helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

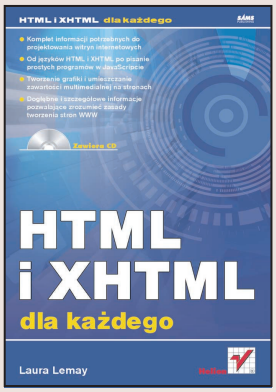
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

HTML i XHTML dla każdego



Autor: Laura Lemay
Tłumaczenie: Łukasz Zieliński (rozdz. 1 – 5, 13, 14),
Marzena Baranowska (rozdz. 6, 7 – 12, 15, 16)
ISBN: 83-7361-239-4
Tytuł oryginału: Teach Yourself Web Publishing with HTML and XHTML
Format: B5, stron: 464

Prawdopodobnie największą zaletą WWW jest to, że nie trzeba prowadzić wielkiego przedsiębiorstwa, aby udostępniać informacje i produkty czytelnikom i klientom na całym świecie. Wystarczy komputer z dostępem do internetu i chęć uczenia się. Skoro czytasz te słowa, prawdopodobnie i Ty chciałbyś zaistnieć w Sieci. Pytanie brzmi: od czego zacząć?

W internecie można znaleźć samouczki, instrukcje, mnóstwo przykładów i darmowych narzędzi mających ułatwić publikowanie na WWW. Jest też wiele innych książek na ten temat. Przewaga książki „HTML i XHTML dla każdego" bierze się z faktu, że wszystkie materiały są dostępne w jednym miejscu: informacje potrzebne do opanowania języka HTML, umieszczania stron na serwerze, tworzenia grafiki na potrzeby WWW i utrzymywania funkcjonalności i sprawności swojej witryny. Znajdziesz tu podpowiedzi, sugestie i przykłady pokazujące, jak projektować całościową strukturę witryny, a nie tylko układ słów na pojedynczych stronach. Ta książka nie uczy jak stworzyć serwis WWW – ona uczy jak stworzyć dobrą i nieprzeciętną stronę internetową.

Książka opisuje między innymi:

- Podstawową strukturę strony WWW
- Tworzenie łączy pomiędzy stronami
- Formatowanie tekstu za pomocą HTML-a i CSS
- Użycie tabel, projektowanie formularzy
- Grafikę i multimedia na stronach WWW
- Podstawy języków JavaScript i DHTML
- Publikowanie witryny na serwerze

Książka nie koncentruje się na konkretnej platformie systemowej. Zawarte w niej informacje zachowują swoją przydatność niezależnie od tego, czy używa się komputera PC i systemu Windows, Macintosha, którejś z odmian Uniksa, czy też jeszcze innego systemu. Uzyskaną tu wiedzę będzie można stosować do rozwijania swoich stron WWW przy użyciu dowolnie wybranego systemu.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

O Autorach **13**

Wprowadzenie **15**

Rozdział 1. Wprowadzenie do języka HTML **19**

Język HTML 19
Język HTML opisuje strukturę strony 20
Język HTML nie opisuje układu strony 20
Dlaczego to działa w ten sposób 21
Język HTML jest językiem oznaczeń 22
Krótka historia znaczników HTML 22
Aktualny standard: XHTML 1.0 23
Jak wygląda język HTML 24
Uwaga o formatowaniu 28
Używanie kaskadowych arkuszy stylów 28
Umieszczanie atrybutów w znacznikach 29
Programy przydatne w pisaniu w języku HTML 30
Podsumowanie 31

Rozdział 2. Rozwój od podstaw **33**

Tworzenie struktury kodu HTML 33
Znacznik <html> 34
Znacznik <head> 34
Znacznik <body> 35
Tytuł 36
Nagłówki 37
Akapity 39
Listy 40
Znaczniki tworzące listy 41
Listy numerowane 42
Dostosowywanie list numerowanych 43
Listy punktowane 46
Dostosowywanie list punktowanych 46
Listy definicji 49
Zagnieżdżanie list 50
Komentarze 51
Podsumowanie 52

Rozdział 3. Wszystko o łączach 55
Jak tworzyć łącza 55
<a> — znacznik tworzący łącza 55
Łączenie stron lokalnych ścieżkami względnymi i bezwzględnymi 60
Ścieżki bezwzględne 61
Które ścieżki są lepsze — względne czy bezwzględne? 62
Łącza do innych dokumentów w sieci 63
Łącza do fragmentów stron 67
Jak tworzyć łącza i zakotwiczenia 68
Podłączanie zakotwiczeń wewnątrz dokumentu 73
Anatomia adresu URL 74
Części adresu URL 74
Znaki specjalne w adresach URL 75
Rodzaje adresów URL 76
Protokół HTTP 76
Anonimowy dostęp do serwerów FTP 77
Dostęp do serwerów FTP z podaniem nazwy użytkownika 78
Adresy URL typu mailto 78
Grupy dyskusyjne Usenet 79
Plikowe adresy URL 80
Podsumowanie 81

Rozdział 4. Formatowanie tekstu 83
Elementy poziomu znakowego 84
Znaczniki stylów logicznych 84
Znaczniki stylów fizycznych 86
Formatowanie znaków właściwościami CSS 87
Ozdabianie tekstu — właściwość text-decoration 88
Właściwości czcionek 89
Zachowywanie układu tekstu 90
Kreski poziome 92
Atrybuty znacznika <hr> 93
Wymuszanie podziału wiersza 95
Sygnaturki 97
Akapity cytowane 98
Znaki specjalne 99
Składnia encji znakowych 99
Zapis znaków zarezerwowanych 100
Wyrównywanie tekstu 102
Wyrównywanie pojedynczych elementów 102
Wyrównywanie wielu elementów jednocześnie 103
Kroje pisma i wielkość czcionki 104
Zmiana wielkości czcionki 105
Zmiana kroju pisma 106
Modyfikacja czcionek arkuszami stylów 107
Znaczniki <nobr> oraz <wbr> 108
Podsumowanie 118

Rozdział 5. Obrazki, kolory i tła 123
Obrazki na stronach WWW 123
Formaty plików graficznych 124
Format GIF 125
Format JPEG 125
Format PNG 126

Obrazki w tekście: znacznik <img> ... 126
Dostarczanie etykiet rezerwowych ... 127
Obrazki a tekst ... 130
Wyrównywanie obrazków i tekstu ... 131
Zawijanie tekstu przy obrazkach ... 133
Dopasowywanie odstępów wokół obrazków ... 136
Obrazki a łącza ... 137
Inne przydatne umiejętności ... 141
Wymiary obrazków i skalowanie ... 141
Jeszcze o obramowaniach obrazków ... 142
Stosowanie koloru ... 143
Nazywanie kolorów ... 143
Zmienianie koloru tła ... 144
Zmienianie koloru tekstu ... 145
Zmienianie koloru znaków ... 146
Określanie kolorów właściwościami CSS ... 147
Ozdabianie tła tapetą ... 147
Podpowiedzi dodatkowe ... 149
Czy ten obrazek naprawdę jest potrzebny? ... 150
Małe jest piękne ... 150
Kulturalne zamieszczanie obrazków ... 151
Podsumowanie ... 151

Rozdział 6. Tabele ... 153
Tworzenie tabel ... 154
Elementy składowe tabeli ... 154
Element <table> ... 155
Krótki opis tabeli ... 155
Wiersze i komórki ... 156
Puste komórki ... 157
Tytuły ... 159
Zmiana rozmiaru tabel, obramowania i komórek ... 162
Ustawianie szerokości tabeli ... 162
Zmiana obramowania tabeli ... 163
Marginesy komórek ... 164
Odstępy między komórkami ... 164
Szerokość kolumn ... 165
Ustawianie podziału wierszy ... 166
Kolor tabeli i komórek oraz wyrównanie ... 167
Zmiana koloru tabeli oraz tła komórek ... 168
Zmiana koloru obramowania ... 169
Wyrównanie zawartości tabeli ... 171
Wyrównanie tabeli ... 171
Wyrównanie zawartości komórek ... 172
Wyrównanie tytułu ... 173
Scalanie komórek ... 174
Zaawansowane sposoby ulepszania tabel ... 182
Grupowanie i wyrównanie kolumn ... 182
Grupowanie i wyrównanie wierszy ... 185
Atrybuty frame i rules ... 187
Inne elementy i atrybuty tabel ... 188
Podsumowanie ... 188

Rozdział 7. Formatowanie stron za pomocą kaskadowych arkuszy stylów..... 193
Uaktywnianie arkuszy stylów na stronie 193
Tworzenie arkuszy stylów na poziomie strony 194
Tworzenie arkuszy stylów na poziomie witryny 194
Selektory 195
Selektory kontekstowe 196
Klasy i identyfikatory 197
Jednostki miar 198
Właściwości pól 199
Sterowanie rozmiarem 200
Obramowanie 200
Marginesy i wypełnienie 202
Elementy pływające 206
Pozycjonowanie CSS 210
Pozycjonowanie względne 211
Pozycjonowanie bezwzględne 212
Nakładanie elementów 215
Modyfikowanie wyglądu tabeli 217
Znacznik <body> 219
Łącza 220
Tworzenie układów z wieloma kolumnami 221
Podsumowanie 224

Rozdział 8. Projektowanie formularzy 225
Forma i funkcja formularzy 226
Zastosowanie etykiety <form> 230
Tworzenie elementów formularza za pomocą znacznika <input> 232
Tworzenie pól tekstowych 232
Tworzenie pola hasła 233
Tworzenie przycisków wysyłających 234
Tworzenie przycisków zerowania 235
Tworzenie pól wyboru 235
Tworzenie przycisków opcji 236
Użycie obrazów jako przycisków wysyłających 236
Tworzenie przycisków rodzajowych 237
Ukryte pola formularza 237
Pole pobierania pliku 238
Użycie innych elementów kontrolnych 238
Użycie elementu button 239
Tworzenie dużych pól tekstowych za pomocą elementu textarea 239
Tworzenie menu za pomocą elementów select i option 240
Elementy dodatkowe 245
Wyświetlanie etykiet za pomocą elementu label 245
Grupowanie elementów za pomocą elementów fieldset i legend 246
Zmiana domyślnej kolejności poruszania się po polach formularza 246
Użycie klawiszy dostępu 247
Tworzenie elementów nieaktywnych i tylko do odczytu 247
Zastosowanie kaskadowych arkuszy stylów 248
Planowanie formularzy 253
Podsumowanie 254

Rozdział 9. Mapy obrazów i grafiki animowane 257
Czym są mapy obrazów? 257
Mapy obrazów działające po stronie klienta 258
Mapy obrazów a przeglądarki tekstowe 259

Tworzenie map obrazów działających po stronie klienta ... 259
Przygotowanie obrazu ... 259
Określanie współrzędnych ... 260
Znaczniki <map> i <area> ... 262
Atrybut usemap ... 264
Tworzenie przezroczystych plików GIF ... 268
Wybór koloru przezroczystości ... 268
Antyaliasowanie i przezroczystość ... 268
Tworzenie animowanych GIF-ów ... 269
Programy ułatwiające kompilację animowanych GIF-ów ... 269
Tworzenie animowanych GIF-ów ... 270
Szukanie optymalnej wielkości animacji ... 271
Programy ułatwiające tworzenie obrazów ... 274
Przydatne właściwości oprogramowania ... 275
Podsumowanie ... 279

Rozdział 10. Dodawanie dźwięków, wideo i innych elementów multimedialnych... 281
Sposoby prezentacji dźwięków oraz obrazów wideo ... 282
Stare, ale użyteczne rozwiązanie — dołączanie ... 283
Osadzanie dźwięków i obrazów wideo ... 290
Użycie elementu <embed> ... 291
Użycie elementu <object> ... 294
Łączenie elementów <embed> i <object> ... 295
Osadzanie animacji Flash ... 295
Osadzanie animacji Shockwave ... 296
Osadzanie RealAudio i RealVideo ... 296
Techniki multimedialne ... 297
Typy plików dźwiękowych i wideo ... 303
Odtwarzacze i moduły rozszerzające ... 306
Windows Media Player ... 306
Flash Player firmy Macromedia ... 307
Shockwave firmy Macromedia ... 308
QuickTime 6 firmy Apple ... 308
RealOne Player ... 309
WinAmp ... 310
Podsumowanie ... 310

