

Agenda Tag 3

- 09:00 → Objective-C Teil 2
iPad
Testen
Wochenaufgabe
Aufgabe: Universal-App + SplitViewController
- 12:00 Pause
- 13:00 → Abschließende Bearbeitung der Aufgaben
Wochenende



Objective-C Teil 2

- ◆ Keine Garbage Collection auf iPhone und iPad
- ◆ Jedes Objekt hat einen "retain count"
 - Zähler für Referenzen
- ◆ Seit iOS 5 automatisches Retain-Zählen (ARC)
 - Bei Projekt-Setup aktivierbar
 - retain und release dürfen dann nicht mehr im Code aufgerufen werden
 - Der Compiler fügt retain und release an den richtigen Stellen im Quellcode ein
 - Sehr starke Vereinfachung!

- ◆ Interesse an Objekt: **retain** aufrufen

```
[objekt retain]; // retain count + 1
```

- ◆ Kein Interesse mehr an Objekt: **release** aufrufen

```
[objekt release]; // retain count - 1
```

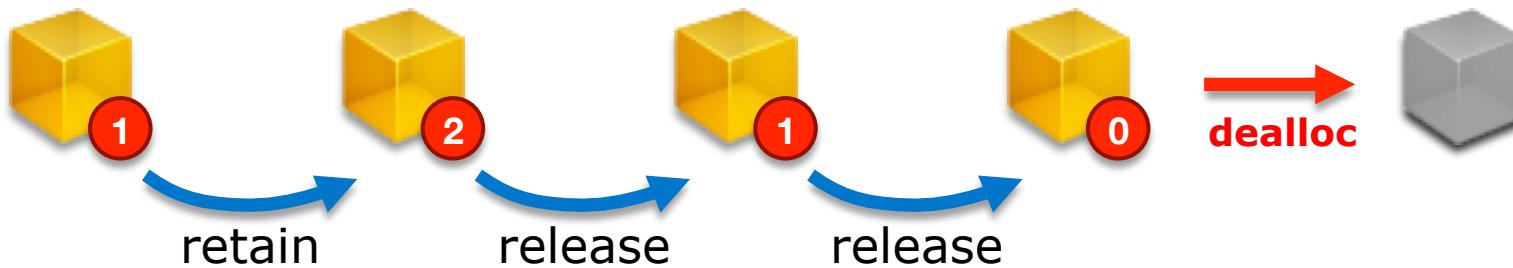
- ◆ Erreicht ein Objekt retain count == 0, wird es automatisch dealloziert
- ◆ Objekte starten mit retain count 1
 - wenn mit **alloc**, **new** oder **copy** erstellt
 - ansonsten nicht

◆ Beispiel

```
konto = [[Konto alloc] init];  
  
[person setKonto:konto];  
  
[konto release];
```

Person.m

```
-(void) setKonto:(Konto*) konto  
{  
    ...  
    [konto retain];  
}  
  
-(void) dealloc  
{  
    ...  
    [konto release];  
}
```



◆ Verzögern von **release**

```
- (Kurs*) seminar {  
    Kurs* derKurs = [[Kurs alloc] init];  
    return derKurs;  
}  
← Retain count ist 1!
```

◆ **Problem:** Wir sollten **release** aufrufen

```
- (Kurs*) seminar {  
    Kurs* derKurs = [[Kurs alloc] init];  
    return derKurs;  
    [derKurs release];  
}  
← zu spät!
```

Autorelease (2)

```
- (Kurs*) seminar {  
    Kurs* derKurs = [[Kurs alloc] init];  
    [derKurs release];  
    return derKurs;  
}
```

zu früh!
Das Objekt wird schon dealloziert

◆ Lösung: autorelease

```
- (Kurs*) seminar {  
    Kurs* derKurs = [[Kurs alloc] init];  
    [derKurs autorelease];  
    return derKurs;  
}
```

