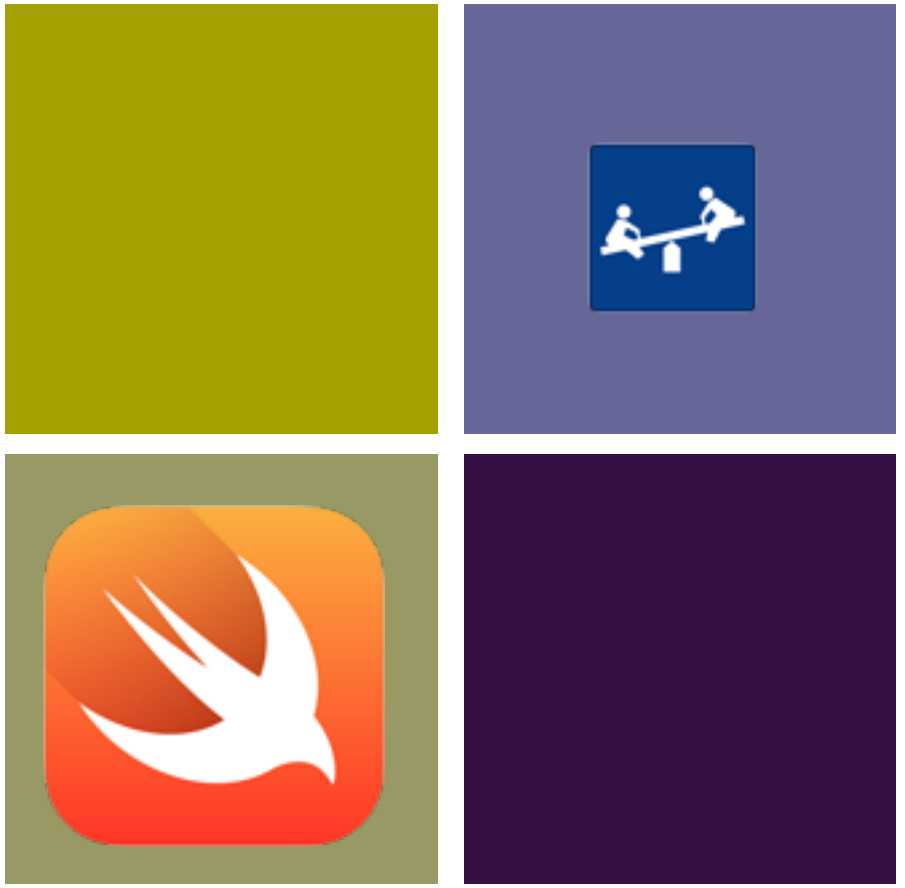
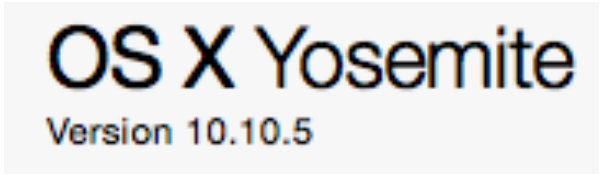


Développement iOS

Introduction iOS

Environnement du cours

- système :



- IDE :

- Xcode 7.0.0 autorise l'utilisation d'un device comme cible pour faire le test des applications, sans programme de développement standard. Un câble suffit donc pour faire le test.

- Virtualisation : Microsoft, vmWare, virtualBox, …

Environnement du cours

- Machine Apple physique ou virtuelle (non-licite)
- Pas nécessaire d'investir dans un programme de développement
- Xcode fait que vous êtes tout de même référencé chez Apple (member developer program)
- Comptes utilisateurs du cours : prévoir un compte App Store pour mettre à jour Xcode.

