Jon Hoffman

Naučite Swift 3

„Zaronite“ u najnovije izdanje Swift programskog jezika pomoću ove knjige o naprednom Apple programiranju

kompjuter biblioteka

Packt›

Jon Hoffman

Naučite Swift 3

„Zaronite“ u najnovije izdanje Swift programskog jezika pomoću ove knjige o naprednom Apple programiranju!

kompjuter biblioteka

Packt›

Izdavač:
kompjuter biblioteka

Obalskih radnika 15, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:
Mihailo J. Šolajić

Autor: Jon Hoffman

Prevod: Slavica Prudkov

Lektura: Miloš Jevtović

Slog : Zvonko Aleksić

Znak Kompjuter biblioteke:
Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2017.

Broj knjige: 489

Izdanje: Prvo

ISBN: 978-86-7310-512-3

Learning Swift 3
by Jon Hoffman

ISBN 978-1-78646-612-9

Kratak sadržaj

POGLAVLJE 1

Prvi koraci u Swiftu 7

POGLAVLJE 2

Učenje o promenljivim, konstantama, znakovnim nizovima i operatorima 33

POGLAVLJE 3

Upotreba Swift kolekcija i tipa torke 63

POGLAVLJE 4

Kontrola toka i funkcije 87

POGLAVLJE 5

Klase i strukture 119

POGLAVLJE 6

Upotreba protokola i ekstenzija protokola 157

POGLAVLJE 7

Protokolno-orijentisano projektovanje 173

POGLAVLJE 8

Pisanje bezbednijeg koda pomoću atributa availability i rukovanjem greškama 187

POGLAVLJE 9

Prilagođeno indeksiranje 201

POGLAVLJE 10

Upotreba opcionih tipova 211

POGLAVLJE 11

Upotreba generičkih tipova 225

POGLAVLJE 12

Upotreba zatvorenog izraza 239

POGLAVLJE 13

Kombinovanje 263

POGLAVLJE 14

Paralelni rad u Swiftu 281

POGLAVLJE 15

Swiftov vodič za formatiranje i stil 299

POGLAVLJE 16

Osnovne biblioteke Swifta 311

POGLAVLJE 17

Usvajanje obrazaca projektovanja u Swiftu 337

INDEKS 369

Sadržaj

POGLAVLJE 1

Prvi koraci u Swiftu 7

Šta je Swift? 7
Funkcije Swifta 9
Playground 11
Početak rada u Playgroundu 11
iOS i OS X Playground 15
Prikazivanje slika u Playgroundu 16
Kreiranje i prikazivanje grafikona u Playgroundu 21
Šta nije Playground 22
Sintaksa Swift jezika 23
Komentari 23
Znak tačka-zarez 26
Zagrade 27
Velike zagrade za kontrolne iskaze 28
Operator dodele ne vraća vrednost 29
Razmaci su opcioni u uslovnim iskazima i iskazima dodele 30
Hello World 31
Rezime 32

POGLAVLJE 2

Učenje o promenljivim, konstantama, znakovnim nizovima i operatorima 33

Konstante i promenljive 34
Definisanje konstanti i promenljivih 35
Bezbednost tipa 36
Utvrđivanje tipa 37
Eksplicitni tipovi 38
Numerički tipovi 39
Celi brojevi 39
Pokretni zarez 42
Bulov tip 43

Znakovni tip 44
Opcione promenljive 48
Nabrajanja 53
Operatori 57
Operator dodele 57
Operatori poređenja 58
Aritmetički operatori 58
Ostatak deljenja 59
Složeni operatori dodele 59
Ternarni uslovni operator 60
Logički operator NOT 60
Logički operator AND 60
Logički operator OR 61
Rezime 61

POGLAVLJE 3

Upotreba Swift kolekcija i tipa torke 63

Tipovi Swift kolekcija 64
Promenljivost 64
Nizovi 65
Kreiranje i pokretanje nizova 65
Pristupanje elementima niza 67
Brojanje elemenata u nizu 68
Da li je niz prazan? 69
Dodavanje u niz 69
Ubacivanje vrednosti u niz 69
Zamena elemenata u nizu 70
Uklanjanje elemenata iz niza 70
Spajanje dva niza 71
Obrtanje niza 71
Preuzimanje podniza iz niza 71
Grupno menjanje nizova 72
Algoritmi za nizove 73
Map 75
Iteracije u nizu 76
Rečnici 77
Kreiranje i pokretanje rečnika 77
Pristupanje vrednostima rečnika 78
Brojanje ključa ili vrednosti u rečniku 78
Da li je rečnik prazan? 79
Ažuriranje vrednosti ključa 79
Dodavanje para ključ-vrednost 80
Uklanjanje para ključ-vrednost 80
Skup 81
Pokretanje skupa 81
Ubacivanje stavki u skup 82
Provera da li skup sadrži stavku 82
Iteracije u skupu 83

Uklanjanje stavki iz skupa ... 83
Operacije skupa ... 83
Torke ... 85
Rezime ... 86

POGLAVLJE 4

Kontrola toka i funkcije ... 87

Šta ste do sada naučili ... 87
Velike zagrade ... 88
Zagrade ... 88
Kontrola toka ... 89
Uslovni iskazi ... 89
Iskaz if ... 89
Uslovno izvršenje koda pomoću iskaza if-else ... 90
Petlja for ... 91
Upotreba petlje for-in ... 91
Petlja while ... 93
Upotreba petlje `while` ... 93
Upotreba petlje repeat-while ... 94
Iskaz switch ... 95
Upotreba iskaza case i where sa uslovnim iskazima ... 99
Filtriranje pomoću iskaza where ... 99
Filtriranje pomoću iskaza for-case ... 100
Upotreba iskaza if-case ... 102
Kontrolni iskazi prenosa ... 103
Iskaz continue ... 103
Iskaz break ... 104
Iskaz fallthrough ... 104
Iskaz guard ... 105
Funkcije ... 107
Upotreba funkcije sa jednim parametrom ... 107
Upotreba funkcije sa više parametara ... 109
Definisanje standardnih vrednosti parametara ... 109
Vraćanje višestrukih vrednosti iz funkcije ... 110
Vraćanje opcionih vrednosti ... 111
Dodavanje naziva eksternog parametra ... 112
Upotreba varijadičnih parametera ... 114
Inout parametri ... 114
Ugnežđivanje funkcija ... 115
Spajanje ... 117
Rezime ... 118

POGLAVLJE 5

Klase i strukture ... 119

Šta su klase i strukture? ... 120
Sličnosti između klasa i struktura ... 120
Razlike između klasa i struktura ... 120

Vrednosti nasuprot referentnih tipova ... 121
Kreiranje klase ili strukture ... 122
Svojstva ... 122
Uskladištena svojstva ... 123
Izračunata svojstva ... 125
Prijemnici svojstva ... 127
Metodi ... 129
Prilagođeni pokretači ... 131
Interni i eksterni nazivi parametara ... 133
Pokretači sa mogućnošću neuspeha ... 134
Nasleđivanje ... 136
Redefinisanje metoda i svojstava ... 139
Promena vrednosti metoda ... 140
Redefinisanje svojstva ... 142
Sprečavanje redefinisanja ... 143
Protokoli ... 143
Sintaksa protokola ... 144
Zahtevi svojstva ... 144
Zahtevi metoda ... 145
Ekstenzije ... 147
Upravljanje memorijom ... 149
Kako funkcioniše ARC ... 149
Ciklusi jakih referenci ... 151
Rezime ... 156

POGLAVLJE 6

Upotreba protokola i ekstenzija protokola ... 157

Protokoli kao tipovi ... 158
Polimorfizam sa protokolima ... 160
Konverzija tipova pomoću protokola ... 161
Ekstenzije protokola ... 163
Rezime ... 172

POGLAVLJE 7

Protokolno-orijentisano projektovanje ... 173

Zahtevi ... 174
Objektno-orijentisano projektovanje ... 174
Protokolno-orijentisano projektovanje ... 179
Nasleđe protokola ... 180
Kompozicija protokola ... 181
Animal-protokolno-orijentisano projektovanje ... 182
Upotreba iskaza `where` u protokolima ... 185
Rezime ... 185

POGLAVLJE 8

Pisanje bezbednijeg koda pomoću atributa availability i rukovanjem greškama 187

Rukovanje greškama pre verzije Swift 2.0 188
Izvorno rukovanje greškama 189
Predstavljanje grešaka 189
Izazivanje grešaka 191
Obrada grešaka 193
Atribut availability 197
Rezime 199

POGLAVLJE 9

Prilagođeno indeksiranje 201

Predstavljanje indeksa 201
Indeksi u Swift nizovima 202
Čitanje i pisanje prilagođenih indeksa 203
Read-only prilagođeni indeksi 204
Izračunati indeksi 205
Vrednosti indeksa 205
Eksterni nazivi za indekse 206
Multidimenzionalni indeksi 207
Kada ne treba koristiti prilagođene indekse 209
Rezime 210

POGLAVLJE 10

Upotreba opcionih tipova 211

Predstavljanje opcionih tipova 211
Potreba za opcionim tipovima u Swiftu 213
Definisanje opcionog tipa 214
Upotreba opcionih tipova 215
Prinudno odmotavanje opcionog tipa 215
Vezivanje opcionih vrednosti 216
Vraćanje opcionih vrednosti iz funkcija, metoda i indeksa 217
Upotreba opcionih vrednosti kao parametara u funkciji ili metodu 219
Vezivanje opcionih vrednosti pomoću iskaza guard 219
Opcioni tipovi u torkama 220
Ulančavanje opcionih vrednosti 220
Operator sjedinjenja nil vrednosti 222
Rezime 224

POGLAVLJE 11

Upotreba generičkih tipova 225

Uvod u generičke tipove 225
Generičke funkcije 226
Generički tipovi 230
Povezani tipovi 234

Kada ne treba koristiti generičke tipove 236
Rezime 237

POGLAVLJE 12

Upotreba zatvorenog izraza 239

Predstavljanje zatvorenog izraza 239
Jednostavni zatvoreni izrazi 240
Skraćena sintaksa za zatvorene izraze 243
Upotreba zatvorenih izraza sa algoritmom Swiftovog niza 246
Samostalni zatvoreni izrazi
i vodič za dobar stil 250
Menjanje funkcionalnosti 253
Selektovanje zatvorenog izraza na osnovu rezultata 256
Kreiranje ciklusa jakih referenci pomoću zatvorenih izraza 258
Rezime 261

POGLAVLJE 13

Kombinovanje 263

Šta je kombinovanje? 263
Kada se upotrebljava kombinovanje 264
Upotreba jezika Swift i Objective-C u istom projektu 265
Kreiranje projekta 265
Dodavanje Swift fajlova u Objective-C projekat 269
Objective-C bridging header fajl – 1. deo 272
Dodavanje Objective-C fajla u projekat 274
Messages Objective-C klasa 277
Objective-C Objective-C Bridging Header fajl –2. deo 278
MessageBuilder Swift klasa – pristupanje Objective-C kodu iz Swifta 278
Objective-C klasa – pristupanje Swift kodu
iz Objective-C-a 279
Rezime 280

