# Introduction au développement iOS

Adrien Humilière
adrien.humiliere@djit.fr

# About


- Adrien Humilière
  adrien.humiliere@djit.fr


- Développeur iOS/Mac à **edjing**


www.edjing.com


REC
TEMPO
AUTO
LOOP
CUE
1:36 - 2:44
Don't Let Me Go John Steff
FX
SYNC
120.2
A
B
SYNC
120.2
Don't Let Me Go John Steff
1:36 - 2:44
CUE
LOOP
TEMPO
FX

# Organisation du cours

- Les bases d’Objective-C
- CocoaTouch (Views, ViewControllers & Storyboard)

- Ressources : adhumi.fr/teaching

# Organisation du cours

**Notation**

- 60% sur la partie Serveur (Olivier Pitton)
- **40% sur la partie iOS** (30% code, 25% fonctionnalités, 20% UI/UX, 25% présentation)
- Une seule présentation (début juin)

- TP ramassés, non-notés

**Objectifs du cours :**

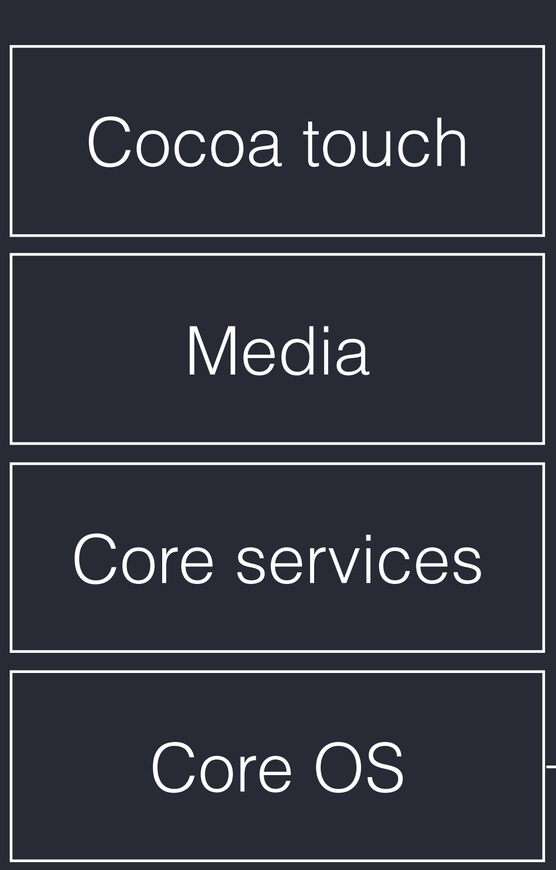
*Bases pour développer une application iOS en autonomie*

# Aujourd’hui

- Qu’est ce qu’iOS et Objective-C ?
- Démo
- Bases de l’Objective-C

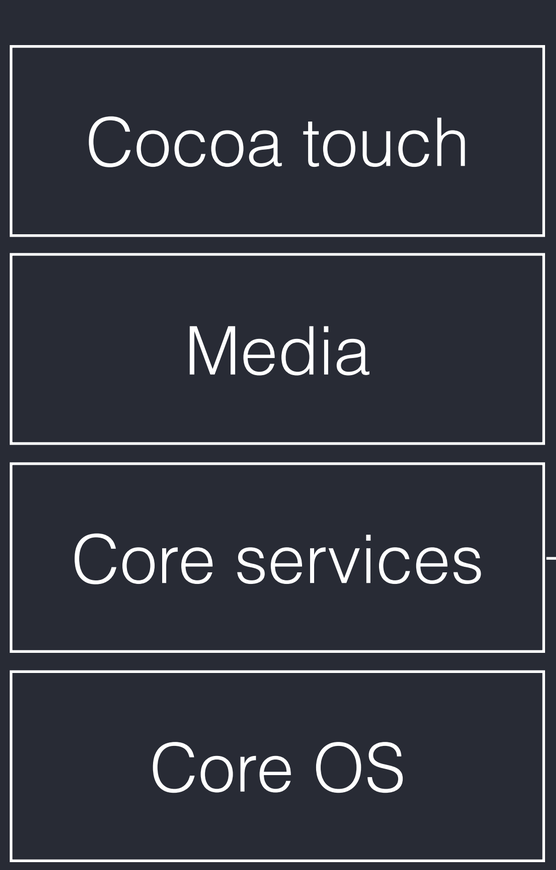
[ alt + maj + ( ]

