

Inhalt

Vorwort	15
---------------	----

TEIL I Swift

1 Hello World!	21
1.1 »Hello World« im Playground	21
1.2 »Hello World« als Terminal-App	31
1.3 »Hello World« auf dem iPad	39
2 Swift-Crashkurs	41
2.1 Elementare Syntaxregeln und Kommentare	41
2.2 Variablen, Konstanten und Datentypen	46
2.3 Strukturierung des Codes	50
2.4 Klassen und Datenstrukturen	54
2.5 Fehlerabsicherung	56
2.6 Xcode-Crashkurs	58
3 Operatoren	71
3.1 Zuweisungs- und Rechenoperatoren	71
3.2 Vergleichsoperatoren und logische Operatoren	76
3.3 Range-Operatoren	80
3.4 Operatoren für Fortgeschrittene	84
3.5 Eigene Operatoren	87
4 Variablen und Optionals	93
4.1 Variablen und Konstanten	93
4.2 Optionals	99
4.3 Wert- versus Referenztypen	104

5	Verzweigungen und Schleifen	107
5.1	Verzweigungen mit if	107
5.2	Inverse Logik mit guard	110
5.3	Verzweigungen mit switch	112
5.4	Versions- oder plattformabhängiger Code	115
5.5	Schleifen	117
5.6	Nicht triviale Schleifen	121
6	Funktionen und Closures	127
6.1	Funktionen definieren und ausführen	127
6.2	Parameter	138
6.3	Standardfunktionen	145
6.4	Funktionale Programmierung	149
6.5	Closures	154
7	Zahlen und geometrische Strukturen	165
7.1	Zahlen und boolesche Werte	165
7.2	NSNumber	170
7.3	CGFloat, CGPoint, CGSize und Co.	171
8	Zeichenketten	179
8.1	Syntax	180
8.2	Bearbeitung von Zeichenketten	184
8.3	Suchen und ersetzen	188
8.4	Zeichenketten zerlegen und zusammensetzen	191
8.5	Zeichenketten und Zahlen umwandeln	197
8.6	Zeichenketten und binäre Daten umwandeln (Data-Struktur)	201
9	Datum und Uhrzeit	203
9.1	Datum und Uhrzeit ermitteln und darstellen	203
9.2	Rechnen mit Datum und Uhrzeit	205
10	Arrays, Dictionaries, Sets und Tupel	207
10.1	Arrays	207
10.2	Arrays und Aufzählungen verarbeiten	217

10.3	Dictionaries	226
10.4	Sets	230
10.5	Option-Sets	231
10.6	Tupel	233
10.7	Lottosimulator	235
11	Objektorientierte Programmierung I	241
11.1	Klassen und Strukturen	242
11.2	Enumerationen	250
11.3	Eigenschaften	254
11.4	Init- und Deinit-Funktion	266
11.5	Methoden	271
11.6	Subscripts	278
11.7	Typ-Alias	281
11.8	Speicherverwaltung	282
12	Objektorientierte Programmierung II	287
12.1	Vererbung	287
12.2	Generics	299
12.3	Protokolle	304
12.4	Standardprotokolle	314
12.5	Extensions	322
12.6	Protokollerweiterungen	328
12.7	Reflection und Metatypen	331
13	Fehlerabsicherung	337
13.1	Fehlerabsicherung mit try und catch	337
13.2	Selbst Fehler auslösen (throws und throw)	346
13.3	Fehler in Funktionen weitergeben (rethrows)	350
13.4	Das Error-Protokoll	354
13.5	Fehlerabsicherung von API-Methoden (NSError)	355
14	Importe, Attribute und Systemfunktionen	359
14.1	Module, Frameworks und Importe	359
14.2	Attribute	363
14.3	Systemfunktionen aufrufen	365

