

Oficjalny
przewodnik
ROBLOX

Roblox Lua

Tworzenie gier
dla początkujących

W **24**
godziny

Tytuł oryginału: Coding with Roblox Lua in 24 Hours:
The Official Roblox Guide (Sams Teach Yourself)

Tłumaczenie: Karolina Liszewska

ISBN: 978-83-283-9449-0

Authorized translation from the English language edition, entitled Coding with Roblox Lua in 24 Hours: The Official Roblox Guide, 1st Edition by Roblox Corporation, published by Pearson Education, Inc, publishing as Sams Publishing, Copyright © 2022 by Roblox Corporation.

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 © 2023.

“Roblox,” the Roblox logo, and “Powering Imagination” are among the Roblox registered and unregistered trademarks in the U.S. and other countries. All rights reserved.

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 wydawca 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 wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<https://helion.pl/user/opinie/roblua>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 230 98 63

e-mail: helion@helion.pl

WWW: <https://helion.pl> (księgarnia internetowa, katalog książek)

Printed in Poland.

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

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

Spis treści

	O autorce	11
	Chcemy poznać Twoje zdanie!	12
	Dla czytelników	12
Godzina 1.	Tworzenie pierwszego projektu	13
	Instalacja Roblox Studio	13
	Chodźmy na wycieczkę	14
	Otwieranie okna Output	16
	Pisanie pierwszego skryptu	17
	Komunikaty o błędach	22
	Zostawianie komentarzy	23
	Podsumowanie	23
	Pytania i odpowiedzi	23
	Warsztaty	24
	Ćwiczenie	24
Godzina 2.	Zmienne i właściwości	26
	Hierarchia obiektów	27
	Słowa kluczowe	28
	Właściwości	29
	Znajdowanie właściwości i typów danych	30
	Tworzenie zmiennych	31
	Zmiana koloru	33
	Instancje	34
	Podsumowanie	35
	Pytania i odpowiedzi	36
	Warsztaty	36
	Ćwiczenia	37

Godzina 3.	Tworzenie funkcji i korzystanie z nich	39
	Tworzenie i wywoływanie funkcji	39
	Rozumienie zakresu	41
	Wykorzystywanie zdarzeń do wywoływania funkcji	41
	Kolejność i rozmieszczenie	44
	Podsumowanie	47
	Pytania i odpowiedzi	48
	Warsztaty	48
	Ćwiczenie	49
Godzina 4.	Parametry i argumenty	50
	Przekazywanie informacji do funkcji	50
	Praca z wieloma parametrami i argumentami	52
	Zwracanie wartości z funkcji	55
	Zwracanie wielu wartości	56
	Zwracanie nil	57
	Radzenie sobie z niezgodnymi argumentami i parametrami	57
	Funkcje anonimowe	58
	Podsumowanie	59
	Pytania i odpowiedzi	59
	Warsztaty	60
Godzina 5.	Wyrażenia warunkowe	61
	Wyrażenia if/then	62
	elseif	65
	Operatory logiczne	65
	else	66
	Podsumowanie	73
	Warsztaty	73
	Ćwiczenie	74
Godzina 6.	Debouncing i debugowanie	76
	Nie niszc — zastosuj debouncing	76
	Dowiadывanie się, w którym miejscu coś poszło nie tak	84
	Podsumowanie	88
	Pytania i odpowiedzi	88
	Warsztaty	89
	Ćwiczenia	90

Godzina 7.	Pętle while	92
	Powtarzaj w nieskończoność, while true do	92
	Kilka kwestii, o których warto pamiętać	93
	Pętle while a zakres	98
	Podsumowanie	98
	Pytania i odpowiedzi	99
	Warsztaty	99
	Ćwiczenia	100
Godzina 8.	Pętle for	102
	Jak działają pętle for	103
	Pętle zagnieżdżone	109
	Wychodzenie z pętli	110
	Podsumowanie	110
	Pytania i odpowiedzi	110
	Warsztaty	111
	Ćwiczenia	111
Godzina 9.	Praca z tablicami	113
	Czym są tablice?	113
	Późniejsze dodawanie elementów	114
	Pobieranie informacji spod określonego indeksu	114
	Drukowanie całej listy za pomocą ipairs()	115
	Foldery i ipairs()	115
	Wyszukiwanie wartości na liście i drukowanie indeksu	120
	Usuwanie wartości z tablicy	120
	Numeryczne pętle for i tablice	121
	Podsumowanie	122
	Pytania i odpowiedzi	123
	Warsztaty	123
	Ćwiczenie	123
Godzina 10.	Praca ze słownikami	125
	Wprowadzenie do słowników	125
	Dodawanie wpisów do słowników i usuwanie ich ze słowników	128
	Usuwanie par klucz-wartość	128
	Praca ze słownikami i funkcją pairs()	130
	Zwracanie wartości z tabeli	130
	Podsumowanie	138

