

# Inhalt

Vorwort .....	17
---------------	----

## TEIL I Swift

---

<b>1 Hello World!</b> .....	23
1.1 »Hello World« im Playground .....	23
1.2 »Hello World« als Terminal-App .....	33
<b>2 Swift-Crashkurs</b> .....	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
<b>3 Operatoren</b> .....	69
3.1 Zuweisungs- und Rechenoperatoren .....	69
3.2 Vergleichsoperatoren und logische Operatoren .....	74
3.3 Range-Operatoren .....	78
3.4 Operatoren für Fortgeschrittene .....	82
3.5 Eigene Operatoren .....	85
<b>4 Variablen und Optionals</b> .....	89
4.1 Variablen und Konstanten .....	89
4.2 Optionals .....	95
4.3 Wert- versus Referenztypen .....	100

<b>5</b>	<b>Verzweigungen und Schleifen</b> .....	103
5.1	Verzweigungen mit if .....	103
5.2	Inverse Logik mit guard .....	106
5.3	Verzweigungen mit switch .....	108
5.4	Versions- oder plattformabhängiger Code .....	111
5.5	Schleifen .....	112
5.6	Nicht triviale Schleifen .....	117
<b>6</b>	<b>Funktionen und Closures</b> .....	123
6.1	Funktionen definieren und ausführen .....	123
6.2	Parameter .....	134
6.3	Standardfunktionen .....	141
6.4	Funktionale Programmierung .....	145
6.5	Closures .....	150
<b>7</b>	<b>Zahlen und geometrische Strukturen</b> .....	161
7.1	Zahlen und boolesche Werte .....	161
7.2	NSNumber .....	166
7.3	CGFloat, CGPoint, CGSize und Co. ....	167
<b>8</b>	<b>Zeichenketten</b> .....	173
8.1	Syntax .....	173
8.2	Bearbeitung von Zeichenketten .....	179
8.3	Suchen und ersetzen .....	183
8.4	Zeichenketten zerlegen und zusammensetzen .....	186
8.5	Zeichenketten und Zahlen umwandeln .....	192
8.6	Zeichenketten und binäre Daten umwandeln (Data-Struktur) .....	195
<b>9</b>	<b>Datum und Uhrzeit</b> .....	197
9.1	Datum und Uhrzeit ermitteln und darstellen .....	197
9.2	Rechnen mit Datum und Uhrzeit .....	199

<b>10</b>	<b>Arrays, Dictionaries, Sets und Tupel</b>	201
10.1	Arrays	201
10.2	Arrays und Aufzählungen verarbeiten	211
10.3	Dictionaries	221
10.4	Sets	225
10.5	Option-Sets	226
10.6	Tupel	228
10.7	Lottosimulator	230
<b>11</b>	<b>Objektorientierte Programmierung I</b>	235
11.1	Klassen und Strukturen	236
11.2	Enumerationen	244
11.3	Eigenschaften	251
11.4	Init- und Deinit-Funktion	263
11.5	Methoden	269
11.6	Subscripts	276
11.7	Typ-Aliasse	279
11.8	Speicherverwaltung	279
<b>12</b>	<b>Objektorientierte Programmierung II</b>	285
12.1	Vererbung	285
12.2	Generics	297
12.3	Protokolle	302
12.4	Standardprotokolle	312
12.5	Extensions	320
12.6	Protokollerweiterungen	325
12.7	Reflection und Metatypen	328
<b>13</b>	<b>Fehlerabsicherung</b>	335
13.1	Fehlerabsicherung mit try und catch	335
13.2	Selbst Fehler auslösen (throws und throw)	344
13.3	Fehler in Funktionen weitergeben (rethrows)	348
13.4	Das Error-Protokoll	351
13.5	Fehlerabsicherung von API-Methoden (NSError)	353
13.6	Result-Datentyp	355

<b>14</b>	<b>Importe, Attribute und Systemfunktionen</b>	361
14.1	Module, Frameworks und Importe	361
14.2	Attribute	365
14.3	Systemfunktionen aufrufen	367