TEIL II App-Programmierung

15	Hello iOS-World!	373
15.1	Projektstart	374
15.2	Gestaltung der App	375
15.3	Steuerung der App durch Code	380
15.4	Actions und Outlets für Fortgeschrittene	385
15.5	Layout optimieren	388
15.6	Textgröße mit einem Slider einstellen	395
15.7	Apps auf dem eigenen iPhone/iPad ausführen	398
15.8	Komponenten und Dateien eines Xcode-Projekts	400
16	iOS-Grundlagen	403
16.1	Model-View-Controller (MVC)	403
16.2	Klassenhierarchie einer App-Ansicht	407
16.3	Die UIViewController-Klasse	410
16.4	Phasen einer iOS-App	414
16.5	Auto Layout	417
16.6	Layoutregeln durch Code definieren	433
16.7	Steuerelemente in einer Stack-View anordnen	435
16.8	Layoutvarianten	440
16.9	Texteingaben	448
16.10	Image-Views und Xcassets	455
17	iOS-Apps mit mehreren Ansichten	459
17.1	Storyboard und Controller-Klassen verbinden	459
17.2	Ansichten durch Segues verbinden	460
17.3	Segues mit Datenübertragung	465
17.4	Navigation-Controller	470
17.5	Tab-Bar-Controller	476
17.6	Split-View-Controller	482
17.7	Popups	493
17.8	Ja/Nein-Dialoge (UIAlertController)	503

18	Hello macOS-World!	507
18.1	Von iOS zu macOS	507
18.2	Lottozahlengenerator	509
19	macOS-Grundlagen	519
19.1	Programme mit mehreren Fenstern	519
19.2	Tab-View-Controller	526
19.3	Standarddialoge	533
19.4	Tastatur	538
19.5	Menüs	543
19.6	Programme ohne Menü	552
20	tvOS	555
20.1	Hello tvOS!	556
20.2	Fernbedienung auswerten	561
20.3	Focus Engine	567

TEIL III Programmier- und Arbeitstechniken

21	Dateien und User-Defaults	577
21.1	User-Defaults	577
21.2	Dateinamen und URLs	582
21.3	Bundle-Dateien und Assets	584
21.4	Standardverzeichnisse	586
21.5	Dateioperationen	590
21.6	Wie geht's weiter?	599
22	Netzwerk, XML und JSON	601
22.1	Dateien per HTTP/HTTPS laden	601
22.2	XML-Dokumente auswerten	610
22.3	JSON-Encoder und -Decoder	614
22.4	Webseiten anzeigen	621

23	Tabellen und Listen darstellen	629
23.1	Listen in iOS-Apps (UITableView)	629
23.2	Prototypzellen	635
23.3	Individuelle Gestaltung von Listenzellen	640
23.4	Veränderliche Listen	647
23.5	Tabellen in macOS-Apps (NSTableView)	650
23.6	Collections asynchron füllen (UICollectionView)	661
24	GPS- und Kompassfunktionen	671
24.1	Hello MapView!	671
24.2	Wegstrecke aufzeichnen	676
24.3	Kompassfunktionen	683
25	Grafik und Animation	687
25.1	Eigene Steuerelemente mit Grafikfunktionen	687
25.2	Kompass-Steuerelement	694
25.3	Core Graphics	702
25.4	Animationen	706
26	Touch, Maus, Gestures und Drag & Drop	711
26.1	Touch	711
26.2	Maus	718
26.3	Gestures	725
26.4	Drag & Drop (macOS)	728
26.5	Drag & Drop (iOS)	742
27	Audio, Video und Fotos	757
27.1	Audio-Wiedergabe mit dem AVAudioPlayer	757
27.2	Audio-Wiedergabe mit dem AVPlayer	768
27.3	Audio-Wiedergabe mit dem AVPlayerViewController	770
27.4	Audio-Aufnahmen mit dem AVAudioRecorder durchführen	772
27.5	Videos abspielen	777
27.6	Videos mit der Picker-View auswählen und aufnehmen	781
27.7	YouTube-Videos abspielen	785

27.8	Fotos mit der Picker-View auswählen und aufnehmen	788
27.9	Fotos in einer AVCaptureSession aufnehmen	791
27.10	Barcodes in einer AVCaptureSession erkennen	799
28	SpriteKit	803
28.1	Hello SpriteKit!	804
28.2	Sprites erzeugen und bewegen	813
28.3	Spielsteuerung durch Touch-Ereignisse	819
28.4	Bewegungssteuerung (Gyroskop und Accelerometer)	825
28.5	Aktionen	832
28.6	Der Game-Loop	838
28.7	Kollisionserkennung	840
28.8	Minispiel: Luftballone abschießen	845
28.9	Physik	852
28.10	Minispiel: Pyramide zerstören	858
28.11	Scene-Editor	865
28.12	Partikel-Emitter	872
29	iCloud	875
29.1	iCloud-Grundlagen	875
29.2	Key/Value-Speicher	881
29.3	CloudKit-Grundlagen	888
29.4	CloudKit-Programmiertechniken	896
29.5	CloudKit-Beispiel	906
30	Asynchrone Programmierung	919
30.1	Hello Grand Central Dispatch!	920
30.2	GCD-Grundlagen	923
30.3	Parallel rechnen	928
30.4	Die Async-Bibliothek	935
31	App Store und Co.	937
31.1	iOS-Artwork (Icons, Launch Screen)	938
31.2	macOS-Artwork (Icon)	940
31.3	tvOS-Artwork (Parallax-Icons, Launch und Top Shelf Image)	941