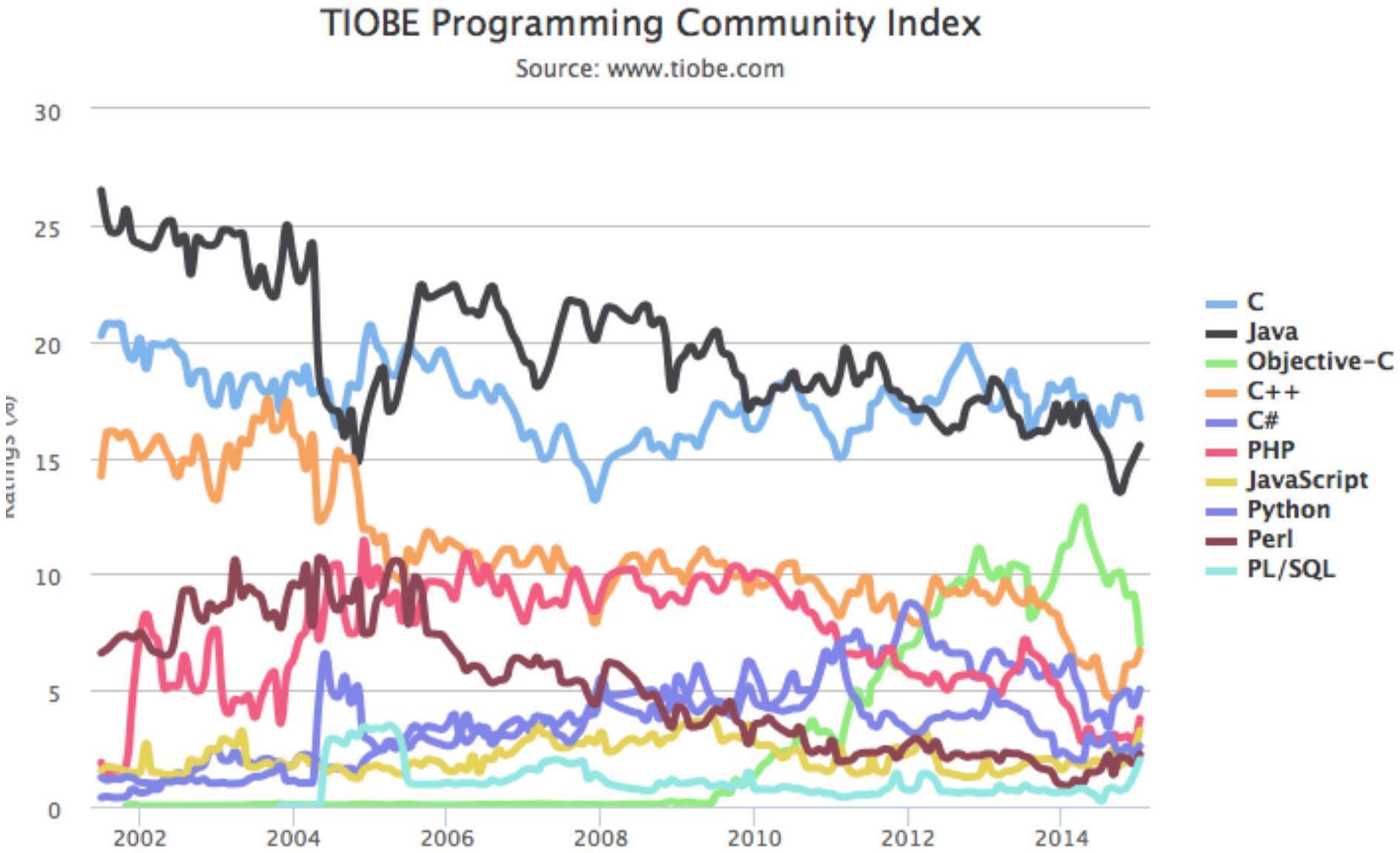
Historique

- Basé sur une technologie des années 80
- ObjC s'inspire de C, Smalltalk
- NextStep (Steve Job)
- Apple rachète NextStep
- Reste de l'histoire dans les types NS…
- Swift rompt avec l'histoire et se rapproche de C#, Java
- Introduction : WWDC du 2 Juin 2014

