The Dark Side of LLDB

Reverse Engineering Cocoa Applications

Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

Selber Schuld!

Selber Schuld!

Selber Schuld!

Selber Schuld!

```
NSXPCInterface *myCookieInterface =  [NSXPCInterface interfaceWithProtocol: @protocol(FeedMeACookie)];
```

Selber Schuld!

```
0x7fff9581a3bb:  movq   %rsp, %rbp
0x7fff9581a3be:  pushq  %r15
0x7fff9581a3c0:  pushq  %r14
0x7fff9581a3c2:  pushq  %r13
0x7fff9581a3c4:  pushq  %r12
0x7fff9581a3c6:  pushq  %rbx
0x7fff9581a3c7:  pushq  %rax
0x7fff9581a3c8:  movq   %rcx, %r15
0x7fff9581a3cb:  movq   %rdx, %r14
0x7fff9581a3ce:  movq   %rdi, %rbx
0x7fff9581a3d1:  movq   -0x18570d70(%rip), %rsi   ; "_bindingAdaptor"
0x7fff9581a3d8:  movq   -0x1878b61f(%rip), %r13   ; (void *)0x00007fff94b920c0: objc_msgSend
0x7fff9581a3df:  callq  *%r13
0x7fff9581a3e2:  movq   -0x1856c9d1(%rip), %rsi   ; "objectDidTriggerAction:"
0x7fff9581a3e9:  movq   %rax, %rdi
0x7fff9581a3ec:  movq   %rbx, %rdx
0x7fff9581a3ef:  callq  *%r13
0x7fff9581a3f2:  leaq   -0x18517c79(%rip), %r12   ; NSApp
0x7fff9581a3f9:  movq   (%r12), %rdi
```

Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

“Die Rückgewinnung des Quellcodes oder einer vergleichbaren Beschreibung aus Maschinencode. Z. B. von einem ausführbaren Programm oder einer Programmbibliothek"

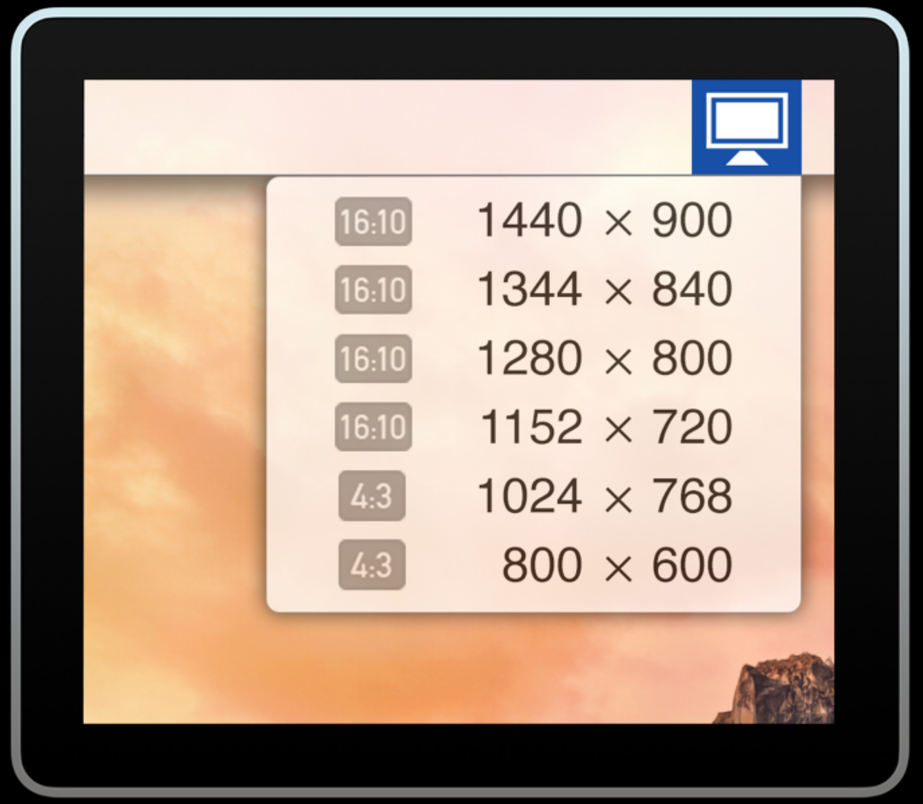
–Wikipedia

Wozu?

- Offenlegen von SPIs oder versteckter Funktionalität, um diese selber zu nutzen
- Erweitern von Software ohne offene Schnittstellen
- Lernen, wie Features z.B. von Konkurrenzprodukten implementiert sind

Wozu?

