

KYLE SIMPSON

NA DRODZE *do* BIEGŁOŚCI

TAJNIKI JĘZYKA

JavaScript
JS

Tytuł oryginału: You Don't Know JS: Up & Going

Tłumaczenie: Robert Górczyński

ISBN: 978-83-283-2173-1

© 2016 Helion SA.

Authorized Polish translation of the English edition of You Don't Know JS: Up & Going
ISBN 9781491924464 © 2015 Getify Solutions, Inc.

This translation is published and sold by permission of O'Reilly Media, Inc.,
which owns or controls all rights to publish and sell the same.

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 the Publisher.

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)

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

Printed in Poland.

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

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

Spis treści

Przedmowa	5
Wprowadzenie	7
1. Wprowadzenie do programowania	13
Kod	14
Wyrażenia	15
Wypróbuj samodzielnie	16
Operatory	20
Wartości i typy	22
Komentarze w kodzie	25
Zmienne	27
Bloki	30
Konstrukcje warunkowe	31
Pętle	33
Funkcje	35
Praktyka	39
Podsumowanie	41

2. Wstęp do programowania w JavaScript	43
Wartości i typy	44
Zmienne	55
Konstrukcje warunkowe	58
Tryb ścisły	60
Funkcje jako wartości	62
Identyfikator this	67
Prototypy	69
Stare i nowe	70
Wykraczamy poza JavaScript	75
Podsumowanie	76
3. O serii TJJS	77
Zakresy i domknięcia	78
Wskaźnik this i prototypy obiektów	78
Typy i składnia	80
Asynchroniczność i wydajność	81
ES6 & Beyond	83
Podsumowanie	84
A Podziękowania	85
Skorowidz	89

Wstęp do programowania w JavaScript

W poprzednim rozdziale przedstawiłem podstawowe elementy konstrukcyjne stosowane w programowaniu, czyli zmienne, pętle, konstrukcje warunkowe i funkcje. Oczywiście cały prezentowany kod został utworzony w języku JavaScript. W tym rozdziale chcę się skoncentrować na rzeczach, które musisz wiedzieć o JavaScript, jeśli zamierzasz stać się programistą korzystającym z tego języka.

Zaprezentuję kilka koncepcji, które dokładnie będą wyjaśnione w kolejnych książkach z tej serii. Potraktuj ten rozdział jako ogólne przedstawienie tematów, których szczegółowe omówienie znajdziesz w pozostałej części serii.

Jeżeli dopiero zaczynasz przygodę z językiem JavaScript, to będziesz musiał poświęcić немало czasu na wielokrotne przeanalizowanie przedstawionych tutaj koncepcji i fragmentów kodu. Każdy dobry fundament jest kładziony cegła po cegle, a więc nie oczekuj, że od razu wszystko będzie zrozumiałe.

W tym miejscu rozpoczynasz podróż, której celem jest dokładne poznanie języka JavaScript.



Jak wspomniałem w rozdziale 1., podczas lektury zdecydowanie powinieneś samodzielnie wypróbować kod przedstawiony w książce. Pamiętaj, że w niektórych fragmentach kodu przyjąłem dostępność funkcjonalności wprowadzonej w najnowszych (gdy powstawała ta książka) wersjach JavaScript (najczęściej można spotkać określenie „ES6” dla szóstego wydania ECMAScript — to oficjalna nazwa specyfikacji JavaScript). Jeżeli używasz starszej wersji przeglądarki internetowej, opracowanej przed wydaniem ES6, przedstawiony w książce kod może nie działać. W takim przypadku warto zastosować najnowszą wersję jednej z najważniejszych przeglądarek, takich jak Chrome, Firefox lub IE.

Wartości i typy

Jak już wspomniałem w rozdziale 1., JavaScript stosuje typy dla wartości, a nie dla zmiennych. Dostępne są następujące wbudowane typy danych:

- string
- number
- boolean
- null i undefined
- object
- symbol (nowość w ES6)

JavaScript oferuje operator `typeof`, który może przeanalizować wartość i podać jego typ:

```
var a;
typeof a;           // "undefined"

a = "Witaj, świecie!";
typeof a;           // "string"

a = 42;
typeof a;           // "number"

a = true;
typeof a;           // "boolean"

a = null;
typeof a;           // "object" <-- Dziwne, błąd.

a = undefined;
typeof a;           // "undefined"

a = { b: "c" };
typeof a;           // "object"
```

Wartość zwrótna operatora `typeof` to zawsze jedna z sześciu (siedmiu w specyfikacji ES6) wartości w postaci ciągu tekstowego. Oznacza to, że wynikiem wywołania `typeof "abc"` jest `"string"`, a nie `string`.