Rozdział 11. JavaScript ... 313
Charakterystyka języka JavaScript ... 313
Zalety języka JavaScript ... 314
Znacznik <script> ... 315
Struktura skryptów pisanych w języku JavaScript ... 316
Atrybut src ... 317
Podstawowe polecenia i struktura języka ... 317
Właściwości i metody ... 318
Zdarzenia i JavaScript ... 320
Zmienne ... 322
Operatory i wyrażenia ... 322
Podstawy programowania w języku JavaScript ... 323
Czym jest program? ... 324
Więcej o programowaniu w JavaScript ... 326
Podsumowanie ... 326

Rozdział 12. Praca z JavaScript 329
Tworzenie generatora losowych połączeń 329
Weryfikacja danych w formularzach 337
Tworzenie podmienianych obrazów 343
Podsumowanie 346

Rozdział 13. Opracowywanie układów ramek i okien łączonych 347
Czym są ramki i w których przeglądarkach działają 347
Tworzenie okien łączonych 349
Znacznik <base> 352
Opracowywanie układów ramek 353
Znacznik <frameset> 354
Znacznik <frame> 357
Znacznik <noframes> 357
Określanie wyglądu rozgraniczeń 359
Dodatkowe atrybuty 360
Tworzenie złożonych układów ramek 360
Magiczne wartości atrybutu target 371
Ramki swobodne 372
Podsumowanie 375

Rozdział 14. Techniki DHTML 377
Czym właściwie są techniki DHTML? 378
Używanie obiektowego modelu dokumentu 380
Typy danych w obiektach DOM 380
Obiekty w strukturze DOM 381
Zastosowanie struktury DOM 381
Techniki DHTML a niespójność przeglądarek 386
Rozpoznawanie przeglądarek 386
Rozpoznawanie możliwości przeglądarki 388
Sprawdzanie obecności obiektów 389
Grupowanie elementów znacznikiem 389
Pozycjonowanie elementów 390
Manipulacja elementami za pomocą skryptów JavaScript 394
Kontynuacja nauki technik DHTML 401
Podsumowanie 402

Rozdział 15. Publikowanie witryny 403
Sposób działania i przeznaczenie serwera WWW 403
Co jeszcze potrafią serwery WWW? 404
Lokalizowanie serwera WWW 405
Korzystanie z serwera WWW w szkole lub w pracy 405
Korzystanie z usług komercyjnych 406
Zakładanie własnego serwera 407
Organizowanie i instalowanie plików HTML 407
Pytania do webmastera 407
Przechowywanie plików w folderach 408
Domyślny plik indeksu i poprawne nazwy plików 408
Publikowanie plików 409
Przemieszczanie plików pomiędzy systemami 410
Usuwanie błędów 413
Brak dostępu do serwera 413
Nie mam dostępu do plików 413
Nie mam dostępu do obrazów 414
Łącza nie działają poprawnie 414
Pliki nie są wyświetlane poprawnie 414

Rejestracja i reklama witryny ... 415
Listy witryn WWW ... 415
Yahoo! ... 415
dmoz: Open Directory Project ... 417
Żółte strony (Yellow Pages) ... 418
Prywatne serwisy katalogowe ... 419
Serwisy indeksujące i wyszukujące ... 419
Google ... 420
AltaVista ... 421
AlltheWeb.com ... 422
Narzędzia automatycznej rejestracji ... 422
Pierścienie internetowe (ringi) ... 422
Wizytówki, papeteria firmowa i broszury ... 423
Jak wpływać na przyjaciół i ludzi ... 423
Informacje o odwiedzających ... 424
Plik dziennika ... 424
Liczniki odwiedzin na stronie ... 425
Podsumowanie ... 426

Rozdział 16. Wykorzystywanie możliwości serwera ... 427
Aplikacje WWW ... 427
CGI ... 428
Active Server Pages ... 429
JSP/J2EE ... 431
PHP ... 433
Wstawki po stronie serwera ... 434
Użycie wstawek po stronie serwera ... 435
Użycie plików kontroli dostępu Apache ... 437
Zarządzanie dostępem do stron ... 438
Przekierowywanie użytkowników ... 440
Podsumowanie ... 441

Skorowidz ... 443

Rozdział 3.

Wszystko o łączach

Strony HTML, które stworzyliśmy w poprzednim rozdziale, są w porządku i ogólnie bez zarzutu, niestety — raczej nudnawe. Prawdziwa zabawa zaczyna się w momencie, gdy na stronach zaczynamy umieszczać łącza do innych witryn. W tym rozdziale zajmiemy się właśnie takimi sprawami. W szczególności omówimy:

- znacznik <a> tworzący łącza,
- łączenie stron znajdujących się na dysku lokalnym łączami względnymi i bezwzględnymi,
- tworzenie łączy do stron WWW za pomocą adresów URL,
- tworzenie łączy do wybranych fragmentów stron za pomocą zakotwiczeń,
- adresy URL, ich elementy i rodzaje.

Jak tworzyć łącza

Aby utworzyć łącze w języku HTML, potrzeba:

- nazwy lub adresu URL pliku, do którego ma prowadzić łącze;
- tekstu, który posłuży za punkt dostępu — tekst ten, wyświetlany z wyróżnieniem, wybrany przez użytkownika spowoduje otwarcie podłączonego pliku.

Z dwóch powyższych jedynie tekst jest widoczny na stronie. Kiedy użytkownik kilka ten tekst, przeglądarka wczytuje plik wskazany przez związany z nim adres URL.

<a> — znacznik tworzący łącza

Aby utworzyć łącze, w kodzie HTML umieszczamy znacznik <a>, nazywany również znacznikiem *zakotwiczenia* (ang. *anchor*; więcej o zakotwiczeniach w dalszej części tego rozdziału). W przeciwieństwie do prostych znaczników omówionych do tej pory, znacznik <a> ma kilka dodatkowych cech. Znacznik otwierający <a>, oprócz samej

nazwy znacznika, zawiera jeszcze parametry łącza. Te parametry nazywane są *atrybutami* znacznika (pojęcie „atrybutu" po raz pierwszy pojawiło się w rozdziale 1). W związku z tym znacznik otwierający `<a>` nie przybiera postaci samej nazwy, umieszczonej pomiędzy nawiasami trójkątnymi, lecz raczej podobną do takiej:

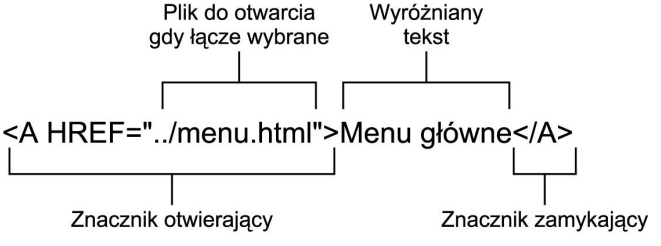
```
<a name="poczatek" href="menu.html" title="Żywoty cezarów">
```

Atrybuty (w tym przykładzie są to: `name`, `href` oraz `title`) służą do opisu łącza. Prawdopodobnie najczęściej przydawać się będzie atrybut `href` (od ang. *hypertext reference* — odwołanie hipertekstowe). Jego wartością jest nazwa lub adres URL podłączanego pliku.

Podobnie jak większość znaczników HTML, znacznik tworzący łącza ma także znacznik zamykający `</a>`. Cały tekst między znacznikiem otwierającym a zamykającym będzie ekranową reprezentacją łącza. Przy wyświetleniu zostanie wyróżniony, zwykle podkreśleniem i kolorem — niebieskim albo czerwonym. To właśnie ten tekst będą klikać użytkownicy, aby skierować przeglądarkę pod adres określony atrybutem `href`.

Rysunek 3.1 przedstawia części znacznika typowego łącza, takie jak atrybut `href`, tekst łącza i znacznik zamykający.

Rysunek 3.1.
Łącze na stronie WWW



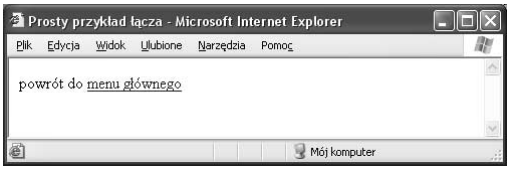
Poniżej inny prosty przykład kodu i prezentacji łącza.

źródło

```
powrót do <a href="menu.html">menu głównego</a>
```

wynik

Rysunek 3.2.
Łącze wyświetlane w przeglądarce



wykonaj

Ćwiczenie 3.1. Łączenie dwóch stron

Czas spróbować utworzyć proste łącze samodzielnie. Połączymy dwie strony HTML zapisane na dysku lokalnym. Do wykonania ćwiczenia posłużymy się edytorem tekstu oraz przeglądarką. Ponieważ oba pliki znajdują się na dysku lokalnym, nie będzie potrzebne połączenie z siecią (prosimy o cierpliwość; do zagadnień sieciowych przejdziemy w następnym podrozdziale).

▼ Należy stworzyć dwie strony HTML i zapisać je w osobnych plikach. Poniżej znaleźć można kod obu stron, które utworzyliśmy dla potrzeb tego ćwiczenia. Nazwy plików

- to *menu.html* oraz *klaudiusz.html*. Wygląd obu stron, jak i nazwy plików nie mają doprawdy znaczenia (trzeba jednak konsekwentnie stosować wybrane nazwy). Oto zawartość pierwszego pliku — *menu.html*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Żywoty cezarów</title>
</head>
<body>
<h1><i>Żywoty cezarów</i> Swetoniusza</h1>
<p>Swetoniusz (właściwie Caius Suetonius Tranquillus) urodził się około roku 70
i zmarł nie później niż w roku 130. Zapisał historię życia cesarzy od
Juliusza do Domicjana (zmarłego w roku 96). Jego prace były głównym źródłem,
z którego korzystali twórcy bestselerowej powieści i serialu telewizyjnego
zatytułowanych <i>Ja, Klaudiusz</i>. Cesarze rzymscy, których biografie napisał
Swetoniusz, to:</p>
<ul>
 <li>Juliusz Cezar</li>
 <li>Oktawian August</li>
 <li>Tyberiusz</li>
 <li>Kaligula</li>
 <li>Klaudiusz</li>
 <li>Neron</li>
 <li>Galba</li>
 <li>Oton</li>
 <li>Witeliusz</li>
 <li>Wespazjan</li>
 <li>Tytus</li>
 <li>Domicjan</li>
</ul>
</body>
</html>
```

Pozycje menu (Juliusz Cezar, August itd.) staną się łączami do innych stron. Na razie wystarczy wpisać je jako zwykły tekst. Później przerobimy je na łącza. A teraz zawartość drugiego pliku, czyli *klaudiusz.html*:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Żywoty cezarów: Klaudiusz</title>
</head>
<body>
<h2>Klaudiusz zostaje cesarzem</h2>
<p>Klaudiusz został cesarzem, mając pięćdziesiąt lat. Raz, bojąc się zamachowców
nasłanych na niego przez Kaligulę, ukrył się za kotarami. Kiedy znalazł go
strażnik, Klaudiusz padł na ziemię i wtedy dowiedział się, że został cesarzem.</p>
<h2>Klaudiusz zostaje otruty</h2>
<p>Według powszechnej opinii Klaudiusz został otruty. Niektórzy podejrzewają
jego małżonkę Agrypinę o zatrucie dania z grzybów (szczególnie lubił grzyby).
Jego śmierć została ujawniona dopiero po podjęciu działań mających na celu
przekazanie władzy synowi Agrypiny -- Neronowi.</p>
<p>powrót do menu głównego</p>
</body>
</html>
```

- Należy upewnić się, że oba pliki są w tym samym katalogu (folderze). Jeśli wybrano inne nazwy niż *menu.html* i *klaudiusz.html*, trzeba je zanotować, ponieważ będą potrzebne później.

