Programowanie w języku Swift

Big Nerd Ranch Guide

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.

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.

Spis treści

Część III Kolekcje i funkcje

1

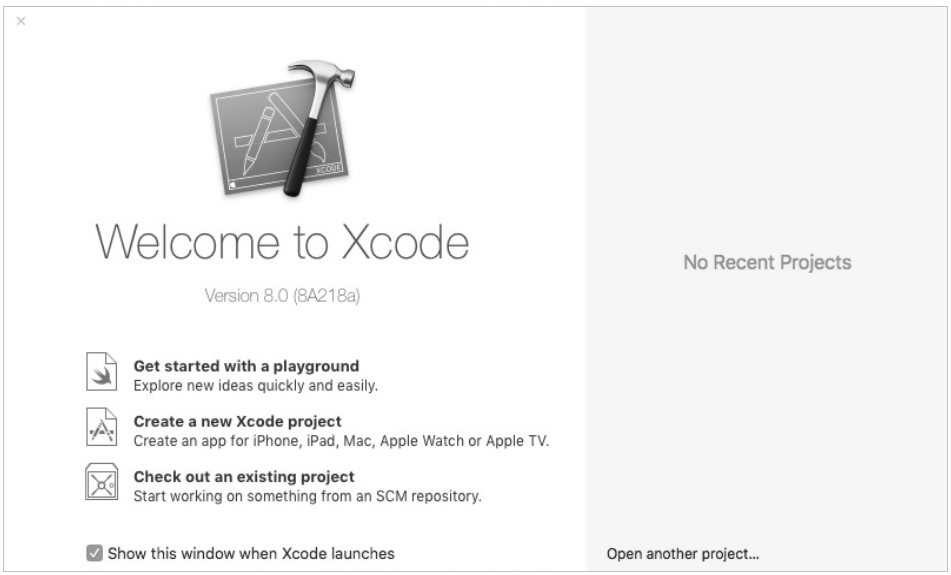
Rozpoczęcie pracy

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.



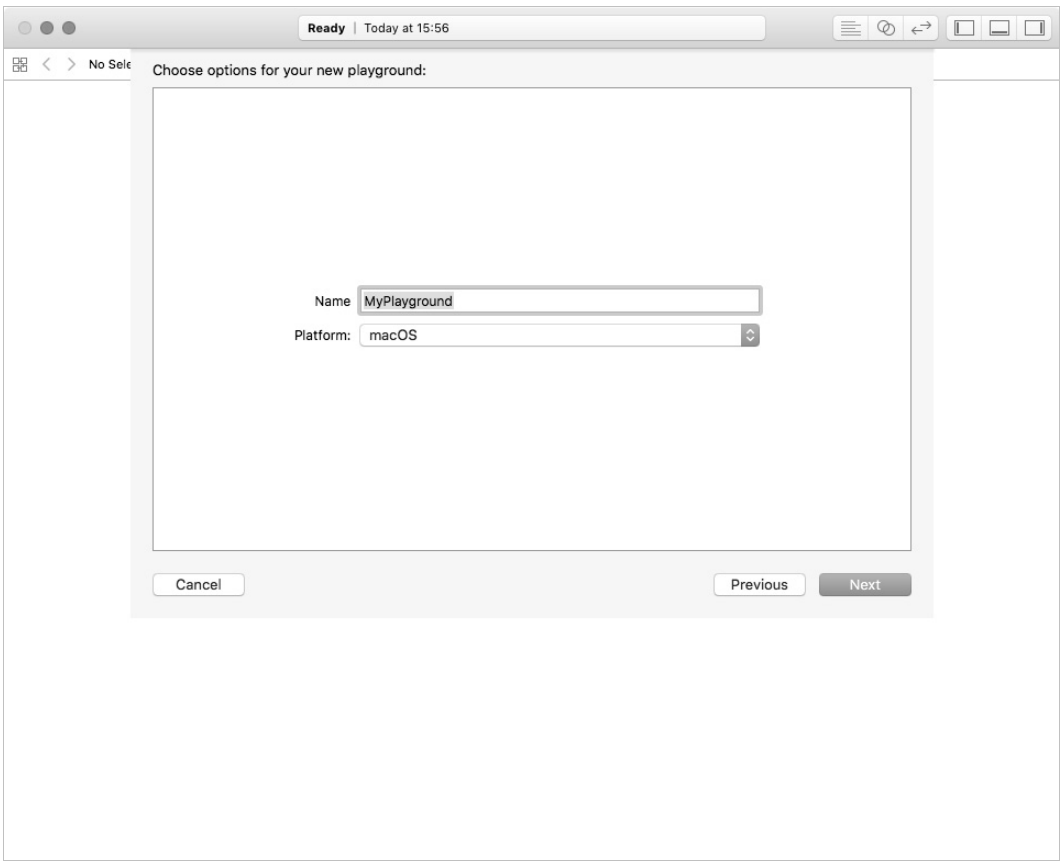
Rysunek 1.1. Ekran powitalny wyświetlany po uruchomieniu Xcode

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.

