

KOMPENDIUM WIEDZY O HTML5!

Apress®

HTML5

Przewodnik
encyklopedyczny

Adam Freeman

Helion 

Tytuł oryginału: The Definitive Guide to HTML5

Tłumaczenie: Maksymilian Gutowski

ISBN: 978-83-246-5081-1

Original edition copyright © 2011 by Adam Freeman.
All rights reserved.

Polish edition copyright © 2013 by Helion S.A.
All rights reserved.

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 jakiejkolwiek 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/htm5pe>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/htm5pe.zip>

Printed in Poland.

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

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

Spis treści

	O autorze	23
	O korektorach merytorycznych	24
	Podziękowania	25
Część I	Na dobry początek	27
Rozdział 1	Czym jest HTML5	29
	Historia HTML	29
	Wprowadzenie JavaScript	30
	Koniec wojen przeglądarek	30
	Dominacja wtyczek	30
	Pojawienie się semantycznego HTML	31
	Trend: standard HTML nie nadąża za sposobami użycia HTML	31
	HTML5	31
	Nowy standard, nowe standardy	32
	Natywna obsługa multimediów	32
	Zasoby programistyczne	32
	Sieć semantyczna	33
	Bieżący stan HTML5	33
	Obsługa HTML5 w przeglądarkach	33
	Wykorzystanie HTML5 na stronach	33
	Struktura tej książki	33
	Więcej informacji o HTML5	34
	Podsumowanie	34
Rozdział 2	Przygotuj się	35
	Wybór przeglądarki	35
	Wybór edytora HTML	36
	Wybór serwera internetowego	36

Pobranie Node.js	36
Pobranie modułu multipart	36
Pobranie przykładowych kodów	37
Podsumowanie	37
Rozdział 3 Wprowadzenie do HTML	39
Korzystanie z elementów	40
Dodatkowe elementy występujące w tym rozdziale	41
Puste elementy	42
Samozamykające znaczniki	42
Elementy void	43
Atrybuty elementów	44
Nadawanie elementom wielu atrybutów	44
Atrybuty boolowskie	44
Atrybuty autorskie	45
Tworzenie dokumentu HTML	45
Nadrzędna struktura	46
Metadane	46
Treść	47
Rodzice, dzieci, potomkowie i bracia	47
Typy elementów	48
Encje HTML	48
Globalne atrybuty HTML5	49
Atrybut accesskey	49
Atrybut class	50
Atrybut contenteditable	52
Atrybut contextmenu	53
Atrybut dir	53
Atrybut draggable	53
Atrybut dropzone	53
Atrybut hidden	53
Atrybut id	55
Atrybut lang	55
Atrybut spellcheck	56
Atrybut style	57
Atrybut tabindex	57
Atrybut title	58
Przydatne narzędzia HTML	59
Podsumowanie	59
Rozdział 4 Wprowadzenie do CSS	61
Definiowanie i nadawanie stylów	61
Właściwości CSS występujące w tym rozdziale	62
Zastosowanie stylu inline	62
Tworzenie osadzonego stylu	63
Zastosowanie zewnętrznego arkusza stylu	65

Kaskadowość i dziedziczenie stylów	67
Style przeglądarek	67
Style użytkowników	68
Kaskadowość stylów	69
Zmiana kolejności ważnymi stylami	69
Precyzja i ocena kolejności w przypadkach spornych	70
Dziedziczenie	73
Praca z kolorami w CSS	74
Szczegółowe wskazywanie kolorów	75
Długości w CSS	75
Długości bezwzględne	76
Długości względne	77
Inne jednostki w CSS	82
Miary kątów w CSS	82
Miary czasu w CSS	82
Sprawdzanie obsługi funkcji CSS	83
Przydatne narzędzia CSS	83
Przeglądarkowe raporty o stylach	83
Tworzenie selektorów przy użyciu SelectorGadget	83
Ulepszanie CSS z użyciem LESS	84
Framework CSS	84
Podsumowanie	85

Rozdział 5	Wprowadzenie do JavaScript	87
	Przygotowanie do pracy z JavaScript	88
	Deklaracje	89
	Definiowanie i używanie funkcji	90
	Definiowanie funkcji parametrami	90
	Definiowanie funkcji, które zwracają wyniki	91
	Zmienne i typy	92
	Typy proste	92
	Tworzenie obiektów	93
	Praca z obiektami	95
	Operatory JavaScript	99
	Operatory równości i identyczności	99
	Konwersja typów	102
	Praca z tablicami	104
	Literał tablicowy	105
	Odczytywanie i modyfikowanie zawartości tablicy	105
	Wyliczanie zawartości tablicy	106
	Metody obsługi tablic	106
	Obsługa błędów	106
	Porównywanie wartości undefined i null	108
	Sprawdzanie, czy zmienna lub właściwość ma wartość null lub undefined	110
	Rozróżnianie między null i undefined	111

Przydatne narzędzia JavaScript	111
Debugery JavaScript	111
Biblioteki JavaScript	112
Podsumowanie	112
Część II	
Elementy HTML	113
Rozdział 6	
Elementy HTML — kontekst	115
Semantyka a prezentacja	115
Jak dobrać elementy	116
Im mniej, tym lepiej	116
Nie nadużywaj elementów	116
Wyrażaj się konkretnie i spójnie	116
Nie lekceważ odbiorcy	117
Struktura opisów elementów	117
Spis elementów	118
Elementy dokumentu i metadanych	118
Elementy tekstowe	118
Grupowanie treści	120
Dzielenie treści na sekcje	120
Tworzenie tabel	121
Tworzenie formularzy	121
Osadzanie zasobów	122
Niewdrożone elementy	123
Podsumowanie	123
Rozdział 7	
Tworzenie dokumentów HTML	125
Ustanowienie podstawowej struktury dokumentu	126
Element doctype	126
Element html	127
Element head	127
Element body	128
Opatrzanie dokumentu elementami metadanych	129
Określenie tytułu dokumentu	129
Ustawienie podstawy dla względnych adresów URL	130
Opisanie dokumentu metadanymi	131
Definiowanie stylów CSS	134
Wskazywanie zewnętrznych zasobów	139
Pobieranie zasobów z wyprzedzeniem	142
Wykorzystanie elementów skryptowych	142
Element script	142
Element noscript	148
Podsumowanie	150
Rozdział 8	
Elementy tekstowe	151
Tworzenie odnośników	152
Tworzenie odnośników zewnętrznych	153
Tworzenie względnych odnośników	154

Tworzenie odnośników wewnętrznych	155
Określanie kontekstu przeglądania	155
Oznaczenie treści podstawowymi elementami tekstowymi	156
Wskazywanie słów kluczowych i nazw produktów	156
Wyróżnianie	157
Wskazywanie wyrazów obcych i terminów technicznych	158
Wskazywanie nieścisłości i poprawek	159
Wskazywanie ważnego tekstu	160
Podkreślanie tekstu	161
Pomniejszony tekst	162
Indeksy górne i dolne	163
Łamanie wiersza	164
Wymuszanie łamania wiersza	164
Wskazanie odpowiedniego miejsca na łamanie wiersza	165
Przedstawianie danych wejściowych i wyjściowych	167
Odwołania, cytaty, definicje i skróty	167
Skróty	168
Definicje	169
Cytowanie treści z innego źródła	169
Odwołania do innych źródeł	171
Elementy obsługi językowej	171
Elementy ruby, rt i rp	172
Element bdo	173
Element bdi	174
Pozostałe elementy tekstowe	176
Wskazywanie zwyczajnego ciągu tekstu	176
Podświetlanie tekstu	177
Wskazywanie dodanych lub usuniętych treści	178
Wskazywanie godzin i dat	180
Podsumowanie	181
Rozdział 9 Grupowanie treści	183
Konieczność grupowania treści	184
Tworzenie akapitów	185
Element div	186
Preformatowana treść	188
Cytowanie innych źródeł	189
Przejścia tematyczne	191
Grupowanie treści w listach	193
Element ol	193
Element ul	195
Element li	196
Listy definicji	197
Listy o specyficznej numeracji	199
Ilustracje	200
Podsumowanie	202

Rozdział 10 Tworzenie sekcji	203
Podstawowe nagłówki	204
Ukrywanie podtytułów	205
Tworzenie sekcji	209
Nagłówki i stopki	211
Blok nawigacyjny	215
Artykuły	218
Paski boczne	220
Dane kontaktowe	223
Sekcja szczegółów	224
Podsumowanie	227
Rozdział 11 Elementy tabel	229
Podstawowa tabela	229
Komórki nagłówków	232
Nadanie tabeli struktury	234
Wskazywanie nagłówków i części głównej tabeli	235
Stopka	236
Nieregularne tabele	238
Łączenie nagłówków z komórkami	242
Podpis tabeli	243
Praca z kolumnami	245
Wskazywanie pojedynczych kolumn	247
Nadawanie obramowania elementowi table	249
Podsumowanie	251
Rozdział 12 Formularze	253
Podstawowy formularz	254
Definiowanie formularza	255
Oglądanie danych z formularza	256
Konfiguracja formularza	258
Konfiguracja atrybutu action formularza	258
Konfiguracja metody HTTP	259
Konfiguracja kodowania danych	259
Określanie uzupełniania formularza	261
Określenie docelowego miejsca wyświetlenia odpowiedzi formularza	262
Nazwa formularza	263
Nadawanie formularzom etykiet	264
Automatyczne uaktywnianie elementów input	265
Dezaktywacja poszczególnych elementów input	266
Grupowanie elementów formularza	267
Nadanie elementowi fieldset etykiety	268
Dezaktywowanie grup elementów input przy użyciu elementu fieldset	269
Element button	270
Zastosowanie elementu button do wysyłania formularzy	271
Zastosowanie elementu button do czyszczenia formularzy	272
Zastosowanie elementu button jako elementu generycznego	273

Praca z elementami spoza formularza	274
Podsumowanie	275
Rozdział 13 Konfiguracja elementu input	277
Zastosowanie elementu input do podawania danych tekstowych	278
Określanie rozmiaru elementu	278
Podawanie wartości i treści zastępczych	280
Lista danych	281
Tworzenie pól tekstowych nieaktywnych i tylko do odczytu	284
Określenie kierunku tekstu	285
Zastosowanie elementu input do pobierania haseł	285
Zastosowanie elementu input do tworzenia przycisków	287
Zastosowanie elementu input do ograniczenia rodzaju danych wejściowych	289
Zastosowanie elementu input do pobrania wartości numerycznej	290
Zastosowanie elementu input do uzyskania liczby z podanego zakresu	291
Zastosowanie elementu input do uzyskania wartości boolowskiej	292
Zastosowanie elementu input do tworzenia określonych list opcji	295
Zastosowanie elementu input do pobierania sformatowanych ciągów	296
Zastosowanie elementu input do pozyskiwania godzin i dat	298
Zastosowanie elementu input do pobierania danych o kolorze	300
Zastosowanie elementu input do pobierania haseł wyszukiwania	302
Zastosowanie elementu input do tworzenia ukrytych obiektów danych	303
Zastosowanie elementu input do tworzenia map i przycisków graficznych	305
Zastosowanie elementu input do wczytywania plików	307
Podsumowanie	308
Rozdział 14 Inne elementy formularzy i weryfikacja danych wejściowych	309
Inne elementy formularzy	309
Tworzenie list opcji	309
Pobieranie wielowierszowego tekstu	313
Wskazanie wyniku obliczenia	316
Generowanie par kluczy — publicznego i prywatnego	316
Weryfikacja danych wejściowych	317
Sprawdzenie, czy użytkownik podał wartość	318
Sprawdzenie, czy wartość mieści się w podanym zakresie	319
Sprawdzenie, czy wartość jest zgodna z wyrażeniem regularnym	321
Sprawdzenie, czy wartość jest adresem e-mailowym lub URL	322
Dezaktywacja weryfikacji danych	322
Podsumowanie	323
Rozdział 15 Osadzanie zasobów	325
Osadzanie obrazów	326
Osadzanie odnośników graficznych	327
Tworzenie mapy obrazu działającej po stronie klienta	329
Osadzanie innych dokumentów HTML	331

Osadzanie zasobów przy użyciu wtyczek	333
Zastosowanie elementu embed	334
Zastosowanie elementów object i param	334
Inne zastosowania elementu object	337
Zastosowanie elementu object do osadzania obrazów	337
Zastosowanie elementu object do tworzenia map obrazu działających po stronie klienta	338
Zastosowanie elementu object jako kontekstu przeglądania	338
Osadzanie wyobrażeń wartości liczbowych	339
Paski postępu	339
Przedstawienie wartości z zakresu	340
Osadzanie innych elementów	342
Osadzanie zasobów audiowizualnych	342
Osadzanie grafiki	342
Podsumowanie	342
Część III Kaskadowe arkusze stylów	343
Rozdział 16 Wprowadzenie do CSS	345
Standaryzacja CSS	345
Model polowy	346
Spis selektorów	346
Spis właściwości	349
Właściwości obramowania i tła	349
Właściwości modelu polowego	350
Właściwości layoutu	351
Właściwości tekstu	352
Właściwości przejść, animacji i przekształceń	353
Inne właściwości	353
Podsumowanie	354
Rozdział 17 Selektory CSS — część I	355
Podstawowe selektory CSS	356
Wybranie wszystkich elementów	356
Wybieranie elementów według typu	357
Wybieranie elementów według klasy	358
Wybieranie elementów według id	359
Wybieranie elementów według atrybutów	360
Łączenie selektorów	363
Tworzenie zestawień selektorów	364
Wybieranie potomków	365
Wybieranie dzieci	366
Wybieranie braci	368
Selektory pseudoelementów	369
Selektor ::first-line	370
Selektor ::first-letter	370

Selektory :before i :after	372
Licznik CSS	373
Podsumowanie	374
Rozdział 18 Selektory CSS — część II	375
Strukturalne selektory pseudoklas	376
Selektor :root	376
Selektor dzieci	377
Selektory n-tego dziecka	381
Selektory pseudoklas interfejsu użytkownika	382
Wybieranie aktywnych bądź nieaktywnych elementów	382
Wybieranie zaznaczonych elementów	383
Wybieranie elementów domyślnych	384
Wybieranie elementów input o poprawnych i niepoprawnych wartościach	385
Wybieranie elementów input na podstawie zakresów	386
Wybór elementów input według obecności atrybutu required	387
Dynamiczne selektory pseudoklas	388
Selektory :link i :visited	388
Selektor :hover	389
Selektor :active	390
Selektor :focus	391
Inne pseudoselektory	392
Selektor negacji	392
Selektor :empty	393
Selektor :lang	394
Selektor :target	394
Podsumowanie	395
Rozdział 19 Obramowania i tła	397
Nadanie obramowania	398
Szerokość obramowania	399
Styl obramowania	399
Nałożenie obramowania na jedną krawędź	399
Właściwości zbiorcze obramowania	401
Obramowania z zaokrąglonymi rogami	402
Obrazy w obramowaniu	404
Tła elementów	409
Kolor i obraz tła	409
Rozmiar obrazu tła	410
Położenie obrazu tła	411
Zaczepienie obrazu tła	412
Pozycja początkowa tła i styl przycinania	413
Właściwość zbiorcza background	415
Cienie	416
Obrysy	418
Podsumowanie	421

