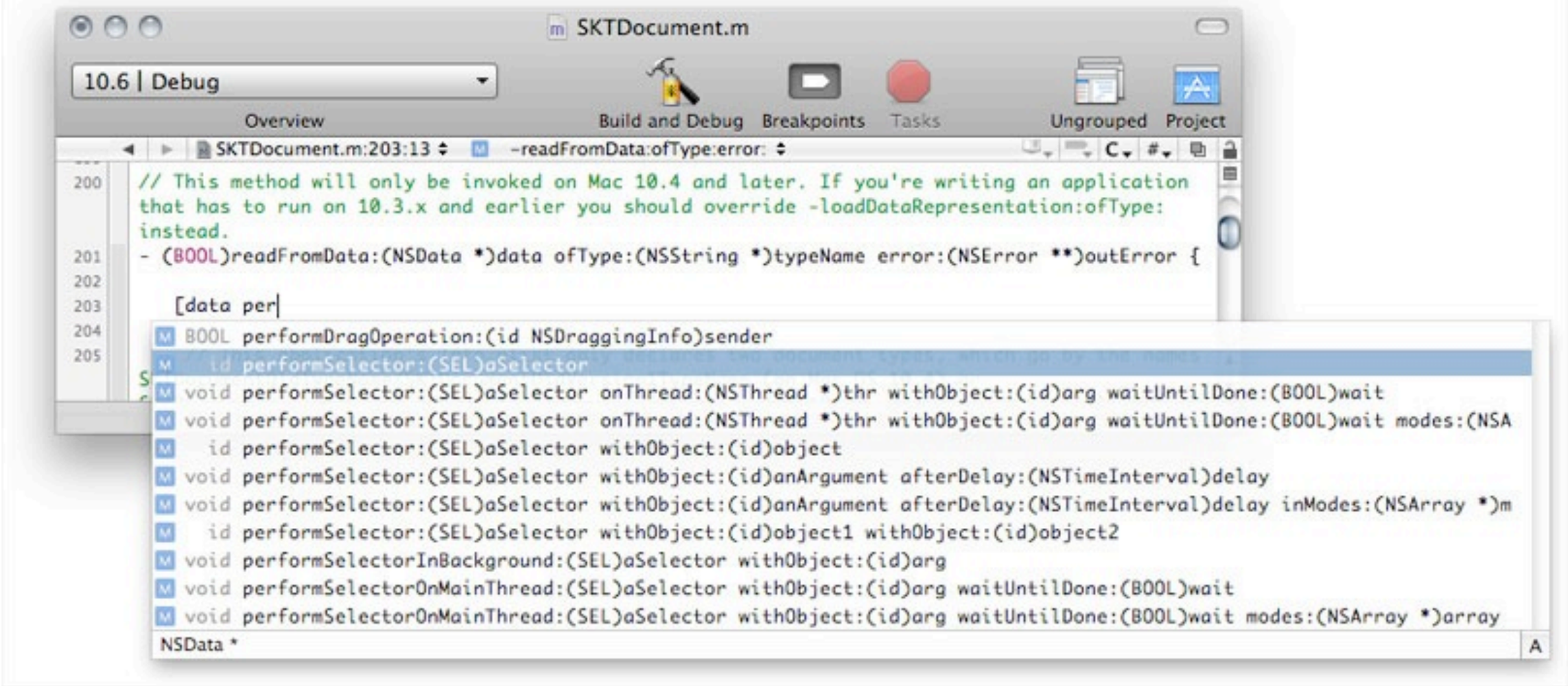
- Editeur de code :





Xcode

Complétion de code avec ESC



Xcode

- Fonction standard d'un IDE :
  - Edition d'un nom sur une portée donnée
  - Edition du nom d'un attribut (refactoring)
  - Indentation auto !
  - Completion de macros (if, while, dealloc, init...)

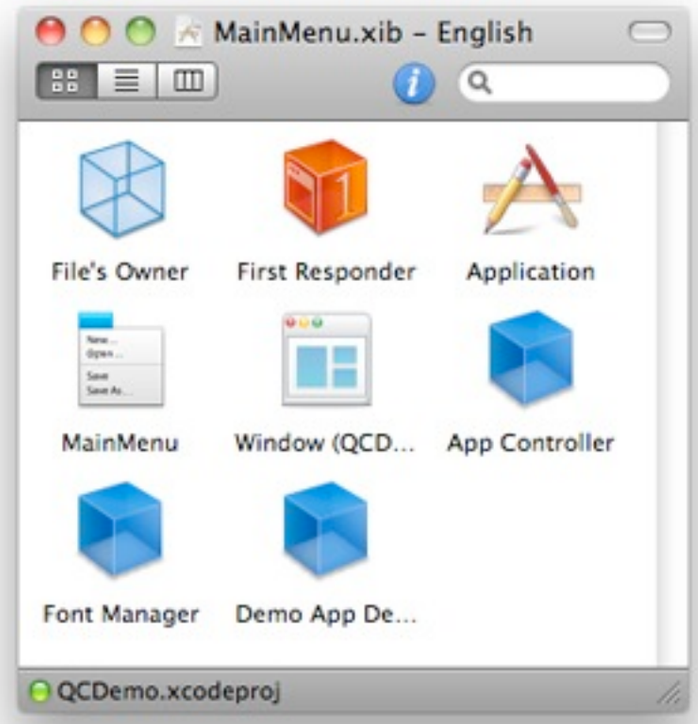
Xcode

- Et plus encore dans les préférences de l'application
- Et encore encore plus dans la documentation

