# HTML i XHTML
## przewodnik encyklopedyczny


Autorzy: Chuck Musciano, Bill Kennedy
Tłumaczenie: Adam Podstawczyński
ISBN: 83-7197-440-X
Tytuł oryginału: HTML and XHTML. The definitive guide. Fourth Edition.
Format: B5, stron: 700

Język HTML zmienia się tak szybko, że trudno nadążyć za coraz to nowszymi technologiami, które się z nim wiążą. Skąd wiedzieć, czego i jak należy używać? „HTML i XHTML. Przewodnik encyklopedyczny" pomaga odpowiedzieć na te pytania. Książka ta sposób najbardziej całościowy ze wszystkich dostępnych w sprzedaży traktuje o języku HTML. Opisuje najświeższe standardy, HTML 4.01 i XHTML 1.0, a także wszystkie funkcje obsługiwane przez popularne przeglądarki.
Poznawanie HTML-a lub XHTML-a to jak poznawanie każdego innego języka. Większość uczących się zaczyna od przyswojenia przykładów. Czerpanie z doświadczenia innych osób to naturalny, prosty i atrakcyjny sposób nauki. Ale czerpanie wiedzy z czyichś rozwiązań ma też swoje ograniczenia. Przecież przykład może być dobry albo zły. Lepiej przyswajać sobie HTML trzymając w ręku całościowy materiał referencyjny, obejmujący składnię, semantykę i wariacje języka oraz pomagający w rozróżnieniu co jest złym, a co dobrym przykładem użycia HTML-a.
Książka pomaga Czytelnikowi na oba sposoby: autorzy szczegółowo opisują każdy element obu standardów i wyjaśniają zasadę jego działania oraz sposób interakcji z innymi elementami. Przedstawiono wiele praktycznych wskazówek pomocnych przy tworzeniu zarówno prostych podręczników online, jak i złożonych prezentacji marketingowych. Setki przykładów ułatwiają Czytelnikowi stworzyć wydajną stronę WWW, a także opanować bardziej zaawansowane mechanizmy publikacji w Sieci. Książka opisuje również, jak „przestawić się" z HTML-a na XHTML.
W książce opisano następujące zagadnienia:

- Arkusze stylów i ich wpływ na wygląd dokumentu
- Tabele (od prostych do złożonych)
- Ramki pomagające w prezentowaniu grup dokumentów
- Sposób projektowania i tworzenia interaktywnych formularzy i dokumentów dynamicznych
- Sposób umieszczania na stronach grafiki, plików dźwiękowych, filmów, apletów i programów JavaScript
- Sposób tworzenia dokumentów dobrze prezentujących się na różnych przeglądarkach
- XHTML – język publikacji elektronicznych przyszłości

STOP – Najnoszwe informacje! Netscape Navigator 6.0! Internet Explorer 5.0! HTML 4.01! XML i XHTML! Arkusze stylów! Przestają być tajemnicą! STOP Tylko z tym przewodnikiem odnajdziesz właściwą drogę – STOP.

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

# *Spis treści*

*Przedmowa* ..... *11*

*Rozdział 1. HTML, XHTML i World Wide Web* ..... *17*

1.1. Internet, intranety i ekstranety ..... 17
1.2. Internetowy żargon ..... 20
1.3. Czym jest HTML? ..... 23
1.4. Czym jest XHTML? ..... 24
1.5. Czym nie są HTML i XHTML? ..... 24
1.6. Niestandardowe rozszerzenia ..... 25
1.7. Narzędzia dla projektanta stron WWW ..... 27

*Rozdział 2. Szybki start* ..... *31*

2.1. Narzędzia ..... 31
2.2. Pierwszy dokument HTML ..... 32
2.3. Zagnieżdżone znaczniki ..... 33
2.4. Szkielet dokumentu HTML ..... 34
2.5. Sedno dokumentu HTML lub XHTML ..... 35
2.6. Tekst ..... 36
2.7. Odsyłacze ..... 40
2.8. Elementy graficzne — coś specjalnego ..... 43
2.9. Listy, dokumenty z możliwością przeszukiwania, formularze ..... 45
2.10. Tabele ..... 47
2.11. Ramki ..... 48
2.12. Arkusze stylów i JavaScript ..... 49
2.13. Co dalej? ..... 50

*Rozdział 3. Anatomia dokumentu HTML ..... 51*
3.1. Zwodniczy wygląd ..... 51
3.2. Struktura dokumentu HTML ..... 52
3.3. Znaczniki i atrybuty ..... 53
3.4. Poprawnie uformowane dokumenty a XHTML ..... 56
3.5. Zawartość dokumentu ..... 57
3.6. Elementy dokumentu HTML ..... 59
3.7. Nagłówek dokumentu ..... 62
3.8. Treść dokumentu ..... 65
3.9. Znaczniki redaktorskie ..... 67
3.10. Znacznik <bdo> ..... 70

*Rozdział 4. Podstawowe operacje na tekście ..... 73*
4.1. Działy i akapity ..... 73
4.2. Nagłówki ..... 80
4.3. Zmiana wyglądu tekstu ..... 86
4.4. Znaczniki stylów opartych na zawartości ..... 87
4.5. Znaczniki stylów fizycznych ..... 94
4.6. Rozszerzona obsługa fontów w HTML-u ..... 98
4.7. Precyzyjne sterowanie odstępami i układem ..... 104
4.8. Cytaty blokowe ..... 117
4.9. Adresy ..... 120
4.10. Specjalne kodowanie znaków ..... 122

*Rozdział 5. Linie, grafika i multimedia ..... 125*
5.1. Linie poziome ..... 125
5.2. Wstawianie elementów graficznych ..... 132
5.3. Kolory dokumentu i grafika w tle ..... 155
5.4. Dźwięk w tle ..... 162
5.5. Animacja tekstu ..... 164
5.6. Inna zawartość multimedialna ..... 167