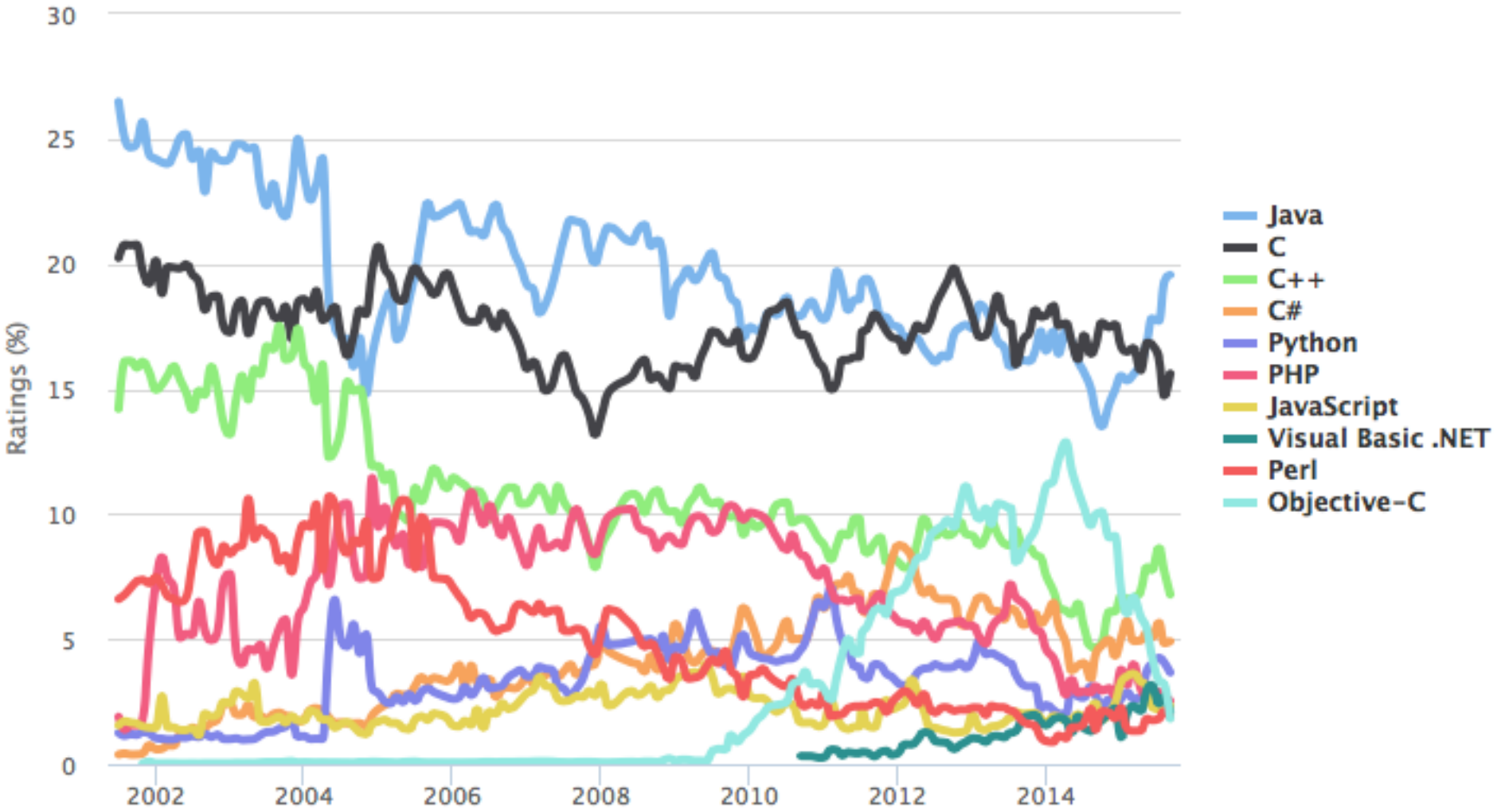
Popularité de l'éco-système

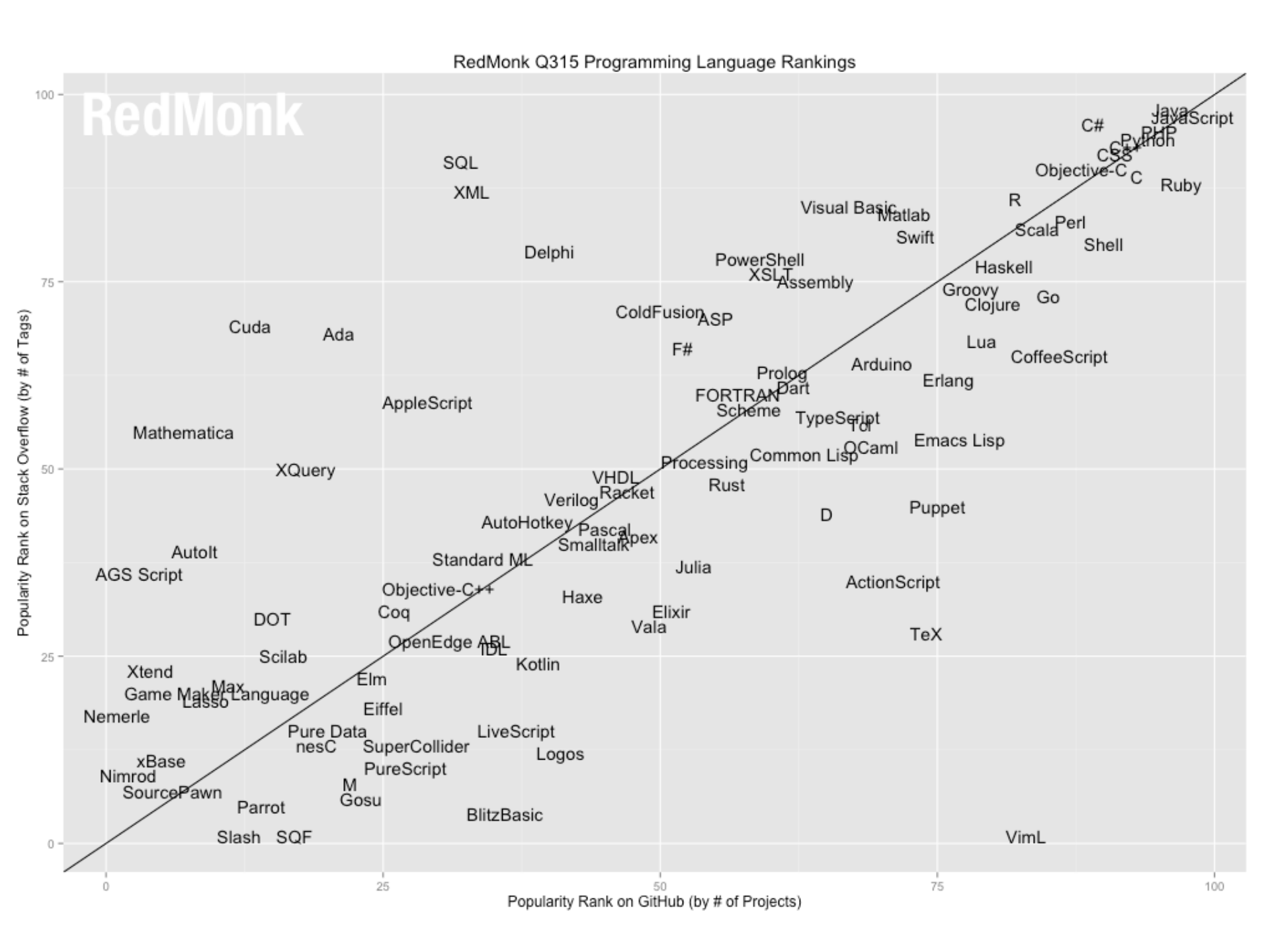
- Point sur l'éco-système iOS :
  - Santé connectée (HealthKit)
  - Domotique (HomeKit)
  - Montre (WatchKit) : Apple Watch
  - Téléphone, tablette, Siri
  - TV
  - iBeacon (Bluetooth LE)
  - iCloud
- Le plus important dans un éco-système, ce n'est le produit (gadget), mais uniquement ce que le gadget autorise derrière.
- Parce que tout les téléphones se ressemblent, la seule différence, c'est l'éco-système auquel il est attaché
- Apple n'est plus une entreprise informatique, au même titre que Google ou Amazon.
- Savoir programmer ou innover ne sert qu'à une seule cause : générer du business et du profit