## TEIL II App-Programmierung

---

<b>15</b>	<b>Hello iOS-World!</b>	375
15.1	Projektstart	376
15.2	Gestaltung der App	377
15.3	Steuerung der App durch Code	381
15.4	Actions und Outlets für Fortgeschrittene	386
15.5	Layout optimieren	389
15.6	Textgröße mit einem Slider einstellen	394
15.7	Apps auf dem eigenen iPhone oder iPad ausführen	396
15.8	Komponenten und Dateien eines Xcode-Projekts	399
<b>16</b>	<b>iOS-Grundlagen</b>	401
16.1	Model-View-Controller (MVC)	401
16.2	Klassenhierarchie einer App-Ansicht	405
16.3	Die UIViewController-Klasse	407
16.4	Phasen einer iOS-App	411
16.5	Auto Layout	414
16.6	Layoutregeln durch Code definieren	429
16.7	Steuerelemente in einer Stack-View anordnen	431
16.8	Layoutvarianten	435
16.9	Texteingaben	442
16.10	Image-Views und Xcassets	450
<b>17</b>	<b>iOS-Apps mit mehreren Ansichten</b>	453
17.1	Storyboard und Controller-Klassen verbinden	453
17.2	Ansichten durch Segues verbinden	454
17.3	Segues mit Datenübertragung	459
17.4	Navigation-Controller	464

17.5	Tab-Bar-Controller .....	469
17.6	Split-View-Controller .....	475
17.7	Popups .....	485
17.8	Ja/Nein-Dialoge (UIAlertController) .....	496
<b>18</b>	<b>Hello macOS-World!</b> .....	<b>499</b>
18.1	Lottozahlengenerator .....	500
<b>19</b>	<b>macOS-Grundlagen</b> .....	<b>509</b>
19.1	Programme mit mehreren Fenstern .....	509
19.2	Tab-View-Controller .....	516
19.3	Standarddialoge .....	523
19.4	Tastatur .....	528
19.5	Menüs .....	533
19.6	Programme ohne Menü .....	542
19.7	Dunkles Erscheinungsbild (»Dark Mode«) .....	544
<b>20</b>	<b>tvOS</b> .....	<b>547</b>
20.1	Hello tvOS! .....	548
20.2	Fernbedienung auswerten .....	552
20.3	Focus Engine .....	558

## TEIL III Programmier- und Arbeitstechniken

---

<b>21</b>	<b>Dateien und User-Defaults</b> .....	<b>567</b>
21.1	User-Defaults .....	567
21.2	Dateinamen und URLs .....	571
21.3	Bundle-Dateien und Assets .....	573
21.4	Standardverzeichnisse .....	575
21.5	Dateioperationen .....	580
21.6	Wie geht's weiter? .....	587
<b>22</b>	<b>Netzwerk, XML und JSON</b> .....	<b>589</b>
22.1	Dateien per HTTP/HTTPS laden .....	589
22.2	XML-Dokumente auswerten .....	597

22.3	JSON-Encoder und -Decoder .....	601
22.4	Webseiten anzeigen .....	608
<b>23</b>	<b>Tabellen und Listen darstellen .....</b>	<b>615</b>
23.1	Listen in iOS-Apps (UITableView) .....	615
23.2	Prototypzellen .....	621
23.3	Individuelle Gestaltung von Listenzellen .....	626
23.4	Veränderliche Listen .....	632
23.5	Tabellen in macOS-Apps (NSTableView) .....	634
23.6	Collections asynchron füllen (UICollectionView) .....	645
<b>24</b>	<b>GPS- und Kompassfunktionen .....</b>	<b>653</b>
24.1	Hello Map-View! .....	653
24.2	Wegstrecke aufzeichnen .....	658
24.3	Kompassfunktionen .....	665
<b>25</b>	<b>Grafik und Animation .....</b>	<b>667</b>
25.1	Eigene Steuerelemente mit Grafikfunktionen .....	667
25.2	Kompass-Steuerelement .....	673
25.3	Core Graphics .....	682
25.4	Animationen .....	685
<b>26</b>	<b>Touch, Maus, Gestures und Drag &amp; Drop .....</b>	<b>691</b>
26.1	Touch .....	691
26.2	Maus .....	698
26.3	Gestures .....	705
26.4	Drag & Drop (macOS) .....	708
26.5	Drag & Drop (iOS) .....	721
<b>27</b>	<b>Audio, Video und Fotos .....</b>	<b>735</b>
27.1	Audiowiedergabe mit dem AVAudioPlayer .....	735
27.2	Audiowiedergabe mit dem AVPlayer .....	745
27.3	Audiowiedergabe mit dem AVPlayerViewController .....	747
27.4	Audioaufnahmen mit dem AVAudioRecorder durchführen .....	749

