阿里云iOS热修复Lua语法说明

热修复补丁使用Lua脚本语言编写，Lua基本语法参考：Lua基本语法。

Lua补丁基础语法

创建对象不需要alloc，直接init，调用方法使用冒号。例如:

```
local view = UIView:init()
```

获取OC属性使用方法，不能使用`点语法`，例如:

```
view:backgroundColor()
```

设置OC属性使用setXXX方法，不能直接`=`复制，例如:

```
view:setBackgroundColor(UIColor:blueColor())
```

BOOL值使用true和false，多个参数的方法使用下划线拼接，括号里依次传递参数。

```
UIApplication:sharedApplication():setStatusBarHidden_animated(true, false)

UIAlertView:initWithTitle_message_delegate("title", "message", nil)
```

注意：Lua语法中只有nil和false为假，其它数字0，空字符串都为真。

如何创建 / 新增一个UIViewController，例如:

```
interface{"MyController", UIViewController}

function init(self)

  -- super很特殊，需要用dot操作

  self.super:initWithNibName_bundle("MyControllerView.xib", nil)

  return self

end

function viewDidLoad(self)

  -- 逻辑处理
```

```
end
```

新增类，并声明协议，例如：

```
interface{"MyClass", NSObject, protocols = {"UITableViewDelegate",
"UITableViewDataSource”}}
```

替换已经存在的class的方法

可不需要声明父类，方法里第一个参数为self，指定需替换的方法。例如:

```
interface{"MyController"}

function viewDidLoad(self)

  --

end
```

替换类的category方法

与替换已经存在的class的方法一致，例如:

```
@interface SDKServer (Async4j)

- (void)startAsync4jRequest;

@end

--Lua代码

interface{"SDKServer"}

function startAsync4jRequest(self)

  --

end
```

调用原有的方法，使用`ORIG`前缀，例如：

```
interface{"MyController"}

function viewDidLoad(self)

  -- 前置逻辑

     self:ORIGviewDidLoad() --不要传self

  -- 后置逻辑

end
```

hook带下划线的方法，可使用 'UNDERxLINE' 代替 '_'。

```
--OC - (void)_prefixA:(NSString *)a _b:(NSString *)b

function UNDERxLINEprefixA_UNDERxLINEb(self, a, b)

    self:ORIGUNDERxLINEprefixA_UNDERxLINEb(TEST_VALUE_STRING, TEST_VALUE_STRING)

end

--OC - (id)__aa_bb_:(NSString *)v1 _cc__dd_ :(NSString *)v2 ___ee___f___:(NSString
*)v3

function
UNDERxLINEUNDERxLINEaaUNDERxLINEbbUNDERxLINE_UNDERxLINEccUNDERxLINEUNDERxLINEddUND
ERxLINE_UNDERxLINEUNDERxLINEUNDERxLINEeeUNDERxLINEUNDERxLINEUNDERxLINEfUNDERxLINEU
NDERxLINEUNDERxLINE(self, v1, v2, v3) return
self:ORIGUNDERxLINEUNDERxLINEaaUNDERxLINEbbUNDERxLINE_UNDERxLINEccUNDERxLINEUNDERx
LINEddUNDERxLINE_UNDERxLINEUNDERxLINEUNDERxLINEeeUNDERxLINEUNDERxLINEUNDERxLINEfUN
DERxLINEUNDERxLINEUNDERxLINE("abc", "efg", "hij")

end
```

带下划线的方法不能直接调用，例如：

```
--有返回值用此方法(如果无返回值用此方法会导致在32位机器crash)

local res = self:performSelector_withObject("_xxxFunction", nil);

self:UNDERxLINExxxFunction();

--无返回值用此方法

self:performSelector_withObject_afterDelay("doRateItem_Function", nil, 0);

self:doRateItemUNDERxLINEFunction()
```

@selector直接使用字符串，例如：

```
if self:respondsToSelector("funca:b:c:") then

     self:funca_b_c(xxx, xx, xx)

end
```

