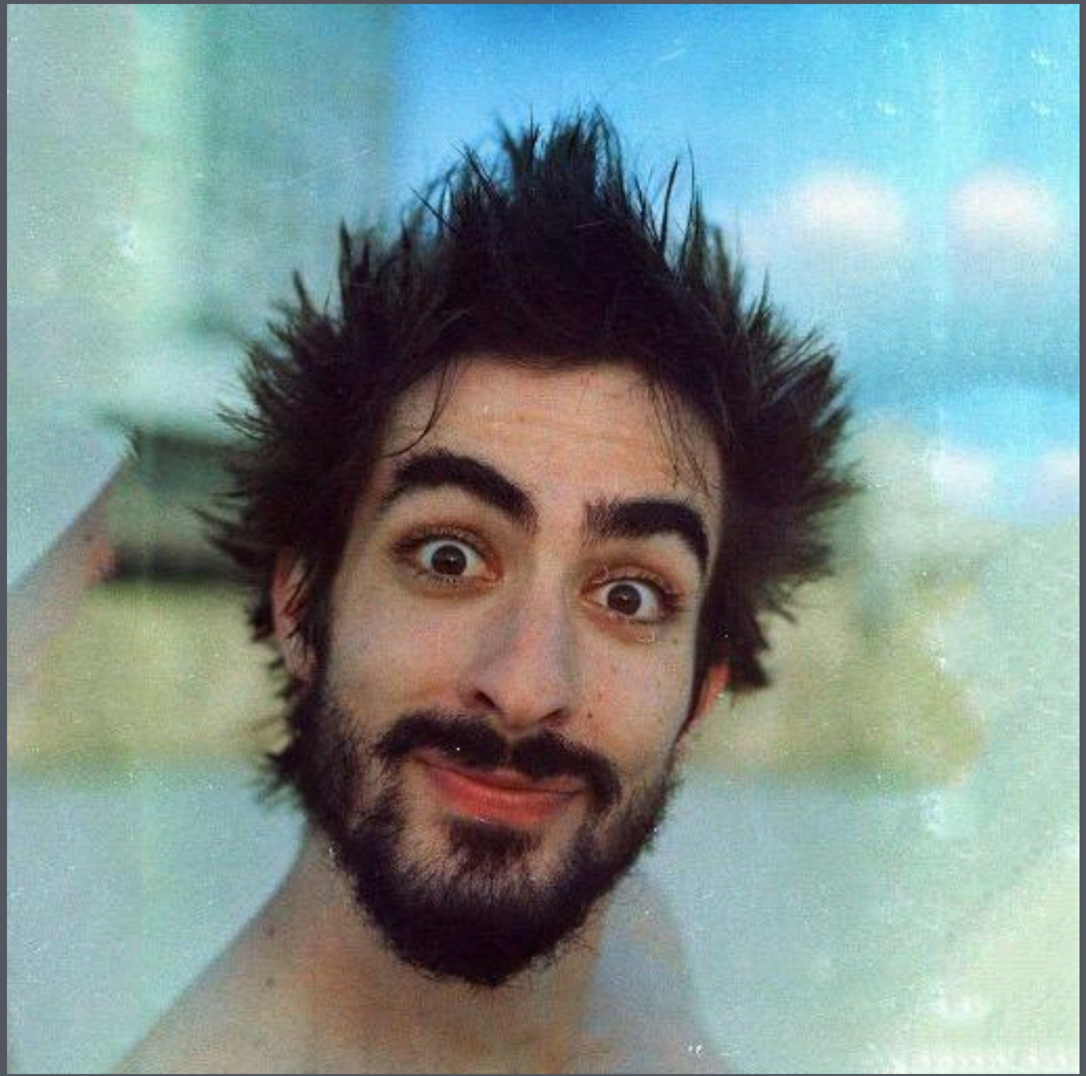
NSFWObjective-C

/me



Wondermall

/me

```
if ([me isKindOfClass:[MetallurgicalEngineer class]]) {
    // I know how to melt metal
    if ([me isKindOfClass:[Designer class]]) {
    // Can tell difference between Arial and San Francisco
        if ([me isKindOfClass:[FlashDeveloper]]) {
            // "Skip Flash Intro"
            if ([me isKindOfClass:[ObjectiveCDeveloper]]) {
                // ❤️
                if let me = self as? SwiftDeveloper {
                    // I'm here
```