27.5	Videos abspielen .....	754
27.6	Videos mit der Picker-View auswählen und aufnehmen .....	757
27.7	YouTube-Videos abspielen .....	761
27.8	Fotos mit der Picker-View auswählen und aufnehmen .....	764
27.9	Fotos in einer AVCaptureSession aufnehmen .....	766
27.10	Barcodes in einer AVCaptureSession erkennen .....	774
<b>28</b>	<b>SpriteKit</b> .....	<b>779</b>
28.1	Hello SpriteKit! .....	780
28.2	Sprites erzeugen und bewegen .....	789
28.3	Spielsteuerung durch Touch-Ereignisse .....	794
28.4	Bewegungssteuerung (Gyroskop und Accelerometer) .....	800
28.5	Aktionen .....	806
28.6	Der Game-Loop .....	812
28.7	Kollisionserkennung .....	814
28.8	Minispiel: Luftballone abschießen .....	820
28.9	Physik .....	826
28.10	Minispiel: Pyramide zerstören .....	832
28.11	Scene-Editor .....	838
28.12	Partikel-Emitter .....	844
<b>29</b>	<b>Core Data und SQLite</b> .....	<b>847</b>
29.1	Hello, Core Data! .....	848
29.2	Core-Data-Klassenüberblick .....	853
29.3	Core-Data-Entities verknüpfen (Relationships) .....	855
29.4	Core-Data-Interna .....	865
29.5	SQLite.swift .....	868
<b>30</b>	<b>iCloud</b> .....	<b>879</b>
30.1	iCloud-Grundlagen .....	879
30.2	Key-Value-Speicher .....	885
30.3	CloudKit-Grundlagen .....	892
30.4	CloudKit-Programmiertechniken .....	899
30.5	CloudKit-Beispiel .....	908

<b>31</b>	<b>Asynchrone Programmierung</b>	921
31.1	Hello Grand Central Dispatch!	922
31.2	GCD-Grundlagen	925
31.3	Parallel rechnen	930
<b>32</b>	<b>App Store und Co.</b>	937
32.1	iOS-Artwork (Icons, Launch Screen)	938
32.2	macOS-Artwork (Icon)	940
32.3	tvOS-Artwork (Parallax-Icons, Launch und Top Shelf Image)	941
32.4	Mehrsprachige Apps	947
32.5	Eigene Apps im App Store anbieten	961
32.6	macOS-Programme selbst weitergeben	970
<b>33</b>	<b>Xcode-Arbeitstechniken</b>	979
33.1	Simulator-Ausgaben stoppen	979
33.2	Header-Code einer eigenen Klasse erzeugen	980
33.3	Versionsverwaltung mit Git	981
33.4	Crashlogs	984
33.5	Refactoring	984
33.6	Projekte umbenennen	986
33.7	Xcode-Verzeichnisse aufräumen	987
33.8	Apple Configurator	989
<b>34</b>	<b>Server-side Swift</b>	991
34.1	Swift unter Linux	994
34.2	Vapor kennenlernen	1000
34.3	Die Vapor-Toolbox	1010
34.4	Vapor-Grundlagen	1012
34.5	Datenbankanbindung mit Fluent	1024
34.6	Authentifizierung und Autorisierung	1037
34.7	Deployment	1045

## TEIL IV Beispielprojekte

---

<b>35</b>	<b>New-York-Times-Bestseller</b> .....	1053
35.1	New-York-Times-API .....	1055
35.2	Benutzeroberfläche .....	1058
35.3	Split-View-Variante .....	1062
<b>36</b>	<b>To-do-Listen</b> .....	1067
36.1	Gestaltung der Benutzeroberfläche .....	1068
36.2	Datenmodell .....	1069
36.3	View-Controller-Klasse .....	1070
36.4	Popup-View-Controller-Klasse .....	1077
36.5	iCloud-Variante .....	1078
36.6	Core-Data-Variante .....	1083
36.7	SQLite-Variante .....	1087
<b>37</b>	<b>Schatzsuche</b> .....	1097
37.1	Aufbau der App .....	1097
37.2	Datenmodell .....	1101
37.3	Location Manager selbst gemacht .....	1104
37.4	Steuerelement zur Richtungsanzeige (UIBezierPath) .....	1108
37.5	Hauptansicht mit Listenfeld .....	1109
37.6	Popup-Dialog zum Speichern .....	1114
37.7	Detailansicht mit Richtungspfeil .....	1115
<b>38</b>	<b>Währungskalkulator</b> .....	1121
38.1	App-Überblick .....	1121
38.2	Kurse ermitteln .....	1127
38.3	Das Datenmodell der App .....	1129
38.4	Umrechnungsansicht .....	1132
38.5	Einstellungsansicht .....	1138
38.6	Internationalisierung und Lokalisierung .....	1143

