

HANSER



Leseprobe

Thomas Sillmann

Swift im Detail

ISBN (Buch): 978-3-446-44294-8

ISBN (E-Book): 978-3-446-44423-2

Weitere Informationen oder Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-44294-8>

sowie im Buchhandel.

Inhalt

1	Apples neue Programmiersprache: Swift	1
1.1	Willkommen bei Swift!	1
1.2	Warum Swift?	1
1.3	Swift und Objective-C	2
1.4	Voraussetzungen für die Swift-Entwicklung	3
1.4.1	Xcode	3
1.4.2	Mac	4
1.5	Swift-Ressourcen	5
1.5.1	Apples Entwickler-Dokumentation	5
1.5.2	Swift-Blog	7
1.5.3	Code-Beispiele des Autors	8
1.5.4	Das Internet	9
2	Grundlagen der Programmierung	11
2.1	Variablen und Konstanten	15
2.1.1	Type Inference und Type Annotation	17
2.2	Abfragen und Schleifen	18
2.2.1	Bedingungen	18
2.2.2	If	21
2.2.3	While	23
2.2.4	Do-While	24
2.2.5	For	25
2.2.6	For-In	26
2.2.7	Switch	28
2.2.8	Control Transfer Statements	31
2.3	Kommentare	33
2.3.1	Verschachtelte Kommentare	33
2.3.2	Schlüsselwörter für Kommentare	34
2.4	Fundamental Types	35
2.4.1	Strings und Characters	36
2.4.2	Arrays	40

2.4.3	Dictionaries	49
2.4.4	Tuples	58
2.5	Funktionen	60
2.5.1	Grundaufbau und Aufruf einer Funktion	61
2.5.2	Eine erste einfache Funktion	61
2.5.3	Funktion mit Parametern	62
2.5.4	Funktion mit Rückgabewert	63
2.5.5	Funktion mit mehreren Rückgabewerten	66
2.5.6	Funktion mit optionalem Rückgabewert	67
2.5.7	Funktion mit optionalen Parametern	68
2.5.8	Local und External Parameter Names	69
2.5.9	Funktionen mit Standardwerten für Parameter	72
2.5.10	Funktionen mit beliebiger Parameterzahl	74
2.5.11	Funktionen mit Variablen als Parameter	75
2.5.12	Funktionen mit veränderbaren In-Out-Parametern	76
2.5.13	Function Types	78
2.5.14	Verschachtelte Funktionen	82
2.6	Closures	84
2.6.1	Closures als Variablen und Konstanten	85
2.6.2	Closures als Parameter für Funktionen	86
2.6.3	Kurzschreibweise für Closures als Parameter von Funktionen	90
2.7	Enumerations	92
2.7.1	Kurzschreibweisen für Enumerations	95
2.7.2	Enumerations mittels Switch abfragen	96
2.7.3	Zusätzliche Informationen in Enumeration-Werten speichern	97
2.7.4	Member einer Enumeration feste Werte zuweisen	99
2.7.5	Enumerations sind Value Types	101
2.8	Structures	102
2.8.1	Erstellen einer neuen Instanz	103
2.8.2	Structures mit Properties	104
2.8.3	Structures mit Methoden	108
2.8.4	Structures sind Value Types	109
3	Objektorientierte Programmierung mit Swift	111
3.1	Swift und objektorientierte Programmierung	111
3.2	Klassen	112
3.2.1	Erstellen und Verwenden einer neuen Instanz	113
3.2.2	Initialisierung von Objekten einer Klasse	114
3.2.3	Klassen sind Reference Types	117
3.2.4	Unterschiede zwischen Klassen und Strukturen	119
3.3	Properties	120
3.3.1	Stored Properties	121
3.3.2	Computed Properties	127

