

A close-up, low-angle shot of a skateboard deck, showing the top surface with its characteristic concave shape and a dark, textured grip tape. The deck is set against a solid blue background.

Programowanie w języku Swift

BIG NERD RANCH GUIDE

A close-up, side-view shot of a skateboard truck and wheel. The truck is a light-colored, metallic material, and the wheel is a dark, textured urethane. The truck is set against a solid blue background.

Matthew Mathias, John Gallagher

Tytuł oryginału: Swift Programming: The Big Nerd Ranch Guide

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-3142-6

Authorized translation from the English language edition, entitled: SWIFT PROGRAMMING: THE BIG NERD RANCH GUIDE, ISBN 0134398017; by Matthew Mathias; and by John Gallagher; published by Pearson Education, Inc, publishing as The Big Nerd Ranch Guides.
Copyright © 2015 Big Nerd Ranch, LLC

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education Inc.

Polish language edition published by HELION S.A. Copyright © 2017.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/pswfdp.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/pswfdp>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Wprowadzenie	13
Poznanwanie języka Swift	13
Dokąd zmierza Objective-C?	13
Przygotowania	14
W jaki sposób zorganizowana jest ta książka?	14
Jak korzystać z tej książki?	15
Zadania	15
Dla bardziej dociekliwych	16
Konwencje typograficzne	16
Niezbędny sprzęt i oprogramowanie	16
Zanim zaczniemy	16

Część I Rozpoczęcie pracy

1 Rozpoczęcie pracy	21
Rozpoczęcie pracy z Xcode	21
Praca z plikiem typu playground	23
Zmienne i wyświetlanie danych w konsoli	25
Jesteś na dobrej drodze!	27
Zadanie na brązowy medal	28
2 Typy, stałe i zmienne	29
Typy	29
Stała kontra zmienna	31
Interpolacja ciągu tekstowego	32
Zadanie na brązowy medal	33

Część II Podstawy

3 Konstrukcje warunkowe	37
Konstrukcja if-else	37
Operator trójargumentowy	40
Zagnieżdżone konstrukcje if	41
Konstrukcja else if	42
Zadanie na brązowy medal	43

4	Liczby	45
	Liczby całkowite	45
	Tworzenie egzemplarza liczby całkowitej	47
	Operacje na liczbach całkowitych	49
	Dzielenie liczb całkowitych	50
	Skróty operatorów	50
	Operatory przepełnienia	51
	Konwersja między typami liczb całkowitych	53
	Liczby zmiennoprzecinkowe	54
	Zadanie na brązowy medal	56
5	Konstrukcja switch	57
	Czym jest konstrukcja switch?	57
	Zaczynamy pracę z konstrukcją switch	58
	Zakres	61
	Dołączanie wartości	62
	Klauzula where	64
	Krotka i dopasowanie wzorca	65
	Konstrukcja switch kontra if-else	68
	Zadanie na brązowy medal	70
6	Pętle	71
	Pętle for-in	71
	Pętla for case	74
	Krótką uwagę dotyczącą inferencji typu	75
	Pętla for	76
	Pętla while	77
	Pętla repeat-while	78
	Polecenia transferu kontroli (ponownie)	78
	Zadanie na brązowy medal	81
7	Ciągi tekstowe	83
	Praca z ciągami tekstowymi	83
	Unicode	85
	Skalary Unicode	85
	Odpowiednik kanoniczny	88
	Zadanie na srebrny medal	91
8	Typ Optional	93
	Typy Optional	93
	Dołączanie typu Optional	95
	Niejawne rozpakowanie typu Optional	98
	Łączenie typów Optional	99
	Modyfikacja typu Optional w miejscu	100
	Operator koalescencji	100
	Zadanie na srebrny medal	102

Część III Kolekcje i funkcje

9	Tablice	105
	Tworzenie tablicy	105
	Uzyskanie dostępu do tablicy i jej modyfikacja	107
	Porównywanie tablic	113
	Tablice niemodyfikowalne	115
	Dokumentacja	116
	Zadanie na brązowy medal	116
	Zadanie na srebrny medal	117
10	Słowniki	119
	Utworzenie słownika	119
	Zapełnienie słownika	120
	Uzyskanie dostępu do słownika i jego modyfikacja	121
	Dodawanie i usuwanie wartości	123
	Użycie pętli wraz ze słownikiem	125
	Słowniki niemodyfikowalne	126
	Konwersja słownika na tablicę	126
	Zadanie na srebrny medal	127
11	Zbiory	129
	Czym jest zbiór?	129
	Utworzenie zbioru	130
	Praca ze zbiorami	131
	Unie	131
	Część wspólna zbiorów	133
	Te same elementy w zbiorach	134
	Zadanie na brązowy medal	135
	Zadanie na srebrny medal	135
12	Funkcje	137
	Prosta funkcja	137
	Parametry funkcji	138
	Nazwy parametrów	139
	Parametry wariacyjne	140
	Wartość domyślna parametru	141
	Parametry in-out	142
	Zwrot wartości przez funkcję	144
	Funkcje zagnieżdżone i zasięg	144
	Zwrot wielu wartości	145
	Wartość zwrotna typu Optional	147
	Wcześniejsze zakończenie wykonywania funkcji	148
	Typy funkcji	149
	Zadanie na brązowy medal	149
	Zadanie na srebrny medal	150

13 Domknięcia	151
Składnia domknięcia	151
Składnia wyrażenia domknięcia	153
Funkcja jako typ wartości zwrotnej	155
Funkcja jako argument	157
Wartości przechwytywane przez domknięcie	159
Domknięcie jako typ odwołania	161
Programowanie funkcyjne	162
Funkcje wyższego rzędu	163
Zadanie na złoty medal	165

Część IV Typy wyliczeniowe, struktury i klasy

14 Typy wyliczeniowe	169
Podstawowe typy wyliczeniowe	169
Wartości pierwotne w typie wyliczeniowym	173
Metody	176
Powiązane wartości	179
Rekurencyjny typ wyliczeniowy	182
Zadanie na brązowy medal	185
Zadanie na srebrny medal	185
15 Struktury i klasy	187
Nowy projekt	187
Struktury	192
Metody egzemplarza	195
Mutowanie metod	196
Klasy	197
Klasa Monster	197
Dziedziczenie	199
Nazwy parametrów metody	203
Której konstrukcji użyć?	204
Zadanie na brązowy medal	204
Zadanie na srebrny medal	204
Dla bardziej dociekliwych — metody typu	204
Dla bardziej dociekliwych — rozwijanie funkcji	205
16 Właściwości	213
Podstawowe właściwości składowane	213
Typy zagnieżdżone	214
Opóźnione właściwości składowane	215
Właściwości obliczane	218
Metody getter i setter	219
Obserwatory właściwości	221
Właściwości typu	222

