

O'REILLY®

Wydanie V



PHP, MySQL i JavaScript Wprowadzenie



Helion 

Robin Nixon

Tytuł oryginału: Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5, 5th Edition

Tłumaczenie: Piotr Cieślak

ISBN: 978-83-283-5149-3

© 2019 Helion S.A.

Authorized Polish translation of the English edition of Learning PHP, MySQL & JavaScript 5e ISBN 9781491978917 © 2018 Robin Nixon

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

Helion SA

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/phmyj5>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Przedmowa	23
1. Wstęp do dynamicznych stron internetowych	27
HTTP i HTML: podstawy wynalazku Bernersa-Lee	28
Procedura żądanie/odpowiedź	28
Zalety PHP, MySQL, JavaScriptu, CSS i HTML5	31
MariaDB — klon MySQL	32
Zastosowanie PHP	32
Zastosowanie MySQL	33
Zastosowanie JavaScriptu	34
Zastosowanie CSS	35
I HTML5 na dokładkę	36
Serwer WWW Apache	37
Obsługa urządzeń mobilnych	37
Kilka słów o open source	38
Zgrany zespół	38
Pytania	40
2. Konfigurowanie serwera	41
WAMP, MAMP, LAMP — a cóż to takiego?	42
Instalowanie pakietu AMPPS w systemie Windows	42
Testowanie instalacji	46
Dostęp do katalogu głównego w systemie Windows	48
Inne pakiety WAMP	49
AMPPS i macOS	49
Dostęp do katalogu głównego w systemie macOS	50
Instalowanie pakietu LAMP pod Linuxem	51
Praca zdalna	52
Logowanie	52
Obsługa FTP	52

Obsługa edytora kodu	53
Obsługa środowiska IDE	54
Pytania	56
3. Wstęp do PHP	57
Dodawanie elementów PHP do kodu HTML	57
Przykłady z tej książki	58
Składnia PHP	59
Zastosowanie komentarzy	59
Podstawowa składnia	60
Zmienne	61
Operatory	65
Przypisywanie wartości zmiennym	68
Instrukcje wielowierszowe	71
Deklaracja typu zmiennych	73
Stałe	74
Stałe predefiniowane	74
Różnica między instrukcjami echo i print	75
Funkcje	76
Zasięg zmiennych	77
Pytania	81
4. Wyrażenia i sterowanie działaniem programu w PHP	83
Wyrażenia	83
Prawda czy fałsz?	83
Literały i zmienne	85
Operatory	86
Priorytet operatorów	86
Asocjacyjność	88
Operatory relacji	89
Wyrażenia warunkowe	93
Instrukcja if	93
Instrukcja else	95
Instrukcja elseif	96
Instrukcja switch	97
Operator ?	99
Pętle	101
Pętla while	101
Pętla do ... while	103
Pętla for	103
Przerywanie pętli	105
Instrukcja continue	106

Rzutowanie jawne i niejawne	106
Dynamiczne linkowanie w PHP	107
Dynamiczne linkowanie w praktyce	108
Pytania	109
5. Funkcje i obiekty w PHP	111
Funkcje PHP	112
Definiowanie funkcji	113
Zwracanie wartości	113
Zwracanie tablicy	115
Przekazywanie argumentów przez referencję	115
Zwracanie zmiennych globalnych	117
Przypomnienie informacji o zasięgu zmiennych	117
Dołączanie i wymaganie plików	118
Instrukcja include	118
Zastosowanie instrukcji include_once	118
Zastosowanie instrukcji require i require_once	119
Sprawdzanie zgodności wersji PHP	119
Obiekty w PHP	120
Terminologia	120
Deklarowanie klasy	121
Tworzenie obiektu	122
Odwoływanie się do obiektów	122
Klonowanie obiektów	124
Konstruktory	125
Destrukctory	125
Tworzenie metod	126
Deklarowanie właściwości	126
Deklarowanie stałych	127
Zasięg właściwości i metod	128
Metody statyczne	129
Właściwości statyczne	129
Dziedziczenie	130
Pytania	133
6. Tablice w PHP	135
Prosty dostęp	135
Tablice indeksowane numerycznie	135
Tablice asocjacyjne	137
Dodawanie pozycji do tablicy przy użyciu słowa kluczowego array	137

Pętla foreach ... as	138
Tablice wielowymiarowe	140
Zastosowanie funkcji do obsługi tablic	143
is_array	143
count	143
sort	143
shuffle	144
explode	144
extract	145
compact	146
reset	147
end	147
Pytania	147
7. PHP w praktyce	149
Zastosowanie funkcji printf	149
Określanie precyzji	150
Dopełnianie łańcuchów tekstowych	152
Zastosowanie funkcji sprintf	153
Funkcje do obsługi daty i czasu	153
Stałe związane z datą	154
Zastosowanie funkcji checkdate	156
Obsługa plików	156
Sprawdzanie istnienia pliku	157
Tworzenie pliku	157
Odczytywanie zawartości plików	158
Kopiowanie plików	160
Przenoszenie pliku	160
Kasowanie pliku	160
Aktualizowanie plików	161
Ochrona plików przed wielokrotnym otwarciem	162
Odczytywanie całego pliku	163
Wysyłanie plików	164
Wywołania systemowe	169
XHTML czy HTML5?	170
Pytania	171
8. Wstęp do MySQL	173
Podstawy MySQL	173
Podsumowanie pojęć dotyczących baz danych	174

Dostęp do MySQL z poziomu wiersza poleceń	174
Uruchamianie wiersza poleceń	174
Obsługa serwera z poziomu wiersza poleceń	178
Instrukcje MySQL	179
Typy danych	184
Indeksy	193
Tworzenie indeksu	193
Tworzenie zapytań do bazy MySQL	198
Łączenie tabel	206
Zastosowanie operatorów logicznych	209
Funkcje MySQL	209
Dostęp do MySQL za pośrednictwem aplikacji phpMyAdmin	209
Pytania	211
9. Zaawansowana obsługa MySQL	213
Projektowanie bazy	213
Klucze główne, czyli kluczowy element relacyjnych baz danych	214
Normalizacja	215
Pierwsza postać normalna	215
Druga postać normalna	218
Trzecia postać normalna	220
Kiedy nie stosować normalizacji	222
Relacje	223
Jeden do jednego	223
Jeden do wielu	224
Wiele do wielu	224
Bazy danych i anonimowość	226
Transakcje	226
Mechanizmy składowania danych z obsługą transakcji	226
Instrukcja BEGIN	227
Instrukcja COMMIT	228
Instrukcja ROLLBACK	228
Instrukcja EXPLAIN	228
Archiwizacja i przywracanie danych	230
Instrukcja mysqldump	230
Tworzenie pliku z kopią zapasową	231
Odtwarzanie danych z pliku kopii zapasowej	233
Zapisywanie danych w formacie CSV	233
Planowanie tworzenia kopii zapasowych	234
Pytania	234

10. Korzystanie z MySQL za pośrednictwem PHP	237
Tworzenie zapytań do bazy MySQL za pośrednictwem PHP	237
Proces	237
Tworzenie pliku logowania	238
Nawiązywanie połączenia z MySQL	239
Praktyczny przykład	244
Tablica \$_POST	246
Usuwanie rekordu	247
Wyświetlanie formularza	247
Wysyłanie zapytań do bazy danych	248
Działanie programu	249
MySQL w praktyce	250
Tworzenie tabeli	250
Wyświetlanie informacji o tabeli	251
Usuwanie tabeli	252
Dodawanie danych	252
Odczytywanie danych	253
Aktualizowanie danych	253
Usuwanie danych	254
Zastosowanie opcji AUTO_INCREMENT	254
Wykonywanie zapytań pomocniczych	255
Zapobieganie próbom ataków	256
Działania prewencyjne	257
Zastosowanie elementów zastępczych	258
Zapobieganie przekazywaniu niepożądanych danych przez HTML	260
Proceduralny wariant zastosowania mysqli	262
Pytania	263
11. Obsługa formularzy	265
Tworzenie formularzy	265
Odczytywanie przesłanych danych	267
Wartości domyślne	268
Rodzaje pól	269
Oczyszczanie danych wejściowych	276
Przykładowy program	278
Usprawnienia w HTML5	280
Atrybut autocomplete	280
Atrybut autofocus	281
Atrybut placeholder	281
Atrybut required	281
Atrybuty nadpisania	281

Atrybuty width i height	282
Atrybuty min i max	282
Atrybut step	282
Atrybut form	282
Atrybut list	283
Pole wejściowe typu color	283
Pola wejściowe typu number i range	283
Selektory daty i czasu	283
Pytania	284
12. Ciasteczka, sesje i autoryzacja	285
Zastosowanie ciasteczek w PHP	285
Tworzenie ciasteczka	287
Dostęp do ciasteczka	288
Usuwanie ciasteczek	288
Autoryzacja HTTP	288
Przechowywanie loginów i haseł	291
Przykładowy program	293
Obsługa sesji	296
Inicjowanie sesji	297
Kończenie sesji	299
Określanie czasu trwania sesji	300
Bezpieczeństwo sesji	300
Pytania	304
13. Zapoznanie z JavaScriptem	305
JavaScript i tekst w HTML	305
Zastosowanie skryptów w nagłówku dokumentu	307
Starsze i niestandardowe przeglądarki	307
Dołączanie plików JavaScript	309
Debugowanie kodu JavaScript	309
Zastosowanie komentarzy	310
Średniki	310
Zmienne	310
Zmienne znakowe	311
Zmienne numeryczne	311
Tablice	311
Operatory	312
Operatory arytmetyczne	312
Operatory przypisania	313
Operatory porównania	313

Operatory logiczne	314
Inkrementacja i dekrementacja zmiennych oraz skrócony zapis tych operacji	314
Konkatenacja łańcuchów znaków	314
Znaki modyfikujące	315
Typowanie zmiennych	315
Funkcje	316
Zmienne globalne	316
Zmienne lokalne	317
Obiektowy model dokumentu	318
Kolejne zastosowanie symbolu \$	319
Zastosowanie obiektowego modelu dokumentu	320
Kilka słów o document.write	321
Zastosowanie funkcji console.log	321
Zastosowanie funkcji alert	321
Umieszczanie tekstu w elementach HTML	321
Zastosowanie funkcji document.write	321
Pytania	322
14. Wyrażenia i sterowanie działaniem programu w JavaScriptcie	323
Wyrażenia	323
Literały i zmienne	324
Operatory	325
Priorytet operatorów	325
Asocjacyjność	326
Operatory relacji	326
Instrukcja with	329
Zdarzenie onerror	330
Konstrukcja try ... catch	331
Wyrażenia warunkowe	332
Instrukcja if	332
Instrukcja else	332
Instrukcja switch	333
Operator ?	335
Pętle	335
Pętla while	335
Pętla do ... while	336
Pętla for	337
Przerywanie pętli	337
Instrukcja continue	338
Typowanie jawne	339
Pytania	339

15. Funkcje, obiekty i tablice w JavaScriptcie	341
Funkcje w JavaScriptcie	341
Definiowanie funkcji	341
Zwracanie wartości	343
Zwracanie tablicy	344
Obiekty w JavaScriptcie	345
Deklarowanie klasy	345
Tworzenie obiektu	347
Dostęp do obiektów	347
Słowo kluczowe prototype	347
Tablice w JavaScriptcie	350
Tablice numeryczne	350
Tablice asocjacyjne	351
Tablice wielowymiarowe	352
Zastosowanie metod do obsługi tablic	353
Pytania	358
16. Weryfikacja danych i obsługa błędów w JavaScriptcie i PHP	359
Weryfikowanie wprowadzonych danych przy użyciu JavaScriptu	359
Dokument validate.html (część pierwsza)	360
Dokument validate.html (część druga)	362
Wyrażenia regularne	365
Dopasowywanie za pomocą metaznaków	365
Dopasowanie „rozmyte”	366
Grupowanie przy użyciu nawiasów	367
Klasy znaków	367
Określanie zakresu	368
Zaprzeczenie	368
Kilka bardziej skomplikowanych przykładów	368
Podsumowanie metaznaków	371
Modyfikatory ogólne	373
Zastosowanie wyrażeń regularnych w JavaScriptcie	373
Zastosowanie wyrażeń regularnych w PHP	373
Ponowne wyświetlenie formularza po weryfikacji w PHP	374
Pytania	380
17. Zastosowanie komunikacji asynchronicznej	381
Czym jest komunikacja asynchroniczna?	382
Zastosowanie obiektu XMLHttpRequest	382
Twój pierwszy program asynchroniczny	384
Zastosowanie metody GET zamiast POST	388