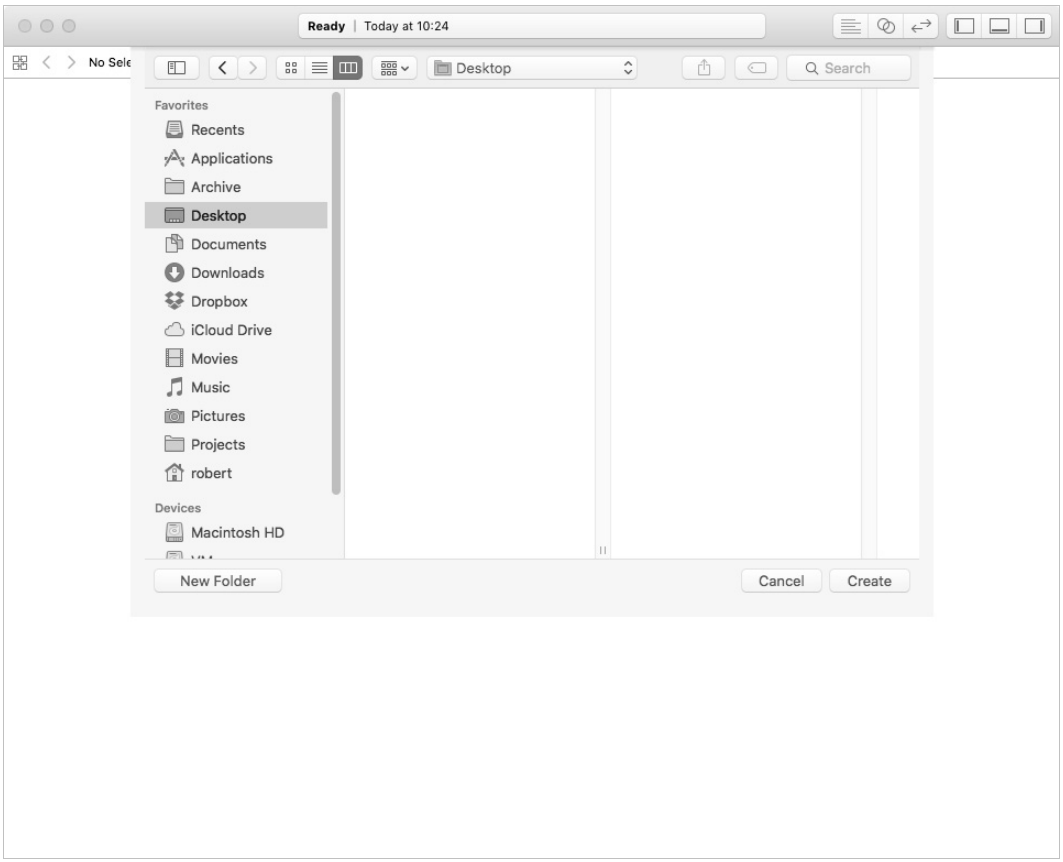
Na ekranie powitalnym kliknij *Get started with a playground*.

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).