31.4	Mehrsprachige Apps	947
31.5	Eigene Apps im App Store anbieten	956
31.6	macOS-Programme selbst weitergeben	964
32	Xcode-Arbeitstechniken	971
32.1	Simulator-Ausgaben stoppen	971
32.2	Header-Code einer eigenen Klasse erzeugen	972
32.3	Versionsverwaltung mit Git	973
32.4	Crashlogs	976
32.5	Refactoring	977
32.6	Projekte umbenennen	978
32.7	Xcode-Verzeichnisse aufräumen	979
32.8	Apple Configurator	982
33	Server-side Swift	985
33.1	Swift unter Linux	988
33.2	Vapor kennenlernen	995
33.3	Die Vapor-Toolbox	1004
33.4	Vapor-Grundlagen	1006
33.5	Datenbankanbindung mit Fluent	1017
33.6	Authentifizierung und Autorisierung	1028
33.7	Deployment	1036
TEIL IV Beispielprojekte		
<hr/>		
34	New-York-Times-Bestseller	1043
34.1	New-York-Times-API	1045
34.2	Benutzeroberfläche	1048
34.3	Split-View-Variante	1053
35	To-do-Listen	1059
35.1	Gestaltung der Benutzeroberfläche	1060
35.2	Datenmodell	1061
35.3	View-Controller-Klasse	1062

35.4	Popup-View-Controller-Klasse	1069
35.5	iCloud-Variante	1070
36	Schatzsuche	1075
36.1	Aufbau der App	1075
36.2	Datenmodell	1080
36.3	Location Manager selbst gemacht	1082
36.4	Steuerelement zur Richtungsanzeige (UIBezierPath)	1086
36.5	Hauptansicht mit Listenfeld	1087
36.6	Popup-Dialog zum Speichern	1092
36.7	Detailansicht mit Richtungspfeil	1093
37	Währungskalkulator	1101
37.1	App-Überblick	1101
37.2	Kurse ermitteln	1107
37.3	Das Datenmodell der App	1110
37.4	Umrechnungsansicht	1113
37.5	Einstellungsansicht	1119
37.6	Internationalisierung und Lokalisierung	1124
38	Fünf Gewinnt	1127
38.1	Die App »Fünf Gewinnt«	1127
38.2	Enumerationen und globale Funktionen (Globals.swift)	1130
38.3	Die Spiellogik (FiveWins.swift)	1134
38.4	Darstellung des Spielbretts und der Steine (BoardView.swift)	1145
38.5	Steuerung des Spielablaufs (ViewController.swift)	1155
38.6	Der Popup-Dialog (PopupVC.swift)	1160
38.7	Erweiterungsmöglichkeiten	1162
39	Icon-Resizer	1165
39.1	App-Überblick	1165
39.2	Icons verwalten (IconSize-Struktur)	1171
39.3	Hauptfenster (ViewController.swift)	1177
39.4	Drag & Drop-Quelle für Icons (IconCellView.swift)	1185
39.5	Drag & Drop-Empfänger für Icons (OriginalIconView.swift)	1186

39.6	Popup-Menü (IconChoiceVC.swift)	1189
39.7	Temporäres Verzeichnis erstellen und löschen	1190
40	Breakout	1193
40.1	Programmaufbau	1194
40.2	Initialisierung	1195
40.3	Spielsteuerung	1202
41	Pac-Man selbst gemacht	1207
41.1	Programmaufbau	1208
41.2	Der Tile-Editor »Tiled«	1210
41.3	Globale Konstanten, Datenstrukturen und Enumerationen	1214
41.4	Initialisierung des Spiels	1216
41.5	Die Maze-Klasse	1220
41.6	Aufbau der Spielszene (setup-Methoden)	1224
41.7	Spielsteuerung (touch-Methoden)	1230
41.8	Bewegung des Pac-Mans	1235
41.9	Steuerung der Monster	1239
41.10	Kollisionen	1245
41.11	Apple-TV-Variante von Pac-Man	1249
41.12	Pac-Man-Figuren zeichnen	1252
42	Asteroids	1255
42.1	Programmaufbau	1256
42.2	Globale Konstanten und Funktionen	1258
42.3	Programmstart und Tastaturereignisse (GameViewController)	1258
42.4	Initialisierung des Spiels (GameScene)	1261
42.5	Spielablauf (ebenfalls in GameScene)	1267
42.6	Fokussierbare Menütexte (MyLabel)	1273
42.7	Der Einstellungsdialog (MainScene)	1274
	Index	1281