Rozdział 20	Model polowy	423
	Dopełnienia	424
	Marginesy	426
	Rozmiar elementu	427
	Pole wymiarów	429
	Określanie minimalnych i maksymalnych wielkości	430
	Przepełnienie	431
	Widoczność elementu	434
	Typy pola elementu	435
	Elementy blokowe	435
	Elementy liniowe	437
	Elementy liniowo-blokowe	439
	Elementy run-in	439
	Ukrywanie elementów	442
	Pływające pola	442
	Zapobieganie stykaniu się elementów	445
	Podsumowanie	448
Rozdział 21	Tworzenie layoutów	449
	Pozycjonowanie	450
	Rodzaje pozycjonowania	450
	Kolejność elementów	452
	Layouty wielokolumnowe	454
	Layouty flexboksowe	456
	Prosty flexbox	459
	Użycie wielu elastycznych elementów	460
	Przestrzeń pionowa	461
	Maksymalne wielkości	463
	Layouty tabelowe	465
	Podsumowanie	467
Rozdział 22	Stylizacja tekstu	469
	Podstawowe style tekstu	470
	Wyrównywanie i justowanie tekstu	470
	Białe znaki	472
	Kierunek tekstu	474
	Odstępy między wyrazami, literami i wierszami	475
	Dzielenie wyrazów	476
	Wcięcie pierwszego wiersza	478
	Dekorowanie i przekształcanie tekstu	479
	Nadawanie tekstowi cienia	480
	Fonty	482
	Wybór fonta	482
	Rozmiar fonta	484
	Styl i grubość fonta	485
	Fonty internetowe	486
	Podsumowanie	487

Rozdział 23	Przejścia, animacje i przekształcenia	489
	Przejścia	490
	Odwrócone przejścia	493
	Wybór sposobu obliczania wartości pośrednich	494
	Animacje	496
	Klatki kluczowe	498
	Określenie kierunku powtórzeń	501
	Stan końcowy	502
	Animowanie elementów layoutu	503
	Wielokrotne wykorzystanie klatek kluczowych	504
	Nadawanie wielu animacji wielu elementom	505
	Wstrzymywanie i wznowianie animacji	507
	Przekształcenia	508
	Nadanie przekształcenia	509
	Określenie punktu wyjścia	510
	Utworzenie animacji lub przejścia z przekształceniem	511
	Podsumowanie	512
Rozdział 24	Inne właściwości i funkcje CSS	513
	Określenie koloru i krycia elementu	513
	Określenie koloru pierwszego planu	514
	Określenie krycia elementu	514
	Obstylowanie tabel	516
	Kondensowanie obramowania tabel	516
	Konfiguracja oddzielonych krawędzi	518
	Obsługa pustych komórek	519
	Położenie podpisu	520
	Określenie układu tabeli	521
	Stylizacja list	523
	Określenie typu znaczników listy	523
	Użycie obrazu jako znacznika listy	524
	Określenie położenia znacznika	525
	Określenie stylu kursora	527
	Podsumowanie	528
Część IV	Praca z DOM	529
Rozdział 25	Wprowadzenie do DOM	531
	Obiektowy model dokumentu	531
	Poziomy DOM i zgodność	533
	Testowanie obsługi funkcji DOM	533
	Spis funkcji DOM	534
	Właściwości Document	534
	Właściwości Window	537
	Właściwości HTMLElement	539

Właściwości CSS w DOM	541
Zdarzenia DOM	544
Podsumowanie	545
Rozdział 26 Obiekt Document	547
Metadane dokumentu	549
Pobieranie informacji z dokumentu	549
Obiekt Location	552
Odczytywanie i zapisywanie ciasteczek	555
Stan żądania	556
Pozyskiwanie szczegółów o implementacji DOM	557
Znajdowanie obiektów elementów HTML	558
Zastosowanie właściwości do pozyskiwania obiektów elementów	559
Pozyskanie nazwanego elementu przy użyciu notacji tablicowej	560
Przeszukiwanie elementów	561
Przeszukiwanie łańcuchowe	564
Nawigacja w drzewie DOM	565
Podsumowanie	567
Rozdział 27 Obiekt Window	569
Pozyskanie obiektu Window	570
Pozyskiwanie informacji o oknie	570
Interakcja z oknem	572
Wyświetlanie zapytań	573
Pozyskiwanie ogólnych informacji	575
Praca z historią przeglądarki	575
Poruszanie się po historii przeglądania	576
Wprowadzenie nowej pozycji do historii	577
Dodanie nowej pozycji z adresem innego dokumentu	579
Zapisywanie kompleksowych stanów w historii	581
Zastąpienie pozycji w historii przeglądarki	583
Przekazywanie komunikatów między dokumentami	584
Liczniki	588
Podsumowanie	589
Rozdział 28 Praca z elementami DOM	591
Praca z obiektami elementów	592
Praca z klasami	593
Praca z atrybutami elementu	596
Praca z tekstem	600
Modyfikacja modelu	602
Tworzenie i usuwanie elementów	603
Powielanie elementów	604
Przenoszenie elementów	605
Porównywanie obiektów elementów	606

Praca z fragmentami HTML	608
Wstawienie elementu do bloku tekstu	612
Podsumowanie	613
Rozdział 29 Stylizacja elementów DOM	615
Praca z arkuszami stylów	616
Pobieranie podstawowych informacji o arkuszach stylów	616
Praca z ograniczeniami medium	617
Dezaktywowanie arkuszy stylów	620
Praca z wybranymi stylami	621
Praca ze stylami elementów	623
Praca z obiektami CSSStyleDeclaration	624
Praca z właściwościami pomocniczymi	625
Praca ze standardowymi właściwościami	628
Zastosowanie szczegółowych obiektów CSS w DOM	632
Praca z obliczonymi stylami	634
Podsumowanie	637
Rozdział 30 Zdarzenia	639
Wykorzystanie prostych procedur obsługi zdarzeń	640
Wprowadzenie prostej procedury obsługi zdarzeń inline	640
Wprowadzenie prostej procedury obsługi zdarzeń	642
Zastosowanie DOM i obiektu Event	643
Odróżnianie zdarzeń według typu	645
Przepływ zdarzeń	647
Praca ze zdarzeniami, które można anulować	653
Praca ze zdarzeniami HTML	654
Zdarzenia obiektów Document i Window	654
Praca ze zdarzeniami myszy	654
Praca ze zdarzeniami fokusowania	657
Praca ze zdarzeniami klawiatury	658
Praca ze zdarzeniami formularza	660
Podsumowanie	661
Rozdział 31 Obiekty poszczególnych elementów	663
Obiekty dokumentu i metadanych	663
Element base	663
Element body	663
Element link	664
Element meta	664
Element script	664
Element style	665
Element title	665
Inne elementy dokumentu i metadanych	665
Elementy tekstowe	665
Element a	665
Elementy del i ins	666

Element q	666
Element time	666
Inne elementy tekstowe	667
Elementy grupujące	667
Element blockquote	667
Element li	667
Element ol	667
Inne elementy grupujące	668
Elementy sekcji	668
Element details	668
Inne elementy sekcji	668
Elementy tabeli	668
Elementy col i colgroup	668
Element table	669
Elementy thead, tbody i tfoot	669
Element th	669
Element tr	670
Inne elementy tabeli	670
Elementy formularza	671
Element button	671
Element datalist	671
Element fieldset	671
Element form	672
Element input	672
Element label	674
Element legend	674
Element optgroup	674
Element option	675
Element output	675
Element select	676
Element textarea	677
Elementy content	678
Element area	678
Element embed	678
Element iframe	679
Elementy img	679
Element map	680
Element meter	680
Element object	680
Element param	681
Element progress	681
Podsumowanie	682

Część V	Zaawansowane funkcje	683
Rozdział 32	Wprowadzenie do Ajaksa. Część I	685
	Początki z Ajaksem	686
	Obsługa odpowiedzi	689
	Problem z Operą	690
	Zdarzenia Ajaksa	691
	Obsługa błędów	694
	Obsługa błędów wejściowych	696
	Obsługa błędów żądań	696
	Obsługa błędów aplikacji	697
	Pobieranie i definiowanie nagłówków	697
	Zmiana metody HTTP żądania	697
	Wyłączenie buforowania treści	699
	Odczytywanie nagłówków odpowiedzi	699
	Wydawanie żądań Ajaksa między różnymi źródłami	701
	Nagłówek żądania źródła	704
	Zaawansowane funkcje CORS	704
	Przerywanie żądań	704
	Podsumowanie	707
Rozdział 33	Wprowadzenie do Ajaksa. Część II	709
	Przygotowanie się do wysłania danych serwerowi	710
	Utworzenie serwera	710
	Problem	712
	Wysyłanie danych formularza	713
	Wysyłanie danych formularza obiektem FormData	715
	Utworzenie obiektu FormData	716
	Modyfikacja obiektu FormData	716
	Wysyłanie danych JSON	718
	Wysyłanie plików	719
	Śledzenie postępów wczytywania	721
	Żądanie i przetwarzanie różnych typów treści	723
	Pobieranie fragmentów HTML	723
	Pobieranie danych XML	726
	Pobieranie danych JSON	728
	Podsumowanie	729
Rozdział 34	Multimedia	731
	Element video	732
	Wczytywanie filmu z wyprzedzeniem	732
	Wyświetlanie obrazu zastępczego	735
	Rozmiar filmu	735
	Źródło i format filmu	736
	Element track	738

Element audio	739
Praca z osadzonymi mediami za pośrednictwem DOM	740
Pobieranie informacji o mediach	741
Określenie możliwości odtworzenia multimediiów	743
Kontrolowanie sposobu odtwarzania	745
Podsumowanie	747
Rozdział 35 Element canvas. Część I	749
Początki z elementem canvas	750
Uzyskanie kontekstu płótna	751
Rysowanie prostokątów	752
Ustawienie stanu rysowania	754
Styl rysowania styku linii	755
Styl wypełnienia kształtu i rysowania linii	757
Gradienty	757
Gradient promieniowy	762
Wzory	765
Zapisywanie i przywracanie stanu rysowania	767
Rysowanie obrazów	769
Obrazy filmowe	770
Obrazy z elementu canvas	772
Podsumowanie	773
Rozdział 36 Element canvas. Część II	775
Rysowanie z użyciem ścieżek	775
Rysowanie ścieżek liniami	776
Rysowanie prostokątów	779
Rysowanie łuków	780
Wykorzystanie metody arcTo	781
Wykorzystanie metody arc	784
Rysowanie krzywych Béziera	785
Rysowanie sześciennych krzywych Béziera	786
Rysowanie kwadratowych krzywych Béziera	787
Utworzenie obszaru przycinania	789
Rysowanie tekstu	790
Wykorzystanie efektów i przekształceń	792
Cienie	792
Przezroczystość	793
Kompozycja	794
Przekształcenia	796
Podsumowanie	798
Rozdział 37 Funkcja przeciągania i upuszczania	799
Tworzenie elementów źródłowych	799
Obsługa zdarzeń przeciągania	800
Utworzenie obszaru docelowego	802
Przyjmowanie upuszczonych treści	804

Praca z obiektem DataTransfer	806
Filtrowanie przeciąganych elementów według treści	808
Przeciąganie i upuszczanie plików	809
Podsumowanie	812
Rozdział 38 Geolokalizacja	813
Wykorzystanie geolokalizacji	813
Pobranie informacji o bieżącej lokalizacji	813
Obsługa błędów funkcji geolokalizacji	816
Definiowanie opcji geolokalizacji	818
Śledzenie lokalizacji	820
Podsumowanie	821
Rozdział 39 Magazynowanie danych	823
Magazyn lokalny	823
Nasłuchiwanie zdarzeń magazynowania	826
Magazyn sesji	827
Podsumowanie	830
Rozdział 40 Aplikacje internetowe działające offline	831
Problem	831
Utworzenie manifestu	833
Określanie sekcji manifestu	835
Sekcja FALLBACK	835
Sekcja NETWORK	838
Sprawdzenie stanu przeglądarki	838
Praca z buforem offline	839
Aktualizowanie	842
Pobranie aktualizacji	843
Zastosowanie aktualizacji	843
Podsumowanie	844
Skorowidz.....	845

Praca z elementami DOM

W poprzednim rozdziale, przy omówieniu funkcji działających na poziomie dokumentu, wspomniałem o niektórych funkcjach obiektu `HTMLElement`. Teraz możemy się już odpowiednio skoncentrować na nim samym. W niniejszym rozdziale przedstawię Ci poszczególne właściwości i metody `HTMLElement` oraz pokażę, jak ich używać. W tabeli 28.1 znajduje się streszczenie rozdziału. Miej w pamięci, że nie wszystkie popularne przeglądarki będą obsługiwać podane tutaj przykłady.

Tabela 28.1. Streszczenie rozdziału

Problem	Rozwiązanie	Listing
Pobranie informacji o elemencie.	Użyj właściwości metadanych <code>HTMLElement</code> .	28.1
Pobranie lub zdefiniowanie pojedynczego ciągu znaków, zawierającego wszystkie klasy, do których element jest przypisany.	Użyj właściwości <code>className</code> .	28.2
Sprawdzenie lub zmodyfikowanie poszczególnych klas elementów.	Użyj właściwości <code>classList</code> .	28.3
Pobranie lub zdefiniowanie atrybutów elementu.	Użyj metod <code>attribute</code> , <code>getAttribute</code> , <code>setAttribute</code> , <code>removeAttribute</code> i <code>hasAttribute</code> .	28.4, 28.6
Pobranie lub zdefiniowanie niestandardowych atrybutów elementu.	Użyj właściwości <code>dataset</code> .	28.5
Praca z zawartością tekstową elementu.	Użyj obiektów <code>Text</code> .	28.7 – 28.9
Tworzenie i usuwanie elementów.	Użyj metod <code>document.create*</code> i <code>HTMLElement</code> służących do zarządzania elementami-dziećmi.	28.10
Powielenie elementu.	Użyj metody <code>cloneNode</code> .	28.11
Przeniesienie elementu.	Użyj metody <code>appendChild</code> .	28.12
Sprawdzenie identyczności dwóch obiektów.	Użyj metody <code>isSameNode</code> .	28.13
Sprawdzenie równości dwóch obiektów.	Użyj metody <code>isEqualNode</code> .	28.14
Bezpośrednia praca nad fragmentami HTML.	Użyj właściwości <code>innerHTML</code> i <code>outerHTML</code> oraz metody <code>insertAdjacentHTML</code> .	28.15 – 28.17
Wstawienie elementu do bloku tekstu.	Użyj metod <code>splitText</code> i <code>appendChild</code> .	28.18

Praca z obiektami elementów

Obiekty `HTMLElement` obsługują szereg właściwości, których można używać do odczytywania i modyfikowania danych reprezentowanego elementu. W tabeli 28.2 znajduje się opis tych właściwości.