3.3.3	Property Observers	133
3.3.4	Globale und lokale Variablen	137
3.3.5	Type Properties	138
3.4	Methoden	141
3.4.1	Instance Methods	141
3.4.2	Type Methods	148
3.5	Subscripts	150
3.5.1	Aufbau von Subscripts	150
3.5.2	Subscript Overloading	154
3.6	Optionals	155
3.6.1	Forced Unwrapping	157
3.6.2	Optional Binding	160
3.6.3	Implicit Unwrapping	161
3.6.4	Optional Chaining	163
3.7	Vererbung	170
3.7.1	Vererbung im Detail	171
3.7.2	Überschreiben von Properties, Methoden und Subscripts	174
3.7.3	Zugriff auf Properties, Methoden und Subscripts der Superklasse	178
3.8	Initialisierung	179
3.8.1	Grundaufbau eines Initializers	179
3.8.2	Initializer mit Parametern	181
3.8.3	Default Initializer	183
3.8.4	Local Parameter Names und External Parameter Names in Initializern	185
3.8.5	Initializer und Optionals	186
3.8.6	Initializer und Constant Stored Properties	188
3.8.7	Erstellen mehrerer Initializer	189
3.8.8	Initializer und Vererbung	194
3.8.9	Deinitialisierung	212
3.9	Speicherverwaltung mit ARC	214
3.9.1	Strong References und Reference Cycles	215
3.9.2	Weak References	218
3.9.3	Unowned References	221
3.9.4	Best Practices zur Speicherverwaltung	227
3.9.5	Closure Capture List	227
3.10	Type Casting	232
3.10.1	Typ prüfen mit is	233
3.10.2	Downcasting mit as	234
3.10.3	Any und AnyObject	235
3.11	Nested Types	238
4	Weiterführende Sprachmerkmale von Swift	241
4.1	Extensions	241
4.1.1	Syntax	242

4.1.2	Computed Properties	242
4.1.3	Methoden	243
4.1.4	Initializer	244
4.1.5	Subscripts	245
4.1.6	Nested Types	246
4.2	Protocols	247
4.2.1	Syntax	248
4.2.2	Deklaration von Properties	249
4.2.3	Deklaration von Methoden	251
4.2.4	Deklaration von Initializern	254
4.2.5	Protocol Type	257
4.2.6	Delegation	258
4.2.7	Protocol Composition	262
4.2.8	Protocols und Extensions	264
4.2.9	Vererbung	266
4.2.10	Class-Only Protocols	268
4.2.11	Protocol Conformance	269
4.2.12	Optionale Eigenschaften	271
4.3	Generics	273
4.3.1	Generic Functions	274
4.3.2	Generic Types	276
4.3.3	Type Constraints	278
4.3.4	Associated Types	279
4.4	Access Control	283
4.4.1	Modules und Source Files	284
4.4.2	Access Levels	285
4.4.3	Syntax	285
4.4.4	Access Levels in Custom Types	286
4.4.5	Access Levels in Getter und Setter einer Property	289
5	Swift, Cocoa und Objective-C	291
5.1	Interoperability	292
5.1.1	Swift Type Compatibility	293
5.1.2	Selectors in Objective-C	295
5.1.3	Optionals in Swift und Objective-C	295
5.1.4	Arbeiten mit dem Interface Builder	296
5.1.5	Arbeiten mit Core Data Managed Object Subclasses	297
5.1.6	Automatic Bridging	298
5.1.7	Cocoa Design Patterns	300
5.2	Mix and Match	300
5.2.1	Mix and Match innerhalb eines App-Targets	301
5.2.2	Mix and Match innerhalb eines Framework-Targets	303
5.3	Migration	304

6	Swift und Xcode	307
6.1	Installation von Xcode	307
6.2	Erstellen eines neuen Swift-Projekts	309
6.3	Der Grundaufbau von Xcode	312
6.4	Neue Swift-Dateien erstellen	316
6.5	Refactoring – leider nein!	318
6.6	Playgrounds im Detail	318
7	Profi-Wissen und Tipps für die tägliche Arbeit	323
7.1	Zahlenwerte übersichtlicher gestalten	323
7.2	Benennung von Variablen und Konstanten mit Sonderzeichen und Emoticons ..	324
7.3	Switch für Fortgeschrittene	325
7.3.1	Tuples	325
7.3.2	Value Binding	326
7.3.3	Where	326
7.4	Kurzschreibweise für Abfragen bei return	327
7.5	Custom Operators	327
7.6	Swift-Beispielprojekte	329
	Index	331

1

Apples neue Programmiersprache: Swift

■ 1.1 Willkommen bei Swift!

Das war schon eine ziemliche Überraschung im Juni 2014 zu Beginn von Apples alljährlicher Entwicklerkonferenz WWDC. Ich als Entwickler habe mit vielem gerechnet, allen voran mit der Vorstellung der neuen Betriebssystemversionen von OS X und iOS. Auch eine iWatch (inzwischen als Apple Watch bekannt) hätte ich mir damals bereits vorstellen können. Doch dass Apple dann nicht nur Unmengen neuer APIs und Frameworks für Entwickler aus dem Hut zaubert, sondern gleich noch eine komplett neue Programmiersprache vorstellt, hat wahrlich meine kühnsten Vorstellungen übertroffen und meine Kinnlade während der Präsentation weit herabsinken lassen. Immerhin bin ich damit nicht alleine, denn Swift war eine spannende und gänzlich unerwartete Vorstellung auf der WWDC 2014. Die Begeisterung und Faszination für diese Sprache ist seitdem ungebrochen und es gibt kaum einen iOS- und OS X-Entwickler, der sich nicht ausgiebig mit Apples neuer Programmiersprache auseinandersetzt. Nicht nur, dass Swift einfacher anzuwenden sein soll als das bisher von Apple präferierte Objective-C, nein, auch sollen zukünftig mit Swift entwickelte Apps bis zu 30% schneller sein als ihre Objective-C-Pendants. Allein das ist Grund genug, dass sich nicht nur Einsteiger und Apple-Neulinge mit Swift intensiv auseinandersetzen.

