iOS

Interview

https://codefights.com/

<https://www.hackerrank.com/dashboard>

<https://leetcode.com/>

<http://www.geeksforgeeks.org/>

SOLID

<https://marcosantadev.com/solid-principles-applied-swift/>

DI

<https://www.objc.io/issues/15-testing/dependency-injection/>

Architecture

<https://swifting.io/blog/2016/09/07/architecture-wars-a-new-hope/>

<https://academy.realm.io/posts/benji-encz-unidirectional-data-flow-swift/>

<https://marcosantadev.com/mvvmc-with-swift/>

<https://marcosantadev.com/new-ios-software-architecture-4v-engine/>

CI

<https://www.thoughtworks.com/continuous-integration>

<https://www.raywenderlich.com/136168/fastlane-tutorial-getting-started-2>

<https://dzone.com/articles/pros-and-cons-of-using-xcode-server-for-ios-contin>

<https://dzone.com/articles/xcode9-xcode-server-comprehensive-ios-continuous-i>

<https://www.safaribooksonline.com/library/view/ios-11-swift/9781491992463/ch01.html>

<https://medium.com/cocoaacademymag/creating-a-marvel-ios-app-from-scratch-part-4-sketch-for-developers-2344a221482a>

<https://medium.com/xcblog/xcode9-xcode-server-comprehensive-ios-continuous-integration-3613a7973b48>

TEST

<https://blog.codecentric.de/en/2015/11/introduction-to-tdd-in-ios/>

<https://dzone.com/articles/tips-for-writing-scalable-xcuitests-wwdc-2017?fromrel=true>

mvvm

<http://www.sprynthesis.com/2014/12/06/reactivecocoa-mvvm-introduction/>

<https://www.slideshare.net/Rambler-iOS/life-with-out-of-memory>

<https://www.slideshare.net/Rambler-iOS/ss-76457847>

comment

<https://developer.apple.com/library/content/documentation/Xcode/Reference/xcode_markup_formatting_ref/index.html#//apple_ref/doc/uid/TP40016497-CH2-SW1>

ObjC

<https://github.com/TinkoffCreditSystems/iosDynamicLibraryBuilder>

<https://www.objc.io/issues/19-debugging/debugging-case-study/>

Swift

pages

<https://www.swiftbysundell.com>

http://agostini.tech

<http://www.ioscookies.com/>

<https://www.toptal.com/developers/blog/mobile>

<https://littlebitesofcocoa.com/>

<https://oleb.net/>

GCD

<https://www.swiftbysundell.com/posts/a-deep-dive-into-grand-central-dispatch-in-swift>

map-filter-reduce

<https://medium.com/@KirillChukhlib/map-filter-reduce-in-swift-simple-higher-order-functions-ff08a686fbc6>

<http://agostini.tech/2017/05/01/creating-a-cocoapod/?utm_content=buffer31a94&utm_medium=social&utm_source=linkedin.com&utm_campaign=buffer>

<http://rasic.info/bindings-generics-swift-and-mvvm/>

<https://medium.com/flawless-app-stories/how-to-use-a-model-view-viewmodel-architecture-for-ios-46963c67be1b>

<https://medium.com/@azamsharp/look-maa-server-side-swift-using-vapor-638cc3f5a5a8>

<https://medium.com/flawless-app-stories/getting-started-with-reusable-frameworks-for-ios-development-f00d74827d11>

<https://jobandtalent.engineering/ios-architecture-an-state-container-based-approach-4f1a9b00b82e>

create lib (cocoapods, carthage, swift package manager)

<https://medium.com/cocoaacademymag/creating-an-ios-framework-with-cookiecutter-swift-97f7965b83df>

brew install carthage

brew install cookiecutter

cookiecutter <https://github.com/cookiecutter-swift/FrameworkTemplate>

create lib (cocoapods)

<http://agostini.tech/2017/05/01/creating-a-cocoapod/>

pod lib create AH...

sudo gem install colored2

sudo gem install cocoapods --pre

pod trunk register aro.hak.25@gmail.com 'Ara Hakobyan' --description='macbook pro'