NSFWObjective-C

NSFWObjective-C

```
#import <objc/runtime.h>
```

Creating classes at runtime

What is it good for?[1]

- Encapsulate functionality.
- Multiple inheritance.
- Build the class from the blocks.
- Temporary functionality to existent instances using `object_setClass`.

[1] War

Creating classes at runtime

Demo

Creating classes at runtime

Next step: assembling classes.

```
[factory buildClassNamed:@"MyHero" usignBlock:^(ClassBuilder *myHero){
    [myHero copyMethod:@selector(flight)
             fromClass:[Superman class]];

    [myHero copyMethod:@selector(voice)
             fromClass:[Batman class]];

    [myHero copyMethod:@selector(retainCount)
             fromClass:[Aquaman class]];

}];
```

Creating classes at runtime

Slightly more useful

```
[factory buildClassNamed:@"CheckboxButton" usignBlock:^(ClassBuilder *button){
    [button copyMethod:@selector(touchUpInside)
             fromClass:[CheckboxBehaviour class]];

    [button copyMethod:@selector(touchDownInside)
             fromClass:[SoundLibrary click]];

    [button copyMethod:@selector(appearance)
             fromClass:[UITheme checkboxAppearance]];
}];
```

KVO

KVO

Demo

KVO

Preparing dynamic subclass

```
Class originalCls = object_getClass(target);
NSString *clsName = [NSString stringWithFormat:@"Xray_%@", originalCls];
Class cls = objc_allocateClassPair(originalCls, clsName.UTF8String, 0);
class_addMethod(cls, @selector(class), imp_implementationWithBlock(^(id self){
    return originalCls;
}), "#16@0:8");
objc_registerClassPair(cls);
object_setClass(target, cls);
```

KVO

```
SEL setterSEL = [self _setterForKey:key];
IMP originalIMP = [self _originalImpForSelector:setterSEL];
class_addMethod(cls, setterSEL, ^(id self, id value){
    originalIMP(self, setterSEL, newValue);
    handle(self, key, newValue);
}, setterSignature);
```

KVO

Where to take it from here:

1. Match KVO's will / did change observation & old and new values.
2. `-(BOOL)shouldSet<Key>`... observer: whether new value is valid.
3. `-(void)transformValue:forKey:` allows to modify getter on the fly
4. Decrypt the value of property upon access by certain classes.

KVO

Summary

- Create you private subclass.
- Swizzle setter and getter, calling original implementation along with `handler`.
- Change instance class to your private subclass.
- Restore when after last `handler` removed.
- For a proper thread-safe KVO, FBKVOController.

Toll-Free Bridging

Toll-Free Bridging

- Bridge between `CoreFoundation` and `Foundation`.
- Many C APIs are still not matched with Objective-C ones.
- Cocoa optimization: vending private subclasses of `NSArray` before it was cool.
- Not possible to have your own[3]

[3] https://mikeash.com/pyblog/friday-qa-2010-01-22-toll-free-bridging-internals.html

Implementing Toll-Free Bridging

Demo

Toll-Free Bridging

Counting members

```
CF_EXPORT CFIndex CFBinaryHeapGetCountOfValue(
    CFBinaryHeapRef heap,
    const void *value
);

@interface BinaryHeap (Counting)

- (NSUInteger)countForObject:(id)object;

@end
```

