

Apps für iOS 10 professionell entwickeln

Sauberer Code schreiben mit Swift 3 und Objective-C. Stabile Apps für iPhone und iPad programmieren.
Techniken & Methoden von Grund auf verstehen

Bearbeitet von
Thomas Sillmann

1. Auflage 2016. Buch inkl. Online-Nutzung. 808 S. Softcover

ISBN 978 3 446 45073 8

Format (B x L): 17,4 x 24,6 cm

Gewicht: 1649 g

[Weitere Fachgebiete > EDV, Informatik > Betriebssysteme](#)

schnell und portofrei erhältlich bei



Die Online-Fachbuchhandlung beck-shop.de ist spezialisiert auf Fachbücher, insbesondere Recht, Steuern und Wirtschaft. Im Sortiment finden Sie alle Medien (Bücher, Zeitschriften, CDs, eBooks, etc.) aller Verlage. Ergänzt wird das Programm durch Services wie Neuerscheinungsdienst oder Zusammenstellungen von Büchern zu Sonderpreisen. Der Shop führt mehr als 8 Millionen Produkte.

HANSER



Leseprobe

Thomas Sillmann

Apps für iOS 10 professionell entwickeln

Sauberen Code schreiben mit Swift 3 und Objective-C. Stabile Apps für iPhone und iPad programmieren. Techniken & Methoden von Grund auf verstehen

ISBN (Buch): 978-3-446-45073-8

ISBN (E-Book): 978-3-446-45206-0

Weitere Informationen oder Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-45073-8>

sowie im Buchhandel.

Inhalt

| | |
|--|------------|
| Vorwort | XXI |
| Danksagung | XXV |
| 1 Über iOS | 1 |
| 1.1 Was ist iOS? | 1 |
| 1.1.1 iOS und macOS | 2 |
| 1.1.2 Besonderheiten der iOS-Plattform | 3 |
| 1.2 iOS für Entwickler | 4 |
| 1.2.1 Hardware für Entwickler | 4 |
| 1.2.2 Software für Entwickler | 6 |
| 1.2.3 Das Apple Developer Program | 6 |
| 1.3 Der Aufbau von iOS | 8 |
| 1.3.1 Die vier Schichten von iOS | 8 |
| 1.4 Die perfekte iOS-App | 10 |
| 1.4.1 iOS Human Interface Guidelines | 11 |
| 2 Die (bisherige) Programmiersprache – Objective-C | 13 |
| 2.1 Über Objective-C und objektorientierte Programmierung | 13 |
| 2.2 Grundlagen der Programmierung | 14 |
| 2.2.1 Objekte | 14 |
| 2.2.2 Primitive Datentypen | 14 |
| 2.2.3 Variablen | 15 |
| 2.2.4 Operatoren | 17 |
| 2.2.5 Abfragen und Schleifen | 17 |
| 2.2.6 Kommentare | 22 |
| 2.3 Aufbau einer Klasse | 23 |
| 2.3.1 Die Header-Datei | 23 |
| 2.3.2 Die Implementation-Datei | 25 |
| 2.3.3 Los geht's: Unsere erste Klasse! | 26 |

| | | |
|--------|---|----|
| 2.4 | Methoden | 30 |
| 2.4.1 | Aufbau von Methoden | 30 |
| 2.4.2 | Methoden in Header- und Implementation-Dateien einer Klasse | 32 |
| 2.4.3 | Implementierung von Methoden | 34 |
| 2.4.4 | Methoden aufrufen | 36 |
| 2.4.5 | Klassen- und Instanzmethoden | 37 |
| 2.5 | Instanzvariablen | 38 |
| 2.6 | Properties | 40 |
| 2.6.1 | Aufbau einer Property | 40 |
| 2.6.2 | Die Punktnotation | 42 |
| 2.6.3 | Optionen | 43 |
| 2.6.4 | Direktzugriff auf Properties | 45 |
| 2.6.5 | Setter und Getter überschreiben | 47 |
| 2.7 | Konstanten | 49 |
| 2.7.1 | Deklaration von Konstanten | 50 |
| 2.8 | Namenskonventionen | 51 |
| 2.8.1 | Klassen | 51 |
| 2.8.2 | Methoden | 51 |
| 2.8.3 | Properties | 52 |
| 2.9 | Strukturen | 52 |
| 2.9.1 | enum | 52 |
| 2.9.2 | typedef | 53 |
| 2.10 | Initialisierung von Objekten | 54 |
| 2.10.1 | alloc und init | 55 |
| 2.10.2 | Zeiger | 57 |
| 2.11 | init im Detail | 58 |
| 2.11.1 | Erstellen eigener init-Methoden | 60 |
| 2.11.2 | Designated Initializer | 61 |
| 2.12 | Vererbung | 63 |
| 2.12.1 | Methoden der Superklasse überschreiben | 65 |
| 2.13 | Kategorien | 67 |
| 2.13.1 | Aufbau von Kategorien | 67 |
| 2.13.2 | Kategorien in Xcode erstellen | 68 |
| 2.13.3 | Verwenden von Kategorien | 70 |
| 2.14 | Erweiterungen | 70 |
| 2.14.1 | Aufbau von Erweiterungen | 71 |
| 2.14.2 | Erweiterungen innerhalb der Implementation-Datei | 71 |
| 2.14.3 | Erweiterungen in Xcode erstellen | 72 |
| 2.15 | Protokolle | 73 |
| 2.15.1 | Aufbau von Protokollen | 74 |
| 2.15.2 | Zuweisen eines Protokolls zu einer Klasse | 75 |
| 2.15.3 | Vererbung in Protokollen | 76 |
| 2.15.4 | Protokolle in Xcode erstellen | 76 |

| | |
|--|-----------|
| 2.16 #import und @class | 78 |
| 2.16.1 #import | 78 |
| 2.16.2 @class | 79 |
| 2.17 Blöcke | 80 |
| 2.17.1 Was sind Blöcke? | 81 |
| 2.17.2 Aufbau eines Blocks | 81 |
| 2.17.3 Zuweisen eines Blocks zu einer Variablen | 82 |
| 2.17.4 Nutzen eines Blocks als Parameter einer Methode | 83 |
| 2.17.5 Blöcke als Properties | 85 |
| 2.17.6 Blockvariablen | 85 |
| 2.17.7 Globale Blöcke | 86 |
| 2.18 Singletons | 87 |
| 3 Der Neue im Club – Swift | 89 |
| 3.1 Programmierst du noch oder swifst du schon? | 89 |
| 3.1.1 Über Swift | 89 |
| 3.1.2 Voraussetzungen zur Nutzung von Swift | 90 |
| 3.1.3 Swift und Objective-C | 90 |
| 3.1.4 Playgrounds | 91 |
| 3.2 Grundlagen der Programmierung | 93 |
| 3.2.1 Swift Standard Library und Fundamental Types | 93 |
| 3.2.2 Variablen und Konstanten | 95 |
| 3.2.3 Operatoren | 97 |
| 3.2.4 Abfragen und Schleifen | 98 |
| 3.2.5 Kommentare | 105 |
| 3.2.6 print | 106 |
| 3.3 Fundamental Types und Swift Standard Library im Detail | 108 |
| 3.3.1 Strings | 108 |
| 3.3.2 Arrays | 111 |
| 3.3.3 Dictionaries | 116 |
| 3.3.4 Any und AnyObject | 120 |
| 3.4 Aufbau einer Klasse | 120 |
| 3.4.1 Erstellen einer Instanz einer Klasse | 123 |
| 3.4.2 Zugriff auf Eigenschaften einer Klasse | 123 |
| 3.5 Methoden | 124 |
| 3.5.1 Methoden mit Rückgabewert | 125 |
| 3.5.2 Methoden mit Parametern | 126 |
| 3.5.3 Local und External Parameter Names | 129 |
| 3.5.4 Methodennamen in Swift | 131 |
| 3.5.5 Aufruf von Methoden einer Klasse | 132 |
| 3.5.6 Zugriff auf andere Eigenschaften und Methoden einer Klasse | 133 |
| 3.5.7 Klassen- und Instanzmethoden | 134 |
| 3.5.8 Verändern von Parametern einer Methode mittels inout | 136 |

| | | |
|----------|--|------------|
| 3.6 | Closures | 137 |
| 3.6.1 | Closures als Variablen und Konstanten | 139 |
| 3.6.2 | Closures als Parameter für Methoden | 140 |
| 3.6.3 | Kurzschreibweise für Closures als Parameter von Methoden | 143 |
| 3.7 | Properties | 146 |
| 3.7.1 | Computed Properties | 147 |
| 3.7.2 | Property Observers | 150 |
| 3.7.3 | Type Properties | 151 |
| 3.8 | Vererbung | 152 |
| 3.8.1 | Überschreiben von Eigenschaften und Methoden der Superklasse | 154 |
| 3.8.2 | Zugriff auf Eigenschaften und Methoden der Superklasse | 155 |
| 3.9 | Optionals | 156 |
| 3.9.1 | Deklaration von Optionals | 156 |
| 3.9.2 | Zugriff auf Optionals | 157 |
| 3.10 | Initialisierung | 160 |
| 3.10.1 | Schreiben von Initializern | 161 |
| 3.10.2 | Designated und Convenience Initializer | 166 |
| 3.10.3 | Initializer und Vererbung | 167 |
| 3.10.4 | Deinitialisierung | 169 |
| 3.11 | Type Casting | 170 |
| 3.11.1 | Typ prüfen | 170 |
| 3.11.2 | Downcasting | 172 |
| 3.12 | Enumerations | 174 |
| 3.12.1 | Zusätzliche Werte in Membern einer Enumeration speichern | 175 |
| 3.13 | Structures | 177 |
| 3.14 | Generics | 178 |
| 3.14.1 | Generic Function | 180 |
| 3.14.2 | Generic Type | 182 |
| 3.15 | Error Handling Model | 184 |
| 3.16 | Extensions | 187 |
| 3.17 | Protocols | 188 |
| 3.17.1 | Protocol Type | 189 |
| 3.18 | Access Control | 191 |
| 4 | Grundlagen der iOS-Entwicklung | 193 |
| 4.1 | Foundation-Framework | 193 |
| 4.1.1 | Die wichtigsten Klassen aus dem Foundation-Framework und ihre Funktionen | 194 |
| 4.2 | UIKit-Framework | 199 |
| 4.3 | Speicherverwaltung | 199 |
| 4.4 | Besonderheiten von Objective-C | 203 |
| 4.4.1 | Kurzschreibweisen zum Erstellen von Objekten | 203 |