http://developer.apple.com/iphone/

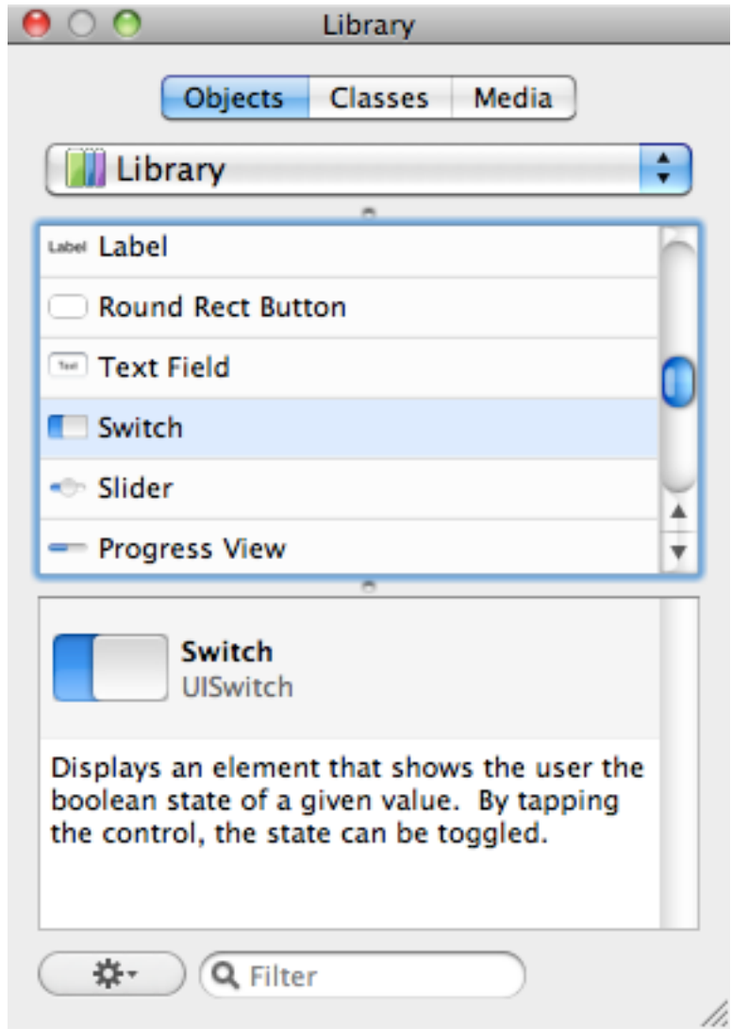
Interface Builder

- Construire une interface visuellement :
  - Dans un fichier XIB (xml)
  - File's Owner : en général l'objet à qui appartient la vue
  - Lien «code» - «interface» via des prises (outlets)



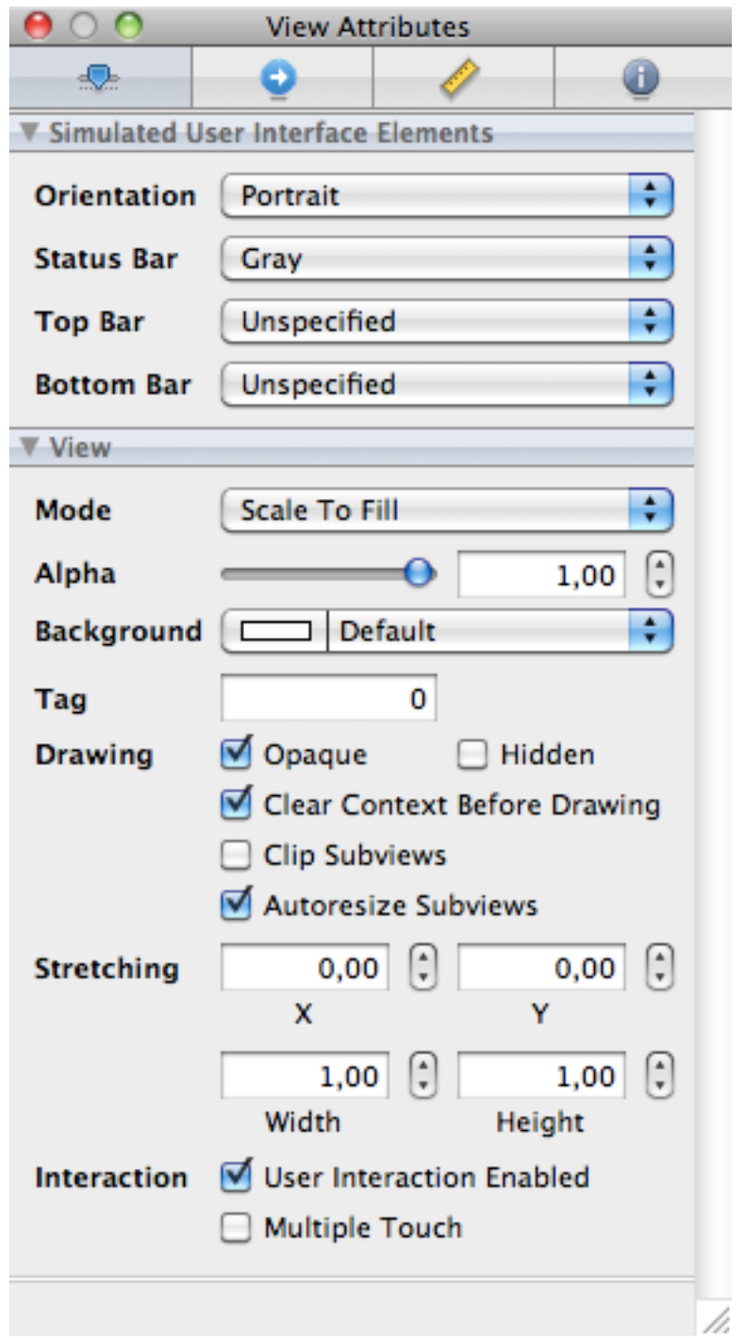
Interface Builder

- Ajouter des objets dans un XIB :
  - Composants graphiques
  - Contrôleurs (cf pattern MVC)
  - Simple et rapide : glisser déposer