Vorwort

Apples größte Innovation des Jahres 2014 war aus meiner Sicht weder die Vorstellung der Apple Watch noch die Auslieferung des Bestsellers iPhone 6. Apple hat sich neben den Arbeiten an diesen Produkten einer anderen Baustelle zugewandt und als Reaktion auf die vielen Mängel, die die rund 20 Jahre alte Programmiersprache Objective-C aufweist, eine vollkommen neue Programmiersprache entwickelt: Swift!

In ersten Kommentaren konnten selbst Apple-Fans ihre Skepsis nicht verbergen: Brauchen wir wirklich eine neue Programmiersprache? Doch je mehr Details Apple auf der World Wide Developers Conference (WWDC 2014) verrät, desto größer wurde die Begeisterung der teilnehmenden Entwickler und der Fachpresse.

Warum Swift?

Swift ist für Apple ein Befreiungsschlag: Objective-C dient dem Apple-Universum seit vielen Jahren als Fundament. Das ändert aber nichts daran, dass Objective-C eine Programmiersprache aus den 1980er-Jahren ist, die in keinerlei Hinsicht mit modernen Programmiersprachen mithalten kann.

Swift ist dagegen ein sauberer Neuanfang. Bei der Vorstellung wurde Swift auch *Objective-C without the C* genannt. Natürlich ist Swift von Objective-C beeinflusst – schließlich muss Swift kompatibel zu den unzähligen Apple-Bibliotheken sein. Swift realisiert viele neue Ideen, greift aber auch Konzepte von C#, Haskell, Java, Python und anderen Programmiersprachen auf. Daraus ergeben sich mehrere Vorteile:

- ▶ Swift zählt zu den modernsten Programmiersprachen, die es momentan gibt.
- ▶ Code lässt sich in Swift syntaktisch eleganter formulieren als in Objective-C.
- ▶ Der resultierende Code ist besser lesbar und wartbar.
- ▶ Swift ist für Programmierer, die schon Erfahrung mit anderen modernen Sprachen gesammelt haben, wesentlich leichter zu erlernen als Objective-C. Vorhandenes Know-how lässt sich einfacher auf Swift als auf Objective-C übertragen.
- ▶ Swift ist ein Open-Source-Produkt und steht auch für Linux zur Verfügung. Der Entwicklungsprozess erfolgt offen und transparent.

Swift ist in den letzten Jahren kometenhaft in die Top-Listen der populärsten Programmiersprachen aufgestiegen. Im TIOBE-Index war Swift zuletzt auf Platz 11 vertreten.

Neu in Swift 4

Langjährige Swift-Entwickler haben Swift 3 in schlechter Erinnerung: Unzählige inkompatible Neuerungen erforderten umfassende Änderungen an vorhandenem Code. Wesentlich besser sieht es bei Swift 4 aus: Ein Großteil der Neuerungen sind Erweiterungen, die keine Inkompatibilitäten verursachen. Und da, wo doch Code-Änderungen erforderlich sind, kümmert sich in der Regel der Code-Konverter von Xcode um die Anpassungen. Kurzum: Die Anpassung von Swift-3-Code an Swift 4 sollte keine großen Probleme verursachen. Ärger lauert allerdings an anderer Stelle: Veränderte Frameworks und APIs sowie Methoden und Klassen, die plötzlich als *deprecated* gelten, zwingen in manchen Apps zu größeren Umbauten. Das wiederum ist aber nicht die Schuld von Swift.

Was sind nun die wichtigsten Verbesserungen in Swift 4?