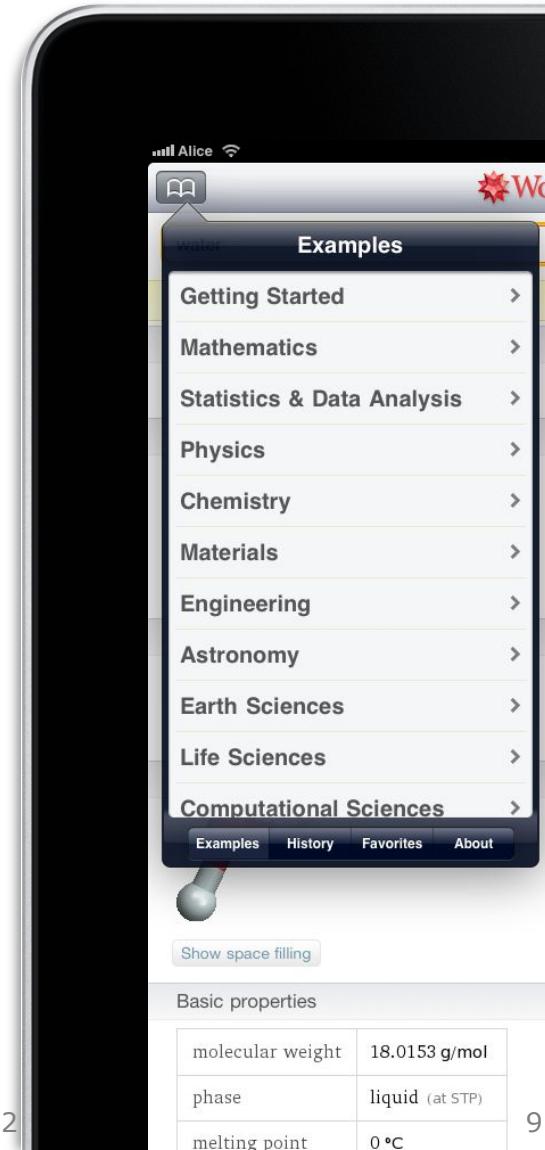
**Release wird „demnächst“
aufgerufen**



iPad

iPad: PopoverController

- ◆ Zeigt einen ViewController als Popup auf dem Bildschirm an
- ◆ Ist selbst kein UIViewController
- ◆ Wird verwendet:
 - ◆ für Zusatzinformationen,
 - ◆ für Werkzeuge,
 - ◆ zur Anzeige von linkem ViewController bei SplitViews in Portrait-Ansicht



iPad: PopoverController

- ◆ **Verwendung von UIPopoverController:**

```
- (IBAction)popupButtonPressed:(id)sender
{
    MyViewController* content = [[MyViewController alloc] init];
    UIP popoverController =
        [[UIPopoverController alloc] initWithContentViewController:content];
    popoverController.delegate = self;

    [popoverController presentPopoverFromBarButtonItem:sender
                                              permittedArrowDirections:UIPopoverArrowDirectionAny
                                              animated:YES];
}
```

