

Swift im Detail



thomas SILLMANN

HANSER

Bleiben Sie auf dem Laufenden!



Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter



www.hanser-fachbuch.de/newsletter



Hanser Update ist der IT-Blog des Hanser Verlags mit Beiträgen und Praxistipps von unseren Autoren rund um die Themen Online Marketing, Webentwicklung, Programmierung, Softwareentwicklung sowie IT- und Projektmanagement. Lesen Sie mit und abonnieren Sie unsere News unter



www.hanser-fachbuch.de/update



Thomas Sillmann

Swift im Detail

HANSER

Der Autor:

Thomas Sillmann, Aschaffenburg

www.thomassillmann.de

Alle in diesem Buch enthaltenen Informationen, Verfahren und Darstellungen wurden nach bestem Wissen zusammengestellt und mit Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Buch enthaltenen Informationen mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor und Verlag übernehmen infolgedessen keine juristische Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieser Informationen – oder Teilen davon – entsteht.

Ebenso übernehmen Autor und Verlag keine Gewähr dafür, dass beschriebene Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Buch berechtigt deshalb auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.



Bibliografische Information der Deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren) – auch nicht für Zwecke der Unterrichtsgestaltung – reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2015 Carl Hanser Verlag München, www.hanser-fachbuch.de

Lektorat: Sieglinde Schär

Copy editing: Sandra Gottmann, Münster-Nienberge

Herstellung: Irene Weilhart

Umschlagdesign: Marc Müller-Bremer, München, www.rebranding.de

Umschlagrealisation: Stephan Rönigk

Gesamtherstellung: Kösel, Krugzell

Printed in Germany

Print-ISBN: 978-3-446-44294-8

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

Für Ela

- auf das, was war, und das, was sein wird.

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.