Popularité de Swift



Popularité de Swift




RedMonk Q315 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)
0
25
50
75
100
Java
JavaScript
C#
PHP
Python
C++
CSS
Objective-C
C
Ruby
R
Visual Basic
Matlab
Perl
Scala
Swift
Shell
SQL
XML
Delphi
PowerShell
XSLT
Assembly
Haskell
Groovy
Clojure
Go
ColdFusion
ASP
Cuda
Ada
F#
Lua
CoffeeScript
Arduino
Prolog
Dart
Erlang
FORTRAN
AppleScript
Scheme
TypeScript
Tcl
Mathematica
OCaml
Emacs Lisp
Common Lisp
Processing
XQuery
VHDL
Racket
Rust
Verilog
Puppet
D
AutoHotkey
Pascal
Apex
Smalltalk
AutoIt
Standard ML
Julia
AGS Script
ActionScript
Objective-C++
Haxe
Coq
Elixir
DOT
Vala
TeX
OpenEdge ABL
IDL
Scilab
Kotlin
Xtend
Elm
Max
Game Maker Language
Lasso
Eiffel
Nemerle
Pure Data
LiveScript
nesC
SuperCollider
Logos
xBase
PureScript
Nimrod
M
SourcePawn
Gosu
Parrot
BlitzBasic
Slash
SQF
VimL

Popularité de Swift

- Redmonk, juin 2015, basé sur GitHub et Stack Overflow
- 1 JavaScript
- 2 Java
- 3 PHP
- 4 Python
- 5 C#
- 5 C++
- 5 Ruby
- 8 CSS
- 9 C
- 10 Objective-C
- 11 Perl
- 11 Shell
- 13 R
- 14 Scala
- 15 Go
- 15 Haskell
- 17 Matlab
- 18 Swift
- 19 Clojure
- 19 Groovy
- 19 Visual Basic

Popularité de Swift

- Classement juillet 2014 :

TIOBE

Jul 2014	Jul 2013	Change	Programming Language	Ratings	Change
1	1		C	17.145%	-0.48%
2	2		Java	15.688%	-0.22%
3	3		Objective-C	10.294%	+0.05%
4	4		C++	5.520%	-3.23%
5	7	⌃	(Visual) Basic	4.341%	+0.01%
6	6		C#	4.051%	-2.16%
7	5	⌄	PHP	2.916%	-4.27%
8	8		Python	2.656%	-1.38%
9	10	⌃	JavaScript	1.806%	-0.04%
10	12	⌃	Transact-SQL	1.759%	+0.19%
11	9	⌄	Perl	1.627%	-0.52%
12	13	⌃	Visual Basic .NET	1.495%	+0.24%
13	37	⌃⌃	F#	1.093%	+0.86%
14	11	⌄	Ruby	1.072%	-0.51%
15	45	⌃⌃	ActionScript	1.067%	+0.86%
16	-	⌃⌃	Swift	1.054%	+1.05%
17	17		Delphi/Object Pascal	1.031%	+0.34%
18	15	⌄	Lisp	0.829%	-0.04%
19	18	⌄	MATLAB	0.781%	+0.10%
20	20		Assembly	0.777%	+0.20%

Sep 2015	Sep 2014	Change	Programming Language	Ratings	Change
1	2	↑	Java	19.565%	+5.43%
2	1	↓	C	15.621%	-1.10%
3	4	↑	C++	6.782%	+2.11%
4	5	↑	C#	4.909%	+0.56%
5	8	↑	Python	3.664%	+0.88%
6	7	↑	PHP	2.530%	-0.59%
7	9	↑	JavaScript	2.342%	-0.11%
8	11	↑	Visual Basic .NET	2.062%	+0.53%
9	12	↑	Perl	1.899%	+0.53%
10	3	⇊	Objective-C	1.821%	-8.11%
11	29	⇈	Assembly language	1.806%	+1.22%
12	13	↑	Ruby	1.783%	+0.50%
13	15	↑	Delphi/Object Pascal	1.745%	+0.59%
14	14		Visual Basic	1.532%	+0.26%
15	17	↑	Pascal	1.298%	+0.40%
16	18	↑	Swift	1.188%	+0.34%