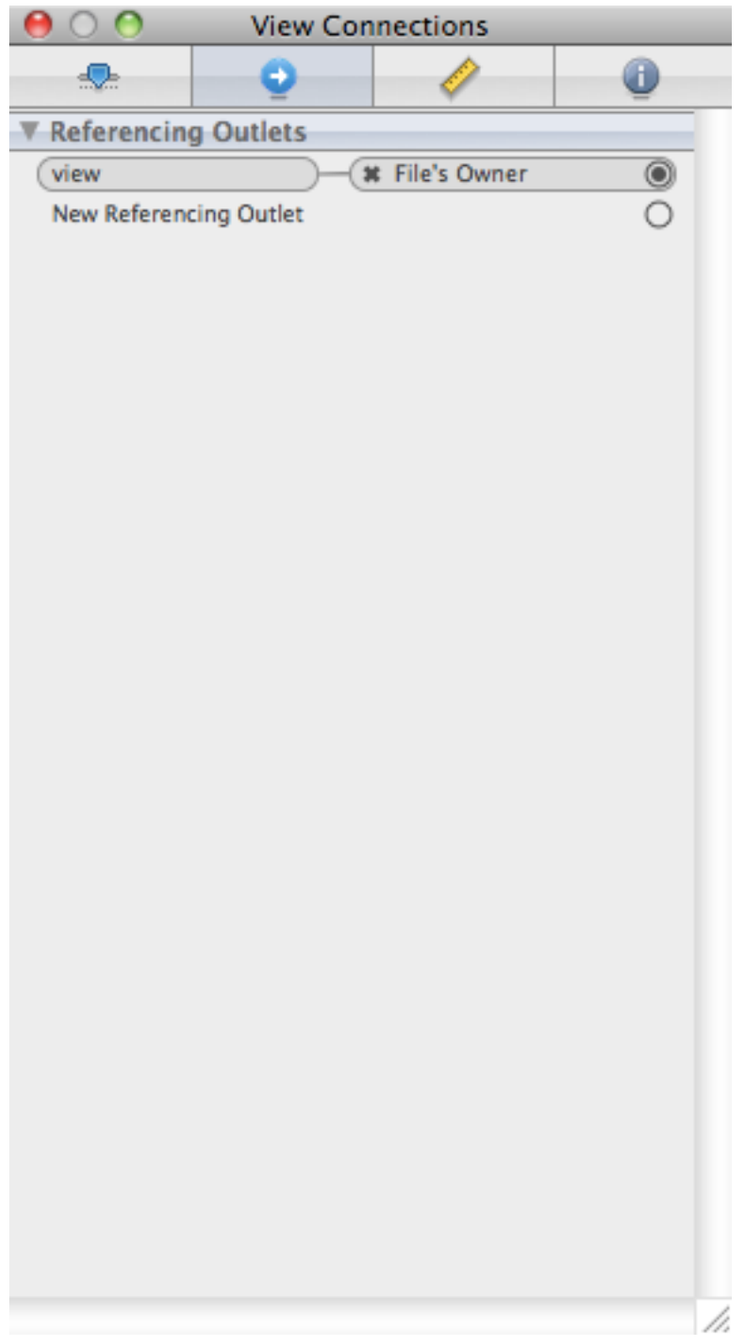
Interface Builder

- Edition des attributs d'un objet
  - Généraux



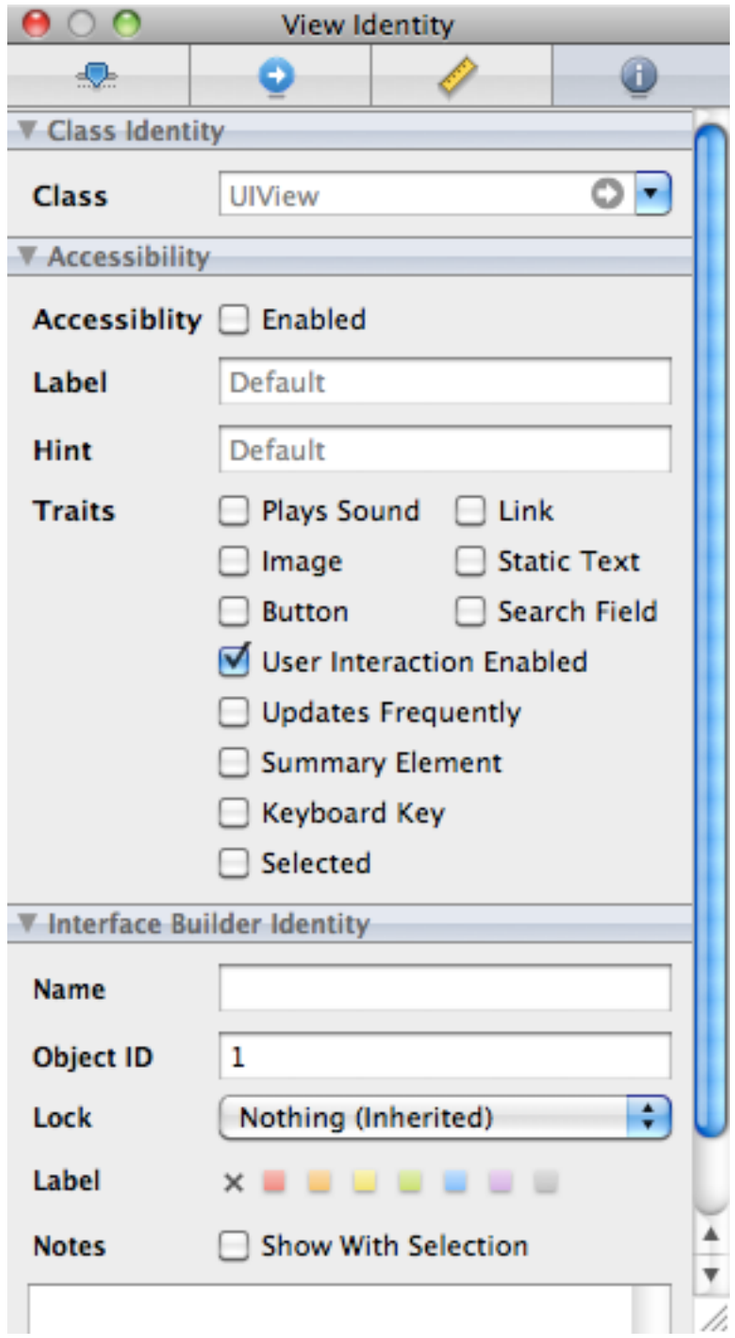
Interface Builder

- Edition des attributs d'un objet
  - Généraux
  - Outlets



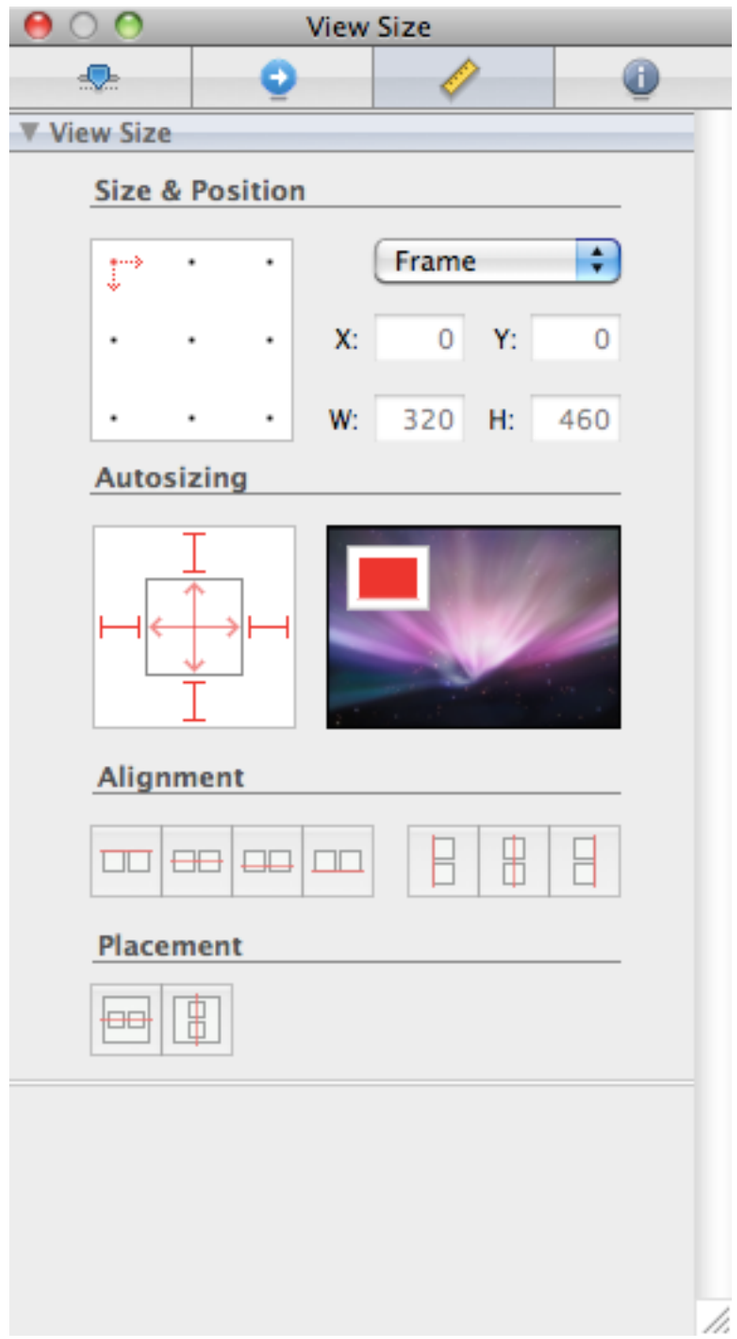
Interface Builder

- Edition des attributs d'un objet
  - Généraux
  - Outlets
  - Class et accessibilité



Interface Builder

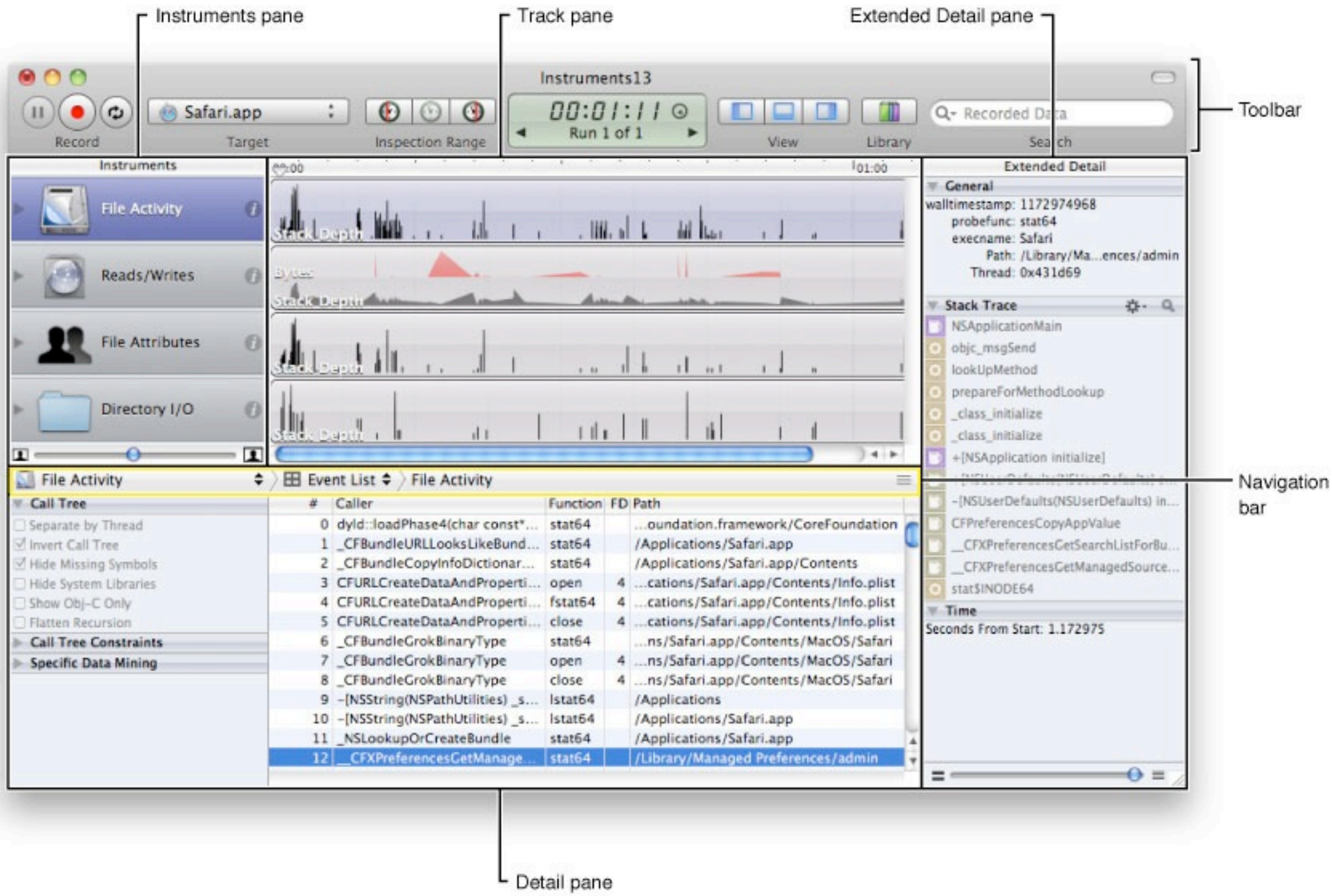
- Edition des attributs d'un objet
  - Généraux
  - Outlets
  - Class et accessibilité
  - Relatif à son positionnement, sa taille



Instruments

- Objectifs :
  - Améliorer les performances d'une App
  - Chercher les fuites de mémoires
  - Surveiller l'utilisation CPU
  - Surveiller l'activité sur les fichiers
  - ...

Instruments



Instruments

- Lancement ou attachement à une App
- Ralenti l'exécution
- Sauvegarde de la trace

iOS Simulator

- Simulation iPhone, iPod et iPad
- Capacités limitées (pas de multitouch, pas de capteur photo, pas d'accéléromètre)
- Amélioration à chaque version du SDKs