Pytania i odpowiedzi	138
Warsztaty	139
Ćwiczenie	139
Godzina 11. Klient kontra serwer	140
Klient i serwer	140
Praca z GUI	141
Rozumienie RemoteFunction	144
Stosowanie RemoteFunction	144
Podsumowanie	152
Pytania i odpowiedzi	152
Warsztaty	152
Ćwiczenia	153
Godzina 12. Zdarzenia zdalne: komunikacja jednostronna	154
Zdarzenia zdalne: ulica jednokierunkowa	154
Komunikacja od serwera do wszystkich klientów	155
Komunikacja od klienta do serwera	158
Komunikacja od serwera do pojedynczego klienta	162
Komunikacja od klienta do klienta	162
Podsumowanie	163
Warsztaty	163
Ćwiczenie	164
Godzina 13. Korzystanie z ModuleScript	165
Programowanie rzeczy tylko raz	165
Umieszczenie ModuleScript	166
Jak działa ModuleScript	166
Nazywanie ModuleScript	166
Dodawanie funkcji i zmiennych	167
Rozumienie zakresu w ModuleScript	168
Korzystanie z modułów w innych skryptach	168
Nie powtarzaj się	174
Operowanie abstrakcjami	174
Podsumowanie	175
Pytania i odpowiedzi	175
Warsztaty	175
Ćwiczenie	176

Godzina 14. Programowanie w przestrzeni 3D	177
Rozumienie współrzędnych X, Y i Z	177
Udoskonalenie rozmieszczenia za pomocą współrzędnych CFrame	178
Offsetowanie ramek CFrame	180
Dodawanie obrotów do ramek CF	180
Praca z modelami	180
Zrozumienie współrzędnych ze świata i lokalnych współrzędnych obiektu	181
Podsumowanie	184
Warsztaty	184
Ćwiczenie	184
Godzina 15. Płynna animacja obiektów	186
Czym są tweeny	186
Ustawianie parametrów TweenInfo	188
Łączenie tweenów	192
Podsumowanie	193
Warsztaty	194
Ćwiczenie	194
Godzina 16. Rozwiązywanie problemów za pomocą algorytmów	196
Definiowanie algorytmów	196
Sortowanie tablicy	197
Sortowanie w kolejności malejącej	199
Sortowanie słownika	199
Sortowanie według wielu rodzajów informacji	202
Podsumowanie	203
Warsztaty	203
Ćwiczenie	204
Godzina 17. Zapisywanie danych	205
Włączanie magazynów danych	205
Tworzenie magazynu danych	205
Korzystanie z danych w magazynie	206
Ograniczanie liczby zapytań	210
Ochrona Twoich danych	210
Zapisywanie danych gracza	211
Stosowanie UpdateAsync do aktualizacji magazynu danych	212
Podsumowanie	212
Pytania i odpowiedzi	213
Warsztaty	213
Ćwiczenie	214

Godzina 18. Tworzenie pętli gry	215
Tworzenie pętli gry	215
Praca z BindableEvent	216
Podsumowanie	225
Pytania i odpowiedzi	226
Warsztaty	226
Ćwiczenie	227
 Godzina 19. Monetyzacja: jednorazowe transakcje	 228
Dodawanie passów do gry	228
Konfiguracja passa	230
Zachęcanie graczy do zakupów w grze	231
Podsumowanie	238
Pytania i odpowiedzi	239
Warsztaty	239
Ćwiczenie	240
 Godzina 20. Programowanie obiektowe	 241
Czym jest programowanie obiektowe?	241
Organizacja kodu i projektów	241
Tworzenie nowej klasy	242
Dodawanie właściwości do klasy	243
Korzystanie z funkcji w klasie	244
Podsumowanie	249
Warsztaty	249
Ćwiczenie	251
 Godzina 21. Dziedziczenie	 252
Konfiguracja dziedziczenia	253
Dziedziczenie właściwości	255
Praca z wieloma klasami potomnymi	257
Dziedziczenie funkcji	258
Rozumienie polimorfizmu	258
Wywoływanie funkcji nadrzędnych	261
Podsumowanie	263
Warsztaty	263
Ćwiczenie	264