Kontrola dostępu	226
Kontrola dostępu metod getter i setter	227
Zadanie na brązowy medal	229
Zadanie na srebrny medal	229
Zadanie na złoty medal	229
17 Inicjalizacja	231
Składnia metody inicjalizacyjnej	231
Inicjalizacja struktury	232
Domyślna metoda inicjalizacyjna dla struktury	232
Niestandardowa metoda inicjalizacyjna dla struktury	233
Inicjalizacja klasy	238
Domyślne metody inicjalizacyjne klas	238
Inicjalizacja i dziedziczenie klasy	238
Wymagane metody inicjalizacyjne dla klasy	245
Deinicjalizacja	246
Metody inicjalizacyjne, których działanie może zakończyć się niepowodzeniem	248
Metoda inicjalizacyjna klasy Town, która może zakończyć działanie niepowodzeniem	248
Metody inicjalizacyjne, których działanie może zakończyć się niepowodzeniem w klasach	251
Inicjalizacja w przyszłości	252
Zadanie na srebrny medal	253
Zadanie na złoty medal	253
Dla bardziej dociekliwych — parametry metody inicjalizacyjnej	253
18 Typy wartości kontra typy odwołania	255
Semantyka wartości	255
Semantyka odwołania	258
Wartość stałej i typ odwołania	260
Użycie typów wartości i odwołania razem	263
Niemodyfikowalne typy odwołania	264
Kopowanie	264
Identyczność kontra równość	267
Którego rozwiązania powinieneś używać?	267

Część V Zaawansowany Swift

19 Protokoły	271
Formatowanie tabeli danych	271
Protokoły	276
Zgodność z protokołem	280
Dziedziczenie protokołu	282
Kompozycja protokołu	283
Mutowanie metod	284
Zadanie na srebrny medal	285
Zadanie na złoty medal	285

20	Obsługa błędów	287
	Klasy błędów	287
	Analiza leksykalna danych wejściowych	288
	Przechwytywanie błędów	296
	Przetwarzanie tablicy tokenów	297
	Obsługa błędów przez chowanie głowy w piasek	302
	Filozofia obsługi błędów w języku Swift	305
	Zadanie na brązowy medal	306
	Zadanie na srebrny medal	307
	Zadanie na złoty medal	307
21	Rozszerzenia	309
	Rozbudowa istniejącego typu	309
	Rozszerzanie własnego typu	311
	Użycie rozszerzenia w celu zapewnienia zgodności z protokołem	312
	Dodanie metody inicjalizacyjnej w rozszerzeniu	313
	Zagnieżdżone typy i rozszerzenia	314
	Rozszerzenia wraz z funkcjami	315
	Pierwsze zadanie na brązowy medal	317
	Drugie zadanie na brązowy medal	317
	Zadanie na srebrny medal	317
22	Generyki	319
	Struktury danych generyków	319
	Funkcje i metody generyków	322
	Ograniczenia typu	324
	Powiązane typy protokołów	325
	Ograniczenia typu i klauzule where	329
	Zadanie na brązowy medal	331
	Zadanie na srebrny medal	331
	Zadanie na złoty medal	331
	Dla bardziej dociekliwych — lepsze zrozumienie typu Optional	331
	Dla bardziej dociekliwych — polimorfizm parametryczny	332
23	Rozszerzenia protokołu	333
	Modelowanie ćwiczenia fizycznego	333
	Rozszerzenie protokołu ExerciseType	335
	Rozszerzenie protokołu wraz z klauzulą where	336
	Domyślne implementacje rozszerzeń protokołu	338
	Nadawanie nazw — opowieść ku przestrodze	340
	Zadanie na brązowy medal	342
	Zadanie na złoty medal	342
24	Zarządzanie pamięcią i ARC	343
	Alokacja pamięci	343
	Cykle silnych odwołań	344
	Cykle odwołań w domknięciach	350
	Zadanie na brązowy medal	354
	Zadanie na srebrny medal	354
	Dla bardziej dociekliwych. Czy można pobierać licznik odwołań egzemplarza?	354

25	Równość i porównywalność	357
	Zgodność z protokołem Equatable	357
	Zgodność z protokołem Comparable	360
	Dziedziczenie logiki porównania	363
	Zadanie na brązowy medal	364
	Zadanie na złoty medal	364
	Zadanie na platynowy medal	364
	Dla bardziej dociekliwych — własne operatory	365

Część VI Aplikacje działające na podstawie zdarzeń

26	Pierwsza aplikacja Cocoa	371
	Rozpoczęcie pracy nad aplikacją VocalTextEdit	372
	Architektura MVC	373
	Przygotowanie kontrolera widoku	374
	Przygotowanie widoków w module Interface Builder	377
	Dodanie przycisków rozpoczynających i kończących odczyt treści	378
	Dodanie widoku tekstowego	379
	System Auto Layout	382
	Utworzenie połączeń	384
	Zdefiniowanie par cel-akcja dla przycisków programu VocalTextEdit	384
	Połączenie outletu widoku tekstowego	385
	Wydobycie głosu przez VocalTextEdit	387
	Zapisywanie i wczytywanie dokumentów	389
	Rzutowanie typu	391
	Zapis dokumentów	392
	Wczytywanie dokumentów	394
	Uporządkowanie kodu zgodnie z architekturą MVC	397
	Zadanie na srebrny medal	399
	Zadanie na złoty medal	399
27	Pierwsza aplikacja iOS	401
	Rozpoczęcie pracy nad aplikacją iTahDoodle	402
	Przygotowanie interfejsu użytkownika	402
	Połączenie ze sobą elementów interfejsu użytkownika	410
	Modelowanie listy rzeczy do zrobienia	412
	Przygotowanie egzemplarza UITableView	416
	Zapisywanie i wczytywanie listy rzeczy do zrobienia	419
	Zapisywanie listy rzeczy do zrobienia	419
	Wczytywanie listy rzeczy do zrobienia	421
	Zadanie na brązowy medal	422
	Zadanie na srebrny medal	423
	Zadanie na złoty medal	423

28 Współdziałanie języków Swift i Objective-C	425
Projekt w Objective-C	425
Utworzenie aplikacji	427
Dodanie kodu w języku Swift do projektu Objective-C	435
Dodawanie osób	438
Dodanie klasy Objective-C	447
Zadanie na srebrny medal	454
Zadanie na złoty medal	454
29 Zakończenie	455
Co dalej?	455
Bezwstydne wtyczki	455
Zaproszenie	456
Skorowidz	457

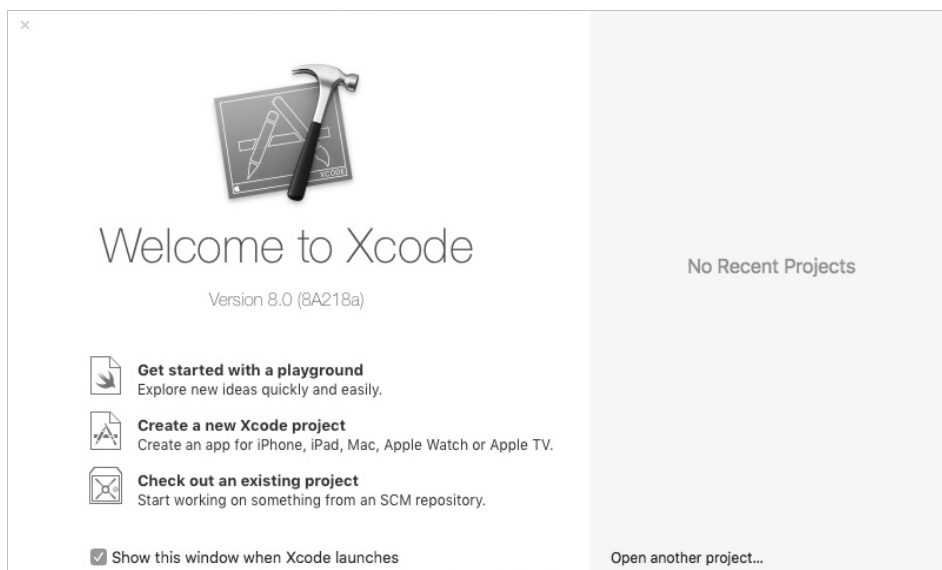
Rozpoczęcie pracy

W tym rozdziale zajmiemy się przygotowaniem środowiska pracy oraz zrobimy małą wycieczkę po narzędziach używanych codziennie przez programistów macOS i iOS. Ponadto utworzymy pierwsze fragmenty prostego kodu, aby jeszcze dokładniej zobaczyć, czym są Swift i Xcode.