- ▶ **Zeichenketten:** Der Umgang mit Zeichenketten ist ein wenig komfortabler geworden: Zum einen lassen sich jetzt sehr elegant mehrzeilige Zeichenketten formulieren, zum anderen ist die ständige Nennung der Eigenschaft überflüssig, um auf die Sequenz der Zeichen einer Zeichenkette zuzugreifen. Die neue `Substring`-Struktur macht die Verarbeitung von Teilzeichenketten effizienter. Weitere Optimierungen wurden im Hintergrund vorgenommen.
- ▶ **Bereiche (Ranges):** Bereiche können nun zur einen oder anderen Seite offen formuliert werden, also in der Form `ar[3...]` (alle Array-Elemente ab dem vierten) oder `s[..endpos]` (die Teilzeichenkette vom Beginn von `s` bis zur Position `endpos`).
- ▶ **Generics:** Die Syntax zur Formulierung generischer Ausdrücke wurde an einigen Punkten erweitert und gibt nun mehr Spielraum als bisher.
- ▶ **JSON-Support:** Die Swift-Standardbibliothek wurde um das Protokoll `Codable` sowie um diverse verwandte Protokolle und Klassen erweitert. Sie machen es möglich, eigene Datentypen mit minimalem Aufwand im JSON-Format zu speichern bzw. aus JSON-Dateien wieder einzulesen.
- ▶ **Xcode:** Der Code-Editor von Xcode wurde komplett neu implementiert. Das merkt man an der höheren Geschwindigkeit, an diversen Instabilitäten, vor allem aber daran, dass es nun endlich Refactoring-Kommandos gibt. Hurra!

Keine Angst vor Swift 5!

Für mich als Autor ist es jedes Jahr ein wenig befremdlich: Ich arbeite mit voller Energie an der Fertigstellung meines Buchs zu Swift 4, da diskutieren die Entwickler bereits über Swift 5! Diese Version soll noch mehr Optimierungen beim Umgang mit Zeichenketten bringen, weitere Generics-Features sowie ein Fundament für asynchrone Programmierkonstrukte. (Deren vollständige Implementierung ist allerdings erst für Swift 6 geplant.)

Generell ist geplant, inkompatible Änderungen auf ein absolutes Minimum zu beschränken. Swift wird also erweitert, aber kaum mehr geändert. Insofern müssen Sie sich nicht vor Swift 5

fürchten: Soweit es wirklich Änderungen gibt, wird der in Xcode integrierte Code-Konverter in bewährter Manier einen Großteil der Änderungen automatisch durchführen.

Außerdem hat sich bei meiner Arbeit an den drei Auflagen dieses Buchs eines herauskristallisiert: Bei der App-Programmierung kostet nicht der Umgang mit Swift an sich Zeit, sondern die Suche nach den geeigneten Klassen, Methoden und Programmieretechniken.

Was bietet dieses Buch?

Dieses Buch vermittelt einen kompakten Einstieg in die Programmiersprache Swift in der Version 4 (Xcode 9). Das Buch ist in vier Teile gegliedert:

- ▶ **Teil I** führt in die Grundlagen von Swift ein. Hier lernen Sie alle wichtigen Sprachdetails kennen. Die Themenpalette reicht vom Umgang mit Variablen und elementaren Datentypen bis hin zur Syntax der objekt- und protokollorientierten Programmierung.
- ▶ **Teil II** ist eine Einführung in die Entwicklung von Apps für iOS, macOS und tvOS. Hier erkläre ich Ihnen beispielsweise, wie der Storyboard-Editor funktioniert, wie Sie Ihre Oberfläche mit eigenem Swift-Code verbinden, eigene `ViewController`-Klassen entwickeln, Apps mit mehreren Dialogen/Views organisieren etc.
- ▶ **Teil III** fasst wichtige Programmieretechniken in Bausteinform zusammen. In Kurzanleitungen zeige ich Ihnen unter anderem, wie Sie auf Dateien zugreifen, XML-Dokumente auswerten, Webseiten anzeigen, Steuerelemente mit eigener Grafik gestalten, Listen und Tabellen in Apps darstellen, geografische Daten auswerten und Spiele mit `SpriteKit` programmieren. Sobald Ihre App zufriedenstellend funktioniert, lernen Sie, wie Sie sie tauglich für den App Store machen und dort einreichen.
- ▶ **Teil IV** zeigt anhand konkreter Beispielprojekte die Praxis der App-Programmierung. Die Apps decken eine ganze Palette von Themen ab: vom praktischen Währungsumrechner über den Icon-Resizer bis hin zu mehreren Spielen.