Toll-Free Bridging

Counting members (CoreFoundation)

```
CFIndex CFBinaryHeapGetCountOfValue(CFBinaryHeapRef heap, const void *value) {
    CFComparisonResult(*compare)(const void *, const void *, void *);
    compare = heap->_callbacks.compare;
    CFIndex cnt = 0;
    for (CFIndex idx = 0; idx < CFBinaryHeapGetCount(heap); idx++) {
        const void *item = heap->_buckets[idx]._item;
        if ((value == item) || (heap->compare &&
            (heap->compare(value, item, info) == kCFCompareEqualTo))) {
            cnt++;
        }
    }
    return cnt;
}
```

Toll-Free Bridging

Counting members (CoreFoundation)

```
CFIndex CFBinaryHeapGetCountOfValue(CFBinaryHeapRef heap, const void *value) {
    if (CFGetTypeID(heap) != CFBinaryHeapGetTypeID()) {
        return [(__bridge BinaryHeap *)heap countForObject:(__bridge id)value];
    }
    // ...
    return cnt;
}
```

Toll-Free Bridging

C-functions swizzling

```
origGetCountOfValue = dlsym(RTLD_DEFAULT, "CFBinaryHeapGetCountOfValue");

rebind_symbols((struct rebinding[1]){
    {"CFBinaryHeapGetCountOfValue", replGetCountOfValue}
}, 1);

CFIndex replGetCountOfValue(CFBinaryHeapRef heap, void *value) {
    if (CFGetTypeID(heap) != CFBinaryHeapGetTypeID()) {
        return [(__bridge BinaryHeap *)heap countForObject:(__bridge id)value];
    }
    return origGetCountOfValue(heap, value);
}
```

Toll-Free Bridging

Counting members (Objective-C)

```
- (NSUInteger)countForObject:(id)object {
    NSUInteger count = 0;
    for (NSUInteger i = 0; i < self.count; ++i) {
        if ([self.buckets[i] isEqual:object]) {
            count++;
        }
    }
    return count;
}
```

Toll-Free Bridging

Counting members (Objective-C)

```
- (NSUInteger)countForObject:(id)object {
    if (CFGetTypeID((CFTypeRef)self) == CFBinaryHeapGetTypeID()) {
        return CFBinaryHeapGetCountOfValue((CFBinaryHeapRef)self,
                                           (void *)object);
    }

    // ...
    return count;
}
```

Toll-Free Bridging

Bridging

```
CFBinaryHeapRef cfHeap = ...
[(__bridge BinaryHeap *)cfHeap count]; // bang!

_CFRuntimeBridgeClasses(CFBinaryHeapGetTypeID(),
                        "BinaryHeap");
```

Toll-Free Bridging

`isa`

```
// objc4-646
@interface NSObject <NSObject> {
    Class isa OBJC_ISA_AVAILABILITY;
    // ...
}

// CF-1151.16
typedef struct __CFRuntimeBase {
    uintptr_t _cfisa;
    // ...
} CFRuntimeBase;
```

Toll-Free Bridging

Summary

- Create you class, matching CF-API.
- In each method check if `self` is not a CF-counterpart.
- Swizzle CF-functions[fh], call Objective-C method if 1st argument is not CF.
- Establish the bridging relationship between CF and Objective-C classes.
- One Objective-C counterpart for both mutable and immutable CF structures. `_NSCFArray` is actually a mutable array.

[fh] http://github.com/facebook/fishhook

Creating protocols at runtime

Creating protocol at runtime

Setup

```
const char *protocolName = ProtocolNameForClass(cls).UTF8String;
Protocol *protocol = objc_allocateProtocol(protocolName);
protocol_addProtocol(protocol, @protocol(JSExport));
// ...
```

Creating protocol at runtime

Protocol-hierarchy

```
Protocol *ExportClass(Class cls) {
    Protocol *protocol = objc_allocateProtocol(protocolName);
    // ...
    Class superclass = class_getSuperclass(cls);
    ExportClass(superclass);
    protocol_addProtocol(protocol, ProtocolForClass(superclass));
    // ...
}
```

Creating protocol at runtime

Instance methods

```
unsigned int count;
Method *methods = class_copyMethodList(object_getClass(cls), &count);
for (unsigned int i = 0; i < count; ++i) {
    Method method = methods[i];
    struct objc_method_description *desc = method_getDescription(method);
    const char *name = desc->name;
    const char *types = desc->types;
    protocol_addMethodDescription(protocol, name, types, YES, NO);
}
```