Rozpoczęcie pracy z Xcode

Jeżeli jeszcze tego nie zrobiłeś, pobierz i zainstaluj narzędzie Xcode, które znajdziesz w sklepie App Store. Upewnij się o pobraniu wydania Xcode 8 lub nowszego.

Po zainstalowaniu Xcode uruchom to narzędzie. Na ekranie powitalnym zobaczysz kilka opcji, między innymi możliwość utworzenia nowego pliku playground (opcja *Get started with a playground*) oraz nowego projektu (opcja *Create a new Xcode project*), jak pokazano na rysunku 1.1.



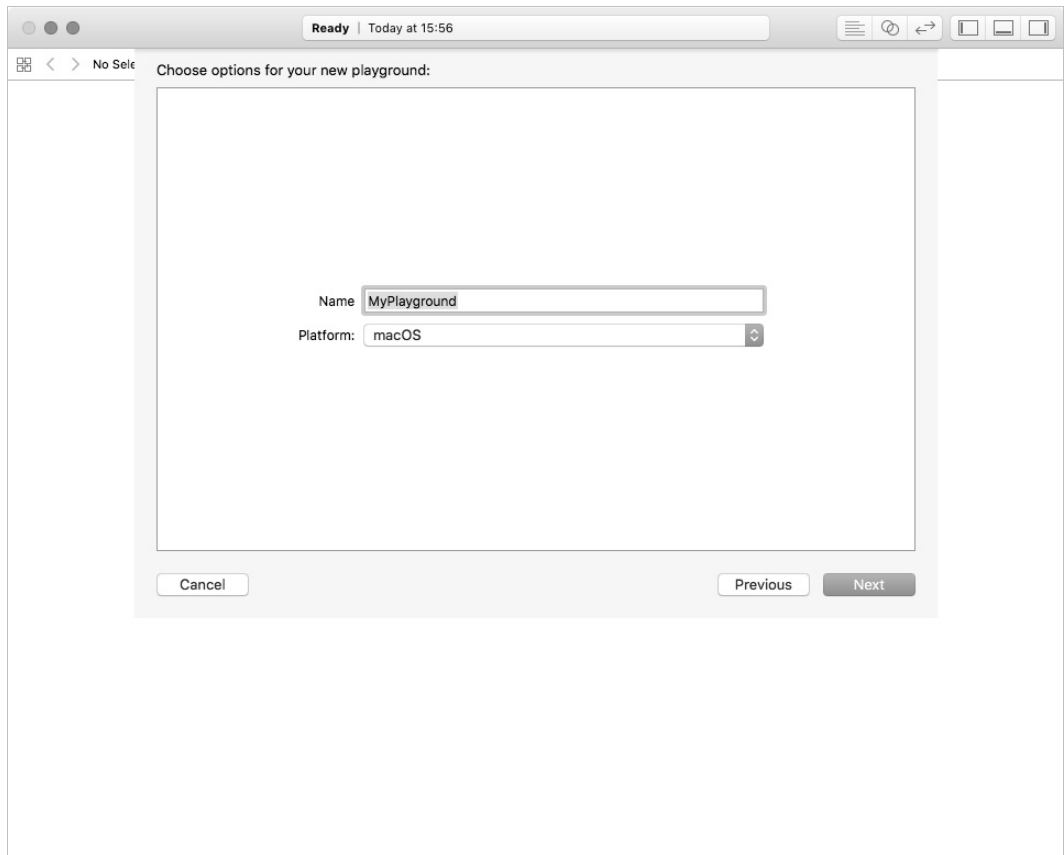
Rysunek 1.1. Ekran powitalny wyświetlany po uruchomieniu Xcode

Plik typu playground to nowa funkcja wprowadzona w Xcode 6. Stanowi rodzaj interaktywnego środowiska przeznaczonego do szybkiego tworzenia i oceny kodu w języku Swift. Plik playground nie wymaga kompilacji i uruchamiania pełnego projektu. Zamiast tego kod Swift jest analizowany i wykonywany „w locie”, więc plik playground doskonale sprawdza się podczas testowania i eksperymentowania z językiem, ponieważ oferuje świetne do tego celu lekkie środowisko. Z plików tych będziesz bardzo często korzystać podczas lektury książki. Gdy wprowadzisz kod Swift w pliku playground, od razu otrzymasz wynik jego wykonania.

Poza plikami playground utworzysz także natywne narzędzia działające na poziomie powłoki. Dlaczego nie pozostaniemy wyłącznie przy plikach playground? Ponieważ wówczas pominęlibyśmy wiele funkcji oferowanych przez Xcode i nie poznałbyś lepiej środowiska IDE. Skoro chcesz zostać programistą, będziesz spędzał dużą ilość czasu na pracy z Xcode i dlatego dobrze jest jak najwcześniej poznać używane narzędzie.

