

Wydanie V

Steven M. Schafer

HTML, XHTML i CSS

Opanuj

możliwości języka HTML
i kaskadowych
arkuszy stylów

Twórz

strony statyczne,
dynamiczne
i na urządzenia mobilne

Odkryj

sposoby rozbudowywania
stron WWW

Helion

Biblia

Wiedza obiecana

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
© Helion 1991–2010

HTML, XHTML i CSS. Biblia. Wydanie V

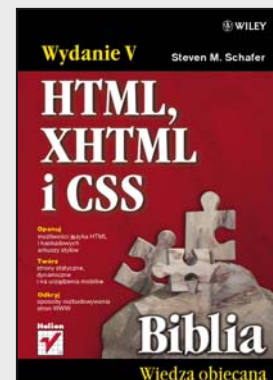
Autor: Steven M. Schafer

Tłumaczenie: Piotr Rajca

ISBN: 978-83-246-2742-4

Tytuł oryginału: [HTML, XHTML, and CSS Bible](#)

Format: 172×245, stron: 768



- Opanuj możliwości języka HTML i kaskadowych arkuszy stylów
- Twórz strony statyczne, dynamiczne i na urządzenia mobilne
- Odkryj sposoby rozbudowania stron WWW

Jeśli czytasz notkę tej książki, zapewne zdecydowałeś się na stworzenie własnej, profesjonalnej strony internetowej. Jedyne, czego Ci teraz trzeba, to wiedza, jak sprawnie wykorzystać niezbędny w tym celu zestaw technologii – języki HTML i XHTML oraz kaskadowe arkusze stylów CSS. To właśnie te narzędzia pozwalają zbudować dokument witryny, sformatować umieszczone na niej teksty, osadzać elementy multimedialne i nadawać jej atrakcyjny wizualnie wygląd oraz nowoczesny, dynamiczny charakter. A jeśli masz jeszcze wątpliwości, czy zadanie to leży w zasięgu Twoich umiejętności, ta książka z pewnością je rozwieje!

Podręcznik ten, adresowany zarówno do początkujących webmasterów, jak i zawodowców, dla pierwszych stanowi solidną podstawę do rozpoczęcia przygody z tworzeniem stron WWW, natomiast dla drugich wyczerpujące kompendium wiedzy o odpowiednich technologiach. Autor tak uporządkował zagadnienia – od prostych po złożone – aby w jak największym stopniu ułatwić ich opanowanie przez czytelnika. Naukę rozpoczniesz zatem od poznania znaczników HTML, struktury i atrybutów tego języka. Dowiesz się, jak używać multimedii i skryptów, oraz skorzystasz z programów wspomagających tworzenie czy testowanie dokumentów. Później przeczytasz o praktycznych rozszerzeniach HTML – XML i HTML Basic – oraz zgłębisz wszystkie kwestie związane ze stosowaniem CSS i publikacją strony. Ogarniesz w ten sposób wszystkie aspekty samodzielnego budowania nowoczesnej, zgodnej ze standardami, estetycznej witryny internetowej!

- Budowanie treści z wykorzystaniem języka HTML
- Tworzenie list, łączy, tabel i ramek
- Osadzanie elementów multimedialnych
- Skrypty serwerowe i skrypty klienckie
- Czym jest i czemu służy DHTML?
- Projektowanie serwisów wielojęzycznych i zasady lokalizacji serwisów
- Publikacja witryny w sieci
- Porządkowanie i walidacja dokumentów
- Zastosowanie języka XML
- Tworzenia stron dla urządzeń mobilnych
- Kontrola prezentacji za pomocą języka CSS
- Praktyczne sztuczki i triki w CSS

Spis treści

O autorze	21
Wprowadzenie	23
Część I Tworzenie treści z wykorzystaniem języka HTML	33
Rozdział 1. Czym jest język znaczników?	35
Co tu robimy?	35
Przedstawienie hipertekstu	36
Przedstawienie instrukcji znacznikowych	37
Przedstawienie języka znaczników	38
Podsumowanie	40
Rozdział 2. Wartości i jednostki w HTML-u	41
Podstawowa postać zapisu atrybutów	41
Wspólne atrybuty	44
Identyfikacja znaczników — identyfikatory i klasy	44
Teksty i komentarze	45
Komentarze	45
Sekcje CDATA	46
Jednolite identyfikatory zasobów	47
Opcje języka i ustawienia międzynarodowe	47
Kod języka	48
Kierunek tekstu	48
Podsumowanie	48
Rozdział 3. Co jest umieszczane w dokumentach HTML?	49
Określanie typu dokumentu	50
Ogólna struktura dokumentu — HTML, nagłówek i treść	50
Znacznik <html>	51
Znacznik <head>	51
Znacznik <body>	52
Definicje stylów	52
Elementy blokowe — oznaczenia definiujące akapity oraz inne bloki treści	53
Sformatowane akapity	54
Nagłówki	54
Cytaty	56
Elementy list	57

Tekst preformatowany	58
Sekcje strony	59
Elementy wewnątrzwierszowe — oznaczenia znaków	61
Podstawowe znaczniki wewnątrzwierszowe	61
Fragmenty tekstu	62
Znaki specjalne (symbole znakowe)	63
Elementy organizacyjne	64
Tabele	64
Formularze	65
Łącza do innych stron	67
Obrazy	68
Komentarze	69
Skrypty	69
Połączenie wszystkich elementów	70
Podsumowanie	71
Rozdział 4. Elementy nagłówka	73
Określanie tytułu dokumentu	73
Podawanie informacji używanych przez wyszukiwarki	74
Określanie domyślnej ścieżki	75
Sekcje skryptów	77
Sekcje stylów	77
Określanie profili	77
Kolor i obraz tła	78
Określanie koloru tła dokumentu	78
Określanie obrazu tła dokumentu	79
Podsumowanie	80
Rozdział 5. Podstawy określania struktury tekstów	81
Formatowanie akapitów	81
Podział wiersza	83
Sekcje	84
Poziome linie	88
Cytaty blokowe	89
Tekst preformatowany	90
Podsumowanie	91
Rozdział 6. Tekst	93
Metody formatowania tekstu	93
Znacznik 	94
Akcentowanie i inne znaczniki dotyczące tekstu	94
Formatowanie tekstu przy użyciu CSS	94
Pogrubienie i kursywa	97
Stosowanie uwypuklenia zamiast kursywy	97
Czcionka o stałej szerokości	98
Indeksy — górny i dolny	98
Skróty	99
Oznaczanie zmian w dokumencie — wstawianie i usuwanie tekstu	99
Grupowanie elementów tekstowych znacznikiem 	100
Podsumowanie	101

Rozdział 7. Listy	103
Omówienie list	103
Listy uporządkowane (numerowane)	104
Listy nieuporządkowane (punktowane)	108
Listy definicji	112
Listy zagnieżdżone	114
Podsumowanie	115
Rozdział 8. Łączy	117
Czym są łączy?	118
Łączy do stron WWW	119
Łączy względne i bezwzględne	120
Docelowe okna łączy	122
Tekst podpowiedzi łączy	123
Skróty klawiaturowe i kolejność uaktywniania łączy	123
Skróty klawiaturowe	124
Kolejność uaktywniania łączy	125
Tworzenie kotwic	125
Dobór kolorów łączy	126
Parametry dokumentu docelowego	128
Znacznik <link>	129
Podsumowanie	130
Rozdział 9. Tabele	131
Części składowe tabeli HTML	131
Szerokość i wyrównanie tabeli	133
Odstępy i otoczenie komórek	137
Obramowanie i krawędzie	138
Obramowanie tabeli	138
Krawędzie tabeli	140
Wiersze	140
Komórki	142
Podpis tabeli	144
Grupowanie wierszy — nagłówek, treść i stopka tabeli	146
Kolor tła	148
Komórki rozciągające się na kilka wierszy lub kolumn	149
Grupowanie kolumn	153
Zastosowanie tabel do formatowania stron	154
Podstawy formatowania z wykorzystaniem tabel	155
Przykłady stron dostępnych w internecie	158
Strony pływające	158
Niestandardowe połączenie grafiki i tekstu	163
Menu nawigacyjne i bloki dokumentów	166
Dokumenty wielokolumnowe	167
Podsumowanie	169
Rozdział 10. Ramki	171
Omówienie ramek	171
Zbiory ramek i zawartość ramek	172
Tworzenie zbioru ramek	173
Marginesy, obramowania i paski przewijania ramek	176
Modyfikacja rozmiaru ramek	179