Zwróć uwagę, że w powyższym fragmencie kodu zmienna `a` przechowuje wartości różnych typów. Pomimo wydania polecenia `typeof a` nie sprawdzamy „typu `a`”, ale „typ wartości aktualnie przechowywanej w `a`”. W języku JavaScript tylko wartości mają przypisany typ; zmienne są po prostu kontenerami przeznaczonymi dla tych wartości.

Interesującym przypadkiem jest `typeof null`, ponieważ otrzymujemy wartość zwrótną `"object"` zamiast oczekiwanej `"null"`.



To znany od dawna błąd w języku JavaScript, ale nie został dotąd usunięty. Ponieważ duża ilość kodu w internecie opiera swoje działanie na tym błędzie, więc jego usunięcie wprowadziłoby jeszcze więcej błędów!

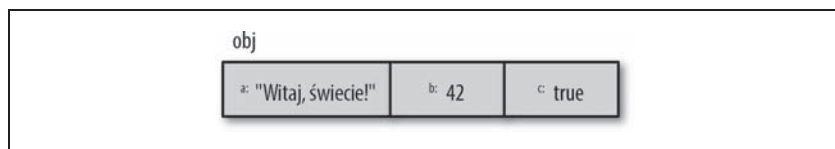
Zwróć również uwagę na `a = undefined`. Zmienną `a` wyraźnie przypisujemy wartość `undefined`, ale tak naprawdę nie różni się to od sytuacji, gdy zmienna w ogóle nie ma przypisanej wartości, czyli jak w wierszu `var a;` na początku omawianego fragmentu kodu. Zmienna może przejść do stanu „niezdefiniowanego” na wiele różnych sposobów, między innymi na skutek działania funkcji, która nie zwraca wartości, i przez użycie operatora `void`.

Obiekty

Typ `Object` odnosi się do wartości złożonej, gdzie można przypisać zbiór właściwości (nazwanych lokalizacji), z których każda będzie przechowywała własną wartość dowolnego typu. To prawdopodobnie jeden z najużyteczniejszych typów wartości w całym języku JavaScript:

```
var obj = {  
  a: "Witaj, świecie!",  
  b: 42,  
  c: true  
};  
  
obj.a;      // "Witaj, świecie!"  
obj.b;      // 42  
obj.c;      // Prawda  
obj["a"];   // "Witaj, świecie!"  
obj["b"];   // 42  
obj["c"];   // Prawda
```

Pomocne może być graficzne przedstawienie wartości `obj`, jak na rysunku 2.1.



Rysunek 2.1. Graficzne przedstawienie obiektu `obj`

Dostęp do właściwości może odbywać się za pomocą *składni z użyciem kropki* (na przykład `obj.a`) lub też *składni z użyciem nawiasu* (na przykład `obj["a"]`). Składnia z użyciem kropki jest krótsza i generalnie łatwiejsza w odczycie, dlatego też jest preferowana, o ile to tylko możliwe.

Z kolei składnia z użyciem nawiasu okazuje się przydatna, gdy nazwy właściwości zawierają znaki specjalne, takie jak `obj["Witaj, świecie!"]` — w kontekście omawianej składni tego rodzaju właściwości są najczęściej określane mianem *kluczy*. Notacja `[ ]` wymaga zmiennej (wyjaśnię to później) lub *literału* w postaci ciągu tekstowego (konieczne jest jego opakowanie w `" .. "` lub `' .. '`).

Oczywiście składnia z użyciem nawiasu jest przydatna, jeśli chcesz uzyskać dostęp do właściwości lub klucza, gdy nazwa jest przechowywana w innej zmiennej, na przykład:

```
var obj = {  
  a: "Witaj, świecie!",  
  b: 42  
};  
  
var b = "a";  
  
obj[b];           // "Witaj, świecie!"  
obj["b"];         // 42
```



Więcej informacji na temat obiektów w JavaScript znajdziesz w książce *Wskaźnik this i prototypy obiektów* z tej serii, w szczególności w rozdziale 3.