| | | |
|----------|---|------------|
| 4.4.2 | Vergleichen der Werte von verschiedenen Objekten | 206 |
| 4.4.3 | Schlüsselwörter zum Zusammenspiel mit Optionals | 207 |
| 4.5 | Besonderheiten von Swift | 208 |
| 4.5.1 | Zusammenspiel zwischen Fundamental Types und Foundation-Framework-Klassen | 208 |
| 4.5.2 | Playgrounds im Detail | 208 |
| 4.6 | Objective-C und Swift vereint | 212 |
| 4.6.1 | Objective-C-Code in Swift verwenden | 213 |
| 4.6.2 | Swift-Code in Objective-C verwenden | 214 |
| 4.7 | NSError | 214 |
| 4.7.1 | Eigene Methode mit NSError-Parameter erstellen | 216 |
| 4.8 | Dokumentation | 217 |
| 4.8.1 | Besonderheiten bei Methoden | 218 |
| 4.8.2 | Doxygen-Dokumentation in Xcode | 220 |
| 4.9 | Nebenläufigkeit mit Grand Central Dispatch | 221 |
| 4.9.1 | Parallel laufenden Code erstellen | 222 |
| 4.10 | Grundlegende Struktur einer App | 224 |
| 4.10.1 | main.m | 224 |
| 4.10.2 | Info.plist | 225 |
| 4.10.3 | App Delegate | 225 |
| 4.11 | Lebenszyklus einer iOS-App | 226 |
| 4.11.1 | Start einer App | 226 |
| 4.11.2 | Lebenszyklus einer App | 227 |
| 4.11.3 | Die Methoden des App Delegate | 228 |
| 4.11.4 | Start der App | 229 |
| 4.12 | Tipps für die tägliche Arbeit | 231 |
| 4.12.1 | Die netten Kleinigkeiten | 231 |
| 4.12.2 | Fast Enumeration in Objective-C | 232 |
| 4.12.3 | Type Casting in Objective-C | 232 |
| 4.12.4 | Xcode-Beispielprojekte | 233 |
| 5 | Die Entwicklungsumgebung – Xcode | 235 |
| 5.1 | Willkommen bei Xcode! | 235 |
| 5.1.1 | Was ist Xcode? | 236 |
| 5.1.2 | Interface Builder und Xcode – endlich vereint! | 236 |
| 5.2 | Arbeiten mit Xcode | 237 |
| 5.2.1 | Dateien und Formate eines Xcode-Projekts | 237 |
| 5.2.2 | Umgang mit Dateien und Ordnern in Xcode | 242 |
| 5.3 | Der Aufbau von Xcode | 245 |
| 5.3.1 | Die Toolbar | 245 |
| 5.3.2 | Die Navigation Area | 247 |
| 5.3.3 | Die Editor Area | 250 |
| 5.3.4 | Die Utilities Area | 252 |
| 5.3.5 | Die Debug Area | 253 |

| | | |
|--------|--|-----|
| 5.4 | Einstellungen in Xcode | 254 |
| 5.4.1 | Anpassen von Xcode | 254 |
| 5.4.2 | General | 254 |
| 5.4.3 | Accounts | 255 |
| 5.4.4 | Behaviors | 256 |
| 5.4.5 | Navigation | 256 |
| 5.4.6 | Fonts & Colors | 257 |
| 5.4.7 | Text Editing | 258 |
| 5.4.8 | Key Bindings | 258 |
| 5.4.9 | Source Control | 259 |
| 5.4.10 | Components | 260 |
| 5.4.11 | Locations | 260 |
| 5.5 | Projekteinstellungen | 261 |
| 5.5.1 | Grundlagen | 261 |
| 5.5.2 | Einstellungen am Projekt | 263 |
| 5.5.3 | Einstellungen am Target | 266 |
| 5.5.4 | Einstellungen am Scheme | 272 |
| 5.6 | Grafiken und Asset-Bundles | 275 |
| 5.7 | Lokalisierung mit Localizable.strings | 277 |
| 5.7.1 | Grundlagen | 277 |
| 5.7.2 | NSLocalizedString | 277 |
| 5.7.3 | Erstellen der Localizable.strings-Datei | 278 |
| 5.7.4 | Localized String mit Parameter | 280 |
| 5.7.5 | Alle Localized Strings automatisch auslesen | 281 |
| 5.8 | Der iOS-Simulator | 282 |
| 5.8.1 | Grundlagen | 282 |
| 5.8.2 | Funktionen und Möglichkeiten des Simulators | 282 |
| 5.8.3 | Performance und Einschränkungen des Simulators | 286 |
| 5.9 | Dokumentation | 286 |
| 5.9.1 | Nichts geht über die Dokumentation! | 286 |
| 5.9.2 | Das Documentation-Window | 289 |
| 5.9.3 | Direktes Aufrufen der Dokumentation aus Xcode heraus | 292 |
| 5.10 | Devices | 293 |
| 5.11 | Organizer | 295 |
| 5.12 | Debugging in Xcode | 298 |
| 5.12.1 | Was ist Debugging? | 298 |
| 5.12.2 | Die Debug Area | 298 |
| 5.12.3 | Die Arbeit mit dem Debugger – NSLog und Breakpoints | 299 |
| 5.12.4 | Debug Navigator | 308 |
| 5.13 | Refactoring | 309 |
| 5.13.1 | Grundlagen | 309 |
| 5.13.2 | Refactoring-Funktionen in Xcode | 310 |

| | | |
|----------|---|------------|
| 5.14 | Instruments | 312 |
| 5.14.1 | Über Instruments | 312 |
| 5.14.2 | Nächste Schritte | 316 |
| 5.15 | Tipps für die tägliche Arbeit | 316 |
| 5.15.1 | Man lernt immer was dazu! | 316 |
| 5.15.2 | Code Snippets | 317 |
| 5.15.3 | Open Quickly | 318 |
| 5.15.4 | Caller einer Methode feststellen | 319 |
| 5.15.5 | Speicherorte für Ordner und Dateien ändern | 320 |
| 5.15.6 | Shortcuts für die Navigation Area | 320 |
| 5.15.7 | Run Without Building | 321 |
| 5.15.8 | Clean Build | 322 |
| 6 | MVC – Model-View-Controller | 323 |
| 6.1 | MVC ... was? | 323 |
| 6.2 | MVC in der Praxis | 325 |
| 6.3 | Kommunikation zwischen Model und Controller | 325 |
| 6.3.1 | Key-Value-Observing | 326 |
| 6.3.2 | Notifications | 332 |
| 6.4 | Kommunikation zwischen View und Controller | 335 |
| 6.4.1 | Target-Action | 335 |
| 6.4.2 | Delegation | 337 |
| 7 | Die Vielfalt der (View-)Controller | 339 |
| 7.1 | Alles beginnt mit einem View-Controller | 339 |
| 7.2 | UIViewController – die Mutter aller View-Controller | 341 |
| 7.2.1 | Wichtige Methoden von UIViewController | 343 |
| 7.2.2 | UIView – fester Bestandteil eines jeden UIViewControllers | 345 |
| 7.3 | View-Controller-Hierarchien | 346 |
| 7.4 | View-Controller erstellen mit dem Interface Builder | 348 |
| 7.4.1 | View-Controller mit NIB-File | 349 |
| 7.5 | Storyboards | 379 |
| 7.5.1 | Über Storyboards | 379 |
| 7.5.2 | Das Storyboard-Projekt | 380 |
| 7.5.3 | Die Klasse UIStoryboard | 390 |
| 7.5.4 | Segues | 392 |
| 7.5.5 | Zugriff über den App Delegate | 395 |
| 7.5.6 | Quo vadis – Storyboard oder NIB-File? | 396 |
| 7.6 | Auto Layout | 397 |
| 7.6.1 | Setzen und Konfigurieren von Constraints | 397 |
| 7.6.2 | Constraints bearbeiten und weiter anpassen | 399 |
| 7.6.3 | „Optimale“ Constraints automatisch setzen lassen | 401 |

| | | |
|----------|--|------------|
| 7.7 | UIViewController und seine Subklassen | 402 |
| 7.7.1 | UINavigationController | 403 |
| 7.7.2 | UITabBarController | 409 |
| 7.7.3 | UITableViewController | 413 |
| 7.7.4 | UICollectionViewController | 420 |
| 7.7.5 | UISplitViewController | 421 |
| 8 | Views erstellen und gestalten | 425 |
| 8.1 | Über Views in iOS | 425 |
| 8.2 | UIView – die Mutter aller Views | 425 |
| 8.3 | Arbeiten mit UIView | 426 |
| 8.3.1 | Programmatisches Erstellen einer UIView | 426 |
| 8.3.2 | View-Hierarchien | 428 |
| 8.3.3 | Weiterführendes zu UIView | 432 |
| 8.4 | Views erstellen mit dem Interface Builder | 433 |
| 8.4.1 | Grundlagen | 433 |
| 8.4.2 | View-Klasse mit NIB-File erstellen | 434 |
| 8.4.3 | Beliebiges NIB-File laden und verwenden | 438 |
| 8.4.4 | NIB-File nachträglich erstellen | 439 |
| 8.4.5 | Unterschiedliche NIB-Files für iPhone und iPad erstellen | 441 |
| 8.5 | Die wichtigsten Views und ihre Funktionen | 443 |
| 8.5.1 | Grundlagen | 443 |
| 8.5.2 | UILabel | 443 |
| 8.5.3 | UIButton | 443 |
| 8.5.4 | UISwitch | 444 |
| 8.5.5 | UISegmentedControl | 444 |
| 8.5.6 | UITextField | 444 |
| 8.5.7 | UIImageView | 445 |
| 8.5.8 | UIPickerView | 445 |
| 8.5.9 | UIDatePicker | 446 |
| 8.5.10 | UIWebView | 446 |
| 8.5.11 | UIMapView | 447 |
| 8.5.12 | UIScrollView | 447 |
| 8.5.13 | UITextView | 448 |
| 8.5.14 | UITableView | 449 |
| 8.5.15 | UICollectionView | 449 |
| 8.5.16 | Wichtig und unerlässlich: die Dokumentation! | 449 |
| 8.5.17 | Views und der Interface Builder | 450 |
| 8.6 | Die Grundlage gut gestalteter Views | 450 |
| 9 | Das Model und die Datenhaltung | 453 |
| 9.1 | Die Logik Ihrer App | 453 |
| 9.2 | Benutzereinstellungen sichern und nutzen | 454 |
| 9.2.1 | Über UserDefaults | 454 |
| 9.2.2 | Standardeinstellungen festlegen | 457 |