Odsyłacze do ramek	179
Zagnieżdżone zbiory ramek	182
Ramki pływające	183
Podsumowanie	186
Rozdział 11. Formularze	187
Podstawowe informacje o formularzach	188
Wstawianie formularzy	191
Metoda GET	191
Metoda POST	192
Dodatkowe atrybuty znacznika <form>	192
Etykiety pól	193
Pola tekstowe	193
Pola haseł	194
Przyciski opcji	194
Pola wyboru	195
Listy wyboru	195
Obszary tekstowe	197
Pola ukryte	199
Przyciski	199
Rysunki	200
Pola plików	201
Przyciski przesyłania danych i czyszczenia pól	202
Kolejność uaktywniania kontrolek i skróty klawiaturowe	203
Uniemożliwienie wprowadzania zmian	203
Grupy pól i ich opis	205
Stosowanie zdarzeń do obsługi formularzy	206
Skrypty formularzy i usługi skryptowe	210
Pobranie programu obsługi formularza	210
Wykorzystanie usług skryptowych	211
Podsumowanie	211
Rozdział 12. Kolory i rysunki	213
Podstawowe informacje o kolorach	213
Inne sposoby określania kolorów	214
Ewolucja kolorów używanych na stronach WWW	215
Stosowanie odpowiednich metod określania kolorów	221
Formaty graficzne stosowane w dokumentach WWW	223
Kompresja obrazu	223
Opcje kompresji	224
Format GIF	225
Format JPEG	225
Format PNG	226
Przygotowanie plików graficznych	226
Najważniejsze funkcje	227
Progresywne obrazy JPEG i rysunki GIF z przeplotem	228
Wykorzystanie efektu przezroczystości	228
Animacje	229
Wstawianie rysunków	230
Rozmieszczanie rysunków	232

Opis wyświetlany w przeglądarkach tekstowych	235
Wymiary i skalowanie rysunków	236
Obramowanie rysunków	238
Mapy obrazu	239
Definiowanie mapy obrazu	240
Definiowanie obszarów aktywnych	240
Połączenie poszczególnych rozwiązań	243
Podsumowanie	244
Rozdział 13. Obiekty multimedialne	245
Animowane obrazy	246
Formaty animacji i klipów wideo, pluginy oraz odtwarzacze	248
Popularne formaty i odtwarzacze	250
Windows Media Player	251
Osadzanie multimediów przy użyciu znacznika <object>	251
Osadzanie odtwarzacza Windows Media Player przy użyciu znacznika <object>	255
Osadzanie klipów wideo z serwisu YouTube	256
Umieszczanie plików dźwiękowych na stronach WWW	260
Tworzenie plików multimedialnych	261
Podsumowanie zagadnień wykorzystania multimediów na stronach WWW	261
Podsumowanie	262
Rozdział 14. Znaki specjalne	263
Kodowanie znaków	263
Znaki specjalne	264
Znaki spacji i myślników	265
Symbol praw autorskich i znaku towarowego	266
Symbole walut	267
Rzeczywiste znaki cudzołowy	267
Strzałki	268
Znaki zawierające symbole akcentu	269
Litery alfabetu greckiego i symbole matematyczne	271
Inne użyteczne symbole	274
Podsumowanie	276
Rozdział 15. Projektowanie stron WWW dla obcokrajowców	277
Projektowanie serwisów wielojęzycznych i zasady lokalizacji serwisów	277
Tłumaczenie strony internetowej	279
Standard Unicode	279
Basic Latin (podstawowy łaćniński, U+0000 – U+007F)	284
Kodowanie polskich znaków diakrytycznych	290
Podsumowanie	290
Rozdział 16. Skrypty	293
Skrypty serwerowe i skrypty klienckie	293
Skrypty klienckie	293
Skrypty serwerowe	294
Określanie domyślnego języka skryptowego	294
Dołączanie skryptów	295
Wywoływanie zewnętrznych skryptów	296

Wywoływanie skryptów za pomocą zdarzeń	297
Ukrywanie skryptów przed starszymi przeglądarkami	301
Podsumowanie	301
Rozdział 17. Dynamiczny HTML	303
Do czego służy dynamiczny HTML?	303
Jak działa DHTML?	304
Obiektowy model dokumentu	305
Historia DOM	305
Opis obiektowego modelu dokumentu	306
Właściwości i metody węzłów DOM	308
Poruszanie się po drzewie DOM i modyfikacja węzłów	309
Obiektowy model dokumentu języka JavaScript	312
Obiekt window	313
Obiekt document	315
Obiekt form	316
Obiekt location	316
Obiekt history	317
Obiekt this	317
Stosowanie procedur obsługi zdarzeń	318
Dostęp do elementów przy wykorzystaniu ich identyfikatorów	319
Zagadnienia zgodności z różnymi przeglądarkami	320
Wykrywanie używanej przeglądarki	320
Wykrywanie obiektów	321
Przykłady rozwiązań DHTML	321
Automatyzacja formularzy — obsługa pól wyboru	322
Tworzenie efektów podmiany przy wykorzystaniu JavaScriptu	323
Rozwijane menu	324
Podsumowanie	327
Rozdział 18. Przyszłość języka HTML — HTML 5	329
Większe możliwości publikowania i określania układu	329
Dostępne multimedia	331
Zmiany — elementy i atrybuty	332
Nowe elementy	333
Nowe atrybuty w elementach	333
Nowe typy pól formularzy (elementu input)	334
Nowe globalne atrybuty	335
Elementy uznane za przedawnione	335
Przedawnione atrybuty	336
Podsumowanie	337
Część II Narzędzia oraz inne wersje języka HTML	339
Rozdział 19. Programy do projektowania stron internetowych	341
Edytory tekstowe	342
Proste edytory tekstowe	342
Inteligentne edytory tekstowe	342
Edytory HTML	343

Edytory HTML pracujące w trybie WYSIWYG	345
NetObjects Fusion	345
Dreamweaver firmy Macromedia	346
Dodatki do przeglądarki Firefox	347
Inne narzędzia	349
Programy graficzne	349
Flash firmy Adobe	351
Podsumowanie	352
Rozdział 20. Publikacja witryn	353
Wprowadzenie do FTP	353
Programy klienty FTP	354
Popularne klienty FTP	356
Podstawowe zasady organizacji plików w obrębie witryny WWW	358
Podsumowanie	359
Rozdział 21. Wprowadzenie do języka XML	361
Podstawy języka XML	362
Składnia języka XML	363
Deklaracje XML i DOCTYPE	364
Elementy	364
Atrybuty	366
Komentarze	367
Dane nieprzetwarzane	367
Stałe tekstowe	367
Przestrzenie nazw	368
Arkusze stylów	369
Definicje typu dokumentu (DTD)	369
Użycie elementów w definicji typu dokumentu	371
Definiowanie atrybutów w DTD	373
Definiowanie i użycie stałych tekstowych w definicji DTD	374
Użycie danych typu PCDATA i CDATA w definicji typu	375
Schematy XML	375
Użycie schematów	376
Zastosowanie dokumentów XML	378
Przekształcenia XSLT	379
Edycja kodu XML	379
Analiza kodu XML	379
Podsumowanie	380
Rozdział 22. Tworzenie stron dla urządzeń przenośnych	381
Ewolucja internetu mobilnego	381
Mroczne początki internetu mobilnego	382
Organizacja Open Mobile Alliance i nowe standardy	383
Podsumowanie	383
Język XHTML Basic 1.1	384
Deklaracja doctype XHTML Basic 1.1	384
Elementy języka XHTML Basic 1.1	384
Zagadnienia wymagające szczególnej uwagi	385
Narzędzia do tworzenia stron dla urządzeń przenośnych	388
Podsumowanie	389