<b>39</b>	<b>Fünf gewinnt</b> .....	1145
39.1	Die App »Fünf gewinnt« .....	1145
39.2	Enumerationen und globale Funktionen (Globals.swift) .....	1148
39.3	Die Spiellogik (FiveWins.swift) .....	1152
39.4	Darstellung des Spielbretts und der Steine (BoardView.swift) .....	1162
39.5	Steuerung des Spielablaufs (ViewController.swift) .....	1171
39.6	Der Popup-Dialog (PopupVC.swift) .....	1176
39.7	Erweiterungsmöglichkeiten .....	1179
<b>40</b>	<b>Puzzle-App</b> .....	1181
40.1	Programmaufbau .....	1182
40.2	Die Klassen »Tile« und »Puzzle« .....	1185
40.3	Verwaltung der Benutzeroberfläche im View-Controller .....	1190
<b>41</b>	<b>Icon-Resizer</b> .....	1195
41.1	App-Überblick .....	1195
41.2	Icons verwalten (IconSize-Struktur) .....	1201
41.3	Hauptfenster (ViewController.swift) .....	1206
41.4	Drag-and-Drop-Quelle für Icons (IconCellView.swift) .....	1213
41.5	Drag-and-Drop-Empfänger für Icons (OriginalIconView.swift) .....	1215
41.6	Popup-Menü (IconChoiceVC.swift) .....	1217
41.7	Temporäres Verzeichnis erstellen und löschen .....	1218
<b>42</b>	<b>Breakout</b> .....	1221
42.1	Programmaufbau .....	1222
42.2	Initialisierung .....	1223
42.3	Spielsteuerung .....	1230
<b>43</b>	<b>Pac-Man selbst gemacht</b> .....	1233
43.1	Programmaufbau .....	1234
43.2	Der Tile-Editor »Tiled« .....	1235
43.3	Globale Konstanten, Datenstrukturen und Enumerationen .....	1240
43.4	Initialisierung des Spiels .....	1242
43.5	Die Maze-Klasse .....	1245

43.6	Aufbau der Spielszene (setup-Methoden) .....	1249
43.7	Spielsteuerung (touch-Methoden) .....	1255
43.8	Bewegung des Pac-Mans .....	1260
43.9	Steuerung der Monster .....	1264
43.10	Kollisionen .....	1269
43.11	Apple-TV-Variante von Pac-Man .....	1273
43.12	Pac-Man-Figuren zeichnen .....	1277
<b>44</b>	<b>Asteroids</b> .....	<b>1279</b>
44.1	Programmaufbau .....	1280
44.2	Globale Konstanten und Funktionen .....	1282
44.3	Programmstart und Tastaturereignisse (GameViewController) .....	1282
44.4	Initialisierung des Spiels (GameScene) .....	1285
44.5	Spielablauf (ebenfalls in GameScene) .....	1291
44.6	Fokussierbare Menütexpte (MyLabel) .....	1296
44.7	Der Einstellungsdialog (MainScene) .....	1297
	Index .....	1303



# Vorwort

Als Apple 2014 die neue Programmiersprache Swift vorstellte, fragten sich viele Entwickler: Brauchen wir wirklich eine neue Programmiersprache? Mittlerweile erübrigt sich die Frage. Swift ist der De-facto-Standard für neue Projekte im Apple-Universum. Objective-C ist damit nicht obsolet (vermutlich gibt es Milliarden Zeilen Code, der weiter gewartet werden muss), aber wer es sich aussuchen darf, wird neue Apps mit Swift entwickeln.

## Warum Swift?

Swift ist für Apple ein Befreiungsschlag: Objective-C ist zwar noch immer das Fundament von macOS, iOS etc. – aber das ändert nichts daran, dass diese Sprache in den 1980er-Jahren entworfen wurde und 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 mit 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.