# Qu’est ce qu’iOS ?



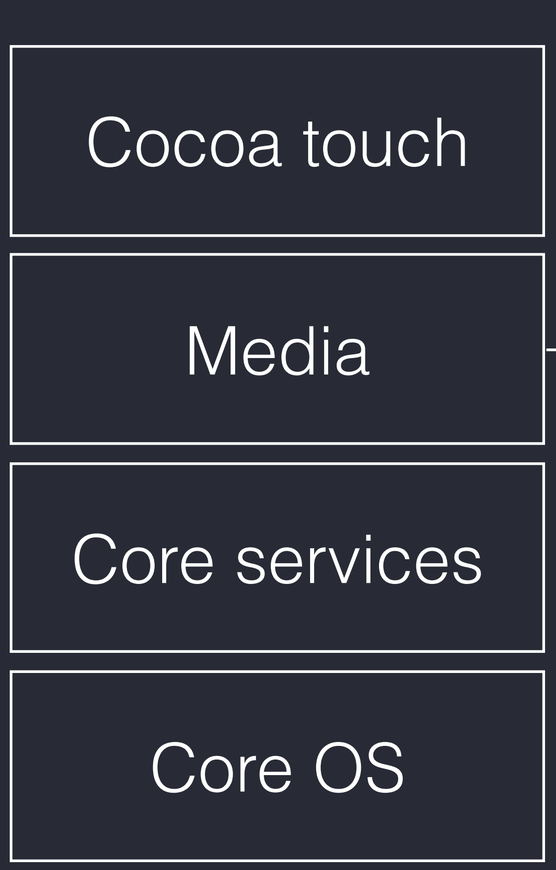

- OSX Kernel
- Mach 3.0
- BSD
- Sockets
- Security
- Power Management
- Keychain Access
- Certificates
- File System
- Bonjour

# Qu’est ce qu’iOS ?



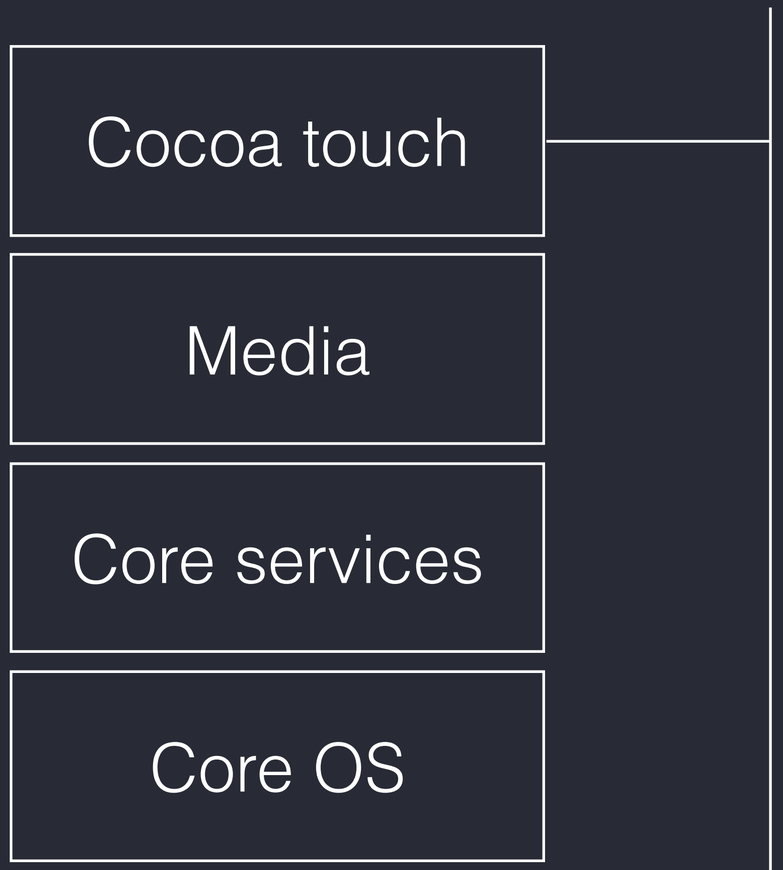

- Collections
- Address Book
- Networking
- File Access
- SQLite
- Core Location
- Net Services
- Threading
- Preferences
- URL Utilities

# Qu’est ce qu’iOS ?




- Core Audio
- JPEG, PNG, TIFF
- OpenAL
- PDF
- Audio Mixing
- Quartz (2D)
- Audio Recording
- Core Animation
- Video Playback
- OpenGL ES

# Qu’est ce qu’iOS ?




- Multi-Touch
- Core Motion
- View Hierarchy
- Localization
- Controls
- Alerts
- Web View
- Map Kit
- Image Picker
- Camera

# Qu’est ce qu’Objective-C ?

- Une extension dynamique, orientée-objet, au langage C.
- Créé en 1980 par Brad Cox et Tom Love.
- Adopté comme langage principal par l’OS NextSTEP, puis de Mac OS X et iOS.
- Presque toujours associé aux librairies Apple.