Przesyłanie żądań XML	390
Zastosowanie bibliotek komunikacji asynchronicznej	395
Pytania	395
18. Wstęp do CSS	397
Importowanie arkusza stylów	398
Importowanie stylów CSS z poziomu HTML	398
Style zagnieżdżone	399
Zastosowanie identyfikatorów ID	399
Zastosowanie klas	399
Zastosowanie średników	400
Reguły CSS	400
Wiele deklaracji	400
Zastosowanie komentarzy	401
Rodzaje stylów	402
Style domyślne	402
Style użytkownika	402
Zewnętrzne arkusze stylów	403
Style wewnętrzne	403
Style bezpośrednie	404
Selektory CSS	404
Selektor typu	404
Selektor potomka	404
Selektor dziecka	405
Selektor identyfikatora	406
Selektor klasy	407
Selektor atrybutu	407
Selektor uniwersalny	408
Selekcja grupowa	408
Dziedziczenie kaskadowe	408
Źródła stylów	409
Metody definiowania reguł	410
Selektory arkuszy stylów	410
Różnica między elementami div i span	412
Jednostki miar	414
Fonty i typografia	416
font-family	416
font-style	417
font-size	417
font-weight	418

Zarządzanie stylami tekstu	418
Efekty tekstowe	418
Odstępy	419
Wyrównanie	419
Wielkość znaków	419
Wcięcia	419
Kolory w CSS	420
Skrócone określenia kolorów	421
Gradientsy	421
Rozmieszczanie elementów	422
Położenie bezwzględne	422
Położenie względne	423
Położenie stałe	423
Pseudoklasy	425
Skracanie reguł	427
Model pudełkowy i układ strony	428
Definiowanie marginesów	428
Definiowanie ramek	430
Definiowanie odstępu	431
Zawartość obiektu	432
Pytania	432
19. Zaawansowane reguły CSS w CSS3	435
Selektory atrybutów	436
Dopasowywanie fragmentów łańcuchów	436
Właściwość box-sizing	437
Tła w CSS3	438
Właściwość background-clip	438
Właściwość background-origin	439
Właściwość background-size	440
Zastosowanie właściwości auto	440
Wiele obrazów w tle	441
Ramki w CSS3	442
Właściwość border-color	442
Właściwość border-radius	443
Cienie	446
Właściwość overflow	446
Układ wielokolumnowy	447
Kolory i przezroczystość	448
Kolory HSL	448
Kolory HSLA	449

Kolory RGB	449
Kolory RGBA	450
Właściwość opacity	450
Efekty tekstowe	450
Właściwość text-shadow	450
Właściwość text-overflow	451
Właściwość word-wrap	451
Fonty internetowe	452
Fonty Google	453
Przekształcenia	454
Przekształcenia 3D	455
Przejścia	456
Właściwości przejść	456
Czas trwania przejścia	457
Opóźnienie przejścia	457
Dynamika przejścia	457
Skrócona składnia	458
Pytania	459
20. Dostęp do CSS z poziomu JavaScriptu	461
Ponowne spotkanie z funkcją getElementById	461
Funkcja O	461
Funkcja S	462
Funkcja C	463
Dołączanie opisanych funkcji	463
Dostęp do właściwości CSS z poziomu JavaScriptu	464
Niektóre typowe właściwości	464
Inne właściwości	465
JavaScript w kodzie HTML	467
Słowo kluczowe this	468
Łączenie zdarzeń i obiektów w skrypcie	468
Odwoływanie się do innych zdarzeń	469
Dodawanie nowych elementów	470
Usuwanie elementów	471
Inne sposoby na dodawanie i usuwanie elementów	472
Zastosowanie przerw	473
Zastosowanie przerwania setTimeout	473
Anulowanie opóźnienia	474
Zastosowanie przerwania setInterval	474
Animacje na bazie przerw	476
Pytania	477

21. Wprowadzenie do jQuery	479
Dlaczego jQuery?	479
Dołączanie jQuery	480
Wybór odpowiedniej wersji	480
Pobieranie	481
Zastosowanie sieci dostarczania treści (CDN)	482
Dostosowywanie jQuery	482
Składnia jQuery	483
Prosty przykład	483
Unikanie konfliktów między bibliotekami	484
Selektory	484
Metoda css	485
Selektor elementów	485
Selektor identyfikatorów	486
Selektor klas	486
Łączenie selektorów	486
Obsługa zdarzeń	486
Oczekiwanie na gotowość dokumentu	488
Funkcje i właściwości związane ze zdarzeniami	489
Zdarzenia blur i focus	489
Słowo kluczowe this	490
Zdarzenia click i dblclick	491
Zdarzenie keypress	492
Przemysłane programowanie	493
Zdarzenie mousemove	494
Inne zdarzenia myszy	496
Inne metody związane z obsługą myszy	497
Zdarzenie submit	498
Efekty specjalne	499
Ukrywanie i wyświetlanie	500
Metoda toggle	501
Stopniowe zanikanie i wyświetlanie	502
Przesuwanie elementów w górę i w dół	502
Animacje	504
Zatrzymywanie animacji	506
Manipulowanie drzewem DOM	507
Różnica między metodami text i html	508
Metody val i attr	508
Dodawanie i usuwanie elementów	509
Dynamiczne stosowanie klas	511

Modyfikowanie wymiarów	512
Metody width i height	512
Metody innerWidth i innerHeight	514
Metody outerWidth i outerHeight	514
Nawigowanie w obrębie drzewa DOM	514
Elementy nadrzędne	515
Elementy potomne	519
Elementy siostrzane	519
Wybieranie poprzedzających i kolejnych elementów	521
Przetwarzanie selekcji w jQuery	522
Metoda is	523
Użycie jQuery bez selektorów	525
Metoda \$.each	525
Metoda \$.map	526
Zastosowanie komunikacji asynchronicznej	526
Zastosowanie metody POST	526
Zastosowanie metody GET	527
Rozszerzenia	528
jQuery User Interface	528
Inne rozszerzenia	528
Pytania	529
22. Wprowadzenie do jQuery Mobile	531
Dołączanie biblioteki jQuery Mobile	532
Zaczynamy	533
Dołączanie stron	534
Dołączanie synchroniczne	535
Odsyłacze w ramach wielostronicowego dokumentu	535
Przejęcia między stronami	536
Stylizowanie przycisków	539
Obsługa list	541
Listy z możliwością filtrowania	542
Separatory list	544
Co dalej?	546
Pytania	546
23. Wstęp do HTML5	549
Obiekt canvas	549
Geolokacja	551
Dźwięk i filmy	552

Formularze	553
Magazyn danych	554
Web workers	554
Pytania	554
24. Obiekt canvas w HTML5	555
Tworzenie elementu canvas i dostęp do niego	555
Funkcja toDataURL	557
Określanie formatu obrazu	558
Metoda fillRect	558
Metoda clearRect	559
Metoda strokeRect	559
Łączenie wymienionych instrukcji	559
Metoda createLinearGradient	560
Szczegółowe informacje o metodzie addColorStop	562
Metoda createRadialGradient	563
Wypełnianie wzorkami	565
Umieszczanie napisów na elemencie canvas	566
Metoda strokeText	567
Własność textBaseLine	567
Własność font	567
Własność textAlign	568
Metoda fillText	568
Metoda measureText	569
Rysowanie linii	570
Własność lineWidth	570
Własności lineCap i lineJoin	570
Własność miterLimit	572
Kreślenie ścieżek	572
Metody moveTo i lineTo	573
Metoda stroke	573
Metoda rect	573
Wypełnianie obszarów	574
Metoda clip	575
Metoda isPointInPath	578
Zastosowanie krzywych	578
Metoda arc	578
Metoda arcTo	581
Metoda quadraticCurveTo	581
Metoda bezierCurveTo	583

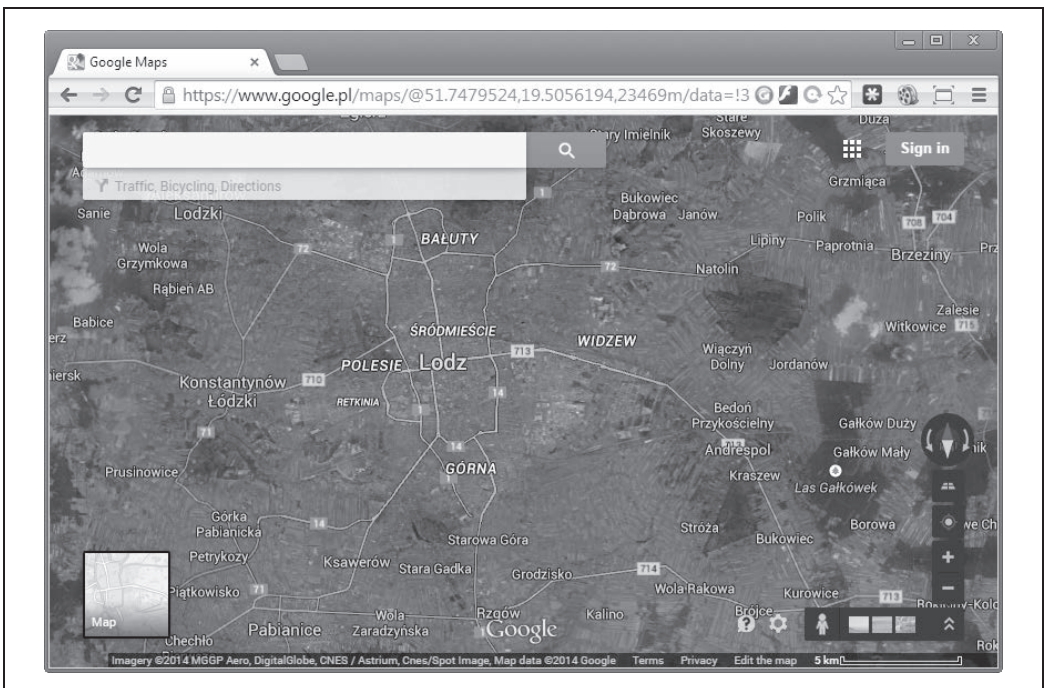
Obsługa obrazków	584
Metoda drawImage	584
Skalowanie obrazu	584
Wybieranie fragmentu obrazu	585
Kopiowanie z elementu canvas	586
Tworzenie cieni	586
Przetwarzanie obrazu na poziomie pikseli	587
Metoda getImageData	588
Metoda putImageData	591
Metoda createImageData	591
Zaawansowane efekty graficzne	591
Własność globalCompositeOperation	591
Własność globalAlpha	593
Przekształcenia	594
Metoda scale	594
Metody save i restore	595
Metoda rotate	596
Metoda translate	596
Metoda transform	598
Metoda setTransform	600
Pytania	600
25. Filmy i dźwięk w HTML5	601
O kodekach	602
Element <audio>	603
Wsparcie dla przeglądarek nieobsługujących HTML5	605
Element <video>	607
Kodeki wideo	607
Obsługa starszych przeglądarek	611
Pytania	612
26. Inne funkcje HTML5	613
Geolokacja i usługi GPS	613
Inne sposoby lokalizacji	614
Geolokacja i HTML5	615
Magazyn lokalny	617
Zastosowanie magazynu lokalnego	618
Obiekt localStorage	618
Web workers	620
Technologia przeciągnij i upuść	622

Komunikacja między dokumentami	624
Inne znaczniki HTML5	628
Pytania	628
27. Zastosowanie wszystkich omówionych technologii	629
Projektowanie aplikacji — serwisu społecznościowego	630
Strona WWW z przykładami	630
functions.php	630
Funkcje	631
header.php	633
setup.php	635
index.php	636
signup.php	637
Sprawdzanie dostępności nazwy użytkownika	639
Logowanie	639
checkuser.php	640
login.php	641
profile.php	642
Dodawanie tekstu „O mnie”	644
Dodawanie zdjęcia profilowego	644
Przetwarzanie obrazu	644
Wyświetlanie bieżącego profilu	645
members.php	647
Wyświetlanie profilu użytkownika	649
Dodawanie i usuwanie znajomych	649
Wyświetlanie listy wszystkich użytkowników	649
friends.php	650
messages.php	653
logout.php	656
styles.css	657
javascript.js	660
A Odpowiedzi na pytania kontrolne	661
B Zasoby internetowe	681
C Słowa z grupy stopwords w MySQL	685
D Funkcje MySQL	689
E Selektory, obiekty i metody jQuery	699
Skorowidz	721

Zastosowanie komunikacji asynchronicznej

Pojęcie *Ajax* po raz pierwszy zostało użyte w 2005 roku. Pochodzi ono od określenia *Asynchronous JavaScript and XML*, a bardziej przystępnie oznacza pewien zestaw wbudowanych metod języka JavaScript, służących do przesyłania danych „w tle” między przeglądarką a serwerem. Termin ten został jednak w dużej mierze zarzucony na rzecz bardziej potocznego określenia *komunikacja asynchroniczna*.