Neu in dieser Auflage sind nicht nur Swift-4-Features. Weitere inhaltliche Schwerpunkte sind Drag & Drop für iOS, die iCloud-Programmierung sowie Server-side Swift.

Selbstverständlich können Sie alle Beispieldateien und -projekte dieses Buchs herunterladen. Einen Download-Link finden Sie hier:

www.rheinwerk-verlag.de/4494

Um von diesem Buch maximal zu profitieren, benötigen Sie weder Vorkenntnisse in Xcode noch in der App-Entwicklung. Ich setze aber voraus, dass Sie bereits Erfahrungen mit einer beliebigen Programmiersprache gesammelt haben. Ich erkläre Ihnen in diesem Buch also, wie Sie in Swift mit Variablen umgehen, Schleifen programmieren und Klassen entwickeln, aber nicht, was Variablen sind, wozu Schleifen dienen und warum Klassen das Fundament der objektorientierten Programmierung sind. So kann ich Swift kompakt und ohne viel Overhead beschreiben und den Schwerpunkt auf die konkrete Anwendung legen.

Lesanleitung

1300 Seiten – das kann schon abschrecken! Dazu besteht aber kein Grund. Ich habe mich beim Schreiben dieses Buchs bemüht, den Inhalt auf möglichst eigenständige Kapitel zu verteilen, aus denen Sie sich wie aus einem Baukasten bedienen können.

Wenn Swift für Sie vollständig neu ist, dann ist die Lektüre der ersten Kapitel aus Teil I natürlich unumgänglich. Besonders wichtig ist, dass Sie die Swift-spezifischen Eigenheiten beim Umgang mit elementaren Datentypen und Aufzählungen (Arrays, Dictionaries etc.) kennenlernen und das Konzept von Optionals verstehen. Interessanterweise hat sich herausgestellt, dass Sie für die Entwicklung erster Apps nicht unbedingt alle Feinheiten im Zusammenhang mit Vererbung, Protokollen etc. beherrschen müssen. Die Basics reichen zumeist.

Teil II richtet sich speziell an Programmierer, die erstmalig Apps für iOS, macOS oder tvOS entwickeln. Wenn Sie bisher Objective-C zur App-Programmierung verwendet haben, werden Sie in Teil II auf viel bekanntes Wissen stoßen.

Bei Teil III habe ich versucht, oft benötigte Programmier Techniken möglichst losgelöst von der Zielplattform zu beschreiben. Beispielsweise erfolgt der Umgang mit Dateien unter iOS ganz ähnlich wie unter macOS. Aus den Kapiteln in Teil III können Sie sich also bedienen, wie Sie es gerade brauchen.

Viele Detailprobleme treten erst dann auf, wenn man den Schritt von kleinen Beispielen hin zu »richtigen« Apps macht. Deswegen stellt Teil IV eine Reihe vollständiger Projekte vor. Auch wenn es unwahrscheinlich ist, dass Sie genau so eine App programmieren möchten wie eines der Beispiele aus Teil IV, so werden Sie in diesen Kapiteln vermutlich doch inhaltlich verwandte Anleitungen und Arbeitstechniken mit einem hohen Praxisbezug finden. Probieren Sie die Apps einfach einmal aus, und blättern Sie dann durch die entsprechenden Kapitel – Sie werden sicher über Details stolpern, die sich später als hilfreich herausstellen werden.

Viel Spaß bei der App-Entwicklung!

Eine neue Programmiersprache zu erlernen ist immer eine Herausforderung. Noch schwieriger ist es, einen Überblick über die schier unüberschaubare Fülle von Bibliotheken zu gewinnen, die Sie zur App-Entwicklung brauchen. Dieses Buch soll Ihnen bei beiden Aspekten helfen und Ihnen ein solides Fundament vermitteln.

Wenn Sie in die App-Entwicklung mit Swift einsteigen, haben Sie das Privileg, mit einer der modernsten aktuell verfügbaren Programmiersprachen zu arbeiten. Sobald Sie die ersten Schritte einmal erfolgreich absolviert haben, wird die Faszination für diese Sprache auch Sie erfassen. Bei Ihrer Reise durch die neue Welt von Swift wünsche ich Ihnen viel Spaß und Erfolg!

Michael Kofler (<https://kofler.info>)

PS: Vielen Dank an Alfred Schilken für sein Feedback zum Manuskript!