| | | |
|-----------|--|------------|
| 9.3 | Zugriff auf das Dateisystem | 457 |
| 9.3.1 | Das Dateisystem von iOS | 457 |
| 9.3.2 | FileManager | 459 |
| 9.3.3 | File-Sharing-Funktion nutzen | 466 |
| 9.4 | Core Data | 467 |
| 9.4.1 | Datenbankverwaltung mit Core Data | 467 |
| 9.4.2 | Wie funktioniert Core Data? | 468 |
| 9.4.3 | Die Klassen und Bestandteile von Core Data | 469 |
| 9.4.4 | Aufbau eines Standard-Core Data Stacks | 470 |
| 9.4.5 | Der Core Data-Editor | 473 |
| 9.4.6 | Erstellen eines neuen Managed-Objects | 481 |
| 9.4.7 | Löschen eines Managed-Objects | 482 |
| 9.4.8 | Laden von Managed-Objects | 482 |
| 9.4.9 | Was kommt als Nächstes? | 484 |
| 10 | Local und Push Notifications | 485 |
| 10.1 | Was sind Notifications? | 485 |
| 10.2 | Registrieren von Notification Types | 487 |
| 10.3 | Registrieren von Notification Categories und Actions | 491 |
| 10.3.1 | Erstellen einer Action | 491 |
| 10.3.2 | Erstellen einer Kategorie | 493 |
| 10.3.3 | Registrieren von Kategorien | 495 |
| 10.3.4 | Reagieren auf eine Action | 496 |
| 10.4 | Local Notifications | 498 |
| 10.4.1 | Konfiguration des Alerts | 498 |
| 10.4.2 | Konfiguration des Sounds | 500 |
| 10.4.3 | Konfiguration des Badge Values | 501 |
| 10.4.4 | Konfiguration von Audio, Bildern und Videos | 501 |
| 10.4.5 | Speichern zusätzlicher Informationen in einer Local Notification | 503 |
| 10.4.6 | Festlegen des Ausführungsergebnisses | 504 |
| 10.4.7 | Erstellen von Notification Requests | 508 |
| 10.4.8 | Registrieren von Local Notifications im System | 509 |
| 10.4.9 | Abbrechen bereits registrierter Local Notifications | 510 |
| 10.4.10 | Reagieren auf den Erhalt einer Notification bei aktiver App | 510 |
| 10.5 | Push Notifications | 512 |
| 10.5.1 | Versand von Push Notifications | 513 |
| 10.5.2 | Erstellen einer Push Notification | 517 |
| 10.5.3 | Quality of Service | 520 |
| 11 | Extensions | 521 |
| 11.1 | Verfügbare Typen von Extensions | 521 |
| 11.2 | Erstellen von Extensions in Xcode | 524 |
| 11.3 | Funktionsweise einer Extension | 527 |
| 11.4 | Wichtige Klassen und Objekte | 528 |

| | | |
|---------|---|-----|
| 11.5 | Unterstützte Dateitypen für Share- und Action-Extensions festlegen | 529 |
| 11.6 | Action Extension | 530 |
| 11.6.1 | Action mit User Interface | 530 |
| 11.6.2 | Action ohne User Interface | 531 |
| 11.7 | Content Blocker Extension | 532 |
| 11.7.1 | Konfiguration eines Content Blockers | 533 |
| 11.7.2 | Aktualisieren eines Content Blockers | 536 |
| 11.7.3 | Die Klasse ContentBlockerRequestHandler | 536 |
| 11.8 | Custom Keyboard | 537 |
| 11.8.1 | Erstellen eines Custom Keyboards | 537 |
| 11.8.2 | Arbeiten mit der Klasse UIInputViewController | 538 |
| 11.8.3 | Bearbeiten und Setzen von Text | 540 |
| 11.8.4 | Mehrsprachige Keyboards | 541 |
| 11.9 | Document Provider | 541 |
| 11.9.1 | Document Provider-Extension | 542 |
| 11.9.2 | File Provider | 545 |
| 11.9.3 | Document Provider aufrufen | 547 |
| 11.10 | iMessage Extension | 549 |
| 11.10.1 | Aufbau und Funktionsweise der iMessage Extension | 549 |
| 11.10.2 | Entwicklung einer iMessage Extension | 553 |
| 11.11 | Intents Extension | 564 |
| 11.11.1 | Domains und Intents | 565 |
| 11.11.2 | Bestandteile einer Intents Extension | 566 |
| 11.11.3 | Funktionsweise einer Intents Extension | 567 |
| 11.11.4 | Übersicht über verfügbare Intents | 576 |
| 11.11.5 | Voraussetzungen zur Verwendung von Siri in einer Intents Extension | 578 |
| 11.11.6 | Erweitern von Siris Vokabular | 581 |
| 11.11.7 | Testen einer Intents Extension | 587 |
| 11.12 | Intents UI Extension | 588 |
| 11.12.1 | Vorbereitung der Intents UI Extension | 589 |
| 11.12.2 | Konfiguration des View-Controllers | 590 |
| 11.12.3 | Default-Layout bei Maps und Messaging deaktivieren | 592 |
| 11.13 | Notification Content Extension | 592 |
| 11.13.1 | UNNotificationContentExtension | 592 |
| 11.13.2 | Konfiguration der Info.plist-Datei | 594 |
| 11.14 | Notification Service Extension | 595 |
| 11.15 | Photo Editing Extension | 596 |
| 11.15.1 | Festlegen der unterstützten Typen zur Bearbeitung | 600 |
| 11.16 | Share Extension | 601 |
| 11.17 | Shared Links Extension | 602 |
| 11.17.1 | Erstellen eines NSExtensionItem | 603 |
| 11.17.2 | NSExtensionItem als Shared Link bereitstellen | 604 |

| | | |
|-----------|--|------------|
| 11.18 | Sticker Pack Extension | 605 |
| 11.18.1 | Erstellen einer Sticker Pack Extension | 606 |
| 11.18.2 | Komplexere Sticker Pack Extensions erstellen | 610 |
| 11.19 | Today Extension | 612 |
| 11.19.1 | Today Extension testen | 614 |
| 11.20 | Watch App | 615 |
| 11.21 | App Groups | 615 |
| 11.21.1 | Registrieren einer App Group im Apple Developer Portal | 616 |
| 11.21.2 | Registrieren einer App Group innerhalb einer App | 617 |
| 11.21.3 | Zugriff auf eine App Group | 618 |
| 12 | App-Entwicklung für die Apple Watch | 621 |
| 12.1 | Apples neues großes Ding: Die Apple Watch | 621 |
| 12.2 | Möglichkeiten, Einschränkungen, Unterschiede | 622 |
| 12.3 | Das WatchKit SDK | 624 |
| 12.3.1 | WKInterfaceController | 625 |
| 12.3.2 | WKInterfaceObject | 625 |
| 12.3.3 | WKExtensionDelegate | 626 |
| 12.3.4 | Weitere Klassen | 626 |
| 12.4 | Aufbau und Funktionsweise von Apple Watch-Apps | 627 |
| 12.4.1 | iPhone-App | 627 |
| 12.4.2 | WatchKit Extension | 627 |
| 12.4.3 | WatchKit App | 628 |
| 12.4.4 | Verbindung von WatchKit Extension und WatchKit App | 628 |
| 12.4.5 | Notification Scene | 628 |
| 12.4.6 | Complications | 629 |
| 12.5 | Erstellen einer WatchKit App mitsamt WatchKit Extension | 629 |
| 12.5.1 | Dateien der WatchKit Extension | 632 |
| 12.5.2 | Dateien der WatchKit App | 633 |
| 12.5.3 | Ausführen und Testen der Apple Watch-App | 634 |
| 12.6 | Lebenszyklus einer WatchKit App | 635 |
| 12.7 | Der WKInterfaceController im Detail | 637 |
| 12.7.1 | Initialisierung | 637 |
| 12.7.2 | Activation Events | 639 |
| 12.7.3 | Setzen des Titels | 639 |
| 12.7.4 | Ein- und Ausblenden von Interface-Controllern | 640 |
| 12.7.5 | Umsetzen eines Navigation Stacks | 642 |
| 12.7.6 | Reaktion auf Storyboard-Events | 643 |
| 12.7.7 | Weitere Attribute | 644 |
| 12.7.8 | Weitere Funktionen von WKInterfaceController | 645 |
| 12.8 | Arbeiten mit dem Interface-Storyboard einer WatchKit App | 645 |
| 12.8.1 | Erstellen und Konfigurieren eines WKInterfaceController | 646 |
| 12.8.2 | Hinzufügen und Konfigurieren von Interface-Elementen | 648 |

| | | |
|---------|---|-----|
| 12.8.3 | Positionierung und Anordnung von Interface-Elementen | 649 |
| 12.8.4 | Ändern der Größe von Interface-Elementen | 649 |
| 12.8.5 | Unterschiedliche Konfigurationen für verschiedene Apple Watch-Größen | 651 |
| 12.8.6 | Gruppierung von Interface-Elementen mittels WKInterfaceGroup | 653 |
| 12.8.7 | Verbindung von Storyboard und Code | 655 |
| 12.8.8 | Zusammenfassen mehrerer Interface-Controller zu einem page-based Interface | 659 |
| 12.8.9 | Erstellen und Konfigurieren von Segues | 659 |
| 12.9 | Erstellen von Tabellen | 662 |
| 12.9.1 | Hinzufügen einer Tabelle im Storyboard | 662 |
| 12.9.2 | Konfiguration einer Zelle | 663 |
| 12.9.3 | Konfiguration einer Tabelle | 667 |
| 12.9.4 | Verwenden verschiedener Zellen in einer Tabelle | 669 |
| 12.9.5 | Zellen hinzufügen und entfernen | 672 |
| 12.9.6 | Direkt zu einer bestimmten Zelle scrollen | 673 |
| 12.9.7 | Aktuelle Anzahl an Zellen auslesen | 673 |
| 12.9.8 | Auf die Auswahl einer Zelle reagieren | 674 |
| 12.9.9 | Segues von Zellen einer Tabelle über das Storyboard konfigurieren | 675 |
| 12.10 | Erstellen von Menüs | 676 |
| 12.10.1 | Erstellen eines Menüs im Storyboard | 677 |
| 12.10.2 | Erstellen eines Menüs im Code | 681 |
| 12.10.3 | Fazit: Menüerstellung im Storyboard oder im Code? | 683 |
| 12.10.4 | Mischen von Menüpunkten aus Storyboard und Code | 684 |
| 12.11 | Eingabe von Text | 684 |
| 12.12 | Notification Scene | 686 |
| 12.12.1 | Short-Look und Long-Look Interface | 687 |
| 12.12.2 | Long-Look Interface im Detail | 688 |
| 12.12.3 | Erstellen eigener Notification Scenes | 689 |
| 12.12.4 | Testen einer Notification Scene | 697 |
| 12.13 | Complications | 697 |
| 12.13.1 | Was sind Complications? | 698 |
| 12.13.2 | Das ClockKit Framework | 698 |
| 12.13.3 | Aufbau und Bestandteile von Complications | 698 |
| 12.13.4 | Vorbereiten des eigenen Projekts | 702 |
| 12.13.5 | Entwicklung einer Complication | 704 |
| 12.13.6 | Bereitstellen der Complication mittels CLKComplicationDataSource | 709 |
| 12.14 | Kommunikation und Datenaustausch zwischen iOS und watchOS | 712 |
| 12.14.1 | Watch Connectivity | 713 |
| 12.15 | Was sonst noch zu sagen und zu beachten ist | 718 |