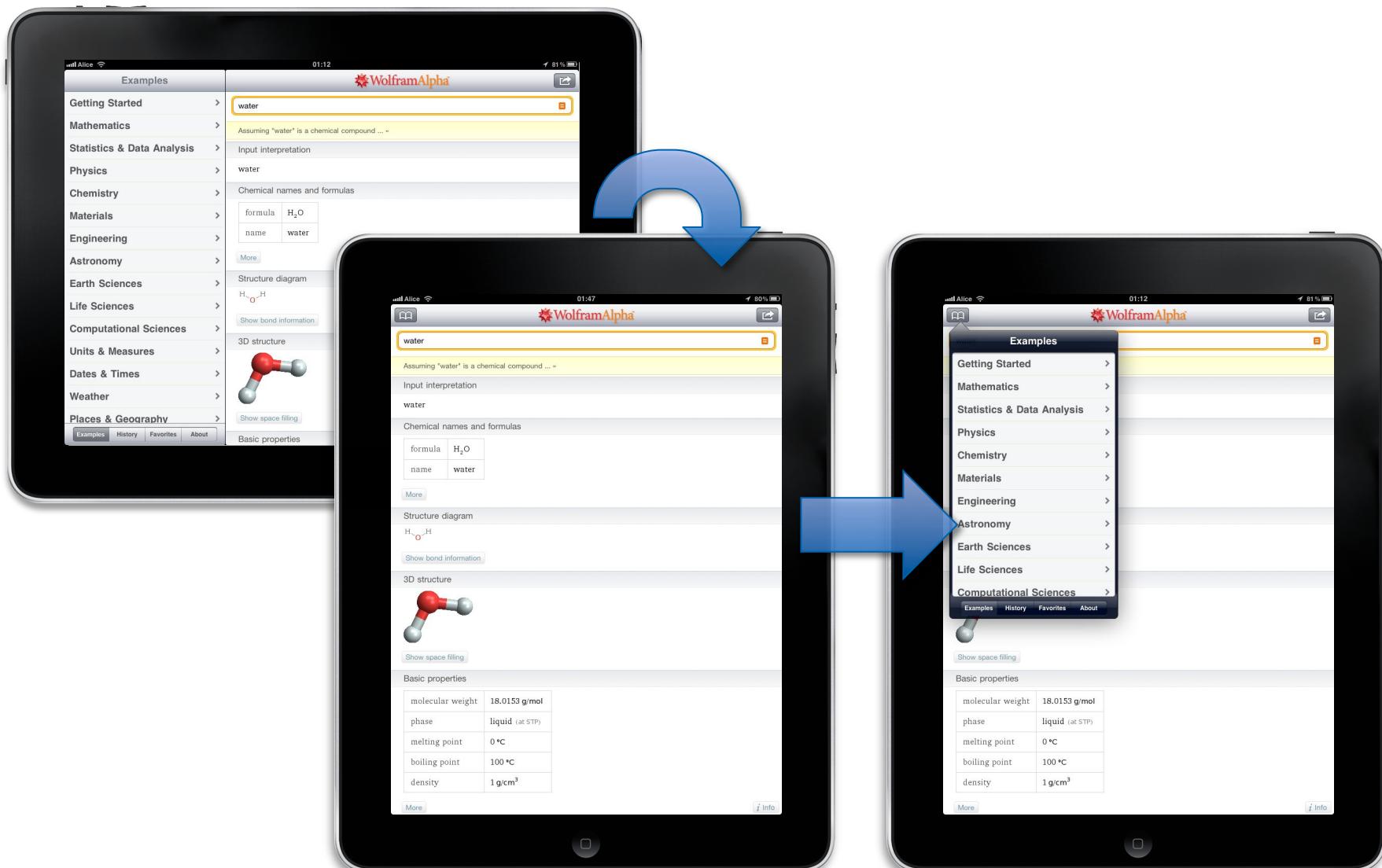
iPad: PopoverController

- ◆ **Entfernen eines UIPopoverController:**

```
- (IBAction)popupButtonPressed:(id)sender
{
    [self.currentSignLocalePopover dismissPopoverAnimated:YES];
}
```