- Offenlegen von SPIs oder versteckter Funktionalität, um diese selber zu nutzen

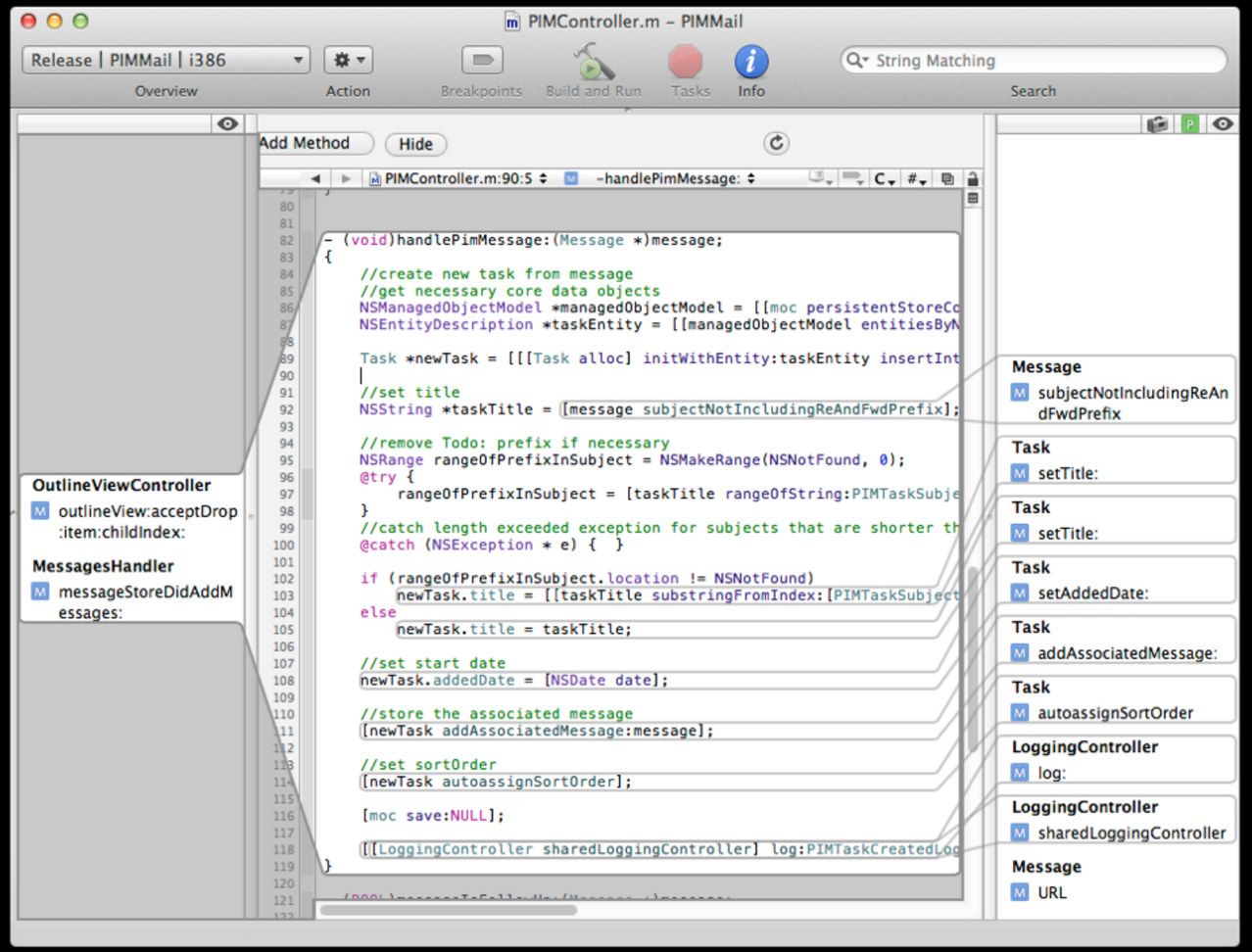


Wozu?

- Verwenden vorhandener Infrastruktur (z.B. zum Lesen von Dateien)

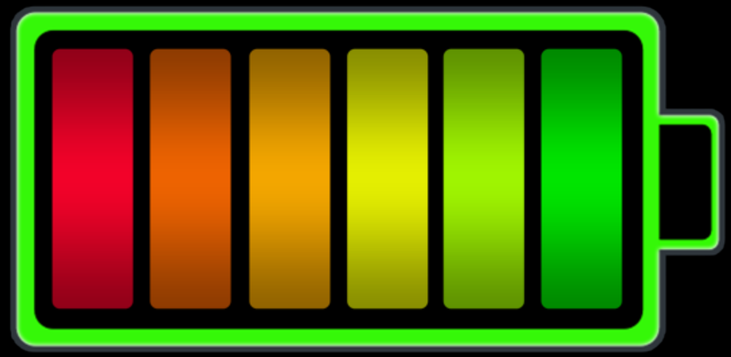
Wozu?

- Erweitern von Software ohne offene Schnittstellen



Wozu?

- Lernen, wie Features z.B. von Konkurrenzprodukten implementiert sind



Battery Health.app

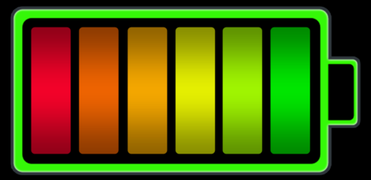
Wozu?