| | | |
|--------------|--|------------|
| 13 | Tests | 721 |
| 13.1 | Unit-Tests | 721 |
| 13.1.1 | Aufbau und Funktionsweise von Unit-Tests | 726 |
| 13.1.2 | Aufbau einer Test-Case-Klasse | 728 |
| 13.1.3 | Neue Test-Case-Klasse erstellen | 730 |
| 13.1.4 | Ausführen von Unit-Tests | 732 |
| 13.1.5 | Was sollte ich eigentlich testen? | 734 |
| 13.2 | Performance-Tests | 734 |
| 13.3 | UI-Tests | 736 |
| 13.3.1 | Klassen für UI-Tests | 737 |
| 13.3.2 | Aufbau von UI-Test-Klassen | 740 |
| 13.3.3 | Automatisches Erstellen von UI-Tests | 740 |
| 13.3.4 | Einsatz von UI-Tests | 741 |
| 13.4 | Test-Driven Development | 741 |
| 14 | Versionierung | 743 |
| 14.1 | Über Versionskontrolle | 743 |
| 14.2 | Basisfunktionen und -begriffe von Git | 744 |
| 14.2.1 | Begriffe | 744 |
| 14.2.2 | Funktionen | 744 |
| 14.3 | Source Control in Xcode | 746 |
| 14.4 | Version Editor und Source Control | 750 |
| 15 | Veröffentlichung im App Store | 753 |
| 15.1 | Zertifikate, Provisioning Profiles und Ihre App | 753 |
| 15.1.1 | Certificates, IDs & Profiles | 755 |
| 15.1.2 | Erstellen von... | 757 |
| 15.2 | Testen auf dem eigenen Endgerät | 770 |
| 15.2.1 | Setzen des Teams | 770 |
| 15.2.2 | Auswahl Ihres iOS-Geräts | 770 |
| 15.3 | iTunes Connect und Veröffentlichung im App Store | 772 |
| 15.3.1 | Vorbereiten der App in iTunes Connect | 774 |
| 15.3.2 | Upload der App in den App Store | 777 |
| 15.3.3 | Wie geht es weiter? | 778 |
| Index | | 779 |

Vorwort

iOS ist und bleibt für Entwickler ein spannendes Feld, nicht zuletzt, da Apple mit dem App Store einen Weg geschaffen hat, mit dem auch einzelne Entwickler Software für einen internationalen Markt verbreiten können, ohne dass sie sich um Dinge wie Bezahlsysteme, Abrechnungen und Download Gedanken machen müssen. Ich selbst habe mich aufgrund dessen vor über fünf Jahren für die iOS-Entwicklung begeistert lassen und habe bis heute nichts von der Faszination für diese spannende und innovative Plattform verloren.

Beim Einstieg in die iOS-Entwicklung habe ich eines aber schmerzlich vermisst: einen einfachen, übersichtlichen und professionellen Einstieg. Ich habe mich mit viel Literatur auseinandergesetzt, war in vielen Foren unterwegs und habe schlicht und einfach viel ausprobiert. Da vieles von diesen Anfängen aber sehr umständlich oder im Nachhinein betrachtet sogar gänzlich falsch war, hat mich das viel Zeit und Lehrgeld gekostet. Und ich habe mir oft gewünscht, man hätte mich von Beginn an an die Hand genommen und mir nicht nur gezeigt, *wie* ich Apps für iOS entwickle, sondern *wie ich gute und professionelle Apps entwickle*, welche Besonderheiten, Best Practices und Design Patterns es gibt und wie ich effektiv und effizient mit der Entwicklungsumgebung arbeiten kann. Und dieser Wunsch hat den Grundstein für dieses Buch gelegt.

Dieses Buch vermittelt Ihnen alle essenziellen Grundlagen und Kenntnisse über die Entwicklung für iOS. Angefangen bei der Vorstellung des Betriebssystems selbst geht es weiter über die Programmiersprachen Objective-C und Swift, deren Struktur und jeweiligen Besonderheiten und all das, was sie ausmacht. Danach rückt die eigentliche Entwicklung für iOS in den Fokus. Sie erfahren alles über die wichtigsten Frameworks von Apple für die App-Entwicklung und lernen typische Best Practices kennen. Besonders wichtig ist hier auch die Vorstellung der Dokumentation, die für Sie als App-Entwickler Bestandteil Ihrer täglichen Arbeit sein wird. Denn letztlich beherbergt die Apple-Dokumentation alle Antworten auf die Fragen, wie Sie bestimmte Probleme in der iOS-Entwicklung angehen und welche Möglichkeiten Ihnen die einzelnen Frameworks von Apple liefern.

Nach der Vorstellung der Plattform und der Programmiersprache(n) geht es weiter mit der Entwicklungsumgebung Xcode, mit der wir unsere Apps für iOS entwickeln. Dabei war es mir besonders wichtig, den genauen Aufbau und die Struktur hinter Xcode vorzustellen sowie alle spannenden Möglichkeiten und Kniffe aufzuzeigen, die Xcode Ihnen bietet und Ihre tägliche Arbeit erleichtern. Auch der Umgang mit Grafiken oder die Lokalisierung Ihrer Anwendung sowie Debugging und Refactoring habe ich in dieses Kapitel mit aufgenommen.

Bis zu diesem Punkt habe ich mich rein mit den essenziellen Grundlagen beschäftigt und ich finde es wichtig, dass auch Sie diese Grundlagen verinnerlicht und verstanden haben, denn sie sind die Grundpfeiler für gute und erfolgreiche Apps. Dazu abschließend folgt im sechsten Kapitel die Vorstellung von MVC – Model-View-Controller. Dabei handelt es sich um eines der wichtigsten Design-Patterns in iOS und ist essenziell für die App-Entwicklung. Aufgrund dessen widme ich MVC ein eigenes Kapitel, stelle es im Detail vor und erkläre, wie und warum Sie es in Ihren eigenen Apps anwenden sollen.

Anschließend geht es im Speziellen um die Entwicklung für iOS und die Nutzung der Funktionen und Frameworks für Apple, unterteilt auf die drei Bereiche Controller, View und Model aus dem MVC-Pattern. Auch in diesen Kapiteln geht es darum, die grundlegenden Besonderheiten zu erläutern und aufzuzeigen, wie Sie mit den einzelnen Elementen arbeiten und diese in Ihren eigenen Apps verwenden. Sie lernen die wichtigsten Elemente aus den jeweiligen Bereichen kennen und erfahren, wie Sie selbstständig mit ihnen arbeiten können und worauf bei der jeweiligen Verwendung zu achten ist. Mit diesem Wissen gewappnet sind Sie imstande, sich selbst in neue Frameworks, Technologien und Themen anhand der Apple-Dokumentation einzuarbeiten und selbstständig Probleme zu lösen. Und genau das ist es, was einen guten und professionellen iOS-Entwickler ausmacht.

Kapitel 10 setzt sich im Detail mit den sogenannten Local und Push Notifications auseinander, die es Ihnen erlauben, Nachrichten aus Ihren Apps an Ihre Nutzer zu senden. Im darauffolgenden Kapitel erstelle ich Ihnen ergänzend dazu dann Extensions vor, die uns Entwicklern ganz neue und innovative Möglichkeiten an die Hand geben, unsere Apps zu erweitern und über das gesamte iOS-System heraus zugänglich zu machen.

Im Anschluss daran widme ich ein eigenes und umfangreiches Kapitel der Entwicklung für die Apple Watch. Die Apple Watch ist die neue spannende Plattform in Apples Ökosystem und ist – was die grundsätzliche App-Entwicklung betrifft – in vielen Dingen recht ähnlich zur iOS-Plattform. Da eine Apple Watch-App immer eine iPhone-App voraussetzt, war es nur passend, auch die Entwicklung für die Apple Watch in diesem Buch im Detail zu beleuchten und zu zeigen, wie Sie Ihre eigenen iPhone-Apps um ein Pendant für die Apple Watch erweitern können. Sie lernen alle Möglichkeiten und Einschränkungen kennen und werden so in die Lage versetzt, selbst mit der Entwicklung eigener Apple Watch-Apps zu beginnen.

Anschließend folgen die Themen Tests und Versionsverwaltung mit Git, die in jeder neuen Xcode-Version mehr und mehr in die Entwicklungsumgebung integriert und unterstützt werden. Ich zeige Ihnen dabei, was Unit-Tests sind, warum Sie sie in Ihren Apps verwenden sollten, wie Sie Unit-Tests schreiben und wie Sie Xcode in Sachen Unit-Tests unterstützt und Ihnen unter die Arme greift. Auch UI-Tests finden dabei Beachtung. Bei der Versionsverwaltung mit Git erfahren Sie alles über die integrierten Funktionen zur Arbeit mit Git in Xcode und wie Sie Änderungen im Code verfolgen und nachvollziehen können.

Zu guter Letzt geht es noch – wie könnte es anders sein? – um die Veröffentlichung Ihrer Apps im App Store und die integrierten Tools in Xcode, die Ihnen bei diesem Prozess unter die Arme greifen. Insbesondere erfahren Sie hier etwas über die Erstellung und Verwaltung Ihrer Apps in Apples Developer Program.

Bei allen diesen Themen soll dieses Buch Sie unterstützen und Ihnen die Grundlagen und das essenzielle Praxiswissen vermitteln und mit auf den Weg geben. Es soll Ihnen nicht Beispielprojekte aufzeigen und deren Verhalten und Eigenschaften erklären (davon gibt es

nämlich von Apple selbst mehr als genug), sondern Ihnen das nötige Wissen mitgeben, um Sie in die Lage zu versetzen, Problemstellungen selbstständig zu lösen und zu verstehen, wie Sie gute und professionelle iOS-Apps entwickeln. Denn wenn Sie diesen Status erreicht haben, können Sie darauf aufbauen, experimentieren und eigene spannende und innovative iOS-Projekte umsetzen. Und ich bin gespannt, welche großartigen Apps wir von Ihnen erwarten dürfen.

Ich wünsche Ihnen viel Freude beim Lesen dieses Buches und viel Erfolg mit all Ihren iOS-Projekten.

Thomas Sillmann



ios-buch.thomassillmann.de

Unter dieser Adresse finden Sie zusätzliche Informationen und Services, die ich ergänzend zu diesem Buch anbiete. Dazu gehören vollständige Code-Beispiele und Projekte, anhand derer Sie Ihr erlangtes Wissen testen und vertiefen können, sowie verschiedene Frameworks, die Sie direkt in eigene Projekte integrieren können. Ebenso stelle ich dort Lern-Videos zur App-Entwicklung bereit und berichte in Blog-Beiträgen über Neuerungen und Aktualisierungen; ein Besuch lohnt sich also. ☺

1

Über iOS

■ 1.1 Was ist iOS?

Auch wenn diese Frage in der heutigen Zeit möglicherweise überflüssig erscheint (und auch in Anbetracht dessen, dass Sie sich dieses Buch gekauft haben), möchte ich zu Beginn doch zumindest kurz darauf eingehen, was eigentlich dieses iOS ist, für das ich mich – und Sie sich offensichtlich auch – als Entwickler so sehr interessiere. Dabei werde ich auch direkt den Spagat schlagen und Ihnen die Geräte vorstellen, auf denen iOS verfügbar ist, und beschreiben, wie sich das System im Laufe der Jahre entwickelt hat.