Popularité de Swift

- Classement juillet 2014 : PyPL

Position June 2014	Position June 2013	Delta in position	Programming language	Share in June 2014	Twelve month trends
1	1		Java	26.9 %	+0.2 %
2	2		PHP	13.2 %	-0.8 %
3	4	↑	Python	10.7 %	+0.7 %
4	3	↓	C#	10.2 %	+0.5 %
5	5	↑	C	8.2 %	-0.5 %
6	6	↓	C++	8.2 %	-1.0 %
7	7		Javascript	7.7 %	+0.1 %
8	8		Objective-C	6.7 %	+0.4 %
9	9	↑	Ruby	3.2 %	+0.5 %
10	10	↑	Swift	3.0 %	+1.0 %
© 2014 Pierre Carbonnelle					

Popularité de Swift

- La baisse de la position d'ObjC, va laisser de la place pour d'autres langages, en l'occurrence Swift

Rénovation du langage

- C/C++ -> ObjC -> Swift (juin 2014)
- Compatibilité encore assurée pour l'instant
- Swift = Obj C sans le C !
- Co-existence des deux langages pour l'instant.
- Langage d'avenir pour Apple, plus facile à prendre en main

Plateforme matérielle

- Plateforme de développement -> Intel (OSX)
- Plateforme d'exécution -> ARM (iOS)
- Rappel : Apple maitrise le processus complet de production de ses appareils.
- Pour le hardware : mac book pro, mac book air, mac mini, …
- Pour le software : OSX (Yosemite)
- Pour les processeurs : la logique est sous licence.
- ARM propose un produit générique paramétrable (trous)
- La basse consommation n'est pas supportée par intel -> ARM

Plateforme matérielle

- L'architecture est celle d'une machine de Turing !
- Restrictions :
  - Mémoire encore limitée (128Go)
  - Puissance de calcul encore limitée (64 bits A9 avec coprocesseur)
  - Energie des batteries : gestions des composants (GPS, caméra), gestion des algorithmes coûteux en énergie
  - Petit appareil, qui autorise des mises à jour, et donc une évolution dans le temps (iPad 2 -> iPad Air 2)
  - Les formats d'écran ont évolué avec le temps : 3.5 (4-4S), 4 (5-5C-5S), 4.7 (6), 5.5 (6+), retina (x2)

Plateforme matérielle

- Mémoire Flash (disque dur) : 16 Go à 128 Go
- Capteurs divers : accéléromètre, gyroscope, GPS, capteur de proximité,
- Les capteurs ne sont pas accessibles dans le simulateur
- Le mailComposer n’est pas accessible dans le simulateur

Plateforme logicielle

- NeXT hérite d'UNIX (1969)
- Apple hérite de NeXT (1986)
- OSX hérite de NeXTStep l'OS NeXT
- OSX est donc un UNIX Like
- Un UNIX like n'est pas un LINUX like -> Androïd
- Les commandes diffères, les paramètres, les dossiers, comportent des différences
- Linux a été développé de manière indépendante, sur les mécanismes UNIX

Interface

- Changement du design depuis iOS 7 (flat design)
- iOS 9 est actuellement utilisé
- Les différents SDK font apparaître :
  - Device -> iPhone, iPad, iPod
  - Wearable -> Apple Watch
  - Car -> carPlay
  - TV -> Apple TV
  - Health -> kit (pas de produit spécifique ->tiers)
  - House -> kit (pas de produit spécifique -> tiers)

Interface

- Programmation réactive -> proche de la programmation événementielle -> événements (TouchDown, ValueChanged, …)
- Gestuel -> événements haut niveau (OS).
- L'OS attrape les événements avec mécanismes de surveillance (pool)
- Interruptions, déroutements (reset, division par 0, …) -> événements bas niveau (matériel)
- Chaque événement matériel, possède une adresse de déroutement
- Modèle MVC par défaut
- Les différents devices, wearables, nécessitent une adaptation de l'affichage des applications (Universal Story Board)

iOS vs Android

- Le marché visé n'est pas le même
- Androïd -> monde libre à la base (Java)
- iOS -> monde plus propriétaire (ObjC, Swift)
- Androïd -> marché de masse, grande diversité de device
- Androïd -> PlayStore : politique de publication (exigences)
- iOS -> marché premium, nombre de device limité
- iOS -> AppStore : exigences plus élevés, contraintes

iOS vs Androïd

- Même format d'appareil -> limite dans les fonctionnalités -> ressemblances logiques
- Java -> Eclipse + SDK Google
- Swift -> Xcode + SDK Apple

Développement

- Compilation croisée : compilation pour une plateforme cible, différente de la plateforme de développement d'origine
- C'est le principe même des applications embarquées (satellites, voitures, avions)
- Cible -> Simulateur, device Physique (Programme de développement plus nécessaire suivant le besoin)
- La publication sur l'App Store nécessite toujours un programme de développement payant (90 euros)

Compilation croisée

- La cible n'est pas la même que l'origine :

édition - compilation
exécution dans un simulateur
(compilation native)

exécution native
(compilation croisée)

Les programmes de développement

iOS

iOS Developer Program
Individual
$99 / Year

For an individual developer who will be creating iOS apps for distribution on the App Store.

iOS Developer Program
Company
$99 / Year

For a company with a development team who will be creating iOS apps for distribution on the App Store.

iOS Developer Enterprise Program
$299 / Year

For a company who will be creating proprietary, in-house iOS apps.