- Lernen, wie Features z.B. von Konkurrenzprodukten implementiert sind

Battery Health.app

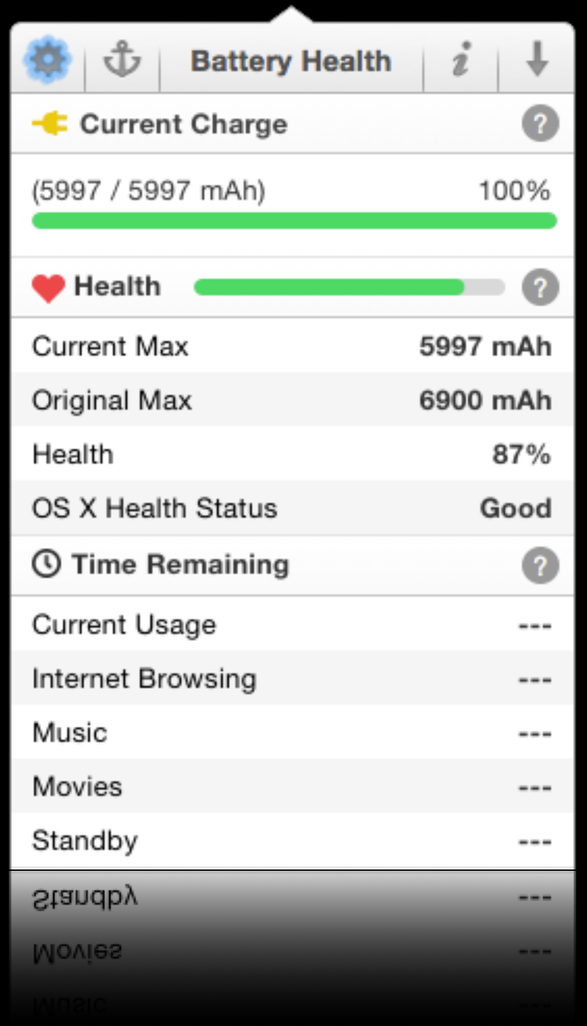
Mitmachen

- *Battery Health.app* aus dem Mac App Store
- Xcode 6.1
- x64 Assembler Cheat Sheet
- Spreadsheet
- class-dump (optional)



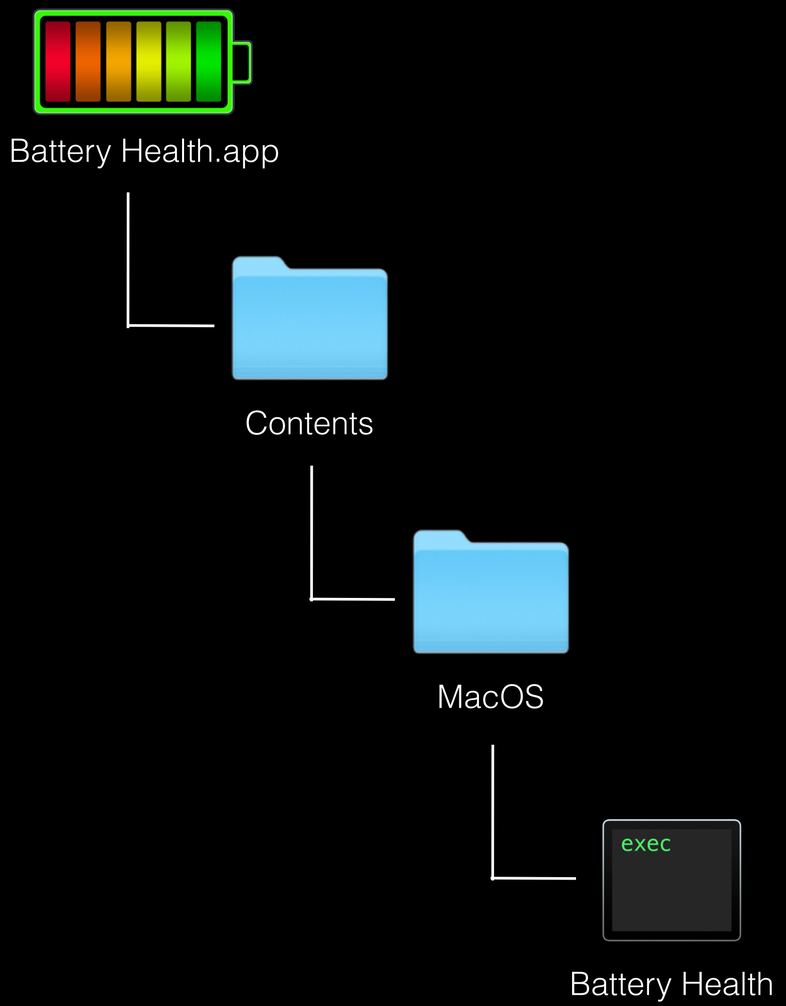
Wie funktioniert das?

- Battery Health hat ein (nützliches?) Feature, welches das Popover Window offen und im Vordergrund hält
- Anker an/aus
- Wir wollen herausfinden, wie das implementiert ist



Was haben wir?

- Laufende App: <Verhalten>
- Defaults Datenbank: <User Preferences>
- Binärcode: <...?...>



Binärdateien untersuchen

- Kommandozeilentools
  - `defaults read com.fiplab.batteryhealth`
  - `cd /Applications/Battery\ Health.app/Contents/MacOS`
  - `strings Battery\ Health > ~/strings.txt`
  - `otool -ov Battery\ Health > ~/otool_ov.txt`
  - bequemer: `class-dump -H -o ~/Headers/ Battery\ Health`

User Defaults

```
defaults read com.fiplab.batteryhealth
{
    FLAnalytics = {
        CleanQuit = 1;
        LastLaunchDate = "2014-10-22 09:08:41 +0000";
        OpenTime = "4.154012024402618";
        TotalLaunches = 19;
    };
    FLCLastUpdate = "2014-10-20 11:23:21 +0000";
    FLULastVersion = 44;
    MacRatingCurrentVersion = 44;
    MacRatingFirstUseDate = "1413804200.874992";
    MacRatingRatedCurrentVersion = 0;
    MacRatingUseCount = 19;
    IsDocked = 1;
    "NSWindow Frame PreferencesWindow" = "1337 779 351 258 0 0 2560 1577 ";
    ShouldKeepAbove = 0;
    UndockedFrame = "692 618 291 712 0 0 2560 1577 ";
}
```