*Rozdział 6. Odsyłacze i „pajęczyny” ..... 171*
6.1. Podstawy hipertekstu ..... 171
6.2. Odwoływanie się do dokumentów: adres URL ..... 172
6.3. Tworzenie odsyłaczy ..... 187
6.4. Wydajne odsyłacze ..... 195

6.5. Obrazki zawierające mapy odsyłaczy ... 199
6.6. Tworzenie dokumentów z możliwością przeszukiwania ... 209
6.7. Relacje ... 212
6.8. Wspomaganie automatyzacji ... 217

***Rozdział 7. Listy formatowane ... 221***
7.1. Listy nieuporządkowane ... 221
7.2. Listy uporządkowane ... 224
7.3. Znacznik <li> ... 227
7.4. Zagnieżdżanie list ... 230
7.5. Listy definicji ... 232
7.6. Poprawne korzystanie z list ... 236
7.7. Listy typu „katalog” ... 237
7.8. Listy typu „menu” ... 238

***Rozdział 8. Kaskadowe arkusze stylów ... 241***
8.1. Elementy stylów ... 242
8.2. Składnia arkuszy stylów ... 250
8.3. Klasy stylów ... 255
8.4. Właściwości ... 260
8.5. Style „bezznacznikowe”: znacznik  ... 288
8.6. Stosowanie stylów w dokumentach ... 289

***Rozdział 9. Formularze ... 293***
9.1. Formularze — podstawy ... 293
9.2. Znacznik <form> ... 294
9.3. Przykład prostego formularza ... 301
9.4. Pobieranie danych poprzez e-mail ... 302
9.5. Znacznik <input> ... 304
9.6. Znacznik <button> ... 315
9.7. Obszary tekstu wielowierszowego ... 317
9.8. Elementy umożliwiające wybór ... 319
9.9. Ogólne atrybuty elementów formularzy ... 323
9.10. Oznaczanie i grupowanie elementów formularza ... 327
9.11. Efektywne formularze ... 331
9.12. Programowanie formularzy ... 334

*Rozdział 10. Tabele ..... 341*
10.1. Standardowy model tabeli ..... 341
10.2. Znaczniki tworzące tabelę ..... 343
10.3. Najnowsze znaczniki związane z tabelami ..... 359
10.4. Więcej niż zwykłe tabele ..... 370

*Rozdział 11. Ramki ..... 371*
11.1. Ramki — charakterystyka ..... 371
11.2. Znaczniki opisujące ramki ..... 372
11.3. Układ ramkowy ..... 373
11.4. Zawartość ramek ..... 379
11.5. Znacznik <noframes> ..... 382
11.6. Ramki zagnieżdżone ..... 383
11.7. Ramki nazwane lub okna docelowe ..... 385

*Rozdział 12. Zawartość wykonywalna ..... 391*
12.1. Aplety i obiekty ..... 391
12.2. Zawartość zagnieżdżona ..... 394
12.3. JavaScript ..... 409
12.4. Arkusze stylów JavaScript ..... 417

*Rozdział 13. Zawartość dynamiczna ..... 425*
13.1. Dokumenty dynamiczne — przegląd ..... 425
13.2. Dokumenty pobierane przez klienta ..... 426
13.3. Dokumenty wypychane przez serwer ..... 430

*Rozdział 14. Rozszerzenia Netscape ..... 435*
14.1. Pusta przestrzeń ..... 436
14.2. Układ wielokolumnowy ..... 440
14.3. Warstwy ..... 445

*Rozdział 15. XML ..... 457*
15.1. Języki i metajęzyki ..... 458
15.2. Dokumenty i definicje DTD ..... 460
15.3. Zrozumieć definicje XML DTD ..... 461
15.4. Gramatyka elementów ..... 465

15.5. Atrybuty elementów....469
15.6. Bloki warunkowe....471
15.7. Tworzenie definicji XML DTD....472
15.8. Korzystanie z XML-a....473

***Rozdział 16. XHTML....477***
16.1. Dlaczego XHTML?....477
16.2. Tworzenie dokumentów XHTML....479
16.3. HTML kontra XHTML....482
16.4. Czy korzystać z XHTML-a?....486

***Rozdział 17. „Kruczki i sztuczki”....491***
17.1. Porada dnia....491
17.2. Drobiazg czy nadużycie?....493
17.3. Specjalne wypunktowanie....493
17.4. Sztuczki z tabelami....494
17.5. Przezroczyste obrazki....501
17.6. Triki z oknami i ramkami....503

**Dodatki....507**

***Dodatek A Gramatyka języka HTML....509***
Konwencje gramatyczne....509
Gramatyka....511

***Dodatek B Znaczniki HTML/XHTML....521***
Podstawowe atrybuty....521
Spis znaczników i atrybutów HTML....522

***Dodatek C Właściwości arkuszy stylów....559***

***Dodatek D Definicja DTD HTML 4.01....565***

***Dodatek E Definicja DTD XHTML 1.0....579***

***Dodatek F Spis encji....595***

*Dodatek G Nazwy i wartości kolorów* ..... *601*
Wartości kolorów ..... 601
Nazwy kolorów ..... 601
Standardowa paleta kolorów ..... 603

*Skorowidz* ..... *605*

# 5

# *Linie, grafika i multimedia*

Tekst stanowi najczęściej większość zawartości dokumentu. „Czyste" informacje tekstowe warto jednak czasem uatrakcyjnić za pomocą linii poziomych, obrazków lub innych elementów graficznych. Takie „wstawki" nie muszą pełnić roli tylko dekoracyjnej. Ożywiają dokument i umożliwiają przekazanie informacji innego typu, często niedostępnej na innych nośnikach (np. w druku). W niniejszym rozdziale dokładnie opisano sposób wstawiania do dokumentów elementów multimedialnych, a także kiedy warto je stosować, a kiedy ich unikać.