Rozdział 23. Porządkowanie i walidacja dokumentów	391
Porządkowanie kodu HTML	391
HTML Tidy	394
Pobieranie narzędzia HTML Tidy	394
Uruchamianie narzędzia HTML Tidy	394
Sprawdzanie poprawności kodu	397
Określanie poprawnego typu dokumentu	397
Narzędzia do weryfikacji poprawności kodu	397
Jak weryfikować dokumenty?	397
Dodatkowe testy i walidacja	399
Testowanie kodu w różnych przeglądarkach	399
Testowanie w różnych rozdzielczościach ekranu	400
Podsumowanie	400
Rozdział 24. Sztuczki i triki w języku HTML	401
Wstępne wczytywanie rysunków	401
Kontrolowanie podziału tekstu w komórkach tabeli	403
Paski tytułu o zmiennej szerokości	404
Symulowanie gazetowego układu kolumn	406
Dołączanie rozmiaru rysunków w celu przyspieszenia ich wczytywania	408
Zabezpieczenia adresów e-mail	409
Automatyzacja formularzy	411
Operacje na obiektach formularzy	411
Weryfikacja wartości pól	413
Modyfikowanie środowiska przeglądarki	416
Koncepcja	416
Implementacja	416
Zastosowane funkcje JavaScript	421
Podsumowanie	422
Część III Kontrolowanie prezentacji za pomocą CSS	423
Rozdział 25. Wprowadzenie do kaskadowych arkuszy stylów	425
Przeznaczenie CSS	425
Style i HTML	426
1., 2. i 3. poziom CSS	428
Definiowanie stylów	429
Kaskada stylów	430
Podsumowanie	432
Rozdział 26. Tworzenie reguł stylów	433
Zapis definicji stylów	433
Przedstawienie selektorów	435
Dopasowywanie elementów według typu	435
Korzystanie z selektora uniwersalnego	435
Dopasowywanie elementów według klasy	436
Dopasowywanie elementów przy użyciu identyfikatora	437
Dopasowywanie elementów, które zawierają określony atrybut	437
Korzystanie z elementów dzieci, potomków oraz elementów przystających	438
Omówienie dziedziczenia	440

Pseudoklasy i ich stosowanie	441
Definiowanie stylów łączy	441
Pseudoklasa :first-child	442
Pseudoklasa :lang	442
Pseudoelementy	443
Stosowanie stylów dla pierwszego wiersza tekstu w elemencie	443
Stosowanie stylów dla pierwszej litery elementu	444
Definiowanie przed danym tekstem i po nim	445
Wyrażenia skrótowe	446
Podsumowanie	448
Rozdział 27. Wartości i jednostki w języku CSS	449
Ogólne zasady podawania wartości właściwości	449
Jednostki wartości właściwości	451
Wartości w postaci słów kluczowych	452
Standardowe jednostki miar	452
Miary rozdzielczości ekranu	453
Miary względne	454
Funkcje związane z kolorami i adresami URL	455
Jednostki dźwiękowe	456
Podsumowanie	457
Rozdział 28. Dziedziczenie i kaskadowanie w języku CSS	459
Dziedziczenie	459
Kaskadowanie	461
Specyficzność selektorów	463
Podsumowanie	464
Rozdział 29. Właściwości czcionek	465
Wprowadzenie do czcionek	465
Rodzaje czcionek	466
Określanie rozmiaru czcionki	468
Określanie stylu czcionki	469
Interlinie	470
Zagnieżdżanie czcionek w dokumencie	470
Podsumowanie	472
Rozdział 30. Formatowanie tekstu	473
Wyrównywanie tekstu	473
Kontrolowanie wyrównania poziomego	474
Kontrolowanie wyrównania pionowego	476
Tworzenie wcięcia w tekście	479
Kontrolowanie znaków niewidocznych w tekście	479
Obiekty przestawne	479
Właściwość white-space	481
Kontrolowanie odstępów między literami i słowami	483
Definiowanie wielkich liter	484
Dekorowanie tekstu	486
Tekst generowany automatycznie	487
Definiowanie stylów tabeli	487

Kontrolowanie atrybutów tabeli	488
Obramowanie tabeli	488
Odstępy w ramce tabeli	489
Pojedyncze obramowanie	491
Obramowania wokół pustych komórek	492
Układ graficzny tabeli	493
Wyrównywanie i pozycjonowanie podpisów	493
Podsumowanie	495
Rozdział 31. Formatowanie list	497
Ogólne informacje o listach	497
CSS — każdy element pasuje	498
Właściwość list-style-type	499
Pozycjonowanie markerów	501
Punktory rysunkowe	501
Podsumowanie	502
Rozdział 32. Obramowania, odstępy i marginesy	503
Omówienie modelu formatowania pojemnika	503
Dodawanie odstępu do elementu	506
Dodawanie obramowania	507
Szerokość obramowania	507
Styl obramowania	508
Kolor ramki	510
Największy skrót: właściwość border	510
Dodatkowe właściwości obramowań	511
Definiowanie marginesów elementu	511
Wykorzystywanie dynamicznego obramowania	513
Podsumowanie	514
Rozdział 33. Kolory i tło	515
Kolory elementów	515
Kolor pierwszoplanowy	515
Kolory tła	516
Obrazy tła	519
Powtarzanie i przewijanie obrazów tła	522
Określanie pozycji obrazów tła	523
Skrótowa właściwość background	525
Podsumowanie	525
Rozdział 34. Definiowanie układów stron	527
Omówienie pozycjonowania elementów	527
Pozycjonowanie statyczne	528
Pozycjonowanie względne	529
Pozycjonowanie bezwzględne	529
Pozycjonowanie stałe	530
Określanie pozycji elementu	532
Elementy dryfujące do lewej lub prawej strony	534
Definiowanie szerokości i wysokości elementu	537
Dokładne definiowanie rozmiarów	537
Definiowanie rozmiaru maksymalnego oraz minimalnego	538
Kontrola przepełnienia elementu	539

Układanie elementów na stosie	540
Kontrolowanie widoczności elementu	544
Podsumowanie	546
Rozdział 35. Pseudoelementy i wygenerowane treści	547
Właściwość content	547
Pseudoelementy	549
Stosowanie stylów dla pierwszego wiersza tekstu w elemencie	550
Stosowanie stylów dla pierwszej litery elementu	550
Pseudoelementy :before i :after	552
Definiowanie znaków cudzozyłowy	553
Automatyczne numerowanie elementów	553
Obiekt counter	554
Zmienianie wartości obiektu counter	554
Przykład zastosowania liczników: numery rozdziałów i podrozdziałów	555
Własne numerowanie list	556
Podsumowanie	558
Rozdział 36. Dynamiczny HTML i CSS	559
Korzystanie z właściwości CSS w kodzie JavaScript	559
Użyteczne operacje z użyciem CSS	565
Ukrywanie i wyświetlanie tekstu	565
Powiększanie obrazków	567
Podmieniane menu	569
Podsumowanie	572
Rozdział 37. Typy mediów i definiowanie stron do druku	573
Typy mediów obsługiwane przez CSS	574
Określanie typu mediów	574
Przygotowywanie dokumentu do drukowania	577
Model formatowania pojemnika strony	577
Definiowanie rozmiaru strony	577
Właściwości page-break	580
Zarządzanie wdowami i sierotami	583
Przygotowanie dokumentu do drukowania dwustronnego	584
Tworzenie dokumentów dla różnych mediów	585
Dokument do prezentacji w internecie	585
Ponowne formatowanie strony	589
Podsumowanie	590
Rozdział 38. Przyszłość CSS — CSS 3	591
Po prostu lepsze	592
Modularność	592
Stosowanie właściwości CSS 3 już dziś	594
Większa kontrola nad wybranymi elementami	595
Zaokrąglone wierzchołki elementów raz jeszcze	596
Podsumowanie	597

Część IV Dodatkowe narzędzia CSS	599
Rozdział 39. Style interfejsu użytkownika	601
Modyfikacje wyglądu wskaźnika myszy	601
Kolory interfejsu użytkownika	603
Czcionki interfejsu użytkownika	606
Podsumowanie	607
Rozdział 40. Testowanie i walidacja kodu CSS	609
Sprawdzanie składni w czasie tworzenia stylów	609
Słowo o formatowaniu	611
Walidacja kodu CSS	612
Dodatki do przeglądarki Firefox służące do edycji CSS	613
Podsumowanie	614
Rozdział 41. Sztuczki i triki w języku CSS	615
Wysunięcie	615
Rozszerzające się przyciski	617
Wyróżnione cytaty	620
Menu w formie zakładek	622
Elementy z zaokrąglonymi wierzchołkami	624
Elementy pływające	627
Tekst otaczający inne elementy	630
Podsumowanie	634
Dodatki	635
Dodatek A Krótki przegląd elementów języka HTML	637
Lista elementów	638
<a>	638
<abbr>	639
<acronym>	639
<address>	640
<area> (rzadko stosowany)	641
	641
<base>	642
<bdo>	642
<big>	643
<blockquote>	643
<body>	644
 	645
<button>	645
<caption>	646
<cite>	647
<code>	647
<col>	648
<colgroup>	648
<dd>	649
	649
<dfn>	650
<div>	650