Doskonałym przykładem zastosowania tej technologii są Mapy Google (rysunek 17.1) — przeglądarka pobiera z serwera nowe części mapy dopiero wtedy, gdy są potrzebne, bez konieczności przeładowania strony.



Rysunek 17.1. Mapy Google to doskonały przykład praktycznego zastosowania komunikacji asynchronicznej

Zastosowanie komunikacji asynchronicznej nie tylko zdecydowanie zmniejsza ilość danych przesyłanych w obu kierunkach, ale sprawia, że strony internetowe obsługują się bardzo interaktywnie i płynnie, co upodabnia je do samowystarczalnych aplikacji. Dzięki temu uzyskuje się wygodę i responsywność interfejsu.

Czym jest komunikacja asynchroniczna?

Początki komunikacji asynchronicznej w jej dzisiejszym wydaniu można datować od premiery Internet Explorera 5, która miała miejsce w 1999 roku. Pojawił się w nim nowy obiekt XMLHttpRequest. XMLHttpRequest to technologia opracowana przez firmę Microsoft, umożliwiająca tworzenie komponentów i kontrolki rozszerzających możliwości przeglądarki. Inni producenci przeglądarek podążyli tym tropem, lecz zamiast korzystać z XMLHttpRequest, zaimplementowali podobne rozwiązania w postaci integralnej części interpretera JavaScriptu.

W pewnej uproszczonej formie mechanizmy tego rodzaju istniały jednak już wcześniej: opierały się one na użyciu na stronie ukrytych ramek, komunikujących się „w tle” z serwerem. Jednym z pierwszych zastosowań komunikacji asynchronicznej były czaty internetowe, w których technologia ta służyła do pobierania i wyświetlania nowych wiadomości bez konieczności przeładowywania strony.

Przyjrzyjmy się, jak korzystać z komunikacji asynchronicznej w JavaScriptcie.

Zastosowanie obiektu XMLHttpRequest

Ze względu na różnice w implementacji obiektu XMLHttpRequest w różnych przeglądarkach najpierw należy napisać specjalną funkcję, która będzie gwarantowała poprawne wykonywanie kodu w każdej popularnej przeglądarce.

Aby to zrobić, należy zapoznać się z trzema sposobami tworzenia obiektu XMLHttpRequest:

- IE 5: `request = new ActiveXObject("Microsoft.XMLHTTP")`
- IE 6: `request = new ActiveXObject("Msxml2.XMLHTTP")`
- wszystkie pozostałe: `request = new XMLHttpRequest()`

Różnice w przeglądarce IE wynikają ze zmiany wprowadzonej przez Microsoft w Internet Explorerze 6 względem poprzedniego wydania. Wszystkie pozostałe przeglądarki używają innej metody. Kod podany w przykładzie 17.1 będzie poprawnie działał we wszystkich przeglądarkach wydanych w ciągu kilku ostatnich lat.

Przykład 17.1. Funkcja komunikacji asynchronicznej kompatybilna z wszystkimi przeglądarkami

```
<script>
function asyncRequest()
{
  try //Przeglądarka inna niż IE?
  { //Tak
    var request = new XMLHttpRequest()
  }
  catch(e1)
  {
    try //IE 6+?
```

```

    { //Tak
      request = new XMLHttpRequest("Msxml2.XMLHTTP")
    }
    catch(e2)
    {
      try //IE 5?
      { //Tak
        request = new XMLHttpRequest("Microsoft.XMLHTTP")
      }
      catch(e3) //Brak obsługi komunikacji asynchronicznej
      {
        request = false
      }
    }
  }
  return request
}
</script>

```

Być może pamiętasz z rozdziału 14. wstępne informacje na temat obsługi błędów za pomocą konstrukcji `try ... catch`. Przykład 17.1 doskonale ilustruje użyteczność tej metody: instrukcja `try` podejmuje próbę wykonania instrukcji dla przeglądarki innej niż IE, a jeśli próba zakończy się powodzeniem, program przeskakuje od razu do ostatniej instrukcji `return`, gdzie następuje zwrócenie nowego obiektu.

W przeciwnym razie instrukcja `catch` wyłapuje błąd i podejmowana jest kolejna próba. I ponownie: jeśli próba ta się powiedzie, zwracany jest nowy obiekt; w przeciwnym razie program przystępuje do wykonywania ostatniej spośród trzech grup instrukcji. Jeśli i ta próba zawiedzie, to znaczy, że przeglądarka nie obsługuje komunikacji asynchronicznej, a obiekt `request` otrzymuje wartość `false`; w przeciwnym razie jest zwracany w zwykły sposób. W ten sposób masz do dyspozycji uniwersalną funkcję, którą możesz z powodzeniem dodać do biblioteki przydatnych funkcji JavaScript.

Wiesz już, jak utworzyć obiekt `XMLHttpRequest`, ale co można z nim zrobić? Każdy z takich obiektów jest wyposażony w zestaw właściwości (zmiennych) oraz metod (funkcji), zebranych w tabelach 17.1 i 17.2.

Tabela 17.1. Właściwości obiektu `XMLHttpRequest`

Właściwość	Opis
<code>onreadystatechange</code>	Określa zdarzenie wywoływane za każdym razem, gdy właściwość <code>readyState</code> obiektu ulegnie zmianie.
<code>readyState</code>	Właściwość przyjmująca wartości całkowite, informująca o bieżącym stanie żądania. Może mieć następującą wartość: 0 – niezainicjalizowane, 1 – przesyłanie, 2 – przesłane, 3 – przetwarzanie, 4 – zakończone.
<code>responseText</code>	Dane w postaci tekstowej zwrócone przez serwer.
<code>responseXML</code>	Dane w postaci XML zwrócone przez serwer.
<code>status</code>	Kod statusu HTTP zwrócony przez serwer.
<code>statusText</code>	Tekst statusu HTTP zwrócony przez serwer.

Tabela 17.2. Metody obiektu XMLHttpRequest

Metoda	Opis
abort()	Przerywa bieżące żądanie.
getAllResponseHeaders()	Zwraca wszystkie nagłówki w postaci łańcucha znaków.
getResponseHeader(parametr)	Zwraca wartość param w postaci łańcucha znaków.
open('metoda', 'url', 'asynch')	Określa metodę przesyłania danych HTTP (GET albo POST), docelowy adres URL oraz czy żądanie ma być obsługiwane asynchronicznie (true albo false).
send(dane)	Wysyła dane do docelowego serwera przy użyciu podanej metody HTTP.
setRequestHeader('parametr', 'wartość')	Określa nagłówek w postaci pary parametr/wartość.

Te właściwości i metody dają kontrolę nad rodzajem danych wysyłanych na serwer i odbieranych z niego, a także umożliwiają określenie sposobu ich wysyłania i odbierania. Można na przykład określić, czy dane mają zostać przesłane w postaci czystego tekstu (który może zawierać kod HTML i inne znaczniki), czy w formie XML, a także wybrać metodę POST lub GET wysyłania danych na serwer.

Przyjrzyjmy się najpierw metodzie POST na podstawie dwóch bardzo prostych dokumentów: pierwszy zawiera kod HTML i JavaScript, a drugi — program PHP, który komunikuje się asynchronicznie z pierwszym. Wystarczy kilka linii kodu JavaScript, by pobrać z zewnętrznego serwera dokument, który następnie jest zwracany do przeglądarki za pośrednictwem Twojego serwera i umieszczany w wybranej części bieżącego dokumentu.

Twój pierwszy program asynchroniczny

Wprowadź kod podany w przykładzie 17.2 i zapisz go pod nazwą *urlpost.html*, ale na razie nie otwieraj go w przeglądarce.

Przykład 17.2. Dokument *urlpost.html*

```
<!DOCTYPE html>
<html> <!-- urlpost.html -->
  <head>
    <title>Przykład komunikacji asynchronicznej</title>
  </head>
  <body style='text-align:center'>
    <h1>Wczytywanie strony do elementu DIV</h1>
    <div id='info'>To zdanie zostanie zastąpione</div>

    <script>
      params = "url=news.com"
      request = new XMLHttpRequest()

      request.open("POST", "urlpost.php", true)
      request.setRequestHeader("Content-type",
        "application/x-www-form-urlencoded")
      request.setRequestHeader("Content-length", params.length)
      request.setRequestHeader("Connection", "close")

      request.onreadystatechange = function()
```



```

    {
      if (this.readyState == 4)
      {
        if (this.status == 200)
        {
          if (this.responseText != null)
          {
            document.getElementById('info').innerHTML =
              this.responseText
          }
          else alert("Błąd komunikacji: Nie otrzymano danych")
        }
        else alert( "Błąd komunikacji: " + this.statusText)
      }
    }
  }

  request.send(params)

  function asyncRequest()
  {
    try
    {
      var request = new XMLHttpRequest()
    }
    catch(e1)
    {
      try
      {
        request = new ActiveXObject("Msxml2.XMLHTTP")
      }
      catch(e2)
      {
        try
        {
          request = new ActiveXObject("Microsoft.XMLHTTP")
        }
        catch(e3)
        {
          request = false
        }
      }
    }
  }
  return request
}
</script>
</body>
</html>

```

Przyjrzyjmy się temu przykładowi i jego działaniu. Kilka pierwszych linii odpowiada za zainicjowanie dokumentu HTML i wyświetlenie nagłówka. W następnej linii jest tworzony element `<div>` o identyfikatorze `info`, który zawiera napis `To zdanie zostanie zastąpione`. Później pojawi się tutaj tekst zwrócony przez wywołanie asynchroniczne.

Następnych sześć linii jest niezbędne do zrealizowania żądania HTTP POST. Najpierw zmiennej `params` zostaje przypisana para w postaci `parametr=wartość`, która zostanie przesłana na serwer. Następnie jest tworzony nowy obiekt żądania. Potem za pomocą metody `open` zostaje określony sposób przesłania żądania (POST), adres (`urlpost.php`) oraz tryb (asynchroniczny). Trzy ostatnie linie w tej sekcji definiują nagłówki, które poinformują docelowy serwer o żądaniu POST.

Właściwość readyState

Teraz możemy zająć się detalami obsługi wywołań asynchronicznych, która opiera się na właściwości readyState. Wywołania te umożliwiają przeglądarce ciągłą interakcję z użytkownikiem i zmianę wyświetlanych danych, podczas gdy program za pośrednictwem właściwości onreadystatechange odwołuje się do funkcji wykonywanej przy każdej zmianie właściwości readyState. W tym przypadku została użyta funkcja anonimowa, umieszczona bezpośrednio w wierszu z żądaniem — można jednak w tym celu użyć osobnej funkcji o zdefiniowanej nazwie. Tego rodzaju funkcje noszą niekiedy nazwę *wywołań zwrotnych* (ang. *callback*), gdyż program zwraca się do nich za każdym razem, gdy zmieni się właściwość readyState.

Składnia wywołania zwrotnego przy założeniu użycia funkcji anonimowej w jednym wierszu wygląda następująco:

```
request.onreadystatechange = function()
{
  if (this.readyState == 4)
  {
    // tu coś się dzieje
  }
}
```

Gdybyś chciał użyć osobnej funkcji o zdefiniowanej nazwie, konstrukcja będzie wyglądała tylko trochę inaczej:

```
request.onreadystatechange = asyncCallback
function asyncCallback()
{
  if (this.readyState == 4)
  {
    // tu coś się dzieje
  }
}
```

Na podstawie tabeli 17.1 wiesz, że właściwość readyState może przyjmować pięć wartości. Ale nas interesuje tylko jedna: wartość 4, która oznacza zakończenie żądania. Jak widać, wywołanie naszej nowej funkcji nie będzie miało żadnych efektów, dopóki właściwość readyState nie przyjmie wartości 4. Gdy wartość ta zostanie wykryta, sprawdzany jest status żądania, aby się upewnić, że ma on wartość 200 — taka wartość oznacza bowiem, że żądanie zostało zakończone powodzeniem. Jeśli wartość jest inna niż 200, na ekranie pojawia się komunikat błędu zawarty we właściwości textStatus.



Zauważ, że zamiast nazwy używanego obiektu, czyli request (np. request.readyState lub request.status), w odwołaniach do właściwości obiektów pojawia się słowo this (np. this.readyState, this.status itd.). Chodzi o to, by móc łatwo przenieść kod do innego programu, który w tej postaci będzie działał z dowolnie nazwanym obiektem: słowo kluczowe this zawsze odnosi się bowiem do bieżącego obiektu.