Mamy jeszcze kilka innych typów wartości często stosowanych w programach JavaScript: *tablice* i *funkcje*. Jednak zamiast traktować je jako poprawne wbudowane typy, tablice i funkcje, można traktować je bardziej jak podtypy — specjalizowane wersje typu `object`.

Tablice

Tablica to obiekt przechowujący wartości (dowolnego typu) nie w nazwanych właściwościach lub kluczach, ale w liczbowo zindeksowanych pozycjach. Na przykład:

```
var arr = [  
  "Witaj, świecie!",  
  42,  
  true
```

```
];

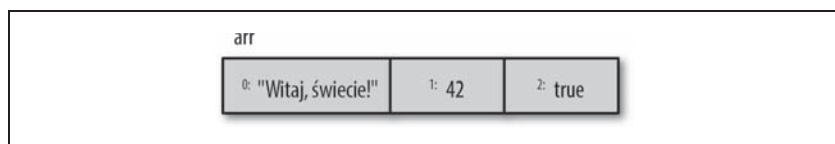
arr[0];      // "Witaj, świecie!"
arr[1];      // 42
arr[2];      // Prawda
arr.length;  // 3

typeof arr;  // "object"
```



W językach programowania, w których zaczynamy odliczanie od zera, na przykład w JavaScript, pierwszy element tablicy ma indeks 0.

Na rysunku 2.2 przedstawiłem graficzną postać tablicy `arr`.



Rysunek 2.2. Graficzne przedstawienie tablicy `arr`

Ponieważ tablice są obiektami specjalnymi (jak wskazuje na to `typeof`), mogą mieć również właściwości. Przykładem jest właściwość `length`, która przechowuje automatycznie uaktualnianą wartość wskazującą wielkość tablicy.

Teoretycznie tablicę można wykorzystać jak zwykły obiekt wraz z właściwościami, ewentualnie można użyć zwykłego obiektu (`object`) i definiować w nim jedynie właściwości o nazwach liczbowych (0, 1 itd.), symulując w ten sposób tablicę. Jednak w takich przypadkach uznaje się, że doszło do nieprawidłowego użycia wymienionych typów.

Najlepsze i najnaturalniejsze podejście polega na wykorzystaniu tablic do przechowywania wartości zindeksowanych liczbowo, natomiast obiektów — do przechowywania nazwanych właściwości.

Funkcje

Innym podtypem `object` często stosowanym w programach JavaScript jest funkcja:

```
function foo() {
    return 42;
}
```

```
foo.bar = "Witaj, świecie!";

typeof foo;           // "function"
typeof foo();         // "number"
typeof foo.bar;       // "string"
```

Jak możesz zobaczyć, funkcja jest podtypem object — operator `typeof` zwraca wartość `"function"` oznaczającą, że `function` jest typem głównym — a więc może mieć właściwości. Jednak właściwości obiektu funkcji (na przykład `foo.bar`) są używane sporadycznie.



Więcej informacji na temat wartości JavaScript oraz ich typów znajdziesz w pierwszych dwóch rozdziałach książki *Typy i składnia* z tej serii.

Wbudowane metody typu

Omówione dotąd typy i podtypy mają pewne zachowanie udostępnione w postaci właściwości i metod, które są całkiem użyteczne i oferują bardzo duże możliwości.

Na przykład:

```
var a = "Witaj, świecie!";
var b = 3.14159;

a.length;           // 11
a.toUpperCase();     // "WITAJ, ŚWIECIE!"
b.toFixed(4);        // "3.1416"
```

„Mechanizm” pozwalający na wywołanie `a.toUpperCase()` jest znacznie bardziej skomplikowany niż metoda istniejąca dla danej wartości.

Pokrótcie: mamy obiekt opakowujący `String` (nazwa rozpoczyna się od dużej litery S), zwykle nazywany „natywnym”, który tworzy parę wraz z typem podstawowym `string`. W swoim prototypie ten obiekt opakowujący definiuje metodę `toUpperCase()`.

Kiedy używasz wartości typu podstawowego, takiej jak `"Witaj, świecie!"`, jako obiektu przez odwołanie się do właściwości lub metody (na przykład `toUpperCase()` w powyższym fragmencie kodu), to JavaScript automatycznie „opakowuje” wartość obiektem opakującym (ukrytym w tle).