Zunächst einmal ist iOS ein Betriebssystem der Firma Apple. Seinen ersten großen Auftritt hatte es im Jahr 2007 zusammen mit der Vorstellung des allerersten iPhone, denn genau auf diesem Gerät lief und läuft bis heute iOS (auch wenn es damals noch iPhone OS hieß). Mit dem iPhone krempelte sich der Markt der Smartphones maßgeblich um und heutzutage sieht man Touch-Smartphones mit dem Bildschirm als Hauptbedienelement allerorten.

Nach mehreren Hardware-Sprüngen des iPhone folgte im Jahr 2010 das nächste iOS-Device von Apple: Das iPad, welches – ebenso wie das iPhone zuvor den Smartphone-Markt – nun den Tablet-Markt ordentlich aufmischt und bis heute den Quasistandard im Bereich Tablets setzt. Auch auf dem iPad läuft Apples Mobil-Betriebssystem iOS (dessen Namensänderung ebenfalls im Jahr 2010 mit dem Erscheinen des iPad von iPhone OS zu iOS erfolgte).

Darüber hinaus läuft iOS auch auf dem iPod touch. Alle Apps, die Sie für das iPhone entwickeln, sind prinzipiell ebenfalls auf dem iPod touch lauffähig, lediglich die zugrunde liegende Hardware unterscheidet sich ein wenig; Telefonieren ist beispielsweise mit dem iPod touch nicht möglich. So kann sich aber ein iPod touch durchaus als günstiges Testgerät für iOS-Apps anbieten (das iPhone spielt da nun mal doch in einer etwas anderen Preisklasse).

Und direkt zu Beginn noch eine kleine Randnotiz: Das von Apple im Jahr 2015 neu vorgestellte und überarbeitete Apple TV besitzt viele Parallelen zu iOS, was die App-Entwicklung betrifft. Zwar verfügt dieses neue Apple TV über ein eigenes Betriebssystem (bekannt unter dem passenden Namen tvOS), dieses besitzt aber viele Gemeinsamkeiten mit iOS. All das Wissen, das Sie sich mithilfe dieses Buches zur iOS-Entwicklung aneignen, können Sie somit auch als perfekte Grundlage für etwaige Apps für das Apple TV heranziehen.



Bild 1.1 iPhone und iPad sind die erfolgreichsten Geräte mit dem Betriebssystem iOS. Daneben verfügt auch Apples iPod touch über iOS als Betriebssystem.

1.1.1 iOS und macOS

Zusammengefasst lässt sich also einfach sagen: iOS ist das Betriebssystem von Apples iPhone-, iPad- und iPod touch-Familie. Sicherlich wissen Sie aber auch, dass Apple nicht nur iOS-Geräte entwickelt und veröffentlicht (auch wenn das aktuell das Geschäft ist, das Apple den größten Umsatz einbringt). Daneben gibt es noch – neben der Apple Watch und dem Apple TV, auf die ich an dieser Stelle ausnahmsweise (noch) nicht eingehe – die Mac-Familie, die Apples Produktplatte aus Notebooks und Desktop-PCs darstellt. Und besonders spannend ist hierbei, dass iOS zu großen Teilen auf macOS – dem Betriebssystem der Macs und ehemals unter dem Namen OS X bekannt – basiert. So sind viele Frameworks, mit denen wir in der iOS-Entwicklung arbeiten werden, unter macOS in derselben oder in einer leicht abgewandelten Form verfügbar. Das bedeutet umgekehrt auch, dass der Einstieg in die macOS-Entwicklung leichter fällt, wenn Sie bereits für iOS entwickelt haben – und umgekehrt. Das aber nur als kleine Randnotiz und mögliche Motivation, sich nach der Lektüre dieses Buches eventuell auch mit der macOS-Entwicklung näher auseinanderzusetzen; Sie werden sehen, über das nötige Rüstzeug verfügen Sie dann. ☺

1.1.2 Besonderheiten der iOS-Plattform

Auch wenn iOS auf macOS basiert, so gibt es doch mannigfaltige Unterschiede zwischen den beiden Betriebssystemen (auch wenn sie sich unter der Haube relativ ähnlich sind).

Entscheidend anders sind die Bedienoberflächen und das Bedienkonzept gestaltet. Während macOS und jedes andere Desktop-Betriebssystem typischerweise mittels Maus und Tastatur gesteuert werden, verfügen iOS-Geräte lediglich über einen Touchscreen, über den mittels Fingergesten und Berührungen alle Aktionen gesteuert werden. Hier gibt es also ganz neue Aspekte, auf die wir als Entwickler achten müssen, um gut funktionierende und intuitiv bedienbare Apps zu entwickeln. Denn ein Finger zum Bedienen eines Touchscreens ist nun mal etwas gänzlich anderes als eine Maus, die ich pixelgenau an jede Position bewegen kann. Ein Finger besitzt wesentlich mehr Fläche und allein das muss bereits beim Konzipieren und Entwickeln eigener Anwendungen für iOS maßgeblich beachtet werden.

Auch sind die Nutzer mit iPhone und iPad mobil unterwegs, was in heutigen Zeiten mit sehr gutem Ausbau des Mobilfunknetzes nichtsdestotrotz bedeutet: Nicht immer ist Internet verfügbar (mal ganz davon abgesehen, dass es das iPad auch in einer reinen WLAN-Version ohne Mobilfunkverbindung gibt) und den Nutzer dazu zu zwingen, eine Internetverbindung herzustellen, sollte nur wirklich dann erforderlich sein, wenn es gar nicht anders geht und ein Internetzugang zwingend für die Nutzung der eigenen App (oder der gerade benötigten Funktion) notwendig ist.

iPhone und iPad sind Mobilgeräte, und genau so werden sie auch genutzt, soll heißen: Viele Nutzer holen ihr Smartphone nur für den Bruchteil eines Augenblicks hervor, checken aktuelle Facebook- oder WhatsApp-Nachrichten und lassen das Handy anschließend wieder verschwinden. Auch für Sie als App-Entwickler gilt: Halten Sie den Nutzer bei dem, was er mit Ihrer App tun will, nicht auf. Weniger ist hier ganz klar mehr. Ihre App soll eine eindeutige Funktion erfüllen, bieten Sie diese darum dem Nutzer so komfortabel, übersichtlich und leicht zugänglich wie nur irgend möglich an.

Daneben gibt es noch einen weiteren wichtigen Aspekt, den wir als Entwickler immer berücksichtigen sollten: Schonender Umgang mit den Akku-Ressourcen. Wenn wir ununterbrochen Bluetooth in Beschlag nehmen und nach anderen Geräten suchen, saugen wir den Akku des Nutzers damit sehr schnell leer und dürfen uns wahrscheinlich im Umkehrschluss über schlechte Kritiken unserer App im App Store „freuen“. Hier gilt ganz klar: Weniger ist mehr, und Ihre App sollte sich immer auf genau die Aufgabe konzentrieren, für die sie gedacht ist.

Sie sehen also, Apps für iOS zu entwickeln besteht nicht nur darin, die Programmiersprache(n) und SDKs zu beherrschen; es geht auch darum, zu verstehen, wie die iOS-Gerätefamilie funktioniert, wie sie genutzt wird und wie Sie mit Ihren Apps den Nutzern das Leben erleichtern.

■ 1.2 iOS für Entwickler

Apple hat mit dem App Store und iOS eine großartige Infrastruktur für uns Entwickler geschaffen. Wir können uns damit voll und ganz auf die Entwicklung unserer Apps konzentrieren, alle sonstigen Modalitäten wie Bezahlmethoden, Zahlungseingang oder Vertrieb übernimmt Apple für uns. Auch wenn Apple dafür einen Obolus in Form eines jährlichen Mitgliedsbeitrags im Apple Developer Program als auch 30 % der Erlöse pro verkaufter App fordert, so stellt der App Store doch eine großartige Möglichkeit dar, die eigene Anwendung binnen kürzester Zeit einem internationalen (und auch durchaus zahlungswilligen) Nutzerkreis zum Download zur Verfügung zu stellen.

Was an dieser Stelle auch gleich gesagt sein muss: Es gibt von Apple keinen anderen vorgesehenen Weg zur Installation einer App auf einem iOS-Gerät außer dem offiziellen App Store (von den Besonderheiten firmeninterner Businessanwendungen sehe ich an dieser Stelle einmal ab). Höchstwahrscheinlich haben Sie bereits einmal von einem Jailbreak gehört, der es ermöglicht, unter anderem den Weg über den App Store zu umgehen und dadurch Apps aus beliebigen Quellen (wie zum Beispiel direkt über die Website eines Anbieters) auf dem eigenen Gerät zu installieren; das sollte aber nicht Ihr bevorzugtes Vorgehen sein, wenn Sie Apps für iOS entwickeln möchten. Zum einen schiebt Apple den Lücken, über die sich ein solcher Jailbreak durchführen lässt, regelmäßig mittels Software-Updates einen Riegel vor, zum anderen ist das schlicht und einfach nicht der Weg, diese Plattform zu nutzen. Apple hat die iOS-Geräte als relativ geschlossene und abgeschottete Systeme konzipiert, und genau so sollten sie auch betrachtet und genutzt werden. Denn auf der anderen Seite sollte nicht der Sicherheitsgewinn vergessen werden, den gerade iOS gegenüber anderen Mobil-Betriebssystemen innehat; Schadsoftware lässt sich nur schwer bis gar nicht auf den Geräten installieren. Möglicherweise halten Sie diese Einstellung für engstirnig und betiteln mich in Gedanken bereits als „Apple Fanboy“, ich versuche aber schlicht, die iOS-Plattform als das zu sehen, was sie ist, und sie so zu nutzen, wie es gedacht ist. Damit ist sowohl uns Entwicklern als auch all den Millionen iOS-Nutzern da draußen am meisten geholfen.

So weit, so gut, doch was benötige ich als Entwickler nun konkret, um mit der Entwicklung für iOS starten zu können?

1.2.1 Hardware für Entwickler

Um für iOS entwickeln zu können, benötigen Sie in jedem Fall einen gewissen Fuhrpark an Apple-Geräten. Zunächst wäre hier einmal der Mac genannt. Ja, ein Mac ist notwendig, um für iOS entwickeln zu können, denn nur unter macOS – dem Betriebssystem des Mac – stehen die SDKs und Frameworks zur Entwicklung für iOS zur Verfügung. Für welches Gerät Sie sich dabei im Detail entscheiden, ist gut und gerne Ihnen und Ihren persönlichen Vorlieben überlassen, leistungstechnisch eignen sich alle. Achten Sie im Idealfall am ehesten noch darauf, einen Mac mit mindestens 8 GB Arbeitsspeicher zu erstehen; für die Entwicklung und zum Kompilieren Ihrer Apps ist das ein sehr angenehmes Mindestmaß, glauben Sie mir (siehe Bild 1.2).



Bild 1.2 Die Macs von Apple eignen sich alle gleichermaßen zur Entwicklung für iOS, auch wenn man mindestens 8 GB Arbeitsspeicher beim Gerätekauf berücksichtigen sollte.