新增方法（返回值和参数均为id类型）

```
function myLostMethod(self)
    print("myLostMethod")
    return nil
end

function myLostMethod_other(self, a, b)
    print("myLostMethod_other");
    return nil
end
```

新增属性property

```
--方式1：直接使用lua自带的新增属性功能，但是只能被Lua使用

self.xx = "abcd"

print(self.xx)

--方式2：模拟OC的增加属性功能，可在OC访问

function xxx(self)

return self._xxx
```

```
end

function setXxx(self, xxx)

self._xxx = xxx

end
```

创建struct，例如:

```
-- 创建一个全局函数叫做CGRect，带有4个float参数；第二个参数"ffff"，定义了4个参数的类型
arp.struct.create("MyRect", "ffff", "x", "y", "width", "height”)

local rect = MyRect(1, 2, 3, 4)

    print(rect.x) --> 1

    rect.x = 200

    print(rect.x) --> 200

注意这些struct已经内置可直接使用,不要创建:CGSize，CGPoint，UIEdgeInsets，CGRect，
NSRange，CLLocationCoordinate2D，MKCoordinateSpan，MKCoordinateRegion，
CGAffineTransform，UIOffset
```

CGRect，CGSize，CGPoint使用：

```
local size = CGSize(1, 2)

local frame = CGRect(1, 2, 3, 4)

self:animateLabel():setFrame(frame)

size = CGSize(frame.width, frame.height)

local point = CGPoint(frame.x, frame.y)

--OC CGFloat width = [UIScreen mainScreen].bounds.size.width;

local width = UIScreen:mainScreen():bounds().width

-- CGSize，CGPoint，UIEdgeInsets，CGRect，NSRange，CLLocationCoordinate2D，
MKCoordinateSpan，MKCoordinateRegion，CGAffineTransform，UIOffset    使用时都不需要加
Make，  如UIEdgeInsets(1, 2, 3, 4)
```

Const常量

不能直接使用, 如 UIKIT_EXTERN NSString *const UIApplicationDidBecomeActiveNotification; 需要你先在OC代码中打印其内容, 然后在lua中使用实际值。

Enum使用

已内置大部分系统的enum。此文件 里定义的都可以直接用，例如:

```
UIButton:buttonWithType(UIButtonTypeCustom)

UITableView:initWithFrame_style(CGRect(0, 0, 100, 100), UITableViewStylePlain);
```

自己写的enum则要么预先定义，要么写实际值。

```
typedef enum : NSUInteger {

    MyEnumValueA=1,

    MyEnumValueB=2,

    MyEnumValueC=3,

} MyEnum;
```

Lua使用:

```
local MyEnumValueA=1,

self:xxx(MyEnumValueA)

self:xxx(2);--MyEnumValueB
```

宏定义:

宏定义define需要还原成实际的代码并翻译成Lua，例如:

```
#define XXX 1

#define YYY 2

-- lua

local XXX = 1

self:xxx(XXX) --或者self:xxx(1)

local YYY = 2

self:yyy(YYY) --或者self:yyy(2)
```

Lua对象转换为OC对象，使用`toobjc`，例如：

```
local testString = "Hello lua!"

local bigFont = UIFont:boldSystemFontOfSize(30)

local size = toobjc(testString):sizeWithFont(bigFont)

puts(size)
```

Lua或者OC对象转换为字符串，使用`tostring`，例如：

```
local num = 123

local str = tostring(num)

打印日志

print("hello world")

print(tostring(xxx))
```

调用类方法

```
类方法

[UT commitEvent:123 arg1:@"456" arg2:@"789"];

--Lua调用类方法

UT:commitEvent_arg1_arg2("123", "456", "789");
```

```
--Lua hook类方法(与实例方法一样)

interface{"UT"}

function commitEvent_arg1_arg2(self, event, arg1, arg2)

--xxx

self:ORIGcommitEvent_arg1_arg2(event, arg1, arg2)

end
```