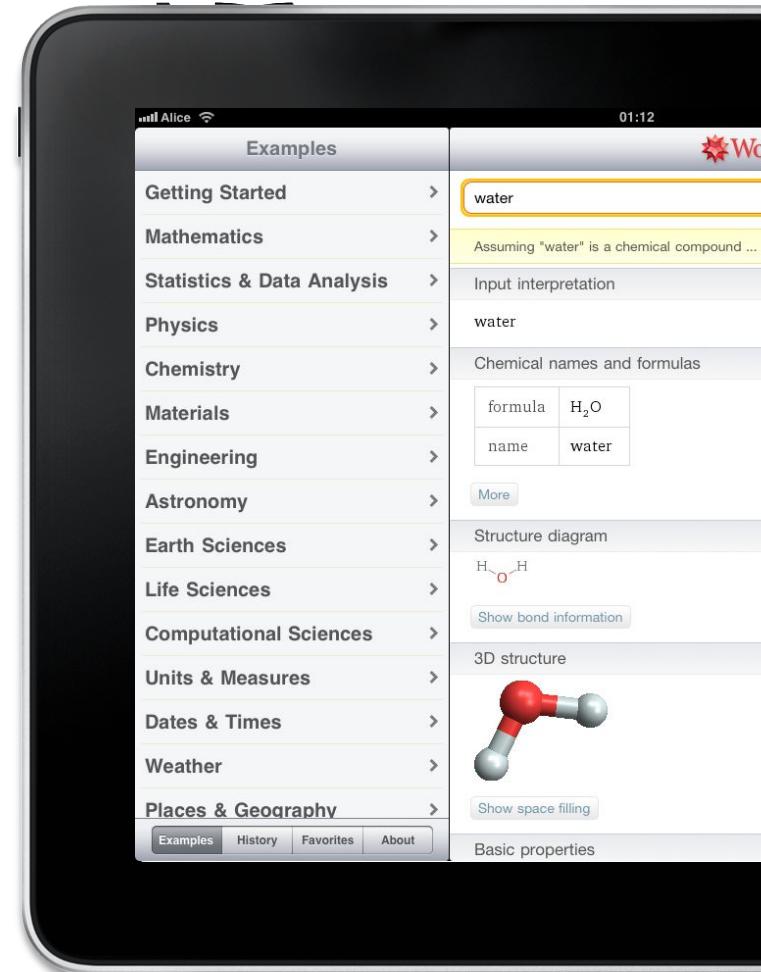
iPad: UISplitViewController



iPad: UISplitViewController

- ◆ Verwaltet 2 ViewController:
 - ◆ Master (linke Seite)
 - ◆ Detail (rechte Seite)

- ◆ Muss ganz oben in der View-Hierarchie stehen
 - ◆ nicht in UINavigationController / TabBarController



iPad: UISplitViewController

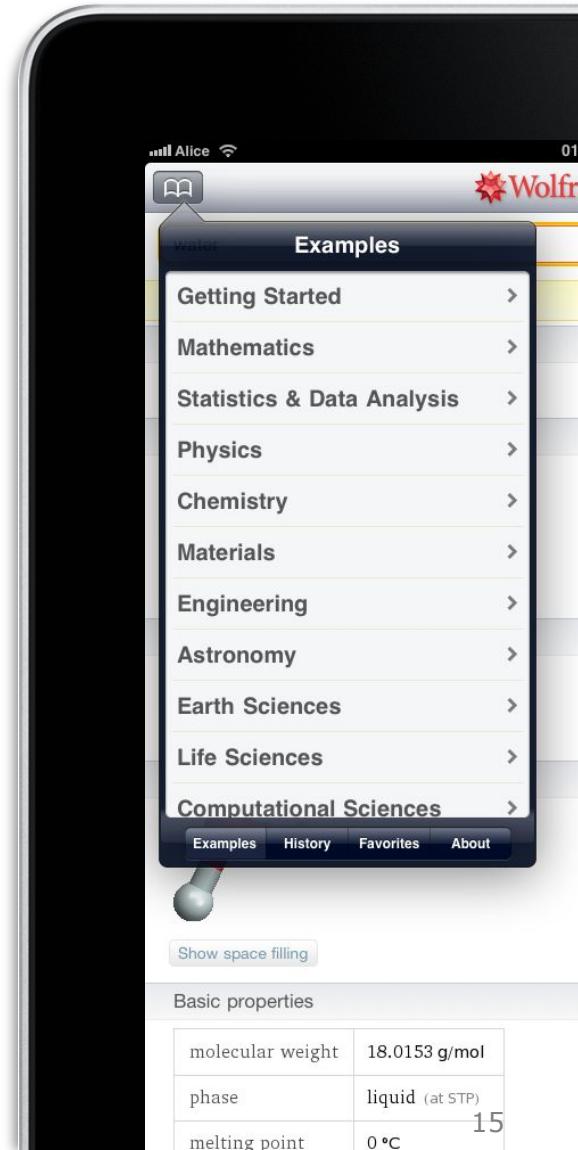
- ◆ **Verwendung von UISplitViewController:**

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {  
    ...  
  
    MasterViewController* masterVC;  
  
    DetailViewController* detailVC;  
  
    UISplitViewController* svc = [[UISplitViewController alloc] init];  
    svc.controllers = [NSArray arrayWithObjects:masterVC, detailVC, nil];  
  
    [window.rootViewController = svc;  
}  
}
```

iPad: SplitView in Portrait-Orientierung

- ◆ Master-View wird im Portrait-Modus ausgeblendet
- ◆ SVC informiert seinen Delegate über Rotation
- ◆ SVC übergibt dem Delegate einen Button + PopoverController zur Anzeige des Master-Views
 - ◆ Popover-Button für Master kann in die Navigationsleiste vom Detail-View
 - ◆ Deshalb DetailController am besten in NavigationController einbetten