WTF

Komplette Header via class-dump

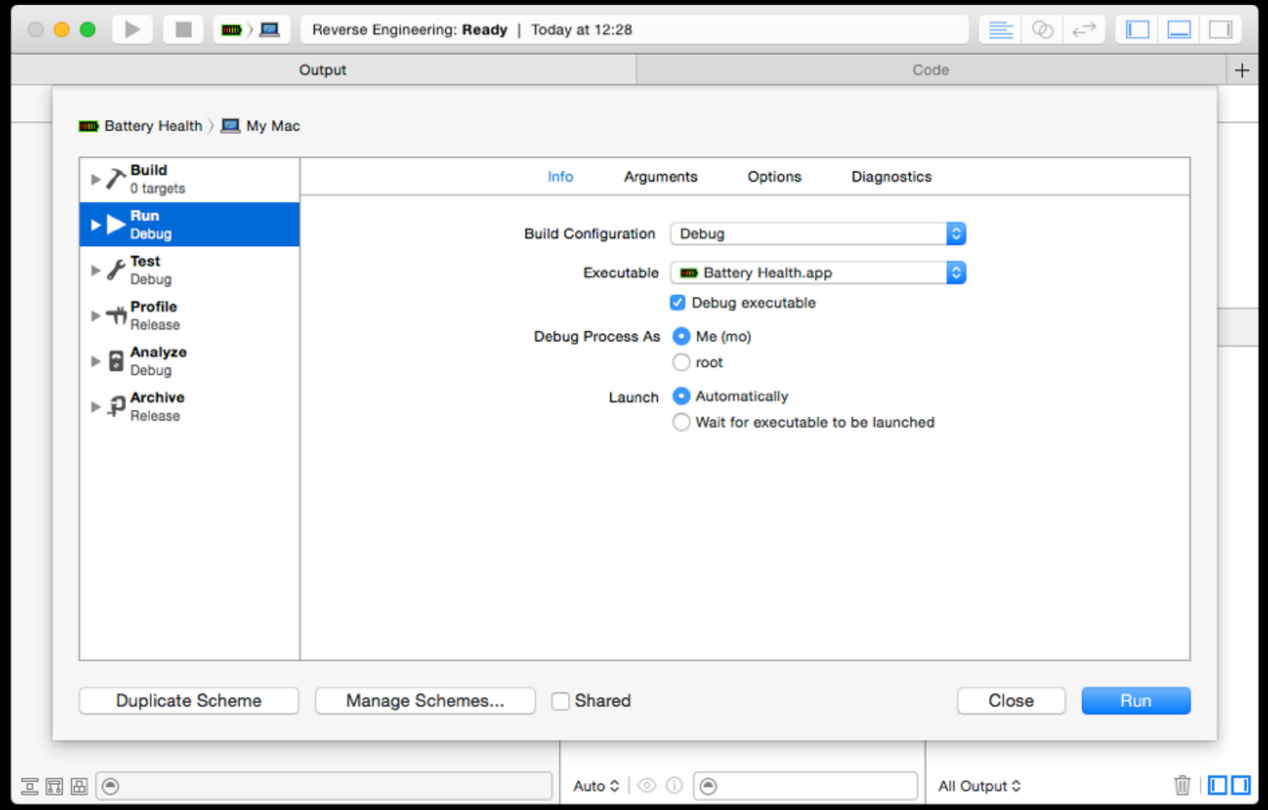
```
BHAppDelegate.h
BHBattery.h
BHBatteryDelegate-Protocol.h
BHChartModule.h
BHCurrentChargeModule.h
BHDetailsModule.h
BHHealthModule.h
BHMainWindowController.h
BHModule.h
BHModuleListViewController.h
BHProgressView.h
BHSectionHeaderView.h
BHTimeRemainingModule.h
BHTipsViewController.h
CDStructures.h
CPTPlotDataSource-Protocol.h
DDNavigationButtonCell.h
DDSwitchView.h
DMGeneralPreferencesViewController.h
FLAboutViewController.h
FLButton.h
FLMavericksButtonCell.h
FLMavericksButtonTheme.h
FLNoScrollView.h
FLPreferences.h
FLPreferencesWindowController.h
FLPushButtonTheme.h
FLTexturedButtonTheme.h
FLTimer.h
FLView.h
FLViewController.h
FLWindowController.h
MacRating.h
NSApplicationDelegate-Protocol.h
NSColor-Utilities.h
NSControlTextEditingDelegate-Protocol.h
NSMenuDelegate-Protocol.h
NSObject-Protocol.h
NSTableViewDataSource-Protocol.h
NSTableViewDelegate-Protocol.h
NSTextFieldDelegate-Protocol.h
NSWindow-ContentView.h
NSWindowDelegate-Protocol.h
STDockableWindowController.h
STGradientLineView.h
STStatusItem.h
STStatusItemDelegate-Protocol.h
STTransparentPanel.h
STWindowBackgroundView.h
__ARCLiteIndexedSubscripting__-Protocol.h
__ARCLiteKeyedSubscripting__-Protocol.h
```

Typen und Adressen?

```
#import <AppKit/NSView.h>

@class NSColor, NSImage, NSViewController;

@interface FLView : NSView
{
    NSViewController *_viewController;    // 152 = 0x98
    NSColor *_backgroundColor;    // 160 = 0xa0
    NSImage *_backgroundImage;    // 168 = 0xa8
    NSColor *_borderColor;    // 176 = 0xb0
    double _borderWidth; // 184 = 0xb8
    double _cornerRadius;    // 192 = 0xc0
}

@property(nonatomic) double cornerRadius; // @synthesize cornerRadius=_cornerRadius;
@property(nonatomic) double borderWidth; // @synthesize borderWidth=_borderWidth;
@property(retain, nonatomic) NSColor *borderColor; // @synthesize borderColor=_borderColor;
@property(retain, nonatomic) NSImage *backgroundImage; // @synthesize backgroundImage=_backgroundImage;
@property(retain, nonatomic) NSColor *backgroundColor; // @synthesize backgroundColor=_backgroundColor;
@property(nonatomic) NSViewController *viewController; // @synthesize viewController=_viewController;
- (void).cxx_destruct;   // IMP=0x000000010000e076
- (void)drawRect:(struct CGRect)arg1;// IMP=0x000000010000db9c
- (void)setShadow:(id)arg1;  // IMP=0x000000010000db55
- (void)setNextResponder:(id)arg1;    // IMP=0x000000010000da5e

@end
```