Po upewnieniu się, że właściwość readyState ma wartość 4, a status wynosi 200, należy sprawdzić właściwość responseText, aby się przekonać, czy zawiera ona jakąś wartość. Jeśli nie, na ekranie pojawi się okno z ostrzeżeniem. W przeciwnym razie treść elementu <div> zostanie zastąpiona wartością właściwości responseText:

```
document.getElementById('info').innerHTML = this.responseText
```

W tej linii kodu metoda `getElementById` odwołuje się do elementu `info`, a wartość zwrócona przez to odwołanie jest przypisywana do właściwości `innerHTML` tego elementu. Efekt jest taki, że zmienia się tylko ten element strony internetowej, a wszystko inne pozostaje bez zmian.

Po wszystkich tych przygotowaniach żądanie asynchroniczne jest wreszcie wysyłane na serwer przy użyciu następującej instrukcji, która przekazuje parametry zdefiniowane uprzednio w zmiennej `params`:

```
request.send(params)
```

Omówiona procedura jest uaktywniana za każdym razem, gdy wartość właściwości `readyState` ulega zmianie.

Pozostała część kodu to funkcja `asyncRequest` z przykładu 17.1, a także znaczniki kończące skrypt oraz dokument HTML.

Proces komunikacji asynchronicznej po stronie serwera

Przejdźmy teraz do drugiej części procedury, realizowanej za pomocą PHP — jej kod został podany w przykładzie 17.3. Przepisz go i zapisz pod nazwą `urlpost.php`.

Przykład 17.3. Dokument `urlpost.php`

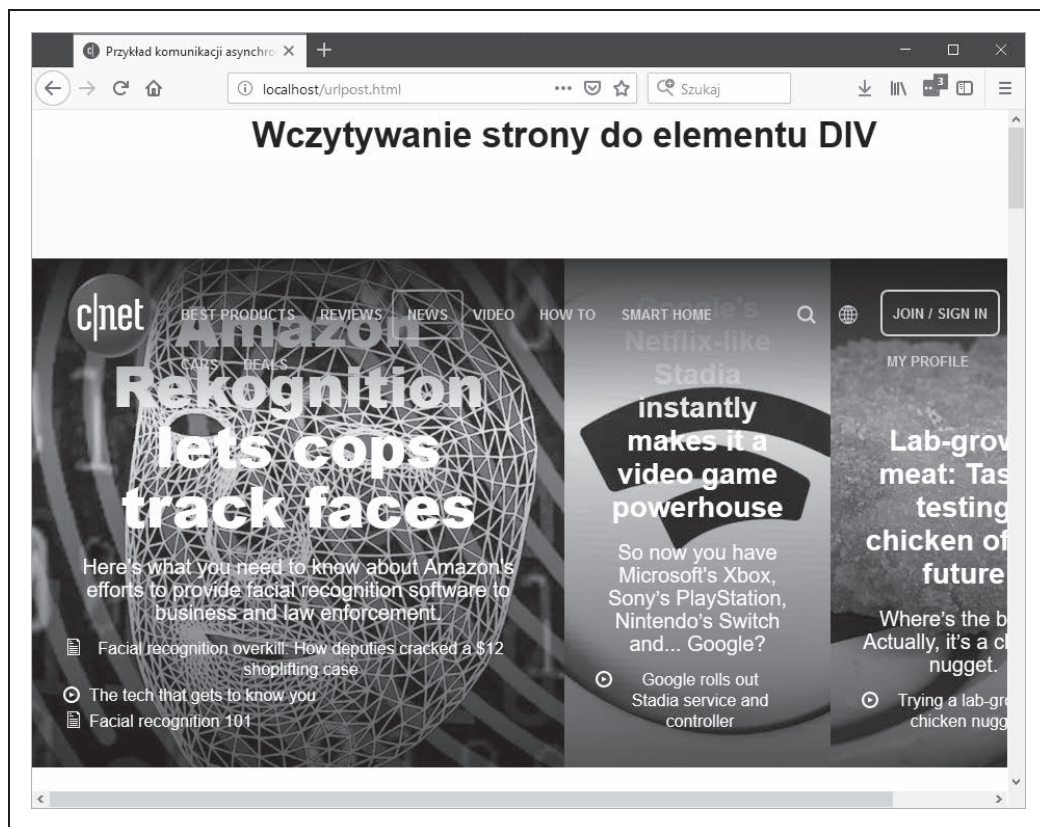
```
<?php // urlpost.php
if (isset($_POST['url']))
{
    echo file_get_contents('http://' . SanitizeString($_POST['url']));
}

function SanitizeString($var)
{
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
}
?>
```

Jak widać, kod jest krótki, prosty i — tak jak to powinno być w przypadku wszystkich przesyłanych danych — robi dobry użytek z bardzo ważnej funkcji `SanitizeString`. W tym przypadku pominięcie procedury oczyszczania danych mogłoby doprowadzić do wstawienia przez użytkownika kodu JavaScript i przejęcia kontroli nad Twoim programem.

Program korzysta z funkcji PHP o nazwie `file_get_contents` do pobrania strony internetowej znajdującej się pod adresem URL przekazanym za pośrednictwem zmiennej `$_POST['url']`. Funkcja `file_get_contents` jest o tyle uniwersalna, że umożliwia załadowanie całej zawartości pliku albo strony internetowej z lokalnego lub zdalnego serwera; uwzględni ona nawet przekierowania (na przykład w przypadku przeniesionych stron).

Po zapisaniu pliku z podanym kodem możesz otworzyć dokument `urlpost.html` w przeglądarce i po kilku chwilach powinieneś zobaczyć stronę internetową serwisu `news.com` wczytaną do elementu `<div>`, utworzonego specjalnie w tym celu. Cały proces będzie trwał trochę dłużej od bezpośredniego otwarcia strony WWW, gdyż dokument zostanie przesłany dwukrotnie: raz na serwer i po raz kolejny z serwera do przeglądarki. Efekt powinien wyglądać podobnie jak na rysunku 17.2.



Rysunek 17.2. Strona główna serwisu news.com wczytana do elementu <div>

W ten sposób udało się nam nie tylko wykonać żądanie asynchroniczne i zwrócić odpowiedź do skryptu JavaScript, ale przy okazji wykorzystać możliwości PHP do połączenia zupełnie niezależnych stron WWW. Tak się bowiem składa, że jeśli spróbowalibyśmy bezpośrednio pobrać tę stronę internetową za pomocą żądania asynchronicznego (z pominięciem skryptu PHP na serwerze), nie udało by się nam to ze względu na istnienie zabezpieczeń chroniących przed wykonywaniem takich żądań między różnymi domenami. Ten prosty przykład stanowi więc zarazem rozwiązanie praktycznego problemu.

Zastosowanie metody GET zamiast POST

Tak jak w przypadku przesyłania dowolnych danych za pośrednictwem formularza, istnieje możliwość przekazania danych w trybie GET, co zarazem pozwala zmniejszyć objętość kodu o kilka linii. Ma to jednak pewną wadę: niektóre przeglądarki mogą przechowywać żądania GET w pamięci cache, co w przypadku żądań POST nigdy nie ma miejsca. Żądania nie powinny być jednak przechowywane w pamięci podręcznej, bo przeglądarka po prostu ponownie wyświetli to, co otrzymała poprzednio, zamiast zwrócić się do serwera o nowe dane. Pewnym obejściem tego problemu jest dodanie do każdego żądania losowego parametru, dzięki czemu docelowy adres URL jest za każdym razem inny.

Przykład 17.4 pokazuje, w jaki sposób uzyskać efekt analogiczny jak w przypadku przykładu 17.2, ale za pomocą żądań typu GET zamiast POST.

Przykład 17.4. Dokument urlget.html

```
<!DOCTYPE html>
<html> <!-- urlget.html -->
  <head>
    <title>Przykład komunikacji asynchronicznej</title>
  </head>
  <body style='text-align:center'>
    <h1>Wczytywanie strony do elementu DIV</h1>
    <div id='info'>To zdanie zostanie zastąpione</div>

    <script>
      nocache = "&nocache=" + Math.random() * 1000000
      request = new asyncRequest()
      request.open("GET", "urlget.php?url=news.com" + nocache, true)

      request.onreadystatechange = function()
      {
        if (this.readyState == 4)
          {
            if (this.status == 200)
              {
                if (this.responseText != null)
                  {
                    document.getElementById('info').innerHTML =
                      this.responseText
                  }
                else alert("Błąd komunikacji: Nie otrzymano danych")
              }
            else alert( "Błąd komunikacji: " + this.statusText)
          }
      }

      request.send(null)

      function asyncRequest()
      {
        try
          {
            var request = new XMLHttpRequest()
          }
        catch(e1)
          {
            try
              {
                request = new ActiveXObject("Msxml2.XMLHTTP")
              }
            catch(e2)
              {
                try
                  {
                    request = new ActiveXObject("Microsoft.XMLHTTP")
                  }
                catch(e3)
                  {

```

```

        request = false
    }
}
return request
}
</script>
</body>
</html>

```

Najważniejsza różnica między dwoma dokumentami została wyróżniona pogrubieniem. Można ją opisać następująco:

- W przypadku żądania typu GET nie trzeba przysyłać nagłóweków.
- W żądaniach typu GET używamy metody `open`. Przekazujemy jej URL zawierający symbol `?`, po którym następuje para parametr-wartość w postaci (tutaj) `url=news.com`.
- Drugą parę parametr/wartość dołączamy za pomocą symbolu `&`. Parametr nosi nazwę `nocache`, a jego wartość jest losowana z zakresu od 0 do 1000000. Dzięki temu każdy URL będzie inny, a kolejne żądania nie będą pobierane z pamięci podręcznej przeglądarki.
- W odwołaniu do metody `send` przekazujemy teraz tylko argument `null`, gdyż w przypadku metody GET nie są przekazywane żadne parametry. Warto pamiętać, że całkowite zrezygnowanie z przekazywania argumentów nie wchodzi w grę, gdyż wywołałoby błąd.

Aby nowy dokument był kompletny, trzeba jeszcze zmodyfikować program w PHP tak, by obsługiwał żądanie GET. Przykład 17.5 należy zapisać w pliku o nazwie `urlget.php`.

Przykład 17.5. Dokument `urlget.php`

```

<?php // urlget.php
if (isset($_GET['url']))
{
    echo file_get_contents("http://".sanitizeString($_GET['url']));
}

function sanitizeString($var)
{
    $var = strip_tags($var);
    $var = htmlentities($var);
    return stripslashes($var);
}
?>

```

Cała różnica pomiędzy powyższym kodem a przykładem 17.3 sprowadza się do tego, że odwołania do tablicy `$_POST` zostały zastąpione odwołaniami do tablicy `$_GET`. Efekt otwarcia dokumentu `urlget.html` w przeglądarce jest taki sam jak `urlpost.html`.

Przesyłanie żądań XML

Choć obiekty, które tworzyliśmy do tej pory, należą do klasy o nazwie `XMLHttpRequest`, jak dotąd w ogóle nie posługiwaliśmy się XML. Przekonałeś się już, że za pośrednictwem żądań asynchronicznych bez trudu da się pobrać cały dokument HTML, ale równie dobrze moglibyśmy zażądać strony czysto tekstowej, łańcucha znaków, liczby, a nawet danych z arkusza kalkulacyjnego.

Zmodyfikujmy zatem poprzedni przykładowy dokument HTML i program PHP w taki sposób, by pobrać dane XML. Aby to zrobić, najpierw przyjrzyjmy się programowi *xmlget.php* pokazanemu w przykładzie 17.6.

Przykład 17.6. Dokument *xmlget.php*

```
<?php //xmlget.php
  if (isset($_GET['url']))
  {
    header('Content-Type: text/xml');
    echo file_get_contents("http://".sanitizeString($_GET['url']));
  }

function sanitizeString($var)
{
  $var = strip_tags($var);
  $var = htmlentities($var);
  return stripslashes($var);
}
?>
```

Jak widać, program został bardzo nieznacznie zmodyfikowany (zmiana została wyróżniona pogrubieniem) w celu przesłania poprawnego nagłówka XML przed odwołaniem się do docelowego dokumentu. W programie nie zastosowano żadnych mechanizmów sprawdzających — przyjąłem założenie, że żądanie będzie rzeczywiście odwoływało się do dokumentu XML.

Przejdźmy teraz do kodu HTML w pliku *xmlget.html* w przykładzie 17.7.