<dl>	651
<dt>	651
	652
<fieldset>	652
<form>	653
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	654
<head>	654
<hr>	655
<html>	655
<i>	656
	656
<input>	657
<ins>	658
@<kbd>	659
<label>	659
<legend>	660
	660
<link>	661
<map>	662
<meta>	662
<noscript>	663
<object>	664
	665
<optgroup>	665
<option>	666
<p>	666
<param>	667
<pre>	668
<q>	668
<samp>	669
<script>	669
<select>	670
<small>	671
	672
	672
<style>	673
<sub>	674
<sup>	674
<table>	674
<tbody>	676
<td>	676
<textarea>	677
<tfoot>	678
<th>	678
<thead>	679
<title>	680
<tr>	680
<tt>	681
	682
<var>	682

Atrybuty zdarzeń	683
Zdarzenia standardowe	683
Inne zdarzenia	683
Inne często spotykane atrybuty	684
Atrybuty podstawowe	684
Atrybuty umiędzynarodawiania	684
Kody często używanych kolorów	684
Dodatek B Krótki przegląd znaków specjalnych języka HTML	685
Dodatek C Krótki przegląd właściwości języka CSS	695
Lista właściwości	696
Lista właściwości — zestawienie	696
Tło	698
Listy	700
Wygenerowane dane	702
Czcionki i tekst	703
Kierunek tekstu	707
Błoki	708
Określanie pozycji elementów	711
Obramowania	713
Tabele	716
Druk	717
Różne	718
Dodatek D Krótki przegląd selektorów języka CSS	721
Podstawowe selektory elementów	722
Selektory potomków	722
Selektory dzieci	722
Selektory pierwszego brata	723
Selektory klas	723
Selektory identyfikatorów	723
Selektory atrybutów	724
Dodatek E Krótki przegląd pseudoelementów i pseudoklas	725
Pseudoelementy	726
Pseudoklasy	726
Skorowidz	729

Rozdział 36.

Dynamiczny HTML i CSS

W tym rozdziale:

- ◆ Korzystanie z właściwości CSS w kodzie JavaScript
- ◆ Użyteczne operacje z życiem CSS

Kaskadowe arkusze stylów mogą być potężnym narzędziem umożliwiającym tworzenie ładnie sformatowanych stron WWW. W tym rozdziale opisano, w jaki sposób w różnych przeglądarkach można manipulować wartościami właściwości CSS, by nadać dokumentom HTML dynamiczny charakter. Dowiesz się z niego, w jaki sposób można uzyskiwać dostęp do właściwości CSS oraz jak operować nimi w skryptach, aby wykonywać takie operacje jak zmiana koloru tekstu. Jak się okaże, można zmienić wartość każdej właściwości CSS.

Jak się przekonasz, niektóre przeglądarki (mamy tu na myśli głównie program Internet Explorer) udostępniają możliwość tworzenia dynamicznych efektów wizualnych, takich jak cienie i rozmycia, z wykorzystaniem składni zbliżonej do CSS.

Korzystanie z właściwości CSS w kodzie JavaScript

Przeglądarki fundacji Mozilla (Firefox) oraz Internet Explorer zapewniają możliwość dostępu do właściwości CSS poziomu 1. w kodzie JavaScript, za pośrednictwem obiektowego modelu dokumentu (DOM). Niestety, pomiędzy DOM stosowanym w przeglądarkach Firefox oraz Internet Explorer występują pewne różnice. W obu tych rozwiązaniach zostały częściowo zaimplementowane możliwości standardu CSS2, jednak nie są to te same możliwości, przez co wykorzystujący je skrypt, który działa w jednej z tych przeglądarek, może nie działać w drugiej. Należy zwrócić uwagę, że silnik Gecko (używany do przetwarzania i wyświetlania stron WWW w przeglądarkach Mozilli) obsługuje wszystkie właściwości standardu CSS poziomu 2.

Ogólnie rzecz biorąc, z właściwości CSS korzysta się w standardowy sposób — ich wartości są odczytywane w sposób typowy dla właściwości i ustawiane przy użyciu metod. Aby odwołać się do właściwości CSS w kodzie JavaScript, należy po prostu podać jej nazwę, o ile tylko nie zawiera ona łącznika. W przypadku nazw kilkuczłonowych znak

łącznika należy usunąć, a znak znajdujący się bezpośrednio za nim należy zapisać wielką literą. Wszystkie pozostałe znaki nazwy właściwości są zapisywane małymi literami. Na przykład właściwość:

```
font-size
```

staje się w kodzie JavaScript:

```
fontSize
```

Tak uzyskaną nazwę właściwości dodaje się do nazwy (lub identyfikatora) obiektu zawierającego kolekcję stylów. Na przykład, aby odwołać się do właściwości `font-size` obiektu o nazwie `bigText`, należy użyć następującego wyrażenia:

```
bigText.style.fontSize
```

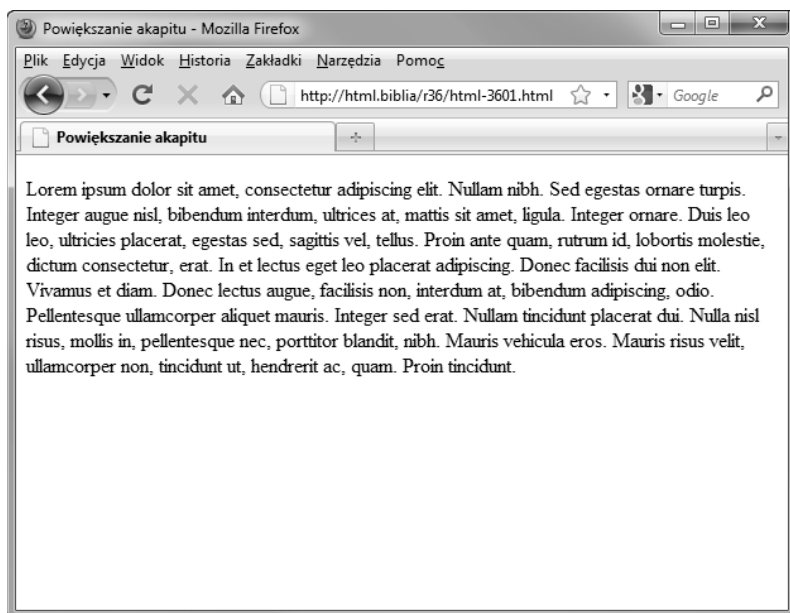
Powyższe wyrażenie może zostać użyte także do określenia nowej wartości właściwości. Na przykład poniższa instrukcja przypisuje właściwości `font-size` obiektu `bigText` wartość `xx-large`:

```
bigText.style.fontSize = "xx-large";
```

Przeanalizuj kod poniższego przykładu. Kiedy klikniemy wyświetlony na stronie akapit tekstu, procedura obsługi zdarzeń `onClick` wywoła funkcję `SuperSizeMe()`, która z kolei przypisuje właściwości `font-size` akapitu wartość `xx-large` (przez co tekst akapitu zostaje powiększony). Początkowy wygląd strony został przedstawiony na rysunku 36.1, natomiast rysunek 36.2 przedstawia tę samą stronę po kliknięciu akapitu.

Rysunek 36.1.