字符串使用

Lua中返回的字符串均为Lua string, 不能直接调用NSString的方法， 否则会报nil value错误, 不要用stringWithFormat。 例如:

```
local x = str:length();

错误:(string) '[string "interface{"AViewController"}..."]:11: attempt to call
method 'length' (a nil value)'

 if not tempDeviceID:isEqualToString(self:deviceID()) then

end

错误: (string) '[string "interface{"AViewController"}..."]:16: attempt to call
method 'isEqualToString' (a nil value)'
```

1. 正确使用方式一：用Lua的字符串操作。

```
local x = string.len(str);

 if not tempDeviceID == self:deviceID() then

end

string1 = "Lua"

string2 = "Tutorial"

number1 = 10

number2 = 20

--拼接
```

```
local s = string1 .. " abc " .. string2

-- 基本字符串格式化

print(string.format("基本格式化 %s %s",string1,string2))

-- 日期格式化

date = 2; month = 1; year = 2014

print(string.format("日期格式化 %02d/%02d/%03d", date, month, year))

-- 十进制格式化

print(string.format("%.4f",1/3))
```

2. 正确使用方式二：将Lua字符串转换成OC NSString对象，然后再调用NSString的OC方法。

```
local x = toobjc(str):length();

 if not toobjc(tempDeviceID):isEqualToString(self:deviceID()) then

end
```

其它使用

NSString *x = NSStringFromClass(“TestVC”)用local x = self:class():description() 代替。

NSArray / NSDictonary在Lua中的使用

NSArray、NSDictionary在Lua中并不能像普通对象一样使用，因为他们会被wax转换成table，所以取而代之的是在Lua中使用table来实现这两者的功能。

在Lua脚本里不能对Array、Dictionary进行初始化，如下用法都会造成crash。

```
NSMutableArray:init()

NSMutableArray:arrayWithObjects("1","2")

NSMutableDictionary:init()

NSMutableDictionary:dictionaryWithObjectsAndKeys("v1","k1")
```

统一使用Lua的table，当你用过之后会发现他更灵活方便。

Lua table

table类型实现了关联数组，不仅可以通过整数来索引它，还可以使用字符串或其它类型的值（除了nil）来索引它。此外，table没有固定的大小，可以动态得添加任意数量的元素到一个table中。table可以简单实现OC的Dictionary和Array功能。

`实现Array的功能`

初始化：

```
local array = {"a", "b", "c"}
```

下标访问（注意Lua的数组下标都是从1开始！）：

添加元素:

```
table.insert(array, "d")--末尾添加

table.insert(array, 2, "e")--插入到第二个位置
```

删除元素：

```
table.remove(array)--删除末尾元素

table.remove(array, 2)--删除第二个元素
```

遍历:

```
for i = 1, #array do

    print(array[i])

end

for i, v in pairs(array) do --i为下标

    print(v)

end
```

排序：

```
table.sort(array)

function cmp( a, b)

    return a > b

end
table.sort(array, cmp)--指定排序方法
```

实现Dictionary的功能

初始化（注意key不需要“”！）

```
local dict = {}

local dict = {k1="v1", k2="v2"}

--下标访问（注意key需要“”！）

print(dict["k1"])

print(dict.k1)
```

添加元素:

```
dict["k3"] = "v3"

dict.k4="v4"

local x = "xx"

dict[x] = "yy"
```

删除元素：

```
dict["k1"] = nil

dict.k2 = nil
```

遍历:

```
for k, v in pairs(dict) do

    print(k .. " : " .. v)

end
```

取出key排序：

```
local keyArray = {}

for k in pairs(dict) do

    table.insert(keyArray, k)

end

table.sort(keyArray)

function cmp( a, b)

    return a > b

end

table.sort(keyArray, cmp)
```

`table互转Array、Dictionary`

假设有三个属性

```
@interface TestTableVC ()

@property (nonatomic, strong) NSMutableDictionary *muteDict;

@property (nonatomic, strong) NSMutableArray *muteArray;

@property (nonatomic, strong) NSMutableDictionary *viewDict;

@end
```