Note: A D-U-N-S Number is required.

iOS Developer University Program
Free

For higher education institutions looking to introduce iOS development into their curriculum.

Mac

Mac Developer Program
Individual
$99 / Year

Mac Developer Program
Company
$99 / Year

Les programmes de développement

- Enterprise Program :

Distribute In-house Apps

Distribute proprietary, in-house iOS apps to your employees. You can also securely host and wirelessly distribute or update in-house apps to employees, keeping them current anywhere, anytime.

Test on iOS Devices

See how your development app will perform in a real-world environment by installing and testing it directly on iPad, iPhone, and iPod touch.

Receive Code Level Technical Support

Request support from Apple engineers for code level issues and technical guidance to fastrack your development process. The iOS Developer Enterprise Program includes two Technical Support Incidents per membership year.

Student Development Team

The iOS Developer University Program allows instructors and professors to create a development team with up to 200 students.

Development Resources

With the suite of sophisticated and elegant tools included in the iOS SDK, and a wide-range of resources in the iOS Dev Center, students participating in the class will have everything they need to create innovative applications for iPad, iPhone, and iPod touch.

Test on iPad, iPhone, and iPod touch

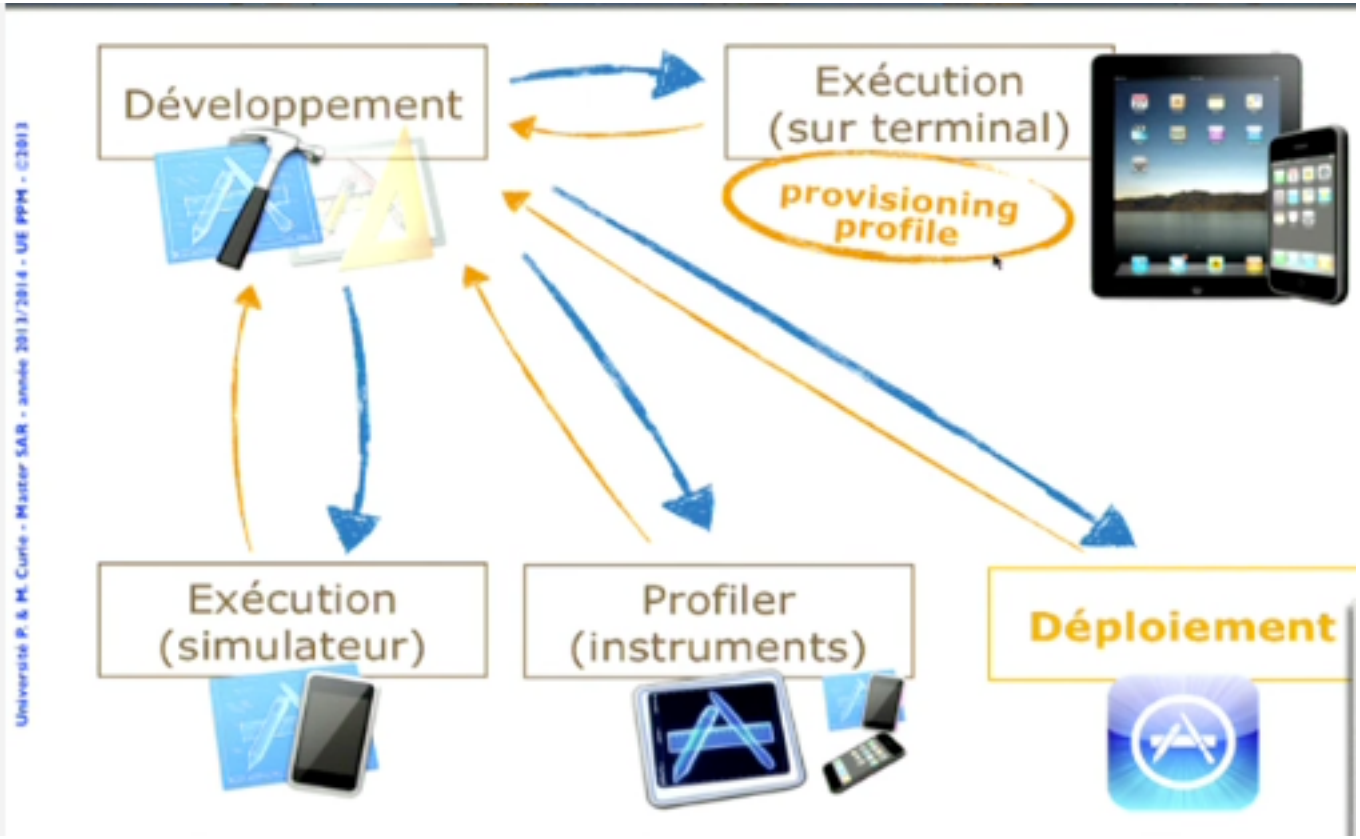
Students can test and debug their applications using Xcode's graphical debugger, or collect real-time performance data in Instruments' timeline view. To see their work in action and ensure proper functionality, students can install their applications directly on iPad, iPhone and iPod touch.