Utworzymy łącze od pliku menu do pliku z treścią. W edytorze otwieramy plik *menu.html* i ustawiamy kursor w linii wyglądającej tak:

```
<li>Klaudiusz</li>
```

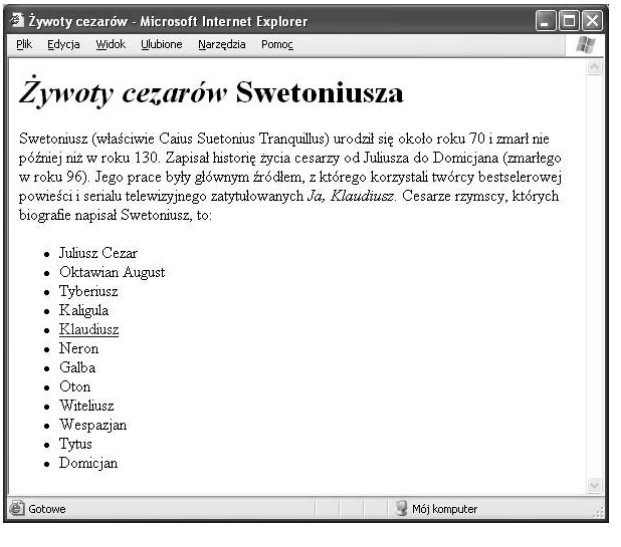
Znaczniki tworzące łącza nie określają formatu samego tekstu, w związku z tym pozostawiamy znaczniki oznaczające element listy, a łącze umieszczamy pomiędzy nimi. Najpierw dopisujemy znaczniki łącza jako takie (czyli znaczniki `<a>` oraz `</a>`) przed i po tekście mającym służyć za łącze:

```
<li><a>Klaudiusz</a></li>
```

Teraz dodajemy nazwę pliku, do którego łącze chcemy utworzyć. Zapisujemy ją w części `href` znacznika otwierającego `<a>`. Nazwę pliku umieszcza się w cudzysłowie (przy czym należy użyć zwykłego znaku cudzysłowu (`"`), nie cudzysłowu drukarskiego czy jeszcze innych znaków). Należy pamiętać o znaku równości (`=`) pomiędzy słowem `href` a nazwą pliku. Nazwy plików w łączach są zapisywane z rozróżnianiem małych i wielkich liter, trzeba się więc upewnić, że nazwa wpisana w znaczniku i nazwa istniejącego pliku są rzeczywiście identyczne (*Klaudiusz.html* to nie ten sam plik, co *klaudiusz.html*; wielkość liter musi się dokładnie zgadzać). Zastosowana tu nazwa to *klaudiusz.html*; w przypadku wybrania innych nazw, należy użyć właśnie ich.

```
<li><a href="klaudiusz.html">Klaudiusz</a></li>
```

Teraz uruchamiamy przeglądarkę, wybieramy polecenie *Plik/Otwórz...* (lub jego odpowiednik w używanej przeglądarce) i otwieramy plik *menu.html*. Tekst wykorzystany jako łącze powinien ukazać się w postaci właściwej dla łączy, czyli w specjalnym kolorze, podkreślony lub wyróżniony w inny sposób. Na rysunku 3.3 przedstawiono uzyskany wygląd strony.



Rysunek 3.3. *Plik menu.html po dodaniu łącza*

-

▼ W tej chwili, jeśli kliknie się łącze, przeglądarka powinna wczytać i wyświetlić stronę zapisaną w pliku *klaudiusz.html*.

Jeśli przeglądarka nie może odnaleźć pliku w momencie kliknięcia łącza, należy upewnić się, czy nazwa pliku umieszczona w części `href` znacznika łącza jest taka sama, jak nazwa pliku istniejącego na dysku, czy zgadza się sposób zastosowania wielkich i małych liter oraz czy oba pliki znajdują się w tym samym katalogu. Trzeba pamiętać, aby na końcu tekstu służącego za łącze pojawiał się znacznik zamykający `</a>`. Warto też sprawdzić, czy na końcu wpisanej nazwy pliku pojawia się znak cudzysłowu (zdarza się czasem o tym zapominać) oraz czy znak cudzysłowu, tak przed, jak i po nazwie, jest właściwego rodzaju (zwykły znak cudzysłowu). Wszystkie te problemy mogą zmylić przeglądarkę i uniemożliwić odnalezienie pliku, a tym samym prawidłowe działanie w momencie wybrania łącza.



Trzeba dobrze zrozumieć problem rozróżniania małych i wielkich liter. W nazwach znaczników HTML nie uwzględnia się wielkości liter (jakkolwiek specyfikacja XHTML 1.0 wymaga stosowania w nich liter małych). Niemniej pliki, o jakich mówimy, znajdują się na jakimś serwerze WWW, a ponieważ serwery WWW często działają pod kontrolą systemów operacyjnych, które rozróżniają wielkość liter (systemy typu UNIX), należy upewnić się co do pełnej zgodności nazw plików w znacznikach łączy.

Teraz można utworzyć łącze w odwrotnym kierunku: ze strony z tekstem do strony z menu. I takie właśnie jest przeznaczenie akapitu kończącego plik *klaudiusz.html*:

```
<p>powrót do menu głównego</p>
```

Dodajemy do tej linii znacznik tworzący łącze z odpowiednim atrybutem `href`, tak jak w poniższym przykładzie, w którym oryginalny plik strony z menu nazywa się *menu.html*:

```
<p>powrót do <a href="menu.html">menu głównego</a></p>
```



Umieszczając znacznik w środku innego znacznika, trzeba uważać, aby w pierwszej kolejności zamknąć ten znacznik, który został otwarty jako ostatni. W przeciwnym razie przeglądarka może się pogubić. Innymi słowy, poprawny zapis to:

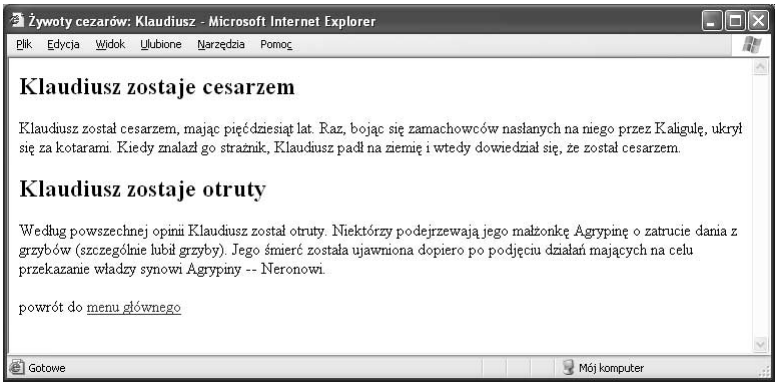
```
<p> <a> .. </a> </p>
```

w przeciwieństwie do:

```
<p> <a> .. </p> </a>
```

Teraz wystarczy ponownie wczytać do przeglądarki stronę o Klaudiuszu, aby uaktywnić łącze (patrz: rysunek 3.4). Można teraz wygodnie przechodzić ze strony z menu na ▲ stronę ze szczegółami, wybierając tak utworzone łącza.

Rysunek 3.4.
Strona zapisana w pliku klaudiusz.html



Łączenie stron lokalnych ścieżkami względnymi i bezwzględnymi

Przykład wykorzystany w poprzednim podrozdziale pokazuje, jak łączyć z sobą strony znajdujące się w tym samym katalogu (folderze) na dysku lokalnym. Teraz rozwiniemy ten wątek — łączeniu podlegać będą strony zapisane na dysku lokalnym, jednak w różnych katalogach (folderach).



Foldery i katalogi to jedno i to samo, ale nazwa zależy od tego, czy używa się systemu Windows, komputera macintosh czy też systemu typu Unix. Żeby uprościć nam wszystkim życie, od teraz nazywane tu będą po prostu *katalogami*.

Kiedy między znakami cudzysłowu pojawia się wyłącznie nazwa podłączanego pliku, jak to miało miejsce dotąd, przeglądarka szuka wskazanego pliku w katalogu, z którego wczytano bieżący plik. Pozostaje to prawdą, nawet jeśli zarówno plik bieżący, jak i plik podłączany znajdują się gdzieś daleko w internecie; mają się znajdować w tym samym katalogu na swoim serwerze. Jest to najprostszy przypadek ścieżki względnej.



Ścieżka względna prowadzi do plików na podstawie ich położenia względem pliku bieżącego. Może zawierać nazwy katalogów lub też wskazywać drogę, którą trzeba przebyć od bieżącego katalogu do danego pliku. Na ścieżkę taką mogłyby na przykład składać się polecenia przejścia najpierw dwa poziomy w górę drzewa katalogów, a następnie przez dwa określone katalogi w dół, aż do pliku.

Aby zapisać ścieżkę względną w parametrach łącza, stosuje się składnię pochodzącą z systemów typu Unix, niezależnie od rzeczywiście używanego systemu. Oznacza to oddzielanie nazw katalogów ukośnikiem (/) oraz używanie dwóch kropek (..) jako uniwersalnego odwołania do katalogu nadrzędnego. Tabela 3.1 pokazuje kilka przykładów ścieżek względnych wraz z wyjaśnieniami.

Tabela 3.1. *Ścieżki względne*

Ścieżka	Wyjaśnienie
`href="plik.html"`	Plik *plik.html* znajduje się w katalogu bieżącym
`href="pliki/plik.html"`	Plik *plik.html* znajduje się w katalogu o nazwie *pliki*, który znajduje się w katalogu bieżącym
`href="pliki/jeszczerazpliki/plik.html"`	Plik *plik.html* znajduje się w katalogu o nazwie *jeszczerazpliki*, który znajduje się w katalogu o nazwie *pliki*, który z kolei znajduje się w katalogu bieżącym
`href="../plik.html"`	Plik *plik.html* znajduje się w katalogu znajdującym się o poziom wyżej niż bieżący (czyli w katalogu nadrzędnym)
`href="../../pliki/plik.html"`	Plik *plik.html* znajduje się w katalogu o nazwie *pliki*, który znajduje się w katalogu położonym dwa poziomy wyżej niż bieżący

W przypadku łączenia stron przechowywanych w komputerze osobistym (typu PC lub macintosh), aby podłączyć plik umieszczony na innym dysku, używa się nazwy lub litery dysku tak samo, jak innych nazw katalogów w ścieżce względnej.

Aby podłączyć plik na dysku lokalnym macintosha, stosuje się taką nazwę dysku, jaka pojawia się przy ikonie właściwego dysku. Przyjmijmy, że pliki HTML znajdują się w folderze *Pliki HTML* na dysku o nazwie *Dysk twardy 2*. Wówczas podłączenie pliku o nazwie *julia.html* w folderze o nazwie *Publiczne* na dysku udostępnionym pod nazwą *Mak Julki* wymaga zastosowania następującej ścieżki względnej:

```
href="../../Mak Julki/Publiczne/julia.html"
```

Kiedy plik zapisany na dysku lokalnym podłącza się w systemie Windows, dysk wskazuje się za pomocą litery — jak moglibyśmy się w istocie spodziewać. Jednakże zapis *c:* albo *d:* zostaje zmodyfikowany poprzez zastąpienie dwukropka znakiem pionowej kreski (|), ponieważ dwukropek ma szczególne znaczenie w adresach URL. Znak pionowej kreski na klawiaturze dzieli zwykle klawisz ze znakiem odwrotnego ukośnika, nadrukowany zaś jest tam w postaci dwóch pionowych kreseczek, jednej nad drugą. Nie należy zapominać o używaniu zwykłych ukośników, jak w systemach typu Unix. Tak więc jeśli bieżący plik znajduje się w katalogu *C:\PLIKI\HTML*, chcemy zaś utworzyć łącze do pliku *D:\PLIKI.NOW\HTML\JESZCZE\INDEX.HTM*, ścieżka względna do tego pliku będzie wyglądać tak:

```
href="../../d|/pliki.now/html/jeszcze/index.htm"
```

W większości przypadków używanie nazw dysków w ścieżkach względnych prawie nigdy nie jest praktyczne — wspominamy o tym dla kompletności opisu. Po przeniesieniu plików na serwer WWW łącza zawierające nazwy dysków nie będą działać. Zwykle przydają się więc łącza względne sformułowane w bardziej uniwersalny sposób.

Ścieżki bezwzględne

Można także utworzyć łącze do innej strony zapisanej w systemie lokalnym za pomocą ścieżki bezwzględnej.



Ścieżka bezwzględna wskazuje drogę do pliku na podstawie jego bezwzględnego (absolutnego) położenia w systemie plików. Podczas gdy ścieżki względne wskazują podłączany plik, opisując jego pozycję w odniesieniu do pliku strony bieżącej, ścieżki bezwzględne wyznaczają położenie, zaczynając od najwyższego katalogu w hierarchii. Następnie pojawiają się nazwy wszystkich kolejnych zawierających się katalogów — aż do tego, w którym znajduje się szukany plik.

Ścieżki bezwzględne zawsze zaczynają się ukośnikiem, co odróżnia je od ścieżek względnych. Po ukośniku następują wszystkie kolejne katalogi od najwyższego poziomu aż do podłączanego pliku.