获取Array、Dictionary对象调用其方法会直接crash

```
self:muteDict():objectForKey("k1")

self:muteArray():objectAtIndex(1)

attempt to call method 'objectForKey' (a nil value)
```

原因是获取到的对象已经转换成table，而table没有这些方法。 要想使用OC的方法可以先通过toobjc将table对象转换成OC对象，然后再调用方法。

```
toobjc(self:muteDict()):objectForKey("k1")

toobjc(self:muteArray()):objectAtIndex(1)
```

`改变Array、Dictionary属性的值`

转换成OC对象后就可以改变属性的值吗？比如:

```
toobjc(self:muteDict()):setObject_forKey("v11", "k1")

print(self:muteDict()["k1"])
```

“k1”对应的值还是“v1”，因为调用self:muteDict()时会得到一个临时的table变量，即使转换成Dictionary再setObject_forKey也是改变的临时变量的值，这种情况先将table改变值，再set回去，会自动将table再转换成Dictionary。

```
local muteDict = self:muteDict()

muteDict["k1"] = "v11"

self:setMuteDict(muteDict)

print(self:muteDict()["k1"])
```

注意元素为0的情况，看此例子：

```
local updateDict = AliNavigatorUtil:updateDictForUsertrack(urlActionRequest);

viewController:setUpdateArgs(updateDict);

viewController:setUtArgs(updateDict);
```

当返回的updateDict元素为0时，viewController:setUpdateArgs将会得到NSArray而不是NSDictionary，后续当做字典使用将会造成crash。 原因是判断将Lua table转换为NSDictionary的条件Lua table的元素非空且第一个key不是数字。所以当Lua table里的元素为空时，就当做了数组NSArray。解决办法为当字典为空时给他加一个没意义的key-value。

```
local updateDict = AliNavigatorUtil:updateDictForUsertrack(urlActionRequest);

local count = 0

for _ in pairs(trackParam) do count = count + 1 end

if updateDict ~= nil and count == 0 then

   updateDict = {k="v"};--无key,value时会被判定位数组

  end

viewController:setUpdateArgs(updateDict);

viewController:setUtArgs(updateDict);
```

例子2

```
@interface TestDynamicFramework : BaseViewController

{

    NSDictionary *homeDate;

}

--Lua

local homeDate_lua = self:getIvarObject("homeDate");

local enterBtnTitle = toobjc(homeDate_lua):objectForKey("enterBtnTitle");
```

原因与上面提到的一致，空元素为字典被转成Lua对象，再toobjc变成了NSArray导致objectForKey不存在。 解决办法为使用Lua获取key-value的方式，如：

```
local enterBtnTitle = homeDate_lua["enterBtnTitle"];
```