Przykład 17.7. Dokument *xmlget.html*

```
<!DOCTYPE html>
<html> <!-- xmlget.html -->
<head>
  <meta charset="utf-8">
  <title>Przykład komunikacji asynchronicznej</title>
</head>
<body>
  <h1>Wczytywanie danych XML do elementu DIV</h1>
  <div id='info'>To zdanie zostanie zastąpione</div>

  <script>
    nocache = "&nocache=" + Math.random() * 1000000
    url      = "rss.news.yahoo.com/rss/topstories"
    out      = "";

    request = new XMLHttpRequest()
    request.open("GET", "xmlget.php?url=" + url + nocache, true)

    request.onreadystatechange = function()
    {
      if (this.readyState == 4)
      {
        if (this.status == 200)
        {
          if (this.responseText != null)
          {
            titles = this.responseXML.getElementsByTagName('title')
          }
        }
      }
    }
  </script>
</body>
</html>
```

```

        for (j = 0 ; j < titles.length ; ++j)
        {
            out += titles[j].childNodes[0].nodeValue + '<br>'
        }
        document.getElementById('info').innerHTML = out
    }
    else alert("Błąd komunikacji: Nie otrzymano danych")
}
else alert( "Błąd komunikacji: " + this.statusText)
}
}

request.send(null)

function asyncRequest()
{
    try
    {
        var request = new XMLHttpRequest()
    }
    catch(e1)
    {
        try
        {
            request = new ActiveXObject("Msxml2.XMLHTTP")
        }
        catch(e2)
        {
            try
            {
                request = new ActiveXObject("Microsoft.XMLHTTP")
            }
            catch(e3)
            {
                request = false
            }
        }
    }
}
return request
}
</script>
</body>
</html>

```

Po raz kolejny zmiany zostały wyróżnione pogrubieniem, co pozwala łatwo zauważyć, że kod jest bardzo podobny do poprzedniej wersji — z tym że docelowy URL *rss.news.yahoo.com/rss/topstories* odwołuje się do dokumentu XML, a konkretnie do kanału RSS z najpopularniejszymi wiadomościami z serwisu Yahoo!.

Inna istotna różnica polega na zastosowaniu właściwości `responseXML`, która zastąpiła właściwość `responseText`. Gdy serwer zwraca dane XML, zostają one zapisane właśnie w `responseXML`.

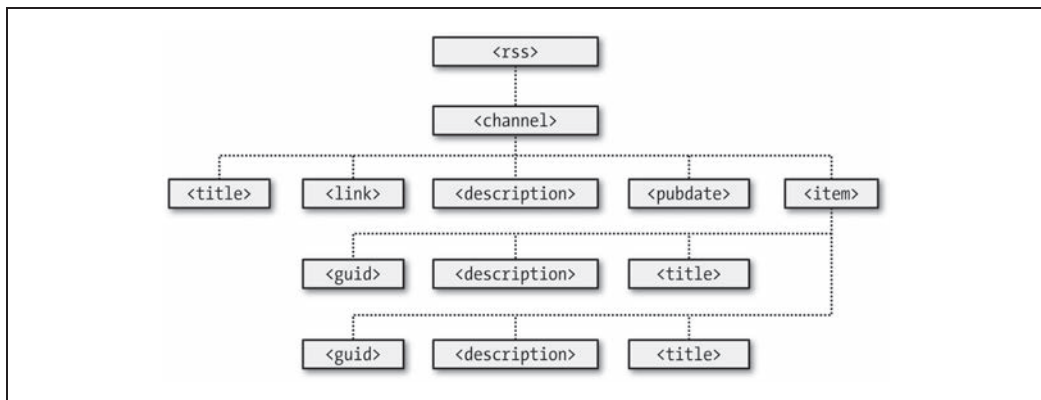
Właściwość `responseXML` nie zawiera jednak zwykłego łańcucha tekstowego XML: tak naprawdę jest to obiekt z dokumentem XML, który można analizować i przetwarzać przy użyciu metod i właściwości typowych dla drzewa DOM. To oznacza, że do takiego obiektu można się odwołać na przykład przy użyciu metody JavaScript `getElementsByTagName`.

Kilka słów o XML

Dokumenty XML, o jakich tutaj mowa, na ogół mają postać danych RSS, jak w przykładzie 17.8. Piękno XML polega jednak na tym, że strukturę tego rodzaju można zapisać w ramach drzewa DOM (rysunek 17.3), co umożliwia jej szybkie i wygodne przeszukiwanie.

Przykład 17.8. Przykładowy dokument XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
  <channel>
    <title>RSS Feed</title>
    <link>http://stronainternetowa.com</link>
    <description>Kanał RSS serwisu stronainternetowa.com</description>
    <pubDate>Mon, 11 May 2020 00:00:00 GMT</pubDate>
    <item>
      <title>Nagłówek</title>
      <guid>http://stronainternetowa.com/naglowek</guid>
      <description>To jest nagłówek</description>
    </item>
    <item>
      <title>Nagłówek 2</title>
      <guid>http://website.com/naglowek2</guid>
      <description>Drugi nagłówek</description>
    </item>
  </channel>
</rss>
```



Rysunek 17.3. Drzewo DOM z przykładu 17.8

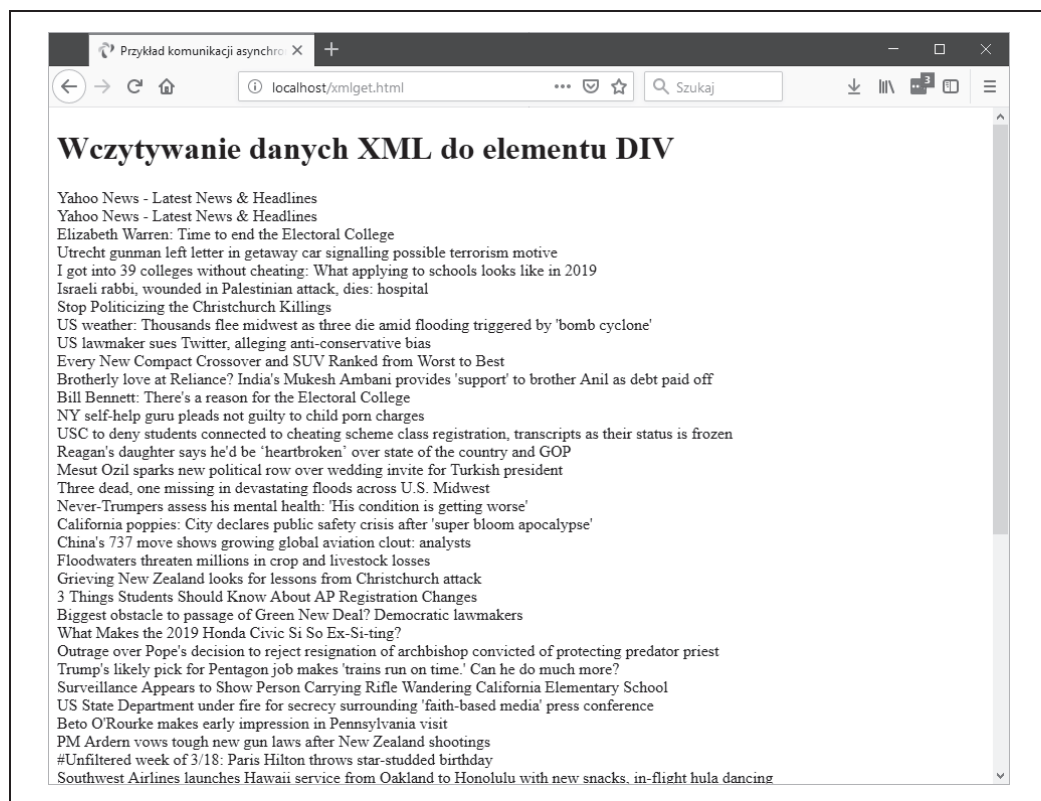
Za pomocą metody `getElementsByTagName` można szybko wyszukać wartości powiązane z różnymi znacznikami, bez czasochłonnego przeszukiwania całego tekstu. Właśnie taki jest cel użycia poniższej instrukcji w kodzie przykładu 17.7:

```
titles = this.responseXML.getElementsByTagName('title')
```

Ta jedna instrukcja wystarczy, by wszystkie wartości elementów `<title>` — czyli tytułów wiadomości — zostały umieszczone w tablicy `titles`. Stąd zaś już tylko krok, by wyodrębnić je przy użyciu następującej składni (gdzie zmienna `j` zawiera wartość całkowitą odpowiadającą pozycji tytułu, do którego się odwołujemy):