# Environnement

- **Xcode** et outils de développement


- **Langages :** Objective-C, Swift & co.
- **API :** Cocoa Touch

# **Démo**

Chronomètre

# Un programme en Objective-C

**Objective-C**
clang /main.m

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

int main(int argc, const char * argv[]) {
    NSLog(@"Hello, World!");
    return 0;
}
```

# Un programme en Objective-C

**Objective-C**
clang /main.m

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

int main(int argc, const char * argv[]) {
    NSLog(@"Hello, World!");
    return 0;
}
```

**Java**
java / HelloWorld.java

```
public class HelloWorld {
   public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

# Classe

**Objective-C**

Cat.h

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

@interface Cat : NSObject

@property NSString   *name;
@property float      age;
@property NSString   *color;

@end
```

Cat.m

```
#import "Cat.h"

@implementation Cat

- (instancetype)init {
    self = [super init];
    if (self) {
        // ...
    }
    return self;
}

- (void)sleep {
    // ...
}

@end
```

# Classe

**Java**

Cat.java

```
public class Cat {
    String name;
    float age;
    String color;

    public Cat() {
        // ...
    }

    void sleep() {
        // ...
    }
}
```

# Méthodes

- Déclaration d’une méthode :

```
- (void)insertObject:(id)object atIndex:(NSUInteger)idx;
```

```
+ (instancetype)sharedInstance;
```

- Implémentation d’une méthode :

```
- (void)insertObject:(id)object atIndex:(NSUInteger)idx {
    // Code de la méthode
}
```

- Appel d’une méthode :

```
[self.myString isEqualToString:@"chaine de caractères"];
```

# Variables

Il existe différents types de variables d’instances :

- Variable d’instance « standard »

```
@interface MyClass : UIView {
    UILabel *_myLabel;
}

@end
```

- Properties

```
@interface MyClass : UIView

@property UILabel *myLabel;

@end
```

# Variables

Il existe différents types de variables d’instances :

- Variable d’instance « standard »

```
@interface MyClass : UIView {
    UILabel *_myLabel;
}

@end
```

- Properties

```
@interface MyClass : UIView

@property (nonatomic, strong) UILabel *myLabel;

@end
```

# Instanciation

```
NSArray *myArray = [[NSArray alloc] init];
```

# Instanciation

```
NSArray *myArray = [[NSArray alloc] init];
```

- D’autres méthodes sont possibles

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray];
```

# Instanciation

```
NSArray *myArray = [[NSArray alloc] init];
```

- D’autres méthodes sont possibles

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray];
```

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray copyItems:YES];
```

# Instanciation

```
NSArray *myArray = [[NSArray alloc] init];
```

- D’autres méthodes sont possibles

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray];
```

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray copyItems:YES];
```

```
NSArray *myArray = [NSArray array];
```

# Instanciation

```
NSArray *myArray = [[NSArray alloc] init];
```

- D’autres méthodes sont possibles

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray];
```

```
NSArray *myArray = [[NSArray alloc]
    initWithArray:anotherArray copyItems:YES];
```

```
NSArray *myArray = [NSArray array];
```

```
NSArray *myArray = [NSArray arrayWithObject:anObject];
```

# Types de données

- Tous les types de données du langage C
  int, float, double, long, …
- Quelques types spécifiques
  id, BOOL, NSInteger, CGFloat, …
- Les classes de la bibliothèque Foundation
  NSArray, NSDictionary, NSString, NSDate, …

# Gestion mémoire

- Historiquement, Objective-C fonctionne avec un compteur de références.
- Depuis 2011, ARC (« Automatic Reference Counting ») transfère cette responsabilité du développeur au compilateur.

# Au prochain cours

- Cocoa Touch
- Le design pattern MVC
- Du networking
- Des concepts plus avancés en Objective-C