Xcode Setup

- Leeren Workspace erstellen
- Neues Scheme für Battery Health
  - App unter Run als Executable setzen
- ⌘ + r



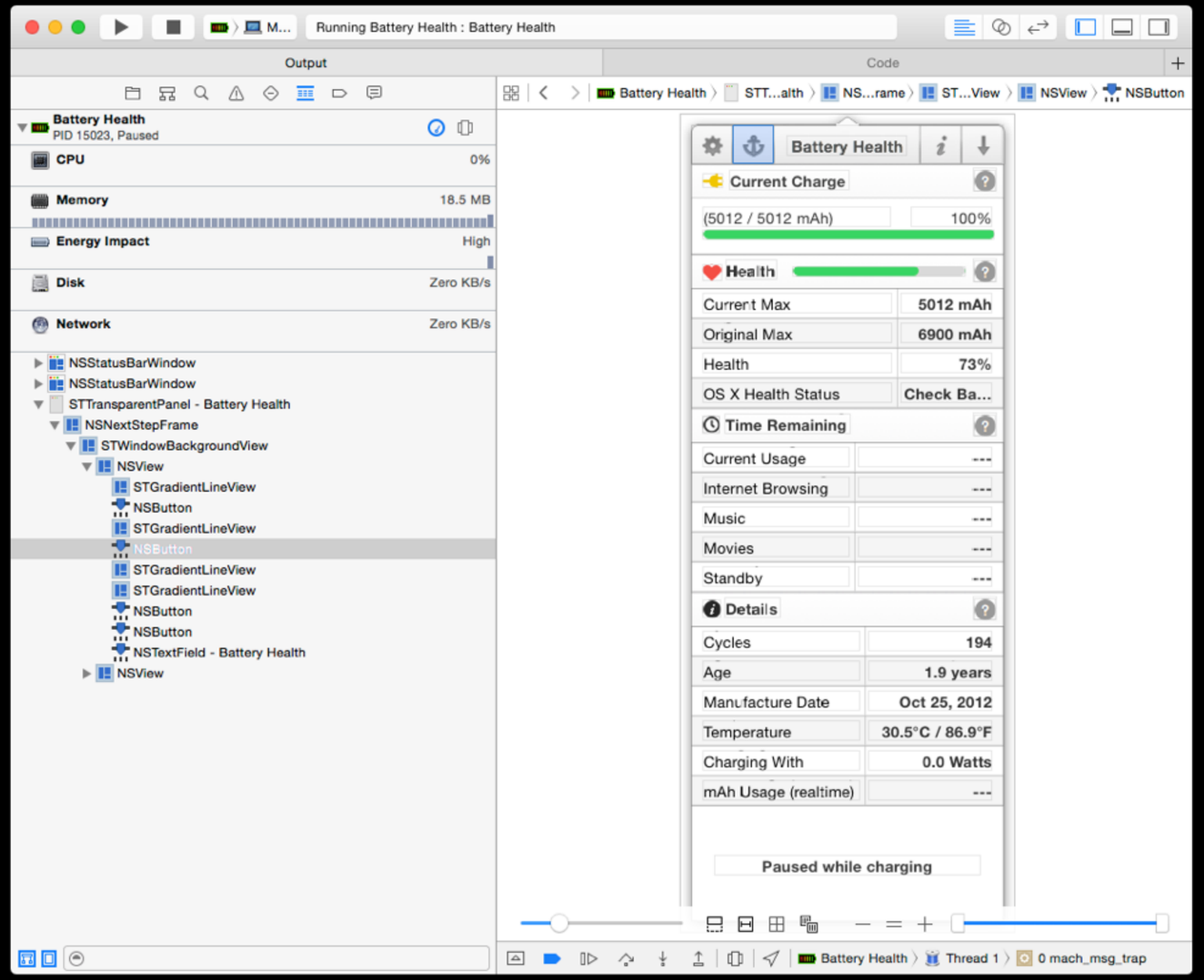
Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

Startpunkt A — View Debugging

- "New in Xcode 6"