Tabela 28.2. Właściwości danych elementów

Właściwość	Opis	Zwracana wartość
<code>checked</code>	Sprawdza lub definiuje obecność atrybutu <code>checked</code> .	<i>wartość_boolowska</i>
<code>classList</code>	Pobiera lub definiuje listę klas, do których element przynależy.	<code>DOMTokenList</code>
<code>className</code>	Pobiera lub definiuje listę klas, do których element przynależy.	<i>tekst</i>
<code>dir</code>	Pobiera lub definiuje wartość atrybutu <code>dir</code> .	<i>tekst</i>
<code>disabled</code>	Sprawdza lub definiuje obecność atrybutu <code>disabled</code> .	<i>wartość_boolowska</i>
<code>hidden</code>	Sprawdza lub definiuje obecność atrybutu <code>hidden</code> .	<i>wartość_boolowska</i>
<code>id</code>	Pobiera lub definiuje wartość atrybutu <code>id</code> .	<i>tekst</i>
<code>lang</code>	Pobiera lub definiuje wartość atrybutu <code>lang</code> .	<i>tekst</i>
<code>spellcheck</code>	Sprawdza lub definiuje obecność atrybutu <code>spellcheck</code> .	<i>wartość_boolowska</i>
<code>tabIndex</code>	Pobiera lub definiuje wartość atrybutu <code>tabindex</code> .	<i>liczba</i>
<code>tagName</code>	Zwraca nazwę znacznika (wskazując tym samym rodzaj elementu).	<i>tekst</i>
<code>title</code>	Pobiera lub definiuje wartość atrybutu <code>title</code> .	<i>tekst</i>

W listingu 28.1 widać przykład zastosowania niektórych z podstawowych właściwości, które podano w liście.

Listing 28.1. Zastosowanie podstawowych właściwości danych elementów

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {border: medium double black;}
    </style>
  </head>
  <body>

    <p id="textblock" dir="ltr" lang="pl">
      Istnieje wiele różnych rodzajów owoców – sam <span id="banana">banan</span> ma ponad
      pięćset odmian. Gdybyśmy uwzględnili niezliczone odmiany <span id="apple">jabłkę</span>,
      <span id="orange">pomarańczy</span> i innych owoców, musielibyśmy dokonać wyboru
      spośród tysięcy możliwości.
    </p>
    <pre id="results"></pre>
    <script>
      var results = document.getElementById("results");
      var elem = document.getElementById("textblock");

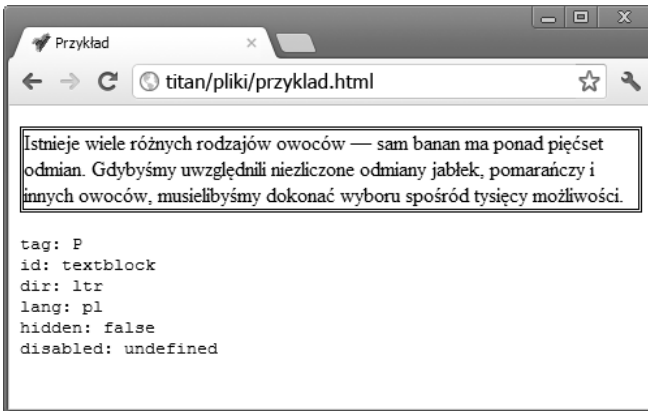
      results.innerHTML += "tag: " + elem.tagName + "\n";
    </script>
  </body>
</html>
```

```

results.innerHTML += "id: " + elem.id + "\n";
results.innerHTML += "dir: " + elem.dir + "\n";
results.innerHTML += "lang: " + elem.lang + "\n";
results.innerHTML += "hidden: " + elem.hidden + "\n";
results.innerHTML += "disabled: " + elem.disabled + "\n";
</script>
</body>
</html>

```

Rezultat zastosowania tych właściwości widać na rysunku 28.1.



Rysunek 28.1. Pobranie informacji o elemencie

Praca z klasami

Praca z klasami, do których przypisany jest element, może się odbywać na dwa sposoby. Pierwszy z nich opiera się na wykorzystaniu właściwości `className`, która zwraca listę klas. Klasy można dodawać i usuwać poprzez zmianę wartości ciągu znaków. Przykład odczytania i zmiany klas tym sposobem widać w listingu 28.2.

Listing 28.2. Zastosowanie właściwości `className`

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {
        border: medium double black;
      }
      p.newclass {
        background-color: grey;
        color: white;
      }
    </style>
  </head>
  <body>

    <p id="textblock" class="fruit numbers">

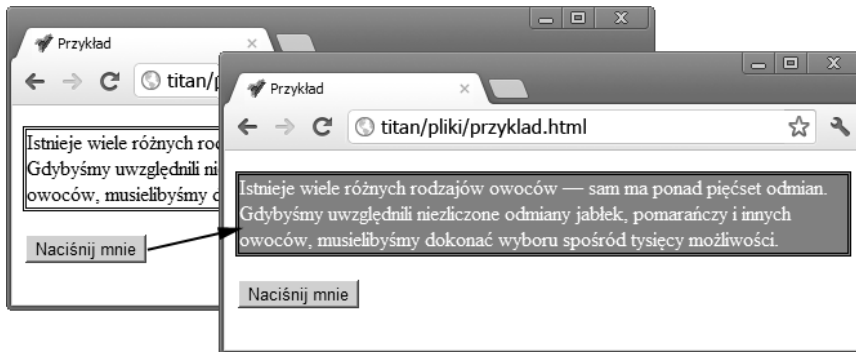
```

Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy dokonać wyboru spośród tysięcy możliwości.

```
</p>
<button id="pressme">Naciśnij mnie</button>
<script>
  document.getElementById("pressme").onclick = function(e) {
    document.getElementById("textblock").className += " newclass";
  };
</script>
</body>
</html>
```

- **Wskazówka** Klasy są standardowo używane do przypisywania elementom stylów. O pracy ze stylami w DOM przeczytasz w rozdziale 29.

W powyższym przykładzie naciśnięcie przycisku przywołuje skrypt, który dodaje nową klasę do listy elementu. Zauważ, że musiałem umieścić spację przed każdą wartością dodaną do właściwości `className`, ponieważ przeglądarka spodziewa się listy oddzielonych od siebie spacjami klas. Po wprowadzeniu takiej zmiany przeglądarka nadaje style zgodnie z selektorami klasowymi, wobec czego dokument widocznie zmienia wygląd, tak jak widać to na rysunku 28.2.



Rysunek 28.2. Zastosowanie właściwości `className`

Właściwość `className` jest łatwa w użyciu, kiedy chcesz szybko przypisać element do nowych klas. O wiele trudniej jednak używać jej do jakichkolwiek innych celów, np. do usunięcia klasy. Na szczęście, możesz też użyć właściwości `classList`, która zwraca obiekt `DOMTokenList`. Obiekt ten obsługuje kilka użytecznych metod i właściwości, które pozwalają na zarządzanie listą klas; opisuje je w tabeli 28.3.

Tabela 28.3. Właściwości i metody `DOMTokenList`

Składowa	Opis	Zwracana wartość
<code>add(<class>)</code>	Dodaje wskazaną klasę do elementu.	<i>void</i>
<code>contains(<class>)</code>	Zwraca wartość <code>true</code> , jeśli element należy do podanej klasy.	<i>wartość_boolowska</i>
<code>length</code>	Zwraca liczbę klas, do których element przynależy.	<i>liczba</i>
<code>remove(<class>)</code>	Usuwa podaną klasę z elementu.	<i>void</i>
<code>toggle(<class>)</code>	Dodaje klasę, jeśli nie jest obecna, a usuwa, jeśli jest obecna.	<i>wartość_boolowska</i>

Abstrahując od tych właściwości i metod, masz również możliwość pobierania klas według numeru indeksowego, z wykorzystaniem notacji tablicowej. Przykład wykorzystania obiektu `DOMTokenList` widać w listingu 28.3.

Listing 28.3. Zastosowanie właściwości `classList`

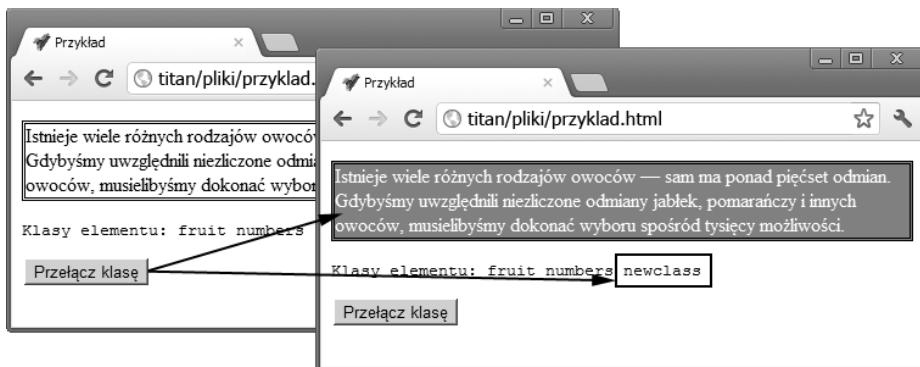
```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {
        border: medium double black;
      }
      p.newclass {
        background-color: grey;
        color: white;
      }
    </style>
  </head>
  <body>
    <p id="textblock" class="fruit numbers">
      Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy
      uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy
      dokonać wyboru spośród tysięcy możliwości.
    </p>
    <pre id="results"></pre>
    <button id="toggle">Przełącz klasę</button>
    <script>
      var results = document.getElementById("results");
      document.getElementById("toggle").onclick = toggleClass;

      listClasses();
      function listClasses() {
        var classlist = document.getElementById("textblock").classList;
        results.innerHTML = "Klasy elementu: "
        for (var i = 0; i < classlist.length; i++) {
          results.innerHTML += classlist[i] + " ";
        }
      }

      function toggleClass() {
        document.getElementById("textblock").classList.toggle("newclass");
        listClasses();
      }
    </script>
  </body>
</html>
```

W tym przykładzie funkcja `listClasses` wykorzystuje właściwość `classList` do pobrania i wyczenia klas, do których przynależy element `p`, wykorzystując przy zwracaniu nazw klas indeksator tablicowy.

Przywoływana po naciśnięciu przycisku funkcja `toggleClass` używa metody `toggle` do dodawania i usuwania klasy `newclass`. Jako że z tą klasą powiązany jest styl, efekt jej zmiany jest widoczny na rysunku 28.3.



Rysunek 28.3. Wyliczenie i przełączenie klasy

Praca z atrybutami elementu

Istnieją właściwości obsługujące niektóre z najważniejszych atrybutów globalnych, ale możliwy jest także odczyt i definiowanie poszczególnych atrybutów elementu. W tabeli 28.4 znajduje się opis służących do tego metod i właściwości obsługiwanych przez obiekt `HTMLElement`.

Tabela 28.4. Właściwości i metody odnoszące się do atrybutów

Składowa	Opis	Zwracana wartość
<code>attributes</code>	Zwraca nadane elementowi atrybuty.	<code>Attr[]</code>
<code>dataset</code>	Zwraca atrybuty <code>data-*</code> .	<code>tekst [<nazwa>]</code>
<code>getAttribute(<nazwa>)</code>	Zwraca wartość podanego atrybutu.	<code>tekst</code>
<code>hasAttribute(<nazwa>)</code>	Zwraca wartość <code>true</code> , jeśli elementowi przypisany jest podany atrybut.	<code>wartość_boolowska</code>
<code>removeAttribute(<nazwa>)</code>	Usuwa podany atrybut elementu.	<code>void</code>
<code>setAttribute(<nazwa>, <wartość>)</code>	Nadaje atrybut o podanej nazwie i wartości.	<code>void</code>

Istnieją cztery proste w użyciu i przewidywalne w działaniu metody, które służą do pracy z atrybutami. W listingu 28.4 widzicie przykład ich zastosowania.

Listing 28.4. Zastosowanie metod atrybutów

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {border: medium double black;}
    </style>
  </head>
  <body>
    <p id="textblock" class="fruit numbers">
      There are lots of different kinds of fruit - there are over 500 varieties
      of banana alone. By the time we add the countless types of apples, oranges,
      and other well-known fruit, we are faced with thousands of choices.
    </p>
  </body>
</html>
```

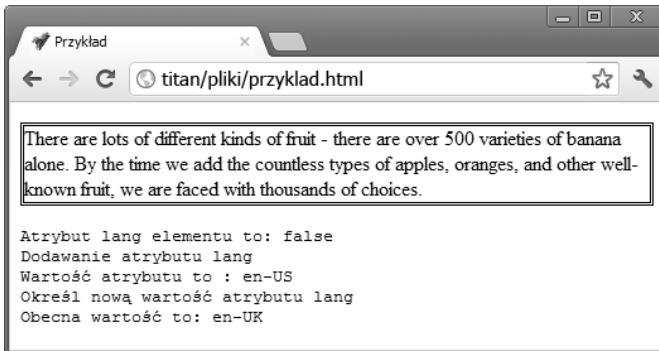
```

</p>
<pre id="results"></pre>
<script>
  var results = document.getElementById("results");
  var elem = document.getElementById("textblock");

  results.innerHTML = "Atrybut lang elementu to: "
    + elem.hasAttribute("lang") + "\n";
  results.innerHTML += "Dodawanie atrybutu lang\n";
  elem.setAttribute("lang", "en-US");
  results.innerHTML += "Wartość atrybutu to : " + elem.getAttribute("lang")
    + "\n";
  results.innerHTML += "Określ nową wartość atrybutu lang\n";
  elem.setAttribute("lang", "en-UK");
  results.innerHTML += "Obecna wartość to: " + elem.getAttribute("lang") + "\n";
</script>
</body>
</html>

```

W tym przykładzie sprawdziłem, dodałem i zmieniłem wartości atrybutu lang. Rezultat zastosowania tego skryptu widać na rysunku 28.4.



Rysunek 28.4. Zastosowanie metod atrybutów

Praca z atrybutami data-*

W rozdziale 3. wspomniałem, że HTML5 obsługuje niestandardowe atrybuty, wykorzystując do tego prefiks data-. Oznacza to, że możesz stworzyć atrybut w rodzaju data-mójnowyatribut. Z niestandardowymi atrybutami w DOM można pracować z wykorzystaniem właściwości dataset, która zwraca tablicę wartości indeksowanych według nazwy owej niestandardowej właściwości. W listingu 28.5 widnieje przykład.

Listing 28.5. Zastosowanie właściwości dataset

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {border: medium double black;}
    </style>
  </head>
  <body>

```

```

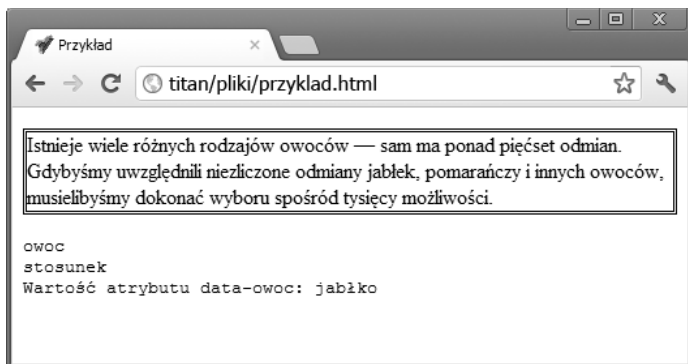
<p id="textblock" class="fruit numbers" data-owoc="jabłko"
  data-stosunek="pozytywny">
  Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy
  uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy
  dokonać wyboru spośród tysięcy możliwości.
</p>
<pre id="results"></pre>
<script>
  var results = document.getElementById("results");
  var elem = document.getElementById("textblock");

  for (var attr in elem.dataset) {
    results.innerHTML += attr + "\n";
  }

  results.innerHTML += "Wartość atrybutu data-owoc: " + elem.dataset["owoc"];
</script>
</body>
</html>

```

Tablica wartości zwróconych przez właściwość `dataset` nie jest indeksowana według pozycji, tak jak ma to miejsce w przypadku standardowych tablic. Do wyciągnięcia atrybutów `data-*` należy użyć deklaracji `for...in`, tak jak w listingu. Możesz też przywołać wartość według jej nazwy. Zauważ, że należy tu podać jedynie tę część nazwy atrybutu, która znajduje się po członie `data-*`. Jeśli więc chcesz sprawdzić wartość atrybutu `data-owoc`, to powinieneś przywołać wartość `dataset["owoc"]`. Rezultat zastosowania tego skryptu widać na rysunku 28.5.