POGLAVLJE 14

Paralelni rad u Swiftu 281

Paralelni rad i paralelizam 282
Grand Central Dispatch 283
Calculation tip 285
Kreiranje redova 285
Metod async, nasuprot metoda sync 290
Izvršavanje koda na glavnoj funkciji reda 290
Upotreba metoda asyncAfter 291
Upotreba tipova Operation i OperationQueue 292
Upotreba BlockOperation implementacije 292
Upotreba metoda addOperation() reda operacije 295
Kreiranje potklase za klasu Operation 296
Rezime 298

POGLAVLJE 15

Swiftov vodič za formatiranje i stil 299

Šta je vodič za stil programiranja? 299
Vodič stila 300
Nemojte upotrebljavati znak tačka-zarez na kraju iskaza 301
Ne koristiti zagrade za uslovne iskaze 301
Imenovanje 301
Prilagođeni tipovi 302
Funkcije i metodi 302
Konstante i promenljive 302
Uvlačenje reda 303
Komentari 304
Upotreba ključne reči self 305
Konstante i promenljive 305
Opcioni tipovi 305
Upotreba vezivanja opcionih vrednosti 306
Upotreba ulančavanja opcionih vrednosti, umesto vezivanja opcionih vrednosti za više odmotavanja 307
Utvrđivanje tipa 307
Upotreba skraćene deklaracije za kolekcije 307
Upotreba iskaza switch, umesto više iskaza if 308
Ne ostavljajte komentarisani kod u aplikaciji 308
Rezime 309

POGLAVLJE 16

Osnovne biblioteke Swifta 311

„Appleov" sistem za učitavanje URL-a 312
URLSession 312
URLSessionConfiguration 313
URLSessionTask 314
URL 314
URLRequest 314
HTTPURLResponse 314
REST veb servisi 315
Kreiranje HTTP GET upita 315
Kreiranje HTTP POST upita 319
Formatter 321
DateFormatter 321
NumberFormatter 323
FileManager 324
JSONSerialization 328
Raščlanjavanje JSON dokumenta 331
Kreiranje JSON dokumenta 334
Rezime 335

POGLAVLJE 17

Usvajanje obrazaca projektovanja u Swiftu 337

Šta su obrasci projektovanja? 338
Obrasci stvaranja 339
Singleton obrazac projektovanja 340
Razumevanje problema 341
Razumevanje rešenja 341
Implementiranje Singleton obrasca 342
Obrazac projektovanja Builder 344
Razumevanje problema 344
Razumevanje rešenja 344
Implementiranje obrasca Builder 344
Obrasci projektovanja struktura 350
Obrazac Bridge 350
Razumevanje problema 350
Razumevanje rešenja 351
Implementiranje obrasca Bridge 351
Obrazac Façade 355
Razumevanje problema 355
Razumevanje rešenja 355
Implementiranje obrasca Façade 356
Obrazac projektovanja Proxy 358
Razumevanje problema 358
Razumevanje rešenja 359
Implementiranje obrasca Proxy 359
Obrasci projektovanja ponašanja 361
Obrasci projektovanja Command 362
Razumevanje problema 362
Razumevanje rešenja 362
Implementiranje obrasca Command 362
Obrazac Strategy 365
Razumevanje problema 365
Razumevanje rešenja 365
Implementiranje obrasca Strategy 365
Rezime 368

INDEKS 369

UVOD

Swift je danas glavni jezik Apple razvoja. Vitalni je deo veštine svakog iOS i OS X programera; pomaže im da izgrade najimpresivnije i popularne aplikacije na App Storeu – vrste aplikacija koje su važne za svakodnevnu upotrebu iPhone i iPad korisnika.

ŠTA OBUHVATA OVA KNJIGA

U Poglavlju 1, „*Prvi koraci u Swiftu*“, prikazaćemo kako da pokrenete i upotrebite Playground za eksperimentisanje sa Swift programiranjem. Takođe ćemo opisati osnovne sintakse jezika Swift i pravilne stilove jezika.

U Poglavlju 2, „*Učenje o promenljivim, konstantama, nizovima i operatorima*“, naučićete kako se upotrebljavaju promenljive i konstante u Swiftu. Takođe ćete upoznati različite tipove podataka i način na koji mogu da se upotrebe operatori u Swiftu.

U Poglavlju 3, „*Upotreba Swift kolekcija i Tuple Typea*“, opisano je kako mogu da se upotrebe tipovi Swift kolekcija za čuvanje sličnih podataka. Ovi tipovi kolekcija su rečnici i tipovi niza. Takođe je opisano kako se upotrebljavaju tipovi podataka Cocoa i Foundation u Swiftu.

Poglavlje 4, „*Kontrola toka i funkcije*“, sadrži opis kontrole toka i funkcije u Swiftu. Veoma je važno da shvatite koncepte u ovom poglavlju pre nego što nastavite učenje. Svaka aplikacija koju pišemo, osim jednostavne aplikacije Hello World, u velikoj meri će se oslanjati na iskaze kontrole toka i funkcije.

Poglavlje 5, „*Klase i strukture*“, posvećeno je Swiftovim klasama i strukturama. Pregledaćemo koje su njihove sličnosti i razlike. Takođe ćemo opisati kontrolu pristupa i objektno-orijentisano projektovanje. Zaključićemo ovo poglavlje opisom upravljanja memorijom u Swiftu.

U Poglavlju 6, „*Upotreba protokola i ekstenzija protokola*“, detaljno ćemo opisati protokole i ekstenzije protokola, jer su protokoli veoma važni u Swift jeziku, a njihovo razumevanje će vam pomoći da pišete fleksibilan i ponovo upotrebljiv kod.

Poglavlje 7, „*Protokolno-orijentisano projektovanje*“, sadrži opis najbolje prakse protokolno-orijentisanog projektovanja u Swiftu. To će biti kratak pregled onoga što je opisano u knjizi „Protocol-Oriented Programming (POP)“.

U Poglavlju 8, „*Pisanje bezbednijeg koda sa dostupnošću i mogućnošću rukovanja greškama*“, upoznaćete Swiftovu funkciju za rukovanje greškama, koja je veoma važna za pisanje bezbednog koda. Iako nije potrebno da koristimo ovu funkciju u svojim prilagođenim tipovima, ipak nam pruža jedinstven način da rukujemo greškama i da ih ispravimo. Apple je takođe počeo da koristi ovu funkciju za rukovanje greškama u svom radnom okviru. Preporučljivo je da koristimo funkciju za rukovanje greškama u našem kodu.

U Poglavlju 9, „*Prilagođeno indeksiranje*“, opisaćemo kako može da se upotrebi prilagođeno indeksiranje u klasama, strukturama i nabrajanjima. Indeksi u Swiftu mogu da se upotrebe za pristupanje elementima u kolekciji. Takođe mogu da se definišu prilagođeni indeksi za klase, strukture i nabrajanja.

U Poglavlju 10, „*Upotreba opcionih tipova*“, saznaćete šta su opcioni tipovi i koji su načini za njihovo odmotavanje i opciono ulančavanje. Za programera koji sada uči Swift opcioni tipovi mogu da budu jedna od stavki za učenje koja najviše zbunjuje.

U Poglavlju 11, „*Upotreba generičkih tipova*“, opisaćemo kako Swift implementira generičke tipove. Generički tipovi omogućavaju da se piše veoma fleksibilan i ponovo upotrebljiv kod kojim se izbegava dupliranje.

U Poglavlju 12, „*Upotreba zatvorenog izraza*“, naučićete kako da definišete i upotrebite zatvorene izraze u kodu. Zatvoreni izrazi u Swiftu su slični blokovima koje sadrži Objective-C, ali je u njima mnogo čistiji i jednostavniji način upotrebe sintakse. Ovo poglavlje ćemo zaključiti odeljkom o načinu izbegavanja ciklusa jakih referenci pomoću zatvorenih izraza.

U Poglavlju 13, „*Upotreba funkcija Mix i Match*“, opisaćemo mešanje i uklapanje i prikazaćemo kako može Swift da se kod uključi u Objective-C projekte i Objective-C kod u Swift projekte. Za sve aplikacije i radne okvire koji su napisani u Objective-Cu, važno je da omoguće međusobnu saradnju Swifta i Objective-Ca.

U Poglavlju 14, „*Paralelni rad u Swiftu*“, prikazaćemo kako se upotrebljavaju Grand Central Dispatch i Operation Queues za dodavanje paralelnog rada i paralelizma u aplikacije. Razumevanje i poznavanje načina za dodavanje paralelnog rada i paralelizma u aplikacije može značajno da poboljša korisničko iskustvo.

U Poglavlju 15, „*Swiftov vodič za formatiranje i dodelu stila*“, definisaćemo vodič za stilove Swift jezika koji mogu da se upotrebe kao šabloni za poslovne programere koji treba da kreiraju vođice stila, jer većina preduzeća ima vođice stila za različite jezike koje koriste.

Poglavlje 16, „*Osnovne biblioteke Swifta*“, posvećeno je o upotrebi Swiftovih osnovnih biblioteka, uključujući čitanje/pisanje fajlova, osnovne oblike mreže i JSON raščlanjavanje.

U Poglavlju 17, „*Usvajanje obrazaca projektovanja u Swiftu*“, prikazaćemo kako se implementiraju neki od najosnovnijih obrazaca projektovanja u Swiftu. Obrazac projektovanja identifikuje uobičajene probleme razvoja softvera i obezbeđuje strategiju za njihovo rešavanje.

ŠTA JE POTREBNO ZA OVU KNJIGU

Da biste pratili primere u ovoj knjizi, potreban vam je Apple računar sa instaliranim OS X 10.11 ili novijim operativnim sistemom. Takođe treba da instalirate Xcode verziju 8.0 ili noviju sa Swift verzijom 3 ili novijom.

ZA KOGA JE OVA KNJIGA

Ova knjiga je namenjena programerima koji žele da „zarone“ u najnoviju verziju Swifta. Ako ste programer koji najbolje uči gledanjem i upotrebom koda, onda je ova knjiga upravo za vas. Osnovno razumevanje „Appleovih“ alatki je korisno, ali nije obavezno.

KONVENCIJE

U ovoj knjizi pronaći ćete veliki broj stilova teksta koji predstavljaju različite vrste informacija. Evo i nekih primera ovih stilova i objašnjanja njihovog značenja.

Reči koda u tekstu, nazivi tabela baze podataka, nazivi direktorijuma, nazivi fajlova, ekstenzije fajla, nazivi putanja, kratki URL-ovi, korisnički unos i Twitter identifikatori su prikazani na sledeći način: „Kada pokušamo da promenimo konstantu `speedOfLightKmSec`, biće prikazan izveštaj o grešci“.