Startpunkt A — View Debugging

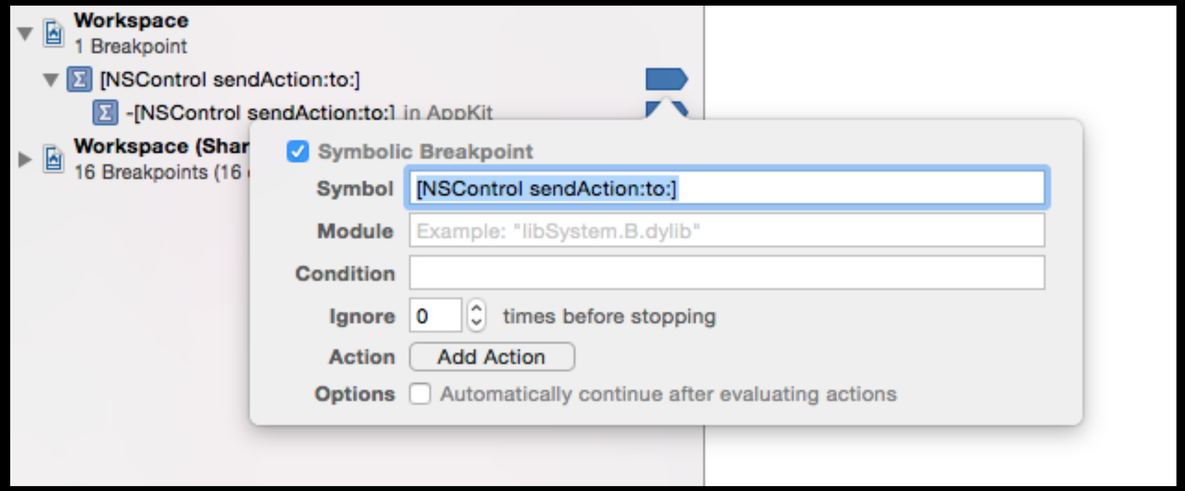
- "New in Xcode 6"



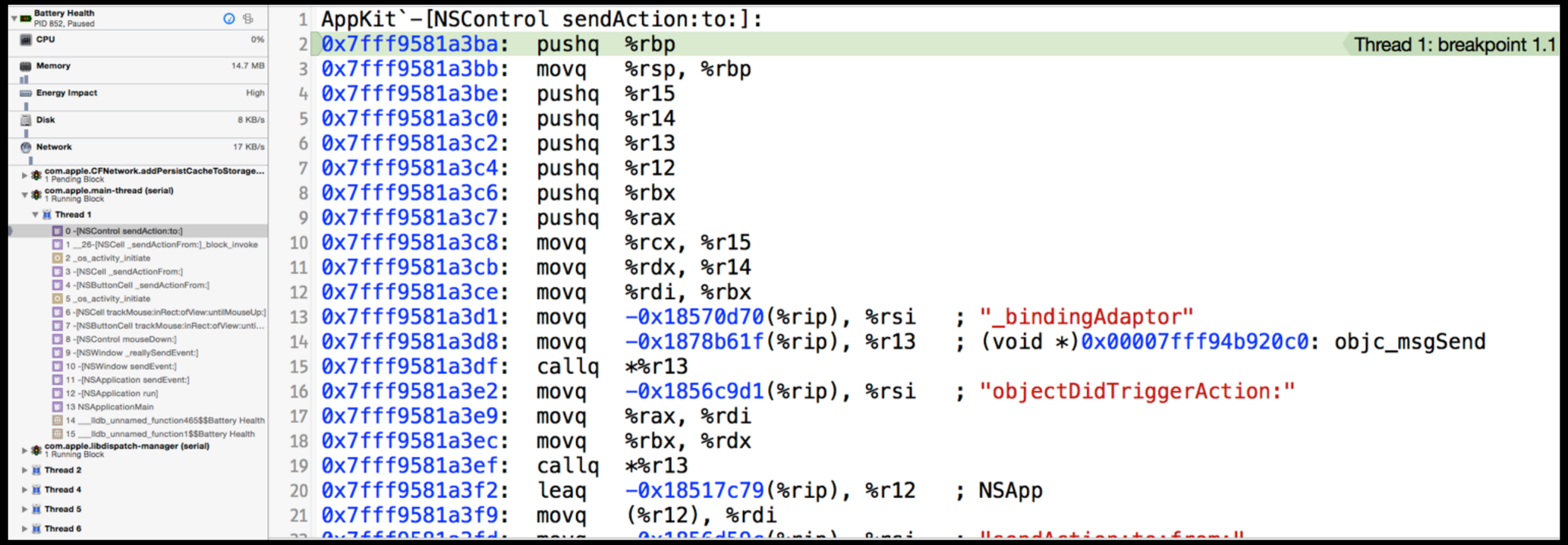
Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

Brechpunkte

- NSButton action wird aufgerufen durch [NSControl sendAction:to:]
- "Create Symbolic Breakpoint"
- (Objective-)C Syntax ohne Variablen



Brechpunkte



Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

Assembly lesen

```
13 0x7fff9581a3d1:  movq   -0x18570d70(%rip), %rsi   ; "_bindingAdaptor"
14 0x7fff9581a3d8:  movq   -0x1878b61f(%rip), %r13   ; (void *)0x00007fff94b920c0: objc_msgSend
15 0x7fff9581a3df:  callq  *%r13
```

Assembly lesen

```
13 0x7fff9581a3d1:  movq   -0x18570d70(%rip), %rsi   ; "_bindingAdaptor"
14 0x7fff9581a3d8:  movq   -0x1878b61f(%rip), %r13   ; (void *)0x00007fff94b920c0: objc_msgSend
15 0x7fff9581a3df:  callq  *%r13
```

- `0x123(a)`: 0x123 + a — Häufig die Position von Konstanten
  - `%rip`: Instruction Pointer, die aktuelle Adresse

Assembly lesen

```
13 0x7fff9581a3d1:  movq   -0x18570d70(%rip), %rsi   ; "_bindingAdaptor"
14 0x7fff9581a3d8:  movq   -0x1878b61f(%rip), %r13   ; (void *)0x00007fff94b920c0: objc_msgSend
15 0x7fff9581a3df:  callq  *%r13
```

- `0x123(a)`: 0x123 + a — Häufig die Position von Konstanten
  - `%rip`: Instruction Pointer, die aktuelle Adresse
- `mov a, b`: kopiere a nach b