Gdzie konkretnie znajduje się ów *najwyższy poziom*, zależy od sposobu wykorzystania plików. Jeśli po prostu podłącza się pliki na dysku lokalnym, najwyższy poziom oznacza najwyższy poziom systemu plików (oznaczany ukośnikiem w systemach typu Unix, a na macintoshu — nazwą dysku). W przypadku umieszczenia plików na serwerze WWW najwyższym poziomem jest katalog, w którym przechowywane są przeznaczone do udostępniania pliki użytkownika. O zagadnieniu ścieżek bezwzględnych na serwerach WWW więcej w rozdziale 15.

Tabela 3.2 przedstawia kilka przykładów ścieżek bezwzględnych wraz z wyjaśnieniami.

Tabela 3.2. *Ścieżki bezwzględne*

Ścieżka	Wyjaśnienie
`href="/u1/lemay/plik.html"`	Plik *plik.html* znajduje się w katalogu *lemay*, który znajduje się w katalogu *u1*, który z kolei znajduje się w katalogu głównym (zapis typowy dla systemów typu Unix)
`href="/d|/pliki/html/plik.html"`	Plik *plik.html* znajduje się w katalogu *html*, który znajduje się w katalogu *pliki*, a ten z kolei w katalogu głównym dysku *D:* (system Windows)
`href="/Dysk%20twardy%201/Pliki HTML/plik.html"`	Plik *plik.html* znajduje się na dysku *Dysk twardy 1* w katalogu *Pliki HTML* (zapis typowy dla macintoshy)

Które ścieżki są lepsze — względne czy bezwzględne?

Odpowiedź na to pytanie brzmi: to zależy. Dla zestawów plików łączonych tylko między sobą zastosowanie ścieżek względnych ma sens. Jeśli natomiast łącza prowadzą do plików poza daną hierarchią, najprawdopodobniej odpowiednie będą ścieżki bezwzględne.

Do lepszego wyjaśnienia przydać się może przykład. Załóżmy, że witryna składa się z dwóch działów: */rzeczy* oraz */sprawy*. Łącze od pliku *index.html* w katalogu */rzeczy* do pliku *historia.html* w katalogu */rzeczy* (lub do jakiegokolwiek innego pliku w katalogu */rzeczy*) utworzymy za pomocą łącza względnego. W ten sposób przeniesienie

katalogu */rzeczy* w inne miejsce nie spowoduje uszkodzenia wewnętrznych łączy. Z drugiej strony, jeśli łącze miałoby prowadzić od pliku */rzeczy/index.html* do */sprawy/index.html*, odpowiednim rozwiązaniem będzie łącze bezwzględne. Jego zastosowanie pozwoli na działanie łącza nawet w przypadku przeniesienia katalogu */rzeczy* do katalogu */jeszczeinny*.

Złota zasada, którą zazwyczaj stosujemy, brzmi: pliki tworzące funkcjonalną całość łączymy względnie, a należące do osobnych grup — bezwzględnie.

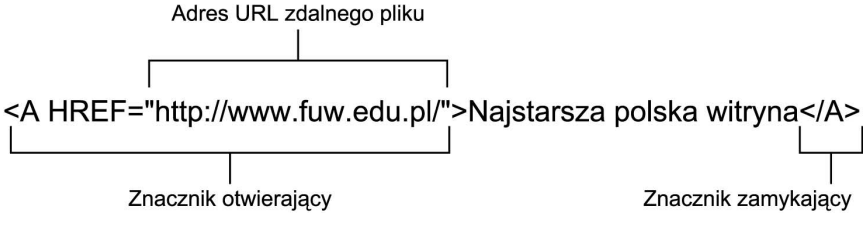
Łącza do innych dokumentów w sieci

Na dysku lokalnym mamy już więc zestaw połączonych z sobą stron. Przydałyby się jednakże odwołania do stron znajdujących się gdzieś w internecie — na przykład do witryny Piotra Kowackiego *Strefa Rzymu*[1], zawierającej dodatkowe informacje o cesarzach rzymskich. Znacznik tworzący łącza doskonale nadaje się także do podłączania stron w internecie, które będziemy nazywać stronami zdalnymi.



Strony zdalne znajdują się gdzieś w sieci, lecz zwykle na innym komputerze niż ten, którym zajmujemy się w danym momencie.

Kod HTML służący do utworzenia łącza do strony zdalnej wygląda dokładnie tak samo, jak kod opisujący łącza między stronami zapisanymi lokalnie. W dalszym ciągu znajduje tu zastosowanie znacznik `<a>` z atrybutem `href`, jednak, jak pokazuje rysunek 3.5, nazwę pliku zastępuje adres URL.



Rysunek 3.5. *Łącze do zdalnego pliku*



Ćwiczenie 3.2. Podłączanie stron

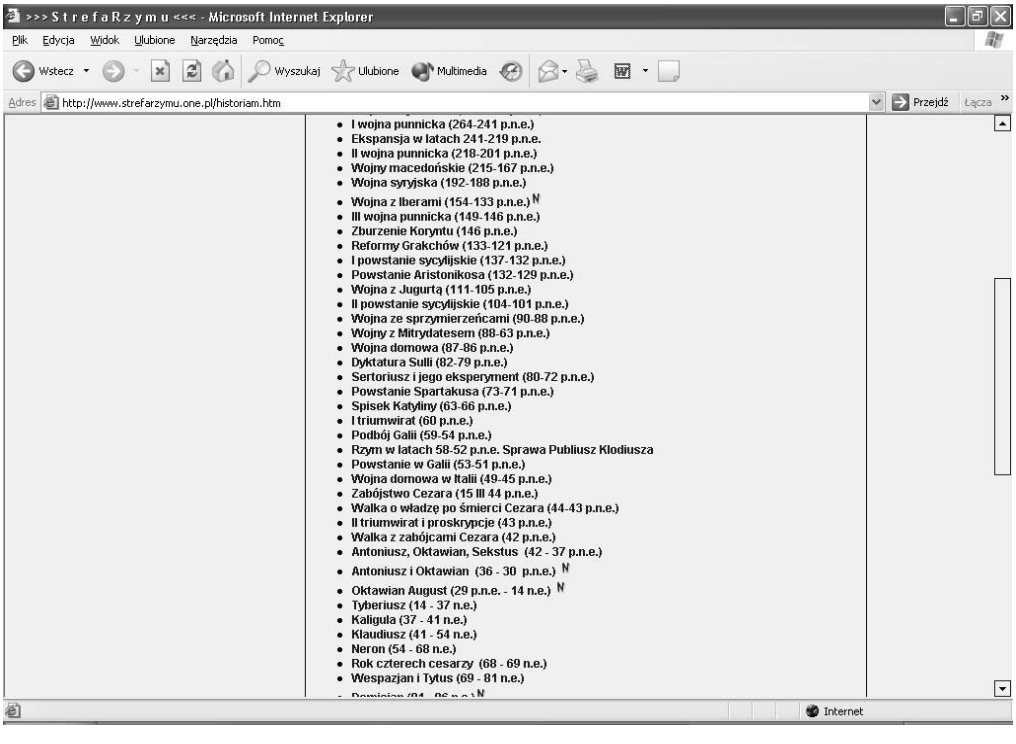
Wróćmy do dwóch stron poświęconych cesarzom, na których wcześniej umieściliśmy łącza. Plik *menu.html* zawiera łącza do stron lokalnych zawierających informacje o dwunastu cesarzach rzymskich.

[1] W polskim wydaniu zastąpiono w ten sposób odwołanie do angielskojęzycznej witryny dr Ellis Knox z Uniwersytetu Stanowego w Broise zatytułowanej *The first caesars page* (ang. *Strona o pierwszych cesarzach*), dostępnej pod adresem *http://history.boisestate.edu/westciv/julio-cl/ — przyp. tłum.*

▼ Załóżmy, że chcemy na tej stronie, poniżej menu, dodać łącze wskazujące spis artykułów o starożytnym Rzymie na witrynie Piotra Kowackiego. Odpowiedni adres URL to *http://www.strefarzymu.one.pl/historiam.htm* . W pierwszej kolejności dopiszmy na końcu strony tekst z opisem przyszłego łącza:

```
<p>Witryna Piotra Kowackiego <i>Strefa Rzymu</i> zawiera więcej informacji o tych
↳władcach.</p>
```

A jeśli nie znalibyśmy adresu URL witryny *Strefa Rzymu* (lub innej strony, do której łącze chcielibyśmy utworzyć)? Jeśli tylko umielibyśmy trafić do niej, przechodząc po kolejnych łączach z innych witryn, nie byłoby problemu. Pierwszym krokiem do określenia adresu URL byłoby otwarcie w przeglądarce strony, którą zamierzalibyśmy podłączyć do swojej. Rysunek 3.6 pokazuje stronę witryny *Strefa Rzymu* wyświetloną w przeglądarce.



Rysunek 3.6. *Spis artykułów na witrynie Strefa Rzymu*



Jeśli połączenie z internetem nie jest aktywne, trzeba je teraz włączyć — inaczej nie będzie możliwe przetestowanie łączy do stron znajdujących się w sieci.

W większości przeglądarek adres URL otwartej w danym momencie strony pojawia się gdzieś w górnej części okna. Dzięki temu łatwo utworzyć łącze do wybranej strony: wystarczy skierować przeglądarkę pod odpowiedni adres URL, skopiować go z okienka adresu, a na koniec wkleić do kodu HTML, nad którym się właśnie pracuje. Żadnego pisania!

▼

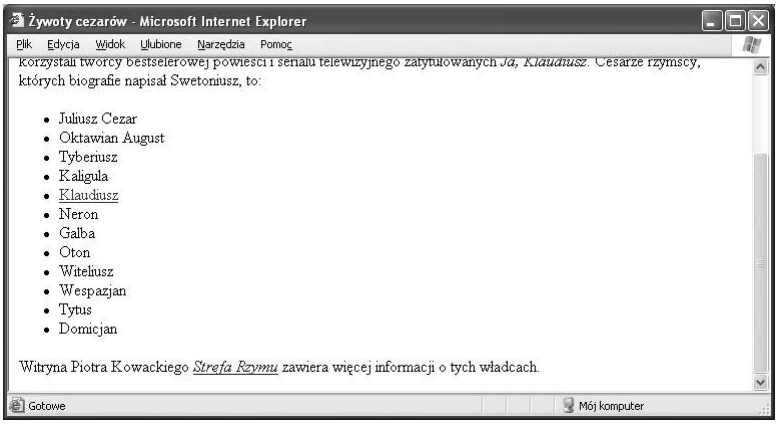
▼ Po zdobyciu adresu URL pożądanej strony można skonstruować znacznik tworzący łącze i umieścić go w pliku menu wraz z wklejonym adresem URL — w taki sposób:

```
<p>Witryna Piotra Kowackiego <i><a
href="http://www.strefarzymu.one.pl/historiam.htm ">Strefa Rzymu</a></i>
zawiera więcej informacji o tych władcach.</p>
```

Naturalnie adres URL strony, którą chce się podłączyć, można po prostu wpisać do części `href` znacznika. Należy jednak pamiętać, że popełnienie nawet drobnego błędu sprawi, że otwarcie w przeglądarce pliku znajdującego się na drugim końcu łącza nie będzie możliwe. Większość adresów URL jest zbyt skomplikowana, żeby człowiek był w stanie je zapamiętać. Zalecamy korzystać z poleceń kopiowania i wklejania — tylko wówczas można uniknąć kłopotów związanych z pomyłką podczas wpisywania adresu URL.

Na rysunku 3.7 przedstawiono, jak wyświetlany jest plik *menu.html* po dodaniu nowego łącza.

Rysunek 3.7.
Łącze do Strefy Rzymu



▲

wykonaj

Ćwiczenie 3.3. Jak utworzyć menu łączy

nowy termin

Gdy już wiesz, jak używać list oraz łączy, możesz stworzyć *menu łączy*. Menu łączy to pewna liczba łączy na stronie WWW, której nadano kształt listy albo inny zwięzły, łatwy do odczytania i zrozumienia format. Rozwiązanie to jest idealne w przypadku stron układających się w hierarchię, nadaje się też do tworzenia spisów treści oraz wskazywania wybranej strony pośród wielu. Strony WWW, które składają się wyłącznie z łączy, często przyjmują właśnie taką formę.