```
// UISplitViewControllerDelegate
-(void)splitViewController:(UISplitViewController *)svc
    willHideViewController:(UIViewController *)aViewController
    withBarButtonItem:(UIBarButtonItem*)barButtonItem
    forPopoverController:(UIPopoverController*)pc;
```



iPad: Universal-Apps

- ◆ Anwendungen können sowohl für iPhones als auch iPads optimiert werden
- ◆ Unterscheidung im Code zwischen aktuell laufendem Gerät möglich:

```
BOOL isiPhone =  
    [[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPhone;
```

- ◆ Beispiel: Unterschiedliche XIBs für UIViewController:

```
if ([[UIDevice currentDevice] userInterfaceIdiom] == UIUserInterfaceIdiomPhone)  
{  
    // iPhone-spezifischer Code ...  
    self.viewController = [[MyViewController alloc]  
        initWithNibName:@"ViewController_iPhone" bundle:nil];  
}  
else {  
    // iPad-spezifischer Code ...  
    self.viewController = [[MyViewController alloc]  
        initWithNibName:@"ViewController_iPad" bundle:nil];  
}
```



Testen

- ◆ Verschiedene Ebenen für Tests:
 - Simulator
 - Test auf realem Gerät
 - Unit Tests
 - UI-Tests
 - Analyse von Speicherlecks
- ◆ Xcode beinhaltet bereits ein Test-Framework für
 - Unit Tests
 - UI-Tests (UIAutomation)

- ◆ Tests werden ähnlich wie in JUnit aufgebaut
- ◆ Testen die Funktion einer Klasse
- ◆ Keine UI-Tests oder Tests der IBOutlets und IBActions
- ◆ Können automatisch bei jedem Start der Anwendung ausgeführt werden
- ◆ Benötigen ein eigenes Target
 - „Unit Test Bundle“ dient als Template
- ◆ Für die Tests sollte das Template „Objective-C test case“ verwendet werden



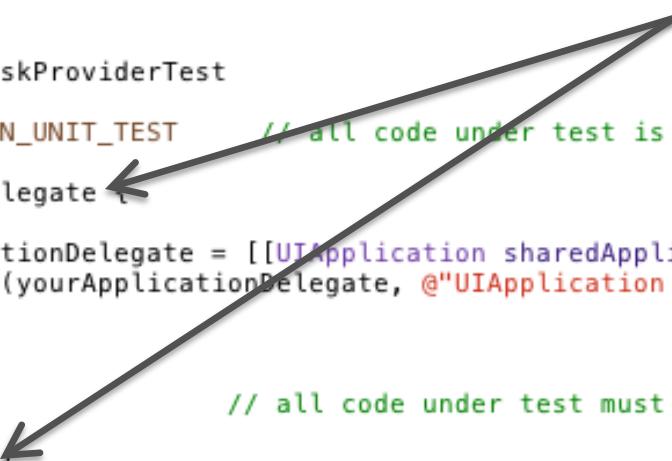
Test Case Header

```
//  
// TaskProviderTest.h  
// TaskViewer  
//  
// Created by Kai Meyer on 11.03.11.  
// Copyright 2011 C1 WPS. All rights reserved.  
//  
// See Also: http://developer.apple.com/iphone/library/documentation/Xcode/Conceptual/iphone  
  
// Application unit tests contain unit test code that must be injected into an application t  
// Define USE_APPLICATION_UNIT_TEST to 0 if the unit test code is designed to be linked into  
  
#define USE_APPLICATION_UNIT_TEST 1  
  
#import <SenTestingKit/SenTestingKit.h> ← Bezug zum Test-Frameworks  
#import <UIKit/UIKit.h>  
//#import "application_headers" as required  
  
@interface TaskProviderTest : SenTestCase {  
}  
  
#if USE_APPLICATION_UNIT_TEST  
- (void) testAppDelegate; // simple test on application  
#else  
- (void) testMath; // simple standalone test  
#endif  
  
@end
```

Test Case Implementation

```
//  
// TaskProviderTest.m  
// TaskViewer  
//  
// Created by Kai Meyer on 11.03.11.  
// Copyright 2011 C1 WPS. All rights reserved.  
  