A

API DOM, 75
ASM.js, 82
asynchroniczność, 81

B

blok, 30
 case, 59
błąd ReferenceError, 57

C

ciąg tekstowy, 23

D

dane, 18
delegacja, 70
DOM, 75
domknięcie, 64, 65, 66, 67
dziedziczenie, 79

E

egzemplarz, 66
Eich Brendan, 7

F

formularz sieciowy, 18
funkcja, 18, 23, 37, 47
 alert, 75
 anonimowa, 62
 argument, 36
 deklarowanie, 62
 jako wyrażenie, 62, 63
 log, 18
 nazwa, 55
 nazwana, 62
 Number, 24
 Number.isNaN, 71
 parametr, *Patrz:* funkcja argument
 prompt, 19
 toFixed, 29
 zasięg, 37

G

generator, 81, 82

H

hoisting, 55, 78
 deklaracji funkcji, 56
 zmiennej, 56

I

identyfikator, 55
IIFE, 63
immediately invoked function
 expression, *Patrz:* IIFE
interpreter, 16, 26
inversion of control, *Patrz:* IoC
IoC, 81

J

JavaScript silnik, *Patrz:* silnik
język
 interpretowany, 16, 78
 programowania, 14

K

klasa, 79
kod, 14
 blok, *Patrz:* blok
 dokumentowanie, 26, 27
 interpretacja, 16
 kompilacja, 16
 sekcja, 36
 źródłowy, *Patrz:* kod
koercja
 na boolean, 50
 niejawna, 49
 wyrażna, 49
komentarz
 jednowierszowy, 26
 wielowierszowy, 26, 27
kompilacja, 25
kompilator, 16, 25, 26
konsola programistyczna, 16, 17
konstrukcja warunkowa, 31, 33, 58

L

liczba, 23
literał, 23

M

mechanizm delegacji, 79
metoda, 48
 getElementById, 75
 toUpperCase, 48
model DOM, 75
moduł, 66
 egzemplarz, *Patrz:* egzemplarz

O

obiekt, 23
 Boolean, 49
 Number, 49
 opakowujący, 48
 prototyp, 79
 String, 49
obietnica, 81, 82
odwrócenie kontroli, *Patrz:* IoC
operator, 14
 *, 20
 ? :, 60
 =, 20
 dekrementacji, 21
 inkrementacji, 21
 matematyczny, 21
 nierówności, 52
 logiczny, 22
 luźnej, 22
 porównania, 22
 ściślej, 22
 przypisania, 21
 równości, 51, 52, 53
 luźnej, 22, 24
 ściślej, 22
 typeof, 44, 48
 void, 45

P

- pętla, 33
 - do-while, 33, 34, 35
 - for, 35
 - while, 33, 34, 35
- polecenie, 14
 - alert, 18
 - break, 34, 59
 - console.log, 18, 19
 - if, 31, 58
 - if-else, 32, 33, 58, 59
 - prompt, 20
 - switch, 33, 58, 60
 - use strict, 60
 - var, 28
 - yield, 81
- polyfill, 71, 72
- program, 14, 15
 - uruchamianie, 16
- programowanie asynchroniczne, 81
- prototyp, 69
- przeglądarka internetowa, 16, 17, 71, 72, 73, 74, 75

R

- równoległość, 81
 - danych, 82
 - działania programu, 82

S

- Sharp Remy, 71
- silnik, 78
- SIMD, 82
- składnia, 14, 72
 - z użyciem
 - kropki, 46
 - nawiasu, 46
- skrypt polyfill, *Patrz*: polyfill

- słowo kluczowe, 55
 - const, 30
 - this, 67, 68, 69, 78, 79
 - var, 21, 30, 55
- słowo zarezerwowane, 55
- stała, 29, 30

T

- tablica, 23, 46, 53
- technika optymalizacji, 82
- test wydajności, 82
- transpiling, 72, 73, 74
- tryb ścisły, 60, 61
- typ, 23
 - boolean, 23, 44, 49
 - koercja, 24, 80
 - niejawna, 24
 - wyraźna, 24
 - konwersja, 24
 - null, 44, 45
 - number, 23, 24, 44, 49
 - object, 44, 45, 47, 53
 - string, 23, 24, 44, 49
 - symbol, 44
 - undefined, 44, 45
 - wymuszenie, 27
- typowanie
 - dynamiczne, 27, 28
 - słabe, 27
 - statyczne, 27, 30