```
titles[j].childNodes[0].nodeValue
```

Wszystkie tytuły są następnie dołączane do zmiennej tekstowej `out`, a po ich przetworzeniu rezultat jest umieszczany w elemencie `<div>` na początku dokumentu. Po otwarciu pliku `xmlget.html` w przeglądarce powinieneś otrzymać efekt podobny jak na rysunku 17.4.



Rysunek 17.4. Asynchroniczne pobieranie nagłówków wiadomości w postaci XML z serwisu Yahoo!



Podsumowując, każdy składnik taki jak `title` jest tzw. węzłem (ang. *node*), to zaś oznacza, że tekst tytułu jest uważany za węzeł. Odwołując się do węzła potomnego, trzeba zażądać go w postaci tekstowej — i temu służy `.nodeValue`. Ponadto przy żądaniu danych XML — tak jak w przypadku formularzy — można użyć metody `POST` albo `GET`; wybór nie ma większego wpływu na rezultat.

Po co używać XML?

Zastanawiasz się być może, po co używać XML w celach innych niż pobieranie dokumentów XML takich jak kanały informacyjne RSS. No cóż, rzeczywiście nie ma takiej konieczności, ale gdyby zależało Ci na tym, by dane przesyłane do aplikacji Ajax miały konkretną strukturę, to jest to znacznie wygodniejsze niż przetwarzanie zwykłego tekstu przy użyciu procedur w JavaScriptcie.

W takich przypadkach lepiej utworzyć dokument XML i przekazać go z powrotem do funkcji wywołującej, która automatycznie umieści go w drzewie DOM, równie łatwym do przetwarzania jak drzewo DOM dokumentu HTML, który poznałeś już wcześniej.

Obecnie programiści częściej stosują do wymiany danych JavaScript Object Notation (<http://json.org>), czyli JSON, prosty podzbiór JavaScriptu.

Zastosowanie bibliotek komunikacji asynchronicznej

Wiesz już, w jaki sposób od podstaw programować procedury komunikacji asynchronicznej, zachęcam Cię więc do zapoznania się z darmowymi bibliotekami (ang. *frameworks*), które zdecydowanie ułatwiają pracę z tą technologią i oferują wiele zaawansowanych funkcji. Szczególnie chciałbym Ci polecić przyjrzenie się bibliotece jQuery, która jest chyba najpopularniejszym rozwiązaniem tego typu — będziesz zresztą mógł o niej przeczytać w rozdziale 21. Tymczasem jednak w kolejnym rozdziale przyjrzymy się modyfikowaniu wyglądu stron WWW przy użyciu technologii CSS.

Pytania

1. Dlaczego należy napisać specjalną funkcję do tworzenia nowych obiektów XMLHttpRequest?
 2. Do czego służy konstrukcja try ... catch?
 3. Ile właściwości i metod ma obiekt XMLHttpRequest?
 4. Jak sprawdzić, czy przetwarzanie żądania asynchronicznego zostało zakończone?
 5. Jak sprawdzić, czy przetwarzanie żądania asynchronicznego zakończyło się powodzeniem?
 6. Jaka właściwość obiektu XMLHttpRequest zwraca odpowiedź w postaci tekstowej na żądanie asynchroniczne?
 7. Jaka właściwość obiektu XMLHttpRequest zwraca odpowiedź w postaci XML na żądanie asynchroniczne?
 8. Jak odwołać się do funkcji zwrotnej obsługującej odpowiedzi na żądania asynchroniczne?
 9. Jaka metoda obiektu XMLHttpRequest służy do inicjalizowania żądania asynchronicznego?
 10. Na czym polegają główne różnice między żadaniami asynchronicznymi typu GET i POST?
- Odpowiedzi na pytania znajdziesz w dodatku A, w sekcji „Odpowiedzi na pytania z rozdziału 17.”.

\$, symbol, 319

A

AAC, kodek audio, 602

adres

- IP, 28, 551
- przekazywanie, 614
- MAC, 614

Ajax, 381, 535, 624

akcja domyślna, 99, 335

definiowanie, 99

algorytm

- BCRYPT, 295
- kompresji
- MP3, 601
- MPEG, 601
- md5, 292
- SHA-1, 292
- TLS, 618

AMPPS, 42, 111, 175, 176, 182

dla OS X, 49

testowanie instalacji, 46

Android, 549

anulowanie

- polecenia, 178
- transakcji, 228

Apple, 549

asocjacyjność, 88

atak cross-site scripting, XSS, 260

atrybut

- AUTO_INCREMENT, 188, 254
- autocomplete, 280
- autofocus, 281
- data-ajax, 535
- data-rel, 536
- form, 282

formation, 281

height, 282

list, 283

min, 282

multiple, 275

placeholder, 281

required, 281

step, 282

viewport, 533

width, 282

atrybuty nadpisania, 281

automatyczna inkrementacja, 189, 254

autoryzacja HTTP, 288

awatar użytkownika, 156

B

BASIC, 61

baza danych

- MariaDB, 32
- MySQL, 33, 156
- MySQL \t, 29

bazy danych

- anonimowość, 226
- dostęp, 181, 222
- efektywność, 216
- odtworzenie, 233
- optymalizacja, 215
- projektowanie, 213
- relacje, 223
- transakcje, 226
- tworzenie, 180
- wyświetlanie zawartości, 231
- zapytania, 213

BBS, 27

Berners-Lee Tim, 27

- Bézierra krzywe, 578
- bezpieczeństwo, 81, 94, 160, 170
- białe znaki, 122
- biblioteka
 - GD, 37
 - jQuery, *Patrz:* jQuery
 - jQuery Mobile, *Patrz:* jQuery Mobile
- BlackBerry, 549
- blog, 108
- błędy
 - dostępu, 176
 - dzielenia przez zero, 106
 - Parse error, 60, 72
 - przenoszenia plików, 160
 - Undefined index, 138
 - Undefined offset, 138
 - wychwytywanie, 332
- Boole George, 83
- boolowska wartość, 89
- boolowskie wyrażenia, 84
- bumpyCaps, konwencja, 346

C

- C, 32
- canvas, 549, 555
 - kreślenie ścieżek, 572
 - określanie formatu obrazu, 558
 - operacje na pikselach, 587
 - przekształcanie elementów, 594
 - rysowanie linii, 570
 - skalowanie obrazu przed umieszczeniem, 584
 - tworzenie, 550
 - tworzenie cieni, 586
 - umieszczanie napisów, 566
 - umieszczanie obrazów, 584
 - wyodrębnianie obrazów, 584
 - wypełnianie obszarów, 574
 - zaawansowane efekty graficzne, 591
 - zastosowanie krzywych, 578
- CERN, 27
- CGI, 31
- Chrome, 549
- ciasteczka, 617
 - dostęp, 288
 - tworzenie, 287
 - usuwanie, 288
 - użycie w sesji, 303
 - zastosowanie, 285

- COBOL, 33
- CSS, 29, 35, 657
 - animacje, 476, 536
 - dołączenie do dokumentu HTML, 398
 - dziedziczenie kaskadowe, 409
 - importowanie stylów, 398
 - jednostki miar, 414
 - kolory, 420
 - gradienty, 421
 - określenia skrócone, 421
 - komentarze, 401
 - model pudełkowy, 428
 - marginsy, 428
 - odstęp, 431
 - ramki, 430
 - pseudoklasy, 425
 - reguły, 400
 - metody definiowania, 410
 - obliczanie specyficzności, 410
 - skracanie, 427
 - z wieloma deklaracjami, 400
 - rozmieszczanie elementów
 - położenie bezwzględne, 422
 - położenie stałe, 423
 - położenie względne, 423
 - selekcja grupowa, 408
 - selektor
 - atrybutu, 407, 436
 - dziecka, 405
 - identyfikatora, 406, 410
 - klasy, 407, 410
 - potomka, 404
 - typu, 404
 - uniwersalny, 408
 - style
 - arkusze zewnętrzne, 403
 - bezpośrednie, inline, 404
 - domyślne, 402
 - użytkownika, 402
 - wewnętrzne, 403
 - zagnieżdżone, 399
 - wersja CSS3, 435
 - właściwości
 - dostęp z poziomu JavaScript, 464
 - pisma, 416
 - zarządzanie stylami tekstu, 418
 - zastosowanie
 - identyfikatorów, 399
 - klas, 399
 - średników, 400

CSS3, 35
 box-sizing
 właściwość, 437
 cienie, 446
 efekty tekstowe, 450
 fonty internetowe, 452
 Google, 453
 kolory
 definiowanie, 448
 luminacja, 449
 modele, 448 – 450
 nasycenie, 448
 przezroczystość, 449, 450
 operator
 \$, 437
 *, 437
 ^, 437
 przejścia, 456, 499
 skrótowa składnia, 458
 przekształcenia, 454
 3D, 455
 ramki, 442
 kolor, 442
 zaokrąglanie rogów, 443
 tło, 438
 położenie obrazka, 439
 umieszczenie kilku obrazów, 441
 widoczność, 438
 wielkość, 440
 właściwość auto, 440
 tworzenie animacji, 456
 układ wielokolumnowy, 447

CSV
 format, 230
 zapisywanie danych, 233

D

dane
 archiwizacja, 230
 konwersja, 151, 153
 komponenty modyfikatora, 151
 przywracanie, 230
 walidacja, 167

data
 format, 154
 sprawdzanie poprawności, 156
 wyświetlanie, 154

debugowanie, 74, 146, 331

definiowanie
 akcji domyślnej, 99
 funkcji, 113, 124
 klas, 124
 typu zmiennej, 315

deklarowanie
 funkcji, 76, 342
 w JavaScript, 316
 klas, 121
 klasy, 345, 348
 User, 345
 metod, 126
 stałych, 127
 typu zmiennych, 73
 właściwości, 126
 jawne, 127
 niejawne, 126
 zmiennych, 61, 80
 o różnym zasięgu, 317

destruktory, 125

dodawanie kolumny, 191
 z automatyczną inkrementacją, 188

DOM, Document Object Model, 393, 397, 488
 dostęp z poziomu JavaScript, 461
 elementy
 dodawanie, 470
 usuwanie, 471
 manipulowanie, 507
 nawigowanie, 514
 struktura, 305
 zastosowanie, 320

dopasowywanie
 metaznaki, 365
 tryb
 niezachłanny, 369
 zachłanny, 369

dyrektywa
 @font-face, 452
 @import, 398
 NATURAL JOIN, 208

dziedziczenie, 130

dzienniki zdarzeń, *Patrz:* logi

E

Editra, 53
edytor kodu, 53
encje HTML, 81

F

Flash, 29, 549, 605, 612

format

GIF, 185

formularze

etykiety, 275

HTML 5, 280

atrybuty nadpisania, 281

autocomplete, 280

autofocus, 281

color, 283

form, 282

height, 282

list, 283

max, 282

min, 282

number, 283

placeholder, 281

range, 283

required, 281

selektory daty i czasu, 283

step, 282

width, 282

listy z opcjami, 274

odczytywanie przesłanych danych, 267

podatność na ataki, 359

pola opcji, 270

pola tekstowe, 269

wielowierszowe, 269

pola ukryte, 273

przełączniki, 272

przycisk wysyłania, 276

tworzenie, 265

wartości domyślne, 268

weryfikacja

powtórne wyświetlenie, 374

wyświetlanie, 247

FreeBSD, 177

FTP, 52

funkcja

alert, 321

checkdate, 156

compact, 146

console.log, 321

copy, 159

count, 143

date, 76, 154

die, 157, 239

document.write, 321

each, 139, 147

end, 147

exec, 170

explode, 144, 146

extract, 145

fclose, 157

fgets, 157, 158

file_exists, 160

file_get_contents, 163

flock, 162

fopen, 157

obsługiwane tryby, 158

fread, 157, 159

fseek, 161

function_exists, 119

fwrite, 105, 157, 162

get_magic_quotes_gpc, 258

getElementById, 461

getnext, 92

getnext(), 92

hsl, 448

hsla, 449

htmlentities, 81, 261

htmlspecialchars, 169, 247, 290

indexOf, 354

ini_set, 303

intval, 107

is_array, 143

isset, 317

jQuery, 483

list, 139

zastosowanie, 140

longdate, 77

mktime, 153

parametry, 153

mysqli

wariant proceduralny, 262

output, 354

password_hash, 292

password_verify, 293

phpinfo, 112

phpversion, 119

preg_match, 373

preg_match_all, 373

preg_replace, 373

print, 113

print_r, 136, 146

- printf, 149
 - argumenty, 149
 - modyfikatory, 150
- rename, 160
- reset, 147
- rgb, 449
- rgba, 450
- session_destroy, 299
- session_start, 297
- setcookie, 287
- shuffle, 144
- some, 353
- sort, 143, 144
- sprintf, 153
- str_repeat, 113
- strrev, 113
- strtolower, 114
- strtoupper, 113
- toDataURL, 557
- ucfirst, 114
- validate, 361
- validateAge, 364
- validateEmail, 364
- validateForename, 363
- validatePassword, 364
- validateSurname, 363
- validateUsername, 363
- funkcje, 76, 111, 143
 - argumenty, 112
 - daty i czasu, 153
 - definiowanie, 113, 124, 341
 - deklaracja, 76
 - jednokierunkowe, 291
 - konstruktory, 345
 - MySQL, 209
 - zwracanie wartości, 113
- funkcji
- file_get_contents, 387
- funkcji funkcja__construct, 125

G

- geolokacja, 551, 613, 615
- georeplikacja, 32
- Google, 38
 - fonty, 453
 - mapy, 381, 551
 - Street View, 614
- GPS, 551, 613

- grupowanie
 - przy użyciu nawiasów, 367