Część Czytelników być może zechce także zerknąć teraz do rozdziału 12., *Zawartość wykonywalna*. Opisane tam zostały znaczniki ogólne: zdefiniowany w HTML 4 i XHTML `<object>` oraz obsługiwany przez Netscape `<embed>`. Umożliwiają one wstawianie do dokumentu dowolnej zawartości i typów danych, w tym multimediów.

## *5.1. Linie poziome*

Linie poziome (ang. *horizontal rules*) pozwalają na wizualne, przejrzyste rozdzielenie części dokumentu. Za ich pomocą można w prosty sposób wydzielić niewielką porcję zawartości, odgraniczyć nagłówek i stopkę, czy dodatkowo podkreślić nagłówki części.

### *5.1.1. Znacznik <hr>*

Znacznik `<hr>` informuje przeglądarkę, że w danym miejscu ma zostać wyświetlona linia pozioma. Podobnie jak znacznik `<br>`, `<hr>` powoduje przełamanie wiersza, ale dodatkowo wymusza jeszcze domyślne wyrównanie następnego akapitu (do lewej strony). Przeglądarka umieszcza linię bezpośrednio pod bieżącym wierszem; normalny „tok" tekstu przywracany jest pod linią.

Sposób wyświetlania linii poziomej zależy od przeglądarki. Zazwyczaj linia rysowana jest na całej szerokości dokumentu. Przeglądarki graficzne mogą ozdabiać linię „efektami specjalnymi" — linia „wyciśnięta" lub „wyżłobiona". Przeglądarki tekstowe wykorzystują w tym miejscu najczęściej sekwencję łączników lub znaków podkreślenia.

Ani nad, ani pod linią nie jest wstawiany żaden dodatkowy odstęp. Jeśli chcemy taki odstęp uzyskać, musimy jawnie umieścić linię w znacznikach akapitu. Na przykład, spójrzmy jak wstawiane są odstępy w poniższym kodzie źródłowym oraz jak wyglądają one na rysunku 5.1.

## <hr>

***Funkcja:***

Przerywa ciągłość tekstu i wstawia poziomą linię

***Atrybuty:***

| | |
|---|---|
| ALIGN | ONMOUSEMOVE |
| CLASS | ONMOUSEOUT |
| COLOR O | ONMOUSEOVER |
| ID | ONMOUSEUP |
| NOSHADE | LANG |
| ONCLICK | DIR |
| ONDBLCLICK | SIZE |
| ONKEYDOWN | STYLE |
| ONKEYPRESS | TITLE |
| ONKEYUP | WIDTH |
| ONMOUSEDOWN | |

***Znacznik zamykający:***

w HTML-u brak, w XHTML-u `</hr>` lub `<hr ... />`

***Zawiera:***

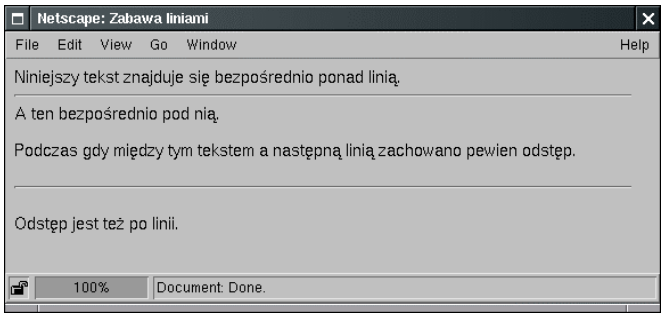
Nic

***Stosowany wewnątrz:***

body_content

```
Niniejszy tekst znajduje się bezpośrednio ponad linią.
<hr>
A ten bezpośrednio pod nią.
<p>
Podczas gdy między tym tekstem a następną linią zachowano pewien odstęp.
<p>
<hr>
<p>
Odstęp jest też po linii.
```

Znacznik akapitu po znaczniku linii jest konieczny, jeśli chcemy, aby tekst pod linią był wyrównany w jakikolwiek inny sposób niż wyrównanie domyślne (do lewej).

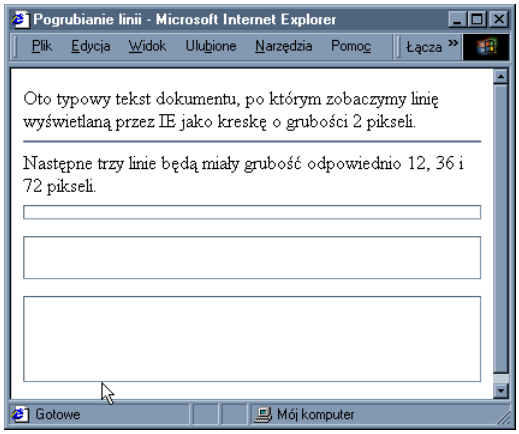



*Rysunek 5.1. Znaczniki akapitu zwiększają odstępy przed linią i po niej*

### *5.1.1.1. Atrybut size*

Zazwyczaj przeglądarki wyświetlają linie poziome o grubości jednego lub dwóch pikseli[1], trójwymiarowe i „wyżłobione" — jakby „wciśnięte" w stronę. Linię można pogrubić za pomocą atrybutu `size`. Wymagana wartość atrybutu to grubość w pikselach. Spójrzmy na poniższy przykład i rysunek 5.2.

```
<p>
Oto typowy tekst dokumentu, po którym zobaczymy linię wyświetlaną
przez IE jako kreskę o grubości 2 pikseli.
<hr>
Następne trzy linie będą miały grubość odpowiednio 12, 36 i 72 pikseli.
<hr size=12>
<hr size=36>
<hr size=72>
```