Rysunek 28.5. Zastosowanie właściwości `dataset`

Praca ze wszystkimi atrybutami

Zbiór wszystkich atrybutów elementu możesz pozyskać przy użyciu właściwości `attributes`, która zwraca tablicę obiektów `Attr`. Właściwości obiektu `Attr` opisane są w tabeli 28.5.

Tabela 28.5. Właściwości obiektu `Attr`

Właściwości	Opis	Zwracana wartość
<code>name</code>	Zwraca nazwę atrybutu.	<i>tekst</i>
<code>value</code>	Zwraca lub definiuje wartość atrybutu.	<i>tekst</i>

W listingu 28.6 widnieje przykład zastosowania właściwości `attributes` i obiektu `Attr` do odczytania i zmodyfikowania atrybutów elementu.

Listing 28.6. Praca z właściwością *attributes*

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {border: medium double black;}
    </style>
  </head>
  <body>
    <p id="textblock" class="fruit numbers" data-owoc="jabłko"
      data-stosunek="pozytywny">
      Istnieje wiele różnych rodzajów owoców — sam ma ponad pięćset odmian. Gdybyśmy
      uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy
      dokonać wyboru spośród tysięcy możliwości.
    </p>
    <pre id="results"></pre>
    <script>
      var results = document.getElementById("results");
      var elem = document.getElementById("textblock");

      var attrs = elem.attributes;

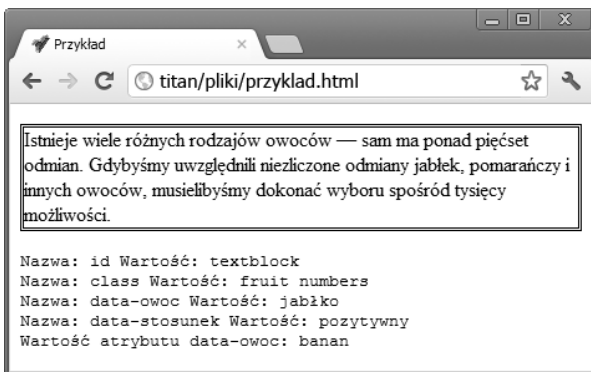
      for (var i = 0; i < attrs.length; i++) {
        results.innerHTML += "Nazwa: " + attrs[i].name + " Wartość: "
          + attrs[i].value + "\n";
      }

      attrs["data-owoc"].value = "banan";

      results.innerHTML += "Wartość atrybutu data-owoc: "
        + attrs["data-owoc"].value;
    </script>
  </body>
</html>

```

Jak widać w listingu, atrybuty w tablicy obiektów `Attr` indeksowane są według pozycji i nazwy. W tym przykładzie wyliczyłem nazwy i wartości przypisanych elementowi atrybutów, a następnie zmieniłem wartość jednego z nich. Rezultat zastosowania tego skryptu widać na rysunku 28.6.



Rysunek 28.6. Zastosowanie właściwości *attributes*

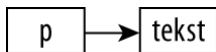
Praca z tekstem

Zawartość tekstową elementu reprezentuje obiekt `Text`, który w modelu dokumentu występuje jako dziecko owego elementu. W listingu 28.7 widzimy element z treścią tekstową.

Listing 28.7. Element z treścią tekstową

```
...
<p id="textblock" class="fruit numbers" data-fruit="apple" data-sentiment="like">
Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy uwzględnili
niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy dokonać wyboru spośród
tysiący możliwości.
</p>
...
```

Reprezentowany w modelu dokumentu element `p` przybiera postać obiektu `HTMLElement`, a jego treść reprezentowana jest przez obiekt `Text`, tak jak widać na rysunku 28.7.



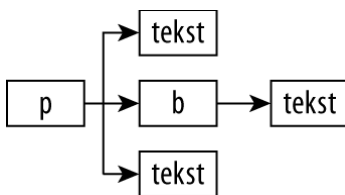
Rysunek 28.7. Relacja pomiędzy obiektami reprezentującymi element i jego treść

Wszystkie zawierające tekst dzieci elementu obsługiwane są w ten sam sposób. W listingu 28.8 zamieściłem w akapicie dodatkowy element.

Listing 28.8. Dodanie elementu do akapitu

```
...
<p id="textblock" class="fruit numbers" data-fruit="apple" data-sentiment="like">
Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy
uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy dokonać
wyboru spośród tysięcy możliwości.
</p>
...
```

Wprowadzenie elementu `b` zmienia hierarchię węzłów tak, by przedstawiony w niej był element `p` i jego zawartość, tak jak widać to na rysunku 28.8.



Rysunek 28.8. Rezultat dodania elementu do akapitu

Pierwsze dziecko elementu `p` jest obiektem `Text`, który reprezentuje tekst od początku bloku do elementu `b`. Element `b` ma własne dziecko, jakim jest obiekt `Text`, który reprezentuje treść zawartą pomiędzy jego otwierającym a zamykającym znacznikiem. Wreszcie, ostatnie dziecko elementu `p` jest obiektem `Text`, reprezentującym tekst znajdujący się po elemencie `b` — do końca bloku. W tabeli 28.6 znajduje się opis składowych obsługiwanych przez obiekt `Text`.

Niestety, nie ma żadnego wygodnego sposobu na znajdowanie elementów `Text`, poza wyszukiwaniem obiektów elementów-rodziców i przechodzeniem przez ich dzieci. Praca z elementami `Text` jest wobec tego trudniejsza, niż powinna. W listingu 28.9 widzimy niektóre z metod i właściwości elementu `Text`.

Po naciśnięciu elementu `button` wyświetlana jest liczba znaków w pierwszym dziecku `Text` elementu `p`, a jego zawartość zostaje zmieniona metodą `replaceWholeText`.

Tabela 28.6. Składowe obiektu *Text*

Składowa	Opis	Zwraca
<code>appendData(<tekst>)</code>	Dodaje określony ciąg znaków na końcu bloku tekstowego.	<i>void</i>
<code>data</code>	Zwraca lub definiuje tekst.	<i>tekst</i>
<code>deleteData(<pozycja>, <liczba znaków>)</code>	Usuwa tekst z ciągu znaków. Pierwsza liczba określa pozycję, a druga liczbę znaków do usunięcia.	<i>void</i>
<code>insertData(<pozycja>, <tekst>)</code>	Zamieszcza podany ciąg znaków na określonej pozycji.	<i>void</i>
<code>length</code>	Zwraca liczbę znaków.	<i>liczbę</i>
<code>replaceData(<pozycja>, <liczba znaków>, <tekst>)</code>	Zamienia wskazany zakres tekstu na określony ciąg znaków.	<i>void</i>
<code>replaceWholeText(<tekst>)</code>	Zamienia cały tekst.	<i>Text</i>
<code>splitText(<liczba>)</code>	Dzieli element <i>Text</i> na dwa osobne elementy we wskazanym miejscu. W punkcie „Wstawienie elementu do bloku tekstu” znajdziesz przykład zastosowania tej metody.	<i>Text</i>
<code>substringData(<pozycja>, <liczba znaków>)</code>	Zwraca ciąg znaków z tekstu.	<i>tekst</i>
<code>wholeText</code>	Zwraca tekst.	<i>tekst</i>

Listing 28.9. Praca z obiektami *Text*

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      p {border: medium double black;}
    </style>
  </head>
  <body>
    <p id="textblock" class="fruit numbers" data-fruit="apple" data-sentiment="like">
      Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy
      uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy
      dokonać wyboru spośród tysięcy możliwości.
    </p>
    <button id="pressme">Naciśnij mnie</button>
    <pre id="results"></pre>
    <script>
      var results = document.getElementById("results");
      var elem = document.getElementById("textblock");

      document.getElementById("pressme").onclick = function() {
        var textElem = elem.firstChild;
        results.innerHTML = "Element zawiera " + textElem.length + " znaków\n";
        textElem.replaceWholeText("0to nowy ciąg znaków");
      };
    </script>
  </body>
</html>

```

- **Ostrzeżenie** Trzeba koniecznie zwrócić uwagę, że przy pracy z tekstem białe znaki nie są kondensowane. Oznacza to, że wszelkie spacje i inne białe znaki służące do ustrukturyzowania kodu HTML są traktowane jako część tekstu.

Modyfikacja modelu

W poprzednich punktach pokazałem Ci, jak używać DOM do modyfikowania pojedynczych elementów. Przy użyciu DOM możesz — na przykład — zmieniać atrybuty i treść tekstową. Jest to możliwe, ponieważ DOM jest aktywnie połączony z dokumentem. Wprowadzenie zmiany w DOM sprawia, że przeglądarka wprowadza odpowiednią zmianę w dokumencie. Dzięki temu połączeniu można nawet zmienić strukturę dokumentu. Można dodawać, usuwać, powielać i kopiować elementy w dowolny sposób. W tym celu wystarczy zmienić hierarchię DOM, a jako że wspomniane połączenie działa na bieżąco, zmiany hierarchii natychmiast stają się widoczne w przeglądarce. W tabeli 28.7 znajduje się opis właściwości i metod służących do zmiany hierarchii DOM.

Tabela 28.7. Składowe służące do manipulacji strukturą DOM

Składnik	Opis	Zwracana wartość
<code>appendChild(<i>HTML</i>Element)</code>	Dodaje podanemu elementowi dziecko określonego typu.	<code>HTML</code> Element
<code>cloneNode(<i>boolean</i>)</code>	Kopiuje element.	<code>HTML</code> Element
<code>compareDocumentPosition(<i>HTML</i>Element)</code>	Określa względne położenie elementu.	<i>liczba</i>
<code>innerHTML</code>	Zwraca lub definiuje treść elementu.	<i>tekst</i>
<code>insertAdjacentHTML(<i><położenie></i>, <i><treść></i>)</code>	Zamieszcza kod HTML w miejscu określonym względem elementu.	<code>void</code>
<code>insertBefore(<i><nowyElement></i>, <i><element></i>)</code>	Zamieszcza podany element przed określonym elementem-dzieckiem.	<code>HTML</code> Element
<code>isEqualNode(<i><HTML</i>Element)</code>	Określa, czy podany element jest równy bieżącemu elementowi.	<i>wartość_boolowska</i>
<code>isSameNode(<i>HTML</i>Element)</code>	Określa, czy podany element jest tożsamy z bieżącym elementem.	<i>wartość_boolowska</i>
<code>outerHTML</code>	Zwraca lub definiuje kod HTML i zawartość elementu.	<i>tekst</i>
<code>removeChild(<i>HTML</i>Element)</code>	Usuwa określone dziecko bieżącego elementu.	<code>HTML</code> Element
<code>replaceChild(<i>HTML</i>Element, <i>HTML</i>Element)</code>	Zastępuje dziecko bieżącego elementu.	<code>HTML</code> Element

Przedstawione tutaj właściwości i metody obsługiwane są przez wszystkie obiekty elementów. Ponadto, obiekt `document` obsługuje dwie metody służące do tworzenia nowych elementów. Funkcja ta jest nieodzowna, kiedy chce się dodać treść do dokumentu. Opis tych metod znajduje się w tabeli 28.8.

Tabela 28.8. Składowe służące do manipulacji strukturą DOM

Składowa	Opis	Zwraca
<code>createElement(<i><znacznik></i>)</code>	Tworzy nowy obiekt <code>HTML</code> Element określonego typu.	<code>HTML</code> Element
<code>createTextNode(<i><treść></i>)</code>	Tworzy nowy obiekt <code>Text</code> o podanej treści.	<code>Text</code>

Tworzenie i usuwanie elementów

Nowe elementy tworzy się przy użyciu obiektu `document`, a następnie wprowadza je poprzez odnalezienie istniejącego `HTML`Element i wykorzystanie jednej z opisanych powyżej metod. W listingu 28.10 znajduje się przykład zastosowania tej funkcji.

Listing 28.10. Tworzenie i usuwanie elementów

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;
        border-collapse: collapse;
        margin: 10px;
      }
      td { padding: 4px 5px; }
    </style>
  </head>
  <body>
    <table border="1">
      <thead><th>Nazwa</th><th>Kolor</th></thead>
      <tbody id="fruitsBody">
        <tr><td>Banan</td><td>Żółty</td></tr>
        <tr><td>Jabłko</td><td>Czerwony/Zielony</td></tr>
      </tbody>
    </table>

    <button id="add">Dodaj element</button>
    <button id="remove">Usuń element</button>

    <script>
      var tableBody = document.getElementById("fruitsBody");

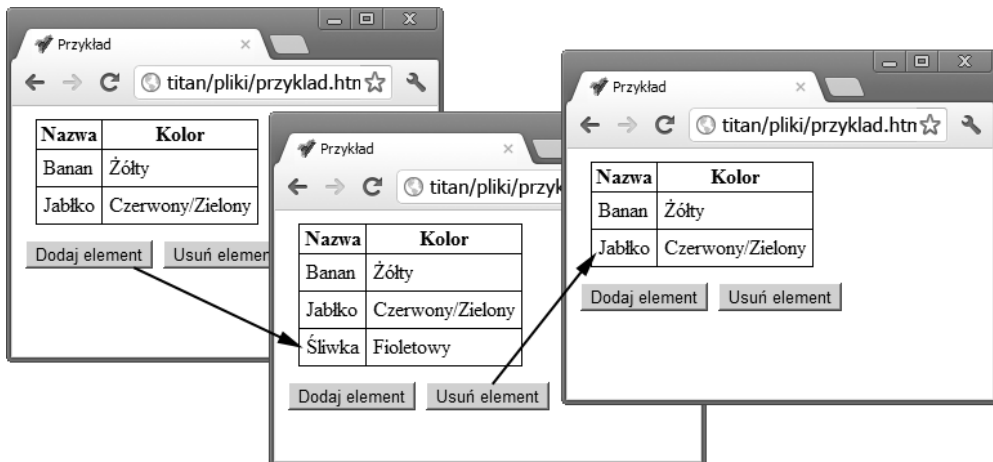
      document.getElementById("add").onclick = function() {
        var row = tableBody.appendChild(document.createElement("tr"));
        row.setAttribute("id", "newrow");
        row.appendChild(document.createElement("td"))
          .appendChild(document.createTextNode("Śliwka"));
        row.appendChild(document.createElement("td"))
          .appendChild(document.createTextNode("Fioletowy"));
      };

      document.getElementById("remove").onclick = function() {
        var row = document.getElementById("newrow");
        row.parentNode.removeChild(row);
      }
    </script>
  </body>
</html>
```

Podany w tym przykładzie skrypt wykorzystuje DOM przy dodawaniu i usuwaniu rzędów w elemencie `table` (o którym pisałem w rozdziale 11.). Dodawanie rzędu zaczynam od utworzenia elementu `tr`, którego

następnie używam jako rodzica obiektów `td` i `Text`. Zauważ, że wykorzystałem rezultaty metody do utworzenia łańcucha przywołań i odrobinę uprościłem kod.

Jak widzisz, tworzenie elementów jest pracochłonne. Trzeba utworzyć element, powiązać go z rodzicem, a następnie powtórzyć proces przy każdym elemencie-dziecku bądź obiekcie `Text`. Usuwanie elementów również jest mało poręczne. W tym celu trzeba znaleźć element, przejść do elementu-rodzica, a następnie użyć metody `removeChild`. Rezultat zastosowania tego skryptu widać na rysunku 28.9.



Rysunek 28.9. Wykorzystanie DOM do tworzenia i usuwania elementów

Powielanie elementów

Metoda `cloneNode` służy do powielania istniejących elementów. Jest ona wygodna o tyle, że pozwala na uniknięcie tworzenia elementów od podstaw. W listingu 28.11 znajduje się przykład zastosowania tej techniki.

Listing 28.11. Powielanie elementów

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;
        border-collapse: collapse;
        margin: 10px;
      }
      td { padding: 4px 5px; }
    </style>
  </head>
  <body>
    <table border="1">
      <thead><tr><th>Pomóż</th><th>Wynik</th></tr></thead>
      <tbody id="fruitsBody">
        <tr><td class="sum">1 x 1</td><td class="result">1</td></tr>
      </tbody>
    </table>
```

```

</table>

<button id="add">Dodaj rząd</button>
<script>
  var tableBody = document.getElementById("fruitsBody");

  document.getElementById("add").onclick = function() {
    var count = tableBody.getElementsByTagName("tr").length + 1;

    var newElem = tableBody.getElementsByTagName("tr")[0].cloneNode(true);
    newElem.getElementsByClassName("sum")[0].firstChild.data = count
      + " + " + count;
    newElem.getElementsByClassName("result")[0].firstChild.data =
      count * count;

    tableBody.appendChild(newElem);
  };
</script>
</body>
</html>

```

W tym przykładzie powieliłem znajdujący się w tabeli rząd. Boolowski argument metody `cloneNode` określa, czy elementy-dzieci również mają być powielone. W tym przypadku podałem wartość `true`, ponieważ chciałem, by elementy `td` zawarte w elemencie `tr` utworzyły strukturę nowego rzędu.