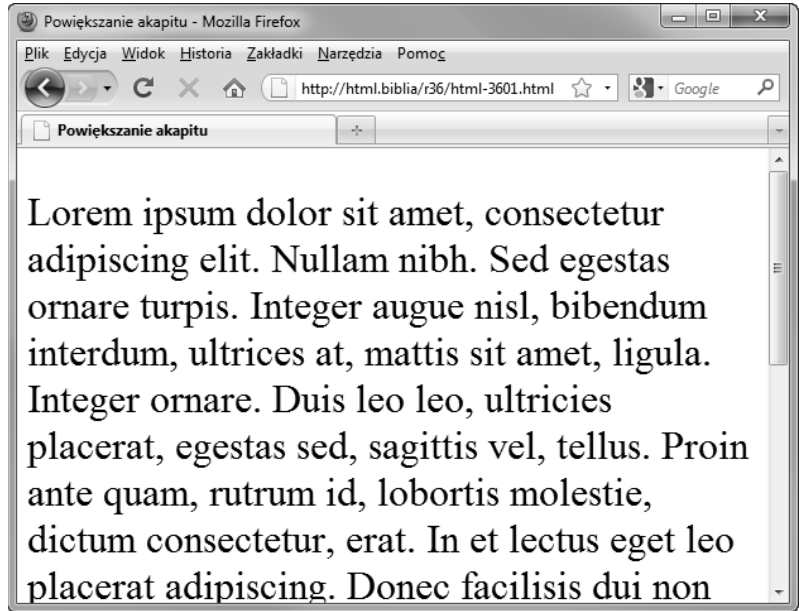
Początkowo, przed kliknięciem, tekst jest wyświetlony czcionką o średniej wielkości (medium)



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
```

Rysunek 36.2.

Po kliknięciu wielkość czcionki zostaje zmieniona na bardzo dużą (*xx-large*)



```

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Powiększanie akapitu</title>
<style type="text/css">
  #bigText { font-size: medium; }
</style>

<script type="text/javascript">

  function SuperSizeMe(obj) {
    obj.style.fontSize = "xx-large";
  }
</script>
</head>

<body>

  <p id="bigText" onClick="SuperSizeMe(this);">Lorem ipsum dolor sit amet, consectetur
  adipiscing elit. Nullam nibh. Sed egestas ornare turpis. Integer augue nisl,
  bibendum interdum, ultrices at, mattis sit amet, ligula. Integer ornare. Duis leo
  leo, ultricies placerat, egestas sed, sagittis vel, tellus. Proin ante quam,
  rutrum id, lobortis molestie, dictum consectetur, erat. In et lectus eget leo
  placerat adipiscing. Donec facilisis dui non elit. Vivamus et diam. Donec lectus
  augue, facilisis non, interdum at, bibendum adipiscing, odio. Pellentesque
  ullamcorper aliquet mauris. Integer sed erat. Nullam tincidunt placerat dui. Nulla
  nisl risus, mollis in, pellentesque nec, porttitor blandit, nibh. Mauris vehicula
  eros. Mauris risus velit, ullamcorper non, tincidunt ut, hendrerit ac, quam. Proin
  tincidunt.</p>
</body>
</html>

```

A teraz musimy poświęcić trochę czasu na to, by dobrze przeanalizować i zrozumieć, jak działa powyższy przykład. Można sądzić, że kolekcja `style` zapewnia dostęp do stylów

przypisanych danemu elementowi, niezależnie od tego, skąd one pochodzą. Okazuje się jednak, że tak nie jest. Kolekcja `style` zapewnia możliwość odwoływania się i operowania na atrybucie `style` umieszczonym w znaczniku obiektu. Oznacza to, że wykonanie poniższego fragmentu kodu JavaScript bezpośrednio po wyświetleniu naszej przykładowej strony w przeglądarce spowodowałoby wyświetlenie okienka informacyjnego z napisem „null”:

```
alert(document.getElementById("bigText").style.fontSize);
```

Właściwość `style.fontSize` ma wartość `null`, ponieważ w elemencie `bigText` nie został określony atrybut `style`. W jaki sposób zatem działa powyższy przykład, skoro nie zmienia on wartości atrybutu `font-size` określonego w arkuszu stylów umieszczonym w elemencie `<style>` w sekcji nagłówka strony? Odpowiedź jest prosta — wcale nie musi on zmieniać wartości właściwości określonych w elemencie `<style>`, zmienia on wartości przechowywane w atrybucie `style` elementu. Wartości te mają bowiem wyższy priorytet niż właściwości zdefiniowane w arkuszach stylów w sekcji nagłówka strony.

Oczywiście, gdyby wartości przechowywane w atrybucie `style` zostały podane, to można by odczytać ich wartości przy użyciu atrybutu `style`.

Aby odczytać właściwości podane w sekcji `<style>`, należy użyć jednej z dwóch metod: jedna z nich działa w przeglądarkach Internet Explorer, a druga w przeglądarkach Firefox. W pierwszym przypadku korzystamy z udostępnianej przez Internet Explorera właściwości `currentStyle`; z kolei w przeglądarce Firefox obiekt `window` udostępnia metodę `getComputedStyle`.

Korzystanie z właściwości `currentStyle` w Internet Explorerze jest wyjątkowo proste — wystarczy odszukać interesujący element strony, używając w tym celu jego identyfikatora, a następnie skorzystać z właściwości, by pobrać wartość odpowiedniego stylu. Pokazano to na poniższym przykładzie:

```
obj = document.getElementById(id);  
value = obj.currentStyle['fontSize'];
```

Należy zwrócić uwagę, że nazwa stylu jest zapisana bez łącznika, zatem ma ona postać `fontSize`, a nie `font-size`.

W przeglądarce Firefox należy wykonać dodatkowy, pośredni krok, gdyż wywołanie metody `getComputedStyle()` zwraca kolekcję, z której dopiero można odczytać interesującą nas wartość stylu. Ten etap pośredni wymaga użycia metody `getPropertyValue()`. Cały proces wygląda następująco:

```
obj = document.getElementById(id);  
objstyles = window.getComputedStyle(obj,null);  
value = objstyles.getProperty('font-size');
```

Warto zauważyć, że w tym rozwiązaniu stosowane są standardowe nazwy właściwości CSS, a nie ich przekształcone wersje bez łączników. Niemniej jednak w obu przedstawionych przypadkach efekt będzie taki sam — w zmiennej `value` zostanie zapisana wartość właściwości `font-size`.

Dzięki wykorzystaniu niezbyt skomplikowanego kodu określającego rodzaj używanej przeglądarki istnieje możliwość połączenia obu przedstawionych wcześniej rozwiązań

i zaimplementowania ich w postaci jednej funkcji. Poniższy listing przedstawia kod funkcji, która zwraca wartość stylu na podstawie przekazanej w wywołaniu nazwy elementu oraz nazwy właściwości CSS (przy czym używana jest tu prawidłowa nazwa właściwości, a nie nazwa zapisywana bez łącznika):

```
// Funkcja zwraca wartość właściwości CSS o nazwie propName
// odczytanej z elementu określonego przy użyciu identyfikatora id

function getStyleVal (id, propName) {
    // Czy w ogóle możemy cokolwiek zrobić [czy uda się nam
    // pobrać obiekt elementu przy użyciu metody getElementById()]?
    if (obj = document.getElementById(id)) {
        // Czy dostępna jest właściwość currentStyle (IE)?
        if (obj.currentStyle) {
            // Konwertujemy nazwę właściwości na format używany w IE
            if (propName.indexOf("-") != -1) {
                hyp = propName.indexOf("-");
                propName = propName.substr(0,hyp) +
                    propName.charAt(hyp+1).toUpperCase() +
                    propName.substr(hyp+2);
            }
            return obj.currentStyle[propName];
        }

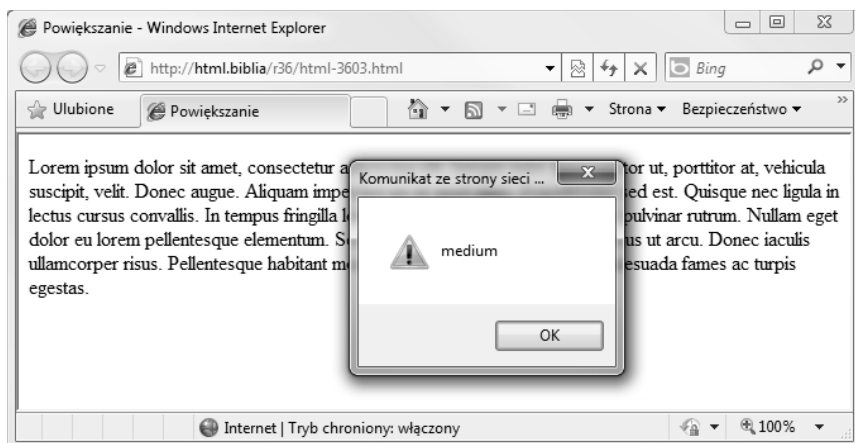
        // Czy jest dostępna metoda getComputedStyle (Mozilla)?
        if (window.getComputedStyle) {
            compStyle = window.getComputedStyle(obj,null);
            return compStyle.getPropertyValue(propName);
        }
    } // Koniec instrukcji if (obj == document.getElementById(id))