*Rysunek 5.2. Internet Explorer i Netscape pozwalają zmieniać grubość linii*

### *5.1.1.2. Atrybut noshade*

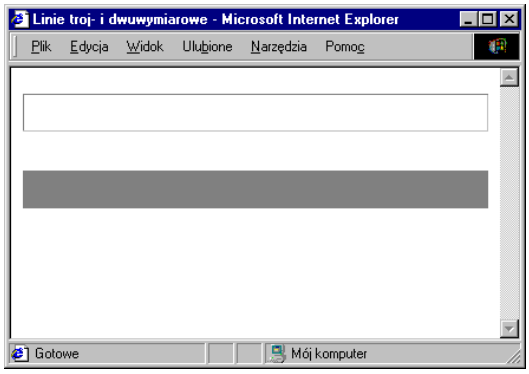
Nie musi nam się podobać linia rysowana „w trzech wymiarach". Po dodaniu atrybutu `noshade` uzyskana linia będzie dwuwymiarowa. W HTML-u atrybut ten nie wymaga podania żadnej wartości, w XHTML-u użyjemy `noshade="noshade"`. Porównajmy „normalną" linię 3D z tą na rysunku 5.3 (specjalnie pogrubioną tak, aby różnica była bardziej ewidentna):

```
<hr size=32>
<p>
<hr size=32 noshade>
```

Standardy HTML 4 i XHTML nie zalecają korzystania z atrybutu `noshade`, ten sam efekt można uzyskać za pomocą stylów.

---

[1] Piksel to jeden z wielu małych punktów składających się na wyświetlany obraz komputerowy. Ekrany mają różne przekątne, ale zazwyczaj jeden piksel równa się jednemu punktowi na monitorze o rozdzielczości 75 dpi (punktów na cal). Punkt to jednostka miary wykorzystywana w drukowaniu i równa ok. 1/72 cala (dokładnie w calu jest 72,27 punkta). Typowy krój czcionki wykorzystywany w popularnych przeglądarkach ma wysokość 12 punktów — czyli na jeden cal przypada sześć linii tekstu.



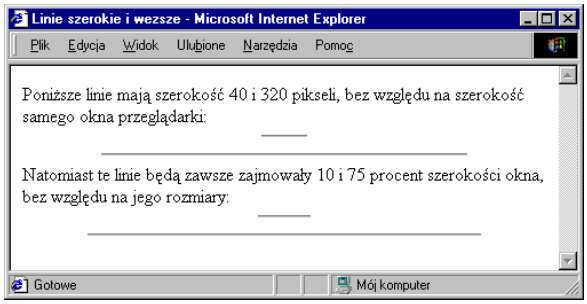

*Rysunek 5.3. Linie trój- i dwuwymiarowa*

### 5.1.1.3. Atrybut width

Domyślnie linia rysowana jest na całej szerokości okna przeglądarki. Można to zmienić za pomocą atrybutu `width` — umożliwia on tworzenie linii albo o bezwzględnej szerokości w pikselach, albo zajmujących określoną szerokość procentową otaczającego tekstu. Większość przeglądarek automatycznie wyśrodkowuje linie, tę cechę można zmienić za pomocą atrybutu `align` (podpunkt 5.1.1.4).

Poniżej przedstawiono kilka przykładów użycia parametru `width` przy tworzeniu linii (rysunek 5.4):

```
Poniższe linie mają szerokość 40 i 320 pikseli,
bez względu na szerokość samego okna przeglądarki:
<hr width=40>
<hr width=320>
Natomiast te linie będą zawsze zajmowały 10 i 75
procent szerokości okna, bez względu na jego
rozmiary:
<hr width="10%">
<hr width="75%">
```




*Rysunek 5.4. Linie długie i krótkie, względne i bezwzględne*

Zauważmy również, że wartość względna (procentowa) w atrybucie `width` została umieszczona w cudzysłowach. Tak naprawdę w standardowym HTML-u nie są one absolutnie wymagane[2], ale ponieważ symbol procentu zazwyczaj służy do oznaczania symbolu zakodowanego, pominięcie cudzysłowów może spowodować niepoprawne wyświetlenie dokumentu.

[2] W XHTML-u wszystkie wartości atrybutów umieszcza się w cudzysłowach.

W większości przypadków nie powinno się określać bezwzględnej szerokości linii. Okno przeglądarki może mieć różne wymiary i to, co u jednego użytkownika będzie krótką kreseczką, u innego może się okazać nieproporcjonalnie długą linią. Dlatego zaleca się stosowanie wartości procentowych — po zmianie rozmiaru okna przeglądarki linie zachowają odpowiednią długość względną.

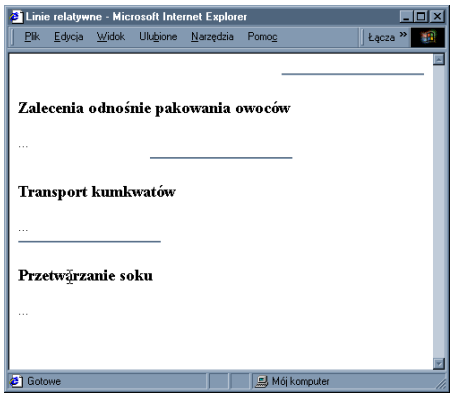
Standardy HTML 4 oraz XHTML nie zalecają już stosowania atrybutu `width` — ten sam efekt można uzyskać za pomocą stylów.

### *5.1.1.4. Atrybut align*

Atrybut `align` w znaczniku `<hr>` może przybierać trzy różne wartości: `left`, `center` lub `right`. Te linie, których szerokość jest mniejsza niż otaczający tekst, zostaną umieszczone w odpowiednim miejscu względem marginesów okna. Domyślne położenie to środek, czyli `center`.