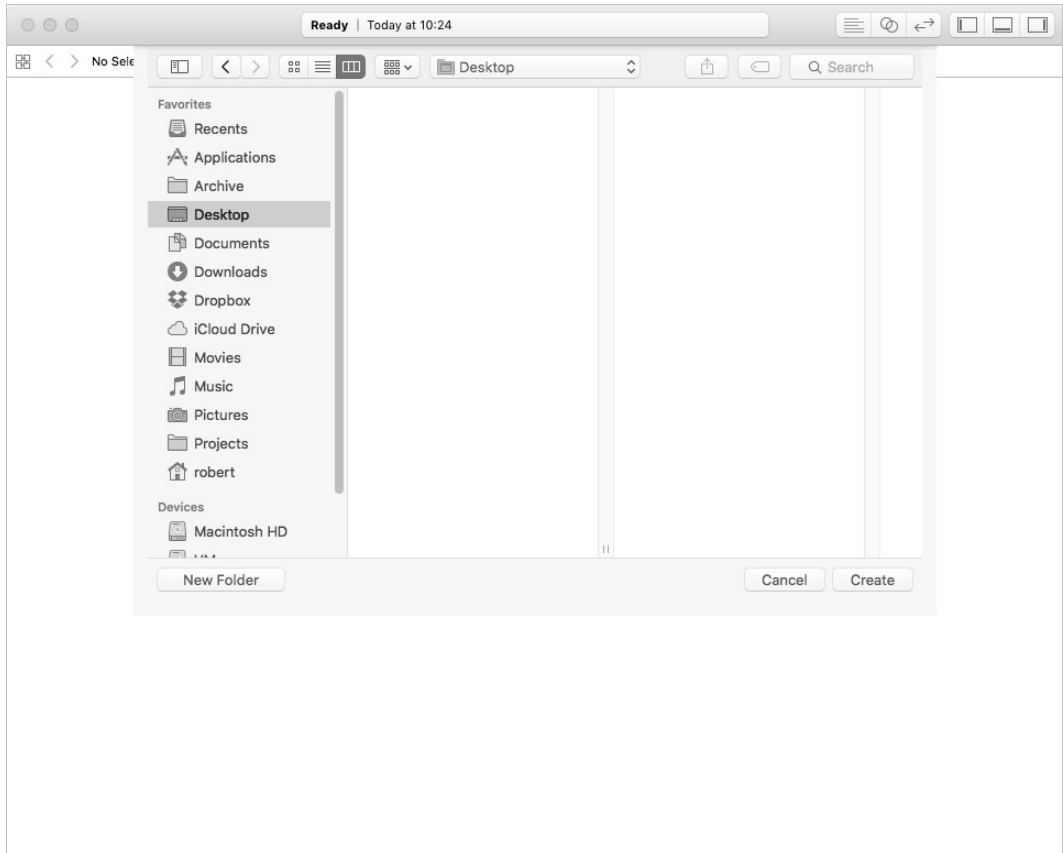
Na ekranie powitalnym kliknij *Get started with a playground*.

Następnie podaj nazwę `MyPlayground` dla tworzonego pliku. Kiedy pojawi się pytanie o wybór platformy (macOS lub iOS), z rozwijanego menu wybierz opcję *macOS* (rysunek 1.2), nawet jeśli jesteś programistą iOS. Omawiane w książce funkcje języka Swift są takie same na obu platformach. Teraz kliknij przycisk *Next*.



Rysunek 1.2. Nadanie nazwy plikowi typu playground

W kolejnym kroku będziesz musiał wskazać miejsce zapisu tworzonego pliku. Ponieważ podczas lektury książki będziemy tworzyć kolejne pliki, więc dobrze jest przygotować dla nich oddzielny katalog. Wybierz dogodny katalog i kliknij przycisk *Create* (rysunek 1.3).

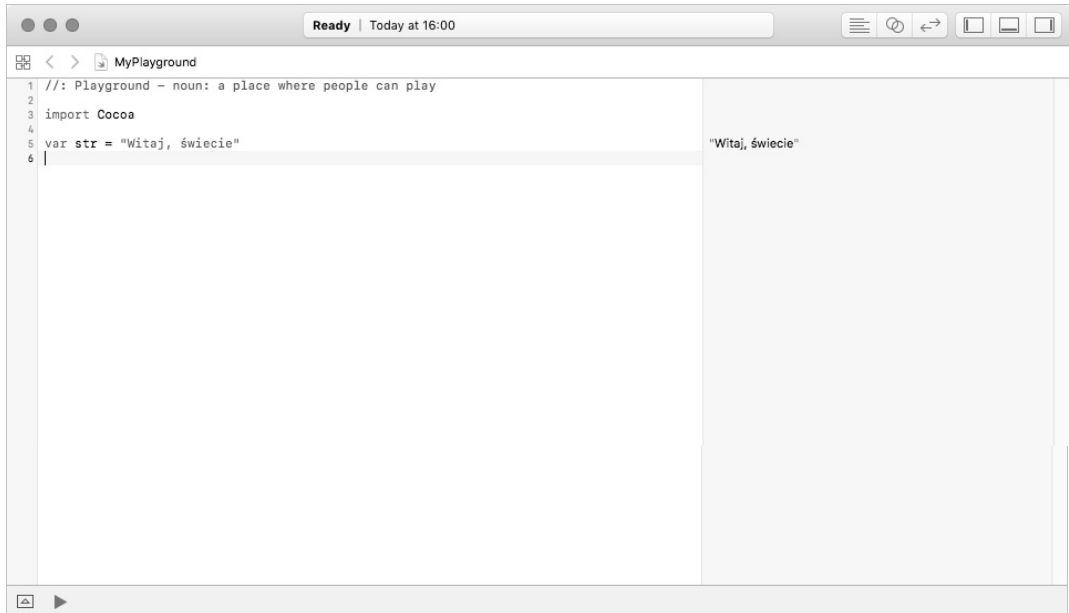


Rysunek 1.3. Zapis pliku typu playground

Praca z plikiem typu playground

Jak możesz zobaczyć na rysunku 1.4, okno utworzonego pliku playground składa się z dwóch części. Po lewej stronie mamy edytor kodu źródłowego Swift, natomiast po prawej widzimy wynik wykonania tego kodu. Polecenia wpisane w edytorze są analizowane i wykonywane, o ile to możliwe, po wprowadzeniu jakiegokolwiek zmiany w kodzie źródłowym. Wynik działania kodu możesz zobaczyć po prawej stronie okna.

Spójrz dokładnie na wyświetlone okno. Zwróć uwagę na fakt, że pierwszy wiersz kodu jest w kolorze zielonym i rozpoczyna się od dwóch ukośników (`//`). Te ukośniki wskazują kompilatorowi, że dany wiersz jest **komentarzem**, a Xcode wyświetla komentarze w kolorze zielonym.



Rysunek 1.4. Nowo utworzony plik typu playground wyświetlony w oknie Xcode

Programiści używają komentarzy w charakterze osadzonej dokumentacji oraz notatek wyjaśniających działanie danego fragmentu kodu. Usuń dwa ukośniki na początku wiersza. Kompilator wygeneruje komunikat informujący o braku możliwości przetworzenia wyrażenia. Przywróć usunięte ukośniki, wykorzystując do tego celu przydatny skrót klawiszowy *Command+I*.

Tuż pod komentarzem znajduje się polecenie importujące framework Cocoa. Użyte tutaj polecenie `import` oznacza, że w danym pliku playground mamy pełny dostęp do całego interfejsu programowania aplikacji (ang. *application programming interface*, API) zdefiniowanego we frameworku Cocoa. (API przypomina zbiór definicji przeznaczonych do tworzenia programów).

Kolejne polecenie ma postać `var str = "Witaj, świecie"`. Tekst ujęty w cudzysłów zostanie wyświetlony po prawej stronie okna, czyli w sekcji wyników, i będzie miał postać "Witaj, świecie". Przeanalizujmy dokładnie ten wiersz kodu.

Po lewej stronie znaku równości znajduje się ciąg tekstowy `var str`. W języku Swift słowo kluczowe `var` służy do zadeklarowania zmiennej. To jest niezwykle ważna koncepcja, którą dokładnie się zajmiemy w następnym rozdziale. Teraz wystarczy zapamiętać, że zmienna przedstawia pewną wartość, która może ulec zmianie.