C函数调用

使用C函数前，先打开绑定C函数开关，在Lua文件的最前面调用如下代码：

```
gsSetConfig({bind_ocf=true})
```

UIKitFunction支持的C函数API表

```
NSData * UIImageJPEGRepresentation ( UIImage *image, CGFloat compressionQuality );

NSData * UIImagePNGRepresentation ( UIImage *image );

void UIImageWriteToSavedPhotosAlbum ( UIImage *image, id completionTarget, SEL
completionSelector, void *contextInfo );

void UISaveVideoAtPathToSavedPhotosAlbum ( NSString *videoPath, id
completionTarget, SEL completionSelector, void *contextInfo );

BOOL UIVideoAtPathIsCompatibleWithSavedPhotosAlbum ( NSString *videoPath );

CGContextRef UIGraphicsGetCurrentContext ( void );

void UIGraphicsPushContext ( CGContextRef context );

void UIGraphicsPopContext ( void );

void UIGraphicsBeginImageContext ( CGSize size );

void UIGraphicsBeginImageContextWithOptions ( CGSize size, BOOL opaque, CGFloat
scale );

UIImage * UIGraphicsGetImageFromCurrentImageContext ( void );

void UIGraphicsEndImageContext ( void );

void UIRectClip ( CGRect rect );

void UIRectFill ( CGRect rect );

void UIRectFillUsingBlendMode ( CGRect rect, CGBlendMode blendMode );

void UIRectFrame ( CGRect rect );

void UIRectFrameUsingBlendMode ( CGRect rect, CGBlendMode blendMode );

void UIGraphicsBeginPDFContextToData ( NSMutableData *data, CGRect bounds,
NSDictionary *documentInfo );
```

```
BOOL UIGraphicsBeginPDFContextToFile ( NSString *path, CGRect bounds, NSDictionary
*documentInfo );

void UIGraphicsEndPDFContext ( void );

void UIGraphicsBeginPDFPage ( void );

void UIGraphicsBeginPDFPageWithInfo ( CGRect bounds, NSDictionary *pageInfo );

CGRect UIGraphicsGetPDFContextBounds ( void );

void UIGraphicsAddPDFContextDestinationAtPoint ( NSString *name, CGPoint point );

void UIGraphicsSetPDFContextDestinationForRect ( NSString *name, CGRect rect );

void UIGraphicsSetPDFContextURLForRect ( NSURL *url, CGRect rect );

CGAffineTransform CGAffineTransformFromString ( NSString *string );

CGPoint CGPointFromString ( NSString *string );

CGRect CGRectFromString ( NSString *string );

CGSize CGSizeFromString ( NSString *string );

NSString * NSStringFromCGAffineTransform ( CGAffineTransform transform );

NSString * NSStringFromCGPoint ( CGPoint point );

NSString * NSStringFromCGRect ( CGRect rect );

NSString * NSStringFromCGSize ( CGSize size );

NSString * NSStringFromUIEdgeInsets ( UIEdgeInsets insets );

NSString * NSStringFromUIOffset ( UIOffset offset );

UIEdgeInsets UIEdgeInsetsFromString ( NSString *string );

UIOffset UIOffsetFromString ( NSString *string );

BOOL UIEdgeInsetsEqualToEdgeInsets ( UIEdgeInsets insets1, UIEdgeInsets insets2 );

CGRect UIEdgeInsetsInsetRect ( CGRect rect, UIEdgeInsets insets );

BOOL UIOffsetEqualToOffset ( UIOffset offset1, UIOffset offset2 );

void UIAccessibilityPostNotification ( UIAccessibilityNotifications notification,
id argument );
```

```
BOOL UIAccessibilityIsVoiceOverRunning ( void );

BOOL UIAccessibilityIsClosedCaptioningEnabled ( void );

void UIAccessibilityRequestGuidedAccessSession ( BOOL enable, void
(^completionHandler)(BOOL didSucceed) );

BOOL UIAccessibilityIsGuidedAccessEnabled ( void );

BOOL UIAccessibilityIsInvertColorsEnabled ( void );

BOOL UIAccessibilityIsMonoAudioEnabled ( void );

void UIAccessibilityZoomFocusChanged ( UIAccessibilityZoomType type, CGRect frame,
UIView *view );

void UIAccessibilityRegisterGestureConflictWithZoom ( void );

CGRect UIAccessibilityConvertFrameToScreenCoordinates ( CGRect rect, UIView *view
);

UIBezierPath * UIAccessibilityConvertPathToScreenCoordinates ( UIBezierPath *path,
UIView *view );

CTTextAlignment NSTextAlignmentToCTTextAlignment ( NSTextAlignment nsTextAlignment
);

NSTextAlignment NSTextAlignmentFromCTTextAlignment ( CTTextAlignment
ctTextAlignment );

UIGuidedAccessRestrictionState UIGuidedAccessRestrictionStateForIdentifier (
NSString *restrictionIdentifier );
```