-
- **Wskazówka** Zauważ, jak niezręcznie wygląda określanie tekstu komórek tabeli w tym przykładzie. Praca z obiektami `Text` jest naprawdę męcząca. Z prostszym podejściem zapoznasz się w punkcie „Praca z fragmentami HTML” w dalszej części tego rozdziału.
-

Przenoszenie elementów

Przenoszenie elementów z jednej części dokumentu do drugiej polega zwyczajnie na powiązaniu elementu z jego nowym rodzicem. Nie musisz przenosić elementu z jego pozycji wyjściowej. W listingu 28.12 widzicie przykład przeniesienia rzędu z jednej tabeli do drugiej.

Listing 28.12. Przenoszenie elementów

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;
        border-collapse: collapse;
        margin: 10px;
        float: left;
      }
      td { padding: 4px 5px; }
      p { clear:left; }
    </style>
  </head>

```

```

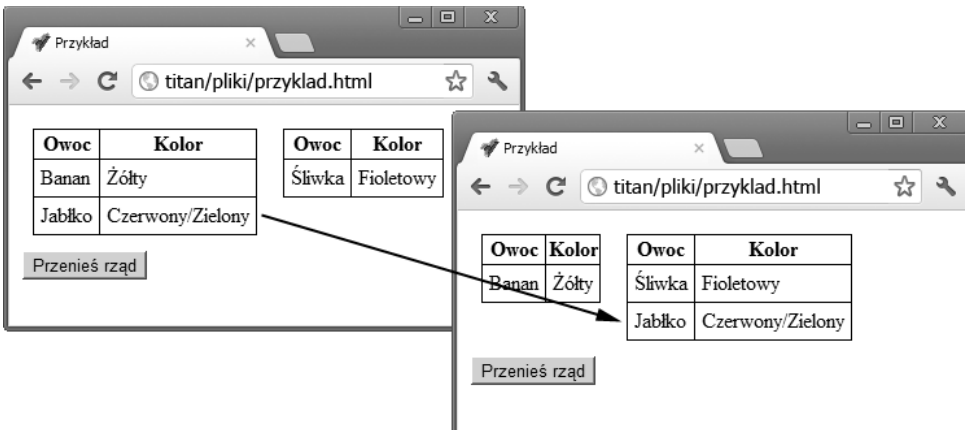
<body>
  <table border="1">
    <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
    <tbody>
      <tr><td>Banan</td><td>Żółty</td></tr>
      <tr id="apple"><td>Jabłko</td><td>Czerwony/Zielony</td></tr>
    </tbody>
  </table>

  <table border="1">
    <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
    <tbody id="fruitsBody">
      <tr><td>Śliwka</td><td>Fioletowy</td></tr>
    </tbody>
  </table>

  <p>
    <button id="move">Przenieś rząd</button>
  </p>
  <script>
    document.getElementById("move").onclick = function() {
      var elem = document.getElementById("apple");
      document.getElementById("fruitsBody").appendChild(elem);
    };
  </script>
</body>
</html>

```

Po naciśnięciu przycisku skrypt przenosi element `tr` z atrybutem `id` o wartości `apple` i używa metody `appendChild` na elemencie `tbody` o atrybucie `id` o wartości `fruitsBody`. Rezultatem jest przeniesienie rzędu z jednej tabeli do innej, co widać na rysunku 28.10.



Rysunek 28.10. Przeniesienie elementu z jednej części dokumentu do innej

Porównywanie obiektów elementów

Obiekty elementów można porównywać na dwa sposoby. Po pierwsze, wystarczy sprawdzić, czy reprezentują ten sam element, co robi się przy użyciu metody `isSameNode`. Pozwala to na porównanie obiektów, które zostały pozyskane w różnych kwerendach, tak jak widać to w listingu 28.13.

Listing 28.13. Porównanie obiektów elementów

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;
        border-collapse: collapse;
      }
      td { padding: 4px 5px; }
    </style>
  </head>
  <body>
    <table border="1">
      <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
      <tbody id="fruitsBody">
        <tr id="plumrow"><td>Śliwka</td><td>Fioletowy</td></tr>
      </tbody>
    </table>
    <pre id="results"></pre>
    <script>

      var elemByID = document.getElementById("plumrow");
      var elemByPos
        = document.getElementById("fruitsBody").getElementsByTagName("tr")[0];

      if (elemByID.isSameNode(elemByPos)) {
        document.getElementById("results").innerHTML = "Obiekty są tożsame";
      }

    </script>
  </body>
</html>

```

Podany w tym przykładzie skrypt znajduje obiekty elementów na dwa różne sposoby: szukając konkretnego id oraz przeszukując element-rodzica według rodzaju znacznika. Metoda `isSameNode` zwraca wartość `true` przy porównywaniu tych obiektów, ponieważ reprezentują ten sam element.

Zamiast tego można też sprawdzić, czy obiekty są sobie równe, co robi się metodą `isEqualNode`. Elementy są równe, kiedy są jednakowego typu, mają te same wartości atrybutów, a także wszystkie ich dzieci są równe i występują w tej samej kolejności. W listingu 28.14 znajduje się przykład pary równych elementów.

Listing 28.14. Praca z równymi elementami

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;
        border-collapse: collapse;

```

```

        margin: 2px 0px;
    }
    td { padding: 4px 5px; }
</style>
</head>
<body>
  <table border="1">
    <thead><tr><th>0woc</th><th>Kolor</th></tr></thead>
    <tbody>
      <tr class="plumrow"><td>Śliwka</td><td>Fioletowy</td></tr>
    </tbody>
  </table>

  <table border="1">
    <thead><tr><th>0woc</th><th>Kolor</th></tr></thead>
    <tbody>
      <tr class="plumrow"><td>Śliwka</td><td>Fioletowy</td></tr>
    </tbody>
  </table>

  <pre id="results"></pre>
  <script>
    var elems = document.getElementsByClassName("plumrow");

    if (elems[0].isEqualNode(elems[1])) {
      document.getElementById("results").innerHTML = "Elementy są równe";
    } else {
      document.getElementById("results").innerHTML = "Elementy NIE SĄ równe";
    }
  </script>
</body>
</html>

```

W tym przykładzie obydwa elementy tr są równe, pomimo że są osobnymi elementami w różnych częściach dokumentu. Gdybym zmienił którykolwiek z atrybutów lub treść elementu-dziecka td, elementy nie byłyby już równe.

Praca z fragmentami HTML

Właściwości innerHTML i outerHTML wraz z metodą insertAdjacentHTML są wygodnymi skrótami składniowymi, które pozwalają na pracę z fragmentami kodu HTML. Dzięki temu można uniknąć tworzenia rozbudowanych hierarchii składających się z obiektów elementów i obiektów tekstowych. W listingu 28.15 widnieje przykład zastosowania właściwości innerHTML i outerHTML do pobierania kodu HTML z elementów.

Listing 28.15. Zastosowanie właściwości innerHTML i outerHTML

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;

```

```

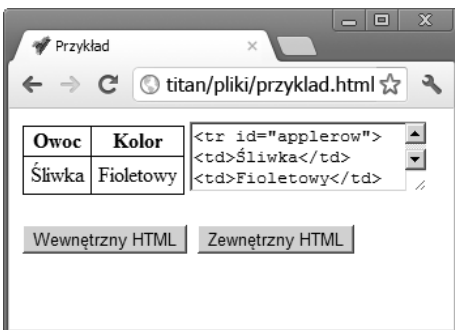
border-collapse: collapse;
margin: 5px 2px;
float: left;
}
td { padding: 4px 5px; }
p {clear: left};
</style>
</head>
<body>
<table border="1">
  <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
  <tbody>
    <tr id="applerow"><td>Śliwka</td><td>Fioletowy</td></tr>
  </tbody>
</table>
<textarea rows="3" id="results"></textarea>
<p>
  <button id="inner">Wewnętrzny HTML</button>
  <button id="outer">Zewnętrzny HTML</button>
</p>
<script>
var results = document.getElementById("results");
var row = document.getElementById("applerow");

document.getElementById("inner").onclick = function() {
  results.innerHTML = row.innerHTML;
};

document.getElementById("outer").onclick = function() {
  results.innerHTML = row.outerHTML;
}
</script>
</body>
</html>

```

Właściwość `outerHTML` zwraca ciąg znaków, w którym znajduje się definiujący element kod HTML oraz kod wszystkich jego dzieci. Właściwość `innerHTML` zwraca kod HTML samych dzieci. W tym przykładzie utworzyłem dwa przyciski, które służą do wyświetlania „wewnętrznego” i „zewnętrznego” kodu HTML rzędu tabeli. Treść wyświetlana jest w elemencie `textarea`, tak by przeglądarka traktowała zwracane przez te właściwości ciągi znaków jako tekst, a nie kod HTML dokumentu. Rezultat zastosowania skryptu widać na rysunku 28.11.



Rysunek 28.11. Wyświetlanie właściwości `outerHTML` rzędu tabeli

Zmiana struktury dokumentu

Właściwości `outerHTML` i `innerHTML` używa się także do zmiany struktury dokumentu. Właściwości `innerHTML` użyłem w wielu przykładach przedstawionych w tej części książki. Jest to wygodny sposób ustawiania zawartości elementów, ponieważ umożliwia podawanie treści tekstowej bez konieczności tworzenia elementów `Text`. W listingu 28.16 widzimy przykład zastosowania tych właściwości w ramach modyfikowania modelu dokumentu.

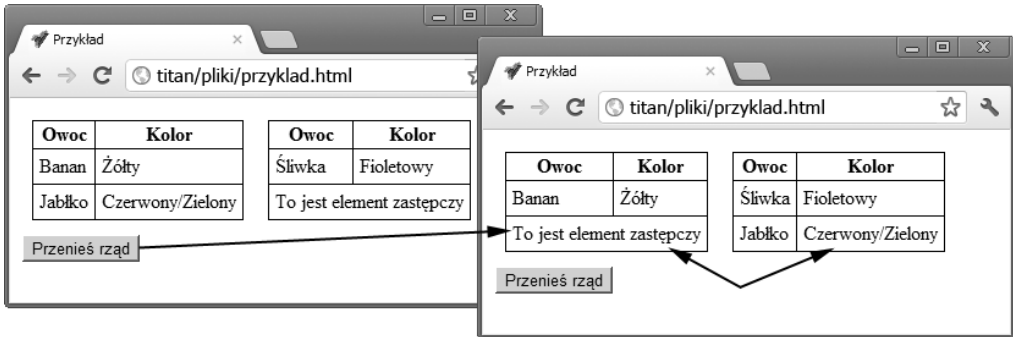
Listing 28.16. Modyfikacja modelu dokumentu

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
    <style>
      table {
        border: solid thin black;
        border-collapse: collapse;
        margin: 10px;
        float: left;
      }
      td { padding: 4px 5px; }
      p { clear: left; }
    </style>
  </head>
  <body>
    <table border="1">
      <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
      <tbody>
        <tr><td>Banan</td><td>Żółty</td></tr>
        <tr id="apple"><td>Jabłko</td><td>Czerwony/Zielony</td></tr>
      </tbody>
    </table>

    <table border="1">
      <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
      <tbody id="fruitsBody">
        <tr><td>Śliwka</td><td>Fioletowy</td></tr>
        <tr id="targetrow"><td colspan="2">To jest element zastępczy</td></tr>
      </tbody>
    </table>

    <p>
      <button id="move">Przenieś rząd</button>
    </p>
    <script>
      document.getElementById("move").onclick = function() {
        var source = document.getElementById("apple");
        var target = document.getElementById("targetrow");
        target.innerHTML = source.innerHTML;
        source.outerHTML = '<tr id="targetrow"><td colspan="2">' +
          'To jest element zastępczy</td>';
      };
    </script>
  </body>
</html>
```


W tym przykładzie użyłem właściwości `innerHTML` do zdefiniowania elementów-dzieci rzędu tabeli oraz `outerHTML` do zastąpienia elementu w wierszu. Właściwości te działają, opierając się na ciągach znaków, co oznacza, że fragmenty HTML pozyskuje się przez odczytanie wartości właściwości lub tworzenie ciągów znakowych od podstaw, tak jak widać to w listingu. Rezultat widać na rysunku 28.12.



Rysunek 28.12. Zastosowanie właściwości `innerHTML` i `outerHTML`

Wstawianie fragmentów HTML

Właściwości `innerHTML` i `outerHTML` przydają się przy zastępowaniu istniejących elementów, ale jeśli chcesz użyć fragmentu HTML do wprowadzenia nowych elementów, to musisz skorzystać z metody `insertAdjacentHTML`. Metoda ta używa dwóch argumentów — pierwszy to wartość z tabeli 28.9, która wskazuje, gdzie fragment powinien być wstawiony względem bieżącego dokumentu, a drugi to właśnie fragment, który ma być wstawiony.

Tabela 28.9. Wartości położenia metody `insertAdjacentHTML`

Wartość	Opis
<code>afterbegin</code>	Wstawia fragment jako pierwsze dziecko bieżącego elementu.
<code>afterend</code>	Wstawia fragment bezpośrednio po bieżącym elemencie.
<code>beforebegin</code>	Wstawia fragment bezpośrednio przed bieżącym elementem.
<code>beforeend</code>	Wstawia fragment jako ostatnie dziecko bieżącego elementu.

W listingu 28.17 widnieje przykład wykorzystania metody `insertAdjacentHTML` przy wstawianiu fragmentów kodu HTML do rzędu tabeli i dookoła niego.

Listing 28.17. Zastosowanie metody `insertAdjacentHTML`

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
  </head>
  <body>
    <table border="1">
      <thead><tr><th>Owoc</th><th>Kolor</th></tr></thead>
      <tbody id="fruitsBody">
        <tr id="targetrow"><td>To jest element zastępczy</td></tr>
      </tbody>
    </table>
  </body>