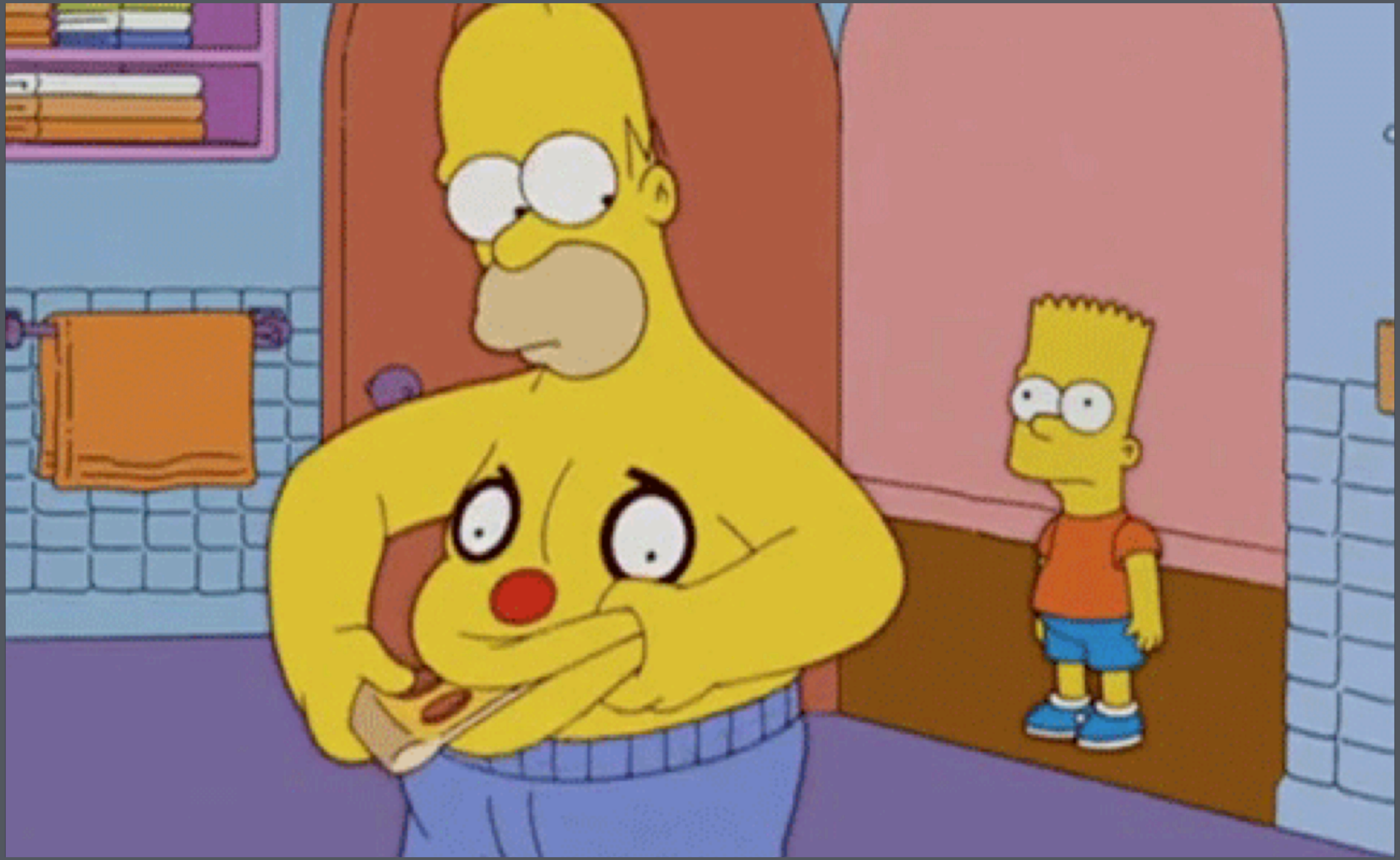
Creating protocol at runtime

Registering protocol

```
objc_registerProtocol(protocol);
class_addProtocol(cls, protocol);
```