Lua使用例子:

```
interface{"AutoTestUIKitFunction"}

function autoTestStart(self)

    local res =
self:respondsToSelector("UIImageWriteToSavedPhotosAlbum:didFinishSavingWithError:c
ontextInfo:")

    print("res=", res)

    UIGraphicsBeginImageContext(CGSize(200,400));

    --renderInContext呈现接受者及其子范围到指定的上下文
```

```
 UIApplication:sharedApplication():keyWindow():layer():renderInContext(UIGraphicsG
etCurrentContext());

    --返回一个基于当前图形上下文的图片

    local aImage =UIGraphicsGetImageFromCurrentImageContext();

    --移除栈顶的基于当前位图的图形上下文

    UIGraphicsEndImageContext();

    --以png格式返回指定图片的数据

    local imageData =UIImagePNGRepresentation(aImage);

    print("imageData.length=", imageData.length);

    local image = aImage;

    -- local image = UIImage:imageNamed("mac_screen");

    local data = UIImageJPEGRepresentation(image, 0.8);

    -- data = toobjc(data)

    print("data.length=", data:length());

    data = UIImagePNGRepresentation(image);

    -- data = toobjc(data)

    print("data.length=", data:length());

    local info = toobjc({key="value"});

    UIImageWriteToSavedPhotosAlbum(image, self,
"UIImageWriteToSavedPhotosAlbum:didFinishSavingWithError:contextInfo:", info);

    local point = CGPoint(10, 10);

    print("point=", point)

    local pointString = NSStringFromCGPoint(point);

    print("pointString=", pointString)

    point = CGPointFromString(pointString);
```

```
    print("point=", point)

    local rect = CGRect(10, 10, 100, 100);

    local rectString = NSStringFromCGRect(rect);

    print("rectString=", rectString)

    rect = CGRectFromString(rectString);

    print("rect=", rect)

end

function UIImageWriteToSavedPhotosAlbum_didFinishSavingWithError_contextInfo(self,
image, error, contextInfo)

    print("error=", error, "contextInfo=", contextInfo);

end
```

dispatch支持的C函数API表:

```
void* dispatch_get_main_queue ( void );

void* dispatch_get_global_queue ( long identifier, unsigned long flags );

void* dispatch_queue_create ( const char *label, void* attr );

void* dispatch_get_current_queue ( void );

const char * dispatch_queue_get_label ( void* queue );

void dispatch_main ( void );

void dispatch_async ( void* queue, void* block );

void dispatch_sync ( void* queue, void* block );

void dispatch_after ( unsigned long long when, void* queue, void* block );

void dispatch_apply ( unsigned long long iterations, void* queue, void*);

void dispatch_once ( long *predicate, void* block );

void* dispatch_semaphore_create ( long value );

long dispatch_semaphore_signal ( void* dsema );
```

```
long dispatch_semaphore_wait ( void* dsema, unsigned long long timeout );

unsigned long long dispatch_time ( unsigned long long when, long long delta );
```