```

```

</table>

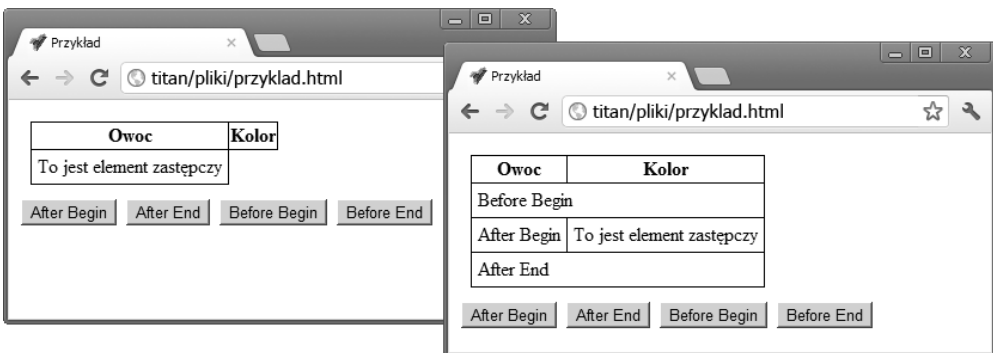
<p>
  <button id="ab">After Begin</button>
  <button id="ae">After End</button>
  <button id="bb">Before Begin</button>
  <button id="be">Before End</button>
</p>
<script>
  var target = document.getElementById("targetrow");
  var buttons = document.getElementsByTagName("button");
  for (var i = 0; i < buttons.length; i++) {
    buttons[i].onclick = handleButtonPress;
  }

  function handleButtonPress(e) {
    if (e.target.id == "ab") {
      target.insertAdjacentHTML("afterbegin", "<td>After Begin</td>");
    } else if (e.target.id == "be") {
      target.insertAdjacentHTML("beforeend", "<td>Before End</td>");
    } else if (e.target.id == "bb") {
      target.insertAdjacentHTML("beforebegin",
        "<tr><td colspan='2'>Before Begin</td></tr>");
    } else {
      target.insertAdjacentHTML("afterend",
        "<tr><td colspan='2'>After End</td></tr>");
    }
  }
}

</script>
</body>
</html>

```

W tym przykładzie użyłem różnych wartości położenia, by zaprezentować zamieszczanie fragmentów HTML w różnych miejscach. Najlepiej poeksperymentować z nimi w przeglądarce, aczkolwiek ogólny efekt widać na rysunku 28.13.



Rysunek 28.13. Wstawianie fragmentów HTML do dokumentu

Wstawienie elementu do bloku tekstu

Istotną wariacją modyfikacji modelu jest możliwość dodania elementu do bloku tekstu reprezentowanego obiektem `Text`. W listingu 28.18 widzicie tego przykład.

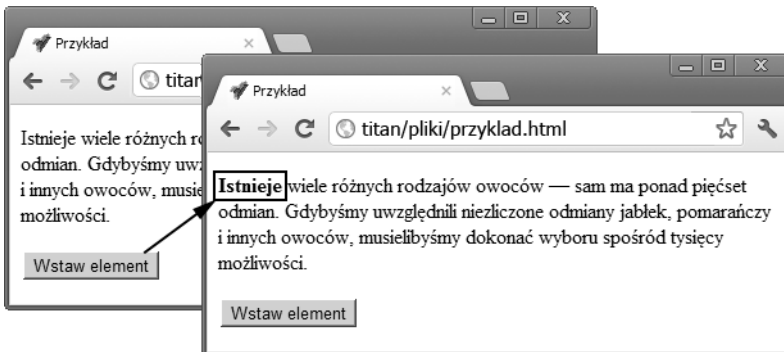
Listing 28.18. Wstawienie elementu do bloku tekstu