Creating protocols at runtime

Eating Xcode's lunch



Creating protocols at runtime

Extended method types

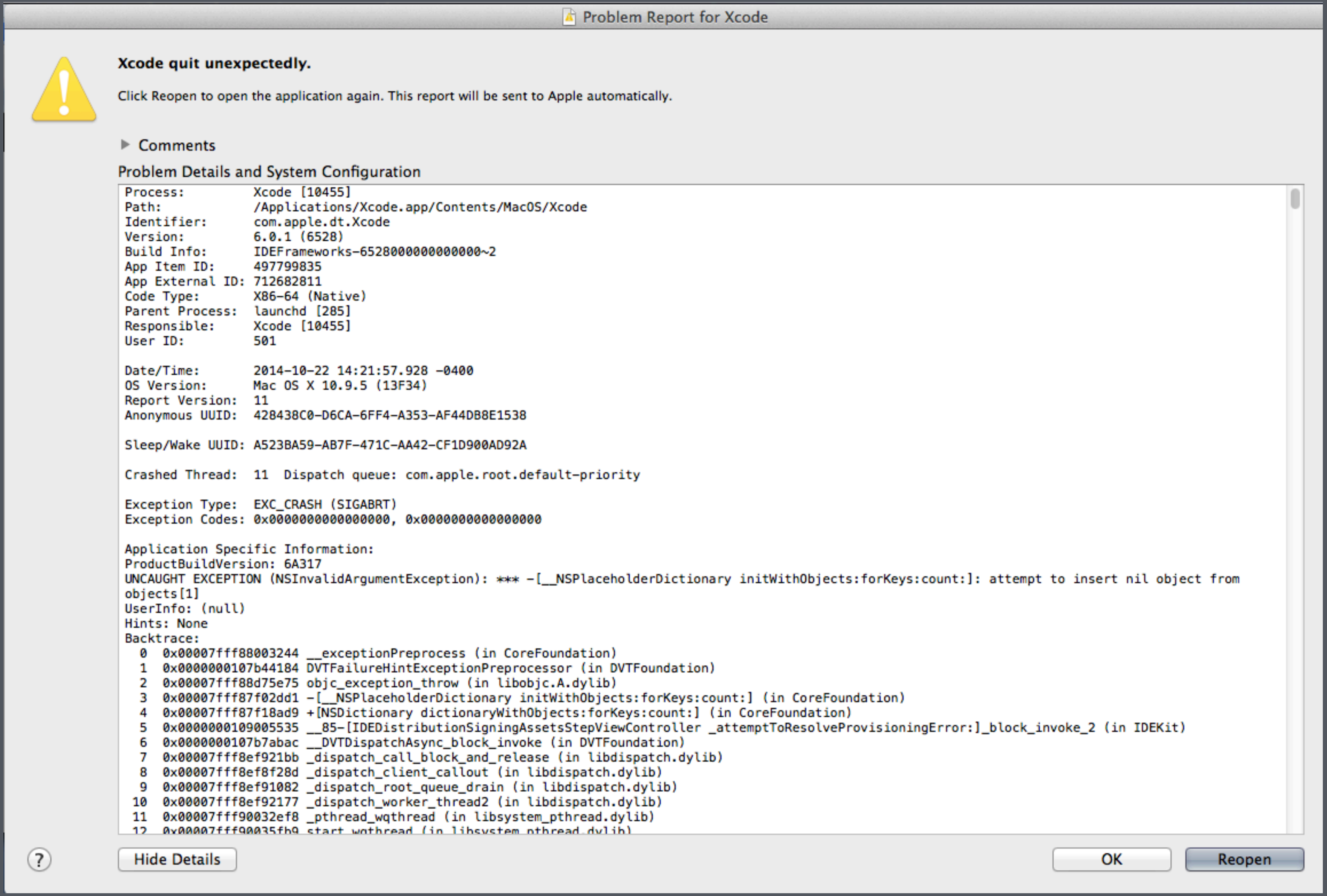
```
unsigned int count;
Method *methods = class_copyMethodList(object_getClass(cls), &count);
for (unsigned int i = 0; i < count; ++i) {
    Method method = methods[i];
    [signatures addObject:@(method_getDescription(method)->types)];
}

protocol_t *myProtocol = (__bridge protocol_t *)protocol;
for (NSUInteger i = 0; i < signatures.count; ++i) {
    const char *signature = signatures[i].UTF8String;
    myProtocol->extendedMethodTypes[i] = signature;
}
```