Lua使用例子:

```
interface{"AutoTestGCD"}

function autoTestStart( self )

    print("begin AutoTestGCD autoTestStart");

    local main_queue = dispatch_get_main_queue();

    print(string.format("main_queue=%s", dispatch_queue_get_label(main_queue)));

    local global_queue =
 dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);

    print(string.format("global_queue=%s",
dispatch_queue_get_label(global_queue)));

    local  create_queue = dispatch_queue_create("com.taobao.test",
DISPATCH_QUEUE_SERIAL);

    print(string.format("create_queue=%s",
dispatch_queue_get_label(create_queue)));

    local  current_queue = dispatch_get_current_queue();

    print(string.format("current_queue=%s",
dispatch_queue_get_label(current_queue)));

    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0),

        toblock(

            function( )

                print(string.format("dispatch_async"));

            end)

    );

    dispatch_async(dispatch_get_main_queue(),
```

```
        toblock(
            function( )
                print(string.format("dispatch_async"));
            end)
    );
    dispatch_sync(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0),
        toblock(
            function( )
                print(string.format("dispatch_sync"));
            end)
    );
    local va = UIView:init()
print("va=", va)
local vb = UIViewController:init()
print("vb=", vb)
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (5 * NSEC_PER_SEC)),
dispatch_get_main_queue(),
        toblock(
            function( )
                print(string.format("dispatch_after"));
            end)
    );
    print("start dispatch_apply")
    local semaphore = dispatch_semaphore_create(0);
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (5 * NSEC_PER_SEC)),
dispatch_queue_create("com.taobao.test", DISPATCH_QUEUE_SERIAL),
```

```
        toblock(
            function()
                print(string.format("dispatch_semaphore_t
dispatch_semaphore_signal"));
                dispatch_semaphore_signal(semaphore);
                print("semaphore=", tostring(semaphore))
            end)
    );
    dispatch_semaphore_wait(semaphore, DISPATCH_TIME_FOREVER);
    local time = dispatch_time(DISPATCH_TIME_NOW, 10);
    print(string.format("DISPATCH_TIME_NOW=%d time=%d", DISPATCH_TIME_NOW, time));
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (5 * NSEC_PER_SEC)),
dispatch_get_main_queue(),
        toblock(
            function( )
                print("lua dispatch_after")
            end)
    );
    --test download
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0),
        toblock(
            function( )
                print("dispatch_async start download");
                local request =
NSURLRequest:requestWithURL(NSURL:URLWithString("https://www.aliyun.com"));
                local data =
NSURLConnection:sendSynchronousRequest_returningResponse_error(request, nil, nil);
```

```
                print("dispatch_async end download");
                dispatch_async(dispatch_get_main_queue(),
                    toblock(
                        function( )
                            print("dispatch_async back to main");
                            print(string.format("data.length=%d", data:length()));
                        end)
                )
            end)
    );
    print("end AutoTestGCD autoTestStart");
end
```

runtime支持的C函数API列表

```
const char * class_getName ( void* cls );
void* class_getSuperclass ( void* cls );
void* class_setSuperclass ( void* cls, void* newSuper );
bool class_isMetaClass ( void* cls );
long long class_getInstanceSize ( void* cls );
void* class_getInstanceVariable ( void* cls, const char *name );
void* class_getClassVariable ( void* cls, const char *name );
void objc_setAssociatedObject ( void* object, void *key, void* value, long policy
);
void* objc_getAssociatedObject ( void* object, const void *key );
void objc_removeAssociatedObjects ( void* object );
```

```
const char * property_getName ( void* property );
```

Lua使用例子:

```
interface{"AutoTestRuntime"}

function autoTestStart(self)

    print("begin AutoTestRuntime autoTestStart")

    local res = objc_getAssociatedObject(self, self:getKey1())

    print('res=', res)

    objc_setAssociatedObject(self, self:getKey1(), "lua_value1", 1)

    local res = objc_getAssociatedObject(self, self:getKey1())

    print('res=', res)

    objc_removeAssociatedObjects(self)

    local res = objc_getAssociatedObject(self, self:getKey1())

    print('res=', res)

    print('class_getName=', class_getName(self:class()))

    print('class_getSuperclass=', class_getSuperclass(self:class()))

    print('class_isMetaClass=', class_isMetaClass(self:class()))

    print('class_getInstanceSize=', class_getInstanceSize(self:class()))

    print('class_getInstanceVariable=', class_getInstanceVariable(self:class(),
"view"))

    print('class_getClassVariable=', class_getClassVariable(self:class(), "view"))

    print("end AutoTe   stRuntime autoTestStart")

end
```

Block使用

`将 Lua 方法转化为 OC block`

如果我们知道了 block 的参数类型，我们就可以构造 block 了。 在 armv7/i386 中，栈里的参数地址比较简单， `char, int, BOOL, pointer, id` 可以被当作 `int` 处理， 我们可以使用包含多参数的函数来构造 block，比如：

```
 - (LongLong (^)(int p, ...))luaBlockReturnLongLongWithFirstIntParamTypeEncoding:
(NSString *)paramsTypeEncoding {
    return [[^LongLong(int q, ...){
        LUA_BLOCK_CALL_ARM32_RETURN_BUFFER(paramsTypeEncoding, LongLong, q);
    } copy] autorelease];
 }
```