Pomysł polega na używaniu krótkich, opisowych etykiet w charakterze łączy — bez dodatkowego tekstu obok łącza, albo właśnie z rozszerzonym opisem następującym zaraz po nim. Menu łączy wyglądają najlepiej jako listy punktowane lub inne listy nieuporządkowane. Można też używać list definicji lub po prostu zwykłych akapitów. Zastosowanie menu łączy pozwala czytającym na szybkie przeglądnięcie spisu łączy — rzecz trudna w przypadku, gdy łącza ukryte są w tekście.

▼

W tym ćwiczeniu stworzymy stronę WWW z recenzjami kilku książek. Strona będzie służyć jako spis recenzji, w związku z czym menu łączy będzie w istocie składać się z nazw książek. Zaczynamy od nieskomplikowanego szkieletu strony — z nagłówkiem pierwszego poziomu i prostym tekstem wprowadzającym:

źródło

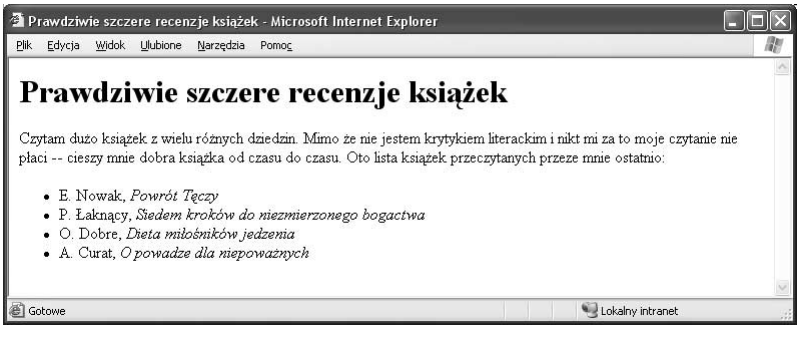
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Prawdziwie szczere recenzje książek</title>
</head>
<body>
<h1>Prawdziwie szczere recenzje książek</h1>
<p>Czytam dużo książek z wielu różnych dziedzin. Mimo że nie jestem krytykiem
literackim i nikt mi za to moje czytanie nie płaci -- cieszy mnie dobra książka
od czasu do czasu. Oto lista książek przeczytanych przeze mnie ostatnio:</p>
```

Teraz dodajemy listę, której elementy staną się łączami — ale na razie bez znaczników tworzących łącza (zawsze łatwiej zacząć od tekstu, a dopiero potem zrobić z niego właściwe łącze). Zastosujemy znaczniki w celu utworzenia punktowanej listy książek. Znacznik `<ol>` niezbyt się tu nadaje, ponieważ pojawienie się numerów mogłoby sprawiać wrażenie, że wymieniamy książki w kolejności według oceny. Poniżej widać kod HTML listy książek, a rysunek 3.8 pokazuje wyświetloną stronę — na obecnym etapie składają się na nią wprowadzenie i lista.

```
<ul>
 <li>E. Nowak, <i>Powrót Tęczy</i></li>
 <li>P. Łaknący, <i>Siedem kroków do niezmierzonego bogactwa</i></li>
 <li>O. Dobre, <i>Dieta miłośników jedzenia</i></li>
 <li>A. Curat, <i>O powadze dla niepoważnych</i></li>
</ul>
```

wynik

Rysunek 3.8. *Lista książek*



Następnie modyfikujemy każdą pozycję listy w taki sposób, aby zawierała znaczniki tworzące łącza. Znaczniki `<li>` zostają na swoich miejscach — wyznaczają przecież początki kolejnych elementów. Znaczniki `<a>` umieszczamy przed i po tekście. W tym przypadku podłączamy pliki znajdujące się na dysku lokalnym, w tym samym katalogu, co nasz plik; każdy z nich zawiera recenzję danej książki:

źródło

```
<ul>
 <li><a href="tecza.html">E. Nowak, <i>Powrót Tęczy</i></a></li>
 <li><a href="bogactwo.html">P. Łaknący, <i>Siedem kroków do niezmierzonego
 bogactwa</i></a></li>
```

```
<li><a href="jedzenie.html">O. Dobre, <i>Dieta miłośników jedzenia</i></a></li>
 <li><a href="powaga.html">A. Curat, <i>O powadze dla niepoważnych</i></a></li>

</ul>
```

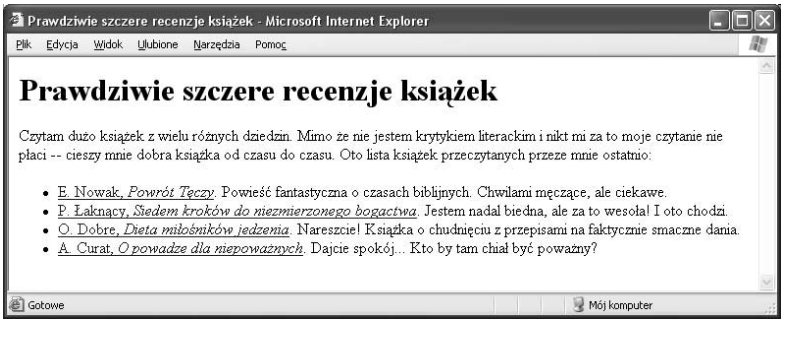
Menu książek wygląda dobrze mimo pewnej zgrzebności. Jego lektura nie dostarcza czytelnikom żadnych informacji o charakterze poszczególnych książek (choć niektóre tytuły zdradzają ogólną tematykę) ani o tym, czy są dobre albo złe. W związku z tym ulepszenie może polegać na rozwinięciu elementów listy o krótkie teksty objaśniające i zasugerowaniu w ten sposób, co czeka czytelnika na drugim końcu łącza:

```
<ul>
 <li><a href="tecza.html">E. Nowak, <i>Powrót Tęczy</i></a>. Powieść fantastyczna
 o czasach biblijnych. Chwilami męczące, ale ciekawe.</li>
 <li><a href="bogactwo.html">P. Łaknący, <i>Siedem kroków do niezmierzonego
 bogactwa</i></a>. Jestem nadal biedna, ale za to wesoła! I oto chodzi.</li>
 <li><a href="jedzenie.html">O. Dobre, <i>Dieta miłośników jedzenia</i></a>.
 Nareszcie! Książka o chudnięciu z przepisami na faktycznie smaczne dania.</li>
 <li><a href="powaga.html">A. Curat, <i>O powadze dla niepoważnych</i></a>.
 Dajcie spokój... Kto by tam chiał być poważny?</li>
</ul>
```

Kompletne menu łączy przedstawiono na rysunku 3.9.

wynik

Rysunek 3.9. *Kompletne menu łączy*



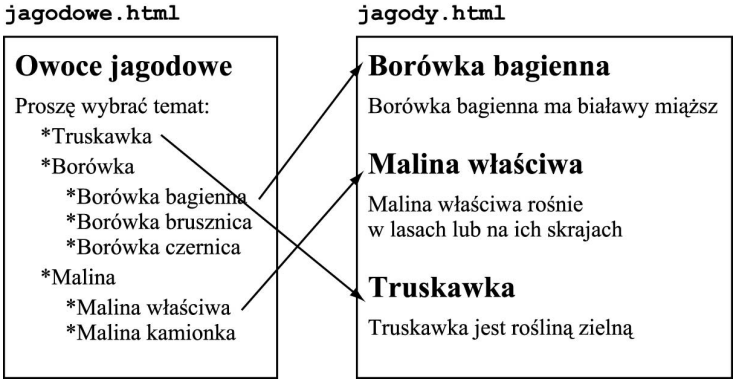
Łącza do fragmentów stron

Łącza omawiane do tej pory przebiegały pomiędzy punktem na stronie źródłowej a stroną docelową. Co jednak należy zrobić, gdy chcemy wskazać wybraną część strony — na przykład czwarty nagłówek od góry?

nowy termin

W języku HTML można to osiągnąć, tworząc *zakotwiczenie* (ang. *anchor*) w kodzie strony docelowej, które umożliwia utworzenie łącza do środka strony. Dzięki zakotwiczeniom możliwe jest przejście od razu do wybranego miejsca w środku dokumentu zamiast do jego początku.

W kodzie łącza na stronie źródłowej wpisuje się, oprócz nazwy podłączanego pliku, również nazwę wskazywanego zakotwiczenia. Dzięki temu wybranie łącza w przeglądarce powoduje wczytanie strony docelowej, a następnie przewinięcie widoku aż do miejsca, gdzie położone jest zakotwiczenie (patrz: rysunek 3.10).



Rysunek 3.10. *Łącza do zakotwiczeń*

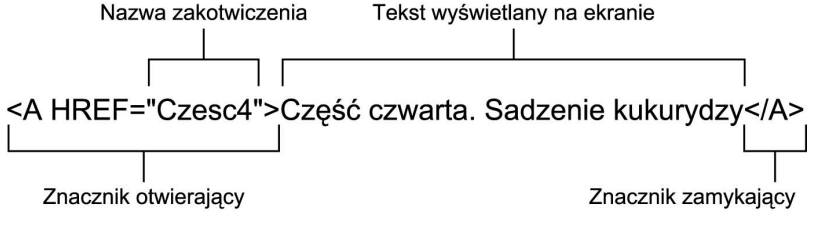
Można używać łączy i zakotwiczeń wewnątrz pojedynczej strony w ten sposób, że wybranie jednego łącza powoduje przewinięcie widoku do punktu zakotwiczenia na tej samej stronie.

Jak tworzyć łącza i zakotwiczenia

Zakotwiczenia tworzy się w podobny sposób, jak łącza: korzystając ze znacznika `<a>`. Dlatego też nazwa tego znacznika jest właśnie taka — nie `<l>`, jak *link* (łącze), lecz `<a>`, jak *anchor* (czyli zakotwiczenie).

W przypadku zapisywania łącza za pomocą znacznika `<a>` potrzebne są dwie części: atrybut `href` w otwierającym znaczniku `<a>` oraz tekst pomiędzy znacznikami otwierającym a zamykającym. Tekst ów pełni wtedy funkcję punktu dostępu do łącza.

Zakotwiczenia powstają prawie tak samo, lecz atrybut `href` zostaje zastąpiony przez atrybut `name`, który przyjmuje jako wartość słowo (lub słowa kluczowe) nazywające to zakotwiczenie. Na rysunku 3.11 wyodrębniono części znacznika `<a>` tworzącego łącze do zakotwiczenia.



Rysunek 3.11. *Zakotwiczenie — znacznik <a>*

Mimo że zakotwiczenia zwykle wskazują miejsce z dokładnością do pojedynczego znaku, wymagają — podobnie jak łącza — umieszczenia tekstu pomiędzy otwierającym a zamykającym znacznikiem `<a>`. Ten właśnie tekst ma ukazać się użytkownikowi, który wybrał łącze do danego zakotwiczenia. Przeglądarka przewija stronę do miejsca, gdzie znajduje się tekst oznaczony zakotwiczeniem — powinien się on znaleźć w górnej części ekranu. W niektórych przeglądarkach może też być podświetlany.

Na przykład utworzenie zakotwiczenia do fragmentu strony rozpoczynającego się od nagłówka „Część czwarta" może polegać na dodaniu do tego nagłówka znacznika `<a>` z etykietą `czesc4`, tak jak poniżej:

```
<h1><a name="czesc4">Część czwarta. Grejpfrut z nieba</a></h1>
```

Inaczej niż w przypadku łączy, tekst wykorzystywany w zakotwiczeniach zwykle nie jest specjalnie formatowany na ekranie, a nazwy zakotwiczeń nie są wyświetlane w tekście strony.

Aby utworzyć łącze do zakotwiczenia, stosuje się taką formę łącza, jaką można by się posłużyć do podłączenia całej strony, czyli z nazwą pliku lub adresem URL jako wartością atrybutu `href`. Różnica polega na dopisaniu do tej wartości znaku # i nazwy zakotwiczenia, dokładnie takiej, jaka pojawia się w atrybucie `name` tego zakotwiczenia (z uwzględnieniem wielkości liter!). Oto przykład:

```
<a href="mojduzydok.html#czesc4">Do części 4.</a>
```

Wybranie takiego łącza w przeglądarce powoduje otwarcie pliku *mojduzydok.html* i przewinięcie do zakotwiczenia o nazwie `czesc4`. Tekst wewnątrz definicji zakotwiczenia powinien się znaleźć w górnej części ekranu.

wykonaj

Ćwiczenie 3.4. Łącza między fragmentami dwóch stron

Czas więc utworzyć dwie przykładowe strony. Załóżmy, że trzeba uzupełnić internetowy leksykon muzyki poważnej, którego każda strona grupuje definicje terminów rozpoczynających się od jednej litery alfabetu (pliki *a.html*, *b.html* itd). Witryna leksykonu mogłaby być również podzielona na strony z pojedynczymi terminami. Taki sposób organizacji spowodowałby jednak konieczność administrowania wieloma stronami, jak również zmuszałby użytkowników do wczytywania wielu stron podczas eksplorowania leksykonu. Wiązanie pokrewnych fragmentów w grupy oznaczone literami jest w tym przypadku bardziej praktyczne.

Stroną, która ma się ukazać jako pierwsza, jest lista haseł na literę M. Jej pierwszy fragment, zapisany w języku HTML, wyglądać będzie tak:

źródło

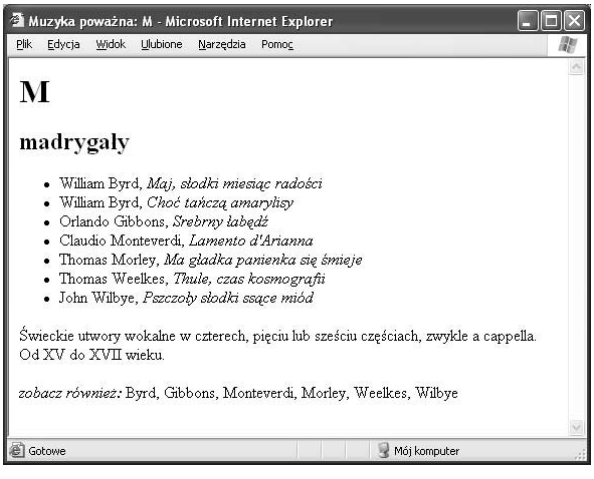
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Muzyka poważna: M</title>
</head>
<body>
<h1>M</h1>
<h2>madrygały</h2>
<ul>
 <li>William Byrd, <em>Maj, słodki miesiąc radości</em></li>
 <li>William Byrd, <em>Choć tańczą amarylisy</em></li>
 <li>Orlando Gibbons, <em>Srebrny łabędź</em></li>
 <li>Claudio Monteverdi, <em>Lamento d'Arianna</em></li>
 <li>Thomas Morley, <em>Ma gładka panienka się śmieje</em></li>
 <li>Thomas Weelkes, <em>Thule, czas kosmografii</em></li>
 <li>John Wilbye, <em>Pszczoły słodki ssące miód</em></li>
```

```
</ul>
<p>Świeckie utwory wokalne w czterech, pięciu lub sześciu częściach,
zwykle a cappella. Od XV do XVII wieku.</p>
<p><em>zobacz również:</em> Byrd, Gibbons, Monteverdi, Morley,
Weelkes, Wilbye</p>
</body>
</html>
```

Rysunek 3.12 pokazuje, jak ten fragment wygląda w przeglądarce.

wynik

Rysunek 3.12. *Internetowy leksykon muzyki poważnej — litera M, hasło „madrygały”*



W ostatniej linii (czyli *zobacz również*) przydałoby się dodać łącza od nazwisk kompozytorów do poświęconych im miejsc na odpowiednich stronach leksykonu. Możliwe jest utworzenie w słowie *Byrd* łącza do pliku *b.html*, zgodnie z poznaną wcześniej procedurą. Gdy użytkownik je kliknie, przejdzie do początku strony z terminami na literę B. Będzie teraz musiał przedzierać się przez informacje o wszystkich kompozytorach, których nazwiska zaczynają się na literę B (a takich jest niemało — Bach, Beethoven, Brahms, Bruckner). Dopiero wtedy dotrze do Byrda. Całkiem wiele zachodu jak na system, który miał ponoć udostępniać połączone informacje szybko i bez trudu.

Zatem, aby ułatwić posługiwanie się leksykonem, trzeba połączyć słowo *Byrd* z dotyczącym kompozytora fragmentem strony *b.html*. Oto część pliku *b.html*, którą zamierzamy podłączyć (w tym przykładzie w celu skrócenia pliku pominięto wszystkie terminy na literę B, prócz Byrda; umówmy się, że one wciąż tam są):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html>
<head>
<title>Muzyka poważna: B</title>
</head>
<body>
<h1>B</h1>
<!-- Żeby było krócej, pominięto wszystkie hasła na literę B aż do "Byrd" -->
<h2>Byrd William, 1543-1623</h2>
<ul>
 <li>madrygały
```

```
<ul>
   <li><em>Maj, słodki miesiąc radości</em></li>
   <li><em>Choć tańczą amarylisy</em></li>
   <li><em>Śpij, dzieciątko me słodkie</em></li>
 </ul>
 </li>
 <li>msze
 <ul>
   <li><em>msza pięciogłosowa</em></li>
   <li><em>msza trzygłosowa</em></li>
   <li><em>msza czterogłosowa</em></li>
 </ul>
 </li>
 <li>motety
 <ul>
   <li><em>Ave verum corpus a 4</em></li>
 </ul>
 </li>
</ul>
<p><em>zobacz również</em> madrygały, msze, motety</p>
</body>
</html>
```



W tym przykładzie można zauważyć pojawienie się znacznika `<em>`, który powoduje wyróżnianie tekstu. Wyróżnianie polega zwykle na wizualizacji czcionką pochyłą — tak dzieje się w przeglądarkach Netscape i Internet Explorer.

Teraz trzeba utworzyć zakotwiczenie w nagłówku fragmentu o Byrdzie, które zostanie podłączone do linii *zobacz również* w pliku z hasłami na literę M.

Jak opisaliśmy to już wcześniej, każde zakotwiczenie składa się z dwóch części: nazwy zakotwiczenia oraz tekstu, który będzie ekranową reprezentacją łącza (w niektórych przeglądarkach wyświetlaną z wyróżnieniem). Drugą z tych części łatwo określić; nagłówek fragmentu doskonale się nadaje — to przecież właśnie ten element, do którego tworzymy łącze.

Można nadać zakotwiczeniu dowolną nazwę, byle tylko nazwy zakotwiczeń na danej stronie nie powtarzały się. Jeśli na stronie znalazłyby się dwa zakotwiczenia o nazwie `fred`, w jaki sposób przeglądarka rozpozna, o które z nich chodzi użytkownikowi wybierającemu łącze? Dobrą, unikatową nazwą zakotwiczenia będzie w tym przypadku po prostu słowo `byrd`, ponieważ pojawia się tylko w jednym miejscu pliku — a tego właśnie potrzebujemy.

Po wybraniu nazwy i tekstu do zakotwiczenia czas utworzyć samo zakotwiczenie w kodzie HTML. Dodajemy znacznik `<a>` do nagłówka `William Byrd`, ale zachowujemy przy tym ostrożność. Otóż gdyby chodziło o zwykły tekst akapitu, objęłoby się po prostu całą linię znacznikami `<a>`. W tym przypadku trzeba jednak pamiętać, że jeśli umieszczamy zakotwiczenie w większym fragmencie tekstu, który z kolei zawiera się w elemencie (takim jak nagłówek lub akapit), wówczas znaczniki zakotwiczenia muszą trafić do wewnątrz nadrzędnego elementu. Innymi słowy, należy wpisać:

```
<h2><a name="byrd">Byrd William, 1543-1623</a></h2>
```

▼ — ale nie wolno wpisać:

```
<a name="byrd"><h2>Byrd William, 1543-1623</h2></a>
```

Drugi przykład może zmylić przeglądarkę. Czy to ma być zakotwiczenie, przeznaczone do sformatowania podobnie jak poprzedzający tekst, z dodatkowymi, tajemniczo zlokalizowanymi znacznikami nagłówka? Czy też raczej nagłówek, który akurat jest także zakotwiczeniem? Stosując prawidłowy zapis kodu HTML, czyli z zakotwiczeniem umieszczonym wewnątrz nagłówka, unika się dwuznaczności.

Łatwo zapomnieć o tej sprawie — zwłaszcza jeśli najpierw wpisujemy tekst, a dopiero później dodajemy znaczniki. Otoczenie całego nagłówka znacznikami `<a>` wydaje się mieć sens. Jednak co w przypadku, gdybyśmy mieli do czynienia nie z krótkim nagłówkiem, a długim akapitem? Skąd przeglądarka miałaby wiedzieć, którą linię akapitu pokazać? Aby utworzyć łącze do wybranego słowa albo zdania, umieszcza się znacznik `<a>` wewnątrz innych znaczników. Trzeba o tym pamiętać i wszystko będzie jasne.

W tym momencie mamy już w nagłówku odpowiednie zakotwiczenie o nazwie `byrd`. Wróćmy teraz do linii z tekstem `zobacz również` w pliku *m.html*:

```
<p><em>zobacz również:</em> Byrd, Gibbons, Monteverdi, Morley, Weelkes, Wilbye</p>
```

Zamierzamy więc utworzyć w słowie *Byrd* łącze, czym zajmowaliśmy się już przecież kilkakrotnie. Lecz jak brzmi adres URL? Jak wspominaliśmy wcześniej, ścieżki do zakotwiczeń powstają według następującego wzorca:

```
nazwa_pliku#nazwa_zakotwiczenia
```

Mamy tu do czynienia z łączem do pliku *b.html*, którego nazwę podstawiamy w atrybucie `href`:

```
<a href="b.html">
```

Jednak chcemy podłączyć konkretne miejsce na tej stronie. Dodajemy więc nazwę zakotwiczenia i nasz znacznik wygląda teraz tak:

```
<a href="b.html#byrd">
```

Zwróćmy uwagę, że nazwę `byrd` zapisano z małej litery. W nazwach zakotwiczeń uwzględnia się wielkość liter; gdyby w wartości `href` wpisać `#Byrd`, łącze mogłoby nie działać. Trzeba sprawdzać, czy nazwa zakotwiczenia użyta w atrybucie `name` i nazwa zakotwiczenia w atrybucie `href` są na pewno identyczne.



Często popełnianym błędem jest poprzedzanie nazwy znacznika znakiem # nie tylko w parametrach łącza, ale i w definicji zakotwiczenia. Jedynym właściwym tu zastosowaniem znaku # jest oddzielanie adresu pliku i nazwy zakotwiczenia. Same nazwy zakotwiczeń nigdy nie powinny zawierać znaków #.

Ostatecznie, po dodaniu łącza do części oznaczonej *zobacz również*, wygląda ona następująco:

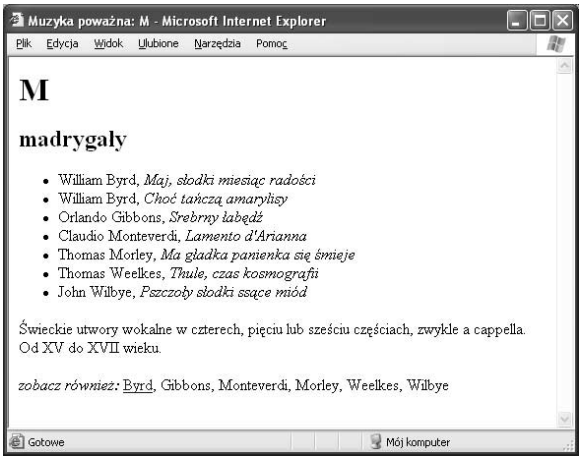
▼