■ 1.2 Warum Swift?

Laut Apple begann die Entwicklung der Programmiersprache Swift im Jahr 2010. Hauptgrund für die Entwicklung von Swift dürfte unter anderem gewesen sein, dass Objective-C als bisherige Hauptsprache zur Programmierung für iOS und OS X den ein oder anderen Entwickler abschreckte. Nicht zuletzt aufgrund seiner Syntax war und ist Objective-C – gerade für Umsteiger von anderen Programmiersprachen – anfangs ein wenig befremdlich und schwer zu verstehen und zu erlernen. Eine Programmiersprache wie Java, die beispielsweise für die Entwicklung von Android-Apps genutzt wird und wesentlich weiter verbreitet ist, lockt da potenzielle App-Entwickler schon eher. Dieses Umstands war sich auch Apple bewusst, und so hat Swift auch einen klaren Grundsatz: Einfach, schnell, Spaßig. Mit Swift

sollen Entwickler schnell und einfach eigene Anwendungen für iOS und OS X schreiben können und wahrlich Spaß an der Entwicklung haben.

Zu diesem Zweck hat Apple für Swift die Vorteile und Merkmale aus verschiedensten Programmiersprachen als Basis genommen und in Swift einfließen lassen. Neben Objective-C dienen so beispielsweise auch Ruby und Python als Vorlage für Swift.

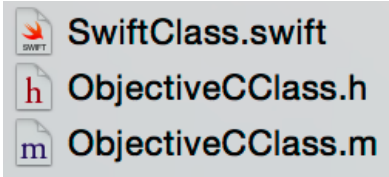
Auch wenn selbstredend aktuell Objective-C bei den meisten App-Entwicklern weiter verbreitet ist, so ändert das aber nichts daran, dass Swift langfristig die bessere Alternative zu Objective-C sein dürfte. Allein die Tatsache, dass mit Swift entwickelte Apps bis zu 30% schneller ausgeführt werden als in Objective-C geschriebene Anwendungen, ist eine deutliche Ansage. Auch macht die Einfachheit von Swift – verbunden mit allen Möglichkeiten und API-Zugriffen wie mit Objective-C auch – die Sprache ebenso für alteingesessene Entwickler attraktiv. Und ob Apple früher oder später nicht doch irgendwann den Objective-C-Hahn zudreht und nur noch Swift unterstützt, bleibt abzuwarten.

Sollten Sie jetzt ganz frisch mit der iOS- und/oder OS X-Entwicklung beginnen, kann ich Ihnen nur wärmstens empfehlen, sich voll und ganz auf Swift zu konzentrieren. Sie erlernen dann eine moderne Sprache mit moderner Architektur, die in den kommenden Jahren noch weitaus mehr Popularität und Anhänger gewinnen wird.

Und wenn Sie bereits mehrere Jahre mit Objective-C entwickeln (so wie ich auch), dann lassen Sie sich nichtsdestotrotz auf Swift ein. Starten Sie entweder ein neues Projekt mit Swift oder verwenden Sie Swift zur Erweiterung Ihrer bestehenden Anwendungen (denn Objective-C und Swift können problemlos gemischt innerhalb eines Projekts verwendet werden, aber dazu gleich mehr). Langfristig wird Swift *die* Sprache zur Entwicklung von iOS- und OS X-Apps sein, und umso früher Sie Ihre Expertise in diesem Bereich ausbauen und Erfahrungen sammeln, desto besser sind Sie auf alle Neuerungen und Erweiterungen der Sprache in Zukunft vorbereitet.

■ 1.3 Swift und Objective-C

Wie bereits geschrieben, ist Swift zum aktuellen Zeitpunkt kein Ersatz, sondern eine neue Programmiersprache neben Objective-C zur Entwicklung von iOS- und OS X-Apps. Damit Sie sich aber nicht ausschließlich für eine der beiden entscheiden müssen, haben Sie die Möglichkeit, Swift- und Objective-C-Code parallel in einem Projekt zu verwenden. Zwar können Sie nicht innerhalb einer Quellcode-Datei Swift- und Objective-C-Code mischen, können aber in einem Projekt sowohl Swift- als auch Objective-C-Klassen verwenden, die dann jeweils Code der entsprechenden Programmiersprache enthalten (siehe Bild 1.1). Das erlaubt Ihnen, selbst bereits bestehende und in Objective-C geschriebene Anwendungen um neue Swift-Klassen und -Funktionen zu erweitern.