在 arm64/x86_64 设备上，更复杂一点， `char, int, BOOL, pointer, id, long` 可以当作 longlong 处理, `float, double` 可以当作 double 处理, 因此我们可以构造一个包含 510 个函数的 block 池来支持最多7个参数的 block 类型。

- 可以使用`toblock(luaFunction, typeArray)` 将 Lua 函数转换为 OC block。除block返回值为空，或者无参数的情况， typeArray 中的第一项必须与返回值类型一致。
- 返回值为void的block，返回 void block。

```
UIView:animateWithDuration_animations_completion(1,
  toblock(
       function()
            label:setCenter(CGPoint(300, 300))
       end
   ),
   toblock(
            function(finished)
                print('lua animations completion ' .. tostring(finished))
            end
   ,{"void", "BOOL"})-- return void
)
```

- `- (void)testReturnIdWithFirstIdBlock:(id(^)(id aFirstId, BOOL aBOOL, int aInt, NSInteger aInteger, float aFloat, CGFloat aCGFloat, id aId))block`

```
local res = self:testReturnIdWithFirstIdBlock(
    toblock(
        function(aFirstId, aBOOL, aInt, aInteger, aFloat, aCGFloat, aId)
            print("lua aFirstInt")
            return "123"
        end
        , {"id", "id", "BOOL", "int", "NSInteger" , "float" , "CGFloat" , "id" })
    )
```

`避免循环引用`

`使用 block 一次`

```
 --OC block void (^)(UIViewController * sourceViewController, UIWebView *
webView);
 local weakSelf = self--temp self
 self:setMyblock(
 toblock(
     function(sourceViewController, webView)
         -- print("lua sourceViewController")
         print(string.format("lua sourceViewController=%s webView=%s
self.price=%s", tostring(sourceViewController), tostring(webView),
tostring(weakSelf:price())))
         weakSelf = nil--make it empty
     end
     , {"void", "id", "id"})
 )
```

多次使用 block

```
 local weak = {}
 b = {__mode = "v"}
 setmetatable(weak, b)
 weak.self = self

 toblock(
    function(sourceViewController, webView)
         -- print("lua sourceViewController")
         print(string.format("lua sourceViewController=%s webView=%s
self.price=%s", tostring(sourceViewController), tostring(webView),
tostring(weakSelf:price())))
         weak.self:xxx()
    end
    ), {"void", "id", "id"})
 )
```

Lua 中调用 OC block

```
 --OC block : void (^)())block
 block()

 --或者像下面这样调用
 gsCallBlockWithParamsType(block, {"void"});


 --OC block : void (^)(NSString * code, NSDictionary * responseData))response
 block("abcd", {k1="v1", k2="v2"})

 --或者像下面这样调用
 gsCallBlockWithParamsType(block, {"void","id", "id"}, str, {k1="v1", k2="v2"})
```

```
 --OC block ： CGFloat(^)(CGFloat aFirstCGFloat, BOOL aBOOL, int aInt, NSInteger
aInteger, float aFloat, CGFloat aCGFloat, id aId))block

 local res = block(TEST_VALUE_CGFLOAT, TEST_VALUE_BOOL,
TEST_VALUE_INTEGER,TEST_VALUE_CGFLOAT)

 --或者像下面这样调用
 local res = gsCallBlockWithParamsType(block, {"CGFloat","CGFloat", "BOOL",
"NSInteger", "CGFloat"}, TEST_VALUE_CGFLOAT, TEST_VALUE_BOOL,
TEST_VALUE_INTEGER,TEST_VALUE_CGFLOAT)
```

Q：什么时候需要使用`gsCallBlockWithParamsType`?
A：当你想执行一个由`toblock`构造的 block 时。

```
 local blockGotBytoblock = toblock(
             function(code, responseData)
                 print("lua code=" .. code .. " responseData=" ..
tostring(responseData))
             end
         , {"void", "NSString *", "NSDictionary *"})

 gsCallBlockWithParamsType(blockGotBytoblock, {"void", "NSString *", "NSDictionary
*"},
         "abc", {k1="v1", k2="v2"})
```