Blok koda je postavljen na sledeći način:

```
var x = 3.14      // Double type
var y = "Hello"   // String type
var z = true      // Boolean type
```

Novi termini i važne reči su napisani masnim slovima. Reči koje vidite na ekranu, na primer, u menijima ili okvirima za dijalog, biće prikazane u tekstu na sledeći način: „Iz ovog menija ćemo želeti da selektujemo opciju Create a new Xcode project“.

Upozorenja ili važne napomene će biti prikazani u ovakvom okviru.

Saveti i trikovi prikazani su ovako.

KORISNIČKA PODRŠKA

Sada ste ponosni vlasnik „Packt“ knjige, a mi imamo mnogo štošta da vam ponudimo da bisno vam pomogli da dobijete maksimum iz svoje narudžbine.

Preuzimanje primera koda

Možete da preuzmete fajlove sa primerima koda za ovu knjigu sa vašeg naloga na adresi http://www.kombib.rs/preuzimanje/kod/Swift-3_Code.zip. Kada su fajlovi preuzeti, ekstrahujte direktorijum, koristeći najnoviju verziju:

- WinRAR / 7-Zip za Windows
- Zipeg / iZip / UnRarX za Mac
- 7-Zip / PeaZip za Linux

Preuzimanje slika u boji za ovu knjigu

Takođe smo obezbedili PDF fajl koji sadrži snimke u boji ekrana/dijagrama upotrebljenih u ovoj knjizi. Slike u boji će vam pomoći da bolje razumete promene u ispisu. Možete da preuzmete ovaj fajl sa adrese:

http://knjige.kombib.rs/images/MasteringSwift3_ColorImages.pdf.

Štamparske greške

Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, moguće su greške. Ako pronađete grešku u nekoj od naših knjiga (u tekstu ili u kodu), bili bismo zahvalni ako biste nam to prijavili. Na taj način možete da poštedite druge čitaoce od frustracija, a nama da pomognete da poboljšamo naredne verzije ove knjige. Ako pronađete neku štamparsku grešku, molimo vas da nas obavestite, tako što ćete posetiti stranicu http://www.packtpub.com/submit-errata, selektovati knjigu, kliknuti na link Errata Submission Form i uneti detalje o grešci koju ste pronašli. Kada je greška verifikovana, vaša prijava će biti prihvaćena i greška će biti aploudovana na naš web sajt ili dodata u listu postojećih grešaka, pod odeljkom Errata za određeni naslov.

Da biste pregledali prethodno prijavljene greške, posetite stranicu https://www.packtpub.com/books/content/support i unesite naslov knjige u polje za pretragu. Tražena informacija će biti prikazana u odeljku Errata.

Piraterija

Piraterija autorskog materijala na Internetu je aktuelan problem na svim medijima. Mi u „Packtu“ zaštitu autorskih prava i licenci shvatamo veoma ozbiljno. Ako pronađete ilegalnu kopiju naših knjiga, u bilo kojoj formi na Internetu, molimo vas da nas o tome obavestite i da nam pošaljete adresu lokacije ili naziv web sajta da bismo mogli da podnesemo tužbu.

Kontaktirajte sa nama na adresi copyright@packtpub.com i pošaljite nam link ka sumnjivom materijalu.

Bićemo vam zahvalni na pomoći u zaštiti naših autora, koja će omogućiti da vam pružimo vredan sadržaj.

1

Prvi koraci u Swiftu

Od svoje 12. godine, kada sam napisao svoj prvi program u BASIC programskom jeziku, programiranje je moja strast. Čak i kada je programiranje postalo moja karijera, i dalje je ostalo više strast nego posao, ali, u poslednjih nekoliko godina, ta strast je oslabila. Nisam bio siguran zašto se to dogodilo. Pokušao sam da strast povratim nekim sporednim projektima, ali ništa nije vratilo uzbuđenje koje sam ranije osećao. Zatim se desilo nešto divno - „Apple“ je objavio Swift, toliko uzbudljiv i napredan jezik da mi je ponovo učinio programiranje interesantnim!

U ovom poglavlju ćete naučiti:

- šta je Swift
- koje su funkcije Swifta
- šta su Playgroundi
- kako se upotrebljavaju Playgroundi
- koje su osnovne sintakse Swift jezika

ŠTA JE SWIFT?

Swift je „Appleov“ novi programski jezik, predstavljen na **WWDC-u** (**Worldwide Developers Conference**) 2014. godine, uz integrisano razvojno okruženje Xcode 6 i OS 8. Swift je, verovatno, najznačajnije izdanje na WWDC-u 2014, a veoma malo ljudi, uključujući „Apple“ insajdere, znalo je za postojanje ovog projekta pre nego što je objavljen.

Bilo je očaravajuće, čak i za „Appleove“ standarde, što je Swift toliko dugo bio tajna. „Apple“ je još jednom privukao veliku pažnju na WWDC-u 2015, kada su objavljeni Xcode 7 i Swift 2. Swift 2 je bio veliko poboljšanje Swift jezika. U toku konferencije *Chris Lattner* je rekao da je značajan deo poboljšanja zasnovan na direktnim povratnim informacijama koje je „Apple“ primio od programerske zajednice.

U decembru 2015. godine „Apple“ je zvanično izdao Swift kao projekat otvorenog koda i pokrenuo je veb sajt swift.org, koji je posvećen zajednici Swift otvorenog koda. Swift skladište se nalazi na „Appleovoj“ GitHub stranici (http://github.com/apple). Swift razvojno skladište (https://github.com/apple/swift-evolution) prati napredak Swifta, dokumentujući predložene promene. U razvojnom skladištu možete da pronađete listu predloženih promena koje su prihvaćene i onih koje su odbijene. Ako vas interesuje u kom pravcu se razvija Swift, treba da pregledate ovo skladište. Interesantno je napomenuti da Swift 3 sadrži nekoliko poboljšanja koje je preporučila zajednica programera.

Swift 3 **NIJE** izvorno kompatibilan sa prethodnim verzijama Swift jezika. On sadrži osnovne promene u samom jeziku i biblioteci Swift standarda. Jedan od osnovnih ciljeva autora Swifta 3 je da on bude kompatibilan na više platformi, tako da kod koji se napiše za jednu platformu bude kompatibilan na svim drugim platformama. To znači da će kod koji se napiše za Mac OS funkcionisati i na Linuxu, mada određeni radni okviri, kao što je `UIKit`, možda neće biti kompatibilni sa drugim platformama.

Razvoj Swifta je započeo 2010. godine Chris Lattner, koji je implementirao veći deo osnovne strukture jezika, za čije je postojanje znala samo nekolicina ljudi. Tek su krajem 2011. godine i drugi programeri počeli da sarađuju na projektu Swift, a u julu 2013. godine on je postao glavni fokus grupe Apple Developer Tools.

Chris Lattner je počeo da radi u „Appleu“ u leto 2005. godine. Obavljao je više funkcija u Developer Tools grupi, u kojoj je trenutno direktor i arhitekta. Na svojoj stranici (http://www.nondot.org/sabre/) navodi da je Xcodeov Playground (malo kasnije u ovom poglavlju možete da pročitate više o Playgroundu) postao njegova lična strast, zato što programiranje čini interaktivnijim i prihvatljivijim. Mi ćemo upotrebljavati često Playground u ovoj knjizi kao platformu za testiranje i eksperimentisanje. Od verzije iOS 10, moći ćemo da upotrebimo Swift Playground na iPadu.

Mogućnost upotrebe Swift Playgrounda na iPadu je za mene veoma uzbudljivo, jer olakšava učenje programiranja korišćenjem Swift jezika. Jedva čekam da pokažem svojim ćerkama kako da upotrebe Playground na svojim iPadovima.

Postoji mnogo sličnosti između Swifta i Objective-C-a. Swift preuzima čitljivost imenovanih parametara i dinamičkog modelovanja objekta Objective-C-a. Kada govorimo o Swiftu kao o modelu dinamičkog objekta, mi govorimo o mogućnosti promene tipova prilikom pokretanja. To uključuje dodavanje novih (osnovnih) i promenu/proširenje postojećih tipova.

Postoje i značajne razlike između Swifta i Objective-C-a. Swiftova sintaksa i formatiranje su mnogo sličniji sintaksi i formatiranju u Pythonu nego u Objective-C-u, ali je „Apple" zadržao velike zagrade. Swift zahteva, u stvari, upotrebu tih zagrada za kontrolne iskaze, kao što su `if` i `while`, što eleminiše greške, kao što je greška `goto` fail u „Appleovoj" SSL biblioteci.

Osim toga, Swift je izgrađen da bude brz. Na WWDC-u 2014 „Apple" je prikazao veliki broj odrednica koje su pokazale da je Swift značajno nadmašio Objective-C. Swift koristi LLVM kompajler koji je uključen u Xcode 7 za transformisanje Swift koda u visokooptimizovani izvorni kod koji je podešen tako da se dobije maksimum iz modernog „Appleovog" hardvera.

Funkcije Swifta

Kada je Apple objavio da je Swift, u stvari, Objective-C bez C-a, saznali smo samo polovinu priče. Objective-C je nadskup C-a i obezbeđuje objektno-orijentisane mogućnosti i dinamičko pokretanje za C jezik. To znači da je za Objective-C „Apple" morao da održava kompatibilnost sa C-om, što je ograničilo poboljšanja koja su mogla da budu izvršena u Objective-C jeziku. Na primer, „Apple" nije mogao da promeni način funkcionisanja iskaza `switch` i da istovremeno zadrži kompatibilnost sa jezikom C.

Pošto Swift ne treba da održava istu kompatibilnost sa C-om, kao što treba čini Objective-C, „Apple" je mogao slobodno da doda funkcije/poboljšanja u jezik. To mu je omogućilo da uključi najbolje funkcije iz mnogih najpopularnijih i modernih jezika, kao što su ObjectiveC, Python, Java, Ruby, C#, Haskell i mnogi drugi.

U sledećem grafikonu prikazana je lista nekih od najinteresantnijih poboljšanja koja uključuje Swift:

SWIFT FUNKCIJA	OPIS
automatsko utvrđivanje tipova	Swift može automatski da utvrdi tip promenljive ili konstante na osnovu inicijalne vrednosti.
generički tipovi	Generički tipovi omogućavaju da se piše kod jednom za izvršenje identičnih zadataka za različite tipove objekata dok se zadržava bezbednost tipa.
promenljivost kolekcije	Swift nema posebne objekte za promenljive ili nepromenljive kontejnere. Umesto toga, možete da definišete promenljivost definisanjem kontejnera kao konstante ili kao promenljive.
sintaksa zatvorenog izraza	Zatvoreni izrazi su samostalni blokovi funkcionalnosti koji mogu da se proslede i upotrebe u kodu.
pseudoklase	Pseudoklasa definiše promenljivu koja možda nema vrednost.
switch iskaz	Switch iskaz je drastično poboljšan funkcijama, kao što su poklapanje šablona i zaštitni uslovi; zahvaljujući njima, izbegnute su automatske greške.
višestruki povratni tipovi	Funkcije mogu da imaju višestruke povratne tipove upotrebom torki.
preklapanje operatora	Klase mogu da obezbede sopstvenu implementaciju postojećih operatora.
nabrajanja sa pratećim vrednostima	U Swiftu može da se uradi mnogo više od jednostavnog definisanja grupe srodnih vrednosti pomoću nabrajanja.