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Przykład</title>
    <meta name="author" content="Adam Freeman"/>
    <meta name="description" content="Prosty przykład"/>
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon" />
  </head>
  <body>
    <p id="textblock">
      Istnieje wiele różnych rodzajów owoców – sam ma ponad pięćset odmian. Gdybyśmy
      uwzględnili niezliczone odmiany jabłek, pomarańczy i innych owoców, musielibyśmy
      dokonać wyboru spośród tysięcy możliwości.
    </p>
    <p>
      <button id="insert">Wstaw element</button>
    </p>
    <script>
      document.getElementById("insert").onclick = function() {
        var textBlock = document.getElementById("textblock");
        textBlock.firstChild.splitText(10);
        var newText = textBlock.childNodes[1].splitText(8).previousSibling;
        textBlock.insertBefore(document.createElement("b"),
          newText).appendChild(newText);
      }
    </script>
  </body>
</html>

```

W tym przykładzie wykonałem niebanalne zadanie, jakim jest przeniesienie wyrazu z istniejącego bloku tekstowego i zrobienie z niego dziecka nowego elementu b. Podobnie jak w poprzednich przykładach, praca z modelem wiąże się z tworzeniem dość rozbudowanego kodu. Rezultat widać na rysunku 28.14.



Rysunek 28.14. Zamieszczenie elementu w bloku tekstu

Podsumowanie

W tym rozdziale przedstawiłem funkcje obiektów `HTMLElement` i `Text`, które — kolejno — reprezentują elementy i treść w dokumentach HTML. Dowiedziałeś się, jak pobierać informacje o elementach z obiektów, jak pracować z treścią tekstową oraz jak wykorzystywać możliwości DOM, by dodawać, modyfikować,

powielać, przenosić i usuwać elementy. Praca z DOM może wymagać tworzenia rozbudowanych skryptów, ale aktywne połączenie między modelem obiektowym a wyglądem dokumentu sprawia, że jest to warte wysiłku.

Skorowidz

A

- ActiveState, 36
- Adobe Flash, *Patrz:* Flash
- adres
 - e-mailowy, 322
 - IP, 815
 - URL, 131, 134, 154, 169, 179, 322, 326, 328, 329, 389, 394, 553, 665, 694, 696
 - względny, 130, 328
- Ajax, 685, 686, 709, 718
- akapit, 185, 191
- animacja, 489, 493, 496
 - deklaracja, 496
 - etap pośredni, 499
 - kierunek powtórzeń, 501
 - layoutu, 503
 - stan końcowy, 502
 - stan początkowy, 498
 - wielu elementów, 505
 - właściwość, 353
 - wstrzymywanie, 507
 - wznawianie, 507
 - z przekształceniem, 511
- API
 - REpresentational State Transfer, 698, *Patrz:* API
 - REST
 - REST, 698
- aplikacja offline, 831, 833
- Apple, 30
- argument funkcji, *Patrz:* funkcja argumenty
- arkusz stylu, 61, *Patrz też:* CSS
 - kodowanie znaków, 67
 - zewnątrzny, 65
- atak cross-site scripting, *Patrz:* XSS
- atrybut, 44
 - accept, 307
 - accesskey, 49
 - action, 258, 271
 - allowfullscreen, 334
 - alt, 305, 330
 - async, 126, 145, 147, 148
 - autocomplete, 261, 262
 - autofocus, 265, 310, 317
 - autoplay, 740
 - autorski, 45
 - boolowski, 44, 46, 53, 180
 - border, 249, 251
 - charset, 143
 - checked, 293
 - circle, 330
 - cite, 169, 179, 189
 - class, 50, 176, 186, 188, 358
 - cols, 314
 - colspan, 238, 239
 - content, 132
 - contenteditable, 52
 - contextmenu, 53
 - controls, 740
 - coords, 329
 - datetime, 179, 180
 - default, 330
 - defer, 126, 143, 145, 147, 556
 - dir, 53, 173
 - dirname, 279, 285
 - disabled, 266, 284, 285, 310, 317
 - draggable, 53, 799
 - dropzone, 53
 - elementu, 596, 598
 - enctype, 259, 271
 - form, 310, 317, 337
 - formation, 272, 305
 - formenctype, 305
 - formmethod, 272, 305
 - formnovalidate, 305, 323
 - formtarget, 305

atrybut

globalny, 44, 49, 62, 186, 243, 359, 596
 headers, 242, 243
 height, 305, 326, 327, 332, 334, 735
 hidden, 53
 high, 341
 href, 44, 130, 131, 151, 153, 154, 327, 330, 665
 hreflang, 153, 330
 http-equiv, 125, 134
 id, 55, 186, 265, 394
 ismap, 327
 keytype, 317
 klasy, 71
 label, 283, 313
 lang, 55, 394
 list, 279, 281, 290, 297, 299
 lokalny, 44, 49, 153
 low, 341
 manifest, 834
 max, 290, 299, 318, 319, 320, 340
 maxlength, 278, 279, 286, 297
 media, 136, 153, 330, 738
 method, 259, 271
 min, 290, 299, 318, 319, 320
 multiple, 307, 310, 311
 name, 132, 263, 310, 317, 332, 338
 niestandardowy, 597
 novalidate, 323
 open, 226
 optimum, 341, 342
 pattern, 279, 286, 297, 318, 321
 placeholder, 279, 280, 281, 286, 297
 poly, 330
 poster, 734, 735
 preload, 732, 734
 pubdate, 180
 readonly, 279, 284, 286, 290, 297, 299
 rect, 330
 rel, 125, 142, 153, 330
 required, 279, 286, 290, 293, 297, 299, 307, 310, 318, 319, 320, 321, 387
 reversed, 195
 rows, 314
 rowspan, 238, 239, 240
 sandbox, 333
 scoped, 135
 seamless, 333
 shape, 329
 size, 278, 279, 280, 286, 297, 310, 311
 span, 246, 249
 spellcheck, 56
 src, 126, 143, 305, 326, 332, 334, 736, 739
 srcdoc, 332
 start, 194, 200
 step, 290, 299
 style, 57, 62, 63, 360, 615

tabindex, 57
 target, 131, 153, 155, 262, 271, 330, 332, 338
 td, 242
 th, 242
 title, 58, 168, 169
 type, 135, 143, 144, 153, 194, 256, 270, 271, 272, 273, 277, 278, 285, 289, 290, 296, 330, 334
 usemap, 331, 338
 value, 197, 200, 279, 280, 281, 286, 290, 293, 297, 299, 319, 340, 341
 width, 305, 326, 327, 332, 334, 735
 wrap, 314
 wszystkie, 598
 audio, 32, 731, 739
 odtwarzanie, 745
 autouzupelnianie, 261, 262, 282

B

barwa, *Patrz:* kolor
 biblioteka, 144
 JavaScript, 33, 575
 Modernizr, *Patrz:* Modernizr
 blog, 220
 blok kodu, 90
 Blueprint, 85, 449
 błąd, 694
 aplikacji, 697
 geolokalizacji, 816
 wejściowy, 696
 żądania, 696
 błędy, 106, 111
 bufora offline, 843
 bracia, 47
 brat, 368
 bufor offline, 831, 839

C

Cascading Style Sheets, *Patrz:* CSS
 Chrome, 33, 35, 68, 210, 211, 214, 261, 280, 291, 300, 301, 319, 338, 386, 387, 399, 493, 574, 622, 691, 715, 731, 744, 809, 815, 816, 834
 ciasteczko, *Patrz:* cookie
 cień, 397, 416, 481, 792
 cookie, 555
 CORS, 703, 704
 cross-document messaging, *Patrz:* przekazywanie komunikatów między dokumentami
 CSS, 31, 41, 45, 47, 48, 61, 67, 116, 134, 345, 616
 deklaracja, *Patrz:* styl deklaracja
 framework, 84
 narzędzia, 83
 selektor, 55
 typu inline, 47

CSS3, 345
 cytat, 167, 171, 189
 czcionka, 482, 484, 485, *Patrz też:* font
 internetowa, 486
 czytnik ekranowy, 242

D

dane, 663, 709
 elementu, 741
 JSON, 728
 kodowanie, 259, 260
 kontaktowe, 224
 magazynowanie, 823
 pobieranie, 726
 poufne, 304
 przesyłanie, 710
 ukryte, 303
 wejściowe, 167
 weryfikacja, 317, 318, 385
 wyjściowe, 167
 wysyłanie, 713, 715
 data, 180, 298
 definicja, 167, 169, 197, 198
 deklaracja, 89
 charset, 67
 for...in, 96
 if z operatorem negacji, 110
 import, 66
 warunkowa, 100
 długość, 75
 DNS, 694
 Document Object Model, *Patrz:* DOM
 Dojo, 112
 dokument, 125, 126, 531, 548, 551, 556, 654
 drzewo, 48
 HTML, *Patrz:* HTML dokument
 nazwa, 129
 osadzony, 585
 pobieranie, 723
 sekcja, *Patrz:* sekcja
 struktura, 610
 tytuł, 129
 ustrukturyzowany, 561
 DOM, 263, 529, 531, 541, 547, 602, 643, 663, 740
 drzewo, 565
 implementacja, 557
 nawigacja, 565
 poziom, 533
 testowanie, 533
 dopełnienie, 403, 424
 dziecko, 47, 48, 346, 366
 n-te, 381
 dziedziczenie, 67, 73
 dzielenie wyrazów, 476

E

edytor
 Komodo Edit, *Patrz:* Komodo Edit
 tekstu, 36
 efekty, 792, *Patrz też:* cień, przezroczystość
 elastyczny layout polowy, *Patrz:* flexbox
 elastyczny model polowy, *Patrz:* flexbox
 element, 40, 115, 116, 125
 a, 42, 118, 128, 131, 151, 152, 154, 161, 217, 327, 338, 665
 abbr, 118, 152, 168, 169, 667
 address, 120, 223, 224, 668
 aktywny, 382
 area, 122, 329, 330, 678
 article, 115, 117, 120, 180, 187, 220, 224, 668
 aside, 120, 220, 668
 atrybut, 596, *Patrz:* atrybut
 atrybuty wszystkie, 598
 audio, 122, 739, 740
 b, 115, 118, 151, 156, 667
 base, 118, 125, 130, 131, 154, 663
 bdi, 152, 174
 bdo, 152, 173
 blockquote, 120, 189, 191, 667
 blokowy, 435, 437
 body, 42, 43, 47, 118, 125, 128, 184, 186, 224, 663
 br, 118, 152, 164, 667
 button, 42, 121, 254, 256, 270, 272, 273, 274, 289, 323,
 341, 444, 687, 714
 canvas, 32, 122, 342, 749, 750, 752, 765, 772, 775
 caption, 121, 243, 244
 checkbox, 294
 cite, 116, 119, 151, 152, 171, 667
 code, 40, 42, 119, 128, 146, 152, 167, 188, 202, 667
 col, 245, 247, 248, 249, 668
 colgroup, 121, 245, 246, 247, 668
 command, 123
 dane, 592, 741
 datalist, 121, 281, 282, 283, 671
 dd, 120, 197, 198, 668
 default, 384
 del, 119, 152, 178, 179, 666
 details, 120, 224, 226, 668
 dfn, 152, 169, 667
 div, 116, 117, 120, 186, 187, 188, 687
 dl, 120, 197
 doctype, 125, 126
 DOCTYPE, 42, 46, 118
 domyślny, 384
 dopełnienie, 346
 dt, 120, 197, 198, 668
 elastyczny, 460
 em, 119, 151, 157, 667
 embed, 122, 333, 334, 678
 fieldset, 122, 267, 268, 269, 671

element

- figcaption, 120, 201, 202, 668
- figure, 115, 120, 200, 202, 668
- fokusowanie, *Patrz:* fokusowanie
- footer, 120, 211, 668
- form, 122, 254, 255, 261, 262, 272, 274, 323, 338, 660, 672, 710, 720
- formularza, 671, 682
- główny, 127, 376
- grupujący, 43, 191, 667, 668
- h1, 204, 205
- h1 – h6, 120, 204, 211, 668
- head, 42, 46, 47, 118, 125, 127, 128, 130, 142, 665
- header, 120, 211, 217, 668
- hgroup, 116, 121, 205, 208, 668
- hr, 42, 43, 120, 191
- html, 42, 43, 46, 118, 125, 127, 665
- HTMLCollection, 560
- i, 119, 151, 158, 667
- identyfikator, 55
- iframe, 122, 131, 331, 332, 333, 338, 679, 720
- img, 122, 154, 326, 327, 328, 337, 560, 679, 766, 769
- input, 42, 121, 122, 254, 255, 259, 265, 274, 277, 283, 285, 287, 289, 290, 291, 292, 295, 298, 300, 302, 305, 307, 323, 385, 386, 672, 710
- dezaktywacja, 266, 269
- ins, 119, 152, 178, 179, 666
- interaktywny, 391
- kbd, 119, 152, 167, 667
- keygen, 122, 309, 316
- kliknięcie, 639
- kolejność rysowania, 452
- krawędź, 399, 424, 447
- label, 42, 122, 264, 265, 674
- legend, 122, 268, 674
- li, 48, 120, 193, 196, 200, 217, 667
- liniowo-blokowy, 439
- liniowy, 437
- link, 125, 139, 140, 141, 664
- map, 122, 329, 331, 680
- margines, 346
- mark, 119, 152, 177, 667
- menu, 123, 196
- meta, 118, 125, 131, 133, 150, 664
- meter, 122, 341, 680
- nav, 121, 211, 215, 668
- nazwany, 560, 561
- nieaktywny, 382
- nietabelowy, 435
- noscript, 118, 126, 142, 148, 150, 665
- obiektu Document, 559
- object, 122, 333, 334, 337, 338, 680
- obramowanie, 346, 356, 397, 398, 399, 401, 407, 418, 426
- obrazkowe, 404
- specjalne, 404
- z zaokrąglonymi rogami, 402
- obrys, 418
- odstęp, 445
- ogólny, 116
- ol, 117, 120, 193, 200, 667
- opis, 117
- optgroup, 122, 309, 312, 313, 674
- option, 122, 282, 283, 310, 311, 313, 675
- osadzające treść w dokumencie, 678
- output, 122, 309, 316, 675
- p, 42, 120, 128, 185, 188, 191, 265, 437
- param, 122, 681
- password, 294
- pole, 423, *Patrz:* pole elementu
- porównywanie, 606
- powielanie, 604
- pozycjonowanie, *Patrz:* pozycjonowanie
- pre, 120, 188
- progress, 123, 339, 340, 723
- przenoszenie, 605
- pusty, 42
- q, 119, 152, 169, 189, 666, 667
- rozmiar, 427, 430
- rp, 119, 152, 172, 173, 667
- rt, 119, 152, 172, 173, 667
- ruby, 119, 152, 172, 667
- run-in, 439
- s, 119, 151, 159, 667
- samp, 119, 152, 167, 667
- script, 52, 88, 92, 118, 126, 142, 143, 144, 146, 147, 148, 664
- section, 121, 187, 209, 211, 668
- sekcji, 668
- select, 122, 309, 311, 312, 676
- semantyczny, 41
- sfokusowany, 391
- skryptowy, 142
- small, 119, 151, 162, 667
- source, 123, 737
- span, 116, 119, 152, 176, 187, 383, 437, 491, 667
- strong, 119, 151, 160, 667
- strukturalny, 48
- styl, 623, 624
- style, 42, 63, 64, 118, 125, 134, 135, 217, 615, 665
- sub, 119, 151, 163, 667
- summary, 121, 225, 668
- sup, 119, 151, 163, 667
- svg, 123
- tabel, 668, 670, 682
- tabelowy, 435
- table, 42, 121, 229, 231, 235, 236, 241, 243, 244, 249, 251, 465, 617, 669
- tbody, 121, 235, 236, 669
- td, 42, 121, 229, 232, 238
- tekstowy, 151, 156, 176, 665
- Text, 600
- textarea, 42, 56, 122, 278, 309, 313, 677

tfoot, 121, 235, 236, 247, 669
 th, 42, 121, 232, 234, 238, 243, 669
 thead, 121, 235, 236, 247, 669
 time, 119, 152, 180, 666
 title, 42, 47, 118, 128, 129, 665
 tło, 409, 410, 411, 412, 413, 415
 tr, 42, 121, 229, 231, 670
 track, 123, 738
 treściowy, 48
 treść, 430
 tworzenie, 603, 604
 typ, 48, 357
 u, 119, 151, 161, 667
 ukrywanie, 442
 ul, 120, 195, 199, 200, 217
 usuwanie, 603, 604
 var, 119, 152, 167, 667
 video, 123, 732, 740, 769, 770
 void, 43
 wbr, 119, 152, 164, 165, 166, 667
 weryfikacja, 385
 widoczność, 434
 wstawianie do bloku tekstu, 612
 wyszukiwanie, 563
 z ograniczeniami, 48
 zawartość, 346
 zaznaczony, 383
 encja, 48, 260

F

favikona, 141
 film, *Patrz:* wideo
 Firebug, 111
 Firefox, 33, 35, 111, 142, 210, 211, 214, 280, 291, 399, 493,
 574, 622, 691, 715, 744, 765, 805, 816, 839
 Flash, 30, 32, 720, 731, 749
 flexbox, 449, 456, 457, 458, 459
 wielkość maksymalna, 463
 flow, *Patrz:* element strukturalny
 fokusowanie, 639, 657
 font, 77, 482, 484, 485, *Patrz też:* czcionka
 internetowy, 486
 format ico, 141
 formularz, 121, 253, 255, 305, 309, 337, 386, 660, 671, 682,
 709, 710, 712, 715
 etykieta, 263, 264
 grupowanie elementów, 267
 konfiguracja, 258
 framework CSS, 449
 funkcja, 32, 90, 94, 644, *Patrz też:* metoda
 analizy stylów, 83
 argumenty, 91
 collapse, 435
 displayErrorMsg, 696
 displayMsg, 589

displayPosition, 814
 geolokalizacji, 813
 handleButtonPress, 688, 714
 handleDescendantEvent, 648
 handleResponse, 689
 HTMLElement, 532
 JavaScript, 639
 listClasses, 595
 marquee, 433
 new Array, 104
 Number, 103
 parametry, 90
 parseFloat, 104
 parseInt, 104
 przeciągnij i upuść, 799, 800, 804, 810
 readCookies, 556
 stopImmediatePropagation, 649, 650
 stopPropagation, 649, 650
 table CSS, 241
 toggleClass, 595
 url, 407
 writeResponse, 712, 723

G

Geolocation API, 813
 geolokalizacja, 813, 815, 818, 820
 Google'a, 816
 godzina, 180, 298
 GPS, 815
 gradient
 liniowy, 757, 758, 760
 promieniowy, 757, 762
 grafika, 342
 grupowanie treści, 183, 185

H

hasło, 285, 286, 688
 wyszukiwania, 302
 host, 130, 584, 694
 HTML
 dokument, 45, 46
 standard, *Patrz:* standard HTML
 HTML5
 specyfikacja bazowa, 31
 standard, *Patrz:* standard HTML5

I

identyfikator fragmentu, 55, 394
 ilustracja, 200
 indeks
 dolny, 163
 górny, 163

indeksownik tablicowy, 561
 interakcja, 689
 niezabezpieczona, 259
 zabezpieczona, 259
 interfejs, 382, 532
 Internet Explorer, 30, 35, 210, 715, 816
 iOS, 30

J

JavaScript, 30, 32, 36, 45, 83, 87, 142, 143, 274, 444, 533
 biblioteka, 112
 debuger, 111
 dezaktywacja, 148
 narzędzia, 111
 obiekt, *Patrz:* obiekt
 operator, *Patrz:* operator
 prymityw, *Patrz:* prymityw
 tablica, *Patrz:* tablica
 JavaScript Object Notation, *Patrz:* JSON
 jednostka, 82
 bezwzględna, 76
 czasu, 82
 nieobdługiwana, 81
 obliczanie, 81
 względna, 77, 80
 jednostki, 633
 konwersja, 634
 język
 ISO, 56
 Java, 30
 skryptowy, 142
 znaczników, 34
 jQuery, 112, 144, 300, 317, 449, 533, 799
 jQuery UI, 112
 JSON, 718, 725, 726, 728

K

kalkulator, 316
 kaskadowanie, 83
 kaskadowość, 67, 69
 kaskadujące arkusze stylów, *Patrz:* CSS
 kąt, 82
 keywords, 133
 klasa, 116, 362, 593
 article, 117
 klatka
 kluczowa, 497, 498, 504
 klauzula
 procentowa, 499
 klient
 użytkownika, 45
 klucz, 823
 prywatny, 316
 publiczny, 316

kodowanie, *Patrz:* dane kodowanie
 kolor, 74, 300, 409, 417, 513, 757
 pierwszego planu, 514
 wypełnienia, 754
 Komodo Edit, 36
 kompas, 815
 kompozycja, 794
 komunikat, 586
 kontekst, 751, 765
 2d, 751
 3d, 751
 przeglądania, 131, 155, 262, 332, 338
 kontener, 346
 krzywa Béziera, 495, 785, 787
 kursor, 389, 527

L