Godzina 22. Raycasting	265
Konfiguracja funkcji służącej do raycastingu	265
Sztuczka matematyczna w trójwymiarowej przestrzeni: uzyskiwanie kierunku	267
Ustawianie parametrów funkcji Raycast	268
Sztuczka matematyczna w przestrzeni 3D: ograniczanie kierunku	270
Podsumowanie	271
Pytania i odpowiedzi	271
Warsztaty	271
Ćwiczenie	272
 Godzina 23. Plopping obiektów w projekcie — część 1.	273
Konfiguracja obiektu	274
Tworzenie przycisku do ploppingu	277
Śledzenie ruchów myszy	278
Podgląd obiektu poddanego ploppingowi	282
Podsumowanie	284
Pytania i odpowiedzi	285
Warsztaty	285
Ćwiczenie	285
 Godzina 24. Plopping obiektów w projekcie — część 2.	287
Wykrywanie danych wejściowych myszy	288
Wysyłanie wiadomości na serwer	290
Otrzymywanie wiadomości	290
Podsumowanie	292
Pytania i odpowiedzi	293
Warsztaty	293
Ćwiczenie	294
 Dodatek A Podstawy Robloksa	295

Godzina 4.

Parametry i argumenty

Czego nauczysz się w tej godzinie:

- ▶ Jak używać parametrów.
- ▶ Jak stosować wiele parametrów i argumentów.
- ▶ Jak zwracać wartości z funkcji.
- ▶ Jak korzystać z funkcji anonimowych.

Funkcje nie tylko wykonują zadania; mogą działać jak małe maszyny w fabryce, które przyjmują rzeczy, przekształcają je, a następnie zwracają wyniki. W tej godzinie skupimy się na informacjach, które znajdują się w nawiasach — na parametrach i argumentach — oraz na tym, co funkcja może z nimi zrobić.

Przekazywanie informacji do funkcji

Funkcje nie muszą być samodzielnymi fragmentami kodu; mogą pobierać informacje do wykorzystania z zewnątrz. Widzieliśmy to przy funkcji `print("Hello")`, która pobiera komunikaty do wyświetlenia, i `wait(3)`, która przyjmuje liczbę sekund, na jaką wstrzymać skrypt.

Wartości przekazywane w nawiasach do funkcji nazywane są argumentami. Tworząc własne funkcje, w których będą przekazywane informacje, musisz utworzyć symbole zastępcze dla argumentów. Te symbole zastępcze nazywane są **parametrami**.

Aby utworzyć własne parametry, dodaj nazwę zmiennej w nawiasie podczas definiowania funkcji, na przykład:

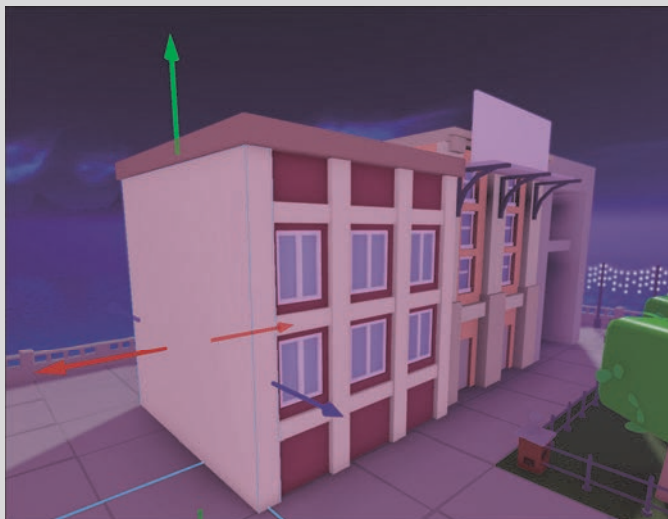
```
local function functionName(parameterName)  
  
end
```

Z parametru można skorzystać tak jak z każdej innej zmiennej wewnątrz funkcji.