Z kolei po prawej stronie znaku równości mamy ciąg tekstowy "Witaj, świecie". W Swiftcie cudzysłów wskazuje na użycie typu **String**, czyli uporządkowanej kolekcji znaków. W wygenerowanym szablonie ta nowa zmienna ma nazwę `str`, choć zmiennym można przypisywać dowolne nazwy. (Oczywiście istnieją pewne ograniczenia. Spróbuj zmienić nazwę `str` na `var`. Co się stało? Jak sądzisz, dlaczego zmiennej nie możesz nadać nazwy `var`? Zanim przejdziesz dalej, koniecznie przywróć zmiennej nazwę `str`).

Teraz powinieneś już lepiej rozumieć tekst wyświetlany w sekcji po prawej stronie okna — to jest po prostu ciąg tekstowy przypisany zmiennej `str`.

Zmienne i wyświetlanie danych w konsoli

Wspomniany wcześniej **String** to tak zwany **typ**. Mówimy więc, że zmienna `str` jest „egzemplarzem typu **String**”. Typy opisują określone struktury przeznaczone do przedstawiania danych. W języku Swift znajdziesz wiele typów, które będą pojawiały się w książce. Poszczególne typy mają konkretne możliwości (co dany typ może zrobić z danymi) oraz ograniczenia (czego dany typ nie może zrobić z danymi). Na przykład typ **String** został zaprojektowany do pracy z uporządkowanymi kolekcjami znaków i zawiera wiele zdefiniowanych funkcji przeznaczonych do pracy z nimi.

Przypomnij sobie, że `str` to **zmienna**. Mamy więc możliwość zmiany wartości przypisanej zmiennej. Teraz dołączymy wykrzyknik na końcu przykładowego ciągu tekstowego, aby tym samym otrzymać poprawne zdanie. (Kiedy do przykładu dodajemy nowy kod, będzie on przedstawiony pogrubioną czcionką. Natomiast kod przeznaczony do usunięcia będzie przekreślony). Kod po wprowadzonych zmianach możesz zobaczyć na listingu 1.1.

Listing 1.1. Właściwa interpunkcja w zdaniu

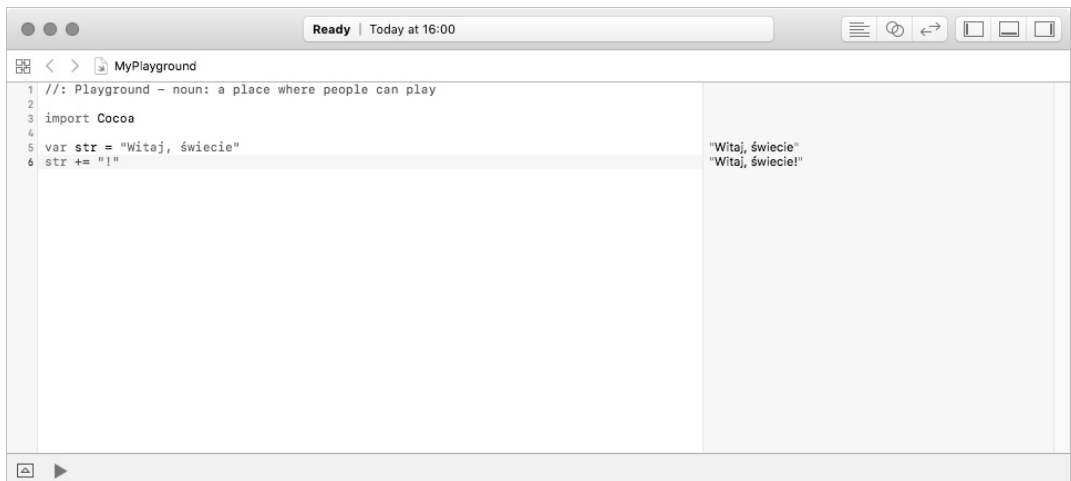
```
import Cocoa

var str = "Witaj, świecie"

str += "!"
```

W celu dodania wykrzyknika zastosowaliśmy **operator dodawania i przypisania** (`+=`). Wymieniony operator łączy w sobie operatory dodawania (`+`) i przypisania (`=`). Więcej informacji na temat operatorów znajdziesz w rozdziale 3.

Czy zauważyłeś coś interesującego w wynikach wyświetlanych po prawej stronie okna? Pojawił się nowy wiersz odzwierciedlający nową wartość zmiennej `str`, którą to wartością obecnie jest pełne powitanie wraz z wykrzyknikiem na końcu (rysunek 1.5).



Rysunek 1.5. Zmiana wartości zmiennej

Rozdział 1. Rozpoczęcie pracy

Przechodzimy teraz do dodania kodu pozwalającego na wyświetlenie w **konsoli** wartości przechowywanej przez zmienną `str`. W narzędziu Xcode konsola wyświetla zdefiniowane w programie komunikaty, które mają być pokazywane po wystąpieniu określonych zdarzeń w programie. Ponadto konsola w Xcode jest wykorzystywana także do wyświetlania komunikatów dotyczących ostrzeżeń i błędów, o ile takie zostaną wygenerowane przez kompilator.

W celu wyświetlenia komunikatu w konsoli należy użyć funkcji `print()`. **Funkcja** grupuje powiązane ze sobą polecenia przekazywane komputerowi, aby wykonać konkretne zadanie. Wymieniona funkcja `print()` jest używana do wyświetlenia w konsoli wartości wraz ze znakiem nowego wiersza. W przeciwieństwie do plików playground projekty Xcode nie zawierają sekcji wyników. Dlatego też z funkcji `print()` będziesz często korzystać podczas tworzenia pełnych aplikacji. Konsola jest również użyteczna do sprawdzania bieżących wartości interesujących Cię zmiennych. Kod po wprowadzonych zmianach możesz zobaczyć na listingu 1.2.