iOS Simulator

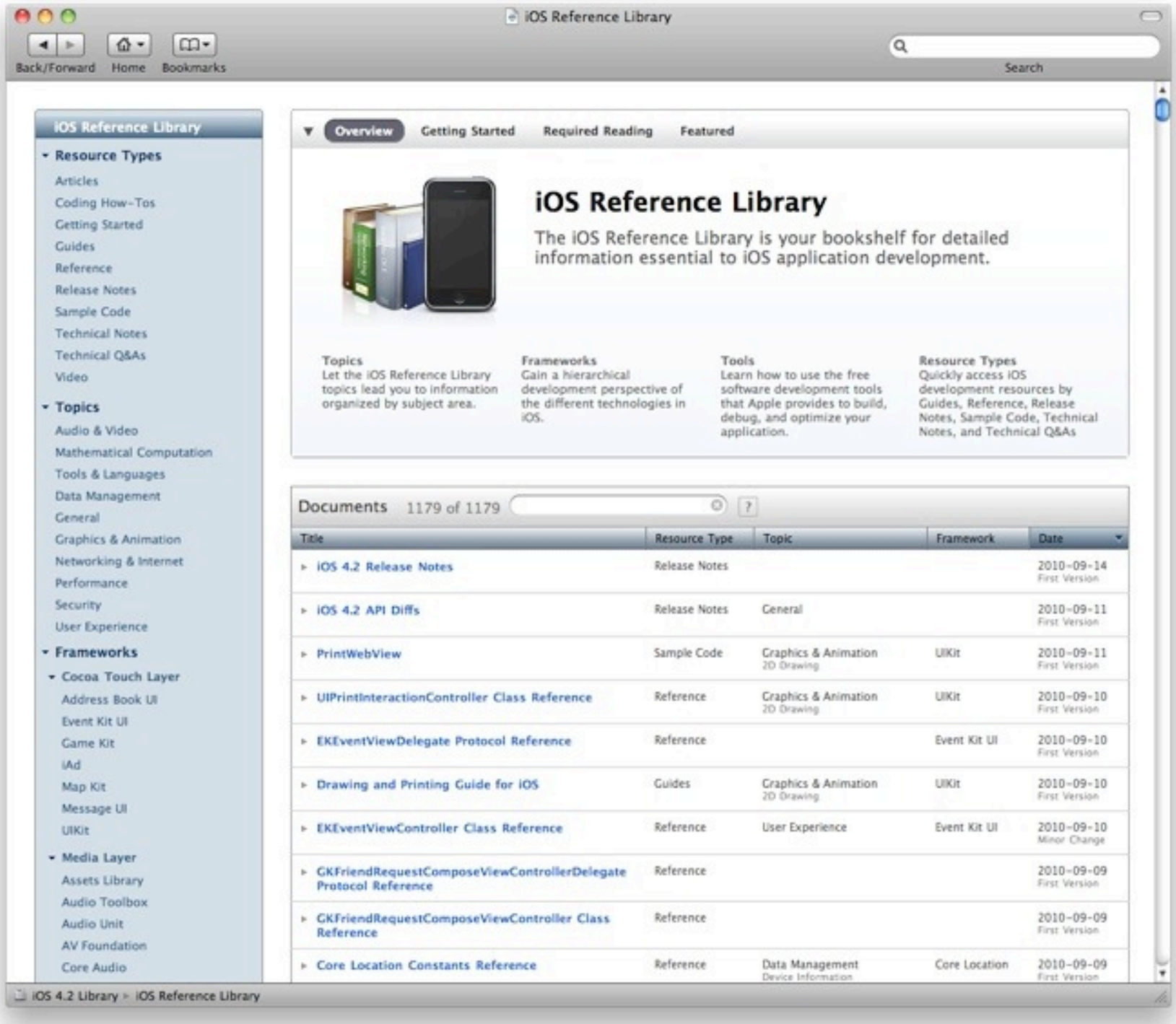
- Rotations
- Secousse
- Memory Warnings
- Simulation d’appel
- Sortie TV

iOS Simulator

- Multitouch : deux doigt avec alt
- GPS, positionnement au 1 Infinite Loop, Cupertino, CA 95014.
- Précision de 100m
- Latitude: 37.3317 North
- Longitude: 122.0307 West

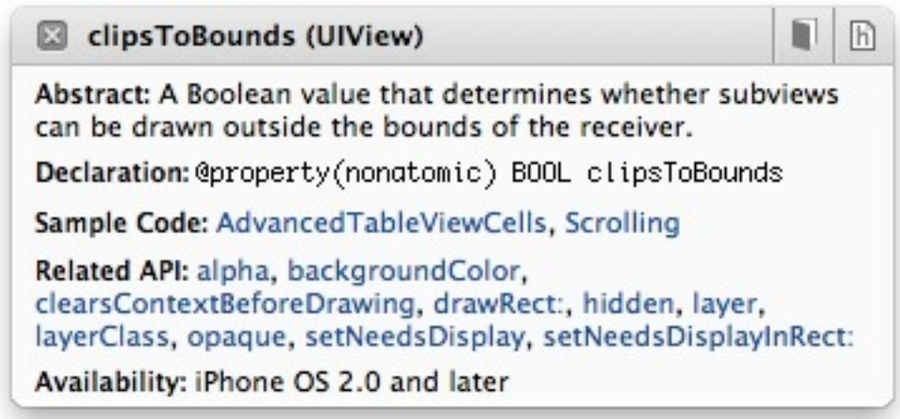
iOS Developer Library

- Références
- Exemples
- Complète !



iOS Developer Library

- Accès rapide depuis Xcode (alt + double click)
- Sinon recherche de la sélection via click droit, puis «Find text in documentation»



Plan du cours

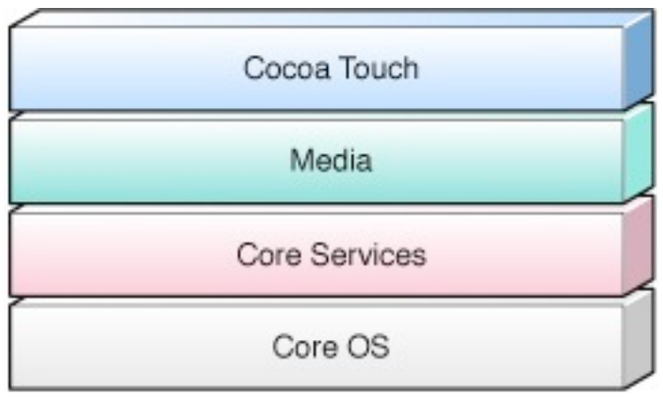
- Composition du Kit de développement
- iOS
- Objective-C

iOS

- Historique :
  - 1.0 en Juin 2007
  - 2.0 en Juillet 2008
  - 3.0 en Juin 2009
  - 3.2 (iPad) en Avril 2010
  - 4.0 en Juin 2010
  - 4.2 en Novembre 2010

iOS

- Basé sur un noyau XNU (comme OS X)
- Modèle en couche :



iOS : Cocoa Touch

- Gestion du multitâche
- Impression sans fil, sans pilote
- Notifications Push
- Gesture recognizers
- Partage de fichier via iTunes
- Peer-to-peer (via GameKit)
- API pour la composition d'email, de sms, prise de photo/vidéo, ajout d'événement au calendrier, d'un contact

iOS : Media

- Core Graphics
- Core Animation
- OpenGL ES
- Core Text
- Lecture Audio Vidéo

iOS : Core Services

- In-App Purchase (StoreKit)
- SQLite, Core Data
- XML
- Core Foundation (API en C) :
  - Collections, String, URLs, Preferences, Threads
  - CFNetwork (Sockets, SSL, HTTP, FTP, Bonjour)

iOS : Core Services

- Core Location
- Core Telephony
- Foundation (Brige Obj-C de CoreFoundation)