- grupowanie zapytań
 - według kategorii, 206

H

- H.264, kodek wideo, 608
- hasła
 - przechowywanie, 291
 - password_hash, 292
 - password_verify, 293
 - solenie, 292
- hermetyzacja, 120
- hierarchia obiektów, 318
- historia przeglądania
 - zabezpieczenie, 320
- HTML, 27
 - importowanie stylów CSS, 398
 - dopasowywanie znaczników, 368
- HTML5, 36, 170
 - audio, 552
 - canvas, *Patrz:* canvas
 - deklarowanie, 170
 - DOCTYPE, 397, 556
 - formularze, *Patrz:* formularze HTML5
 - geolokacja, 551, 613, 615
 - lokalny magazyn danych, 554
 - magazyn lokalny
 - dostęp, 618
 - obsługiwane kodeki, 602
 - odtwarzanie
 - filmów, 607
 - pliku muzycznego, 604
 - technologia przeciągnij i upuść, 622
 - video, 552
 - wątki robocze, web workers, 554, 620
 - zgodność, 170

I

- IDE, 54, 57, 62
- identyfikatory alfanumeryczne, 135, 142
- indeksy, 193
 - dodawanie, 193
 - przy tworzeniu tabel, 194
 - rodzaje, 193
 - tworzenie, 193
 - typu FULLTEXT, 182, 202
 - dodawanie, 197

- instancja klasy, 345
- instrukcja
 - ALTER, 190
 - BEGIN, 227
 - break, 98, 105, 337
 - case, 98, 334
 - COMMIT, 227
 - continue, 106, 338
 - CREATE INDEX, 194
 - define, 74
 - DESCRIBE, 183, 251
 - document.write, 306
 - echo, 71, 75, 102, 137, 141
 - else, 95, 332
 - elseif, 96
 - endswitch, 99
 - EXPLAIN, 228
 - GRANT, 181
 - parametry, 181
 - if, 90, 93, 327, 332
 - if ... OR, 92
 - include, 118, 239
 - include_once, 118
 - mysqldump, 230
 - print, 75
 - przerwanie działania, 98
 - require, 119
 - require_once, 119
 - ROLLBACK, 228
 - SELECT, 204
 - switch, 97, 98, 333
 - alternatywna składnia, 99
 - with, 329
- instrukcje, 85
 - SQL, 232
 - wielowierszowe, 71
- Internet Explorer, 382
- iOS, 549

J

- Java, 31, 549
- JavaScript, 31, 34, 305, 660
 - błąd składni, 310
 - dekrementacja, 312
 - zmiennych, 314
 - document.write, 320
 - dodawanie, odejmowanie, mnożenie, dzielenie, 312
 - dołączenie plików, 308
 - elementy DOM
 - dodawanie, 470
 - usuwanie, 471
 - formularz z weryfikacją danych, 360
 - funkcje, 316, 341
 - dołączanie do strony, 463
 - osobny plik, 364
 - służące do zmieniania typów, 339
 - hierarchia obiektów, 318
 - inkrementacja, 312, 314
 - klasy znaków, 367
 - literały, 324
 - typy, 324
 - łańcuchy znaków
 - cudzysłowy, 311
 - konkatenacja, 314
 - łączenie zdarzeń i obiektów, 468
 - modulo, 312
 - obiektowy model dokumentu, 317
 - obiekty, 345
 - odczytywanie adresu URL łączy, 318
 - operatory, 312, 325
 - arytmetyczne, 312
 - dwuargumentowe, 325
 - jednoargumentowe, 325
 - logiczne, 314
 - porównania, 313
 - przypisania, 313
 - trzyargumentowy, 325
 - typy, 325
 - pętle, 335
 - przeglądarki, starsze wersje, 307
 - przerwanie
 - setInterval, 474
 - setTimeout, 473
 - składnia komentarzy, 309
 - sprawdzenie wartości wyrażenia, 324
 - średniki, 310
 - tablice, 311, 350
 - wielowymiarowe, 311
 - tekst w HTML, 305
 - typowanie zmiennych, 315
 - umieszczanie tekstu
 - w elementach HTML, 321
 - w kodzie HTML, 467
 - weryfikowanie danych, 359

K

- wyrażenia, 323
 - if, 313
 - regularne, 373
- wyświetlanie błędów, 309
- zastosowanie komentarzy, 309
- zdarzenia, 330, 469
- zmienne, 310, 324
 - globalne, 316
 - lokalne, 316
 - numeryczne, 311
 - zasady nazewnictwa, 310
 - znakowe, 311
- znaki modyfikujące, *Patrz:* modyfikatory
- zestawienie, 314
- znaki specjalne, 371
- język programowania
 - C, 135
 - COBOL, 33
 - Java, 31, 549
 - JavaScript, *Patrz:* JavaScript
 - PHP, *Patrz:* PHP
 - Python, 61
 - VBScript, 308
- jQuery, 395, 629
 - animacje, 504
 - dołączanie, 480
 - dostosowywanie, 482
 - dynamiczne stosowanie klas, 511
 - gotowość dokumentu, 488
 - jQuery User Interface, 528
 - łączenie selektorów, 486
 - manipulowanie drzewem DOM, 507
 - metody, 704
 - obiekty, 703
 - obsługa zdarzeń, 486, 489
 - pobieranie, 481
 - przetwarzanie elementów specjalnych, 499
 - rozszerzenia, 528
 - selektor elementów, 485
 - selektor identyfikatorów, 486
 - selektor klas, 486
 - selektory, 699
 - unikanie konfliktów, 484
- jQuery Mobile, 629
 - API, 532
 - dołączanie, 532
 - obsługa list, 541
 - usprawnianie progresywne, 531
- JSON, 395
- Kindle, 549
- klasa, 120
 - definiowanie, 346
 - deklarowanie, 348
 - instancja, 345
 - instancja klasy, 120
 - pochodna, 121
 - Subscriber, 130
- klasy, 111
 - definiowanie, 124, 130
 - deklarowanie, 121
 - dziedziczenie, 130, 132
 - rozszerzanie, 130
 - User, 129
 - znaków, 367
- klucze, *Patrz:* indeksy
 - główne, 214
 - obce, 217
- klucze główne, 195
 - dodawanie, 196
- kod JavaScript
 - debugowanie, 309
- kodeki
 - audio, 602
 - wideo, 607
- kolumny
 - usuwanie, 192
 - zmiana nazwy, 192
- komentarze JavaScript
 - składnia, 309
- komunikacja asynchroniczna, 382, 526
 - po stronie serwera, 387
- komunikat
 - Query OK, 180, 183, 189
- konstrukcja
 - foreach, 272
 - foreach ... as, 147
 - if, 102
 - if ... else, 95, 363
 - if ... else if ..., 333
 - if ... else if ... else, 96, 97
 - JOIN ... ON, 208
 - MATCH ... AGAINST, 202
 - tryb boolowski, 203
 - try ... catch, 331, 383
 - UPDATE ... SET, 204

konstruktory, 125, 345

odwołanie, 132

podklas, 132

konwersja

jawna, 106

liczby na łańcuch znaków, 73

łańcucha znaków na liczbę, 73

niejawna, 106

typów, 328

kopie zapasowe, planowanie, 234

kopiowanie bazy danych, 232

kwalifikator

DISTINCT, 200

LIKE, 201

LIMIT, 200, 201, 205

UNSIGNED, 187

WHERE, 200

ZEROFILL, 187

L

LAMP, 42

instalowanie, 51

linkowanie dynamiczne, 107

Linux, 169, 176, 177

literały, 85, 324

L

łańcuchy znaków, 70, 184

długość, 152

dopełnianie, 152

konkatenacja, 69

konwersja, komponenty modyfikatora, 152

ogranicznik, 144

rozdzielanie, 144

typy, 70

łańcuchowanie metod, 344

M

macOS, 169, 176

magiczne stałe, *Patrz*: stałe predefiniowane

MAMP, 42

MariaDB, 32

mechanizm renderujący

Mozilla, 454

WebKit, 454

metaznaki, 365

metoda

\$.each, 525

\$.map, 526

addColorStop, 562

arc, 578

arcTo, 581

attr, 508

bezierCurveTo, 583

bind_param, 259

clearRect, 559

clip, 575

close, 243

concat, 354

createImageData, 591

createLinearGradient, 560

createRadialGradient, 563

css, 485

definiowanie, 346

drawImage, 584

fetch_assoc, 240

fillRect, 558

fillText, 568

forEach, 354

GET, 246, 265, 384, 388, 527

get_password, 126

getElementById, 319

getElementsByTagName, 393

getImageData, 588

height, 512

hide, 500

hover, 497

html, 508

innerWidth, 514

is, 523

isPointInPath, 578

join, 355

lineTo, 573

measureText, 569

moveTo, 573

mysqli, 239

noConflict, 484

open, 390

pop, 355

POST, 246, 265, 384, 526

preventDefault, 494

push, 351, 355

putImageData, 591

quadraticCurveTo, 581

- ready, 488
- rect, 573
- replace, 349, 373
- reverse, 357
- send, 390
- show, 500
- sort, 357
- stroke, 573
- strokeRect, 559
- strokeText, 567
- test, 373
- text, 508
- toDataURL, 565
- toggle, 501
- toUpperCase, 330
- val, 508
- width, 512
- metody, 111, 120
 - deklarowanie, 126
 - final, 133
 - tworzenie, 133
 - priority, 131
 - statyczne, 129, 349
 - dostęp, 129
 - zasięg, 128
 - zastosowanie do obsługi tablic, 353
- Microsoft
 - Internet Explorer, 382
 - SQL Server, 173
- MIME, *Patrz:* typy mediów
- modyfikatory, 373
- modyfikowanie
 - funkcji
 - korzystającej z tablicy arguments, 342
- MP3, kodek audio, 602
- MP4, kontener, 607
- MySQL, 173, 629
 - aktualizowanie danych, 253
 - aplikacja
 - phpMyAdmin, 209
 - archiwizacja danych, 230
 - dodawanie danych, 252
 - dostęp z wiersza poleceń, 174
 - elementy zastępcze, 258, 276
 - funkcje, 209
 - funkcje wbudowane, 689
 - informacje o tabeli, 251
 - InnoDB, 182
 - instrukcje, 179
 - interpretacja zapytań, 228
 - nawiązywanie połączenia, 239
 - obsługa
 - transakcji, funkcje, 226
 - zaawansowana, 213
 - odczytywanie danych, 253
 - pobieranie rezultatu, 240
 - podstawy, 173
 - polecenia, 179
 - przechowywanie haseł, 291
 - przykładowa baza danych, 174
 - przywracanie danych, 230
 - stopwords, 685
 - średnik, zastosowanie, 178
 - tworzenie
 - tabeli, 250
 - użytkowników, 180
 - zapytań, 198
 - typy danych, 184
 - binarnych, 185
 - uprawnienia, 182
 - usuwanie
 - danych, 254
 - rekordu, 247
 - tabeli, 252
 - zamykanie połączenia, 243
 - zapobieganie atakom, 256
 - zapytania, 173, 240
 - pomocnicze, 255
 - zdalny serwer, 177
 - znaki zachęty, 179

N

- nagłówki dokumentu, 307
- nawiasy
 - klamrowe, 94, 103, 142
 - kwadratowe, 143
- Netscape Navigator
 - przeglądarka, 305
- niedozwolony znak, 363
- nocache, 390
- normalizacja
 - bazy danych, 215
 - przeciwskazania, 222

O

- obiekt, 345
 - \$conn, 239
 - \$result, 240
 - \$stmt, 259
 - canvas, 549, 555, *Patrz też:* canvas
 - event, 494
 - idata, 589
 - localStorage, 618
 - metody, 619
 - password, 126
 - request, 386
 - tworzenie, 347
 - window, 465
 - właściwości, 120, 345
 - XMLHttpRequest
 - implementacja, 382
- obiektyowy model dokumentu, *Patrz:* DOM
- obiekty, 111, 120
 - dostęp, 347
 - interfejs, 120
 - klonowanie, 124
 - odwołanie, 122
 - rozszerzanie, 349
 - tworzenie, 122
 - właściwości, 123
- obrazy, wysyłanie, 164
- oczyszczanie kodu, 81
- odtwarzanie danych, 233
- Ogg, kontener, 608
- Open Source, 38
- Opera, 549
- operator
 - ?, 99, 335
 - <<<, 71
 - clone, 124
 - identyczności, 90, 327
 - identyczności, ===, 291
 - OR, 92
 - podwójny dwukropek, 127
 - przypisania, 88
 - równości, ==, 291
 - równoważności, 90, 327
 - trzyargumentowy, 335
 - typeof, 315, 317
 - zaprzeczenia identyczności, 91
 - zaprzeczenia równości, 91

- operatory, 65, 86
 - arytmetyczne, 66, 86
 - asocjacyjność, 326
 - dekrementacji, 69
 - dwuargumentowe, 86
 - hierarchia, 86, 87
 - inkrementacji, 69
 - JavaScript, 312
 - jednoargumentowe, 86
 - logiczne, 67, 83, 86, 91, 92, 328
 - zastosowanie, 209
 - porównania, 67, 86, 91, 328
 - priorytet, 86, 92, 325
 - przypisania, 66, 85, 86
 - relacji, 89, 326
 - trzyargumentowy, 86
- Oracle, 32, 173

P

- PCM, kodek audio, 603
- Perl, 31
- pętle, 101
 - do ... while, 103, 336
 - for, 103, 104, 136, 337
 - zastosowanie, 337
 - foreach ... as, 138, 141
 - switch, 337
 - while, 101, 104, 139, 335
- PHP, 29, 57, 633
 - białe znaki, 61
 - błędne zagnieżdżanie komentarzy, 60
 - ciasteczka, 285
 - debugowanie, 74
 - deklarowanie zmiennych, 61
 - dodawanie elementów do HTML, 57
 - funkcje, 76, 111, 112
 - instrukcje wielowierszowe, 71
 - komentarze, 59, 60
 - komunikacja z MySQL, 237
 - linkowanie dynamiczne, 107
 - literały, 85
 - łańcuchy, 62, 70
 - łączenie, 69
 - magic quotes, 257, 277
 - obiekty, 120
 - oczyszczanie danych z formularzy, 276
 - operatory, 65, 86

- pętle, 101
- pliki, 156
- rzutowanie, 107
- składnia, 59
- stałe, 74
- tablice, 63, 135
- wersja, 119