Postoji jedna funkcija koju nisam pomenuo u prethodnoj tabeli, zato što ona tehnički nije funkcija Swifta, već Xcodea i kompajlera. To je **Mix and match**. Ona omogućava da kreiramo aplikacije koje sadrže Objective-C i Swift fajlove. To omogućava da sistematski ažuriramo aktuelne Objective-C aplikacije pomoću Swift klasa i da upotrebimo Objective-C biblioteke/radne okvire u Swift aplikacijama.

Pre nego što započnemo naše „putovanje“ u predivan svet Swift programiranja, hajde da malo skrenemo sa puta i da posetimo mesto koje volim od kada sam bio dete – igralište (playground).

PLAYGROUND

Kada sam bio mali, najbolji deo dana je bio odlazak na igralište. Uopšte mi nije bilo bitno koja igra se igra; znao sam da će biti zanimljivo sve dok smo na igralištu. Kada je „Apple“ predstavio Playground kao deo Xcodea 6, pitao sam se da li „Apple“ može da učini svoj Playground toliko zanimljivim kao što su bila igrališta iz moje mladosti. Iako „Appleov“ Playground ne može da bude toliko zanimljiv kao što je bilo šutiranje lopte kada sam imao devet godina, uveo je zabavu u eksperimentisanje i igranje kodom.

Početak rada u Playgroundu

Playground je interaktivno radno okruženje koje omogućava da pišemo kod i odmah vidimo rezultate čim su promene izvršene u kodu. To znači da je upotreba Playgrounda odličan način za učenje i eksperimentisanje u Swiftu.

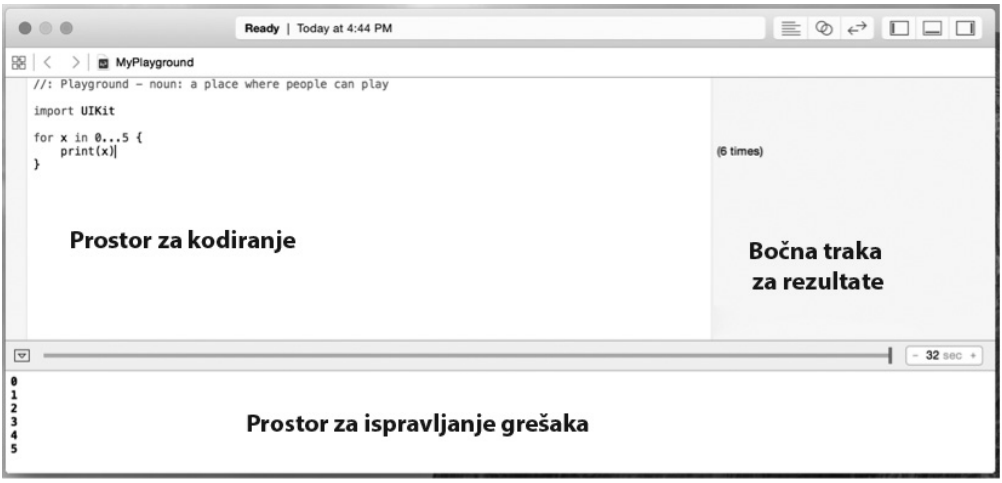
Playground neverovatno olakšava isprobavanje novih API-ja, kreiranje novih algoritama i demonstriranje kako kod funkcioniše. Mi ćemo u ovoj knjizi koristiti Playground da bismo prikazali kako naš kod funkcioniše. Stoga, pre nego što započnete programiranje u Swiftu, treba da naučite nešto više o Playgroundu i upoznate ovo radno okruženje.

Ne brinite ako Swift kod sada nema mnogo smisla; kako budete napredovali kroz ovu knjigu, smisao koda će početi sve više da se ispoljava. Sada samo imate zadatak da bolje upoznate Playground.

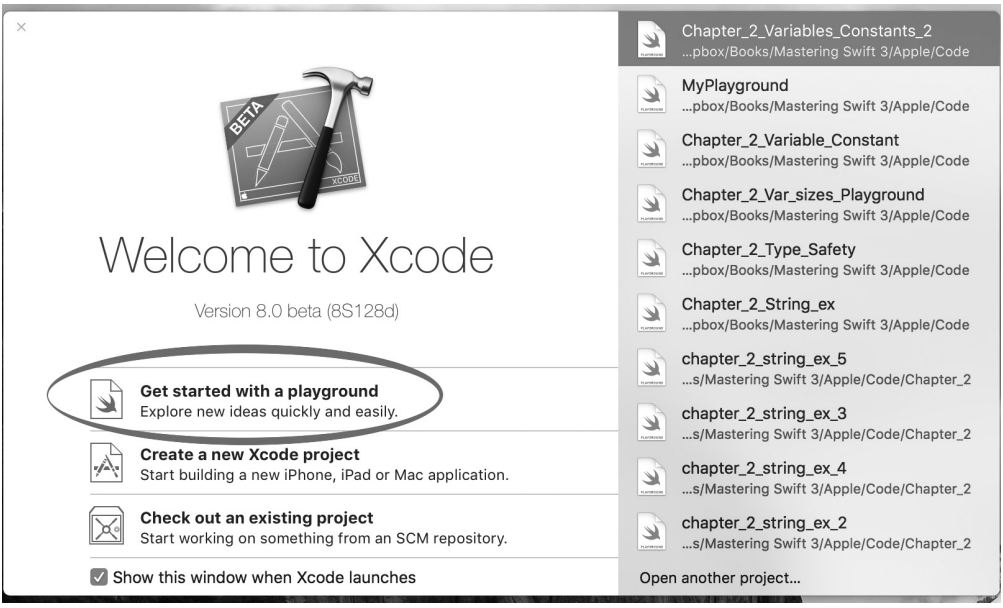
Playground može da ima nekoliko odeljaka; tri odeljka koja ćemo često koristiti u ovoj knjizi su:

- **prostor za kodiranje** - Ovo je odeljak u koji unosimo Swift kod.
- **bočna traka za rezultate** - U ovom odeljku su prikazani rezultati koda. Svaki put kada ukucate novu liniju koda, rezultati se ponovo procenjuju i bočna traka sa rezultatima se ažurira novim rezultatima.
- **prostor za ispravljanje grešaka** - Ovaj odeljak prikazuje ispis koda i može da bude veoma koristan za ispravljanje grešaka.

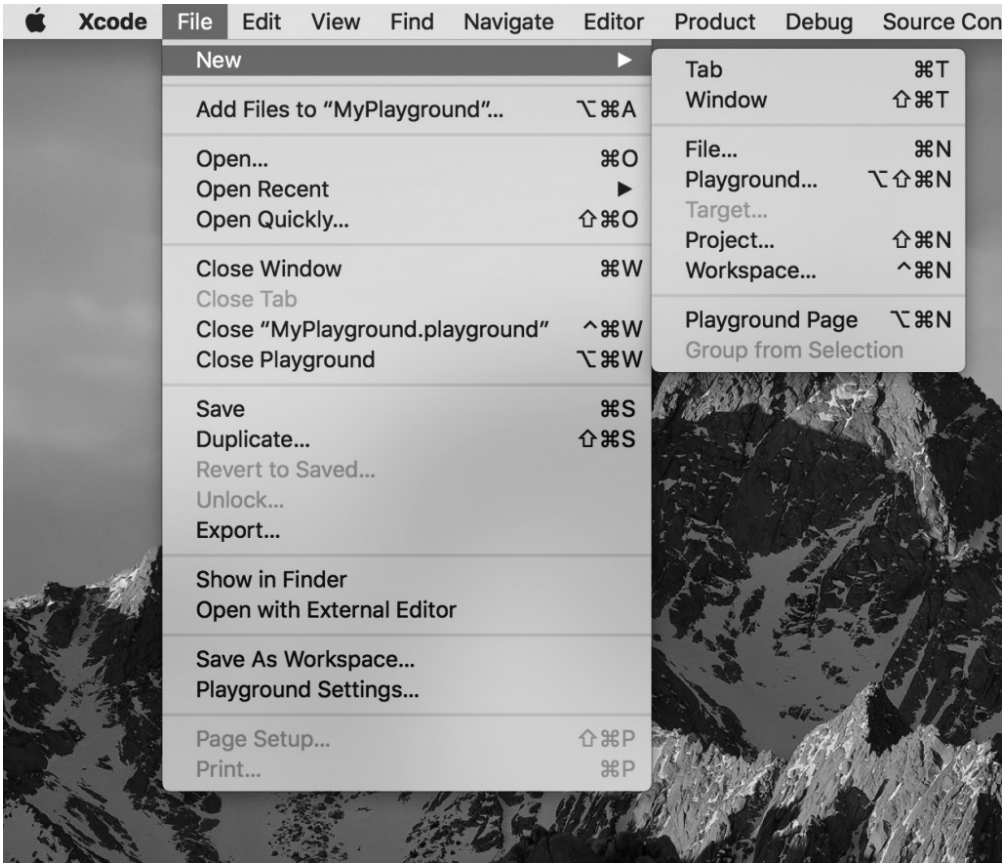
Na sledećoj slici prikazano je kako su uređeni odeljci u Playgroundu:



Hajde da otvorimo novi Playground. Prvo treba da pokrenemo Xcode. Kada je Xcode pokrenut, selektovaćemo opciju **Get started with a playground**, kao što je prikazano na sledećoj slici:

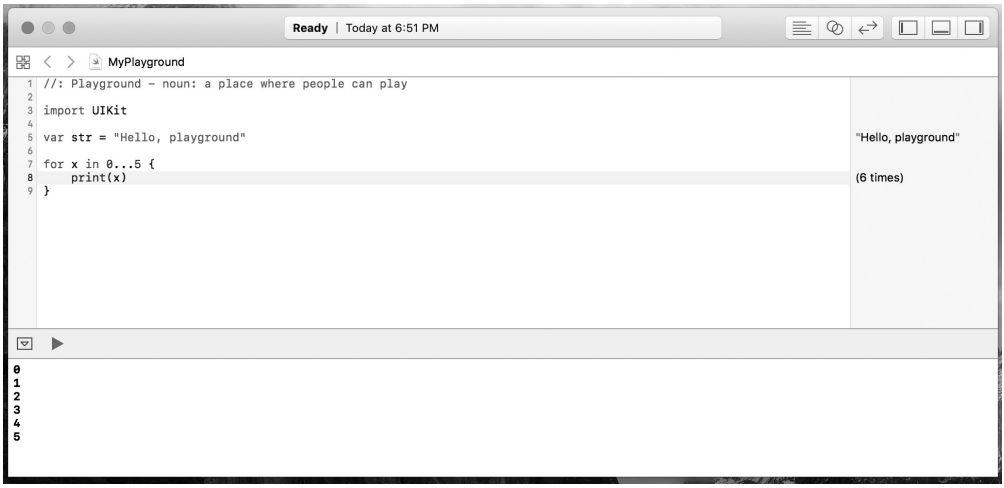


Alternativno, možemo da otvorimo Playground, tako što ćemo kliknuti na File | New u gornjoj liniji menija, kao što je prikazano na sledećoj slici:



Sada bi trebalo da vidimo ekran sličan onome koji je prikazan na sledećoj slici; ovde možemo da dodelimo naziv za Playground i da selektujemo da li je ovo iOS Playground ili OS X Playground. U većini primera iz ovog poglavlja slobodno možete da izaberete bilo koju opciju - ili iOS ili OS X, ako nije drugačije istaknuto:

Na kraju, biće zatražena lokacija za snimanje Playgrounda. Nakon što izaberemo lokaciju, Playground će se otvoriti i izgledaće slično kao na sledećoj slici:



Na prethodnoj slici možete da vidite da prostor za kodiranje Playgrounda izgleda slično prostoru za kodiranje za Xcode projekat. Ono što se razlikuje je bočna traka sa desne strane prozora. Ova bočna traka je mesto na kojem će biti prikazan rezultat koda. Kod na prethodnoj slici importuje iOS `UIKit` radni okvir i postavlja promenljivu pod nazivom `str` u Hello, `playground` niz. Sadržaj `str` niza možemo da vidimo u bočnoj traci desno od koda. Takođe smo kreirali petlju `for` koja ispisuje brojeve od 0 do 5 u konzoli.

Prema standardnom podešavanju, novi Playground ne otvara prostor za ispravljanje grešaka. Možemo ovaj prostor da otvorimo ručno, tako što ćemo istovremeno pritisnuti tastere shift + command + Y. Kasnije u ovom poglavlju ćete videti zašto je prostor za ispravljanje grešaka toliko koristan.

iOS i OS X Playground

Kada pokrenemo novi iOS Playground, Playground importuje `UIKit` (Cocoa Touch). To omogućava pristup `UIKit` radnom okviru koji obezbeđuje osnovnu infrastrukturu za iOS aplikacije. Kada pokrenemo novi OS X Playground, Playground importuje Cocoa. To omogućava pristup OS X Cocoa radnom okviru.

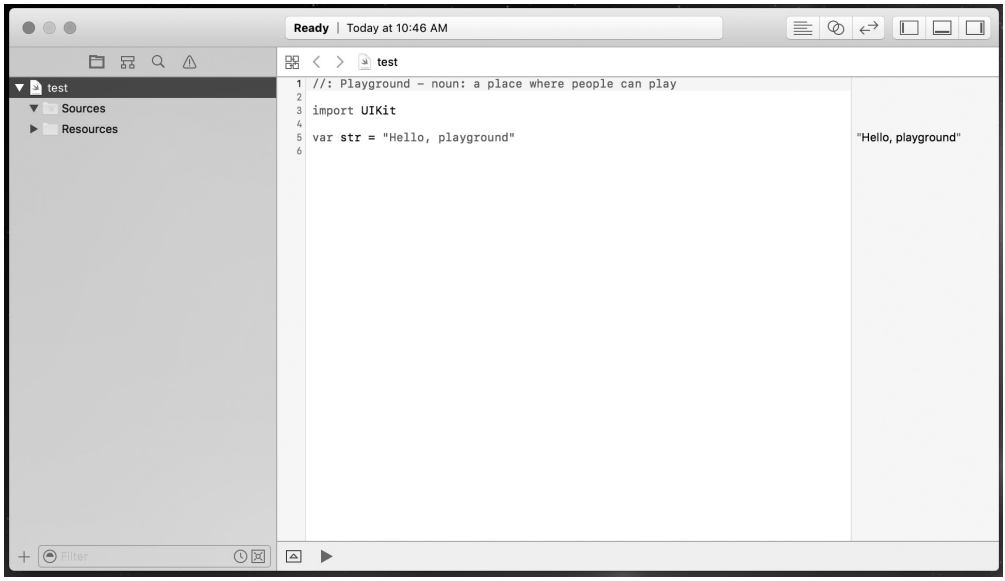
Ako želimo da eksperimentišemo sa specifičnim funkcijama UIKita ili Cocoa, potrebno je da otvorimo odgovarajući Playground. Na primer, ako želimo da otvorimo iOS Playground i da kreiramo objekat koji predstavlja boju, upotrebićemo `UIColor` objekat. Ako imamo otvoren OS X playground, upotrebićemo objekat `NSColor` za predstavljanje boje.

Prikazivanje slika u Playgroundu

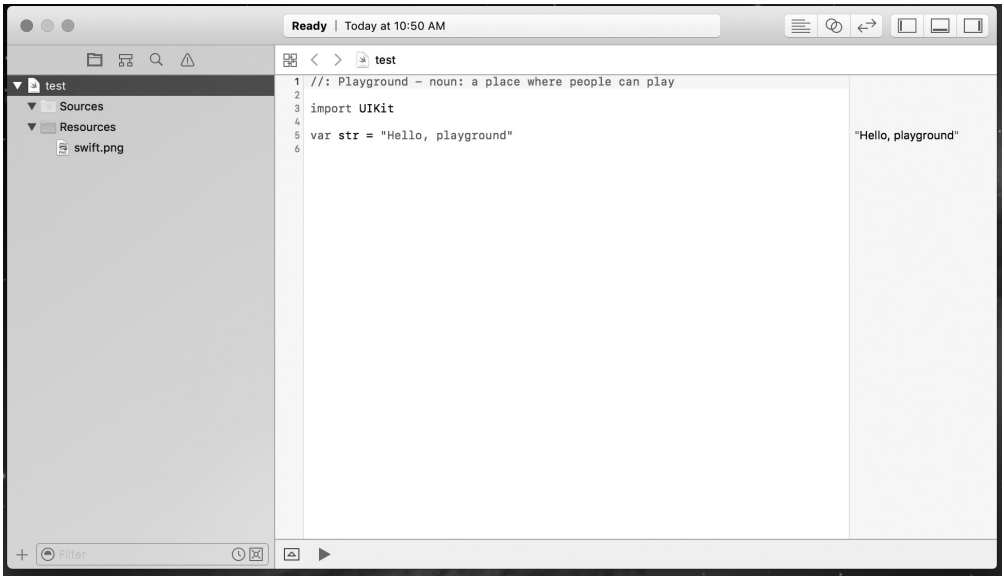
Playground je odličan za prikazivanje rezultata koda kao teksta u bočnoj traci za rezultate, ali on može da ponudi mnogo više. Možemo da prikažemo i druge stavke, kao što su slike i grafikoni. Hajde da pogledamo kako možemo da prikažemo sliku u Playgroundu. Prvo treba da učitamo sliku u izvorni direktorijum Playgrounda.

Sledeći koraci omogućavaju učitavanje slike u izvorni direktorijum:

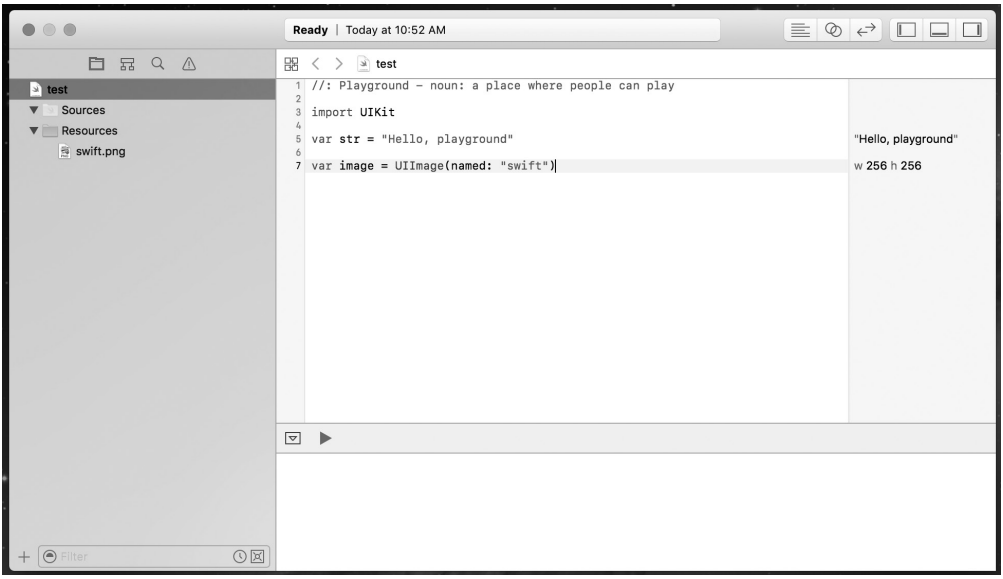
1. Započnimo tako što ćemo prikazati bočnu traku Project Navigator. Da bismo to uradili, u gornjoj liniji menija ćemo kliknuti na View | Navigators | Show Project Navigator ili ćemo upotrebiti prečicu na tastaturi command + 1. Project Navigator izgleda slično ovome:



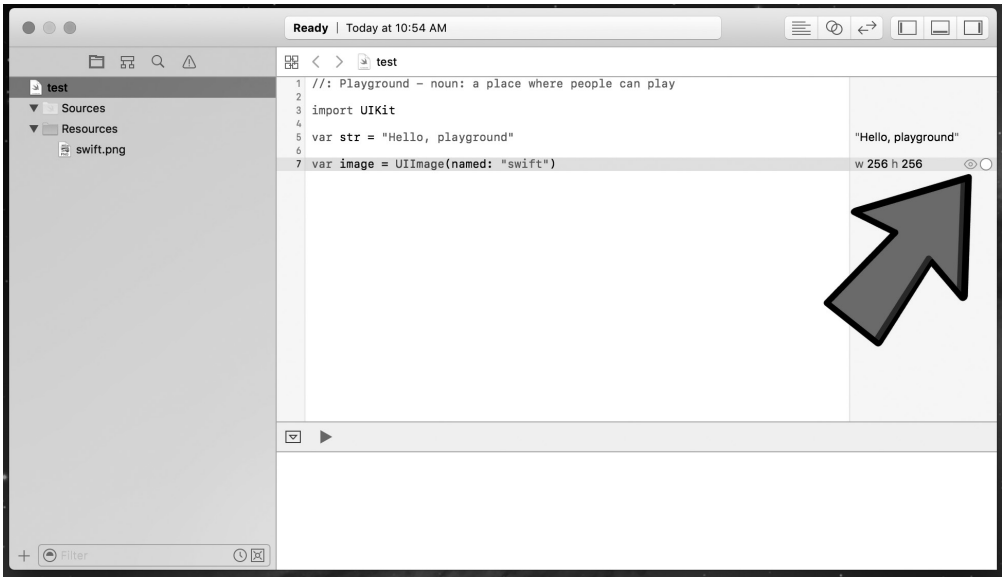
2. Kada je otvoren Project Navigator, možemo da prevučemo sliku u direktorijum Resources da bismo mogli da joj pristupimo iz koda. Kada je prevučemo i otpustimo fajl slike u direktorijum, slika će biti prikazana u direktorijumu Resources, kao na sledećoj slici:



3. Sada imamo pristup slici koja se nalazi u direktorijumu Resources unutar koda. Na sledećoj slici prikazano je kako možemo da pristupimo slici. Aktuelni kod koji koristimo za pristup sada nije toliko važan koliko je važno da znamo kako da pristupimo izvorima unutar Playgrounda:



4. Da bismo prikazali sliku, potrebno je da postavimo kursor iznad odeljka koji prikazuje širinu i visinu slike u bočnoj traci za rezultate. U našem primeru odeljak za širinu i visinu prikazuje w 256 h 256. Kada postavimo kursor iznad širine i visine, trebalo bi da vidimo dva simbola, kao što je prikazano na sledećoj slici:

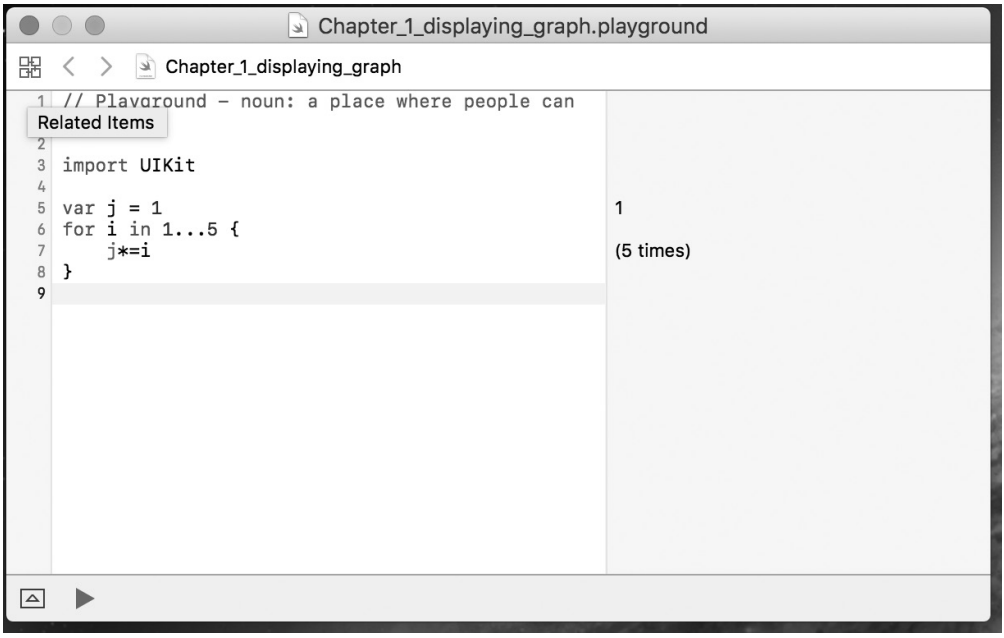


5. Možemo da kliknemo na bilo koji simbol da bismo prikazali sliku. Simbol koji ima oblik kruga sa znakom plus će prikazati sliku unutar odeljka za kodiranje u Playgroundu, dok će drugi simbol koji izgleda kao oko prikazati sliku u iskačućem prozoru van Playgrounda. Na sledećoj slici vidite šta će biti prikazano ako kliknemo na simbol kruga sa znakom plus:

Mogućnost kreiranja i prikazivanja grafikona može da bude veoma korisna kada želimo da vidimo napredak koda. Hajde da vidimo kako možemo da kreiramo i prikažemo grafikone u Playgroundu.

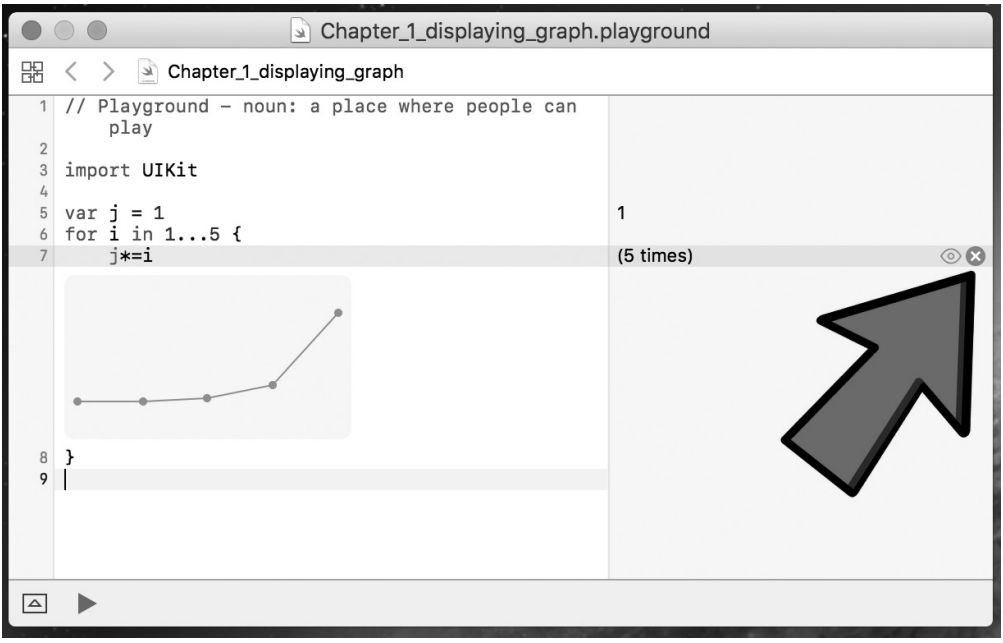
Kreiranje i prikazivanje grafikona u Playgroundu

Kreiranje i prikazivanje grafikona je stvarno korisno kada izrađujemo prototip novih algoritama, zato što grafikoni omogućavaju da vidimo vrednost promenljive tokom celog proračuna. Da biste videli kako funkcioniše prikazivanje pomoću grafikona, pogledajte sledeći Playground:



U ovom Playgroundu podesili smo promenljivu `j` na 1. Zatim smo kreirali petlju `for`, koja promenljivoj `i` dodeljuje brojeve od 1 do 5. U svakom koraku u petlji `for` podesili smo vrednost promenljive `j` na aktuelnu vrednost promenljive `j` pomnožene sa vrednošću promenljive `i`. Grafikon prikazuje vrednosti promenljive `j` u svakom koraku petlje `for`. Petlju `for` ćemo detaljnije opisati kasnije u ovoj knjizi.

Da bismo prikazali grafikon, kliknućemo na simbol oblika kruga sa tačkom. Zatim možemo da pomerimo klizač vremenske linije da bismo videli vrednosti promenljive `j` u svakom koraku petlje `for`. U sledećem Playgroundu prikazano je kako će izgledati grafikon:



Šta nije Playground

Postoji još mnogo štošta što možemo da uradimo u Playgroundu - samo smo „zagrebali površinu“ u ovom našem kratkom uvodu. Pre nego što završimo ovaj kratak uvod, hajde da pogledamo šta Playground nije da biste bolje razumeli kada ne treba da upotrebite Playground:

- **Playground ne treba da se koristi za testiranje performanse** - Perfomansa koju vidimo iz bilo kog koda koji se pokreće u Playgroundu ne predstavlja brzinu kojom će se kod pokretati u projektima.
- **Playground ne podržava interakciju sa korisnikom** - Korisnici ne mogu da vrše interakciju sa kodom koji je pokrenut u Playgroundu.
- **Playground ne podržava izvršenje na uređaju** - Ne možemo da pokrenemo kod na eksternom uređaju iz Playgrounda.

Sintaksa Swift jezika

Ako ste Objective-C programer, a nisu vam poznati moderni jezici, kao što su Python ili Ruby, kod na prethodnoj slici će vam, možda, izgledati čudno. Sintaksa Swift jezika se veoma razlikuje od Objective-C-a, koji je, uglavnom, zasnovan na Smalltalku i C-u.

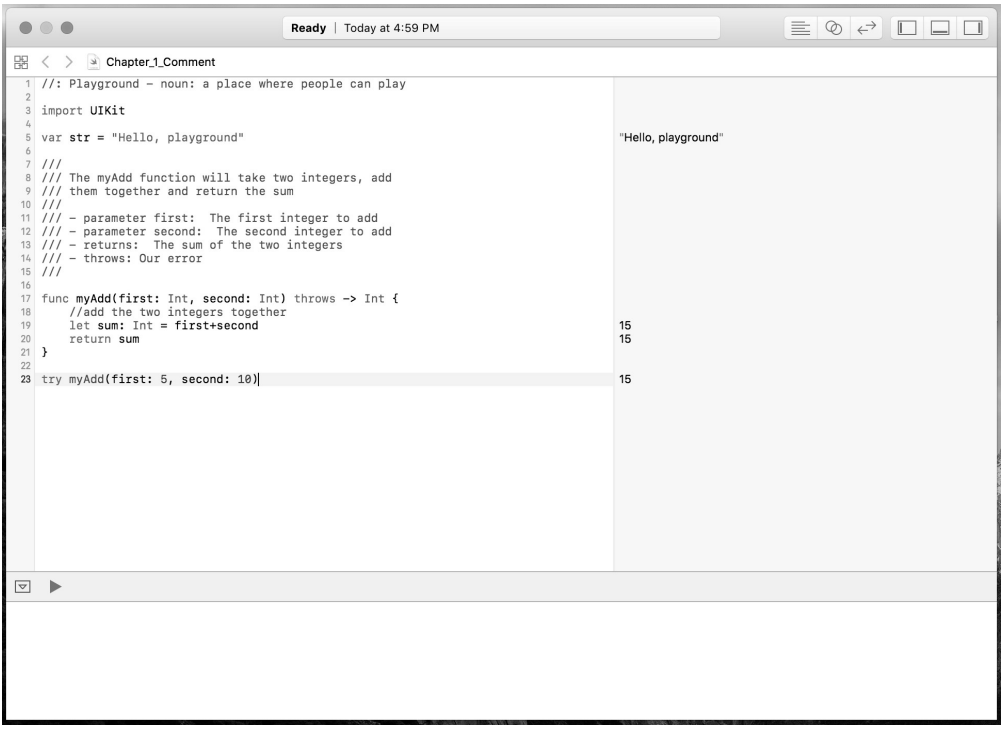
Swift jezik koristi moderne koncepte i sintaksu za kreiranje veoma konciznog i čitkog koda. Takođe je posebno istaknuto eliminisanje uobičajenih programerskih grešaka. Pre nego što započnemo rad u samom Swift jeziku, treba da pogledamo neke od osnovnih sintaksi ovog jezika.