▼ SPRÓBUJ SAM

Utwórz funkcję malującą

Bok budynku na rysunku 4.1 zostanie zmieniony poprzez utworzenie nowej funkcji o nazwie `paint()`, która ma parametr określający kolor ściany, na jaki powinna zostać przemalowana. Zamiast korzystać z budynku, możesz poćwiczyć na zwykłej, starej części lub modelu bez tekstury.



RYSUNEK 4.1. Skorzystaj z funkcji, aby przemalować część budynku

1. Wewnątrz części lub modelu dodaj nowy skrypt o nazwie `Paint`.
2. Utwórz nową zmienną lokalną przypisaną do części nadrzędnej (rodzica), a następnie nową funkcję lokalną o nazwie `paint()`:

```
local wall = script.Parent
local function paint()
```

end
3. Dodaj parametr o nazwie `paintColor`, który będzie kontenerem na kolor, na jaki trzeba będzie przemalować ścianę:

```
local wall = script.Parent
local function paint(paintColor)
```

end
4. Wewnątrz funkcji ustaw kolor ściany na kolor przechowywany w `paintColor`:

```
local wall = script.Parent
local function paint(paintColor)
    wall.Color = paintColor
end
```

5. Dodaj jedną lub dwie zmienne przechowujące różne kolory RGB, na które możesz zechcieć pomalować ścianę:

```
local wall = script.Parent

local blue = Color3.fromRGB(29, 121, 160)
local yellow = Color3.fromRGB(219, 223, 128)

local function paint(paintColor)
    wall.Color = paintColor
end
```

Wskazówka

Wskazówka

Umiejscowienie zmiennych

Może już zauważyliście, że zmienne zazwyczaj są umieszczane na górze skryptu lub fragmentu kodu, do którego należą.

6. Wywołaj funkcję malującą i przekaz jej jeden z kolorów jako argument:

```
local wall = script.Parent

local blue = Color3.fromRGB(29, 121, 160)
local yellow = Color3.fromRGB(219, 223, 128)

local function paint(paintColor)
    wall.Color = paintColor
end

paint(blue)
```

Przetestuj kod. Część zostanie przemalowana na wybrany kolor!

Praca z wieloma parametrami i argumentami

W poprzednim bloku „Spróbuj sam(a)” do funkcji przekazano wybrane kolory, ale obiekt, który miał zostać pomalowany, został z góry zakodowany. Innymi słowy, kod ten działa tylko na tym obiekcie.

Takie podejście ogranicza użyteczność funkcji, chyba że planujesz często zmieniać kolor tej jednej ściany. Na szczęście do funkcji można przekazywać więcej niż jeden argument. Należy tutaj utworzyć więcej niż jeden parametr.

Nazwy parametrów przy definiowaniu funkcji oddziela się przecinkiem, jak pokazano tutaj:

```
local function functionName(firstParameter, secondParameter)
    print(firstParameter .. " and " .. secondParameter)
end
```

Wskazówka

Ile parametrów to zbyt wiele?

Nie ma technicznych ograniczeń co do liczby parametrów, ale nie więcej niż trzy to dobra zasada.

Przekazywane argumenty zawsze „wypełniają” parametry w podanej kolejności. Pierwszy argument zawsze jest reprezentowany przez pierwszy parametr, a drugi argument przez drugi parametr:

```
local first = "first"
local second = "second"

local function practice(firstParameter,secondParameter)
    print(firstParameter .. " and " .. secondParameter)
end

practice(first, second) -- Drukuje "first and second"
practice(second, first) -- Drukuje "second and first"
```

▼ SPRÓBUJ SAM

Przełącz kolor i obiekt

Spraw, aby funkcja malowania była bardziej użyteczna, i utwórz zmienną, która pobiera zarówno obiekt do przemalowania, jak i wybrany kolor. Rysunek 4.2 przedstawia samochód i budynek pomalowane obecnie na biało, co jest niesamowicie nudne i nie pasuje do sceny. Skorzystaj z tego samego kodu, co poprzednio, ale zmień go tak, aby można było przekazać obiekt, który chcesz pomalować. W ten sposób kod może zostać użyty zarówno do pomalowania budynku, jak i samochodu:



RYSUNEK 4.2. Funkcji z dwoma parametrami można użyć do wyznaczenia obiektu do pomalowania oraz koloru

1. Stwórz nowy skrypt w ServerScriptService.
2. Przypisz zmienne do dwóch różnych kolorów i dwóch różnych obiektów:

```
-- Dostępne kolory
local red = Color3.fromRGB(170, 0, 0)
local olive = Color3.fromRGB(151, 15, 156)

-- Obiekty do pomalowania
local car = workspace.Car
local restaurant = workspace.Buildings.Restaurant
```

Wskazówka

Wskazówka

Znajdowanie zagnieżdżonych obiektów

Zwróć uwagę, że drugi obiekt w tym przykładzie, restauracja, znajdował się w folderze, więc do przejścia o jeden poziom w dół hierarchii użyto operatora kropki.

3. Utwórz funkcję, która posiada parametr odpowiadający obiektowi do pomalowania i parametr odpowiadający kolorowi:

```
-- Maluje obiekty
local function painter(objectToPaint, paintColor)
    objectToPaint.Color = paintColor
end
```

4. Wywołaj funkcję i przekaz obiekt do pomalowania oraz kolor:

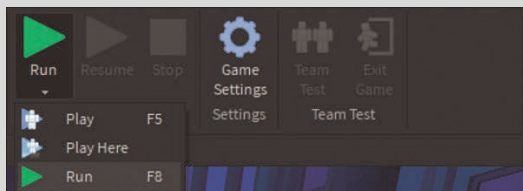
```
-- Dostępne kolory
local red = Color3.fromRGB(170, 0, 0)
local olive = Color3.fromRGB(151, 15, 156)

-- Obiekty do pomalowania
local car = workspace.Car
local restaurant = workspace.Buildings.Restaurant

-- Maluje obiekty
local function painter(objectToPaint, paintColor)
    objectToPaint.Color = paintColor
end

painter(restaurant, olive)
painter(car, red)
```

5. Przetestuj swój kod. Jeśli chcesz zobaczyć zmiany wprowadzone w świecie bez grania, w menu rozwijanym wybierz Run zamiast Play (rysunek 4.3).



RYСУNEK 4.3. Wybierz Run, aby testować kod bez ładowania awatara postaci

Na rysunku 4.4 pokazano pokolorowane samochód i budynek, które nie są już białe.



RYSUNEK 4.4. Po uruchomieniu skryptu kolory budynku i samochodu uległy zmianie

Zwracanie wartości z funkcji

Wartości mogą być nie tylko przekazywane do funkcji, ale także przekazywane z powrotem. Klasycznym przykładem jest kalkulator, taki jak ten w telefonie. Wartości są przekazywane do programu, a wynik jest **zwracany**. W poniższym przykładzie funkcja jest przypisana do zmiennej. Gdy zmienna jest używana, funkcja wykonuje się, a wynik jest odsyłany za pomocą słowa kluczowego `return`:

```
-- Dodaje dwie dowolne liczby
local function add(firstNumber, secondNumber)
    local sum = firstNumber + secondNumber
    return sum -- Wysła sumę tam, gdzie funkcja została wywołana
end

-- Liczby do wykorzystania
local rent = 3500
local electricity = 128

-- Użyj add(), aby dodać rent i electricity oraz zwrócić wynik
local costOfLiving = add(rent, electricity)
print("Rent in New York is " .. costOfLiving)
```

Zwracanie wielu wartości

Czasami można chcieć zwrócić wiele wartości z funkcji. Przykładem może być zwrócenie liczby wygranych, przegranych i remisów użytkownika. Aby zwrócić wiele wartości, użyj jak zwykle `return` i oddziel wartości przecinkiem.