iOS : Core OS

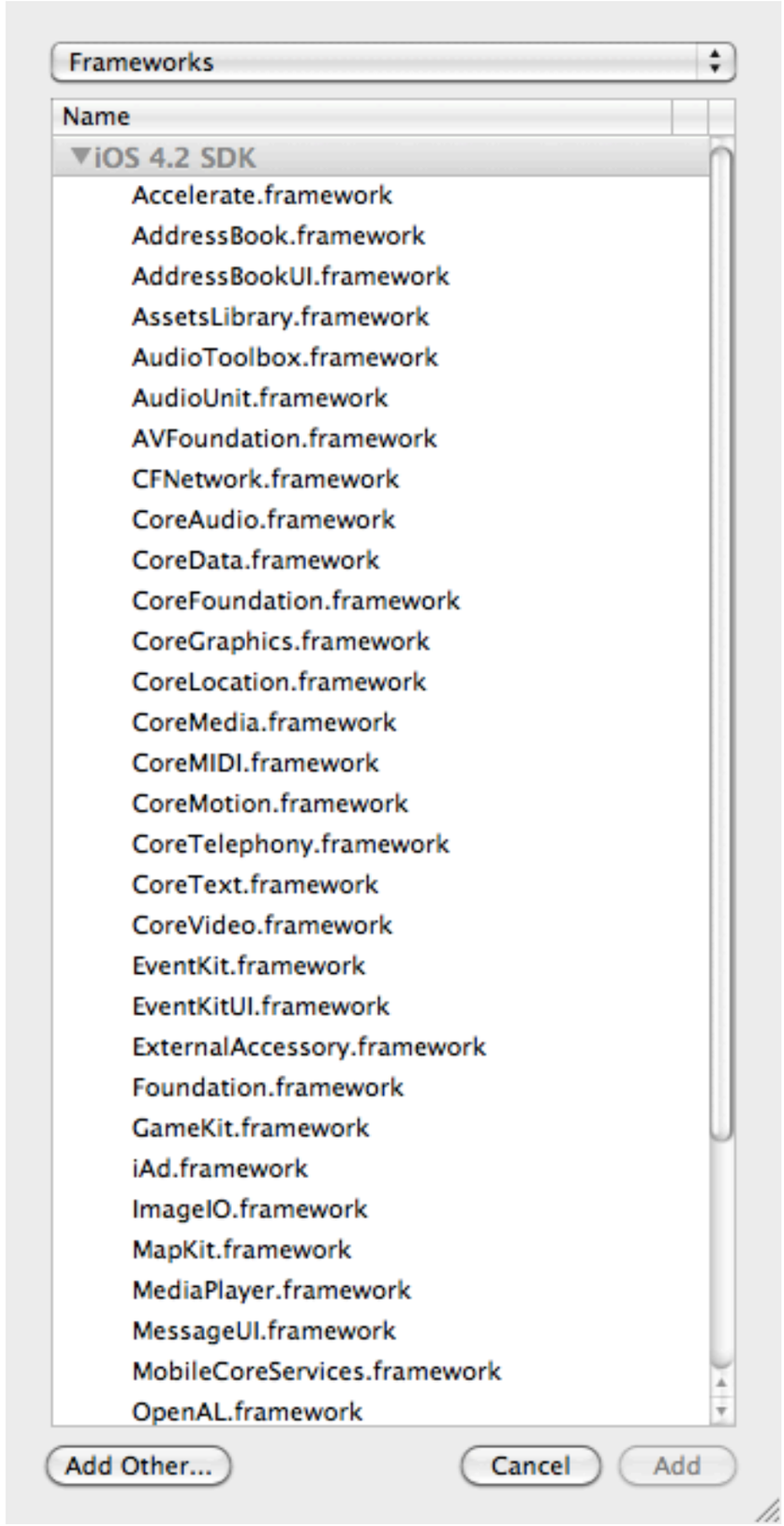
- Accelerate : Big Number & DSP
- External Accessory
- Security (Common Crypto, MD5, SHA, AES...)
- System :
  - Thread posix, Socket BSD, File I/O, Memory allocation, Math

iOS : Frameworks

- Framework ~ Bibliothèque partagée
- Principaux Framework utilisés :
  - UIKit
  - Foundation
  - CoreGraphics

iOS

- Quelques frameworks :
  - MapKit
  - CoreLocation
  - CoreMotion (gyroscope)
- Utilisation de Framework d'autres sources possible



Plan du cours

- Composition du Kit de développement
- iOS
- Objective-C

Objective-C

- Langage Objet
- Superset of C
  - .h / .m ou .mm
  - #import assure l'inclusion unique d'un header

Objective-C

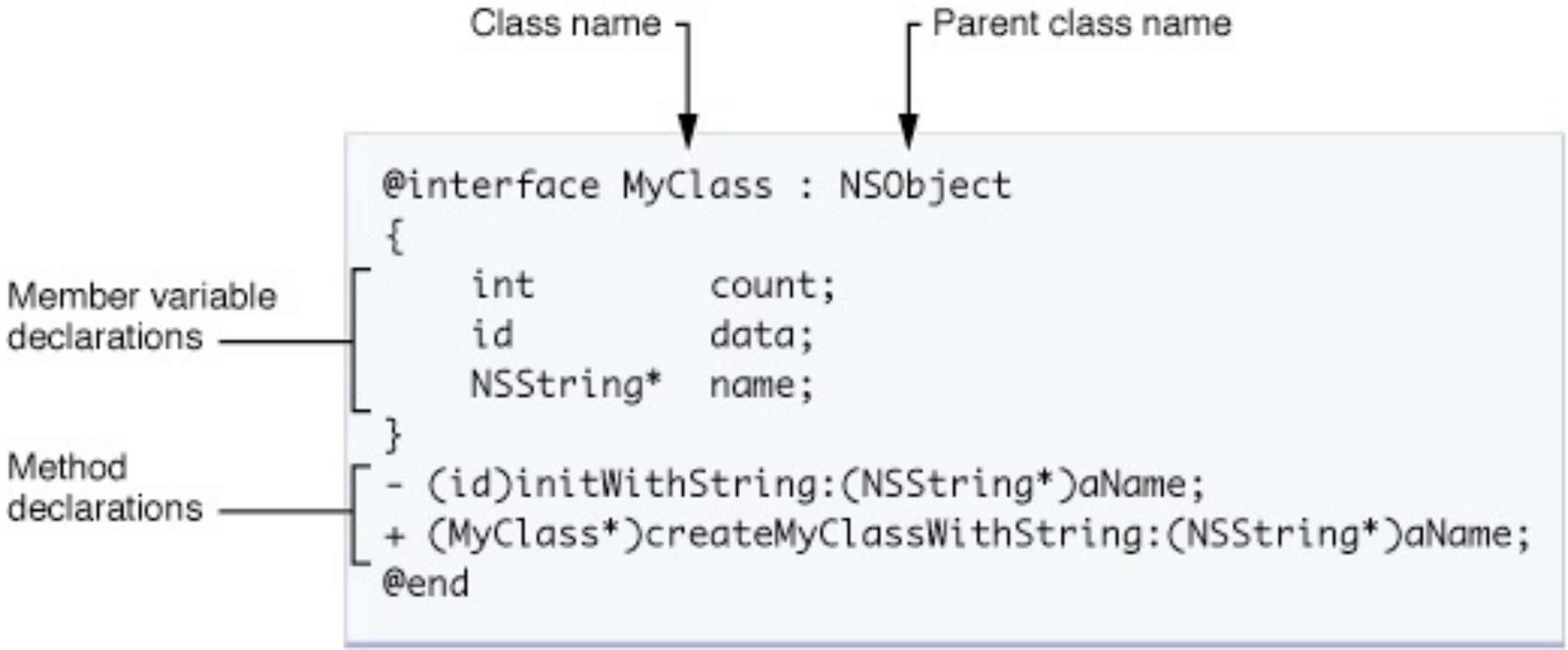
- Typage faible / typage fort :

```
MyClass *myObject1;
id      myObject2;
```