## Neu in Swift 5

Swift 5 wurde von Apple als die Version postuliert, mit der Swift gleichsam »erwachsen« wird. Apple garantiert mit Swift 5 die ABI-Stabilität. Das *Application Binary Interface* (ABI) beschreibt die Schnittstelle zwischen Programmen bzw. Bibliotheken auf binärer Ebene, also zur Laufzeit. Im Unterschied dazu betrifft das *Application Programming Interface* (API) »nur« den Quellcode.

Das Erreichen der ABI-Stabilität bedeutet, dass sich ein mit Swift kompiliertes Programm darauf verlassen kann, dass das Zusammenspiel mit einer zu einem anderen Zeitpunkt und von einem anderen Entwickler kompilierten Bibliothek funktioniert — auch dann, wenn die App mit Swift 5.0 kompiliert ist, die (z. B. unter iOS vorinstallierte) Bibliothek aber bereits mit Swift 5.1.

Das Erreichen der ABI-Stabilität hat insofern große praktische Auswirkungen, als Swift-Bibliotheken nun direkt als Teil von iOS, macOS usw. ausgeliefert werden. In Swift entwickelte Apps können sich ab jetzt darauf verlassen, dass die Standardbibliotheken auf der Zielplattform zur Verfügung stehen. Es ist nicht mehr nötig, mit jeder App diverse Bibliotheken mitzuliefern. Das macht Kompilate kleiner – und natürlich auch die daraus resultierenden Apps.

Davon losgelöst bietet Swift 5 natürlich eine Menge Neuerungen im Vergleich zu Swift 4. (Die folgende Aufzählung berücksichtigt auch Features, die bereits in Swift 4.1 oder 4.2 eingeführt wurden.)

- ▶ **Zeichenketten:** Swift unterstützt jetzt *Raw*-Zeichenketten, also Zeichenketten, die Sonderzeichen wie `\` enthalten.
- ▶ **Umgang mit Fehlern:** Die Swift-Standardbibliothek enthält den neuen Datentyp `Result`, der dabei hilft, Ergebnisse einer Funktion oder Methode und eventuell aufgetretene Fehler gemeinsam zurückzugeben. Das ermöglicht einen einfacheren Umgang mit Fehlern, insbesondere in asynchronem Code.

Eine praktische Verbesserung ist das *Optional Flattening* des Schlüsselworts `try?`: Es verhindert eine Verschachtelung von Optionals.

- ▶ **Aufzählungen:** Die Swift-Standardbibliothek wurde um diverse Methode ergänzt, die die Verarbeitung von Arrays, Dictionaries und anderen Aufzählungen vereinfacht. Dazu zählen `allSatisfy`, `compactMap`, `compactMapValues`, `count(where:)` und `removeAll`.
- ▶ **Zufallszahlen:** Swift 5 stellt endlich konsistente Methoden zum Erzeugen von Zufallszahlen zur Verfügung. `Int.random(in: 1...10)` liefert beispielsweise zufällige ganze Zahlen zwischen 1 und 10.

Auch praktisch: `randomElement` liefert ein zufälliges Element eines Sets, eines Arrays bzw. ein zufälliges Zeichen aus einer Zeichenkette.

- ▶ **Equatable und Hashable:** Selbst definierte Strukturen und Enumerationen erfüllen jetzt in den meisten Fällen automatisch die Protokolle `Equatable` und `Hashable`. Selbst wenn dieser Automatismus nicht greift (z. B. bei eigenen Klassen), ist die Implementierung der `hash`-Methode aufgrund der neuen Methode `Hasher.combine` einfacher denn je.
- ▶ **Dynamische Eigenschaften:** Das neue Attribut `@dynamicMemberLookup` gibt Ihnen die Möglichkeit, Typen zu definieren, auf deren Daten Sie in der Form `obj.name [= ...]` zugreifen können, obwohl die Eigenschaft `name` gar nicht statisch definiert ist.

## Was bietet dieses Buch?

Dieses Buch vermittelt einen kompakten Einstieg in die Programmiersprache Swift in der Version 5 (Xcode 10.2). 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 Programmiertechniken 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. Sie lernen, wie Sie Ihre App, sobald sie zufriedenstellend funktioniert, 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-5-Features, sondern auch neue Beispiel-Apps sowie ein neues Kapitel zu Core Data und SQLite.

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

[www.rheinwerk-verlag.de/4749](http://www.rheinwerk-verlag.de/4749)

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.

## Leseanleitung

1.300 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 Programmiertechniken 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>)