▼ SPRÓBUJ SAM

Zwróć liczbę wygranych, przegranych i remisów

Postępuj zgodnie z instrukcjami, aby utworzyć niestandardową funkcję, która po wywołaniu zwraca wygrane, przegrane i remisy gracza. Przypisz zwrócone wartości do zmiennej:

1. Utwórz spersonalizowaną funkcję ze zmiennymi odpowiadającymi wygranym, przegranym oraz remisom.
2. Napisz `return`, a następnie nazwy wybranych zmiennych. Skorzystaj z przecinka, aby je rozdzielić:

```
local function getWinRate()
    local wins = 4
    local losses = 0
    local ties = 1
    return wins, losses, ties
end
```

3. Zamiast tworzyć zmienne dla każdej otrzymanej wartości w osobnych wierszach, utwórz je w tym samym wierszu, jak w poniższym przykładzie. Zostaną „wypełnione” zwracanymi wartościami w odpowiedniej kolejności:

```
local function getWinRate()
    local wins = 4
    local losses = 0
    local ties = 1
    return wins, losses, ties
end
local userWins, userLosses, userTies = getWinRate()
```

4. Wydrukuj zmienne, aby zobaczyć wyniki.

```
local function getWinRate()
    local wins = 4
    local losses = 0
    local ties = 1
    return wins, losses, ties
end

local userWins, userLosses, userTies = getWinRate()
print("Your wins, losses, and ties are: " .. userWins .. " , " .. userLosses .. " , " .. userTies)
```

Zwracanie nil

Nil oznacza coś, czego nie da się znaleźć albo coś, co nie istnieje. Jeśli widzisz `nil` zamiast oczekiwanego wyniku, wykonaj następujące czynności:

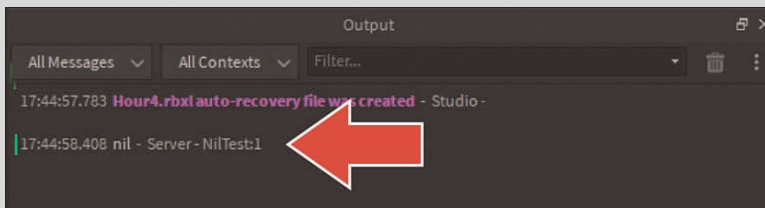
- ▶ Sprawdź, czy liczba otrzymanych wartości jest taka sama jak tych zwróconych.
- ▶ Sprawdź, czy wartości zwracane i odbierane są oddzielone przecinkami.
- ▶ Sprawdź, czy z funkcją wszystko w porządku.

▼ SPRÓBUJ SAM

Zwracanie czegoś, co nie istnieje

Jeśli spróbujesz użyć zmiennej lub funkcji, która nie istnieje, w oknie *Output* wyświetlone zostaną słowo kluczowe `nil` oraz informacja o miejscu, w którym wystąpił błąd.

1. W dowolnym skrypcie przekaz do funkcji `print()` fałszywą nazwę zmiennej, jak `doesntExist`.
2. Uruchom kod i sprawdź okno *Output*. Słowo kluczowe `nil` powinno się pojawić obok nazwy skryptu i numeru wiersza, w którym zmienna nie mogła zostać znaleziona, tak jak w przypadku błędu pokazanego na rysunku 4.5.



RYСУNEK 4.5. Komunikat o błędzie mówi, że wartość w pierwszej linii skryptu, o nazwie `NilTest`, nie została znaleziona

Radzenie sobie z niezgodnymi argumentami i parametrami

Ważne, aby zdawać sobie sprawę, co się stanie, jeśli zostanie przekazana do funkcji lub zostanie z niej zwrócona niewłaściwa liczba wartości. Może to spowodować błędy i zawieszenie kodu.

Jeśli do funkcji zostanie przekazana niewystarczająca liczba argumentów, wystąpi błąd, gdy funkcja osiągnie wartość `nil`:

```
local function whoWon (first, second)
  print("First place is " .. first .. "Second place is ")
end
```

`whoWon("AngelicaIsTheBest")` -- Spowoduje błąd, ponieważ nie podano drugiego parametru

Jeśli przekazemy więcej wartości, niż jest dostępnych zmiennych, wartości zostaną uzupełnione w podanej kolejności, a wszelkie pozostałe zmienne zostaną usunięte i utracone. W tym przykładzie przekazywane są trzy wartości, ale miejsce jest tylko dla dwóch:

```
local function giveBack()  
    local a = "Apple"  
    local b = "Banana"  
    local c = "Carrot"  
    return a, b, c  
end  
  
local a, b = giveBack() -- c zostało zagubione  
  
print(a, b, c) -- Wyświetli Apple, Banana, nil
```

"Carrot" istnieje tylko w obrębie funkcji i nigdy nie jest zwracane, więc dla trzeciej wartości wyświetlone zostanie nil.