- id : pointeur vers un objet
- nil : équivalant à NULL

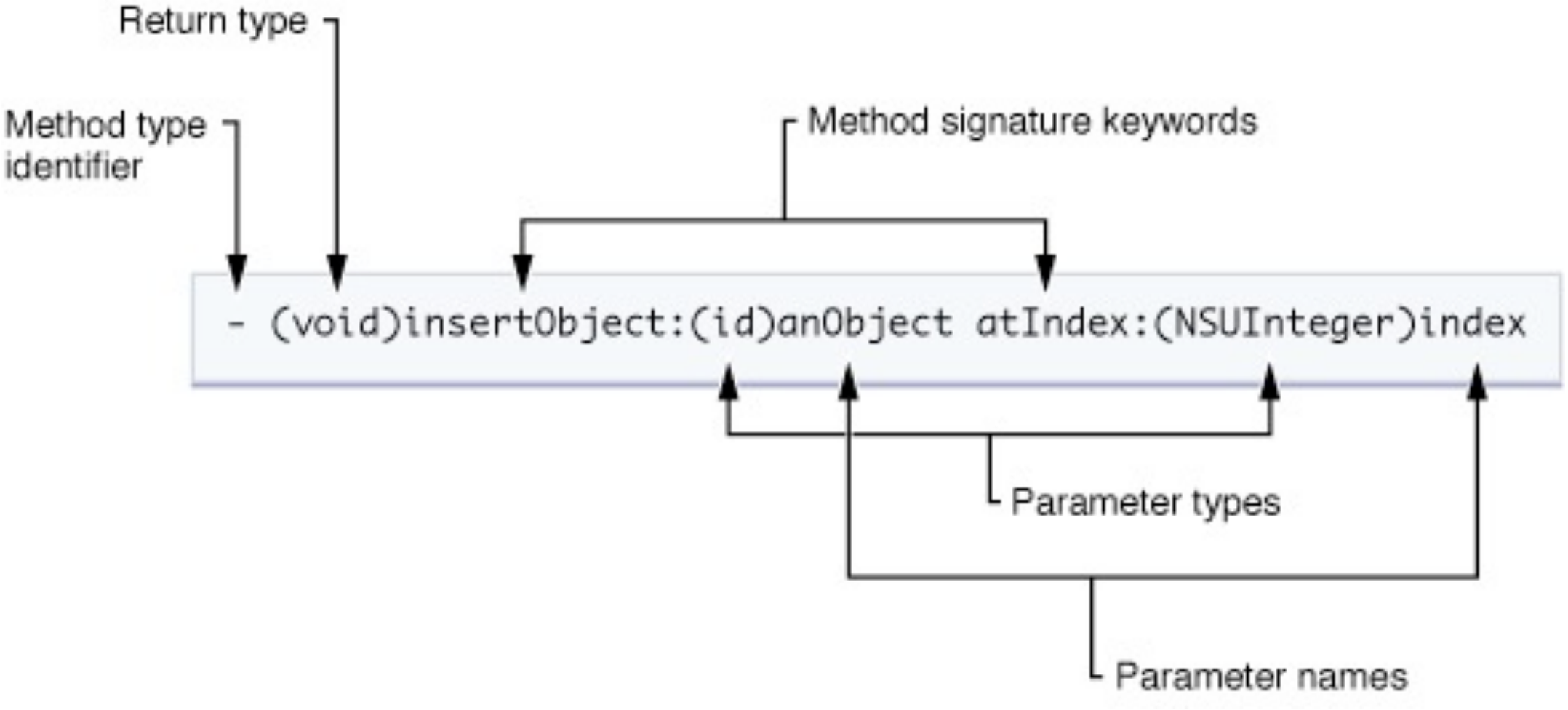
Objective-C

- Définition d'une classe :



Objective-C

- Appel de méthode, envoi de message :



Exemple : [myArray insertObject:obj atIndex:12];
id obj = [myArray objectAtIndex:12];

Objective-C

- Envoi de message à nil :

```
id anObject = nil;

if( [anObject methodReturningADouble] == 0.0 )
{
    // Code qui sera exécuté
}
```

- nil réponds à tous les messages par nil ou 0

```
id array = [[anObject getAnotherObject] getArray];
int count = [array count];
```

Objective-C

- Implémentation :

```
@interface MyClass : NSObject
{
    int        count;
    id         data;
    NSString*  name;
}
- (id)initWithString:(NSString*)aName;
+ (MyClass*)createMyClassWithString:(NSString*)aName;
@end
```

```
@implementation MyClass

- (id)initWithString:(NSString *)aName
{
    self = [super init];
    if (self) {
        name = [aName copy];
    }
    return self;
}

+ (MyClass *)createMyClassWithString: (NSString *)aName
{
    return [[[self alloc] initWithString:aName] autorelease];
}
@end
```

Objective-C

- Déclaration de propriétés :

```
@property BOOL flag;

@property (copy) NSString *nameObject;  // Copy the object during assignment.

@property (readonly) UIView *rootView;  // Declare only a getter method.
```

- Implémentation :

```
@synthesize flag;

@synthesize nameObject;

@synthesize rootView;
```

Objective-C

- Utilisation d'une propriété «dot syntax» :

```
myObject.flag = YES;
```

<=>

```
[myObject setFlag:YES];
```

Objective-C

- Retour sur @property et @synthesize :

```
@property BOOL flag;

---

@synthesize flag;
```

<=>

```
- (void)setFlag:(BOOL)aFlag;
- (BOOL)flag;

---

- (void)setFlag:(BOOL)aFlag
{
    flag = aFlag;
}

- (BOOL)flag
{
    return flag;
}
```

Objective-C

- Options sur les propriétés déclarées :
  - getter=getterName, setter=setterName
  - readwrite ou readonly
  - assign ou retain ou copy

Exemples :

```
@property (readonly, getter=trackCount) int count;
@property (retain) NSArray *tracks;
```