**Bild 1.1**

Ja, sie verstehen sich; Objective-C- und Swift-Klassen können problemlos parallel in einem Projekt verwendet werden.

Mehr zur parallelen Verwendung von Swift und Objective-C erfahren Sie im Kapitel 5 „Swift, Cocoa und Objective-C“.

■ 1.4 Voraussetzungen für die Swift-Entwicklung

1.4.1 Xcode

Sie können Swift momentan ausschließlich für die Entwicklung von Apps für iOS und OS X verwenden. Grundlage für die Entwicklung solcher Apps ist Xcode – Apples hauseigene Entwicklungsumgebung (siehe Bild 1.2). Xcode ist ausschließlich auf dem Mac verfügbar und läuft in der aktuellen Version 6 ab OS X 10.9.4. Neben dem Editor und weiteren Tools für die Software-Entwicklung enthält Xcode auch alle benötigten SDKs sowie den Compiler, um Anwendungen in Swift für iOS und OS X zu schreiben und auszuführen.

**Bild 1.2**

Apples Entwicklungsumgebung Xcode enthält alles, was Sie für die Entwicklung von Anwendungen mit Swift benötigen.

Die Installation von Xcode gestaltet sich sehr simpel: Sie finden die Anwendung über den Mac App Store, der inzwischen fester Bestandteil des Betriebssystems OS X des Mac ist. Darüber können Sie Xcode einfach herunterladen und installieren (siehe Bild 1.3). Aufgrund seiner Größe von etwas mehr als zwei Gigabyte kann es aber sein, dass der Download – je nach vorhandener Internetverbindung – eine gewisse Zeit in Anspruch nimmt.



Bild 1.3 Xcode können Sie direkt aus dem App Store heraus auf Ihrem Mac installieren.

Neben dem Editor und den SDKs bringt Xcode unter anderem auch iOS-Simulatoren, Apples Entwickler-Dokumentation, eine integrierte Versionsverwaltung und weitere hilfreiche Tools mit; eine ausführliche Vorstellung der IDE würde den Rahmen dieses Buches bei Weitem sprengen. Alle notwendigen Grundlagen, um die in diesem Buch beschriebenen Beispiele nachzuvollziehen und mit der Entwicklung von Apps mit Swift zu beginnen, werde ich aber selbstverständlich im Detail vorstellen und erläutern. Dazu finden Sie in Kapitel 6, „Swift und Xcode“, weitere Informationen zur Installation und Arbeit mit Xcode im Zusammenspiel mit Apples neuer Programmiersprache Swift.

1.4.2 Mac

Wie Sie sehen, ist auch ein Mac Voraussetzung, um mit Swift entwickeln zu können, denn Apple stellt seine SDKs ausschließlich für die eigene Plattform bereit. Welchen Mac Sie letztlich für die Arbeit mit Xcode verwenden, bleibt gut und gerne Ihren persönlichen Vorlieben überlassen; leistungsstark genug sind alle verfügbaren Modelle (siehe Bild 1.4). Lediglich beim Arbeitsspeicher sollten es nach Möglichkeit mindestens 8 GB sein; das erleichtert das Ausführen Ihrer Apps ungemein und macht die Arbeit mit Xcode komfortabler.



Bild 1.4 Die Wahl des passenden Mac zur Entwicklung mit Swift bleibt komplett Ihren persönlichen Vorlieben überlassen.

■ 1.5 Swift-Ressourcen

Ich darf Ihnen an dieser Stelle gratulieren: Eine der wichtigsten Ressourcen für Ihre tägliche Arbeit mit Swift halten Sie bereits in Händen. ☺ Nein im Ernst, natürlich werden Sie mithilfe dieses Buches die Entwicklung von Apps mittels Swift erlernen und in die Lage versetzt, eigene Anwendungen mit Swift zu schreiben und Programmierprobleme selbstständig zu lösen. Aber es ist eine Tatsache, dass dieses Buch nicht jeden Teilbereich vollumfänglich abdecken kann und sich darüber hinaus Swift im Laufe der Jahre auch noch weiterentwickeln und möglicherweise verändern wird. Aus diesem Grund ist es wichtig zu wissen, welche weiteren Ressourcen Ihnen zur Verfügung stehen und welche Sie für Ihre tägliche Arbeit nutzen können.