Tipp

Ein Detail, auf das Sie möglicherweise ebenfalls beim Mac-Kauf achten sollten, ist das Display. In den letzten Jahren hat Apple sein sogenanntes Retina-Display nämlich nicht nur in iPhones und iPads, sondern auch in immer mehr Mac-Modellen verbaut. Aktuell ist das MacBook Air der einzige verbliebene Mac, der noch nicht über ein so hochauflösendes Display verfügt, während sich bei MacBook (12 Zoll), MacBook Pro und iMac inzwischen passende Modelle mit Retina-Display finden.

Der Grund, warum ein Retina-Display für die App-Entwicklung von Vorteil sein kann, ist im Testen Ihrer Anwendungen im sogenannten Simulator begründet. Ein iOS-Simulator „simuliert“ iPhone und iPad auf Ihrem Mac und erlaubt es somit, Ihre Apps direkt auf dem Mac auszuführen und zu testen. Verfügt Ihr Mac dann selbst über ein solches Retina-Display, nimmt der Simulator nicht so viel Platz ein wie auf einem Nicht-Retina-Display. Das liegt daran, dass Retina doppelt oder sogar dreifach so viele Pixel für einen Punkt darstellt, und bei Displays, die über keine so hohe Auflösung verfügen, verschlingt der Retina-Simulator entsprechend deutlich mehr Platz (keine Sorge, wenn Ihnen von den vielen Worten wie Retina, Pixel und Punkte ein wenig schwindelig geworden ist, später gehe ich noch näher darauf ein ☺).

Kurzum: Für die App-Entwicklung für iOS lohnt es sich, ein Retina-Display einzusetzen. ☺

Neben Ihrem Mac sollten Sie auch mindestens ein iPhone und/oder iPad besitzen – eben je nachdem, ob Sie nur für eine oder für beide Plattformen Apps entwickeln. Zwar haben Sie die Chance, all Ihre entwickelten Apps auch in einem Simulator auszuführen und zu testen, Sie sollten aber in jedem Fall vor der Veröffentlichung Ihrer App im App Store diese auch auf einem richtigen iOS-Gerät ausführlich geprüft haben. Der Simulator nutzt nämlich die komplette Hardware-Power Ihres Mac, wodurch es passieren kann, dass zwar im Simulator alles schnell und fluffig läuft, aber auf einem richtigen (möglicherweise auch schon etwas betagteren) iOS-Gerät alles ruckelt oder sogar abstürzt; dazu aber später noch mehr. ☺

1.2.2 Software für Entwickler

Das Programm, mit dem Sie als Entwickler die meiste Zeit verbringen werden, ist Xcode. Xcode ist kostenlos über den App Store Ihres Mac erhältlich und ist die komplette IDE von Apple zur Softwareentwicklung für iOS, macOS, watchOS und tvOS (eben für all die verfügbaren Betriebssystemplattformen von Apple). Es liefert alle Werkzeuge, die Sie benötigen, inklusive aller SDKs und Simulatoren (sehen Sie dazu auch das fünfte Kapitel, „Die Entwicklungsumgebung – Xcode“). Bild 1.3 zeigt das App-Icon von Xcode.



Bild 1.3

Xcode: Mit dieser IDE legen Sie in der iOS-Entwicklung richtig los!

Daneben gibt es noch weitere Software, die Ihnen in Ihrer täglichen Entwicklerarbeit das Leben erleichtern kann, zum Beispiel spezielle Clients zur Arbeit mit der Versionsverwaltung Git oder Apps zur Arbeit mit Datenbanken. Diese sind aber nicht notwendig und werden von mir an passender Stelle im Buch in der benötigten Tiefe vorgestellt.

1.2.3 Das Apple Developer Program

Wie eingangs erwähnt, ist es mit der Entwicklung einer App allein noch nicht mit der Veröffentlichung im App Store getan. Neben den 30 % Verkaufserlös, die Apple automatisch von Ihren App-Verkäufen abzwackt, müssen Sie auch noch Mitglied im Apple Developer Program sein. Als Mitglied können Sie dann ein eigenes Entwicklerzertifikat beantragen und damit Ihre Apps signieren, was nötig ist, um diese in den App Store einzureichen als auch um sie auf einem eigenen iOS-Gerät testen zu können. Insgesamt gibt es vier verschiedene Formen des Apple Developer Program, die hier nun einmal in Kürze vorgestellt werden sollen:

Apple Developer Program Individual

Das ist das Entwicklerprogramm für alle, die alleine bzw. freiberuflich oder selbstständig Apps für iOS entwickeln und veröffentlichen möchten. Die Mitgliedschaft kostet 99 \$ pro Jahr und erlaubt den vollen Zugriff auf alle Entwicklerressourcen von Apple, einschließlich Vorabversionen der Entwicklungsumgebung Xcode als auch iOS selbst.

Apple Developer Program Company

Das Apple Developer Program Company entspricht im Großen und Ganzen dem Individual Program, nur dass dieses hier explizit für Firmen ausgelegt ist. Dieses Programm bietet daher auch die Möglichkeit, mehrere Teammitglieder und Entwickler mit verschiedenen Rollen anzulegen und zu verwalten. Die Kosten belaufen sich – ebenfalls wie bei Individual – auf 99 \$ im Jahr.

Apple Developer Enterprise Program

Das Enterprise Program ist für Firmen gedacht, die Apps für interne Geschäftszwecke entwickeln und nutzen möchten. So erlaubt dieses Programm das Veröffentlichen von Apps auf beliebig vielen im Programm registrierten Geräten, es ist allerdings keine Veröffentlichung von Apps in Apples App Store möglich. Die Kosten für dieses Programm liegen bei 299 \$ pro Jahr.

Apple Developer University Program

Das Apple Developer University Program ist das einzige kostenlose Entwicklerprogramm von Apple. Wie der Name bereits andeutet, richtet es sich an Studierende und Lehrkräfte von Hochschulen und Universitäten, um dort beispielsweise iOS-Entwicklung zu unterrichten (siehe Bild 1.4).



Bild 1.4 Auf <https://developer.apple.com/programs> finden Sie weitere Informationen zum Apple Developer Program und gelangen von dort auch zur Registrierung. (Bild: developer.apple.com)

Weitere Informationen zum Apple Developer Program sowie das Formular für die Registrierung finden Sie unter <https://developer.apple.com/programs>.

■ 1.3 Der Aufbau von iOS

Wir wissen nun also, was iOS ist, auf welchen Geräten es läuft und was diese Plattform ausmacht. Für uns als Entwickler ist aber besonders interessant, wie dieses System aufgebaut ist und wie die Architektur von iOS aussieht.

1.3.1 Die vier Schichten von iOS

iOS fußt auf insgesamt vier Schichten (den sogenannten Layern). Es gibt hier den Core OS Layer, den Core Services Layer, den Media Layer und den Cocoa Touch Layer (siehe Bild 1.5). All diese Schichten machen iOS zu dem, was es ist, und Sie als Entwickler nutzen die verschiedenen Funktionen der einzelnen Schichten, um Ihre Anwendungen zu entwickeln. Im Folgenden sollen diese Schichten einmal im Detail vorgestellt werden.

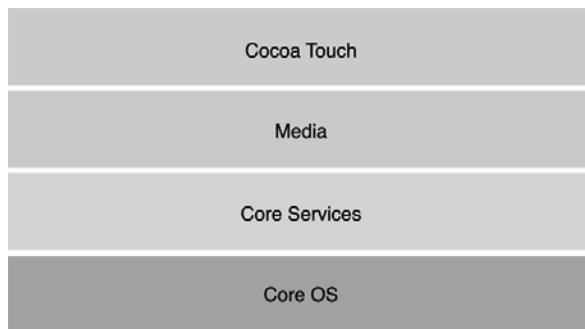


Bild 1.5

Die vier Schichten von iOS, chronologisch geordnet von unten nach oben. (Bild: Xcode-Dokumentation)

Core OS

An allererster Stelle steht das Core OS (und es ist somit die unterste Schicht des Systems). Es beherbergt die grundlegenden Funktionen und Frameworks zur Arbeit mit iOS, wobei wir die meiste Zeit über mit den darüber liegenden Schichten zu tun haben. Es ist aber wichtig zu wissen, dass es diese Funktionen und Frameworks gibt und wofür sie gut sind. Auf diesem Layer finden sich so beispielsweise das Core Bluetooth, das Security und das External Accessory Framework. Ebenfalls kümmert sich das Core OS um die Netzwerkkommunikation, Input/Output, Zugriffe auf das Dateisystem und mathematische Operationen.

So wichtig all diese Funktionen auch sind, so werden wir aber meist in unserer täglichen Arbeit – wie eben erwähnt – eher mit Frameworks arbeiten, die auf dem Core OS aufbauen und damit einen erleichterten und komfortableren Zugang auf diese Ressourcen erlauben. Sie sehen aber, dass Sie auch die Möglichkeit haben, selbst in diese Bereiche vorzudringen (sollte das nötig sein).

Core Services

Die nächste Schicht in der iOS-Systemarchitektur ist der Core Services Layer. Wie der Name bereits andeutet, stellt diese Schicht grundlegende Dienste zur Verfügung und setzt damit auf dem Core OS auf. Beispielsweise enthält dieser Layer die Peer-to-Peer-Services, mit denen Verbindungen zwischen verschiedenen iOS-Geräten via Bluetooth hergestellt werden können. Wie Sie sehen, brauchen Sie für eine solche Aufgabe nicht zwingend das Core Bluetooth Framework aus dem Core OS zu nutzen; das entsprechende Multipeer Connectivity Framework aus dem Core Services Layer setzt darauf auf und bietet bereits Lösungen für den Verbindungsaufbau zwischen verschiedenen iOS-Geräten an.

Daneben enthält Core Services noch Frameworks zur Arbeit mit iCloud, die automatische Speicherverwaltung ARC (mehr dazu in Kapitel 4, „Grundlagen der iOS-Entwicklung“), Frameworks für In-App-Käufe und mehr.

Mit am wichtigsten für uns als Entwickler ist mit Sicherheit das Foundation-Framework, welches grundlegende Datentypen wie Strings, Arrays etc., Funktionen zur Berechnung von Datum und Uhrzeit und weitere grundlegende Klassen liefert, die für die tägliche Arbeit unabdingbar sind, aber dazu später mehr.

Media

Hier wird es multimedial (im wahrsten Sinne des Wortes). Egal ob Frameworks zur Arbeit mit Bildern und Videos, zur Arbeit mit Animationen, zum Arbeiten mit Audio oder das Erstellen eigener Grafiken, im Media Layer findet sich all das Handwerkszeug, das Sie als Entwickler für alle Aufgaben rund um verschiedene Medien benötigen. Auch die Frameworks zur Arbeit mit AirPlay oder zum Verwalten eigener Game Controller finden sich auf dieser Schicht. Die meisten dieser Dinge werden aber ausschließlich für spezielle Einzelfälle benötigt, sodass wir in diesem Buch nur grundlegend auf das ein oder andere Framework näher eingehen werden (Sie werden aber sehen, dass Sie nach der Lektüre dieses Buches in der Lage sind, sich das nötige Know-how für all die anderen Media-Frameworks ohne Schwierigkeiten selbst anzueignen, versprochen).