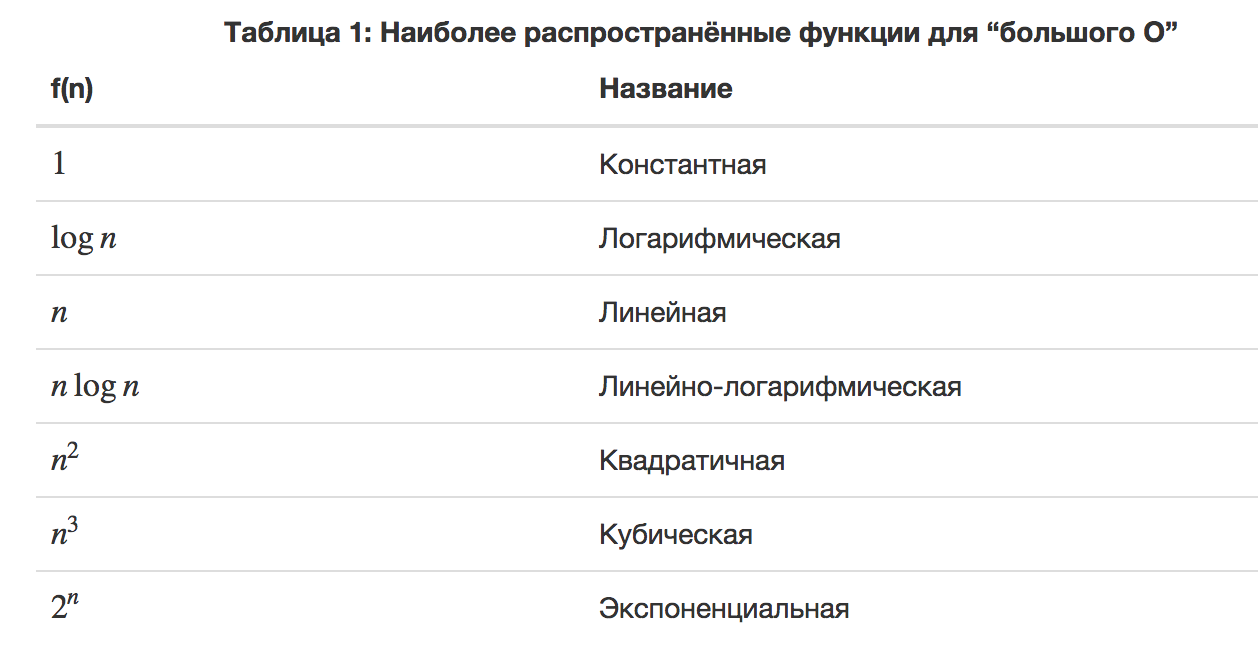
echo "3.2" > .swift-version

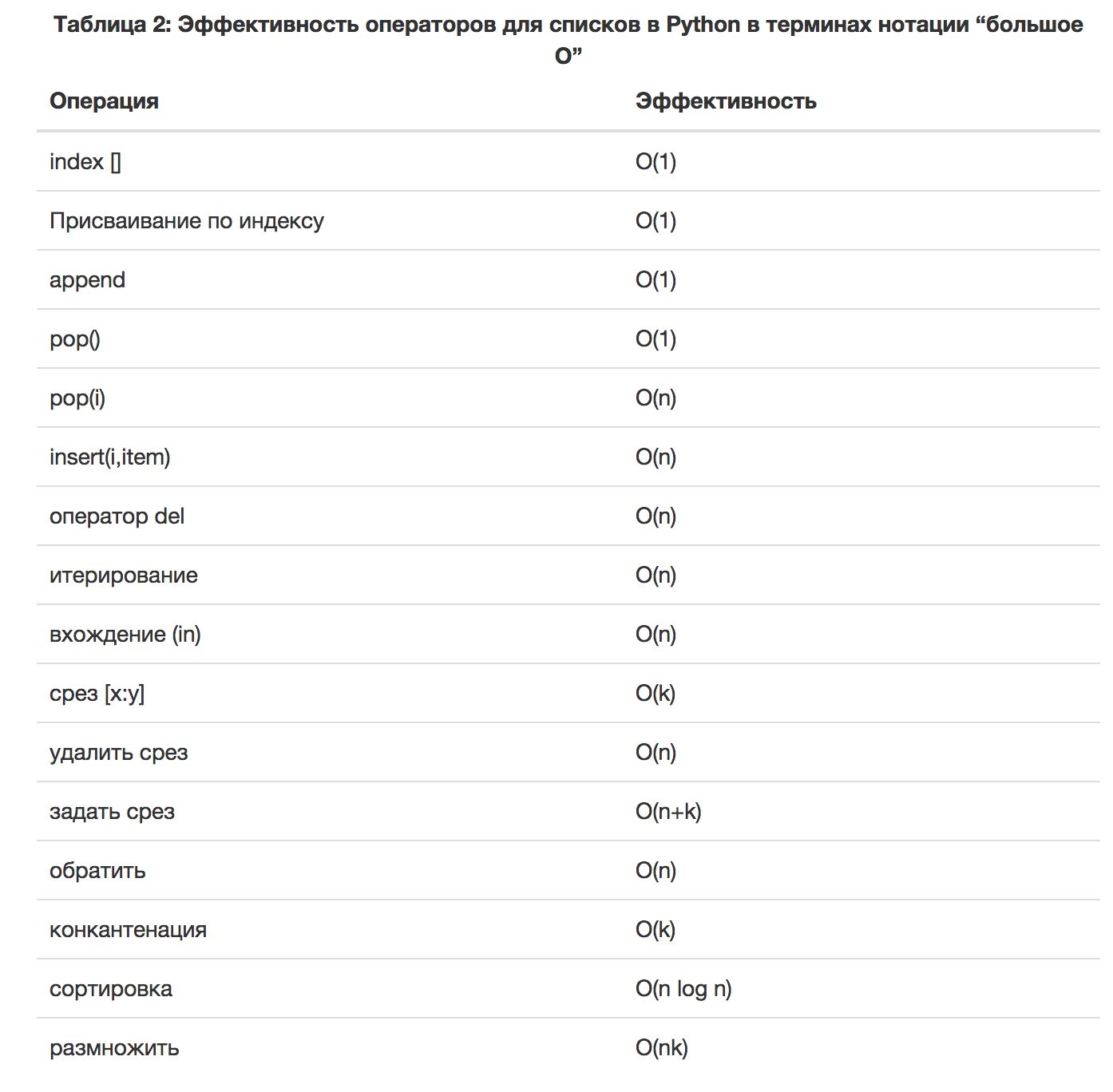
pod lib lint AH.podspec --no-clean

pod trunk push AH.podspec --verbose

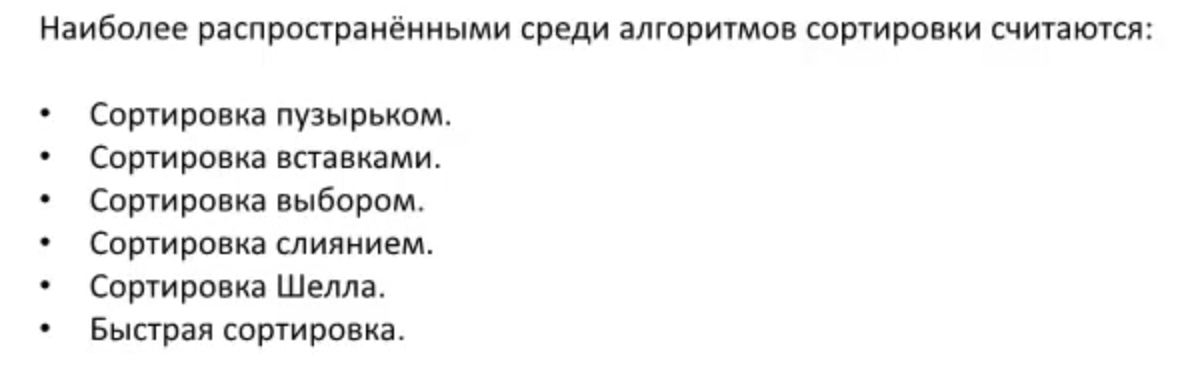
Algoritms and Structures

Macintosh HD:Users:test:Desktop:sum.png





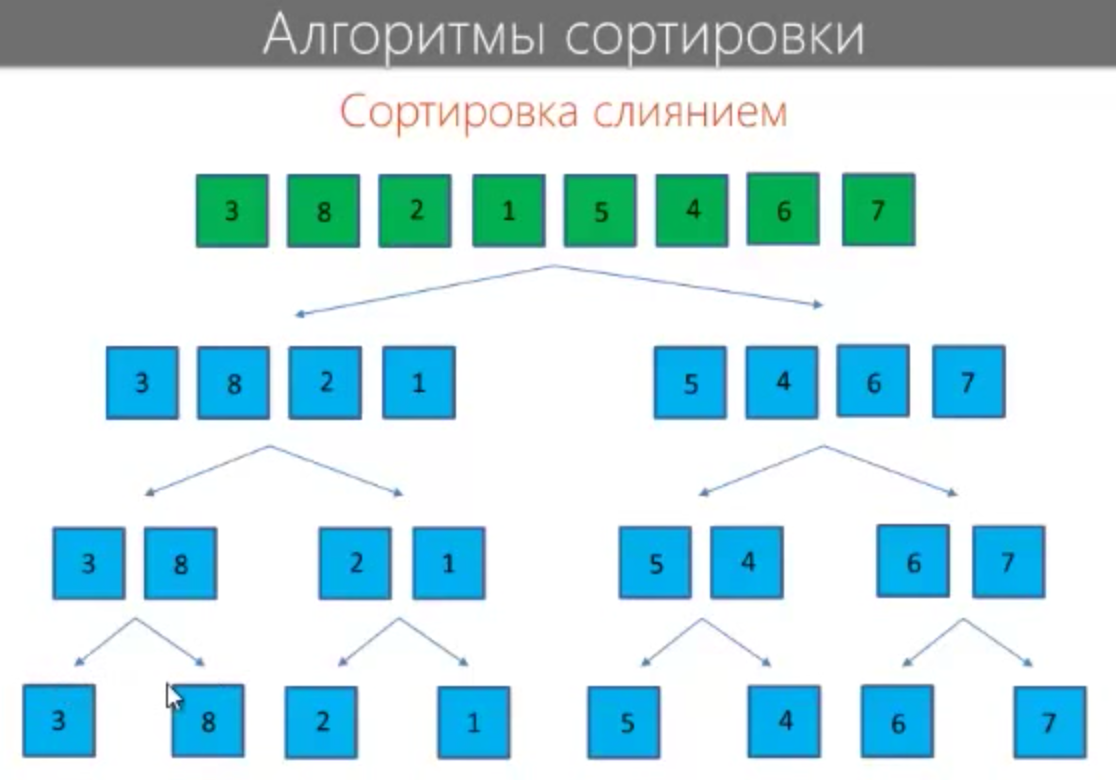










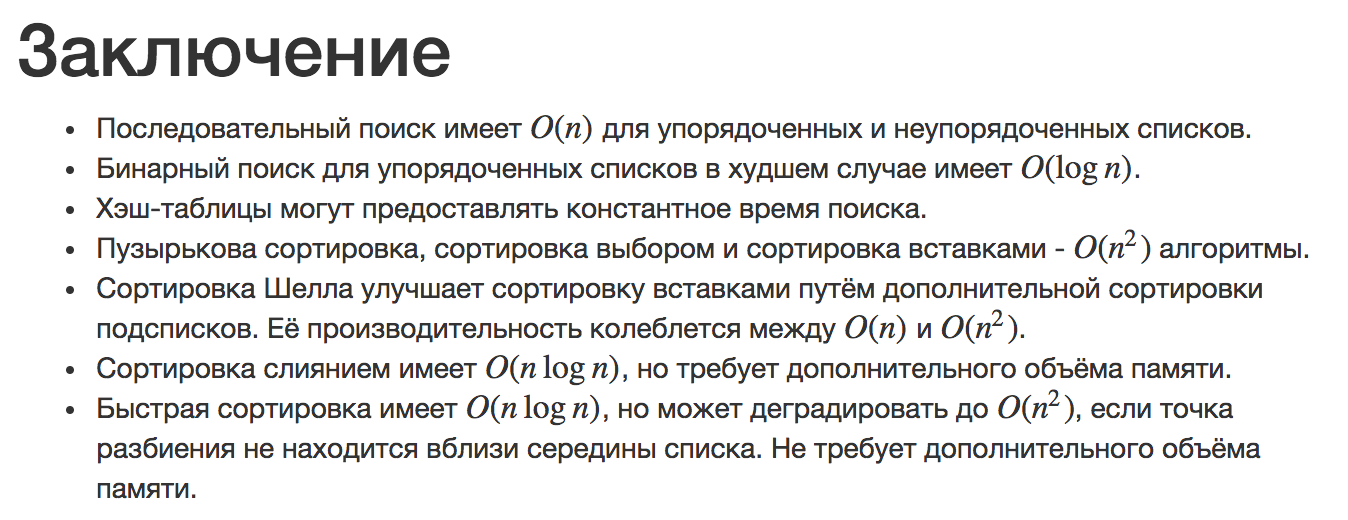




Основные структуры данных



Сортировка и поиск



|  |  |
| --- | --- |
| Create and have ownership of it | alloc/ne alloc/new/copy/mutableCopy group |
| Take ownership of it | retain |
| Relinquish it | release |
| Dispose of it | dealloc |

non-ARC

id obj = [[NSObject alloc] init];

/\*

\* Now you have ownership of the object.

\*/