Sharing Applications

The iOS Developer University Program allows students and professors within the same team to share their development applications with each other through email, or by posting them to a private website for presentation and grading purposes.

Workflow apple

- L'enchaînement des étapes :



Déploiement

- Déploiement local : USB (provisioning profile)
- Via AppStore (30% de taxe)
- Dans les deux cas Xcode gère le processus

- Via intranet, Xcode génère un fichier .ipa
- .ipa déposé sur serveur
- Récupération de l'.ipa par iTunes d'un client, installation par câble
- OTA : c'est ce que vous faîtes tous les jours

Qualité des applications

- Le processus de déploiement est assez complexe
- Beaucoup de certificats et donc de système de sécurité
- Seule assurance, de retrouver l'auteur d'un malWare (AppStore)
- La validation vérifie les API invoquées dans les App, afin de ne pas autoriser n'importe quoi
- Seule garantie d'avoir une application conforme
- Format propriétaire imposé : pas de discussion possible

Qualité des applications

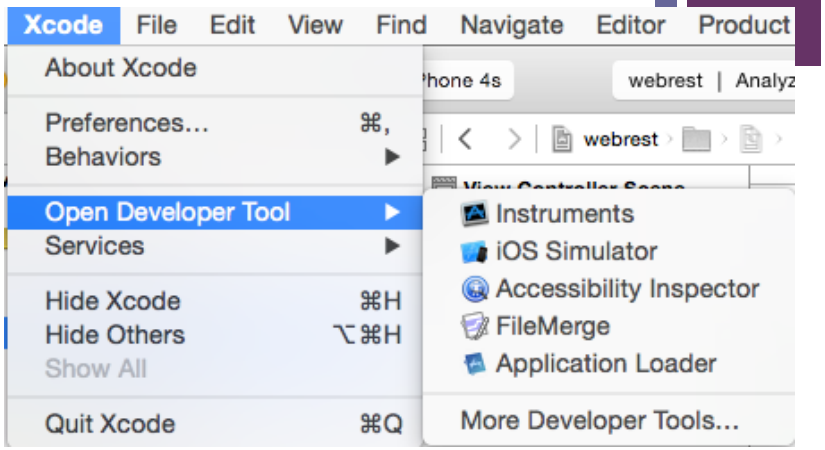
- Androïd rattrappe son retard sur la vérification des Apps (conditions moins contraignantes)
- La qualité des Apps, fait la qualité de la plateforme, de l'écosystème, et donc la réputation
- Un « scp » est encore possible sur Androïd
- Impossible avec iOS (nativement)
- La taille des apps limitée à 2 Go depuis la création de l'app store, passe à 4 Go en 2015
- Se pose la question d'une mémoire de 16 Go ?

IDE

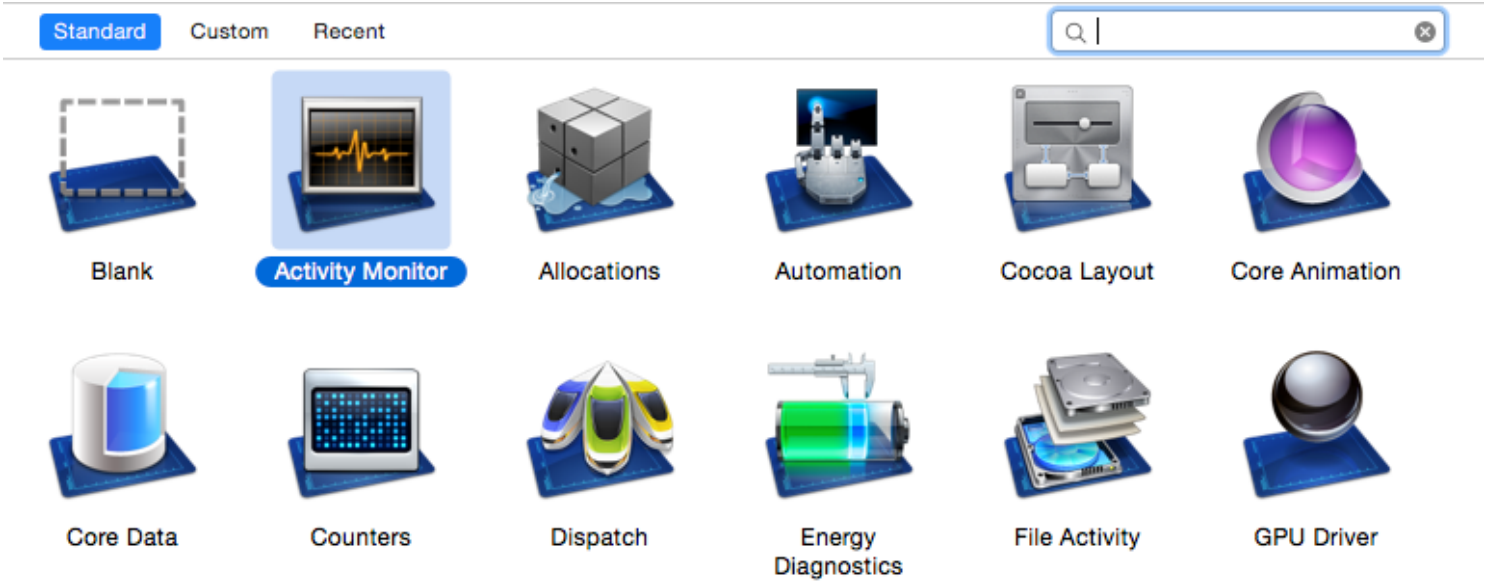
- Plateforme de développement : Xcode
- Xcode (X86 -> 64 bits) : cible Intel (OSX 64 bits) ou ARM (iOS 32-64 bits)
- Xcode se rapproche de plus en plus de VS ou Eclipse
- Xcode : Interface Builder -> GUI
- Compilateur : LLVM (principe de machine virtuelle) -> frameWork
- Debugger : LLDB (GDB pour LLVM)