Creating protocols at runtime

Demo

- How is it different from react-native?
- No way to destroy protocol created at runtime.
- Random code execution without recompiling.
- Downloading a scripted walk-through.
- Working around production bugs.
- JSPatch is another take on random code execution through JS - Objective-C bridge https://github.com/bang590/JSPatch

One more thing…



Xcode quit unexpectedly.

Click Reopen to open the application again. This report will be sent to Apple automatically.

Problem Details and System Configuration

```
Process:         Xcode [10455]
Path:            /Applications/Xcode.app/Contents/MacOS/Xcode
Identifier:      com.apple.dt.Xcode
Version:         6.0.1 (6528)
Build Info:      IDEFrameworks-6528000000000000~2
App Item ID:     497799835
App External ID: 712682811
Code Type:       X86-64 (Native)
Parent Process:  launchd [285]
Responsible:     Xcode [10455]
User ID:         501

Date/Time:       2014-10-22 14:21:57.928 -0400
OS Version:      Mac OS X 10.9.5 (13F34)
Report Version:  11
Anonymous UUID:  428438C0-D6CA-6FF4-A353-AF44DB8E1538

Sleep/Wake UUID: A523BA59-AB7F-471C-AA42-CF1D900AD92A

Crashed Thread:  11  Dispatch queue: com.apple.root.default-priority

Exception Type:  EXC_CRASH (SIGABRT)
Exception Codes: 0x0000000000000000, 0x0000000000000000

Application Specific Information:
ProductBuildVersion: 6A317
UNCAUGHT EXCEPTION (NSInvalidArgumentException): *** -[__NSPlaceholderDictionary initWithObjects:forKeys:count:]: attempt to insert nil object from
objects[1]
UserInfo: (null)
Hints: None
Backtrace:
  0  0x00007fff88003244 __exceptionPreprocess (in CoreFoundation)
  1  0x0000000107b44184 DVTFailureHintExceptionPreprocessor (in DVTFoundation)
  2  0x00007fff88d75e75 objc_exception_throw (in libobjc.A.dylib)
  3  0x00007fff87f02dd1 -[__NSPlaceholderDictionary initWithObjects:forKeys:count:] (in CoreFoundation)
  4  0x00007fff87f18ad9 +[NSDictionary dictionaryWithObjects:forKeys:count:] (in CoreFoundation)
  5  0x0000000109005535 __85-[IDEDistributionSigningAssetsStepViewController _attemptToResolveProvisioningError:]_block_invoke_2 (in IDEKit)
  6  0x0000000107b7abac __DVTDispatchAsync_block_invoke (in DVTFoundation)
  7  0x00007fff8ef921bb _dispatch_call_block_and_release (in libdispatch.dylib)
  8  0x00007fff8ef8f28d _dispatch_client_callout (in libdispatch.dylib)
  9  0x00007fff8ef91082 _dispatch_root_queue_drain (in libdispatch.dylib)
 10  0x00007fff8ef92177 _dispatch_worker_thread2 (in libdispatch.dylib)
 11  0x00007fff90032ef8 _pthread_wqthread (in libsystem_pthread.dylib)
 12  0x00007fff90035fb9 start_wqthread (in libsystem_pthread.dylib)
```

twitter: **@zats**

github: **github.com/zats**

email: **sash@zats.io**

icq: **5559218**