W

- wartość
 - boolowska, 23
 - NaN, 72
 - podstawowa, 23
 - porównywanie, 49
 - złożona, 45

- właściwość, 48
 - length, 47
 - nazwa, 55
- współbieżność, 81
- wyrażenie, 15
- wywołanie zwrotne, 81, 82
- wzorzec
 - modułu, 66
 - projektowy, 79
 - zorientowany obiektowo, 67

Z

- zakres, 37, 55, 78
 - zagnieżdżony, 56
- zmienna, 27
 - deklarowanie, 21, 55, 57
 - nazwa, 21, 38, 55
 - typ, 27, *Patrz też*: typ
 - zakres, *Patrz*: zakres
- znak
 - ' , 23
 - !=, 22, 51, 52
 - !==, 22, 51
 - " , 23
 - &&, 22

- (), 63
- *, 20, 21
- */ , 26
- *=, 21
- ., 22
- / , 21
- /* , 26
- /=, 21
- ? : , 60
- { } , 30
- || , 22
- + , 21
- ++ , 21
- +=, 21
- < , 22, 53
- <=, 22, 53
- = , 20, 21
- =, 21
- ==, 22, 24, 51, 52, 53
- ===, 22, 51, 52, 53
- > , 22, 53
- >=, 22, 53
- ASCII, *Patrz*: ASCII
- średnik, 14
- Unicode, *Patrz*: Unicode

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>

GRUPA WYDAWNICZA

 **Helion SA**

PAMIĘTAJ, TWÓJ KOD JEST TWOIM DZIEŁEM!

Początkujący programista może bardzo szybko zacząć tworzyć proste aplikacje w JavaScriptcie. Jednak ten, kto chce osiągnąć prawdziwą biegłość w tym języku, musi opanować trudniejsze zagadnienia, na przykład asynchroniczność czy techniki związane z wydajnością w JavaScriptcie, takie jak obietnice, generatory i wątki robocze. Powinien nie tylko umieć napisać kod, który działa, ale także bardzo dokładnie rozumieć, *dla czego* i *w jaki sposób* działa. Okazuje się, że w przypadku JavaScriptu bardzo często nie jest to ani proste, ani oczywiste.

Niniejsza książka jest pierwszą częścią serii w całości poświęconej temu językowi. Autor skupia się na trudniejszych aspektach JavaScriptu, dogłębnie je analizuje, a następnie przedstawia praktyczne zastosowanie opisanych koncepcji. Książka jest przeznaczona dla osób dopiero rozpoczynających przygodę z programowaniem. Omówiono w niej istotniejsze koncepcje programowania i podano głębokie podstawy języka JavaScript. Jednym zdaniem, książka ta umożliwia na bardzo wysokim poziomie zrozumienie i przyswojenie najważniejszych zasad rządzących tym językiem.

Dzięki tej książce:

- poznasz najważniejsze koncepcje dotyczące programowania
- nauczysz się stosować takie elementy programistyczne jak typy, zmienne, konstrukcje warunkowe, pętle i funkcje
- poznasz podstawy JavaScriptu
- nauczysz się stosować najważniejsze mechanizmy JavaScriptu: wartości, domknięcia funkcji, słowo kluczowe *this* oraz prototypy
- przekonasz się, czy warto zgłębić trudniejsze aspekty JavaScriptu, i dowiesz się, jak Ci w tym pomogą pozostałe książki z tej serii

KYLE SIMPSON

— jest Teksańczykiem, propagatorem Open Web i wielkim pasjonatem wszystkiego, co związane z językiem JavaScript. Ma dar przekazywania wiedzy, a przy tym zaraża entuzjazmem. Píše książki, prowadzi warsztaty, występuje na konferencjach o tematyce technicznej oraz pozostaje aktywnym członkiem społeczności OSS.



42042	numer katalogowy	Sprawdź najnowsze promocje: ● http://helion.pl/promocje Książki najchętniej czytane: ● http://helion.pl/bestsellery Zamów informacje o nowościach: ● http://helion.pl/novosci
	księgarnia internetowa	
	http://helion.pl	
	zamówienia telefoniczne	
	0 801 339900	Helion SA ul. Koszalska 1c, 44-100 Gliwice tel.: 32 230 98 63 e-mail: helion@helion.pl http://helion.pl
	0 601 339900	
Informatyka w najlepszym wydaniu		