Za pomocą różnie wyrównanych linii można w ciekawy sposób rozdzielać poszczególne części dokumentu. Na przykład, w poniższym fragmencie linia o 35-procentowej szerokości wyświetlana jest kolejno po prawej stronie, pośrodku i po lewej (rysunek 5.5):

```
<hr width="35%" align=right>
<h3>Zalecenia odnośnie pakowania owoców</h3>
...
<hr width="35%" align=center>
<h3>Transport kumkwatów</h3>
...
<hr width="35%" align=left>
<h3>Przetwarzanie soku</h3>
...
```




*Rysunek 5.5. Rozdzielanie części dokumentu za pomocą linii wyrównanych w różny sposób*

Standardy HTML 4 oraz XHTML nie zalecają już stosowania atrybutu `align` — ten sam efekt można uzyskać za pomocą stylów.

### *5.1.1.5. Atrybut color*

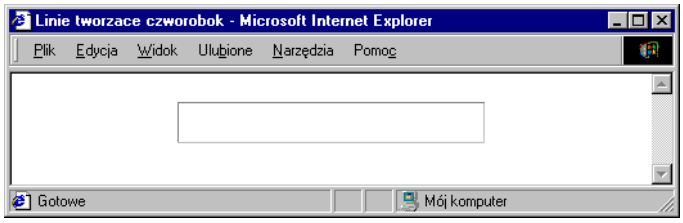
Atrybut `color` jest obsługiwany tylko przez przeglądarkę Internet Explorer i umożliwia zmianę koloru linii. Wartość tego atrybutu to albo nazwa koloru, albo zbiór trzech cyfr szesnastkowych opisujących jego wartość. Pełną listę nazw i wartości kolorów znajdziemy w dodatku G.

Domyślnie linia jest tego samego koloru, co tło dokumentu i ma „wyżłobione" krawędzie (nieco ciemniejsze i jaśniejsze od tła). Po określeniu własnego koloru, czy to poprzez atrybut `color`, czy przez style, efekt trójwymiarowości ulega zniweczeniu.

### 5.1.1.6. Łączenie atrybutów znacznika <hr>

Atrybuty opisujące wygląd linii można łączyć, a ich kolejność nie ma znaczenia. Na przykład, żeby uzyskać duży prostokąt, łączymy atrybuty `size` i `width` (rysunek 5.6):

```
<hr size=32 width=50% align=center>
```




*Rysunek 5.6. Łączenie atrybutów znacznika <hr> pozwala osiągnąć „efekty specjalne"*

Zresztą łączenie niektórych atrybutów w przypadku linii jest wymagane — na przykład, sam atrybut `align` w zasadzie nie ma żadnego znaczenia, bo przecież domyślnie linia rozciąga się na całej szerokości okna.

### 5.1.1.7. Atrybuty class, dir, event, id, lang, style i title

W wielu znacznikach opisujących zawartość obsługiwany jest pewien zestaw wspólnych atrybutów. Pozwalają one na identyfikację (`title`) oraz oznaczenie (`id`) zawartości znacznika w celu późniejszego odwołania się do danego elementu lub uproszczenia automatycznego przetwarzania. Inne umożliwiają zmianę wyglądu elementu (`class`, `style`) oraz określenie języka i kierunku wyświetlania tekstu (`lang` i `dir`). Oczywiście trudno powiedzieć, w jaki sposób te dwa ostatnie mogą wpłynąć na wyświetlanie linii poziomej, tym niemniej stanowią one standardowe atrybuty tego znacznika. *[atrybut `dir`, 3.6.1.1] [atrybut `lang`, 3.6.1.2] [atrybut `id`, 4.1.1.4] [atrybut `title`, 4.1.1.5] [atrybut `style`, 8.1.1] [atrybut `class`, 8.3]*

Ponadto istnieją jeszcze atrybuty opisujące reakcję na różne zdarzenia związane z danym elementem i wymagające pewnych zabiegów programistycznych (atrybuty `on...`). *[procedury obsługi zdarzeń JavaScript, 12.3.3]*

## 5.1.2. Użycie linii do podziału dokumentu

Linie poziome bardzo upraszczają nawigację w obrębie dokumentu. Żeby jednak ich obecność była uzasadniona, najpierw trzeba określić, ile poziomów nagłówków znajduje się w dokumencie oraz ile miejsca zostało przewidziane na każdą część. Dopiero wtedy można zdecydować, w których nagłówkach użycie linii poziomej będzie uzasadnione.

Liniami można również oddzielać informacje „organizacyjne" dokumentu — np. spis treści, indeks, bibliografię czy spis rysunków.

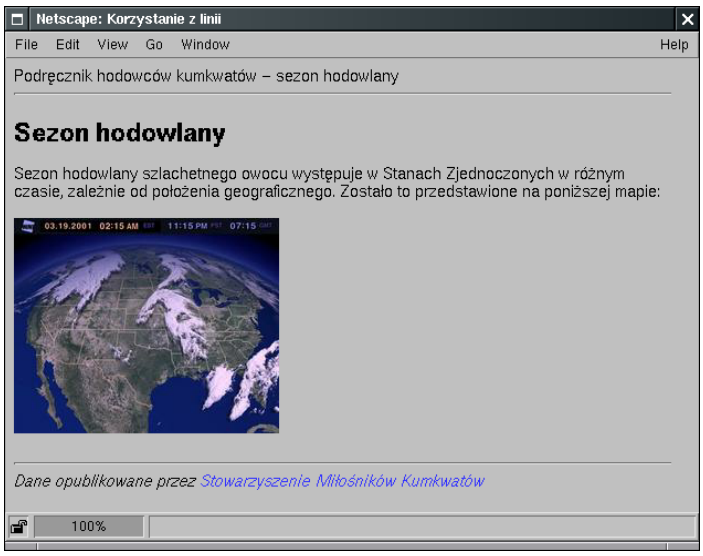
Doświadczeni autorzy wykorzystują również linie poziome do oznaczania początku i końca formularza. Jest to szczególnie przydatne w długich formularzach, wymagających przewijania okna przeglądarki. Konsekwentne oznaczanie początku i końca formularza upraszcza poruszanie się po nim i gwarantuje, że użytkownik nie przeoczy tej części, która znajduje się akurat poza „polem widzenia" przeglądarki.