Komentari

Pisanje komentara u Swift kodu se malo razlikuje od pisanja komentara u Objective-C kodu. I dalje možemo da upotrebimo dvostruku kosu crtu // za jednolinijske komentare i /* i */ za višelinijske komentare. Međutim, ako želimo da upotrebimo komentare za dokumentovanje koda, potrebno je da upotrebimo trostruku kosu crtu ///. Za dokumentovanje koda ćemo upotrebiti polja koja Xcode prepoznaje:

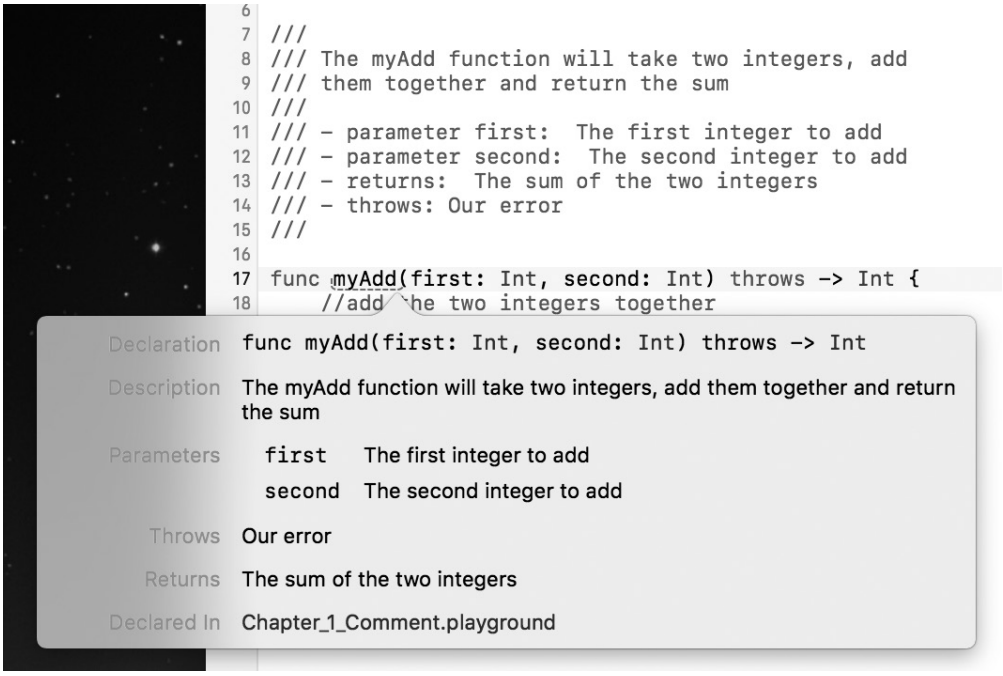
- **parametar** - Kada započnemo liniju sa - `parameter {param name}:`, Xcode prepoznaje ovo kao opis za parametar.
- **povratna vrednost** - Kada započnemo liniju sa - `returns:`, Xcode prepoznaje ovo kao opis za vraćenu vrednost.
- **odredbe** - Kada započnemo liniju sa - `throws:`, Xcode prepoznaje ovo kao opis za grešku koju metod može da izazove.

Na sledećoj slici prikazan je Playground sa primerima za jednolinijske i višelinijske komentare i prikazano je kako se upotrebljvaju polja komentara:



Da bi bili napisani dobri komentari, preporučljivo je da se upotrebe jednolinijski komentari unutar funkcije radi obezbeđenja objašnjenja koda u jednoj liniji. Zatim, treba upotrebiti višelinijske komentare van funkcija i klasa da bi bilo objašnjeno šta funkcija i klasa izvršavaju. Na prethodnoj slici Playground ilustruje dobru upotrebu komentara. Upotrebom odgovarajuće dokumentacije, kao što je urađeno na prethodnoj slici, možemo da upotrebimo funkciju dokumentacije unutar Xcodea. Ako držimo pritisnut taster option, a zatim kliknemo na naziv funkcije bilo gde u kodu, Xcode će prikazati iskačući prozor sa opisom funkcije.

Na sledećoj slici prikazano je kako može izgledati iskačući prozor:

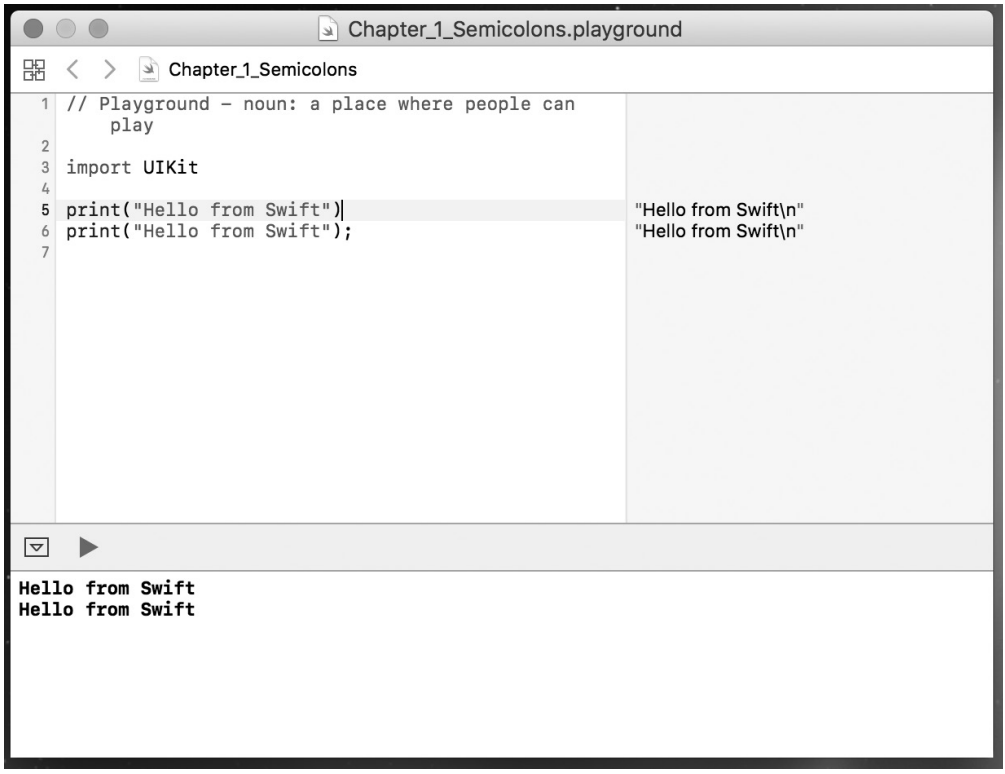


Na slici je prikazana funkcija dokumentacije Xcodea ako držimo pritisnut taster option, a zatim kliknemo na metod `myAdd()`. Možemo da vidimo da dokumentacija sadrži šest polja:

- **deklaracija** - Ovo je deklaracija funkcije.
- **opis** - Ovo je opis funkcije, onako kako se prikazuje u komentarima.
- **parametri** - Opisi parametara imaju oznaku - `Parameters`: kao prefiks u odeljku komentara.
- odredbe - Opis odredbe ima oznaku - `throws`: kao prefiks, a prikazuje koje greške su podigli metodi.
- **vraćene vrednosti** - Opis vraćene vrednosti ima oznaku - `returns`: kao prefiks u odeljku komentara.
- **fajl sa deklarisanom funkcijom** - Ovo je fajl u kojem je deklarisana funkcija da bismo mogli lako da ga pronađemo.

Znak tačka-zarez

Možda ste primetili iz primera koda do sada da nismo koristili znak tačka-zarez na kraju linija - on je opcioni u Swiftu, pa su obe linije u Playgroundu na sledećoj slici validne. Možete da vidite rezultate koda u bočnoj traci za rezultate, kao što je prikazano na sledećoj slici:



Upotreba znaka tačka-zarez je potrebna samo kada postavljamo uzastopne iskaze u jednoj liniji - na primer, ako imamo liniju kao što je sledeća:

```
print(„Hello from Swift“); return 42
```

U ovoj situaciji je potreban znak tačka-zarez između iskaza `print` i return. Preporučljivo je da ne stavljate više iskaza u istu liniju, ali, ako, ipak, želite da ih stavite, ne zaboravite da je potreban i znak tačka-zarez.

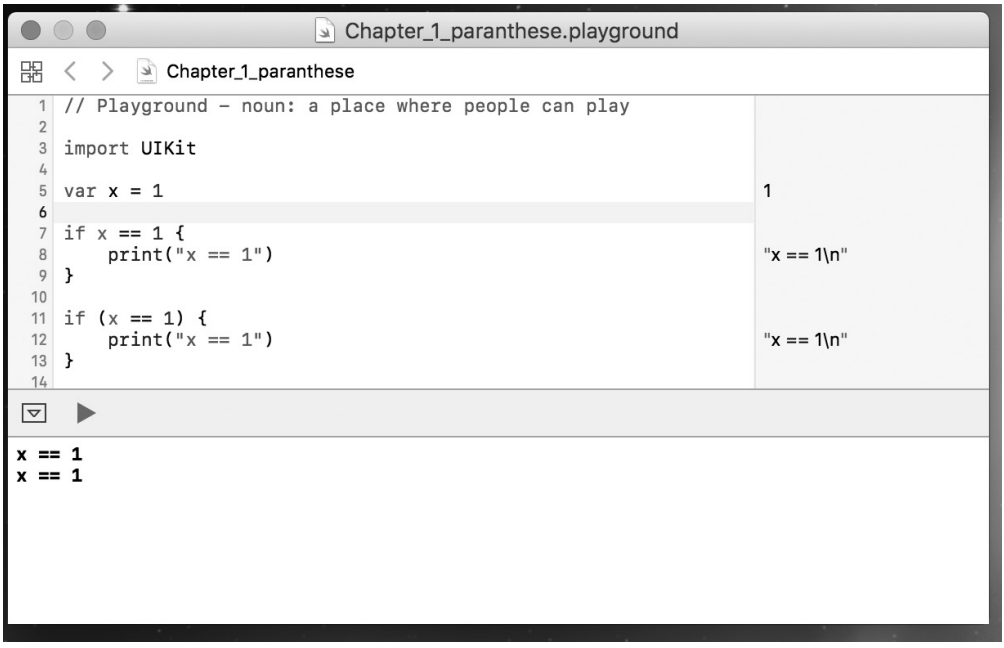
Za namenu stila je preporučljivo da ne koristite taj znak u Swift kodu. Ako nameravate da ga koristite u kodu, budite dosledni i upotrebite ga u svakoj liniji koda; međutim, Swift vas neće upozoriti u slučaju da ga izostavite. Još jednom naglašavam da upotreba tog znaka nije preporučljiva u Swiftu.

Zagrade

U Swiftu zagrade oko uslovnih iskaza su opcione; na primer, oba `if` iskaza u sledećem Playgroundu su validna. Možete da vidite rezultate koda u bočnoj traci.