1.5.1 Apples Entwickler-Dokumentation

Am wichtigsten hierbei ist Apple mit seiner ausführlichen und umfangreichen Entwickler-Dokumentation. Diese Dokumentation ist sowohl online für registrierte Apple-Entwickler als auch über Apples Entwicklungsumgebung Xcode verfügbar. Sie enthält nicht nur die Beschreibung aller verfügbaren APIs für die iOS- und OS X-Entwicklung, nein, sie enthält darüber hinaus auch Code-Beispiele, Schritt-für-Schritt-Anleitungen und stets die topaktuellsten Informationen zu Swift.

Als registrierter Apple-Entwickler können Sie die Online-Dokumentation auf Apples Entwicklerplattform unter <https://developer.apple.com> abrufen (siehe Bild 1.5). Dort erhalten Sie auch Zugang zu Vorabversionen von Xcode, iOS und OS X, müssen dazu aber einem der kostenpflichtigen Apple Developer Programs beitreten.

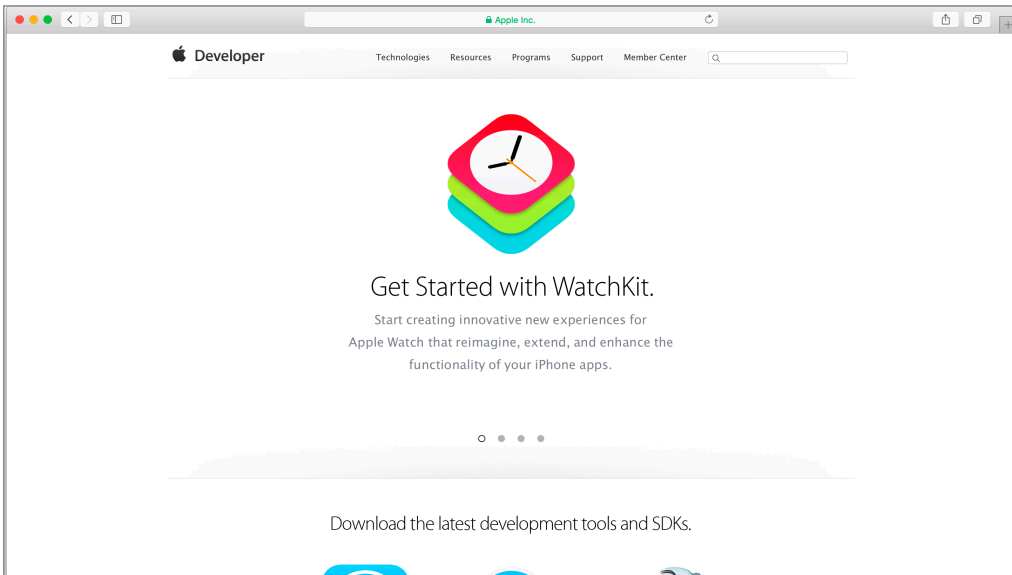


Bild 1.5 Der Ausgangspunkt für jeden Apple-Entwickler: Apples Developer-Portal.



Entwickler-Dokumentation herunterladen

Wie beschrieben ist die Entwickler-Dokumentation von Apple nicht nur online, sondern auch offline über die IDE Xcode verfügbar. Dazu müssen Sie diese aber erst einmal über Xcode herunterladen, denn standardmäßig greift auch Xcode auf die Online-Version der Dokumentation zu.

Gehen Sie zum Download der Entwickler-Dokumentation wie folgt vor:

1. Starten Sie Xcode
2. Wechseln Sie in die Einstellungen über das Menü **XCODE** → **PREFERENCES...** oder verwenden Sie das Tastaturkürzel **CMD + ,**
3. Wechseln Sie in den Reiter **DOWNLOADS**

Dort werden Ihnen im Abschnitt „Documentation“ alle verfügbaren Dokumentationen angezeigt (siehe Bild 1.6). Rechts daneben befindet sich ein Download-Button, durch den Sie den Download einer Dokumentation anstoßen können.

Damit Xcode automatisch immer den aktuellsten Stand der Entwickler-Dokumentation herunterlädt und somit offline vorhält, können Sie den Haken bei *Check for and install updates automatically* setzen. Damit prüft Xcode bei jedem Start, ob eine aktualisierte Version der Dokumentation vorhanden ist, und lädt diese bei Bedarf automatisch im Hintergrund herunter.