id obj = [NSMutableArray array];

/\*

\* The obtained object exists and you don’t have ownership of it.

\*/

[obj retain];

/\*

\* Now you have ownership of the object.

\*/

[obj release];

/\*

\* The object is relinquished.

\* You can’t access the object anymore.

\*/

ARC

1. <http://xydan.livejournal.com/7531.html>

Ownership qualifier

1. \_\_strong
2. \_\_weak
3. \_\_unsafe\_unretained
4. \_\_autoreleasing

id \_\_strong obj = [NSMutableArray array];

/\*

\* Obtain an object without creating it yourself or having ownership

\*/

id \_\_weak obj = [[NSObject alloc] init];

/\*

\* weak reference does not have ownership of the object.

\* The variable obj is qualified with \_\_weak.

\* When the source code is compiled, the compiler shows a warning message.

\*/

id \_\_strong obj0 = [[NSObject alloc] init];

/\*

\* The variable obj0 is qualified with \_\_strong.

\* Which means, it has ownership of the object.

\*/

id \_\_weak obj1 = obj0;

/\*

\* variable obj1 has a weak reference of the created object

\*/

\*\_\_weak example

id \_\_weak obj1 = nil;

{

id \_\_strong obj0 = [[NSObject alloc] init];

obj1 = obj0;

}

The result is:

A: <NSObject: 0x753e180>

B: (null)

\*\_\_unsafe\_unretained example

id \_\_unsafe\_unretainedobj1 = nil;

{

id \_\_strong obj0 = [[NSObject alloc] init];

obj1 = obj0;

}

The result is:

A: <NSObject: 0x753e180>

B: (crash)

* Forget about using retain, release, retainCount, and autorelease.
* Forget about using NSAllocateObject and NSDeallocateObject.
* Follow the naming rule for methods related to object creation.
* Forget about calling dealloc explicitly.
* Use @autoreleasepool instead of NSAutoreleasePool.
* Forget about using Zone (NSZone).
* Object type variables can’t be members of struct or union in C language.
* ‘id’ and ‘void\*’ have to be cast explicitly.

\*non-ARC

id obj = [[NSObject alloc] init];

void \*p = (\_\_bridge void \*)obj;

id o = (\_\_bridge id)p;

\*ARC

id obj = [[NSObject alloc] init];

void \*p = (\_\_bridge\_retained void \*)obj;

\*non-ARC

id obj = (id)p;

[obj retain];

\*ARC

id obj = [[NSObject alloc] init];

void \*p = (\_\_bridge\_transfer void \*)obj;

\*non-ARC

id obj = (id)p;