```
<p><em>zobacz również</em> <a href="b.html#byrd">Byrd</a>, Gibbons, Monteverdi,
↳Morley, Weelkes, Wilbye</p>
```

▼ Naturalnie nie ma żadnych przeszkód, aby wzbogacić o zakotwiczenia i łącza wszelkie wzmianki o pozostałych kompozytorach.

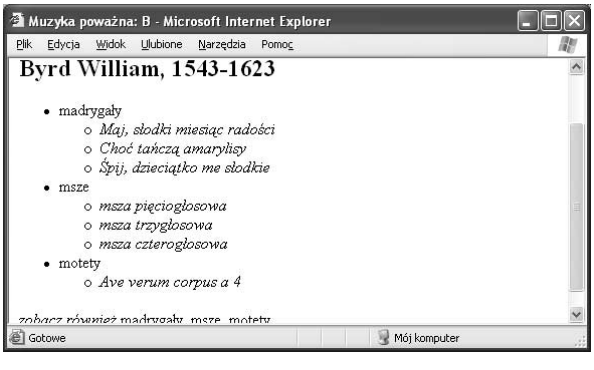
Po umieszczeniu na właściwych miejscach wszystkich zakotwiczeń i łączy nadchodzi czas na przetestowanie ich. Rysunek 3.13 pokazuje opis hasła *madrygały* z łączem do hasła *Byrd* gotowym do kliknięcia.

Rysunek 3.13.
Hasło madrygały z łączem do hasła Byrd



Na rysunku 3.14 przedstawiono ekran, jaki ukazuje się po kliknięciu słowa *Byrd*.

Rysunek 3.14.
Hasło Byrd



▲

Podłączanie zakotwiczeń wewnątrz dokumentu

A co można zrobić w sytuacji, gdy wszystko znajduje się na jednej obszernej stronie i trzeba utworzyć łącza od fragmentu do fragmentu? Zakotwiczenia sprawdzają się nadal, pozwalając na łatwe przewijanie dłuższych stron do wybranego miejsca. Aby podłączać wybrane fragmenty tekstu, umieszcza się na ich początkach zakotwiczenia — tak jak zwykle. Wówczas, zapisując kod łącza, pomija się nazwę pliku strony i podaje nazwę zakotwiczenia poprzedzoną znakiem #. Jeśli, dajmy na to, potrzebujemy łącza do zakotwiczenia o nazwie `punkt5` znajdującego się na tej samej stronie, co łącze, kod łącza wygląda następująco:

```
przejście do <a href="#punkt5">punktu 5.</a>
```

Jeśli nazwa pliku nie została podana, przeglądarka przyjmuje, że chodzi o stronę bieżącą, i gdy takie łącze zostaje wybrane, przewija widok do odpowiedniego miejsca. Będziemy mogli podziwiać tę funkcję w akcji w rozdziale następnym. Zajmiemy się tam konstruowaniem kompletnej strony WWW, na początku której znajdzie się spis treści. W spisie tym czytelnik będzie mógł wybierać łącza, aby szybko przewijać widok do różnych części tej samej strony. Łącza w spisie treści będą prowadzić do nagłówków wszystkich kolejnych części tekstu. I odwrotnie — na końcu każdego fragmentu umieścimy łącze pozwalające szybko wrócić do spisu treści, czyli na początek strony.

Anatomia adresu URL

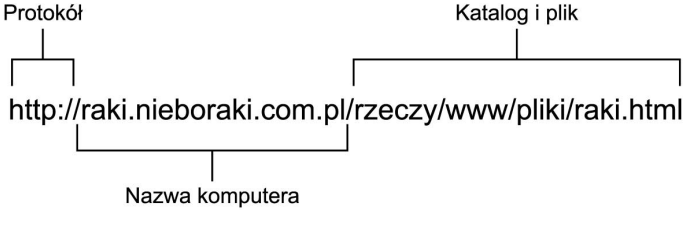
W tym rozdziale wspominaliśmy już o adresach URL (kiedy zajmowaliśmy się łączami do stron zdalnych). Ktokolwiek przeglądał strony WWW, siłą rzeczy zetknął się z pewną liczbą adresów URL. Przede wszystkim jakikolwiek adres jest potrzebny, by w ogóle zacząć eksplorować zasoby WWW.

Skrót URL pochodzi od angielskich słów *Uniform Resource Locator* — jednolity lokalizator zasobów. Adresy URL są jakby nazwami ulic i numerami — lecz nie domów, a kawałków informacji. W większości przypadków można łatwo określić potrzebny adres URL. Wystarczy skierować przeglądarkę w pożądane miejsce, a następnie przekopiować do kodu łącza długi łańcuch znaków pojawiający się w okienku adresu. Wiedza o tym, jak zbudowany jest adres URL (i czemu bywa tak długi i skomplikowany), bywa czasem jednak przydatna. Co więcej, umiejętność określenia adresu URL własnej witryny okaże się konieczna, aby inni mogli ją odwiedzić.

W tym podrozdziale omówimy poszczególne części adresów URL, ich zastosowanie w lokalizowaniu informacji na WWW oraz rodzaje adresów URL, których się zwykle używa (protokoły HTTP, FTP, typ mailto i inne).

Części adresu URL

Na większość adresów URL składają się trzy (z grubsza) części: protokół, nazwa komputera i nazwa katalogu lub pliku (rysunek 3.15).



Rysunek 3.15. *Części adresu URL*

Protokołem nazywamy sposób dostępu do danych; innymi słowy jest to środek komunikacji służący przeglądarce do pobrania pliku. Jeśli przeglądarka ma użyć protokołu HTTP, w części adresu określającej protokół pojawić się musi skrót `http`. Jeśli przeglądarka ma użyć protokołu FTP, ma to być skrót `ftp`. Wskazany protokół musi od-

powiadać używanemu przez serwer, który ma udostępnić informacje. Nie można oczekiwać uzyskania danych, dajmy na to, protokołem FTP, jeśli na zdalnym komputerze program serwera FTP nie jest zainstalowany.

Nazwa komputera określa system, w zasobach którego informacje są przechowywan, np. *www.helion.pl*, *ftp.apple.com* lub *www.gazeta.pl*. Można też używać tej samej nazwy komputera z różnymi protokołami. Powstają wtedy inne adresy URL — jak na przykład:

http://mojsystem.com.pl

ftp://mojsystem.com.pl

Takie dwa adresy URL umożliwiają dostęp do dwóch różnych usług informacyjnych pojedynczego komputera. Przeglądarka zastosuje inną metodę łączenia się w każdym z tych przypadków. Jeśli oba typy programów serwera są zainstalowane na komputerze zdalnym, nie będzie problemu.

Część adresu URL określająca nazwę komputera może zawierać numer portu. Potrzebny jest on tylko wówczas, gdy serwer został ustawiony w taki sposób, że odpowiada na połączenia skierowane właśnie pod ten jeden, wyznaczony numer portu. Jeśli serwer wysłuchuje zapytań pod domyślnym numerem portu, można ów numer spokojnie pominąć. Jednak gdy podanie numeru portu jest konieczne, umieszcza się je po nazwie komputera, lecz przed nazwą katalogu, jak poniżej:

http://moj-publicznie-dostepny-unix.com.pl:1550/pub/plik

Wreszcie nazwy katalogu i pliku określają plik lub innego rodzaju cząstkę danych, których oczekujemy od serwera. Nazwa katalogu może odpowiadać rzeczywistej ścieżce dostępu na dysku serwera, może też być pewnego rodzaju wyznacznikiem używanym w danym protokole do odwołania się do informacji. Na przykład katalogi protokołu Świstak (ang. *Gopher*, poprzednik WWW) nie są dosłownie katalogami.

Znaki specjalne w adresach URL

Znakiem specjalnym w adresie URL nazywamy taki znak, który nie jest ani małą, ani wielką literą, ani cyfrą, ani nie jest to znak dolara (`$`), myślnik (`-`), znak podkreślenia (`_`), kropka (`.`) lub znak plus (`+`). W przypadku wszelkich innych znaków może zajść potrzeba zastępowania ich specjalnymi kodami sterującymi, aby zapobiec fałszywej ich interpretacji.

Kody sterujące w adresach URL składają się ze znaku procenta (`%`) i dwucyfrowej liczby szesnastkowej. Liczba ta odpowiada kodowi znaku według numeracji zestawu znaków ISO-Latin-1 (nadzbiór standardu ASCII). Przykładowo zapis `%20` oznacza znak spacji, `%3f` to znak zapytania, a `%2f` — ukośnik.

Załóżmy, że mamy katalog o nazwie *wszystkie moje pliki*. Pierwsza próba zapisania adresu URL zawierającego tę nazwę zakończyłaby się zapewne tak:

http://mojkomputerglowny.pl/dysktwardy/wszystkie moje pliki/www/plik.html

Jeśli umieściłoby się taki adres URL ujęty w cudzysłów w znaczniku tworzącym łącze, mogłoby to nawet zadziałać (lecz tylko pod warunkiem, że użyłoby się cudzysłowu). Jednakże ze względu na fakt, że spacje w adresach URL mają charakter znaków specjalnych, niektóre przeglądarki mogłyby mieć z nimi kłopot i nie odczytać poprawnie zawierającego je adresu. Dla pełnej zgodności z wszelkimi przeglądarkami należy stosować dla spacji zapis `%20`:

http://mojkomputerglowny.pl/dysktwardy/wszystkie%20moje%20pliki/www/plik.html

Jak można zauważyć, pojawia się problem z zapisem samego znaku procenta jako takiego. Jeśli nazwa pliku zawiera taki znak, w adresie URL zastępuje się go zapisem `%25`.

W większości przypadków, jeśli konsekwentnie stosuje się krótkie nazwy katalogów złożone wyłącznie ze znaków alfanumerycznych, nie trzeba używać w adresach URL żadnych znaków specjalnych. Warto o tym pamiętać podczas pracy nad stronami WWW.



W języku HTML 4.01 przewidziano jeszcze inne prócz dotąd omawianych atrybuty znacznika `<a>`, które są jednak rzadziej spotykane. Należą do nich:

- `tabindex` — pozwala autorom stron definiować kolejność zmian aktywnego łącza. Naciskając klawisz tabulacji, użytkownik może zmieniać aktualnie aktywny element w sposób analogiczny, jak w oknach dialogowych systemów Windows lub Mac OS;
- atrybuty do obsługi zdarzeń, takie jak `onclick`, `onfocus` oraz `onblur`.

Rodzaje adresów URL

Specyfikacja definiuje wiele rodzajów adresów URL (specyfikacją URL, podobnie jak i specyfikacją HTML, zajmują się instytucje zrzeszone w konsorcjum W3C). Ten podrozdział opisuje niektóre popularne rodzaje adresów URL i potencjalne pułapki czyhające na użytkowników.

Protokół HTTP

Protokołu HTTP (ang. *Hypertext Transport Protocol* — protokół przesyłu hipertekstu) używają serwery WWW do przesyłania informacji do przeglądarek. Adresy URL typu HTTP są znacznie popularniejsze od jakichkolwiek innych, ponieważ wskazują dokumenty na WWW. Powielają one podstawowy wzorzec adresu URL:

```
http://www.przyklad.pl/rzeczy/blablabla/
```

Jeśli adres URL kończy się ukośnikiem, ostatni człon adresu uważany jest za nazwę katalogu. Serwer w odpowiedzi na takie żądanie wysyła plik domyślny dla danego katalogu. Zwykle jest to plik o nazwie *index.html* (jeśli dana strona WWW jest stroną główną, nadrzędną wobec pozostałych plików w katalogu, nazwanie jej *index.html* jest bardzo dobrym pomysłem).

Można także określić nazwę żądanego pliku dosłownie. Wówczas wczytywany jest plik, którego nazwa znajduje się na końcu adresu URL, jak w następujących przykładach:

```
http://www.przyklad.pl/rzeczy/blablabla/index.html
http://www.przyklad.pl/rzeczy/blablabla/glowna.html
```

Adresy URL takie, jak poniżej (w którym człon *blablabla* jest nazwą katalogu) też zwykle są akceptowane:

```
http://www.przyklad.pl/rzeczy/blablabla
```

W tym ostatnim przypadku, ponieważ *blablabla* jest nazwą katalogu, na końcu powinien pojawiać się ukośnik. Większość serwerów WWW potrafi domyślić się, że chodzi o katalog, i przekierować przeglądarkę do właściwego pliku. Jednak niektóre przeglądarki i starsze serwery mogą sprawiać trudności, napotkawszy taki adres URL. Lepiej zawsze wskazywać katalogi i pliki dosłownie, upewniając się, czy plik o domyślnej nazwie istnieje (jeśli podajemy tylko nazwę katalogu).

Anonimowy dostęp do serwerów FTP

Adresy URL typu FTP służą do wskazywania plików zlokalizowanych na serwerach FTP. Zwykle chodzi o serwery dopuszczające dostęp anonimowy, czyli takie, do których można się zalogować, podając nazwę użytkownika `anonymous` i, jako hasło, adres poczty elektronicznej. Adresy URL typu FTP powielają podstawowy wzorzec adresów URL, co pokazują poniższe przykłady:

```
ftp://ftp.przyklad.pl/rzeczy/blablabla
ftp://ftp.przyklad.pl/rzeczy/blablabla/stronaglowna.html
```

Ze względu na możliwość uzyskania z serwera FTP spisu plików w danym katalogu ograniczenia dotyczące stosowania na końcu adresu ukośnika są odmienne niż w przypadku protokołu HTTP. Podanie pierwszego z przykładowych adresów oznacza żądanie spisu plików w katalogu *blablabla*. Podanie drugiego powoduje pobranie i parsowanie pliku *stronaglowna.html* w katalogu *blablabla*.



Przeglądanie katalogów na serwerach FTP za pomocą przeglądarki WWW bywa znacznie wolniejsze niż przeglądanie za pomocą specjalizowanego programu klienta FTP. Przeglądarka bowiem nie utrzymuje otwartego połączenia FTP, lecz zamyka je natychmiast po pobraniu pliku lub spisu zawartości katalogu. Jeśli użytkownik wybiera wyświetlone łącze do pliku lub katalogu z pobranego spisu, przeglądarka układa nowy adres URL typu FTP z nazwą wybranego pliku, ponownie otwiera połączenie FTP z nowym adresem, pobiera plik lub spis katalogu i znowu zamyka połączenie. W tej sytuacji adresy URL typu FTP najlepiej sprawdzają się wówczas, gdy użytkownik wie dokładnie, który plik chce pobrać, a nie wtedy, gdy chodzi o przeglądanie archiwum plików.

Mimo że przeglądarka pobiera plik protokołem FTP, jeśli tylko jest to plik HTML — wyświetli go tak samo, jak gdyby korzystano z protokołu HTTP. W przypadku przeglądarek WWW metoda dostępu nie ma znaczenia. Jeśli tylko rozpoznają plik HTML — czy to dzięki jednoznacznej informacji z serwera, czy to po rozszerzeniu nazwy pliku — parsują i wyświetlają treść pliku jako stronę WWW. Jeśli plik nie zostaje rozpoznany jako

kod HTML, nie dzieje się nic strasznego. Przeglądarki umieją określić i wyświetlić różne rodzaje plików, poza tym umożliwiają zapisywanie na dysku plików wszelakiego rodzaju.

Dostęp do serwerów FTP z podaniem nazwy użytkownika

Wszystkie adresy URL typu FTP omawiane w poprzednim podrozdziale służą do łączenia się z serwerami FTP umożliwiającymi dostęp anonimowy. Można ułożyć także adres URL do nazwanych kont na serwerach FTP, jak poniżej:

```
ftp://login:haslo@ftp.przyklad.pl/rzeczy/blablabla/stronaglowna.html
```

W takim adresie URL *login* oznacza nazwę konta na serwerze FTP, natomiast *haslo* — odpowiednie dla podanej nazwy konta hasło. Należy zwrócić uwagę, że hasło zapisane jest dosłownie w adresie URL. Trzeba bardzo uważać, żeby nikt nie oglądał w ten sposób zapisanych adresów — i nie umieszczać ich tam, gdzie ktoś niepowołany mógłby je odczytać[2].

Ponadto używane adresy URL mogą zostać zapisane w pamięci podręcznej lub pliku dziennika tak w komputerze lokalnym, jak i w serwerze pośredniczącym, gdzieś pomiędzy przeglądarką a serwerem docelowym. Z tego powodu prawdopodobnie najlepiej w ogóle unikać stosowania adresów URL typu FTP z nazwą użytkownika.

Adresy URL typu mailto

Adresy URL typu mailto (ang. *mail to*, dosłownie — „poczta do") służą do wysyłania wiadomości poczty elektronicznej. Jeśli przeglądarka obsługuje adresy URL typu mailto, wybranie łącza z takim adresem powoduje otwarcie okna dialogowego, gdzie można wpisać temat i treść listu, który następnie zostaje wysłany pod określony adres poczty elektronicznej. Ze względu na niuanse ustawień przeglądarki i programu pocztowego w komputerze użytkownika, łącza mailto mogą zupełnie nie działać.

Składnia adresu URL typu mailto różni się od podstawowego wzorca adresu URL i ma następującą postać:

```
mailto:adres_poczty_elektronicznej
```

— czyli na przykład:

```
mailto:lemay@lne.com
```

W przeciwieństwie do pozostałych opisywanych tu typów adresów URL, obsługa adresu URL typu mailto zachodzi wyłącznie po stronie klienta. Jest to jak gdyby polecenie dla przeglądarki, aby utworzyć wiadomość poczty elektronicznej zaadresowaną do

[2] Warto pamiętać, że każda przeglądarka umożliwia wgląd w kod źródłowy dowolnej strony HTML. Na przykład w przeglądarce Internet Explorer, aby zobaczyć kod źródłowy aktualnie otwartej strony, wystarczy wybrać polecenie *Źródło* z menu *Widok — przyp. tłum.*

określonego odbiorcy. Sposób, w jaki się to odbywa, zależy od przeglądarki. Większość przeglądarek umożliwia także ustalenie w adresie URL domyślnego tematu nowego listu, na przykład:

```
mailto:lemay@lne.com?subject=Pozdrowienia z Polski
```

Wybranie łącza przez użytkownika spowoduje w większości przeglądarek wstawienie podanego tematu do odpowiedniego okienka.

Grupy dyskusyjne Usenet

Adresy URL grup dyskusyjnych przyjmują jedną z dwóch postaci:

```
news:nazwa_grupy
news:identyfikator_artykułu
```

W pierwszym przypadku chodzi o wskazanie grupy dyskusyjnej, czyli na przykład *pl.comp.www* albo *alt.gothic*. Jeśli przeglądarka obsługuje adresy URL grup dyskusyjnych (bezpośrednio lub za pomocą programu czytnika), wyświetli się spis artykułów dostępnych w danej grupie dyskusyjnej.

Druga postać pozwala przeczytać określony artykuł w danej grupie dyskusyjnej. Każdy artykuł ma swój unikatowy identyfikator (ang. *message ID* — identyfikator wiadomości), który zwykle wygląda tak[3]:

```
<lemayCt76Jq.CwG@netcom.com>
```

Aby ułożyć adres URL zawierający identyfikator artykułu, trzeba usunąć nawiasy trójkątne i dodać przedrostek `news:`:

```
news:lemayCt76Jq.CwG@netcom.com
```

Należy pamiętać, że artykuły nie są wieczne — po upływie pewnego czasu są usuwane z serwera. W związku z tym identyfikator artykułu, który w danym momencie jest poprawny, po krótkim czasie może się zdezaktualizować. Aby na stałe dołączyć atrykuł do swojej witryny, trzeba go po prostu skopiować i umieścić w osobnym pliku.

W przypadku obu postaci adresu zakłada się, że użytkownik ma dostęp do grup dyskusyjnych przez serwer NNTP, a do takiego można odwołać się tylko wówczas, gdy w zmiennych środowiskowych lub w ustawieniach przeglądarki określono jego nazwę. Z tej przyczyny adresy URL grup dyskusyjnych są najbardziej przydatne po prostu do otwierania określonych artykułów lokalnie, a do tworzenia łączy na stronach — niekoniecznie.

[3] Identyfikator artykułu można określić, wyświetlając jego kod źródłowy w czytniku grup dyskusyjnych lub za pomocą odpowiedniej funkcji takiego programu — *przyp. tłum.*



Adresy URL grup dyskusyjnych, podobnie jak adresy typu mailto, mogą nie być obsługiwane przez wszystkie przeglądarki.

Plikowe adresy URL

Plikowe adresy URL są przeznaczone do odwoływania się do plików przechowywanych na dysku lokalnym. Innymi słowy, wskazują pliki zlokalizowane w tym samym systemie plików, w którym działa przeglądarka. Adresy URL plików lokalnych przyjmują jedną z dwóch postaci: w pierwszej brak jest nazwy komputera (a więc pojawiają się trzy ukośniki pod rząd), w drugiej nazwa komputera jest wyszczególniona — w poniższym przykładzie brzmi `localhost`:

```
file:///kat1/kat2/plik
file://localhost/dir1/dir2/plik
```

W zależności od używanej przeglądarki odpowiedni będzie pierwszy, drugi lub oba schematy.

Plikowe adresy URL są bardzo podobne do adresów URL typu FTP. De facto, jeśli nazwa komputera jest podana i nie jest nią słowo `localhost`, przeglądarka podejmie próbę pobrania pliku za pomocą protokołu FTP. Oba poniższe przykłady spowodowałyby wczytanie pliku w ten sam sposób:

```
file://jakistamsystem.pl/pub/kat/blablabla/plik.html
ftp://jakistamsystem.pl/pub/kat/blablabla/plik.html
```

Prawdopodobnie najlepszym zastosowaniem plikowego adresu URL jest wskazanie strony startowej przeglądarki (nazywanej też *stroną główną*). W tym przypadku, ponieważ prawie zawsze chodzi o odwołanie do dysku lokalnego, używanie plikowego adresu URL ma sens.

Kłopot z plikowymi adresami URL polega na tym, że odwołują się one do plików lokalnych, przy czym lokalność oznacza tutaj umiejscowienie w systemie plików, w którym działa przeglądarka — a nie w systemie, z którego pobrano stronę! Jeśli jeden użytkownik zastosuje plikowe adresy URL w łączach na swojej stronie, a drugi, na swoim komputerze, wybierze jedno z tych łączy — przeglądarka podejmie próbę otwarcia pliku na dysku komputera drugiego użytkownika (zakończoną zwykle fiaskiem). Co więcej, ponieważ plikowy adres URL zawiera bezwzględną ścieżkę dostępu do pliku, zastosowanie na stronie plikowego adresu URL uniemożliwia przeniesienie strony w inne miejsce lub do innego komputera.

Jeśli potrzebne jest odwołanie do plików, które znajdują się w tym samym systemie plików lub nawet w tym samym katalogu, co bieżąca strona, należy używać ścieżek względnych zamiast plikowych adresów URL. Dzięki ścieżkom względnym (w odniesieniu do plików lokalnych) i innym typom adresów URL (w odniesieniu do plików zdalnych) w ogóle nie powinna wystąpić potrzeba stosowania plikowych adresów URL.

Podsumowanie

W tym rozdziale scharakteryzowaliśmy łącza. Łącza zmieniają WWW z kolekcji niezwiązanych z sobą stron w olbrzymi system zjednoczonych informacji (jeśli wolno tu przywołać wielkie słowa).

Łącza tworzy się za pomocą pary znaczników `<a>…</a>`, czyli znaczników tworzących łącza lub zakotwiczenia. Najważniejszym atrybutem znacznika tworzącego łącze jest atrybut `href` (wskazuje pliki, do których prowadzi łącze), a znacznika tworzącego zakotwiczenia — atrybut `name` (określa nazwę zakotwiczenia).

Kiedy łączy się strony przechowywane na dysku lokalnym, ich położenie można zapisać w atrybucie `href` jako ścieżkę względną lub bezwzględną. W przypadku łączy lokalnych zalecane są ścieżki względne, które umożliwiają łatwe przenoszenie stron lokalnych do innego katalogu lub na inny komputer. Zastosowanie ścieżek bezwzględnych grozi uszkodzeniem łączy podczas przemieszczania plików.

Aby podłączyć stronę dostępną na WWW (stronę zdalną), jako wartość atrybutu `href` trzeba wpisać adres URL tej strony. Można ten adres łatwo skopiować: po prostu otwiera się stronę w przeglądarce, a potem zaznacza się i kopiuje z okienka adres URL, aby na koniec wkleić go w odpowiednie miejsce w kodzie znacznika.

Utworzenie łącza do określonej części strony polega na przygotowaniu tam zakotwiczenia za pomocą znacznika `<a>…</a>`. Atrybut `href` zastępuje się atrybutem `name`, któremu przypisuje się nazwę zakotwiczenia. Można wówczas tworzyć łącza do zakotwiczenia o znanej nazwie za pomocą adresu złożonego z adresu strony, znaku # i nazwy zakotwiczenia.

Na końcu opisano adresy URL, których funkcją jest wskazywanie stron, plików i innych zasobów informacji w internecie. W zależności od rodzaju zasobu adresy URL składają się z różnych części, ale zwykle jest to po prostu nazwa protokołu i określenie położenia. Adresy URL mogą służyć do lokalizowania różnych rodzajów zasobów informacji, zwykle jednak wskazują strony WWW (przedrostek `http`), katalogi lub pliki na serwerach FTP (`ftp`), adresy poczty elektronicznej (`mailto`) lub grup dyskusyjnych Usenet (`news`).