## 5.1.3. Użycie linii do oddzielenia nagłówka i stopki

Przy tworzeniu całych zbiorów dokumentów bardzo istotne jest, aby zachować spójny i konsekwentny układ stron. Dotyczy to również obecności standardowego nagłówka i stopki na każdej z nich. Zazwyczaj w nagłówku znajdują się narzędzia nawigacyjne umożliwiające proste przejście do części wewnętrznych danego dokumentu oraz do innych dokumentów. W stopce znajdują się informacje o autorze i dokumencie oraz np. adres umożliwiający wysłanie komentarza do opiekuna strony.

Linie pozwalają odgraniczyć nagłówek i stopkę od reszty dokumentu. Spójrzmy na poniższy kod oraz sposób jego wyświetlenia (rysunek 5.7):

```
Podręcznik hodowców kumkwatów - sezon hodowlany
<hr>
<h1>Sezon hodowlany</h1>
Sezon hodowlany szlachetnego owocu występuje w Stanach
Zjednoczonych w różnym czasie, zależnie od położenia
geograficznego. Zostało to przedstawione na poniższej mapie:
<p>
<img src="mapa.jpg">
<p>
<hr>
<i>Dane opublikowane przez
<a href="komentarze.html">Stowarzyszenie Miłośników Kumkwatów</a></i>
```




*Rysunek 5.7. Odgraniczenie nagłówka i stopki za pomocą linii poziomych*

# 5.2. Wstawianie elementów graficznych

Jedną z najatrakcyjniejszych cech standardów HTML i XHTML jest możliwość urozmaicenia dokumentu tekstowego elementami graficznymi. Elementy takie mogą zostać albo wstawione bezpośrednio do dokumentu, albo opisane odsyłaczami i udostępnione do oddzielnego pobrania, mogą również stanowić tło dokumentu. Rozsądne korzystanie z grafiki — animowanych ikon, obrazków, ilustracji, rysunków itd., zwiększa atrakcyjność dokumentu, przyciąga uwagę czytelnika i przyczynia się do „profesjonalnego" wyglądu. Rysunek można również przygotować tak, że będzie stanowił mapę odsyłaczy. Jednak nadmiar grafiki może spowodować bałagan na stronie, utrudnić wyszukiwanie informacji oraz wydłużyć czas jej ładowania.

## 5.2.1. Formaty plików

Ani HTML, ani XHTML nie definiują „oficjalnego" formatu plików graficznych. Jednak popularne przeglądarki obsługują tylko pewną grupę takich formatów — przede wszystkim GIF i JPEG (ich opisy znajdują się poniżej). Przeglądanie większości innych formatów multimedialnych wymaga zastosowania specjalnego oprogramowania, instalowanego dodatkowo. Nic więc dziwnego, że GIF i JPEG to *de facto* standardowe pliki graficzne sieci WWW.

Oba te formaty były szeroko stosowane jeszcze zanim pojawiła się usługa WWW, a więc istnieje cała gama programów pozwalających na przygotowanie grafiki w tej postaci. Każdy z tych formatów ma swoje zalety i wady, posiadają one także pewne cechy wykorzystywane przez niektóre przeglądarki w sposób specjalny.

### 5.2.1.1. GIF

Graphics Interchange Format (GIF) to format pierwotnie opracowany w celu przesyłania obiektów graficznych w sieci CompuServe. Pewne cechy tego formatu zadecydowały o jego popularności wśród autorów stron HTML i XHTML. Algorytm kodowania GIF jest niezależny od platformy, a więc oprogramowanie dekodujące (wbudowane w większość przeglądarek) pozwala na wyświetlenie np. pliku stworzonego na macintoshu, czy w komputerze PC pod kontrolą systemu Windows. Druga istotna cecha to specjalna technologia kompresji, zapewniająca znaczące zmniejszenie rozmiaru pliku graficznego i tym samym szybsze przesyłanie w sieci. Kompresja GIF jest „bezstratna", tzn. dane oryginalnego obrazu nie są zmieniane czy „gubione"; po dekompresji obraz wygląda tak, jak pierwowzór. Do tego dochodzi jeszcze łatwość animacji obrazów GIF.

Pliki GIF mają niezmiennie rozszerzenie *.gif* (lub *.GIF*), ale tak naprawdę istnieją dwie wersje tego formatu: oryginalny GIF87 oraz rozszerzony GIF89a, bogatszy o nowe możliwości, często wykorzystywane przez autorów stron WWW: przezroczyste tła, zachowywanie z przeplotem, animacja (patrz podpunkt 5.2.1.2). Współczesne przeglądarki obsługują obie wersje formatu. W obu tych wersjach zastosowano taki sam typ kodowania: 8-bitowe wartości odpowiadające poszczególnym pikselom odwzorowywane są na paletę kolorów, zawierającą maksymalnie 256 kolorów dla jednego pliku. Większość obrazków GIF zbudowana jest z nawet mniejszej liczby kolorów, istnieją też narzędzia do zmniejszania palety nawet w bogatych w barwy grafikach. Im prostszy obrazek GIF, tym mniejsza wystarczy mu paleta kolorów i tym większa kompresja (a więc szybsze ładowanie).

Jednakże, właśnie z powodu ograniczonej liczby kolorów, format GIF nie nadaje się do wszystkich zastosowań. Szczególnie dotyczy to fotografii (patrz opis formatu JPEG, podpunkt 5.2.1.3). „GIF-y" świetnie spisują się natomiast w ikonkach, obrazkach o niewielkiej liczbie kolorów i rysunkach.