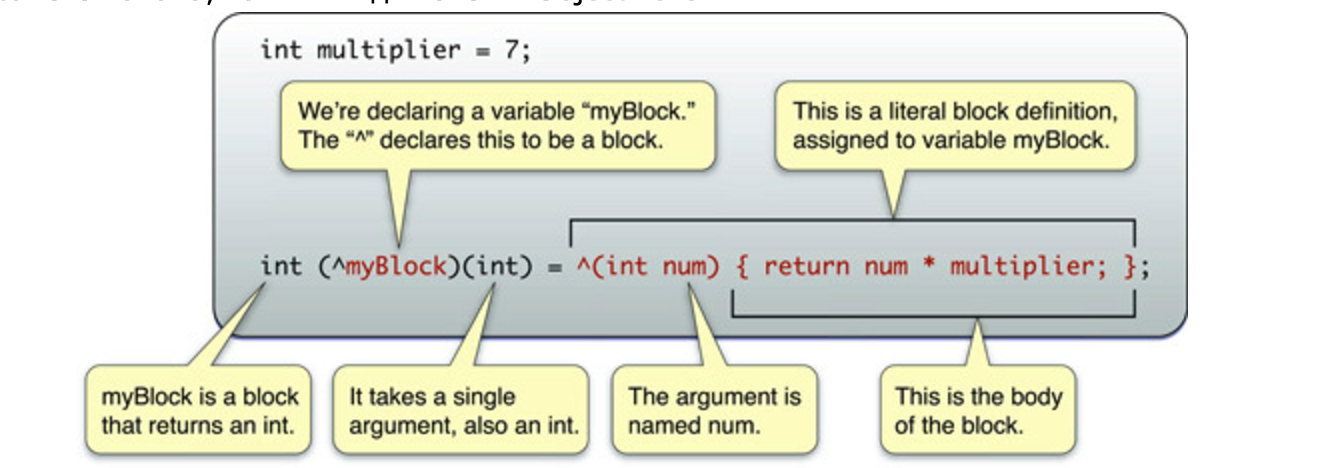
[obj retain];

[(id)p release];

|  |  |
| --- | --- |
| Property modifier | Ownership qualifier |
| assign | \_\_unsafe\_unretained(note: new copied object is assigned.) |
| copy | \_\_strong |
| retain | \_\_strong |
| strong | \_\_strong |
| unsafe\_unretained | \_\_unsafe\_unretained |
| weak | \_\_weak |

For static arrays, the compiler will insert code to release all the entries automatically when the variable scope is left. But for dynamic arrays, the compiler can’t detect their lifetimes.

BLOCK



1. <https://habrahabr.ru/post/271255/>
2. <https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/WorkingwithBlocks/WorkingwithBlocks.html#//apple_ref/doc/uid/TP40011210-CH8-SW1>

^int (int count) { return count + 1; }

/\*­

\* full syntaxis

\*/

^(int count) { return count + 1; }

/\*

\* short syntaxis

\*/

int (^blk)(int) = ^(int count) { return count + 1; };

int (^blk1)(int) = blk;

int (^blk2)(int);

blk2 = blk1;

/\*

\* declare variables and assigning

\*/

* (void)someMethod:(returnType (^)(parameterTypes))blockName;

/\*

\* declare block function parameter

\*/

typedef int (^blk\_t)(int);

/\*

\* declare variables of “blk\_t” type

\*/

\_\_block int val = 0;

void (^blk)(void) = ^{

val = 1;

};

blk();

/\*

\* with \_\_block

\*/

id array = [[NSMutableArray alloc] init];

void (^blk)(void) = ^{

id obj = [[NSObject alloc] init];

[array addObject:obj];

};

blk();

/\*

\* without \_\_block

\*/

const char text[] = "hello";

const char \*text = "hello";

void (^blk)(void) = ^{

printf("%c\n", text[2]);

};

/\*

text[] wrong

\*text right

\*/

GCD

Serial Dispatch Queue

/\*

\* Waiting the task is finished

\*/

Concurrent Dispatch Queue

/\*

\* Not waiting the task is finished

\*/

dispatch\_queue\_t myConcurrentDispatchQueue = dispatch\_queue\_create(

"com.example.gcd.MyConcurrentDispatchQueue", NULL);

dispatch\_async(myConcurrentDispatchQueue,

^{NSLog(@"block on myConcurrentDispatchQueue");});

/\*

\* In this source code, the Block runs in the serial dispatch queue.