Objective-C

- Chaînes de caractères, classe NSString :

```
NSString *myString = @"My String\n";

NSString *anotherStr = [NSString stringWithFormat:@"%d %@",
                                                  1, @"String"];


// Create an Objective-C string from a C string

NSString *fromCString = [NSString stringWithCString:"A C string"
encoding:NSASCIIStringEncoding];
```

Objective-C

- Définition d'une interface (java) ou classe virtuelle pure (C++), dans un .h :

```
@protocol MyProtocol

- (void)myProtocolMethod;

@optional:

- (int)anOptionalyImplementedMethod;

@end
```

Objective-C

- Déclaration d'une classe implémentant un protocol :

```
@interface MyClass : NSObject <UIApplicationDelegate, MyProtocol> {

}

@end
```

- Déclaration d'une méthode ayant un objet qui doit implémenter un protocol en paramètre :

```
- (void)methodNeeding:(id<AProtocol>)anObjectImplementingAProtocol;
```

Objective-C

- Quelques protocoles souvent implémentés :
  - UIApplicationDelegate
  - UIAlertViewDelegate
  - UITableViewDataSource
  - ...

Objective-C

- Gestion mémoire
  - Mécanisme pour retenir un objet en mémoire : nombre de fois retenu (retainCount)
  - alloc, new, ou méthode contenant copy retourne un objet avec RC=1
  - release équivaut à RC--, autorelease à un RC-- appelé dans un futur proche

Objective-C

- Exemple :

```
Stock *myStock = [[Stock alloc] init];

// ...

NSArray *pieces = [myStock inventaire];

// ...

[stock release];
```

Objective-C

- Autorelease

```
- (NSArray *)pieces {
    NSArray *array = [[NSArray alloc]
initWithObjects:pieceRouges, pieceBleues, nil];

    return array;
}
```

Objective-C

- Autorelease

```
- (NSArray *)pieces {
    NSArray *array = [[NSArray alloc]
initWithObjects:pieceRouges, pieceBleues, nil];
    [array release];
    return array;
}
```

Objective-C

- Autorelease

```
- (NSArray *)pieces {
    NSArray *array = [[NSArray alloc]
initWithObjects:pieceRouges, pieceBleues, nil];

    return [array autorelease];
}
```

Objective-C

- Autorelease

```
- (NSArray *)pieces {
    NSArray *array = [NSArray arrayWithObjects:pieceRouges,
pieceBleues, nil];

    return array;
}
```

Objective-C

- Propriétés déclarées avec retain :

```
– (void)setObject:(NSObject *)anObject {

    [object autorelease];

    object = [anObject retain];

}
```

Objective-C

- Propriétés déclarées avec copy :

```
– (void)setObject:(NSObject *)anObject {

    [object autorelease];

    object = [anObject copy];

}
```

Objective-C

- Désallocation d'objet :

```
- (void)dealloc {

    [createdObjectInConstructor release];

    [retainedProperty release];

    [super dealloc];

}
```

Important: You should never invoke another object's `dealloc` method directly.

Objective-C

- Exercice :
  - Définir un objet correspondant à un véhicule (nombreDeRoue, taille, occupants)
  - Créer une instance et positionner la liste des occupants
  - Détruire l'instance

Objective-C

- Cycle de retenu :
  - A a une propriété B (retain)
  - B a une propriété A (retain)
- A ne pas faire !

Objective-C

- Utilisation de «références faibles»
  - A a une proprité B
  - B a un pointeur vers un A

Utilisé dans le pattern de délégation avec une référence faible sur un objet implémentant un protocol donné

Objective-C

- Exemple de référence faible :

```
@protocol UITableViewDelegate

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath;


@end

@interface UITableView : UIView {
    id<UITableViewDelegate> delegate;
}

@property (assign) id<UITableViewDelegate> delegate;

@end
```

Objective-C

- Mots clefs :
  - id : pointeur vers un objet
  - nil : objet magique

Objective-C

- Enumérations rapides :
  - NSFastEnumeration Protocol

```
NSArray *array = [NSArray arrayWithObjects:
        @"One", @"Two", @"Three", @"Four", nil];

for (NSString *element in array) {
    NSLog(@"element: %@", element);
}
```

Objective-C

```
NSDictionary *dictionary = [NSDictionary dictionaryWithObjectsAndKeys:
    @"quattuor", @"four", @"quinque", @"five", @"sex", @"six", nil];

for (NSString *key in dictionary) {

    NSLog(@"English: %@, Latin: %@",
          key,
          [dictionary objectForKey:key]);

}
```

Objective-C

- NSEnumerator :

```
NSArray *array = [NSArray arrayWithObjects:
        @"One", @"Two", @"Three", @"Four", nil];

NSEnumerator *enumerator = [array reverseObjectEnumerator];
for (NSString *element in enumerator) {
    if ([element isEqualToString:@"Three"]) {
        break;
    }
}
```

Objective-C

- Selecteurs :

```
@interface MyFrame : NSObject {}
- (void)setWidth:(int)w height:(int)h;
- (void)anotherMethod;
@end

SEL setWidthHeight = @selector(setWidth:height:);
SEL anotherSelector = @selector(anotherMethod);

// A partir d'une chaine :
setWidthHeight = NSSelectorFromString(aBuffer);

// Nom de la méthode
NSString *method;
method = NSStringFromSelector(setWidthHeight);
```

Objective-C

- Appel d'un selecteur :

```
[friend performSelector:@selector(gossipAbout:)
withObject:aNeighbor];
```

<=>

```
[friend gossipAbout:aNeighbor];
```

Objective-C

- Target-Action Design Pattern :

```
[myButtonCell setAction:@selector(reapTheWind:)];
[myButtonCell setTarget:anObject];
```

Avec UIKit :

```
UIButton *btn = [[UIButton alloc] init];

[btn addTarget:self action:@selector(myBtnCallback:)
forControlEvents:UIControlEventTouchUpInside];
```

Objective-C

- Vérifier si un objet répond à une méthode :

```
if ( [anObject respondsToSelector:@selector(setVal:)] )

    [anObject setVal:0.0];

else

    fprintf(stderr, "%s can't be placed\n",
        [NSStringFromClass([anObject class]) UTF8String]);
```

Objective-C

- Exceptions :

```
Cup *cup = [[Cup alloc] init];

@try {
    [cup fill];
}
@catch (NSException *exception) {
    NSLog(@"main: Caught %@: %@", [exception name], [exception
reason]);
}
@finally {
    [cup release];
}
```

Objective-C

- Exceptions :

```
NSException *exception = [NSException exceptionWithName:@"HotTeaException"
                                   reason:@"The tea is too hot"  userInfo:nil];

@throw exception;
```

Peu utilisée : resource-intensive.

Objective-C

- Thread & synchronisation

```
- (void)criticalMethod
{
    @synchronized(self) {
        // Critical code.
        ...
    }

}
```