Cocoa Touch

Mit Cocoa Touch erreichen wir die oberste Schicht in der iOS-Architektur und damit auch die Schicht, die den größten Unterschied zwischen iOS und macOS darstellt. Ganz grundlegend gesprochen stellt Cocoa Touch alle Frameworks und Funktionen zur Verfügung, um Apps für iOS zu entwickeln. Es enthält allen voran das UIKit-Framework, welches verschiedene Views und Controller zum Bau von iOS-Apps enthält und womit dann das komplette Aussehen und Design einer App umgesetzt werden kann. Es enthält auch Frameworks und Funktionen für die grundlegende App-Infrastruktur, so beispielsweise ein System, um den Nutzer mittels Notifications zu informieren, oder die Arbeit mit verschiedenen Touch-Gesten umzusetzen. Viele enthaltene Frameworks bauen auf den Funktionen der darunter liegenden Schichten auf, womit Cocoa Touch einfach nutzbare und objektorientierte Schnittstellen bietet.

Zur Entwicklung von iOS-Apps ist dieses Framework essenziell und für grundlegende Apps auch ausreichend. Dennoch ist es wichtig, als Entwickler zu wissen, aus welchen Schichten sich iOS aufbaut und welche Funktionen Ihnen als Entwickler zur Verfügung stehen.

■ 1.4 Die perfekte iOS-App

Zum Abschluss dieses ersten Kapitels schöpfe ich noch einmal aus dem Vollen und spreche direkt von der perfekten iOS-App. O je, solch ein reißerischer Titel direkt zu Beginn dieses Buches? Nun ja, auch wenn ich Ihnen hier kein Patentrezept zu einer erfolgreichen und ertragreichen App geben kann, so kann ich Ihnen aber sagen, was Sie beachten müssen, wollen Sie mit Ihrer App-Idee wirklich erfolgreich sein. Und ich teile diese Weisheiten bereits mit Ihnen hier zu Beginn dieses Buches, weil es grundlegende Aspekte sind, die Sie bei jedem Schritt der Entwicklung (und bereits davor) im Hinterkopf behalten und absolut verstehen sollten.

Zunächst einmal: Achten Sie bei jedem Schritt der Entwicklung auf die Besonderheiten der iOS-Plattform (wie in den vorigen Abschnitten beschrieben). Sie haben es hier mit Mobilgeräten zu tun, die die Nutzer in unregelmäßigen Abständen nutzen und mit denen sie meist gezielt eine bestimmte Aufgabe erfüllen möchten. Halten Sie den Nutzer nicht mit ewig langen Hinweisen oder Meldungen auf und achten Sie auf die Akku-Ressourcen und gar zu stromhungrige Funktionen, die Sie eigentlich gar nicht benötigen. Wenn Sie das beherzigen (und zwar am besten bereits in der Planungs- und Konzeptionsphase, bevor die erste Zeile Code geschrieben ist), haben Sie bereits einen essenziellen Grundstein für eine erfolgreiche App gelegt. Denn Sie machen sich keine Vorstellung, bei wie vielen Apps genau diese Aspekte bereits schieflaufen und zu entsprechend schlechten Kritiken im App Store führen.

Darüber hinaus gibt es noch eine weitere Weisheit, die zwischen Erfolg und Misserfolg Ihrer App entscheiden kann und daher auch von Beginn an berücksichtigt werden sollte: Konzentrieren Sie sich auf genau die eine Aufgabe, die Ihre App erfüllen soll! Denn dafür sind Apps für iOS gedacht: Für eine Aufgabe, die der Nutzer dank der App schnell, komfortabel und intuitiv lösen können soll. Was sich jetzt möglicherweise einfach und simpel anhört, ist in der Praxis bisweilen ein schmaler Grat und alles andere als einfach umzusetzen. Geben Sie sich daher im Vorhinein die Mühe und überlegen Sie, wie Sie sich selbst Ihre Traum-App vorstellen würden. Wie würde sie aussehen, welche Funktionen muss sie anbieten? Wie bedient sie sich, wie nutzt sie Animationen zum besseren Verständnis? Werden Sie sich über diese Dinge klar, bevor Sie beginnen, massiv Code zu schreiben, denn das geht meist nach hinten los. Und denken Sie daran, den Nutzer nicht mit unzähligen (und unnützen) Features zu überfrachten, die er gar nicht oder zumindest so gut wie gar nicht braucht; der Nutzer wird Ihnen diese Überfrachtung der App nicht danken. Deshalb betone ich es noch einmal, einfach weil es so wichtig ist und zwischen Erfolg und Misserfolg Ihrer Idee und Ihrer App entscheiden kann:



Wichtig

Konzentrieren Sie sich auf die eine Aufgabe, die Ihre App erfüllen soll ...
und dann geben Sie Vollgas!

1.4.1 iOS Human Interface Guidelines

Bevor Sie dann mit der Entwicklung loslegen, ist eines auch ganz besonders wichtig: Werden Sie mit dem Betriebssystem und dem iPhone bzw. iPad als Zielplattform vertraut. Kennen Sie diese Geräte bisher lediglich aus der Presse und der Werbung und haben sich selbst noch nie mit ihnen über einen längeren Zeitraum beschäftigt, holen Sie das erst nach. Sonst kann es leicht passieren, dass Sie die Besonderheiten der Plattform und das typische Verhalten von iOS-Apps bei Ihrer Planung und Konzeption komplett missachten und schließlich eine App entwickeln, die sich vor schlechten Kritiken kaum retten kann. Eine große Hilfe an dieser Stelle sind die iOS Human Interface Guidelines, die wertvolle Tipps zum Aufbau und zur Gestaltung Ihrer App geben. In diesem Dokument hat Apple alle Informationen zu Design und Aufbau typischer für iOS optimierter Apps zusammengefasst, es ist daher nicht nur für Einsteiger in die iOS-Entwicklung einen Blick wert (siehe Bild 1.6). Sie finden diese Guidelines online unter <https://developer.apple.com/ios/human-interface-guidelines> sowie als eigenständiges Buch in iBooks.

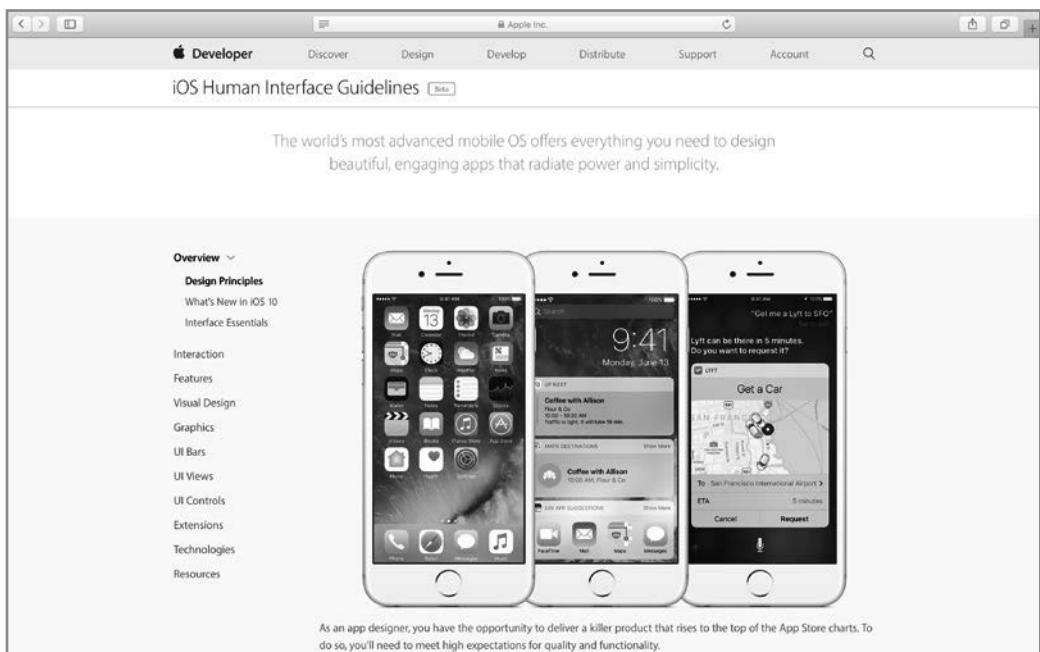


Bild 1.6 Die iOS Human Interface Guidelines sind eine sehr gute Anlaufstelle für die perfekte Gestaltung von iOS-Apps; selbst als erfahrener Entwickler lohnt ein Blick in dieses Entwicklerdokument von Apple. (Bild: <https://developer.apple.com/ios/human-interface-guidelines>)

Berücksichtigen Sie bitte diesen Ratschlag, denn ich kann aus eigener Erfahrung sagen: Erst wenn Sie wissen, wie Sie mit iPhone und iPad effektiv arbeiten und wie Sie die verschiedenen Gesten und Strukturen der Apps nutzen, werden Sie imstande sein, selbst eine großartige und innovative App für genau diese Plattformen zu entwerfen und zu entwickeln. Und schließlich soll das doch das große Ziel sein, nicht wahr?

Index

Symbolle

@class 79
@implementation 25
#import 29, 78
@interface 24
@optional 75
@protocol 74
@synthesize 45
@testable 727

A

Abfragen 17, 98
Access Control 191
Accessibility 738
Accounts 255
Action

- mit User Interface 530
- ohne User Interface 531

Action Extension 530
Activation Events 639
Alert 487
alloc 55
Any 120
AnyObject 112, 120
APN 513
App Delegate 225
App Group 615

- Registrierung 616
- Zugriff 618

App-Icon 276
App ID 763
Apple Developer Enterprise Program 7
Apple Developer Program 6
Apple Developer Program Company 7
Apple Developer Program Individual 6

Apple Developer University Program 7
Apple Human Interface Guidelines 11
Apple Push Notification Service 513
Apple Watch 621
Apple Watch-Extension 526
App Store 753
App Store Review Guidelines 777
Archiv 772
Array 94, 111
Asset-Bundle 275
Atomarität 44
Attributes Inspector 356
Auto Layout 397
Automatic Reference Counting 199
Autoresizing Masks 397

B

Background Loop 227
Background Transfers 715
Badge 487
Badge Value 486
Bedienkonzept 3
Bedingung 17
Beispielprojekte 233
Benutzereinstellungen 454
Block 80

- als Parameter 83
- als Property 85
- Aufbau 81
- globaler 86
- variablen 85

Bluetooth 3
bool 15, 94
Branch 744, 745
Breakpoint Navigator 306
Breakpoints 302

Build Phases 271
Build Rules 271
Build Settings 270
Bundle ID 776
Bundle Identifier 240