layout, 241, 249, 251, 265, 418, 430, 437, 449
 animacja, 503
 elastyczny, 457
 siatkowy, 449
 szablonowy, 449
 tabelowy, 465
 wielokolumnowy, 454
 właściwość, 351
 LESS, 84
 liczba
 całkowita, 93
 rzeczywista, *Patrz:* liczba zmiennoprzecinkowa
 zmiennoprzecinkowa, 93
 licznik, 588, 771
 CSS, 373
 czasu, 589
 odstępów, 589
 linia, 778
 pozioma, 43
 prosta, 776
 stykająca się, 755
 szerokość, 754
 zakończenie, 778
 lista, 193, 513, 523
 definicji, 197
 klas, 593
 nieuporządkowana, 48, 195, 200
 o specyficznej numeracji, 199
 uporządkowana, 48, 193, 197
 wewnętrzna, 200
 znacznik, 523, 524, 525
 literał
 obiektowy, 94, 97
 tablicowy, 105
 logo, 211
 logogram, *Patrz:* pismo logograficzne, znak ruby

Ł

łuk, 780

M

magazyn

- danych, 823
- lokalny, 823, 826
- sesji, 823, 827

manifest, 833, 835

mapa, 305, 331

- obrazu działająca po stronie klienta, 329, 338
- obrazu przetwarzana po stronie serwera, 327

margines, 426, 427

menu, 48

- kontekstowe, 53

metadane, 46, 48, 118, 125, 127, 128, 129, 131, 133, 549, 663, 734

metoda, 94, 97, *Patrz też:* funkcja

- addColorStop, 759
- addEventListener, 587, 644, 645
- append, 716
- arc, 780, 784, 785
- arcTo, 780, 781, 782
- assign, 554
- back, 576
- bezierCurveTo, 786
- canPlayType, 743
- clearInterval, 588
- clearRect, 753
- clearTimeout, 588
- cloneNode, 604, 605
- closePath, 777, 780
- createLinearGradient, 759, 762
- createPattern, 765
- createRadialGradient, 763
- CSSStyleDeclaration.getPropertyCSSValue, 632
- DELETE, 698
- do interakcji z oknem, 572
- Document, 561
- document.defaultView.getComputedStyle, 635
- document.writeln, 89
- drawImage, 769, 770, 772
- fill, 777, 778, 780, 789
- fillRect, 753, 757, 762, 779
- forward, 576
- GET, 698
- getAllResponseHeaders, 699
- getComputedStyle, 635
- getContext, 751
- getCurrentPosition, 813, 816, 818, 821
- getElement, 562

- getElementById, 564
 - getElementsByTagName, 548
 - getPropertyPriority, 632
 - getPropertyValue, 632
 - getResponseHeader, 699
 - hasFeature, 557
 - History.pushState, 577, 582
 - HTMLElement, 591
 - HTTP, 259, 689, 697
 - insertAdjacentHTML, 608, 611
 - isEqualNode, 607
 - isSameNode, 606
 - moveTo, 780
 - namedItem, 560
 - open, 688
 - pause, 747
 - play, 747
 - POST, 698
 - postMessage, 586
 - preventDefault, 714
 - przeszukiwania, 564
 - PUT, 698
 - quadraticCurveTo, 787
 - querySelectorAll, 564
 - rect, 779
 - removeEventListener, 644
 - replace, 554
 - restore, 767
 - save, 767
 - send, 689, 719
 - setInterval, 588
 - setItem, 823
 - setRequestHeader, 699
 - setTimeout, 588
 - showModalDialog, 575
 - stringify, 719
 - stroke, 777, 780, 789
 - strokeRect, 753, 754, 755, 757, 779
 - toggle, 595
 - toString, 102
 - watchPosition, 820, 821
 - writeln, 89, 548
- Microsoft, 30
- MIME, 732, 833
- model
- polowy, 346, 449
 - zbudowany z obiektów, 531
- Modernizr, 33, 83
- moduł
- multipart, 36
- modyfikator, 137
- Mozilla Developer Network, 34
- multimedia, 731, 747

N

nagłówek, 211, 235, 697
 Cache-Control, 699
 header, 220
 HTTP, 701
 odpowiedzi, 699
 ządania, 704
 nawigacja, 553
 nazwa użytkownika, 688
 Netscape, 30
 Netscape Navigator, 30
 Node.js, 36, 253, 256, 710, 723
 notacja
 ruby, *Patrz*: pismo logograficzne, znak ruby
 tablicowa, 560, 595
 numer indeksowy, 595

O

obiekt, 93, 95
 ApplicationCache, 839, 841
 Array, 106
 CanvasGradient, 758, 759
 Coordinates, 814
 CSSPrimitiveValue, 632
 CSSRuleList, 621
 CSSStyleDeclaration, 541, 624, 625, 635
 CSSStyleRule, 621
 CSSStyleSheet, 616
 DataTransfer, 806, 808
 document, 89, 603
 Document, 532, 534, 547, 549, 558, 561, 564, 570, 654
 elementy, 559
 DOM, 532, 533
 DOMTokenList, 594
 DragEvent, 806
 elementów, 608
 Event, 643, 645, 654, 691, 692
 FocusEvent, 659
 FormData, 715, 716, 719, 720
 Geolocation, 813
 History, 538, 575
 HTMLAnchorElement, 665
 HTMLAreaElement, 678
 HTMLAudioElement, 740
 HTMLBaseElement, 663
 HTMLBodyElement, 663
 HTMLBRElement, 667
 HTMLCanvasElement, 751
 HTMLDataListElement, 671
 HTMLDetailsElement, 668
 HTMLElement, 532, 539, 549, 564, 591, 592, 644
 tablica, 561
 HTMLEmbedElement, 678, 679
 HTMLFieldSetElement, 671
 HTMLFormElement, 672
 HTMLHeadElement, 665
 HTMLHeadingElement, 668
 HTMLHtmlElement, 665
 HTMLHttpRequest, 719
 HTMLImageElement, 560, 679, 765
 HTMLInputElement, 672
 HTMLLabelElement, 674
 HTMLLegendElement, 674
 HTMLLIElement, 667
 HTMLLinkElement, 664
 HTMLMapElement, 680
 HTMLMediaElement, 740, 741, 745
 HTMLMetaElement, 664
 HTMLMeterElement, 680
 HTMLModElement, 666
 HTMLObjectElement, 680
 HTMLLOListElement, 667
 HTMLOptGroupElement, 674
 HTMLOptionElement, 675
 HTMLOutputElement, 675
 HTMLProgressElement, 681
 HTMLQuoteElement, 666, 667
 HTMLScriptElement, 664
 HTMLSelectElement, 676
 HTMLSpanElement, 667
 HTMLStyleElement, 665
 HTMLTableColElement, 668
 HTMLTableElement, 669
 HTMLTableHeaderCellElement, 669
 HTMLTableRowElement, 670
 HTMLTableSectionElement, 669
 HTMLTextAreaElement, 677
 HTMLTimeElement, 666
 HTMLTitleElement, 665
 HTMLVideoElement, 740
 JavaScript, 685
 JSON, 719
 KeyboardEvent, 659
 Location, 536, 552, 553, 554
 MediaList, 617
 MessageEvent, 586, 587
 NodeList, 561
 Position, 814
 PositionOptions, 818
 ProgressEvent, 692
 Screen, 538
 stanu, 582
 Storage, 823, 827
 StorageEvent, 826
 tekstowy, 608
 Text, 541, 600, 612
 Window, 537, 569, 570, 573, 575, 584, 588, 654, 827
 właściwość, 96, 97, 98
 XMLHttpRequest, 686, 691, 697, 704, 721

obramowanie, 249, 251, 751, 757, *Patrz:* tabela
 obramowanie, *Patrz:* element obramowanie
 obraz, 326, 404, 405, 409, 524, 765, 769
 zastępczy, 735
 obsługa błędów, 106
 obszar przycinania, 790
 odnośnik, 152, 153, 154, 161
 graficzny, 327
 oparty na obrazach, 154
 wewnętrzny, 155
 odtwarzanie natywne, 32, 33
 odwołanie, 167
 okno, 155, 570, 654
 opcja wyboru, 309
 Opera, 35, 211, 283, 300, 301, 387, 455, 493, 690, 691, 715,
 742, 765, 816, 834
 operator, 99, 111

P

parametr funkcji, *Patrz:* funkcja parametry
 parser XML, 46
 parsowanie, 556
 pasek postępu, 339, 340
 phrasing, *Patrz:* element treściowy
 piksel, 79, 326
 referencyjny, 79
 pismo logograficzne, 171, 172
 plik, 709
 audio, *Patrz:* audio
 graficzny, 326
 odtwarzanie, 745
 przesyłanie, 719
 wczytywanie, 307, 308
 wideo, *Patrz:* wideo
 płótno, *Patrz:* element canvas
 podpis, 243
 podścieżka, 776, 780
 podtytuł, 205
 pole
 elementu, 346, 435
 pływające, 442
 przycinania, 415
 tekstowe, 313
 wyboru, 295
 położenie geograficzne, 813
 port, 130, 584
 8080, 257
 potomek, 47, 365
 pozycjonowanie, 450
 prawda/fałsz, 292
 prefiks, 346, 491
 data-, 597
 programowanie obiektowe, 532
 prostokąt, 752, 775, 779

protokół, 130, 584
 ftp, 154
 http, 154
 HTTP, 133
 https, 154
 mailto, 154
 prymityw, 100
 przeciąganie, 53, 799, 800
 przeglądarka, 30, 31, 32, 33, 45, 67, 146, 154, 165, 345,
 547, 558
 Apple Safari, *Patrz:* Safari
 Google Chrome, *Patrz:* Chrome
 historia, 575, 576, 577, 579, 581, 583
 Internet Explorer, *Patrz:* Internet Explorer
 Mozilla Firefox, *Patrz:* Firefox
 NCSA Mosaic, 29
 Netscape Navigator, *Patrz:* Netscape Navigator
 offline, 838
 Opera, *Patrz:* Opera
 prefiks, *Patrz:* prefiks
 przejście, 489, 490, 494, 503
 odwrócone, 493
 tematyczne, 191
 właściwość, 353
 z przekształceniem, 511
 przekazywanie komunikatów między dokumentami, 584
 przekształcenie, 489, 508, 509, 511, 792, 796
 punkt wyjścia, 510
 właściwość, 353
 wypełnienie, 429, 431, 433
 tekstowe, 478
 przestrzeń pionowa, 461
 przeszukiwanie łańcuchowe, 564
 przezroczystość, 514, 793
 przycisk, 386
 graficzny, 305
 przyspieszeniomierz, 815
 pseudoklasa, 375, 388
 interfejsu, 382
 pseudoselektor, 369, 372, 392

R

ramka, 155, 332
 rodzic, 47, 48, 74, 346

S

Safari, 35, 211, 338, 493, 715, 816
 Same origin policy, *Patrz:* żądanie międzyzródłowe
 Scalable Vector Graphics, *Patrz:* SVG
 sekcja, 203, 209, 225, 668
 CACHE, 835, 836
 FALLBACK, 835, 836

- sekcja
 - NETWORK, 838
 - tytuł, 225
- SelectorGadget, 83
- selektor, 234, 392
 - active, 390
 - after, 170, 199, 372, 373
 - atributów, 360, 362
 - before, 170, 199, 200, 372, 373
 - bezpośredni, 368
 - brata, 368, 383
 - checked, 383
 - CSS, 63, 346, 355, 563, 564
 - disabled, 382
 - dynamiczny, 388
 - dzieci, 366, 377
 - empty, 393
 - enabled, 382
 - first-child, 377
 - first-letter, 370
 - first-line, 370
 - focus, 391
 - hover, 389, 490
 - ID, 151, 359
 - in-range, 386, 387
 - invalid, 385
 - klasy, 358, 362
 - lang, 394
 - last-child, 378
 - link, 388
 - łączenie, 363
 - negacji, 383, 392
 - n-tego dziecka, 381
 - ogólny, 368
 - only-child, 379
 - only-of-type, 380
 - optional, 387
 - out-of-range, 386
 - podstawowy, 356
 - potomków, 365
 - pseudoelementów, 369, 370
 - pseudoelementów CSS, 170
 - pseudoklas, 369, 376, 388
 - pseudoklas interfejsu, 382
 - required, 387
 - root, 376
 - statyczny, 388
 - strukturalny, 376
 - target, 394
 - uniwersalny, 356, 358
 - valid, 385, 386
 - visited, 388, 389
- serwer, 709, 838
 - titan, 257
 - utworzenie, 710
- skalowanie, 797
 - grafiki wektorowe, *Patrz:* SVG
- skrót, 167, 168
 - klawiszowy, 49
- skrypt, 142, 147, 256, 556
 - inline, 88, 143, 147
 - komunikacja, 584
 - ślędzący, 148
 - wykonywalny po stronie klienta, 30
 - zaufany, 587
 - zewnętrzny, 88, 147
 - źródło, 584
- słowo kluczowe
 - function, 90
 - new, 688
 - return, 91
 - this, 95
 - var, 92
- spacja, 472
- stan rysowania, 754, 767
- standard, 34
 - HTML, 31
 - HTML5, 33
 - RFC3339, 180
- stopka, 211, 235, 236
 - footer, 220
- styl
 - aktywowanie, 620
 - analiza, 83
 - definiowanie, 62, 63, 65, 67
 - deklaracja, 61
 - dezaktywowanie, 620
 - dostęp, 621
 - dziedziczenie, *Patrz:* dziedziczenie
 - elementu, 623, 624
 - graficzny, 751
 - importowanie, 66
 - inline, 62, 69, 549
 - kaskadowość, *Patrz:* kaskadowość, CSS
 - nakładanie, 62
 - obliczony, 615, 635
 - osadzony, 63, 69
 - precyzja, 70
 - prosty, 61
 - przełładarki, 67, 69
 - przetworzony, 83
 - selektor, 65, 83
 - użytkownika, 68, 69, 70
 - wartość, 62
 - inherit, 74
 - właściwość, 61, 67, 69
 - width, 75
 - zewnętrzny, 69
- SVG, 34

system
 GPS, 815
 operacyjny Apple iOS, *Patrz:* iOS
 szyfrowanie, 286, 316

Ś

ścieżka, 776, 780, 789
 śledzenie postępów wczytywania danych, 721

T

tabela, 229, 234, 236, 242, 249, 430, 435, 465, 513, 516, 668, 682
 krawędzie, 518
 nieregularna, 238
 obramowanie, 516, 519
 odstępy pomiędzy komórkami, 518
 podpis, 520
 puste komórki, 519
 układ, 521
 tablica, 104, 105, 106, 560
 obiektów Attr, 598
 obiektów HTMLElement, 561
 wartości indeksowanych według nazwy właściwości, 597
 tabulator, 472
 tag, *Patrz:* znacznik
 tekst, 469, 600
 cień, 481
 dekorowanie, 479
 dodany, 178
 dzielenie wyrazów, 476
 fonty, 482, 484, 485, 486
 justowanie, 470
 kierunek, 474
 odstępy między wyrazami, 475
 podkreślony, 161
 podświetlony, 177
 pomniejszony, 162
 przekreślony, 159
 przekształcanie, 479
 rysowanie, 790
 usunięty, 178
 ważny, 160
 wcięcie pierwszego wiersza, 478
 właściwość, 352
 wstawianie elementu, 612
 zawijanie, 472
 tło, *Patrz:* element tło
 treść
 numeryczna, 373
 zapasowa, 750
 zastępcza, 326, 336, 732
 trójkąt, 780

tryb offline, 832
 typ
 boolean, 92, 93, 109
 konwersja, 102, 103
 MIME, 732, 833
 number, 92, 93, 103, 109
 prosty, 92
 string, 92, 109
 typografia, 482

U

upuszczanie, 53, 799, 800

W

W3C, 31, 32, 34, 259, 345
 wartość, 132
 domyślna, 280
 liczbowa, 339
 noarchive, 132
 nofollow, 132
 noindex, 132
 null, 108, 109, 110, 111, 563
 numeryczna, 290
 obliczona, 626
 skonfigurowana, 626
 undefined, 108, 109, 110, 111
 ważność, 631, 632
 wcięcie pierwszego wiersza, 478
 Webkit, 417
 wideo, 32, 731, 732, 765, 770
 format, 737
 odtwarzanie, 745
 źródło, 736
 właściwość, 353
 animacji, 353, 497
 counter, 199, 200
 counter-reset, 373
 CSSStyleDeclaration, 627
 layoutu, 351
 length, 563
 pomocnicza, 625, 627
 przejścia, 353, 491, 494
 przekształcenia, 353
 sprawdzanie automatyczne, 630
 tekstu, 352
 width, 137
 World Wide Web Consortium, *Patrz:* W3C
 współrzędne, 306
 wtyczka, 30, 31, 32, 111, 333, 342
 wynik obliczenia, 316
 wyobrażenie wartości liczbowych, 339
 wypełnienie, 757
 wzór, 765

X

XHTML, 46
XML, 46
XSS, 701, 703

Y

YouTube, 33, 334

Z

zakładka, 155
zapytanie, 573
zasoby
 audiowizualne, 342
 osadzanie, 325, 333, 342
 pobieranie z wyprzedzeniem, 142
 źródła, 584
zdarzenie, 531, 554, 569, 584, 639
 atrybut, 640
 bąbelkowanie, 651, 653
 blur, 544
 cel, 647
 click, 544, 653
 dblclick, 544
 DOM, 544
 faza bąbelkowania, 647, 649, 651
 faza celu, 647, 649, 650
 faza przechwytywania, 647
 focus, 544
 focusin, 544
 focusout, 544
 fokusowania, 657
 formularza, 660
 keydown, 544
 keypress, 544
 keyup, 544
 klawiatury, 658
 message, 586
 mousedown, 544
 mouseenter, 544
 MouseEvent, 806
 mouseleave, 544
 mousemove, 544
 mouseout, 544, 654
 mouseover, 544, 640, 654
 mouseup, 544

mysz, 654
nasłuchiwaniec, 644, 648, 650
nasłuchiwanie, 586
onabort, 544
onafterprint, 544
onbeforeprint, 544
onhashchange, 544
onmouseover, 640
onpopstate, 544
onresize, 544
onunload, 544
progress, 691
prosta procedura obsługi, 640, 642
przeciąganie, 800, 806
readystatechange, 557, 688, 691, 692
reset, 544
storage, 826
submit, 544
typ, 645
zmienna, 92
 globalna, 92
 lokalna, 92
 typ, 92, *Patrz też:* typ
 window, 570
znacznik, 40, 116, 127
 otwierający, 42, 44, 548
 otwierającym, 732
 pojedynczy, 42, 44
 zamykający, 42, 44, 145, 732
znak
 biały, 472
 ruby, 172
 specjalny, 260

Ź

źródło
 audio, 739
 zaufane, 587

Ż

żądanie, 689, 694, 701, 712
 anulowanie, 704
 GET, 259, 689
 międzyźródłowe, 701
 nagłówków, 697, 704
 POST, 259, 689
 preflight, 70

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>

HTML5 to hit ostatnich miesięcy. Jego możliwości robią ogromne wrażenie na projektantach stron internetowych, a użytkownicy na tym korzystają. Usługi geolokalizacyjne, zaawansowane funkcje graficzne oraz rozbudowana obsługa multimedialnych – to tylko niektóre z atutów HTML5. Można śmiało i bez przesady powiedzieć, że język ten będzie gorącym tematem jeszcze przez długi czas. Dlatego warto już dziś sprawdzić, jak wykorzystać jego potencjał.

Dzięki tej książce szybko opanujesz zasady HTML5. Już wkrótce bez najmniejszego problemu będziesz korzystał z nowych znaczników canvas, audio i video. Ponadto nauczysz się stosować mechanizmy do przechowywania danych w lokalnych zasobach przeglądarki oraz sprawdzisz, gdzie w danej chwili znajduje się użytkownik. Autor książki kładzie nacisk na poprawność semantyczną tworzonego kodu. Jest to niezwykle istotne w obecnych czasach, gdy Twoje dzieło będzie oglądane na przeróżnych urządzeniach – zarówno stacjonarnych, jak i mobilnych. Książka ta jest rewelacyjnym źródłem informacji o językach HTML5, CSS oraz JavaScript. Musisz ją mieć!

HTML5 to nowe możliwości:

- obsługa plików multimedialnych,
- zaawansowane funkcje graficzne,
- usługi geolokalizacyjne,
- możliwość pracy bez połączenia z siecią,
- dostępność na różnych urządzeniach!

Apress®

Nr katalogowy: 11939

Księgarnia internetowa:
 <http://helion.pl>

Zamówienia telefoniczne:
 0 801 339900

 0 601 339900

helion.pl
 księgarnia
 internetowa

Sprawdź najnowsze promocje:

• <http://helion.pl/promocje>

Książki najchętniej czytane:

• <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

• <http://helion.pl/nowosci>



Helion

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

sięgnij po WIĘCEJ



KOD KORZYŚCI

Cena 129,00 zł
 ISBN 978-83-246-5081-1



9 788324 650811