#import "TaskProviderTest.h"  
  
@implementation TaskProviderTest  
  
#if USE_APPLICATION_UNIT_TEST // all code under test is in the iPhone Application  
- (void) testAppDelegate {  
    id yourApplicationDelegate = [[UIApplication sharedApplication] delegate];  
    STAssertNotNil(yourApplicationDelegate, @"UIApplication failed to find the AppDelegate");  
}  
  
#else // all code under test must be linked into the Unit Test bundle  
- (void) testMath {  
    STAssertTrue((1+1)==2, @"Compiler isn't feeling well today :-(" );  
}  
  
#endif  
  
@end
```

Test-Methoden



- ◆ UIAutomation testet das Verhalten der Anwendung
 - ◆ Programmatisches Bedienen von UI-Elementen
 - ◆ Tests werden als Skripte in JavaScript geschrieben
-
- ◆ Konfiguration der Tests ist aufwendiger
 - ◆ Tests sind im Simulator sowie auf echtem Gerät ablauffähig

- ◆ Voraussetzungen für Tests auf Geräten sind:
 - Development Certificate ist für den Entwickler vorhanden
 - Testgeräte sind registriert
 - App-ID ist erstellt
 - Provisioning Profil ist für die App-ID und die Testgeräte erstellt
 - Ein Testgerät ist an dem Entwicklungsrechner angeschlossen
- ◆ Über Xcode kann mit dem Provisioning-Profil der Code kompiliert und auf das Testgerät übertragen werden



Instruments

Activity-Monitor

Leaks

The screenshot shows two instances of the Instruments application. The left instance is titled "Activity Monitor" and displays a graph of system load over time. The right instance is titled "Leaks" and shows a timeline with spikes indicating memory leaks. A large arrow points from the "Activity Monitor" section towards the "Leaks" section.

Graph	Category	Live Bytes	# Living	# Transitory	Overall Bytes	# Overall	# Allocations (Net / Overall)
Allocation Lifespan	* All Allocations *	3,74 MB	32116	0	3,74 MB	32116	+++
	Malloc 16 Bytes	112,45 KB	7197	0	112,45 KB	7197	
	Malloc 32 Bytes	157,19 KB	5030	0	157,19 KB	5030	
	CFString	120,12 KB	3616	0	120,12 KB	3616	
	Malloc 8 Bytes	21,98 KB	2813	0	21,98 KB	2813	
	Malloc 48 Bytes	55,83 KB	1191	0	55,83 KB	1191	
	CFBasicHash	47,45 KB	1075	0	47,45 KB	1075	
	Malloc 96 Bytes	89,53 KB	955	0	89,53 KB	955	
	CFBasicHash (value-store)	53,02 KB	888	0	53,02 KB	888	
	CFBasicHash (key-store)	44,27 KB	723	0	44,27 KB	723	
	Malloc 160 Bytes	103,44 KB	662	0	103,44 KB	662	
	CFNumber	9,80 KB	627	0	9,80 KB	627	
	Malloc 64 Bytes	37,44 KB	599	0	37,44 KB	599	
	Malloc 80 Bytes	39,30 KB	503	0	39,30 KB	503	
	Malloc 112 Bytes	49,66 KB	454	0	49,66 KB	454	
	CFArray	12,73 KB	382	0	12,73 KB	382	
	CFArray (store-deque)	22,14 KB	325	0	22,14 KB	325	
	Malloc 128 Bytes	38,12 KB	305	0	38,12 KB	305	
	CFString (store)	34,36 KB	292	0	34,36 KB	292	
	CGPath	51,56 KB	275	0	51,56 KB	275	
	Malloc 144 Bytes	36,56 KB	260	0	36,56 KB	260	
	Malloc 1,00 KB	177,00 KB	177	0	177,00 KB	177	

- ◆ Zombies
 - Analyse von „over-released“ Objekten
- ◆ Threads
 - Anzahl und Status von Threads
- ◆ Multicore
 - MultiCore Performance
- ◆ GC Monitor
 - Garbage Collector
- ◆ File Activity
 - Arbeit mit dem Dateisystem
- ◆ Core Data
 - Core Data Aufrufe
- ◆ Core Animation
 - Animationen (fps)
- ◆ Leaks
 - Speicherlecks
- ◆ CPU Sampler
 - CPU-Auslastung der Anwendung
- ◆ Allocations
 - Alloc-Aufrufe
- ◆ Activity Monitor
 - Virtueller Speicher

- ◆ Für die Veröffentlichung muss ein Distributions-Profil erstellt werden
- ◆ Die kompilierte App wird von Apple ausführlich getestet bevor sie in den App-Store hochgeladen wird