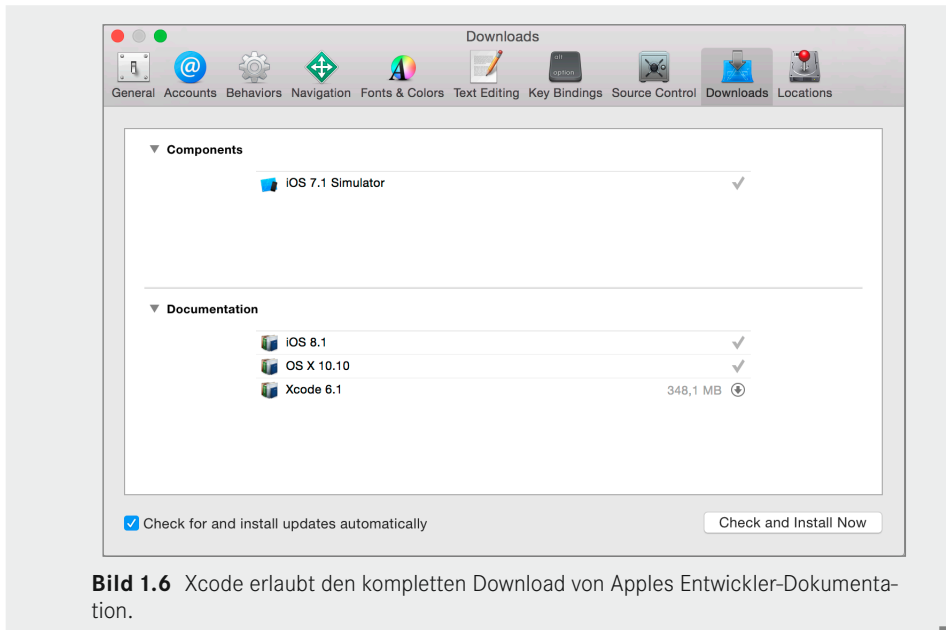


Bild 1.6 Xcode erlaubt den kompletten Download von Apples Entwickler-Dokumentation.

1.5.2 Swift-Blog

Mit der Vorstellung von Swift startete Apple auf seiner Entwickler-Website einen eigenen Swift-Blog. Dieser informiert einerseits über die Fortschritte und die Entwicklung der Sprache, zeigt aber auch typische Design Patterns und Best Practices auf. Der Blog ist also immer eine großartige Informationsquelle, um Neuigkeiten über Swift zu erfahren. Darüber hinaus erlaubt der Blog den direkten Zugriff auf wichtige Entwickler-Ressourcen wie passende WWDC-Sessions zu Swift, die Entwicklungsumgebung Xcode sowie Code-Beispiele von Apple zu Swift.

Der Swift-Blog ist über den URL <https://developer.apple.com/swift/> erreichbar (siehe Bild 1.7).

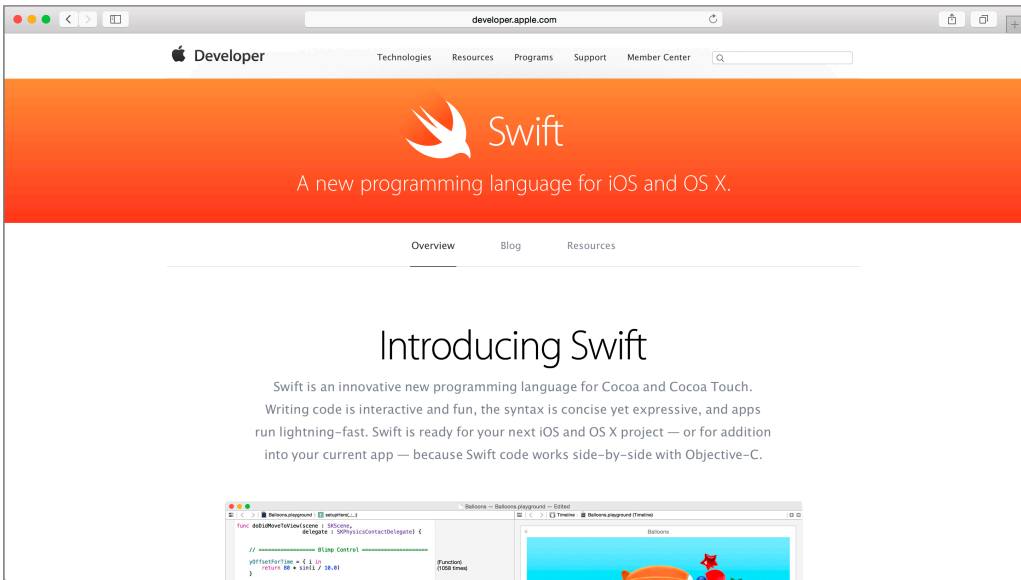


Bild 1.7 Der Swift-Blog von Apple ist immer eine lohnenswerte Anlaufstelle für Entwickler.