- wstawianie symboli specjalnych, 70
- wyrażenia, 83
 - regularne, 373
- wysyłanie zapytań do bazy danych, 248
- zalecana wersja, 111
- zmiennne, 61, 85
 - numeryczne, 62
 - tekstowe, 61
- znaki modyfikujące, 70
- phpDesigner, 54
- phpMyAdmin, 209
- pierwsza postać normalna, 215
- plik
 - checkuser.php, 640
 - exec.php, 169
 - friends.php, 650
 - functions.php, 630
 - header.php, 633
 - index.php, 636
 - javascript.js, 660
 - login.php, 238, 641
 - logout.php, 656
 - members.php, 647
 - messages.php, 653
 - signup.php, 637
 - styles.css, 657
 - validate.html, 360
- pliki
 - aktualizowanie, 160
 - blokada, 162
 - dołączanie, 118
 - jednokrotne, 118
 - kasowanie, 160
 - kopia zapasową, 231
 - kopiowanie, 159
 - logowania, 238
 - nazewnictwo, 156
 - nazwy, 168
 - obsługa, 156
 - procedura, 157
 - ochrona przed wielokrotnym otwarciem, 162
 - odczytywanie, 158, 163
 - odwołanie, 156
 - odwoływanie
 - bezpieczne, 160
 - operacje, 157
 - przenoszenie, 160
 - przesyłanie, 168
 - sprawdzenie istnienia, 156
 - tworzenie, 157
 - uchwyt, 157
 - wczytywanie, 119
 - wielkość znaków, 156
 - wskaźnik, 160
 - wysyłanie, 164
 - zamykanie, 157
 - zapisywanie, 105, 157
- podklasa, *Patrz też:* klasa pochodna
 - konstruktor, 132
- podwójny ukośnik, (*//*), 310
- pole
 - color, 283
 - input, 265
 - number, 283
 - range, 283
 - typu CHAR, 184
 - typu VARCHAR, 184
- połączenie
 - break, 334
 - CREATE TABLE, 187
 - DELETE, 200
 - dir, 169
 - INSERT, 190
 - ls, 169
 - SELECT, 198
 - SELECT COUNT, 199
 - SELECT DISTINCT, 199
 - SHOW databases, 177
- polskie znaki, 141
- port Apache
 - zmiana, 62
- postać normalna
 - druga, 218
 - pierwsza, 215
 - trzecia, 220
- praca zdalna, 52
- priorytety
 - asocjacyjność, 88, 89
 - operatory, 88
- procedura żądanie/odpowiedź, 28

- projektowanie
 - aplikacji, 630
 - bazy danych, 213
 - protokół
 - HTTP
 - autoryzacja, 288
 - HTTPS, 301
 - SSH, 177
 - SSL, 301
 - TLS, 301
 - przekazanie
 - przekazywanie argumentów przez
 - referencję, 115
 - zawartości zmiennej jako argumentu, 78
 - przerywanie
 - działania instrukcji, 334
 - pętli, 337
 - przypisywanie wartości
 - elementom, 350
 - właściwościom, 127
 - zmiennym, 68, 315
 - Python, 61
- R**
- relacja
 - jeden do jednego, 223
 - jeden do wielu, 224
 - wiele do wielu, 224
 - rozszerzanie obiektów, 350
 - równoważność, 90
 - sprawdzanie, 90
 - rzutowanie
 - jawne, 106
 - niejawne, 106
- S**
- Safari, 549
 - serwer Apache, 37
 - autoryzacja HTTP, 288
 - obsługa z poziomu wiersza poleceń, 178
 - sesje
 - bezpieczeństwo, 300
 - inicjowanie, 297
 - kończenie, 299
 - obsługa, 296
 - określanie czasu trwania, 300
 - przechowywanie danych sesji, 303
 - śledzenie, 285
 - wymuszanie korzystania z ciasteczek, 303
 - zapobieganie atakom session fixation, 302
 - zapobieganie przejmowaniu, 301
 - sieci dostarczania treści, CDN, 482
 - silnik
 - InnoDB, 182
 - WebKit, 549
 - składnia heredoc, 71
 - składowanie danych, 226
 - słowo kluczowe
 - array, 137, 351
 - AS, 208
 - catch, 330
 - class, 121
 - default, 335
 - DROP, 192
 - extends, 130
 - final, 133
 - global, 79
 - GROUP BY, 206
 - MODIFY, 191
 - ORDER BY, 205
 - parent, 131
 - private, 128
 - protected, 128
 - prototype, 347, 349
 - public, 128
 - self, 127, 130, 132
 - this, 386, 468, 490
 - try, 330
 - VALUES, 190
 - var, 316
 - WHERE, 200
 - sortowanie
 - numeryczne
 - malejące, 358
 - rosnące, 358
 - rosnące lub malejące, 205
 - w odwrotnej kolejności alfabetycznej, 357
 - sprawdzanie
 - adresu e-mail, 364
 - hasła, 364
 - imię, 363
 - nazwisko, 363
 - nazwy użytkownika, 363
 - wieku, 364
 - zasięgu zmiennych, 317

SQL, 33
instrukcje, 232
stała
 __CLASS__, 75
 __DIR__, 75
 __FILE__, 75
 __FUNCTION__, 75
 __LINE__, 75
 __METHOD__, 75
 __NAMESPACE__, 75
 FALSE, 84
 NULL, 84
 ROOT_LOCATION, 74
 TRUE, 84
stałe, 74
 deklarowanie, 127
 magiczne, 74
 nazewnictwo, 74
 predefiniowane, 74
strona WWW, tworzenie, 135
Sun Microsystems, 32
superklasa, 121
SVG, 540
symbol
 #, 399
 \$, 126, 146, 483
 zastosowanie, 319
symbole specjalne, 70
szesnastkowy system, 149

Ś

środowiskami programistyczne
zintegrowane, IDE, 54

T

tabele
 AUTO_INCREMENT, 188
 dodawanie danych, 227
 indeksy, 193
 łączenie, 206
 nadmiar danych, 218
 powielanie danych, 199
 sprawdzanie, 183
 tworzenie, 182, 192, 207
 kopii zapasowej, 232, 233
 z kluczem głównym, 196

usuwanie, 192
 wierszy, 200
wprowadzanie danych, 189
wyświetlanie, 192
z obsługą transakcji, 226
zliczanie wierszy, 199
zmiana nazwy, 190
tablica
 \$_FILES, 165
 zastosowanie, 166
 zawartość, 166
 \$_GET, 246, 276
 \$_POST, 246, 268, 276, 387
 \$_SERVER, 289
 \$_SESSION, 297
 arguments, 342
tablice, 135
 asocjacyjne, 137, 145, 351
 odwołanie, 137
 tworzenie, 140
 dodawanie elementów, 136, 137
 dwuwymiarowe, 64
 deklarowanie, 64
 funkcje, 143
 globalne, 145
 identyfikatory, 138
 indeksowane numerycznie, 135
 indeksy liczbowe, 142
 jednowymiarowe, 63
 numeryczne, 350
 odwołania, 141
 prosty dostęp, 135
 przeglądanie, 138
 przypisywanie
 numerów elementom, 137
 wartości, 136
 umieszczanie elementów, 135
 wielowymiarowe, 140, 352
 tworzenie, 142
 wyświetlanie elementów, 136
 zagnieżdżone, 141
 zliczanie elementów, 143
Tclm 308
Theora, kodek wideo, 608
Tizen, 549
transakcje, 226
 anulowanie, 228
 zastosowanie, 227

- tworzenie
 - dynamicznych stron WWW, 305
 - indeksu, 193
 - pliku
 - z kopią zapasową, 231
 - tabeli, 189
- typowanie
 - jawne, 339
 - słabe, 73, 90, 106, 315
- typy danych
 - binarnych, 185
 - BINARY, 185
 - BLOB, 186
 - CHAR, 185
 - DATE i TIME, 187
 - liczbowych, 186
 - TEXT, 185
 - zmiana typu w kolumnie, 191

U

- układ wielokolumnowy, 447
- Unix, 169, 177
- uprawnienia, 166
- usuwanie wierszy, 200
- użytkownicy
 - tworzenie, 180
 - uprawnienia, 160

V

- VBScript, 308
- Vorbis, kodek audio, 603
- VP8, kodek wideo, 608
- VP9, kodek wideo, 608

W

- WAMP, 42
- wątki robocze, web workers, 554
- WebGL, 556
- WebM, kontener, 608
- WebSQL, 554
- wersja, sprawdzanie zgodności, 119
- wiersz poleceń, 174
- Windows, 175
- właściwości
 - deklarowanie, 126
 - kolumny, 188

- obiektu, 345
- statyczne, 129, 349
- zasięg, 128
- właściwość
 - \$static_property, 130
 - auto, 440
 - AUTO_INCREMENT, 188
 - background-clip, 438
 - background-origin, 439
 - background-size, 440
 - border-color, 442
 - border-radius, 443
 - box-shadow, 446
 - display, 500
 - font, 567
 - innerHTML, 322
 - INT UNSIGNED, 188
 - KEY, 188
 - length, 343
 - lineCap, 570
 - lineJoin, 570
 - lineWidth, 570
 - miterLimit, 572
 - NOT NULL, 188
 - onreadystatechange, 386
 - opacity, 450
 - overflow, 446
 - readyState, 386
 - wartości, 386
 - textAlign, 568
 - textBaseLine, 567
 - text-overflow, 451
 - text-shadow, 450
 - transform, 454
 - word-wrap, 451
- WordPress, 108, 109
- wrażenia, 83
 - boolowskie, 323
 - logiczne, 93
 - regularne, 365, 370
 - klasy znaków, 367
 - przykłady, 372
 - zakres, 368
 - warunkowe, 93, 332
- wyświetlanie danych
 - precyzja, 150
- wywołania
 - systemowe, 169
 - zwrotne, 386

wyzwalacze
wzorzec
 negative lookahead, 368

X

XHTML, 170
XML, 384, 393

Z

zamiana
 na wielkie litery, 343
zaprzeczenie
 identyczności, 91
 równości, 91
zapytanie
 optymalizacja, 229
 SELECT
 łączenie tabel, 207
 WHERE, 217
zdarzenie
 blur, 489
 click, 491
 dblclick, 491
 event.prevent.Default, 535
 focus, 489
 keypress, 492
 mouseenter, 496
 mouseleave, 496
 mousemove, 494
 mouseout, 497
 mouseover, 497
 onerror, 330
 onload, 488
 submit, 498
zliczanie wierszy, 199
zmienna
 \$_COOKIE, 80
 \$count, 62
 \$_ENV, 80
 \$_FILES, 80
 \$_GET, 80
 \$_POST, 80
 \$_REQUEST, 80
 \$_SERVER, 80
 \$_SESSION, 80
 \$count, 80
 \$finished, 92

\$GLOBALS, 80
\$this, 126, 131
counter, 336
string, 330
zmiennie, 85, 324
 globalne, 78, 118, 316
 zwrocenie, 117
 lokalne, 77, 117, 316
 przypisywanie wartości, 68
 statyczne, 79, 118
 superglobalne, 80
 zasady nazewnictwa, 65
 zasięg, 77, 117
 zmiana wartości, 69
znacznik
 <?php>, 58
 <audio>, 602, 603
 <canvas>, Patr: canvas
 <div>, 412
 <form>, 265, 536
 <label>, 275
 <link>, 398
 <meta>, 533
 <pre>, 152
 <script>, 467
 <select>, 274
 <source>, 603, 607
 , 412
 <video>, 607
 IMG, 165
 input, 269
znak
 \$, 126
znaki
 modyfikujące, 70
 określanie zakresu, 368
 specjalne, 169
znaki specjalne, 371
zwrocenie
 łańcucha znaków, 343
 tablicy, 344
 wartości, 343

Z

żądanie
 HTTP POST, 385
 status, 386
 XML, 390

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

PHP, MySQL i JavaScript: klasyczne narzędzia dla nowoczesnych rozwiązań!

Trio: PHP, MySQL i JavaScript jest znane jako najwygodniejszy zestaw narzędzi do tworzenia dynamicznych stron internetowych, które do działania wymagają bazy danych. Mimo upływu lat i dynamicznego rozwoju konkurencyjnych technologii twórcy aplikacji WWW wciąż cenią PHP, MySQL i JavaScript za otwarte źródła, brak opłat za korzystanie, elastyczność i łatwość w nauce. Każdy ambitny programista posługujący się systemem Unix czy Linux z serwerem Apache powinien zapoznać się z tymi narzędziami. W połączeniu z takimi technologiami jak jQuery, CSS i HTML5 pozwalają budować serwisy porównywalne z gigantami w rodzaju Facebooka, Twittera czy Gmaila.

Ta książka jest kolejnym, uzupełnionym i zaktualizowanym wydaniem znakomitego wprowadzenia do projektowania dynamicznych stron internetowych. Oprócz przystępnego omówienia technik tworzenia responsywnych stron WWW znalazły się tu solidne podstawy PHP, MySQL, JavaScriptu, CSS i HTML5; opisano też możliwości bibliotek jQuery i jQuery Mobile. Pokazano, jak połączyć możliwości tych technologii, oraz opisano zalecane metody programowania. Co istotne, w książce zawarto wskazówki dotyczące optymalizacji stron WWW pod kątem urządzeń mobilnych. Dzięki licznym przykładom uzyskaną wiedzę można na bieżąco testować w praktyce.

Robin Nixon – od ponad 40 lat tworzy oprogramowanie, strony internetowe i aplikacje. Jest autorem przeszło 500 artykułów, ponad 30 książek oraz internetowych kursów wideo. Interesują go psychologia, motywacja, sztuczna inteligencja i różne gatunki muzyki; z pasją oddaje się studiom nad filozofią i kulturą.

W tej książce między innymi

- baza danych i zapytania MySQL
- podstawy tworzenia dynamicznych stron PHP
- mechanizm sesji i zachowanie zasad bezpieczeństwa
- biblioteki jQuery i jQuery Mobile oraz żądania Ajax
- CSS2 i CSS3 oraz nowe funkcje HTML5: geolokacja, obsługa dźwięku i filmów, Canvas

Helion

 helion.pl

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

Sprawdź nasze szkolenia!
 SZKOLENIA

 AKADEMIA IT & BUSINESS
 WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
 Sięgnij po więcej! ▶



ISBN 978-83-283-5149-3



9 788328 351493