    // Nie udało się odszukać elementu — zwracamy pusty łańcuch znaków
    return "";
}
```

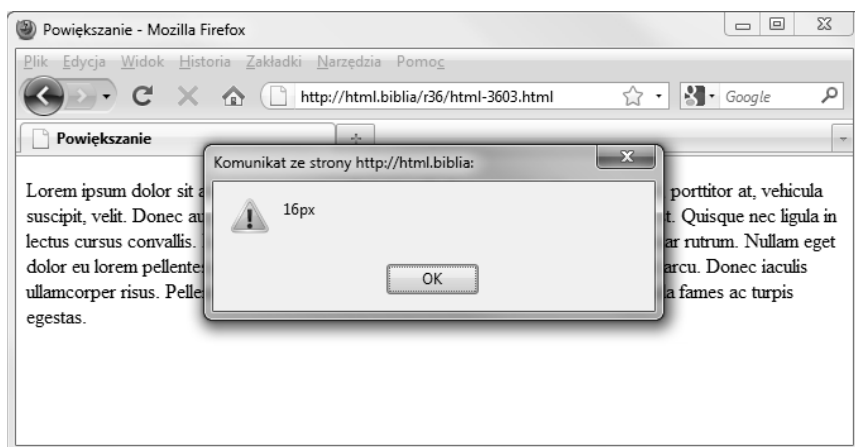
Zwróć uwagę, że w pierwszej kolejności funkcja sprawdza, czy używana przeglądarka udostępnia metodę `document.getElementById()` (wszystkie nowoczesne przeglądarki ją udostępniają). Jeśli metoda ta nie jest dostępna, funkcja kończy działanie, zwracając pusty łańcuch znaków. Następnie funkcja sprawdza, z jakiego sposobu odczytu wartości stylu może skorzystać — tego stosowanego w przeglądarkach Internet Explorer czy tego z przeglądarek Mozilli. Korzystając z jednego z tych dwóch sposobów, funkcja odczytuje wartość podanej właściwości stylu i zwraca ją. Dodatkowo, w razie potrzeby, funkcja konwertuje nazwę właściwości na format stosowany w przeglądarkach IE. Praktyczne wyniki wykorzystania tej funkcji zostały przedstawione na rysunkach: 36.3 (w Internet Explorerze) oraz 36.4 (w przeglądarce Firefox). W obu przypadkach użyto tej samej strony, zawierającej następujący fragment kodu:

```
<style type="text/css">
    #bigText { font-size: medium; }
</style>
...
<p id="bigText" onClick="alert(getStyleVal('bigText','font-size')):">Lorem
ipsum dolor sit amet, consetetur adipiscing elit, ...
```

Po kliknięciu tekstu wyświetlonego na stronie zostanie wyświetlone informacyjne okienko dialogowe, a w nim informacje o wartości właściwości `font-size` we wskazanym akapicie.



Rysunek 36.3. Funkcja `getStyleVal()` została tu zastosowana do wyświetlenia wartości właściwości `font-size` w przeglądarce Internet Explorer



Rysunek 36.4. Funkcja `getStyleVal()` została tu zastosowana do wyświetlenia wartości właściwości `font-size` w przeglądarce Firefox



Warto zwrócić uwagę, że w przypadku wykonywania funkcji `getStyleVal()` w przeglądarce Firefox zwraca ona wartość bezwzględną wyrażoną w pikselach. Ze względu na sposób działania tej funkcji może się okazać, że zwracane przez nią wartości będą zapisane w innym formacie niż ten, jakiego użyto do określania danej właściwości stylu. Na przykład, mimo że kolor jakiegoś elementu określisz w stylu przy użyciu wartości szesnastkowej `#FFA500`, funkcja zwróci nazwę koloru — `orange`. Albo, jak w zaprezentowanym wcześniej przykładzie, zamiast wielkości względnych (`medium`) — bezwzględne wartości wyrażone w pikselach.

A zatem dlaczego nie można by używać dwóch przedstawionych wcześniej rozwiązań do manipulowania stylami? Wytlumaczenie jest proste — obie pozwalają jedynie na odczyt wartości stylów. Wartości stylów można podawać bezpośrednio, w sposób przedstawiony na początku tego podrozdziału.

Użyteczne operacje z użyciem CSS

Przykłady przedstawione we wcześniejszej części rozdziału pokazały, w jaki sposób można operować na właściwościach związanych z postacią czcionki. Choć są to całkiem przydatne operacje, to jednak bardziej złożone manipulacje właściwościami CSS z poziomu kodu JavaScript mogą pozwolić nam tworzyć jeszcze bardziej atrakcyjne efekty wizualne. W tej części rozdziału znajdziesz kilka przykładów, które możesz wykorzystać jako punkt startowy do dalszych, własnych eksperymentów.

Ukrywanie i wyświetlanie tekstu

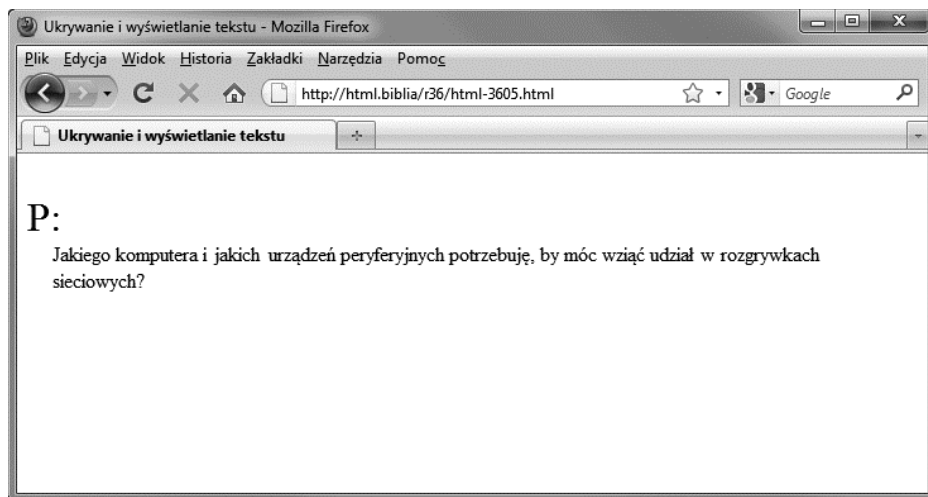
Dzięki wykorzystaniu kaskadowych arkuszy stylów i JavaScriptu bez większych problemów można ukrywać i wyświetlać teksty oraz inne elementy stron. Rozwiązanie takie może być wykorzystane w przeróżnych sytuacjach, takich jak tworzenie rozwijanego menu bądź ukrywanie fragmentu strony do momentu, gdy użytkownik zdecyduje się go wyświetlić. Na przykład przeanalizujmy stronę zawierającą listę pytań i odpowiedzi. Można sądzić, że użytkownik nie będzie chciał oglądać całej listy odpowiedzi, już bezpośrednio po wyświetleniu strony. Jest bardziej prawdopodobne, że będzie chciał selektywnie wyświetlać odpowiedzi tylko na te pytania, które go interesują. Dzięki zastosowaniu właściwości CSS `display` napisanie skryptu zapewniającego takie możliwości funkcjonalne nie nastęrcza żadnych problemów.

Przeanalizuj poniższy kod strony, w którym odpowiedzi na pytania są początkowo ukryte, a przeglądarka wyświetli je dopiero po kliknięciu dużej litery „P” lub tekstu pytania. Rysunek 36.5 przedstawia stronę bezpośrednio po wyświetleniu (z ukrytą odpowiedzią), natomiast rysunek 36.6 — stronę po wyświetleniu odpowiedzi.

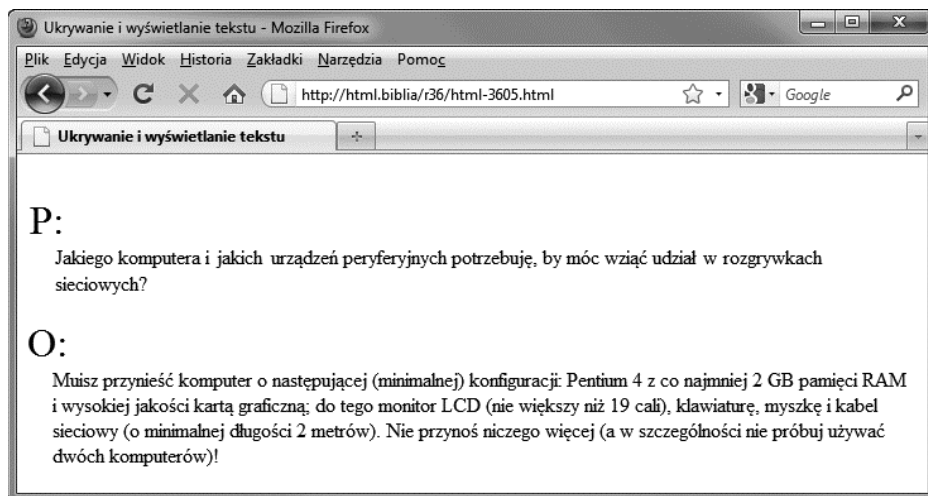
```
!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Ukrywanie i wyświetlanie tekstu</title>
<style type="text/css">