\*/

dispatch\_queue\_t myConcurrentDispatchQueue = dispatch\_queue\_create(

"com.example.gcd.MyConcurrentDispatchQueue", DISPATCH\_QUEUE\_CONCURRENT);

dispatch\_async(myConcurrentDispatchQueue,

^{NSLog(@"block on myConcurrentDispatchQueue");});

/\*

\* In this source code, the Block runs in the concurrent dispatch queue.

\*/

dispatch\_queue\_t mainDispatchQueue = dispatch\_get\_main\_queue();

/\*

\* How to get the main dispatch queue

\*/

dispatch\_queue\_t globalDispatchQueueHigh =

dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_HIGH, 0);

/\*

\* How to get a global dispatch queue of high priority

\*/

dispatch\_queue\_t globalDispatchQueueDefault =

dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0);

/\*

\* How to get a global dispatch queue of default priority

\*/

dispatch\_queue\_t globalDispatchQueueLow =

dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_LOW, 0);

/\*

\* How to get a global dispatch queue of low priority

\*/

dispatch\_queue\_t globalDispatchQueueBackground =

dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_BACKGROUND, 0);

/\*

\* How to get a global dispatch queue of background priority

\*/

dispatch\_time\_t time = dispatch\_time(DISPATCH\_TIME\_NOW, 3ull \* NSEC\_PER\_SEC);

dispatch\_after(time, dispatch\_get\_main\_queue(), ^{

NSLog(@"waited at least three seconds.");

});

/\*

\* dispatch\_after

\*/

dispatch\_queue\_t queue =

dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0);

dispatch\_group\_t group = dispatch\_group\_create();

dispatch\_group\_async(group, queue, ^{NSLog(@"blk0");});

dispatch\_group\_async(group, queue, ^{NSLog(@"blk1");});

dispatch\_group\_async(group, queue, ^{NSLog(@"blk2");});

dispatch\_group\_notify(group,

dispatch\_get\_main\_queue(), ^{NSLog(@"done");});

dispatch\_release(group);

/\*

\* dispatch\_group

\*/

#define ANIMATION\_DURATION 0.3

static const NSTimeInterval kAnimationDuration = 0.3; (.m)

extern NSString \*const EOLoginNotification; (.h)

NSString \*const EOLoginNotification = @"EOLoginNotification"; (.m)

Item 12: Understand Message Forwarding

1. Dynamic Method Resolution

The first method that’s called when a message is passed to an object that it doesn’t

understand is a class method on the object’s class:

+ (BOOL)resolveInstanceMethod:(SEL)selector

2. Replacement Receiver

The second attempt to handle an unknown selector is to ask the receiver whether a

replacement receiver is available to handle the message instead. The method that handles

- (id)forwardingTargetForSelector:(SEL)selector

3. Full Forwarding Mechanism

If the forwarding algorithm has come this far, the only thing left to do is to apply the

full forwarding mechanism. This starts by creating an NSInvocation object to wrap up all

the details about the message that is left unhandled. This object contains the selector,

target, and arguments. An NSInvocation object can be invoked, which causes the

message-dispatch system to whir into action and dispatch the message to its target.

The method that gets called to attempt forwarding this way is:

- (void)forwardInvocation:(NSInvocation\*)invocation



1. self? + runtime - <https://habrahabr.ru/post/270913/>
2. interview- <https://github.com/sashakid/ios-guide>
3. Intruduce yourself (background, work)
4. Property (strong, copy, weak, assign, anseft)
5. Block, Dispatch, NSOperetaion (deadlock)
6. UIKit (UIWindow, loadView)
7. NSURLSession
8. CoreData
9. CI, CD

<https://github.com/vsouza/awesome-ios>

<https://www.youtube.com/channel/UCml4lCH0xdl6Jm91RiPPIig>

<https://zeef.com/>

<https://hackr.io/>

<https://medium.com/flawless-app-stories/debugging-swift-code-with-lldb-b30c5cf2fd49>