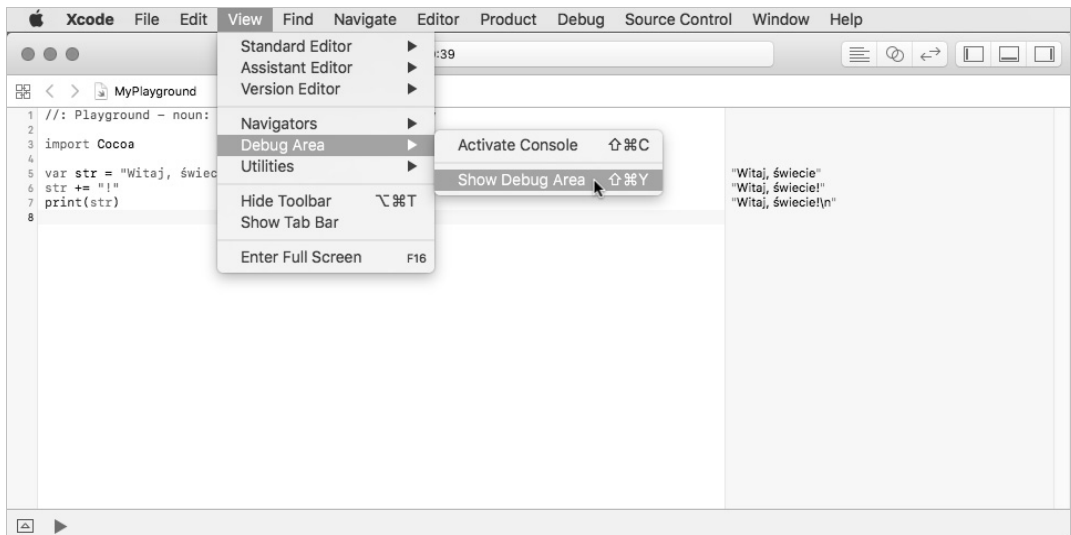
Listing 1.2. Wyświetlenie danych w konsoli

```
import Cocoa

var str = "Witaj, świecie"

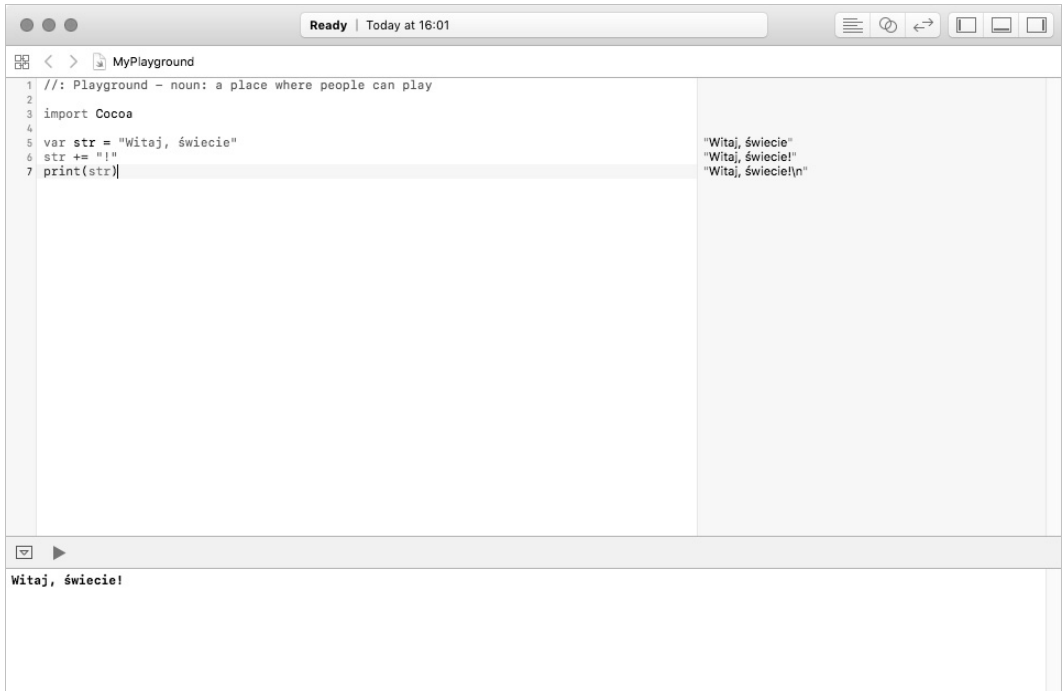
str += "!"
print(str)
```

Obecnie plik playground nie wyświetla sekcji konsoli. Konieczne jest więc aktywowanie sekcji *Debug Area*. W tym celu kliknij menu *View/Debug Area* i wybierz opcję *Show Debug Area* (rysunek 1.6). Zwróć uwagę na istnienie skrótu klawiszowego dla ostatniego z wymienionych kroków. Naciśnięcie klawiszy `Shift+Command+Y` powoduje wyświetlenie lub ukrycie sekcji konsoli.



Rysunek 1.6. Wyświetlenie konsoli (Debug Area) w oknie pliku typu playground

Po wyświetleniu konsoli okno edytowanego pliku playground będzie wyglądało podobnie do pokazanego na rysunku 1.7.



Rysunek 1.7. Twój pierwszy kod w języku Swift

Jesteś na dobrej drodze!

Przypomnijmy raz jeszcze zadania, które wykonałeś w tym rozdziale:

- zainstalowałeś Xcode;
- utworzyłeś plik typu playground i zaznajomiłeś się z nim;
- użyłeś zmiennej i zmodyfikowałeś ją;
- dowiedziałeś się o istnieniu typu **String**;
- użyłeś funkcji w celu wyświetlenia danych w konsoli.

To brzmi dobrze! Już wkrótce będziesz mógł samodzielnie tworzyć własne aplikacje. Jednak zanim to nastąpi, kontynuuj lekturę. Przekonasz się, że niemal wszystko to, co zostało omówione w książce, stanowi pewną odmianę zagadnień poruszonych w tym rozdziale.

Zadanie na brązowy medal

Wiele rozdziałów zawiera na końcu ćwiczenie lub ćwiczenia do wykonania. Te wyzwania są dla Ciebie, mają pomóc Ci w jeszcze lepszym poznaniu języka Swift i zdobyciu dodatkowego doświadczenia. Twoje pierwsze wyzwanie znajduje się poniżej. Zanim do niego przystąpisz, utwórz nowy plik playground.

W tym rozdziale dowiedziałeś się o istnieniu typu **String** oraz możliwości wyświetlenia danych w konsoli dzięki użyciu polecenia `print()`. W utworzonym przed chwilą pliku playground utwórz nowy egzemplarz typu **String**. Wartością tego egzemplarza powinno być Twoje nazwisko. Wyświetl tę wartość w konsoli.

Skorowidz

A

- adnotacja typu, 30
- aktywacja syntezy mowy, 387
- alokacja pamięci, 343
- analiza leksykalna, 288
- aplikacja
 - Cocoa, 371
 - CyclicalAssets, 344
 - iOS, 401
 - iTahDoodle, 402, 404
 - VocalTextEdit, 372, 387
- ARC, automatic reference counting, 344
- architektura MVC, 373
- argument funkcji, 138
- asercja, 291
- Auto Layout, 382
- automatyczne dziedziczenie metody inicjalizacyjnej, 239

B

- bloki
 - case, 59
 - switch, 61
- błędy, 287

C

- ciąg tekstowy, 32, 83
- cykle
 - odwołań w domknięciach, 350
 - silnych odwołań, 344, 348
- część wspólna zbiorów, 133

D

- definiowanie
 - funkcji, 137
 - klasy, 198
- deinicjalizacja, 246
- dekrementacja, 50
- delegacja metody inicjalizacyjnej, 236, 244
- desygnowana metoda inicjalizacyjna dla klasy, 240
- dodanie
 - asercji, 291
 - klasy Objective-C, 447
 - osób, 438
 - widoku tabeli, 407
 - widoku tekstowego, 379
 - właściwości, 194
- dokumentacja, 116
 - protokołu UITableViewDataSource, 414
- dołączanie typu Optional, 95
- domknięcie, 151
 - jako typ odwołania, 161
- domyślna metoda inicjalizacyjna
 - dla klasy, 238
 - dla struktury, 232
- domyślne implementacje rozszerzeń protokołu, 338
- domyślny obraz, 454
- dopasowanie
 - wzorca, 65, 66
 - zakresu, 61, 66
- dostęp
 - do tablicy, 107
 - do słownika, 121