- ◆ Verwaltung der Zertifikate, Profile und die Registrierung der Testgeräte wird im iOS Provisioning Portal vorgenommen
- ◆ Das Provisioning Portal ist über das iOS Dev Center erreichbar

<http://developer.apple.com/iphone/>

- ◆ Die Sprache Objective-C
 - ◆ Klassen-Definition, Property-Definition, Speichermanagement
- ◆ Das iOS und die Tools
 - ◆ Xcode, Interface Builder, Simulator, Instruments
- ◆ MVC Pattern
 - ◆ UIView ⇔ UIViewController ⇔ Modell
- ◆ Verbindungen zwischen UI und Code
 - ◆ IBOutlets und IBAction
- ◆ Bedienkonzepte
 - ◆ UINavigationController, UITabBarController, ModalViewController
- ◆ Gesten-Erkennung
 - ◆ Pinch, Tab, Swipe, Pan, ...
- ◆ Testen und Deployment

Was haben wir nicht behandelt

◆ iOS

- Animation
- Zugriff auf Beschleunigungssensoren, Gyroskop, Kamera und Kontakte
- URL-Schemas
- Umgang mit Open GL
- Push-Benachrichtigungen
- Storyboards

◆ Objective-C

- Categories
- Blocks

Wo geht es weiter?

- ◆ iOS Dev Center:
<http://developer.apple.com/devcenter/ios/index.action>
- ◆ Beispiel-Programme (erreichbar über Xcode Hilfe)
- ◆ Developing Apps for iOS (Stanford)
<http://itunes.stanford.edu/>



Aufgabe 5

Universal-App und SplitViewController

Aufgabe 5: Verwenden des SplitViewControllers

Ziel:

- ◆ Eine SplitView ist zu erstellen, um die RSS-Reader-App für den Einsatz auf dem iPad anzupassen. Die Anwendung soll sowohl auf iPhone als auch auf iPad lauffähig sein.

Weg:

- ◆ (falls noch kein Universal-Projekt:) Erstelle ein neues Universal-Projekt (iPad und iPhone) und kopiere deine vorhandenen Dateien dort hinein.
- ◆ Es werden für die iPad- und iPhone-Versionen unterschiedliche XIB-Dateien benötigt.
 - Erstelle zu jeder iPhone-XIB-Datei eine neue XIB-Datei im iPad-Format, falls noch nicht vorhanden.
 - Baue die UI-Elemente aus den alten XIB-Dateien auch in die neuen ein.

Aufgabe 5: Verwenden des SplitViewControllers

- Setze im InterfaceBuilder die Klasse des Files Owner.
- Verbinde die UI-Elemente erneut mit den IBOutlets und IBActions des Files Owner, insbesondere auch die View.
- ◆ Verwende als obersten ViewController für die iPad-Version einen SplitViewController.
 - Unterscheide im AppDelegate, ob iPhone-oder iPad-Version und setze entsprechend den ContainerController (Split oder Navigation).
 - Die Feed-Tabelle und Details sollen gleichzeitig dargestellt werden.
 - Überlege, was passieren soll wenn die Webseite geöffnet wird.
 - Ermögliche Rotation

Hinweis: Dazu ist es sinnvoll, in dem SplitViewController sowohl auf der linken, als auch auf der rechten Seite einen UINavigationController zu haben. So kann in dem NavigationItem der Knopf integriert werden, um die Artikel-Liste in der Portrait-Ansicht anzuzeigen.

Aufgabe 5: Verwenden des SplitViewControllers

- ◆ Der SplitViewController kann ein Delegation-Objekt haben, dem er bei Rotation in den Portrait-Mode einen fertigen Button für die Einbindung in die Toolbar der rechten Seite schickt.
 - Implementiere das Protokoll für dieses Delegate in deinem DetailViewController.
 - Setze deinen DetailViewController als Delegate an dem SplitViewController.
 - Binde den Button ein, wenn die Delegations-Methoden an dem Detail-ViewController aufgerufen werden.

Vielen Dank für
Ihre Aufmerksamkeit und Ihr Interesse



Feedback, Fragen und Anregungen
sind willkommen

Wochenaufgabe

- ◆ Vorstellung
- ◆ Terminfindung