Ponieważ większość przeglądarek graficznych obsługuje format GIF bez żadnych dodatkowych zabiegów, obecnie jest to najpopularniejszy format graficzny sieci WWW. Elementy GIF można dołączać bezpośrednio do dokumentu, jak i odwoływać się do nich za pośrednictwem odsyłaczy. W razie wątpliwości odnośnie tego, jaki format graficzny wybrać, wybierzmy GIF. W większości sytuacji będzie to wybór trafny.

### 5.2.1.2. Przeplot, przezroczystość i animacja

Format GIF umożliwia uzyskanie trzech efektów specjalnych: przeplotu, przezroczystości oraz animacji. Dzięki przeplotowi grafika ładowana na stronie WWW nie jest wyświetlana po kawałku od góry w dół, ale sprawia wrażenie stopniowej „materializacji". Zazwyczaj obrazek zakodowany w formacie GIF to sekwencja danych o pikselach, pogrupowanych w rzędy, w kolejności od góry do dołu. Typowy element w formacie GIF jest wyświetlany od góry i przypomina zsuwanie podciągniętych żaluzji. Obrazek z przeplotem pojawia się od razu cały i jest stopniowo „uzupełniany" — przypomina obracanie już zsuniętych żaluzji. W „GIF-ie" z przeplotem wyświetlany jest najpierw co czwarty rząd pikseli. Najpierw grafika jest więc rozmyta (ale obraz widać czterokrotnie szybciej niż zwykle), a potem coraz ostrzejsza. Ten rozmyty obrazek pozwala już dobrze zorientować się odnośnie zawartości pliku graficznego i tym samym upraszcza użytkownikom „surfowanie".

Wszystkie przeglądarki graficzne potrafią wyświetlać pliki GIF z przeplotem, jednak nie we wszystkich udaje się osiągnąć efekt „materializacji". A nawet w tych, które to potrafią, użytkownik może zawsze włączyć opcję mówiącą przeglądarce, że ma odczekać aż do pełnego pobrania obrazka i dopiero wtedy go wyświetlić. Starsze przeglądarki zawsze pobierają i dekodują takie elementy przed wyświetleniem, zupełnie ignorując „materializację".

Kolejny popularny efekt specjalny formatu GIF89a to przezroczystość. W takich obrazkach część grafiki jest „niewidzialna" i prześwituje przez nią to, co znajduje się pod spodem (zazwyczaj tło strony). W takich plikach GIF jeden z kolorów w palecie został określony jako kolor tła. Przeglądarka po prostu ignoruje ten kolor i z tego wynika efekt „prześwitywania". Staranne dobranie rozmiarów obrazka oraz użycie jednolitego tła umożliwia uzyskanie efektu „wtopienia się" lub „unoszenia" grafiki nad stroną.

Przezroczyste elementy GIF świetnie nadają się wszędzie tam, gdzie grafika ma zostać wpleciona w dokument i gdzie zależy nam na uniknięciu prostokątnego kształtu obrazka. Popularne są na przykład przezroczyste emblematy firm, ikony i ozdobniki — czyli wszystko to, co ma zostać wyświetlone w naturalnym kształcie. Przezroczysty GIF może również zostać wpleciony w sam tekst i pełnić rolę jakiegoś symbolu, niemożliwego do uzyskania w standardowy sposób.

Kłopot z przezroczystością polega na tym, że taki obrazek GIF będzie wyglądał mało zachęcająco, jeśli nie usuniemy jego otoczki (ramki) będącej rezultatem wstawienia tego elementu w odsyłacz (znacznik `<a>`) albo po prostu stanowiącej część stylu. Wszystko to, co „opływa" obrazek, przylega do jego kształtu faktycznego — czyli prostokątnego, a nie tylko tego „widzialnego". To potrafi bardzo zepsuć wygląd strony.

Trzeci trik wynikający ze stosowania formatu GIF89a, to możliwość tworzenia animacji. Za pomocą specjalnych narzędzi tworzy się jeden plik składający się z wielu „klatek" GIF. Przeglądarka wyświetla kolejne klatki takiego elementu, podobnie jak klatki filmu rysunkowego. Dzięki istnieniu

specjalnych segmentów sterujących pomiędzy poszczególnymi częściami składowymi obrazka możliwe jest określenie liczby powtórzeń całego „filmu", ustalenie odstępów pomiędzy poszczególnymi klatkami, określenie, czy poprzednia klatka na czas wyświetlania następnej ma zostać przeniesiona do tła, itp. Połączenie tych wszystkich funkcji sterujących z innymi cechami „GIF-ów" (indywidualne palety kolorów, przezroczystość i przeplot) umożliwia tworzenie naprawdę pięknych i wyszukanych animacji[3].

Proste animacje z użyciem plików GIF są ciekawym rozwiązaniem z jeszcze jednego powodu: aby je wstawić do dokumentu, nie musimy wykonywać żadnych dodatkowych zabiegów. Ale wszystko ma swoją cenę: pliki GIF z animacją (poza tymi małymi — ikonkami i symbolami) bywają bardzo duże objętościowo, nawet jeśli przy ich tworzeniu zabiegaliśmy o to, aby w animacji nie były powtarzane miejsca statyczne obrazu. A jeśli w jednym dokumencie znajduje się wiele animacji, to jego ładowanie może się bardzo wydłużyć. Tak więc, przy projektowaniu strony trzeba bacznie uważać, aby nie przesadzić z animacjami.