Za namenu stila preporučljivo je da ne uključite zagrade u kod, osim ako imate više uslovnih iskaza u istoj liniji. Zbog čitkosti, dobra praksa je da postavite zagrade oko pojedinačnih uslovnih iskaza koji se nalaze u istoj liniji.

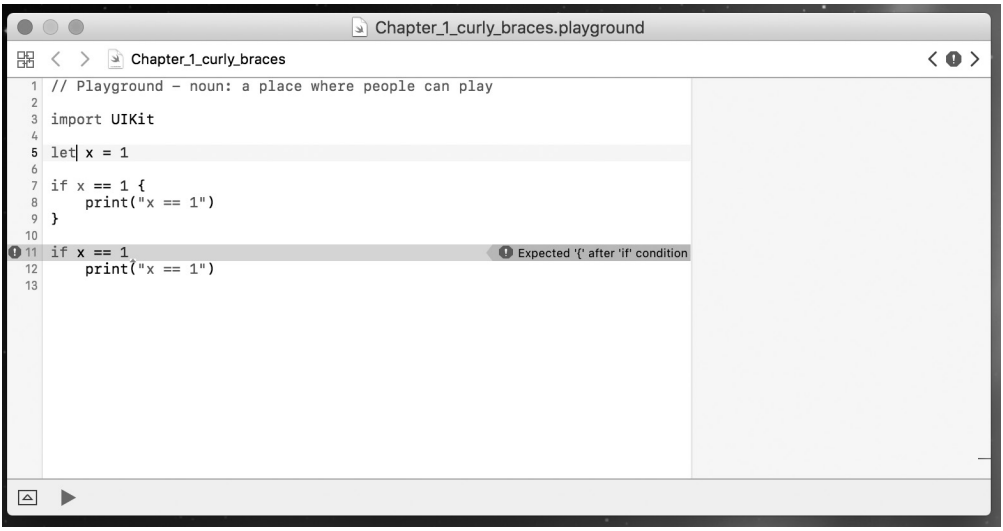
Vidite sledeći Playground za primere:



Velike zagrade za kontrolne iskaze

U Swiftu je velika zagrada potrebna iza uslovnih iskaza ili iskaza petlje. To je jedna od bezbednosnih funkcija koje su ugrađene u ovaj jezik. Verovatno je postojalo mnogo bezbednosnih grešaka koje su bile sprečene kada je programer upotrebio velike zagrade. Ove greške su mogle biti sprečene i drugim sredstvima, kao što su testiranje koda i pregled koda, ali je zahtev upotrebe velikih zagrada, po mom mišljenju, dobar bezbednosni standard.

Sledeći Playground prikazuje koju ćemo grešku dobiti ako zaboravimo da uključimo velike zagrade:



Operator dodele ne vraća vrednost

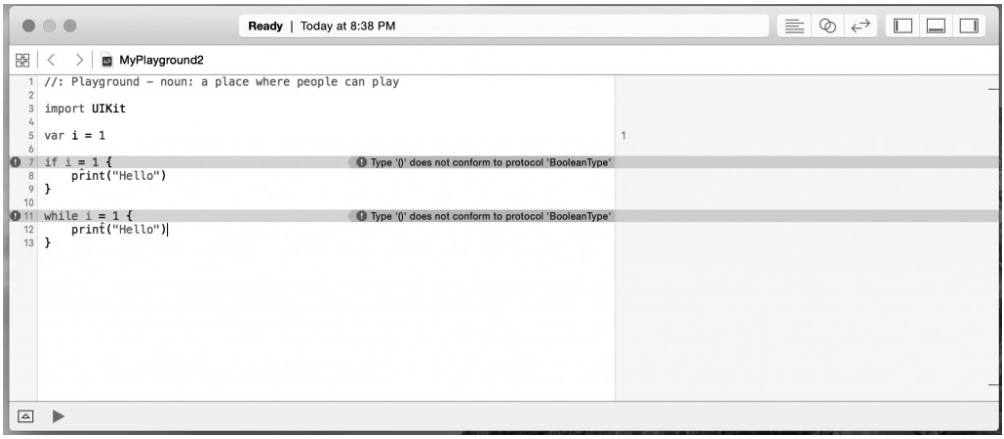
U većini drugih jezika sledeća linija koda je validna:

```
if (x = 1) {}
```

Preuzimanje primera koda

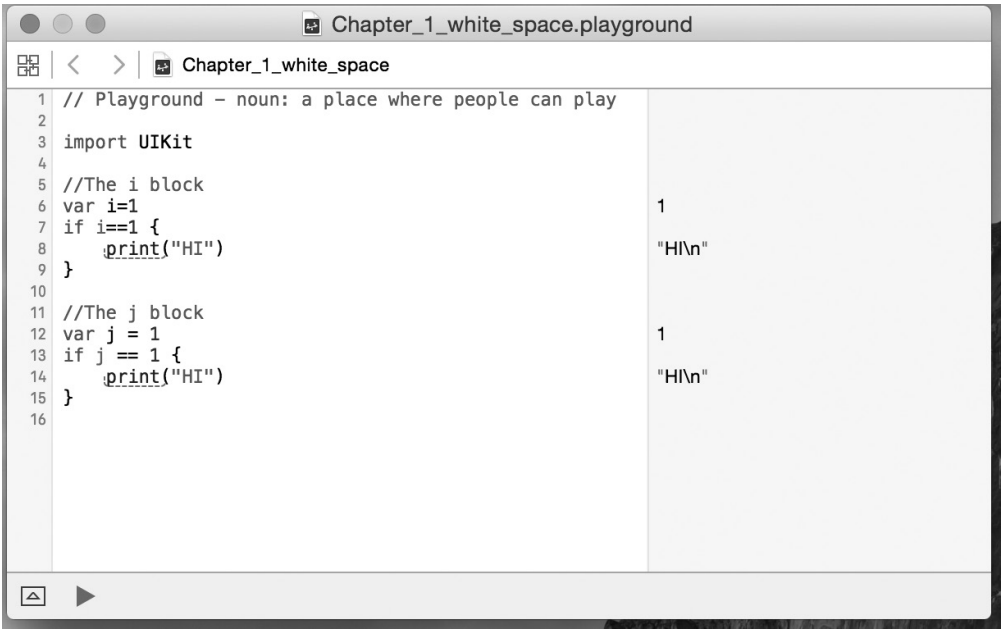
Možete da preuzmete fajlove primera koda sa vašeg naloga na adresi http://www.packtpub.com za sve knjige u izdanju „Packt Publishinga" koje ste kupili. Ako kupite ovu knjigu na nekom drugom mestu, možete da posetite stranicu http://www.packtpub.com/support i da se registrujete; dobićete fajlove direktno na e-mail.

U Swiftu ovaj iskaz nije validan. Upotreba operatora dodele (=) u uslovnom iskazu (`if` i `while`) podići će grešku. Ovo je još jedna bezbednosna funkcija koja je ugrađena u Swift. Ona sprečava da programer zaboravi drugi znak jednakosti (=) u iskazu poređenja. Ta greška je prikazana u sledećem Playgroundu:



Razmaci su opcioni u uslovnim iskazima i iskazima dodele

I za uslovne iskaze (`if` i `while`) i za iskaze dodele (=) razmaci su opcioni. Stoga, u sledećem Playgroundu kodovi `The i block` i `The j block` su validni:



Za stil preporučujem dodavanje razmaka (kao što je The j block, zbog čitkosti), ali ako izaberete jedan stil i dosledno ga koristite, bilo koji stil bi trebalo da bude prihvatljiv.

HELLO WORLD

Sve dobre računarske knjige koje su napisane za učenje programskih jezika imaju odeljak koji prikazuje korisniku kako da napiše Hello World aplikaciju. Ni ova knjiga nije izuzetak. Pokazaćemo vam kako da napišete dve različite Hello World aplikacije.

Naša prva Hello World aplikacija će biti tradicionalna Hello World aplikacija koja jednostavno ispisuje Hello World u konzoli. Započećemo kreiranjem novog Playgrounda, kome ćemo dodeliti naziv `Chapter_1_Hello_World`. Playground može da bude iOS ili OS X Playground.

U Swiftu za štampanje poruke u konzoli upotrebićemo funkciju `print()`. Ovu funkciju u njenom najosnovnijem obliku upotrebićemo za štampanje jedne poruke, kao što je prikazano u sledećem kodu:

```
print("Hello World")
```

Kada upotrebimo funkciju `print()`, obično želimo da odštampamo više, a ne samo statični tekst.

Možemo da uključimo vrednost promenljivih i/ili konstanti korišćenjem specijalne sekvence karaktera \( ) ili razdvajanjem vrednosti unutar funkcije `print()` pomoću zareza. U sledećem kodu je prikazano kako se to radi:

```
var name = "Jon"
var language = "Swift"

var message1 = " Welcome to the wonderful world of "
var message2 = "\(name) Welcome to the wonderful world of \
(language)!"

print(name, message1, language, "!")
print(message2)
```

U funkciji `print()` možemo da definišemo dva parametra koji menjaju način na koji je poruka prikazana u konzoli. Ovi parametri su separator i `terminator` parametri. Parametar separator definiše niz koji se koristi za razdvajanje vrednosti promenljivih/konstanti u funkciji `print()`. Prema standardnom podešavanju, funkcija `print()` razdvaja svaku promenljivu/konstantu pomoću razmaka. Parametar `terminator` definiše koji karakter je postavljen na kraj linije. Prema standardnom podešavanju, dodat je karakter nove linije na kraj linije.

U sledećem kodu je prikazano kako možemo da kreiramo listu razdvojenu zarezima, koja nema karakter nove linije na kraju:

```
var name1 = "Jon"
var name2 = "Kim"
var name3 = "Kailey"
var name4 = "Kara"

print(name1, name2, name3, name4, separator:", ", terminator:"")
```

Postoji još jedan parametar koji možemo da dodamo u funkciju `print()`. Njegov naziv je `toStream`. Taj parametar omogućava da preusmerimo ispis funkcije `print()`. U sledećem primeru preusmerićemo ispis u promenljivu pod nazivom line:

```
var name1 = "Jon"
var name2 = "Kim"
var name3 = "Kailey"
var name4 = "Kara"

var line = ""

print(name1, name2, name3, name4, separator:", ", terminator:"", to:&line)
```

Funkcija `print()` je bila jednostavna, korisna alatka za osnovno ispravljanje grešaka. Nova i poboljšana funkcija `print()` se može upotrebiti za mnogo više namena.

REZIME

U ovom poglavlju smo prikazali kako da pokrenete i upotrebite Playgrounde za eksperimentisanje u Swift programiranju. Takođe smo opisali osnovne sintakse jezika Swift i pravilne stilove jezika. Poglavlje smo zaključili primerima Hello World.

U sledećem poglavlju ćete naučiti kako se upotrebljavaju promenljive i konstante u Swiftu. Takođe ćete videti različite tipove podataka i način upotrebe operatora u Swiftu.