Funkcje anonimowe

Funkcje anonimowe to, jak sama nazwa wskazuje, funkcje. Wyjątkowymi czyni je to, że kiedy są definiowane po raz pierwszy, pozostają bez nazwy. Oznacza to, że można je zdefiniować w tym samym miejscu, w którym są wywoływane. Porównaj dwa następujące fragmenty kodu z już znaną Ci prostą pułapką połączoną ze zdarzeniem Touched. Zdarzenie Touched zwraca nazwę części wyzwalającej, która jest następnie niszczona.

Oto skrypt, w którym funkcja nazwana jest tworzona i następnie wywoływana za każdym razem, gdy wyzwala jest zdarzenie Touched:

Przykład z funkcją nazwaną

```
local trap = script.Parent  
  
local function onTouch(otherPart)  
    otherPart:Destroy()  
end  
  
trap.Touched:Connect(onTouch)
```

Oto kod, który robi to samo, tyle że funkcja jest tworzona w miejscu, w którym jest wywoływana:

Przykład z funkcją anonimową

```
local part = script.Parent  
  
part.Touched:Connect(function(otherPart)otherPart:Destroy() end)
```

Gdyby uruchomić oba fragmenty kodu, zrobiłyby one dokładnie to samo: zniszczyłyby wszystko, co dotyczyłoby rodzica skryptu. Dlaczego więc nie używać funkcji anonimowej? W tabeli przedstawiono niektóre zalety i wady funkcji nienazwanych.

Zalety	Wady
Szybsze w pisaniu.	Trudniejsze do zrozumienia.
Mogą być stosowane do tworzenia funkcji, które inaczej nie zwróciłyby wartości.	Trudniejsze we wprowadzaniu zmian oraz w ponownym użyciu.
	Nie mogą być wywoływane z innego miejsca, ponieważ nie mają nazwy, przez którą można by się do nich odnieść.

Wskazówka

Funkcje nazwane ułatwiają współpracę

Przewodnik *Roblox Lua Style Guide* odradza korzystanie z funkcji anonimowych, gdy nie jest to konieczne, ponieważ większość projektów ma wielu programistów, a funkcje anonimowe znacznie utrudniają odczytywanie i aktualizowanie kodu.

Podsumowanie

Funkcje mogą być używane i ponownie wykorzystywane na wiele różnych sposobów. Można ich użyć do stworzenia jakiejś rzeczy, na przykład NPC w godzinie 2. Można je wykorzystać do wprowadzania zmian w obiektach poprzez aktualizację właściwości lub nawet całkowite zniszczenie obiektów, jak w przypadku części pułapki. Aby to zrobić, funkcje mogą pobierać zmienne spoza swojego zakresu, poprzez przekazywanie przez programistę ich wartości jako parametrów. Rzeczywiste porcje informacji, które są przekazywane przez parametry, nazywane są argumentami.

Gdy funkcja zakończy swoją pracę, wyniki mogą zostać zwrócone i wykorzystane przez skrypt. Klasycznym przykładem zwracania informacji jest kalkulator w telefonie. Jeśli użyjesz kalkulatora do dodania dwóch liczb, zwróci on odpowiedź. Innym przykładem, który widzieliśmy, jest to, że za każdym razem, gdy uruchamiane jest zdarzenie Touched, przekazuje ono nazwę obiektu, który spowodował jego uruchomienie.

Czasami, jeśli nie ma nic do zwrócenia, możesz użyć funkcji anonimowej i utworzyć funkcję w tym samym miejscu, w którym jest wywoływana. Może to być wygodne, ale sprawia też, że kod jest znacznie trudniejszy do zrozumienia, co może spowolnić pracę członków zespołu, którzy zajmują się skryptem. Może to nawet Ci utrudnić pracę, jeśli zechcesz kiedyś dokonać aktualizacji kodu.

Pytania i odpowiedzi

P. Czy istnieje maksymalna liczba parametrów funkcji?

O. Nie ma ścisłego maksimum, ale w większości przypadków będziesz chciał ograniczyć je do trzech. Im więcej parametrów, tym łatwiej pomylić kolejność i tym trudniej zapamiętać, do czego służy każdy z nich.