Instruments

- Xcode offre des outils d'analyse :



- Outils



Instruments

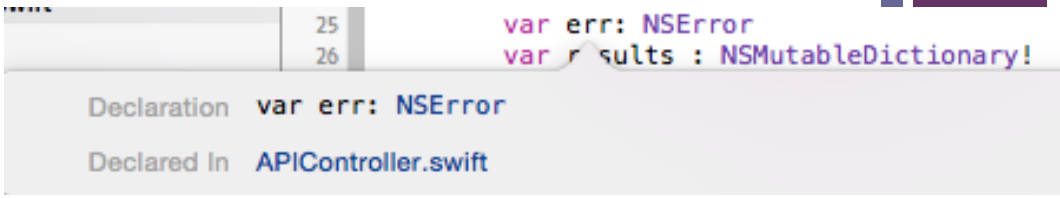
- Gestion mémoire : Allocations
- Gestion système : CPU, Activity Monitor, File Activity
- Gestion graphique : GPU, Core Animation
- Gestion données : Core Data
- Scénarios de test : Automation

Versioning

- Git, SVN
- Repository Local si activé dans le projet
- OSX serveur nécessaire, pour Team development
- Client gitHub conseillé pour github.com

Aide

- Aide en ligne : alt+ click
- Accès interface = Cmd + click
- Esc : complétion

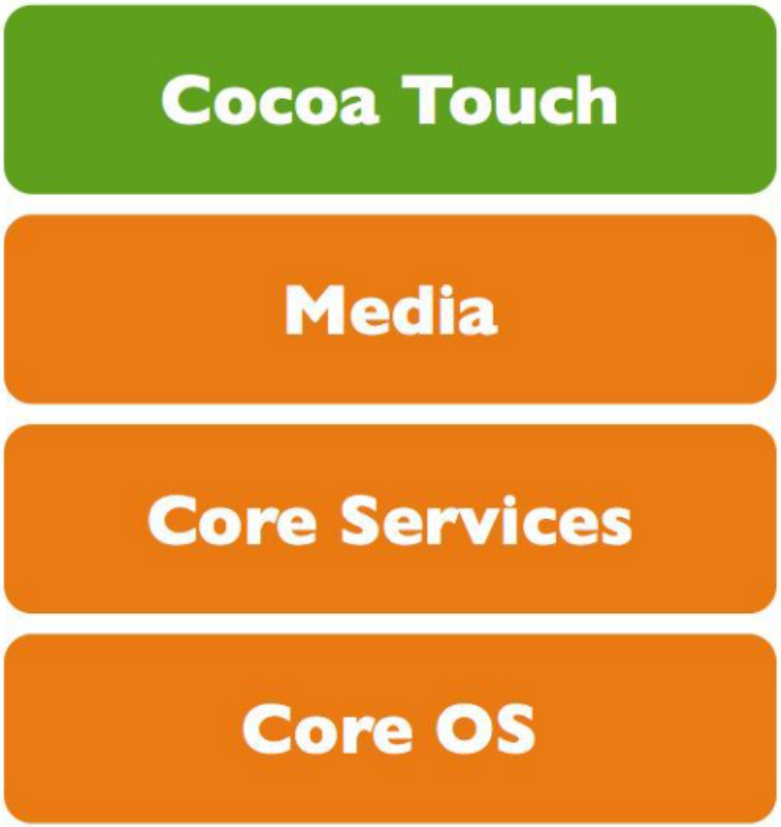


Couches iOS

- Core OS : même noyau OSX (noyau BSD Unix) + micro noyaux mach3
- Core OS : système de fichiers, trousseau, « Bonjour »,
- Core Service : services réseaux, géo-localisation, préférences, Contacts, SQLite
- Core médias : format de fichiers (TIFF, JPG, PNG, PDF), audio, vidéo, openGL, sprite kit
- Cocoa touch :UIKit, API iOS, Foundation, services de base
- Androïd possède Linux à la place de BSD

Couches iOS

- En bas les couches proches du matériel :



Couches iOS

- Beaucoup de parties anciennes sont écrites en langage C
- C : possibilité de manipuler des pointeurs -> adressage, mappage
- Assembleur : manipulation du jeu d'instructions du processeur (démarrage, routines système, …)
- ObjC et Swift ont tendance à encapsuler les pointeurs
- ARC -> très simplifié avec Swift (Automatic Reference Counting)
- Néanmoins quelques connaissances de base sont évidentes pour gérer la mémoire (alloc/dealloc)