Assembly lesen

```
13 0x7fff9581a3d1:  movq   -0x18570d70(%rip), %rsi   ; "_bindingAdaptor"
14 0x7fff9581a3d8:  movq   -0x1878b61f(%rip), %r13   ; (void *)0x00007fff94b920c0: objc_msgSend
15 0x7fff9581a3df:  callq  *%r13
```

- `0x123(a)`: 0x123 + a — Häufig die Position von Konstanten
  - `%rip`: Instruction Pointer, die aktuelle Adresse
- `mov a, b`: kopiere a nach b
- `call a`: springe zu Adresse a und komme bei `ret` nach hier zurück

Assembly lesen

```
13 0x7fff9581a3d1:  movq   -0x18570d70(%rip), %rsi   ; "_bindingAdaptor"
14 0x7fff9581a3d8:  movq   -0x1878b61f(%rip), %r13   ; (void *)0x00007fff94b920c0: objc_msgSend
15 0x7fff9581a3df:  callq  *%r13
```

- `0x123(a)`: 0x123 + a — Häufig die Position von Konstanten
  - `%rip`: Instruction Pointer, die aktuelle Adresse
- `mov a, b`: kopiere a nach b
- `call a`: springe zu Adresse a und komme bei `ret` nach hier zurück
- `jmp a`: springe zu Adresse a

Funktionsaufrufe

1. Wie funktionieren Funktionsaufrufe?
   - x86-64 Function Calling Conventions
   - Wo liegen die Argumente (Register, Stack)
2. Objective C ist auch nur C
   - objc_msgSend(self, selector, arg1, arg2)
   - self -> $rdi
   - selector -> $rsi

```
(lldb) po $rdi
<NSButton: 0x608000140c60>

(lldb) p (SEL)$rsi
(SEL) $1 = "sendAction:to:"
```

Calling Convention Cheat Sheet

Arg	Objective-C Bedeutung	Register
1	`self`	`$rdi`
2	`selector`	`$rsi`
3	`Argument 1`	`$rdx`
4	`Argument 2`	`$rcx`
5	`Argument 3`	`$r8`
6	`Argument 4`	`$r9`
	`Rückgabe`	`$rax`

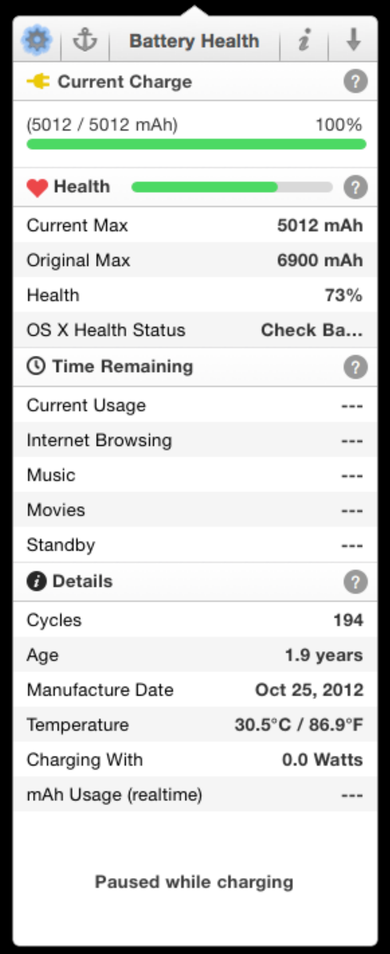
Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

Call Pfad

- AppKit`-[NSControl sendAction:to:]
- libobjc.A.dylib`objc_msgSend
- AppKit`-[NSApplication sendAction:to:from:]
- AppKit`symbol stub for: _os_activity_initiate
- libsystem_trace.dylib`_os_activity_initiate
- AppKit`__36-[NSApplication sendAction:to:from:]_block_invoke
- libobjc.A.dylib`objc_msgSend
- libobjc.A.dylib`-[NSObject performSelector:withObject:]
- libobjc.A.dylib`objc_msgSend
- Battery Health`___lldb_unnamed_function617$$Battery Health

Startpunkt B — Hypothesen

- Header durchsuchen
- Das ist kein normales Menü
  - STTransparentPanel als Ersatz?
  - Zu allgemein (kein Breakpoint)
- [BHMainWindowController toggleAnchored:] hört sich interessant an
  - Klar definierter Breakpoint



Xcode weiß mehr

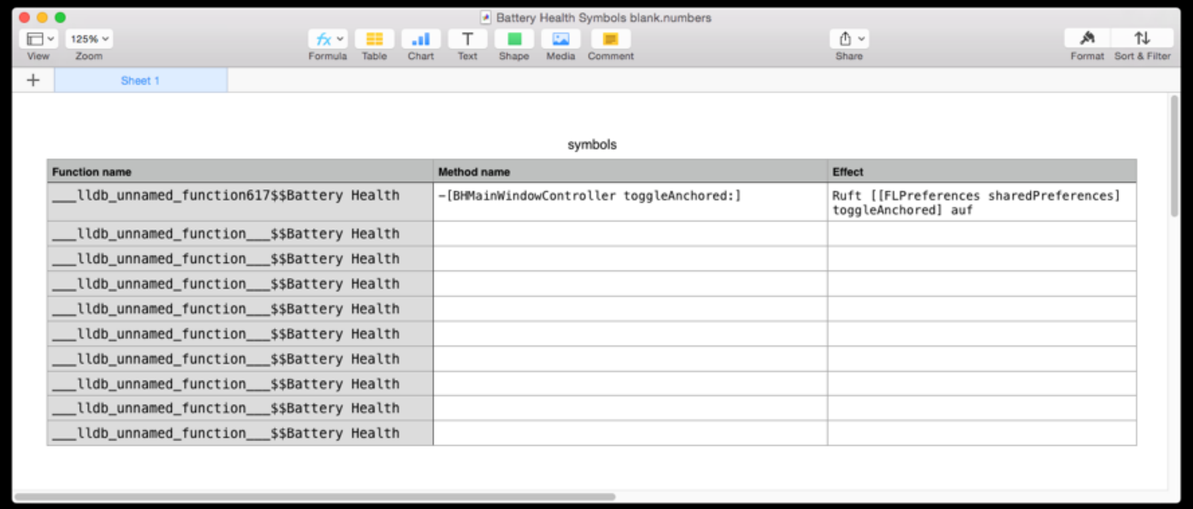
- otool liefert Adressen
- Debugger kann Symbole auflösen
- Beispiel toggleAnchored:
  - Adresse aus otool: `0x100016931`
  - Debugger Befehl: `image lookup --address 0x100016931`