1.5.3 Code-Beispiele des Autors

Im Zuge meines vorangegangenen Buchprojekts „Apps für iOS 8 professionell entwickeln“ habe ich auf meiner Website einen eigenen Bereich für diverse vorgefertigte Klassen eingefügt. Dabei handelt es sich um generische Klassen, die Sie so problemlos und in der Regel ohne Anpassungen am Code in Ihren eigenen Projekten verwenden können. All diese Klassen habe ich sowohl in Objective-C als auch in Swift geschrieben und sie stehen auf der Website zum Buch zum Download bereit. Sie können diese Klassen einerseits dazu nutzen, um einmal einen Blick auf praktisch umgesetzten Swift-Code zu werfen sowie die ein oder andere dieser Klassen möglicherweise auch in eigenen Projekten einzusetzen. Darüber hinaus können Sie damit einen direkten Vergleich zwischen Objective-C- und Swift-Code anstellen, indem Sie sich beide Sprachversionen zu einer Klasse herunterladen und anschließend vergleichen.



thomassillmann.de/swift-buch

Schauen Sie doch einfach mal auf meiner Website zum Buch vorbei, hier finden Sie die Code-Beispiele sowie zusätzliche Informationen zum Thema.

1.5.4 Das Internet

Wenn Sie kein Neuling in der App-Entwicklung (oder auch allgemein in der Software-Entwicklung) sind, brauche ich Ihnen das sicherlich gar nicht zu erzählen, doch für alle Neulinge tue ich es trotzdem: Sind alle bereits vorgestellten Ressourcen erschöpft und Sie kommen partout bei einem Problem nicht weiter, hilft Ihnen womöglich das Wissen des World Wide Web weiter. Eine passende Anfrage bei der Suchmaschine Ihrer Wahl liefert Ihnen meist eine gute Auswahl passender Tutorials und Hilfestellungen.

Häufig werden Sie dabei auch sicherlich auf das Portal Stack Overflow (*stackoverflow.com*) stoßen. Dabei handelt es sich um eine Plattform für Software-Entwickler aus allen Bereichen, die dort Fragen einstellen können, die sodann andere Mitglieder aus der Stack Overflow-Community beantworten können.

Auch darüber hinaus werden Sie in den Weiten des Internets viele verschiedene Informationen finden. Gerne können Sie sich auch von meinen Blog-Beiträgen zum Thema App-Entwicklung inspirieren lassen, schauen Sie dazu einfach auf *thomassillmann.de* vorbei.

Index

Symbole

@objc 269

A

Abfrage 18
– Kurzschreibweise 327
Access Control 283
– Access Levels 285
– Default Access Levels 285
– Internal Access 285
– Module 284
– Private Access 285
– Public Access 285
– Source File 284
Any 235, 237
AnyObject 235f.
ARC 214
Array 12, 40
– Änderung 44
– Erstellen 41
– immutable 44
– Leeren 47
– mutable 44
– Zugriff 44
as 234
Auto Completion 252
Automatic Bridging 298
Automatic Reference Counting 212

B

Bedingung 18
Beispiele 8f.

Beispielprojekte 329
Bool 12
break 29, 31

C

Character 36
Closure 84
– Trailing Closures 91
Closure Capture List 227
Cocoa 291
continue 31
Control Transfer Statement 31
Core Data 297
Custom Operators 327

D

Debug Area 315
Deinitialisierung 212
Deinitialization 212
Delegation 258, 300
Design Patterns 300
Dictionary 12, 49
– Änderung 53
– Erstellen 50
– Zugriff 53
Double 12
Do-While 24
Downcasting 234

E

Editor Area 314
Emoticons 324

- Entwickler-Dokumentation 5
 - Download 6
- Enumeration 92
 - Associated Values 97
 - Benennung 94
 - Kurzschreibweisen 95
 - Members 93
 - Member Values 93
 - Raw Values 99
- Equatable 282
- Extension 241
- External Parameter Name 69, 142, 145, 185
 - Shorthand External Parameter Name 71

F

- fallthrough 32
- FIXME 34
- Fließkommazahl 12
- Float 12
- For 25
- For-In 26
- fromRaw() 100
- Function Type 78
- Fundamental Types 12, 35
- Funktion 60
 - Aufruf 61
 - Globale Funktionen 83
 - Grundaufbau 61
 - Verschachtelte Funktionen 82f.