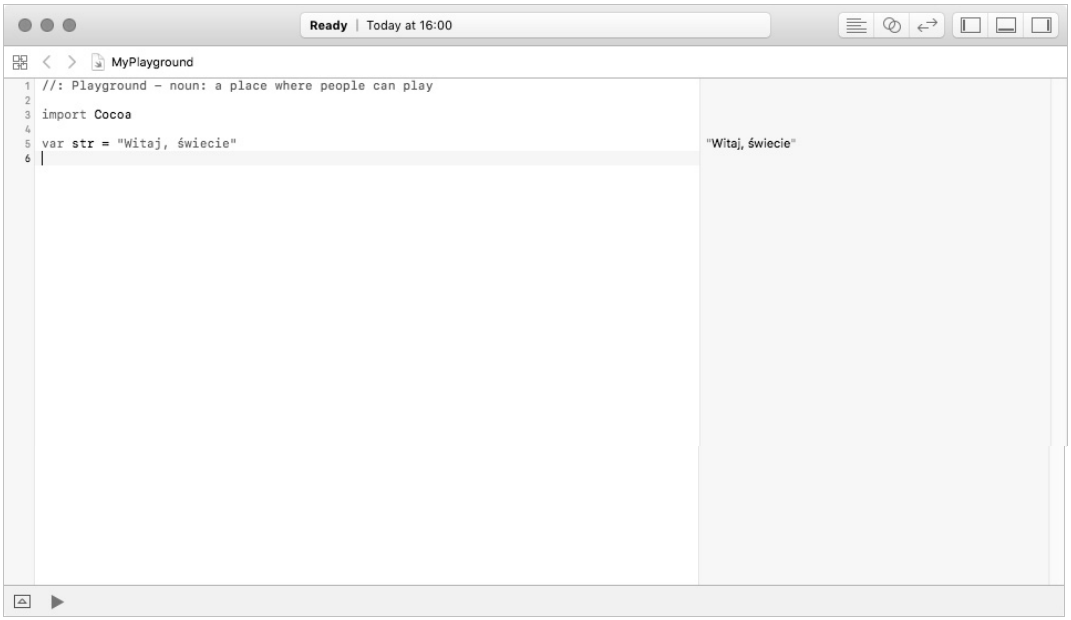


Rysunek 1.3. Zapis pliku typu playground

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 jakiejkolwiek 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+/*.

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 dokładniej 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 Swifcie 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.

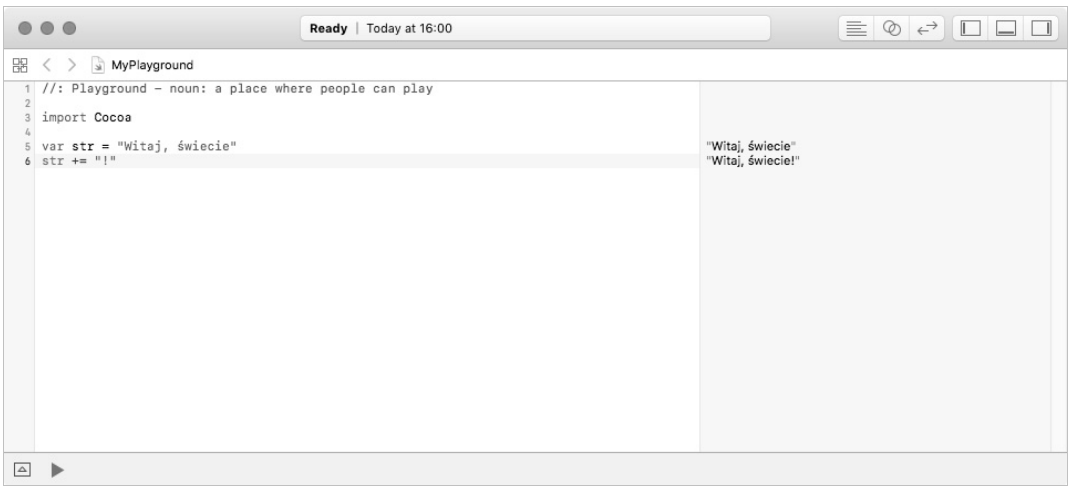
Przypomnij sobie, że `str` to **zmienna**. Mamy więc możliwość zmiany wartości przypisanej zmiennej. Teraz dołączymy wykrzyknik na końcu przykładowego ciągu tekstowego, aby tym samym otrzymać poprawne zdanie. (Kiedy do przykładu dodajemy nowy kod, będzie on przedstawiony pogrubioną czcionką. Natomiast 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