/* Początkowo ukrywamy wszystkie odpowiedzi */
.hideseek { display: none; }

/* Różne stany widoczności pytań i odpowiedzi */
.Q { font-size: xx-large;
padding-bottom: 0;
margin-bottom: 0;
cursor: pointer; }
.Qtext { margin-left: 20px;
margin-top: 0;
padding-top: 0; }
.A { font-size: xx-large;
padding-bottom: 0;
margin-bottom: 0;
clear: left; }
```



Rysunek 36.5. Odpowiedzi na pytania są początkowo ukryte



Rysunek 36.6. Odpowiedź jest wyświetlana po kliknięciu dużej litery P (bądź dowolnego innego fragmentu pytania). Ponowne kliknięcie powoduje ukrycie odpowiedzi

```
.Atext { margin-left: 20px;
        margin-top: 0;
        padding-top: 0; }
```

```
</style>
```

```
<script type="text/javascript">
```

```
    // Naprzemiennie ukrywamy i wyświetlamy element
    function hideseek(id) {
        obj = document.getElementById(id);
        // Jeśli styl nie jest określony, to możemy założyć,
        // że element nie był jeszcze modyfikowany i jest niewidoczny
```



```
        if ((obj.style.display == "" ||
            (obj.style.display == "none"))) {
            obj.style.display = "block";
        } else {
            obj.style.display = "none";
        }
    }
}

</script>
</head>
<body>
  <div onClick="hideseek('A1');">
    <p class="Q" >P:</p>
    <p class="Qtext">Jakiego komputera i urządzeń peryferyjnych potrzebuję, by móc
    wziąć udział w rozgrywkach sieciowych?</p>
  </div>
  <div id="A1" class="hideseek">
    <p class="A" >O:</p>
    <p class="Atext">
      Musisz przynieść komputer o następującej (minimalnej) konfiguracji: Pentium 4
      z co najmniej 2 GB pamięci RAM i wysokiej jakości kartą graficzną; do tego
      monitor LCD (nie większy niż 19 cali), klawiaturę, myszkę i kabel sieciowy
      (o minimalnej długości 2 metrów). Nie przynoś niczego więcej
      (a w szczególności nie próbuj używać dwóch komputerów)!
    </p>
  </div>
</body>
</html>
```

Powyższy przykład wykorzystuje funkcję JavaScript, która odczytuje bieżącą wartość właściwości `display` i ustawia ją na wartość przeciwną, co powoduje naprzemienne wyświetlanie i ukrywanie elementu. Funkcja ta jest wywoływana przez procedurę obsługi zdarzeń `onClick`, zdefiniowaną w elemencie `<div>` zawierającym pytanie. W wywołaniu funkcji umieszczony jest identyfikator elementu `<div>` zawierającego odpowiedź, dzięki czemu wie ona, na jakim elemencie ma operować.

Jak już wcześniej wspomniano, takie rozwiązanie może być stosowane w przeróżnych celach. Wystarczy ukryć element, który początkowo ma być niewidoczny (wykorzystując do tego celu właściwość `display` z wartością `none`), a następnie wyświetlać go przy użyciu funkcji wywoływanej po kliknięciu przycisku lub zajściu jakiegokolwiek innego zdarzenia.

Powiększanie obrazków

Innym często spotykanym zastosowaniem możliwości CSS do tworzenia ciekawych efektów wizualnych jest powiększanie obrazków, które początkowo są wyświetlane w postaci miniaturki. Technika ta jest powszechnie wykorzystywana we wszelkiego typu galeriach internetowych lub w innych witrynach, na których wyświetlanie obrazków w pełnej wielkości jest pożądanym, lecz kosztownym.

Prezentowana tu technika jest podobna do rozwiązania przedstawionego w poprzednim punkcie rozdziału — także w tym przypadku generowane przez użytkownika zdarzenie wywołuje skrypt, który odpowiednio modyfikuje właściwości CSS danego elementu.

W prezentowanym przykładzie skrypt operuje na właściwości `display` miniatury oraz pełnowymiarowego obrazka. Poniżej przedstawiony został kod przykładowej strony WWW:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Powiększanie obrazków</title>
  <style type="text/css">
    .zoom { display: none; margin: 0px 10px;
            float: left;}
    .zoom p {margin: 0;}
    .thumb { display: block; margin: 0px 10px;
            float: left; }
    .thumb p {margin: 0;}
  </style>
  <script type="text/javascript">
    function PicZoom(id) {
      pic = document.getElementById(id);
      thum = document.getElementById("T"+id);
      if ((pic.style.display == "") ||
          (pic.style.display == "none")) {
        pic.style.display = "block";
        thum.style.display = "none";
      } else {
        pic.style.display = "none";
        thum.style.display = "block";
      }
    }
  </script>
</head>

<body>

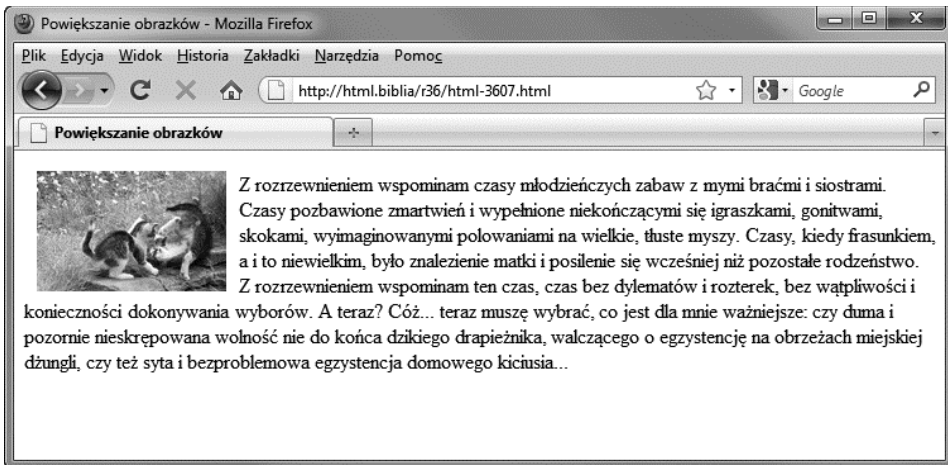
  <div id="1" class="zoom"><p> </p></div>
  <div id="T1" class="thumb"><p></p></div>
  <div class="text"><p>
    Z rozrzewnieniem wspominam czasy młodości z moimi braćmi i siostrami.
    Czasy pozbawione zmartwień i wypełnione niekończącymi się igraszkami,
    gonitwami, skokami, wyimaginowanymi polowaniami na wielkie, tłuste
    myszy. Czasy, kiedy frasunkiem, a i to niewielkim, było
    znalezienie matki i posilenie się wcześniej niż pozostałe
    rodzeństwo. Z rozrzewnieniem wspominam ten czas, czas bez
    dylematów i rozterek, bez wątpliwości i konieczności
    dokonywania wyborów. A teraz? Cóż... teraz muszę wybrać,
    co jest dla mnie ważniejsze: czy duma i pozornie nieskrępowana
    wolność nie do końca dzikiego drapieżnika, walczącego o
    egzystencję na obrzeżach miejskiej dżungli, czy też syta i
    bezproblemowa egzystencja domowego kiciusia...
  </p></div>
</body>
</html>