Warsztaty

Teraz, gdy skończyłeś(-łaś), przyjrzyjmy się temu, czego się nauczyłeś(-łaś). Poświęć chwilę na to, aby odpowiedzieć na następujące pytania.

Quiz

1. Podawanie informacji z zewnątrz funkcji do wewnątrz to _____.
2. Jakie słowo kluczowe umożliwia zwracanie wartości po wykonaniu funkcji?
3. Symbole zastępcze dla wartości, które będą później używane przez funkcję, to _____.
4. Rzeczywiste wartości używane przez funkcje to _____.
5. Słowo kluczowe używane, gdy nie można znaleźć wartości lub gdy ona nie istnieje, to _____.

Odpowiedzi

1. Przekazywanie argumentów.
2. return.
3. Parametry.
4. Argumenty.
5. nil.

Ćwiczenie

Wszyscy programiści czasami sprawdzają, jaki jest efekt pracy innych osób. Jednak kod znaleziony w internecie może nie działać dokładnie tak, jak tego chcesz, lub może być nie-sformatowany w sposób czytelny dla członków Twojego zespołu. Ważne, abyś poświęcił(a) czas na sprawdzenie zapożyczonego kodu i wprowadzenie ulepszeń tam, gdzie to możliwe. W tym przykładzie wykonaj ćwiczenie, próbując przeformatować funkcję anonimową na funkcję nazwaną:

```
script.Parent.Touched:Connect(function(otherPart) local fire = Instance.new"Fire"  
fire.Parent = otherPart end)
```

Rozwiązanie znajduje się w dodatku.

PROGRAM PARTNERSKI

— GRUPY 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>

GRUPA
Helion

ROBLOX to popularna platforma do tworzenia gier, z której korzystają użytkownicy o różnym poziomie umiejętności w zakresie ich projektowania. W Robloxie używa się języka programowania Lua. Jest to język, którego można się szybko i łatwo nauczyć. Roblox Studio i Lua zapewniają idealne środowisko dla twórców gier, a także prosty dostęp do serwerów multiplayer, narzędzi do modelowania światów, systemów monetyzacji i wielu innych przydatnych funkcji. Ty możesz skupić się na tym, co najlepsze: na pracy twórczej!

Oto wyjątkowy przewodnik po platformie Roblox. Składa się z 24 lekcji skonstruowanych w sposób ułatwiający szybkie opanowanie materiału; przeczytanie każdej z nich i wykonanie podanych ćwiczeń zajmie Ci najwyżej godzinę. Lekcje i zadania łącznie tworzą kurs, dzięki któremu nauczysz się samodzielnie kodować w Robloxie. Dzięki tej książce przygotujesz sobie środowisko pracy, zaczniesz się płynnie posługiwać niezbędnymi narzędziami i szybko napiszesz swoją pierwszą grę. Naukę ułatwią Ci instrukcje krok po kroku, liczne ćwiczenia i pytania sprawdzające, a także quizy, wskazówki i przydatne ostrzeżenia. Szybko się przekonasz, jak proste, przyjemne i satysfakcjonujące jest używanie Robloksa!

W książce między innymi:

- podstawowe informacje o platformie Roblox, Roblox Studio i języku Lua
- właściwości, zmienne, funkcje, instrukcje warunkowe i pętle
- korzystanie z tabel i ze słowników
- programowanie zdarzeń
- abstrakcje i wprowadzenie do programowania zorientowanego obiektowo
- utrwalanie danych i użycie ich w grach
- stosowanie ray castingu

Roblox Studio: tak się dzisiaj tworzy gry!

Genevieve Johnson jest starszą projektantką materiałów edukacyjnych dla Robloksa. Wcześniej była kierownikiem do spraw treści edukacyjnych w iD Tech, programie edukacji technicznej. W ramach tej inicjatywy współtworzyła program Steam dla dziewcząt, a jej zespół opracował treści edukacyjne dla ponad 60 kursów z różnych dziedzin, od programowania po robotykę i projektowanie gier.

Helion 



helion.pl



HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-283-9449-0



Cena: 69,00 zł