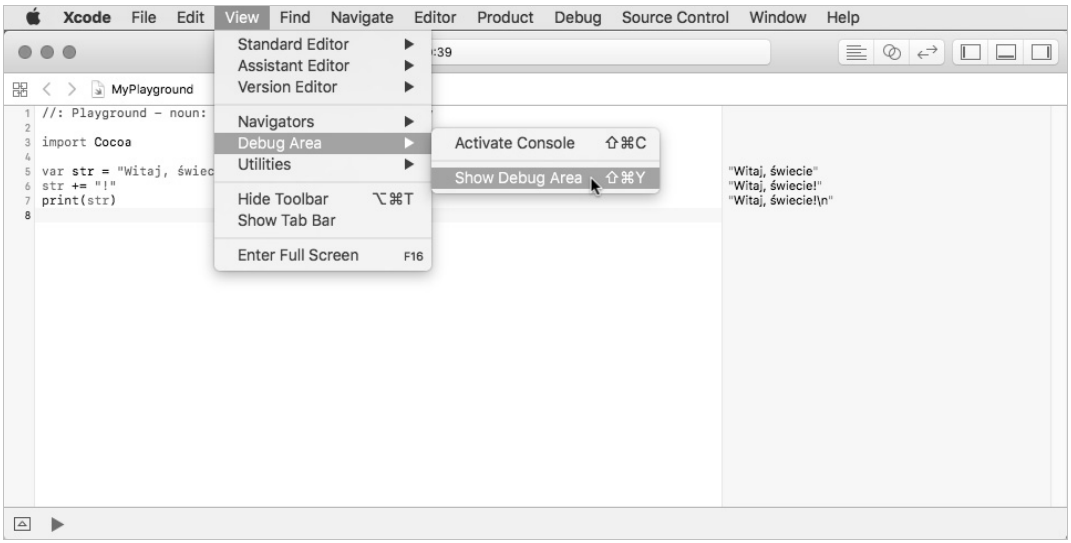
Przechodzimy teraz do dodania kodu pozwalającego na wyświetlenie w **konsoli** wartości przechowywanej przez zmienną `str`. W narzędziu Xcode konsola wyświetla zdefiniowane w programie komunikaty, które mają być pokazywane po wystąpieniu określonych zdarzeń w programie. Ponadto konsola w Xcode jest wykorzystywana 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

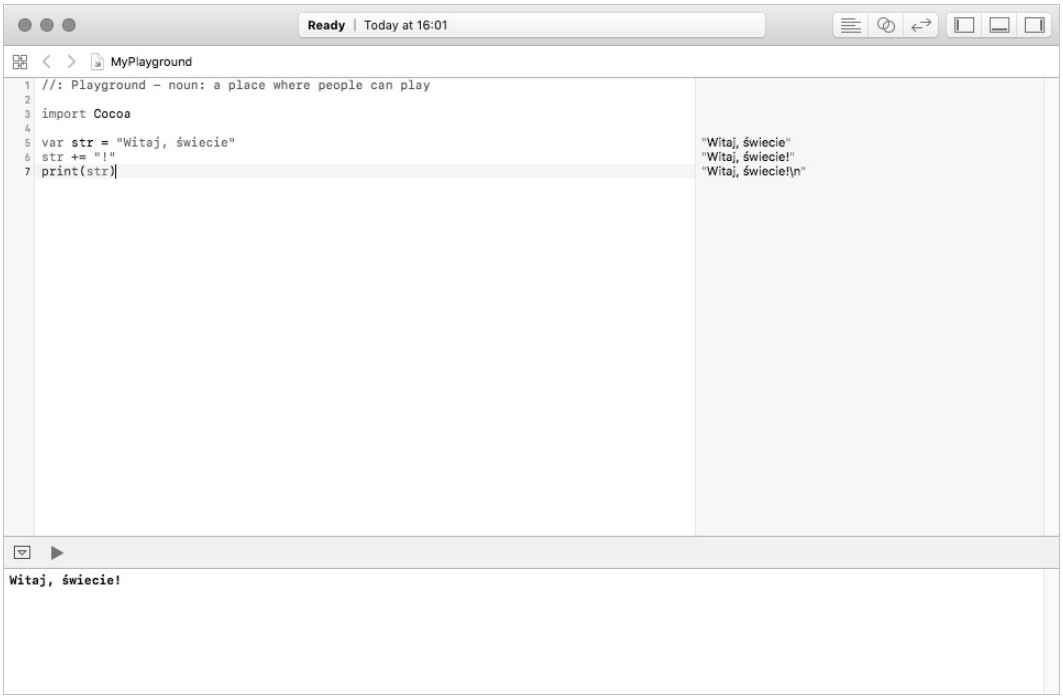
```
import Cocoa

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

B

C

D

L

Ł

M

N

O

P

R

S

Ś

T

U

W

Z

Ź

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 posługiwania 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. Znawca 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/nowosci

Big Nerd Ranch

ISBN 978-83-283-3142-6

cena: 79,00 zł