```

Aby ułatwić formatowanie strony i zapewnić większą elastyczność, wszystkie obrazki, zarówno miniaturka, jak i obrazek pełnowymiarowy, zostały umieszczone w elementach `<div>`. Właściwość `display` elementu `<div>` zawierającego miniaturkę ma początkowo wartość `block`, co sprawia, że element ten jest widoczny. Natomiast w elemencie `<div>`

zawierającym pełnowymiarowy obrazek właściwość `display` ma początkowo wartość `none`, a zatem element ten jest początkowo ukryty. W elemencie `img` prezentującym miniaturkę zdefiniowano procedurę obsługi zdarzeń `onMouseOver`, w której wywoływana jest funkcja `PicZoom()`. Dzięki temu funkcja ta zostanie wywołana w momencie umieszczenia wskaźnika myszy na miniaturce. Działanie funkcji `PicZoom()` polega na zamianie wartości właściwości `display` elementów `<div>` zawierających obrazki; innymi słowy, pojawi się pełnowymiarowy obrazek, a miniaturka zniknie. Wskaźnik myszy pozostanie w tym samym położeniu — w obszarze pełnowymiarowego zdjęcia. Kiedy użytkownik przesunie go poza obrazek, zostanie zgłoszone zdarzenie `onMouseOut`, a w efekcie ponownie zostanie wywołana funkcja `PicZoom()`, która tym razem ukryje pełnowymiarowy obrazek i wyświetli miniaturkę.

Działanie tej strony zostało zilustrowane na rysunkach 36.7 oraz 36.8. Pierwszy z nich przedstawia stronę w jej początkowej postaci, z widoczną miniaturką; natomiast na rysunku 36.8 widoczna jest strona po wyświetleniu obrazka pełnowymiarowego.



Rysunek 36.7. Początkowo, po wyświetleniu strony, widoczna jest miniaturka, natomiast pełnowymiarowy obrazek jest ukryty



Podobnie jak w pozostałych przykładach przedstawionych w tym rozdziale, podobny efekt można uzyskać na wiele sposobów. Jeden z nich polega na umieszczeniu wszystkich obrazków na jednym dużym i przesuwanie go w taki sposób, by w danym momencie był widoczny jego odpowiedni fragment. Inne rozwiązanie polega na modyfikowaniu atrybutu `src` znacznika `` tak, by wyświetlany był w nim odpowiedni obrazek. Istnieje także bardziej złożona technika, bazująca na wykorzystaniu warstw i odpowiednich modyfikacjach właściwości `z-index` wybranego elementu. Nic nie stoi na przeszkodzie, byś wymyślił także swoje własne rozwiązanie.

Podmieniane menu

We wszystkich przykładach przedstawionych we wcześniejszej części rozdziału uzyskiwaliśmy zamierzony efekt wizualny, stosując kod JavaScript do modyfikowania właściwości CSS. Jak na razie nie skorzystaliśmy z pseudoklas CSS dostępnych w elementach łączy, które pozwalają na uzyskanie podobnych efektów.



Rysunek 36.8. Po umieszczeniu wskaźnika myszy na miniaturce, zamiast niej w przeglądarce jest wyświetlany obrazek w pełnych wymiarach

Pseudoklasy elementów <a> przedstawione zostały w tabeli 36.1.

Tabela 36.1. Pseudoklasy elementu <a>

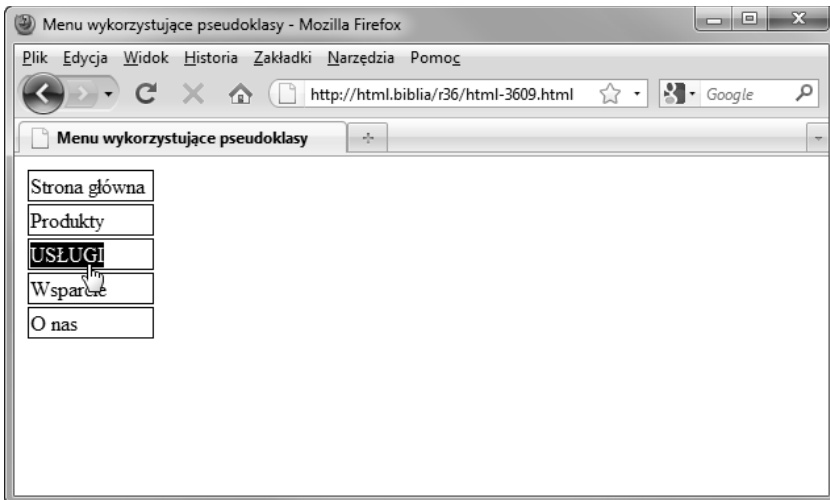
Pseudoklasa	Zastosowanie lub efekt
:link	Formatuje elementy wybrane przez selektor jako łącza, które jeszcze nie zostały odwiedzone.
:visited	Formatuje elementy wybrane przez selektor jako odwiedzone łącza.
:hover	Formatuje elementy wybrane przez selektor w momencie umieszczenia na nich wskaźnika myszy.
:active	Formatuje elementy wybrane przez selektor jako aktywne łącza.

Pseudoklasy wymienione w tabeli 36.1 są zazwyczaj stosowane do określania wyglądu elementów traktowanych jak dynamiczne łącza. Na przykład, korzystając z pseudoklasy :hover, można dynamicznie zmieniać wygląd elementu w momencie umieszczenia nad nim wskaźnika myszy — czyli dokładnie tak samo, jak zachowują się łącza.

Na przykład przedstawiony poniżej dokument HTML wykorzystuje pseudoklasę :hover, by zmieniać wygląd łączy po wskazaniu ich myszą. Uzyskany efekt przypomina dynamicznie modyfikowane menu tworzone przy użyciu kodu JavaScript. Efekty jego działania zostały przedstawione na rysunku 36.9.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Menu wykorzystujące pseudoklasy</title>
  <style type="text/css">
    .nav tr td {border: 1px solid black;}
    .menu { color: black;
            background-color: white;
            text-transform: none;
            text-decoration: none; }
    .menu:hover { color: white;
                  background-color: black;
                  text-transform: uppercase;
                  text-decoration: none; }
    .menucase { width: 100px; }
  </style>
</head>

<body>
  <div class="menucase">
    <table border="0" width="100%" class="nav">
      <tr><td><a class="menu" href="index.html">Strona główna</a></td></tr>
      <tr><td><a class="menu" href="produkty.html">Produkty</a></td></tr>
      <tr><td><a class="menu" href="uslugi.html">Usługi</a></td></tr>
      <tr><td><a class="menu" href="wsparcie.html">Wsparcie</a></td></tr>
      <tr><td><a class="menu" href="onas.html">O nas</a></td></tr>
    </table>
  </div>
</body>
</html>
```



Rysunek 36.9. Pseudoklasy mogą być stosowane do tworzenia dynamicznego menu, w tym przypadku wskazany element menu jest podświetlany

Należy zauważyć, że choć technika polegająca na zastosowaniu pseudoklas w celu uzyskania ciekawych efektów wizualnych jest bardzo popularna, to jednak nie jest ona zgodna z zasadą separacji działań od prezentacji. Dlatego też znacznie lepszym rozwiązaniem jest tworzenie podobnych efektów przy wykorzystaniu kodu JavaScript.



Zwróć uwagę, że opisana tu technika może być stosowana w połączeniu niemal ze wszystkimi elementami HTML. Jednak pseudoklas można używać wyłącznie do formatowania znaczników łącza, a zatem by odpowiednio formatować łącza w dokumencie, należy korzystać z innych znaczników.

Podsumowanie

W tym rozdziale przedstawione zostały technologie DHTML i CSS. Dowiedziałeś się, jak używać JavaScriptu, by modyfikować style CSS elementów i uzyskiwać ciekawe efekty wizualne, oraz jak tworzyć takie efekty przy użyciu pseudoklas CSS. W następnym rozdziale zamieszczone zostały informacje dotyczące stosowania kaskadowych arkuszy stylów do określania postaci stron przeznaczonych do wydruku. W kolejnych rozdziałach, od 38. do 41., przedstawione zostały bardziej specjalistyczne zagadnienia związane ze stosowaniem kaskadowych arkuszy stylów.