Aus Xcode starten!

```
Address: Battery Health[0x0000000100016931] (Battery Health.__TEXT.__text + 83969)
Summary: Battery Health`___lldb_unnamed_function617$$Battery Health
```

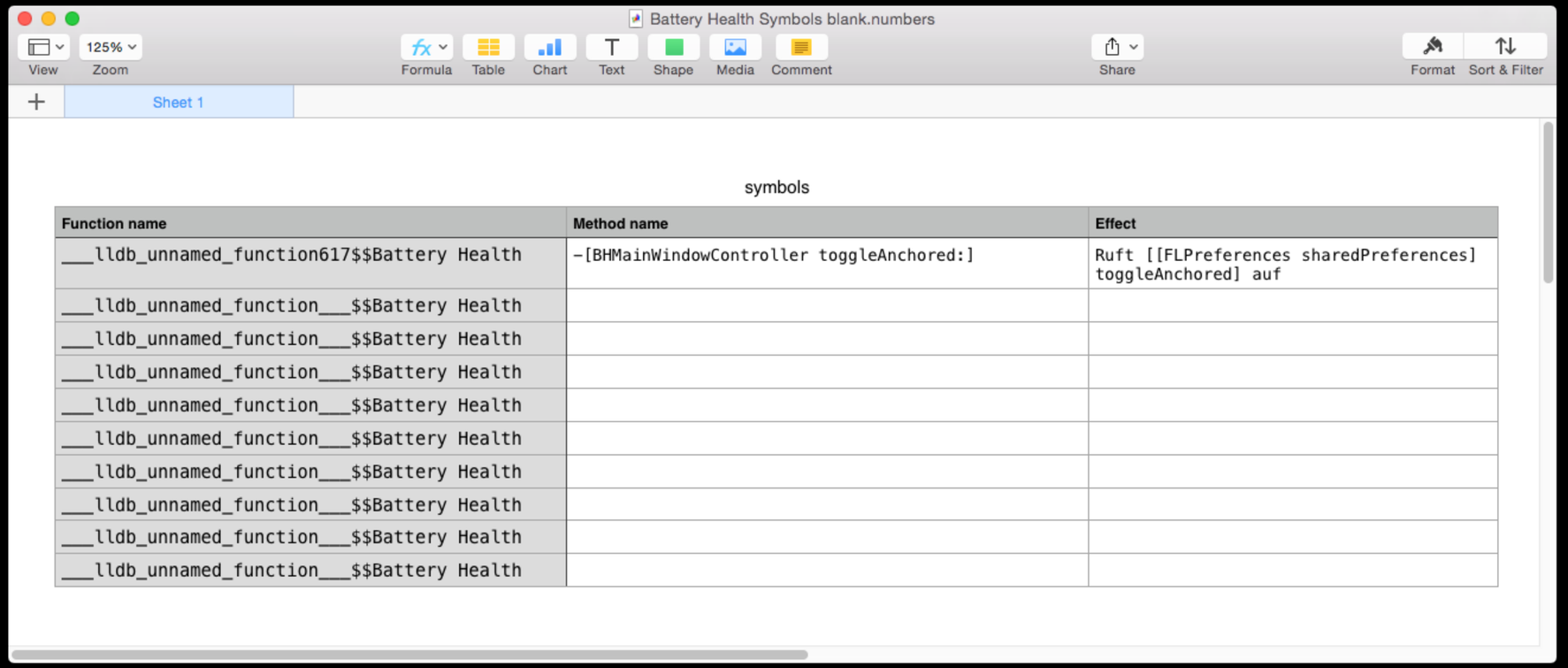
Handarbeit

- Tabellen anlegen mit
  - Funktionsname
  - Zugehörige Obj-C Methode
  - Hypothese über Funktionalität und Seiteneffekte



Function name	Method name	Effect
___lldb_unnamed_function617$$Battery Health	-[BHMainWindowController toggleAnchored:]	Ruft [[FLPreferences sharedPreferences] toggleAnchored] auf
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		

Handarbeit



symbols

Function name	Method name	Effect
___lldb_unnamed_function617$$Battery Health	-[BHMainWindowController toggleAnchored:]	Ruft [[FLPreferences sharedPreferences] toggleAnchored] auf
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		
___lldb_unnamed_function___$$Battery Health		

Moritz Wittenhagen, Thorsten Karrer — Cocoa Heads AC — Okt 2014

Bleibt die Frage

Wieso bleibt das Panel offen und im Vordergrund?