C

Caller 319
Capabilities 268
CarPlay 487
char 15
Child-View-Controller 347
Clean Build 322
Clean Code 324
CLKComplicationDataSource 701, 709
ClockKit Framework 698
Closures 137
– Trailing Closures 145
Cocoa Touch 9
Code Signing 264
Code Snippets 317
Commit 744
Complication 629, 697
– Aufbau 698
– Bestandteile 698
– Template 698
– Timeline 701
Components 260
Computed Properties 147
Connections Inspector 358
const 50
Constraint 397
Content Blocker
– Aktualisierung 536
– Konfiguration 533
Content Blocker Extension 532
Controller 340
Convenience Initializer 166
copy 44
Core Bluetooth 8
Core Data 467
– Bestandteile 469
Core Data-Editor 473
Core Location 506
Core OS 8
Core Services 9
Crashes 295
Crash-Reports 296
CSR 759

Custom Keyboard 537
– Erstellen 537
– Mehrsprachigkeit 541

D

Dangling Pointer 202
Data Source 338
Dateisystem 457
– Documents 458
– Library 458
– tmp 458
Datentypen, primitive 14
dealloc 203, 332
Debug 285
Debug Area 253
Debugging 298
Debug Navigator 308
Default Initializer 161
deinit 332
Deinitialisierung 169
Delegation 337
Deprecated Segues 389
Designated Initializer 61, 166
Development Profile 765
Devices 240, 293, 757, 763
Dictionary 94, 116
Direkte Typzuweisung 96
dispatch_once 88
dispatch_once_t 88
Document Provider 541
– aufrufen 547
Dokumentation 22, 217, 286, 449
– Download 287
Double 94
Downcasting 172
Dynamic Notification Interface 688

E

Editor Area 250
Elternklasse 24
Embed In Application 525
Entity 475
Entity Relationship Model 477
Entwickler 4
Entwickler-Account 754
Entwickler-Einstellungen 771
Entwicklerzertifikat 755
– erstellen 757
enum 52

Enumeration 52, 174
ERM 477
Error Handling Model 184
ErrorType 185
Erweiterung 70
Extensions 187, 521
– Einschränkungen 527
– erstellen 524, 530
– Funktionsweise 527
– Typen 521
External Accessory Framework 8
External Parameter Name 129

F

Failable Initializer 166
Fast Enumeration 232
File Inspector 353
FileManager 459
File Provider 545
File-Sharing 466
File's Owner 351
First Responder 352
float 15, 94
for 21
Forced Unwrapping 158
Force Touch 677
for-in 104
Foundation-Framework 29, 193
Frameworks 268
Function Type 138
Fundamental Types 93, 108, 196, 198

G

Generics 178
– Generic Function 180
– Generic Type 182
genstrings 281
Getter 41, 44
Git 743
Grand Central Dispatch 221

H

Hardware 4
Header 23
Hilfslinien 360

I

IBAction 370
IBOutlet 366
IDE 235
Identifier 756
– erstellen 762
Identity Inspector 354
if 17, 98
Image-Provider 707
iMessage Extension 549
Implementation 25
Implicit Unwrapping 159
Info 270
Info.plist 225
init 55, 58
Initialisierung 54, 160
Initializer 161
init-Methoden 60
inout 136
instancetype 58
Instanzmethoden 37, 134
Instanzvariable 38
Instruments 312
int 15, 94
Intents Extension 564
– Domain 565
– Intent 565
Intents UI Extension 588
Interactive Messaging 717
Interface Builder 236, 348
internal 191
Inverse Relationship 479
iOS 1
IPA 296
iPad 1
iPhone 1
iPod touch 1
iTunes Connect 772, 774

J

Jailbreak 4
JavaScript 90

K

Kategorien 67
keyPath 328
Key-Value-Observing 326

Klasse 23, 108, 120
 - Header 23
 - Implementation 25
 Klassenmethoden 37, 134
 Kommentare 22, 105
 Konsole 299
 Konstanten 49, 95
 Koordinatensystem 427

L

Language 240
 Launch Image 276
 Layer 8
 Lebenszyklus
 - iOS-App 226
 - WatchKit App 635
 Localizable.strings 277
 Local Notifications 498
 - Alert 498
 - Badge Value 501
 - Notification Request 508
 - registrieren 509
 - Sound 500
 - zusätzliche Informationen 503
 Local Parameter Name 129
 Locations 260
 Logik 453
 Lokalisierung 277
 Long-Look Interface 687, 688

M

Mac 4
 Magic Numbers 50
 main.m 224
 Media 9
 Mehrfachvererbung 65
 Meine Apps 774
 Member Center 754
 Memberwise Initializer 177
 Menü 676
 Methoden 30, 124
 - Implementierung 34
 - Instanzmethode 37, 134
 - Klassenmethode 37, 134
 - Methodenaufruf 36
 - Methodennamen 32
 - Method Signature Keyword 31
 - Method Type Identifier 30
 - Parameter 31

- Rückgabewert 30
 - überschreiben 65
 Mix & Match 213
 Mobilfunknetz 3
 Module 191
 Multitasking 226
 Mutable 197
 mutating 183
 MVC 323

N

Namenskonventionen 51
 Navigation Area 247
 Navigation Stack 403, 642, 660
 Nebenläufigkeit 221
 Network Link Conditioner 772
 Netzwerkkommunikation 8
 new 56
 NextStep 194
 NIB 349, 434
 nil 42
 Notification Action 491
 Notification Category 491
 NotificationCenter 333
 Notification Content Extension 592
 Notification Payload 515, 517
 Notifications 332, 485
 Notification Scene 628, 686
 - erstellen 689
 - Test 697
 Notification Service Extension 595
 Notification Type 487
 NSArray 196
 NSBundle 373
 NSData 197
 NSDate 197
 NSDictionary 197
 NSError 214
 NSExtensionItem 603
 NSLayoutConstraint 402
 NSLocalizedString 277
 NSLog 299
 NSManagedObject 469
 - Subklasse 480
 NSManagedObjectContext 469
 NSManagedObjectModel 469
 NSNotification 332
 NSNumber 196
 NSObject 195
 NSPersistentStore 470

NSPersistentStoreCoordinator 470
NSSet 196
NSString 195
NULL 42

O

Objective-C Bridging Header 213
Object Library 359
Objekt 14
Objektorientierte Programmierung 13
Open Quickly 318
Operator 17, 97
Optionals 156
Organization Identifier 239
Organization Name 239
Organizer 295
override 154

P

page-based Interface 659, 661, 676
Parent-View-Controller 347
Performance-Tests 734
Photo Editing Extension 596
Playgrounds 91
PLIST-Datei 225
Primärsprache 776
Primitive Datentypen 14
print 93, 106, 301
private 191
Product Name 239
Programmierung, objektorientierte 13
Projekt 238
Projekteinstellungen 261
Properties
- Computed Properties 147
- Property Observers 150
- Read-Only Computed Properties 149
- Stored Properties 146
- Type Properties 151
Properties (Objective-C) 39, 40
- Aufbau 40
- Direktzugriff 45
- Schreibbarkeit 44
Properties (Swift) 146
Protocol 188
Protocol (Objective-C) 73
- Protokoll zuweisen 75
- Vererbung in Protokollen 76
Protocol Type 189

Provider 515
Provisioning Profile 756
- erstellen 765
public 191
Pull 745
Punktnotation 42
Push 745
Push Notifications 512
- erstellen 517
- Versand 513
Python 90

Q

Quality of Service 520
Quick Help Inspector 354
Quick Look 212, 305

R

Refactoring 309
Relationship 477
Remote Notifications 512
Repository 744
Resource Tags 269
Retain-Cycles 202
return 35, 125
rootViewController 340
Run Loop 227
Run Without Building 321

S

Schemes 241
Schichten von iOS 8
Schleifen 17, 98
Schlüsselbundverwaltung 759
SDKs 6
Security Framework 8
Segue 392, 659
selector 329
self 41, 134
Set 94
Setter 41, 44
Shared Links Extension 602
Share Extension 601
Shortcuts 320
Short-Look Interface 687
Simulator 282
- Hardware 283
Singleton 87

Siri 564
Size Inspector 357
Skriptsprache 90
SKU 776
Software 6
Sound 487
Source Control 746, 750
Source File 191
Speicherverwaltung 43, 199
Standortabfrage 506
Static Notification Interface 688
Sticker 605
Storyboard 379
Storyboard ID 391
Strings 94, 108
strong 43, 200
Structures 177
Struktur einer App 224
Strukturen 52
Subklasse 65, 153
Subview 429
super 66, 155
Superklasse 24, 65, 153
Superview 429
Swift 89
– Grundlagen 93
– Voraussetzungen 90
Swift Standard Library 108
switch 20, 100

T

Tab-Bar 409
Tab Bar Items 412
Tabellen 662
Table Row Controller 664, 665
Target 241
Target-Action 335
Team 239, 770
Test-Case-Klasse 728
Test-Driven Development 741
Test Navigator 732
Tests 721
Text-Provider 707
Today Extension 612
Toolbar 245
Trailing Closures 145
Type Casting 170, 232
typedef 53
Typzuweisung, direkte 96

U

UIButton 443
UICollectionView 420, 449
UICollectionViewController 420
UICollectionViewFlowLayout 421
UIControl 368
UIDatePicker 446
UIImageView 445
UIInputViewController 538
UIKit 199
UILabel 443
UIImageView 447
UINavigationController 403
UINib 438
UIPickerView 445
UI Recording 740
UIScrollView 447
UISegmentedControl 444
UISplitViewController 346, 421
UIStoryboard 390
UISwitch 444
UITabBarController 409
UITableView 413, 449
UITableViewCell 414
UITableViewController 413
UITableViewDataSource 414
UITableViewDelegate 417
UI-Tests 736
UITextField 444
UITextView 448
UIUserInterfaceIdiom 378
UIUserInterfaceIdiomPad 378
UIUserInterfaceIdiomPhone 378
UIView 425
UIViewController 341
UIWebView 446
Uniform Type Identifiers 544
Unit-Tests 721
unsigned int 15
UserDefaults 454
Utilities Area 252, 353

V

Variablen 15, 95
– globale 35
Variables View 298
Vererbung 152
Veröffentlichung 772
Version Editor 750

Versionskontrolle 743
View-Controller 339
View-Controller-Hierarchien 346
View-Hierarchien 428
View Lifecycle 343
Views 425
void 30

W

Watch App 615
Watch Connectivity 713
WatchKit App 628
WatchKit Extension 627
WatchKit SDK 624
watchOS 621
weak 44, 200
Wertebereich 14
while 21, 103
Wiederverwendbarkeit 323
WKExtensionDelegate 626
WKInterfaceController 625, 637
- Attribute 644
- Initialisierung 637

WKInterfaceGroup 650, 653
WKInterfaceObject 625
WKInterfaceTable 662
WKUserNotificationInterfaceController 694
Workspace 238

X

Xcode 6, 26, 235
- Aufbau 245
- Einstellungen 254
Xcode-Generated Header 214
Xcode Server 743
XCT-Bedingung 727
XCTest 722
XCTest Framework 722
XIB 349, 435

Z

Zeiger 55, 57
Zukunftssicherheit 324