G

- Ganzzahl 12
- Generics 273
 - Associated Type 279
 - Generic Function 274
 - Generic Type 276
 - Type Constraints 278
- Globale Variablen 137
- Golden Master 111

I

- Identity Operators 118
- If 21
- Implementierung 102

- import 284, 292
- init 180
- Initialisierung 179
 - Default Initializer 183
- Initializer 179
 - Convenience Initializer 194
 - Designated Initializer 194
 - Initializer Chaining 195
 - Initializer Delegation 192
 - required 211
- In-Out 76
- Instantvariablen 127
- Int 12
- Interface Builder 296
- Internet 9f.
- Interoperability 292
- Introspection 300
- is 233

J

- Java 1

K

- Klasse 112
 - Initialisierung 114
 - Instanz 113
- Kommentar 33
- Konstante 15

L

- let 16
- Local Parameter Name 69, 142, 145, 185
- Lokale Variablen 137

M

- Mac 4f.
- MARK 34
- Methoden 141
 - Instance Methods 141
 - Klassenmethoden 148
 - mutating 146
 - Mutating Behavior 147
 - Type Methods 141, 148
- Migration 304

- Mix and Match 300
- Objective-C Bridging Header 301
- Xcode-Generated Header 302

N

- Navigation Area 313
- Nested Types 238
- nil 54, 155
- NSArray 42
- NSDictionary 51
- NSNumber 298
- NSString 36

O

- Objective-C 291
- Objekt 112
- Objektorientierte Programmierung 111
- Operator 19
- Optional 54, 116
- Optionals 155
 - Einsatzgebiete 156
 - Forced Unwrapping 157
 - Implicitly Unwrapped Optionals 161
 - Implicit Unwrapping 161
 - Optional Binding 160
 - Optional Chaining 163
- Ownership 215

P

- Parameter 62
 - Beliebige Parameterzahl 74
 - Optionaler Parameter 68
 - Standardwerte für Parameter 72
 - Variablen als Parameter 75
 - Variadic Parameters 74
- Playground 13, 61, 318
- Pointer 118
- println() 11
- Product Module Name 300
- Product Name 300
- Property 120
 - Computed Property 127
 - Constant Stored Property 121
 - Getter 127
 - Lazy Stored Property 122

- Read-Only Computed Property 132
- Setter 127
- Shorthand Setter 132
- Stored Property 121
- Type Property 138
- Variable Stored Property 121
- Property Observer 133
- Protocol 247
 - Class-Only Protocols 268
 - Optionale Eigenschaften 271
 - Protocol Composition 262
- Protocol Conformance 269
- Protocol Type 257
- Python 2

Q

- Quick Look 321

R

- Refactoring 318
- Reference Cycle 215
- Reference Types 117
- Referenz 117
- Ressourcen 5
- Retroactive Modeling 241
- return 64
- Ruby 2
- Rückgabewert 63
 - Mehrere Rückgabewerte 66
 - Optionaler Rückgabewert 67

S

- Schleife 18
- Selectors 295, 300
- self 145
- Semikolon 11
- Sicherheitsprüfungen 198
- Sonderzeichen 324
- Speicherverwaltung 212, 214
 - Best Practices 227
- Stack Overflow 9f.
- String 12, 36
 - Erstellen 37
 - Präfix 40
 - String Interpolation 39

- Suffix 40
- Verändern 37
- Verbinden 37
- Strong Reference 215
- Structure 102
 - Benennung 103
 - Instanzen 103
 - Memberwise Initializers 106, 108
 - Methoden 108
 - Properties 104
- Subscripts 150
 - Read-Only Subscript 150
 - Subscript Overloading 154
- Swift 1
 - Blog 7f.
 - Grundlagen 11
- Swift Type Compatibility 293
- switch
 - Range Matching 30
- Switch 28

T

- Target-Action 295, 300
- TODO 34
- toRaw() 100
- Tuple 58
- Type Annotation 17
- Type Casting 232, 300
- Type Inference 17
- Typsicherheit 41, 50

U

- Unowned Reference 221
- Utilities Area 314

V

- Value Types 101, 109
- var 15
- Variable 15
- Vererbung 170
 - Base Class 173
 - Basisklasse 173
 - Mehrfachvererbung 173
 - override 176
 - Overriding 174
 - Subclass 172
 - Subklasse 172
 - Superclass 171
 - Superklasse 171
- Voraussetzungen 3

W

- Wahrheitswert 12
- Weak Reference 218
- While 23
- WWDC 1

X

- Xcode 3, 307
 - Assistent Editor 318
 - Beta-Versionen 308
 - Installation 307

Z

- Zählvariable 25
- Zeichenkette 12
- Zwei-Phasen-Initialisierung 197