działanie pętli

for, 77

for case, 75

while, 80

dziedziczenie, 199

logiki porównania, 363

protokołu, 282

dzielenie liczb całkowitych, 50

E

egzemplarz

liczby całkowitej, 47

UITableView, 416

ekran powitalny, 21

elementy interfejsu użytkownika, 410

F

filozofia obsługi błędów, 305

format dwójkowy, 46

formatowanie tabeli danych, 271

framework Cocoa, 24

funkcja, 26

filter(_:), 164

map(_:), 163

reduce(_:combine:), 165

funkcje, 137

argument, 138

czyste, 162

generyków, 322

jako argumenty, 157

jako typ wartości zwrotnej, 155

parametry, 138

in-out, 142

wariadyczne, 140

pierwszej klasy, 162

rozwijanie, 205

typy, 149

wartość zwrotna, 144

wartość zwrotna typu Optional, 147

wcześniejsze zakończenie, 148

wyższego rzędu, 163

zagnieżdżone, 144

zasieg, 144

zwrot wielu wartości, 145

G

generyki, 319

funkcje, 322

metody, 322

graficzny interfejs użytkownika, 381

H

hermetyzacja, 271

I

identyczność, 267

indeks, 89

inferencja typu, 29, 75

inicjalizacja, 231, 252

klasy, 238

struktury, 232

inkrementacja, 50

interfejs, 271

użytkownika, 402, 410

interpolacja ciągu tekstowego, 32, 38

iteracja przez zakres wartości, 72

iterator, 72

J

jawne nazwy parametrów, 140

język

Objective-C, 425

Swift, 425

K

klasa, 197

Asset, 345

Cocoa Touch, 412

DefaultImage, 452

Monster, 197

Parser, 298

UIViewController, 439

klasy

błędów, 287

dziedziczenie, 199, 238

główne, 413

inicjalizacja, 238

klauzula where, 64, 329, 336
 klucz, 119
 kodowanie znaków, 85
 kolejność wykonywania operacji, 49
 komentarz, 23
 komórka wielokrotnego użycia, 416
 kompozycja protokołu, 283
 komunikat ostrzeżenia, 433
 konsola, 26
 konstrukcja warunkowa, 37

- else if, 42
- if, 41
- if-else, 37, 41, 68
- switch, 57
 - dołączanie wartości, 62
 - dopasowanie wzorca, 65
 - klauzula where, 64
 - krotka, 65
 - zakres, 61

 kontrola dostępu, 226

- metod getter i setter, 227

 kontroler, 374

- widoku, 374, 440, 442
- widoku tabeli, 432

 konwersja

- słownika na tablicę, 126
- typu, 53

 kopia

- głęboka, 264, 266
- płytką, 264

 kopiowanie, 264
 krotka, 65

L

liczby

- całkowite, 45
- całkowite bez znaku, 47
- zmiennoprzecinkowe, 54

 licznik odwołań, 343, 354
 LIFO, last-in, first-out, 319
 lista

- przechwytywania, 353
- rzeczy do zrobienia, 412, 419

 logika porównania, 363

Ł

łączenie typów Optional, 99

M

mantysa, 54
 mechanizm ARC, 344
 metoda

- advance(), 291
- evaluate(input:), 296
- lex(), 296
- map(_:), 280

 metody, 176

- egzemplarza, 195
- generyków, 322
- getter i setter, 219
- inicjalizacyjne klasy, 231, 243
 - automatyczne dziedziczenie, 239
 - delegacje, 236
 - desygnowane, 240
 - domyślne dla klasy, 238
 - domyślne dla struktury, 232
 - niestandardowe dla struktury, 233
 - składnia, 231
 - w rozszerzeniu, 313
 - wygodne, 243
 - wymagane, 245
 - zakończone niepowodzeniem, 248
- mutowanie, 284
- uniemożliwienie nadpisania, 201
- typu, 204

 model, 373
 modele obiektów, 275
 modelowanie ćwiczenia fizycznego, 333
 moduł Interface Builder, 377
 modyfikacja typu Optional, 100
 mutowanie metod, 196, 284
 MVC, model-view-controller, 373

N

nadawanie nazw, 340
 nadpisanie metody, 201
 narzędzie powłoki, 187
 nawiasy, 49

nazwy parametrów, 139, 203
niejawne rozpakowanie typu Optional, 98
niemodyfikowalne
 słowniki, 126
 tablice, 115
 typy odwołania, 264
niemodyfikowalność, 162
niestandardowa metoda inicjalizacyjna, 233
nowy projekt, 187

O

obiekty pierwszej klasy, 156
Objective-C
 dodanie kodu Swift, 435
 tworzenie aplikacji, 427
obserwatory właściwości, 221
obsługa błędów, 287, 302
 analiza leksykalna, 288
 chowanie głowy w piasku, 302
 filozofia, 305
 przechwytywanie błędów, 296
odczyt treści, 378, 388
odpowiednik kanoniczny, 88
odwołanie, 258
 silne, 345
 słabe, 349
ograniczenia
 przycisku, 382, 383
 systemu Auto Layout, 405–409, 450
 typu, 324, 329
 widoku obrazu, 449
okno opcji projektu, 189
opcje konfiguracyjne projektu, 403
operacje na liczbach całkowitych, 49
operator
 dodawania i przypisania, 25
 koalescencji, 100
 trójargumentowy, 40
operatory
 infiks, 359
 logiczne, 40
 porównania, 39
 przepelnienia, 51
 własne, 365
outlet widoku tekstowego, 385

P

pamięć, 343
para klucz-wartość, 119
parametry
 funkcji, 138
 in-out, 142
 metody, 203
 wariacyjne, 140
 wartość domyślna, 141
 pasek menu, 191
pętla
 for, 76
 for case, 74
 for-in, 71
 repeat-while, 78
 while, 77
pętle nieskończone, 80
pierwsza aplikacja
 Cocoa, 371
 iOS, 401
plik
 main.swift, 189
 Town.swift, 193
pliki typu playground, 22, 23
podklasa Zombie, 199
pole tekstowe, 404
polecenia
 kontrolni przepływu, 35
 transferu kontroli, 78
polecenie
 break, 80
 continue, 78
 fallthrough, 60
 transferu kontroli, 60
polimorfizm parametryczny, 332
połączenie, 384
 outletu widoku tekstowego, 385
porównywalność, 357, 363
porównywanie tablic, 113
powiązane
 typy protokołów, 325
 wartości, 179
programowanie funkcyjne, 162
projekt w Objective-C, 425

protokoły
 dziedziczenie, 282
 kompozycja, 283
 rozszerzenia, 333
 protokół, 271, 276
 Comparable, 360
 CustomStringConvertible, 344
 Equatable, 357
 ExerciseType, 335
 UITableViewDataSource, 414
 przechwytywanie błędów, 296
 przeciążenie, 359
 przepełnienie, 51
 przetwarzanie tablicy tokenów, 297
 przycisk
 do odczytu, 378
 Odczytaj, 382
 wyświetlający wynik, 72
 przytrzymanie obiektu, 343
 pula komórek wielokrotnego użycia, 416

R

refaktoryzacja, 112
 rekurencyjny typ wyliczeniowy, 182
 rozbudowa istniejącego typu, 309
 rozpakowanie typów Optional, 97
 rozszerzanie własnego typu, 311
 rozszerzenia, 309
 protokołu, 333
 ExerciseType, 335
 wraz z funkcjami, 315
 wraz z klauzulą where, 336
 rozszerzony klaster grafu, 87
 rozwijanie funkcji, 205
 równość, 267, 357
 rzutowanie typu, 391

S

sekcje, 190
 semantyka
 odwołania, 258
 wartości, 255
 silne odwołania, 348

skalary
 połączone, 86
 Unicode, 85
 składnia
 domknięcia, 151
 metody inicjalizacyjnej, 231
 wyrażenia domknięcia, 153
 z kropką, 84
 skróty operatorów, 50
 słabe odwołania, 349, 350
 słowniki, 119
 dodawanie wartości, 123
 konwersja na tablicę, 126
 modyfikacja, 121
 niemodyfikowalne, 126
 tworzenie, 119
 usuwanie wartości, 123
 uzyskanie dostępu, 121
 zabezpieczenie, 120
 słowo kluczowe
 extension, 310
 override, 199
 sprawdzenie typu, 29
 stała, 31
 stos, 319
 struktura, 192
 Department, 278
 Point, 359
 Stack, 319
 struktury
 danych generyków, 319
 domyślna metoda inicjalizacyjna, 232
 niestandardowa metoda inicjalizacyjna, 233
 syntezytor mowy, 387
 system Auto Layout, 382, 405

Ś

ściśle typowanie, 162

T

tablice, 105
 modyfikacja, 107
 niemodyfikowalne, 115
 porównywanie, 113

tablice

- tokenów, 297
- tworzenie, 105
- uzyskanie dostępu, 107

token, 297

tokenizacja, 288

tworzenie

- aplikacji, 427
- egzemplarza liczby całkowitej, 47
- egzemplarza struktury, 194
- filtru, 64
- kontrolera widoku, 374
- krotki, 65
- narzędzia powłoki, 187
- pliku Town.swift, 193
- płytkiej kopii t, 266
- połączeń, 384
- rozszerzenia, 314
- słownika, 119
- tablicy, 105
- widoków, 377
- zbioru, 130

typ, 29

- Array, 105
- Boolean, 79
- Character, 85
- Dictionary, 119
- Double, 54
- FileManager, 420
- GreekGod, 265
- Int, 32, 45
- Int32, 46
- NSArray, 420
- odwołania, 260
- Optional, 93, 175, 331
 - dołączanie, 95
 - łączenie, 99
 - modyfikacja, 100
 - niejawne rozpakowanie, 98
 - operator koalescencji, 100
- Point, 358
- Set, 129
- Stack, 320
- String, 25, 85
- TextAlignment, 175

typy

- funkcji, 149
- odwołania, 255
 - niemodyfikowalne, 264
- powiązane, 325
- wartości, 255
- wyliczeniowe, 169
 - podstawowe, 169
 - rekurencyjne, 182
 - wartości pierwotne, 173
- zagnieżdżone, 214

U

Unicode, 85

unie, 131

uporządkowanie kodu, 397

użycie

- gotowego znaku, 88
- jawnych nazw parametrów, 140
- konstrukcji if-case, 68
- operatora trójargumentowego, 40
- pętli wraz ze słownikiem, 125
- polecenia break, 80
- polecenia continue, 78
- typów odwołania, 263
- typów wartości, 263

W

wartości

- pierwotne, 173
- przechwytywane przez domknięcie, 159

wartość, 255

- domyślna parametru, 141
- stałej, 260
- zwrotna funkcji, 144
- zwrotna typu Optional, 147

wczytywanie

- dokumentów, 389, 394
- listy, 419, 421

widok, 374

- obrazu, 448
- tekstowy, 379

własne operatory, 365

właściwości, 213
obliczane, 218
składowane
opóźnione, 215
podstawowe, 213
typu, 222
wtyczki, 455
wybór
punktu na wykresie, 73
szablону projektu, 188
widoku tekstowego, 380
wyciek pamięci, 348
wykładnik, 54
wymuszone rozpakowanie, 95
wyśrodkowanie widoku obrazu, 448
wyświetlanie
danych, 25
konsoli, 27
pojedynczych znaków, 85

Z

zagnieżdżone
dołączanie typu Optional, 96
konstrukcje if, 41
rozszerzenia, 314
typy, 314

zakończone niepowodzeniem w klasach, 251
zakres, 89
zapisywanie
dokumentów, 389, 392
plików, 23
listy, 419
zarządzanie pamięcią, 343
zasięg funkcji, 144
zastępowanie elementu tablicy, 110
zbiory, 129
część wspólna, 133
powtarzające się elementy, 134
tworzenie, 130
zdarzenia, 369
zgodność z protokołem, 280, 312
Comparable, 360
Equatable, 357
zliczanie elementów, 89
zmiana
wielkości widoku tabeli, 408
właściwości przycisku, 407
zmienna, 25, 31

Ż

źródła danych, 271

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

Swift: wyjątkowe narzędzie do budowy wyjątkowych aplikacji dla Apple!

Swift, następca języka Objective-C, pojawił się na rynku w 2014 roku i bardzo szybko zdobył popularność wśród programistów tworzących aplikacje dla iOS oraz macOS. Charakteryzuje się zwięzłą i przejrzystą składnią, a co więcej, pozwala na korzystanie z nowoczesnych, wysokopoziomowych struktur i cech języka, takich jak typy generyczne czy domknięcia. Jest przy tym wygodny i elastyczny, a jego nauka nie powinna sprawiać problemów nawet tym, którzy dopiero rozpoczynają przygodę z programowaniem.

Ta książka jest starannie przygotowanym, praktycznym podręcznikiem efektywnego programowania w języku Swift 3. Wyczerpująco przedstawiono tu zasady postępowania się środowiskiem programistycznym Xcode 8 i dokumentacją Apple. Dzięki lekturze zyskasz wiedzę i umiejętności pozwalające na samodzielne rozwiązywanie problemów programistycznych z użyciem języka Swift. Innymi słowy, zaczniesz programować kompletne, efektywne i dojrzałe aplikacje dla platform iOS i macOS.

W tej książce:

- przedstawiono podstawy składni języka Swift
- omówiono konstrukcje służące do kontroli przepływu działania programu
- pokazano, jak korzystać z kolekcji, typów wyliczeniowych, struktur i klas
- zaprezentowano zasady budowania eleganckiego, czytelnego i efektywnego kodu
- przedstawiono metody projektowania aplikacji opartej na zdarzeniach



Matt Mathias

jest doktorem socjologii i dyrektorem działu szkoleń w firmie Big Nerd Ranch. Uczy programowania na platformie iOS. Uwielbia jazdę na rowerze, komiksy oraz wszelkiego rodzaju gry.



John Gallagher

jest inżynierem oprogramowania i instruktorem w Big Nerd Ranch. Znamca systemów wbudowanych i superkomputerów, pasjonuje się poznawaniem nowych języków programowania. Kiedy nie zajmuje się pracą, spędza czas wolny z żoną i z trzema córkami.

Helion

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

informatyka w najlepszym wydaniu

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

Sprawdź najnowsze promocje:
● <http://helion.pl/promocje>
Książki najchętniej czytane:
● <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
● <http://helion.pl/nawosci>



**Big Nerd
Ranch**

ISBN 978-83-283-3142-6



9 788328 331426

cena: 79,00 zł

sięgnij po **WIĘCEJ**



KOD KORZYŚCI