Wszystkie te triki z plikami GIF — przeplot, przezroczystość i animacja, nie dzieją się tak same z siebie. Do przygotowania takiego pliku wymagane jest specjalne oprogramowanie. Wiele narzędzi graficznych pozwala zachować wykonaną pracę w formacie GIF i określić miejsca przezroczyste. Są również programy shareware i freeware wyspecjalizowane w tworzeniu przezroczystych i animowanych elementów graficznych GIF — wystarczy poszukać ich w archiwach oprogramowania w Internecie. Dodatkowe informacje o tworzeniu elementów przezroczystych znajdują się w rozdziale 17., *„Kruczki i sztuczki"*.

### 5.2.1.3. JPEG

Format kodowania obrazu JPEG powstał w wyniku prac grupy Joint Photographic Experts Group. Podobnie jak pliki GIF, obrazy JPEG są niezależne od platformy i skompresowane, co ułatwia transport w sieciach cyfrowych. W przeciwieństwie do formatu GIF, obrazy JPEG mogą się składać z dziesiątków tysięcy kolorów, a więc format ten lepiej nadaje się do prezentacji wysublimowanych, fotorealistycznych elementów graficznych. W formacie JPEG zastosowano specjalne algorytmy pozwalające na uzyskanie o wiele większego stopnia kompresji. Całkiem powszechne są sytuacje, w których 200-kilobajtowy plik GIF można przekonwertować do postaci 30-kilobajtowego „JPEG-a". Tak wielki stopień kompresji wynika z faktu, że JPEG jest formatem „stratnym". Jednak stopień „stratności" można regulować za pomocą specjalnych, przeznaczonych do tego celu narzędzi — a więc, choć zdekompresowany obraz może nieco różnić się od oryginału, to ta różnica będzie na tyle niewielka, że większość osób jej po prostu nie zauważy.

JPEG świetnie nadaje się do prezentacji fotografii, ale gorzej do zwykłych ilustracji, czy rysunków. Zastosowane algorytmy kompresji i dekompresji powodują pozostawianie zauważalnych „otoczek" przy dużych obszarach jednolitego koloru. Jeśli więc trzeba przedstawić rysunek, GIF będzie się nadawał do tego lepiej.

Pliki formatu JPEG (rozszerzenie *.jpg* lub *.JPG*) są rozumiane przez niemal wszystkie współczesne przeglądarki graficzne. Rzadko już spotyka się te starsze, nie obsługujące tego formatu.

[3] Wydawnictwo Songline Studios opublikowało całą książkę o animacji w formacie GIF: *GIF Animation Studio* Richarda Komana.

## 5.2.2. Kiedy wstawiać elementy graficzne?

Dobry obrazek jest więcej wart niż tysiąc słów. Trzeba jednak uważać, żeby tymi „tysiącami słów" nie przegadać odbiorcy. Przede wszystkim należy pamiętać, że elementy graficzne na stronie są narzędziem wizualnym, a nie przynętą na czytelnika. Mają wspierać zawartość tekstową i pomagać w nawigacji. Mają wyjaśniać, ilustrować lub służyć jako przykład. Fotografie wzbogacające treść, wykresy, diagramy, mapy i rysunki — to właśnie doskonali kandydaci do umieszczenia na stronie WWW. Na przykład, w katalogach sklepowych zdjęcia produktów są wręcz nieodzowne. Natomiast ikony lub symbole-odsyłacze (także animowane) bardzo wspomagają nawigację po wewnętrznych i zewnętrznych zasobach strony. Jeśli element graficzny nie pasuje do żadnej z powyższych kategorii, warto zastanowić się, czy w ogóle jest potrzebny!

Jednym z najważniejszych problemów związanych z obecnością grafiki w dokumencie jest wydłużony czas ładowania. Problem ten doskwiera szczególnie osobom korzystającym z modemów. Typowy dokument tekstowy ma najwyżej 10 – 15 tysięcy bajtów; obrazki mogą tę objętość zwiększyć o setki i tysiące bajtów każdy. A całkowity czas pobierania dokumentu nie zależy wyłącznie od sumy objętości wszystkich komponentów, ale także od opóźnień przy ich pobieraniu.

W zależności od szybkości połączenia — tzw. przepustowości (ang. *bandwidth*) zazwyczaj wyrażanej w bitach lub bajtach na sekundę oraz aktualnego ruchu w sieci, jeden dokument zawierający 100-kilobajtowy obrazek może się ładować od 15 sekund (modem 57,6 Kb/s wczesnym rankiem) aż do ponad *dziesięciu minut* (modem 9600 b/s o północy). Tak to właśnie wygląda.

Ale, oczywiście, powszechność grafiki i innych obiektów multimedialnych skłania usługodawców internetowych do oferowania szybszych i wydajniejszych połączeń. Wkrótce modemy 57,6 Kb/s odejdą w niepamięć (tak jak odeszły te o prędkości 9600 b/s) na korzyść modemów kablowych i technologii ADSL. Wkrótce większość użytkowników będzie łączyła się z prędkościami niedawno dostępnymi tylko dla najbogatszych — ponad megabit na sekundę.

Ale w miarę obniżania cen dostępu do Internetu, wzrasta także liczba użytkowników i tym samym ruch w sieci. Jeśli staramy się uzyskać dostęp do przeciążonego serwera, prędkość naszego połączenia nie ma w ogóle praktycznego znaczenia.

## 5.2.3. Kiedy używać tekstu?

Tekst nie przestał być modny. W przypadku niektórych użytkowników jest to jedyna zawartość strony, do jakiej mają dostęp. W większości wypadków powinno się tak tworzyć dokumenty, aby mogli z nich skorzystać także ci, którzy nie mogą zobaczyć elementów graficznych lub w swoich przeglądarkach wyłączyli automatyczne ładowanie takich elementów (np. w celu przyspieszenia pobierania stron). Pokusa wzbogacania wszystkich dokumentów obrazkami może być silna, trzeba jednak pamiętać, że w niektórych sytuacjach o wiele sensowniejszy będzie po prostu zwykły tekst.

Dokumenty konwertowane do postaci HTML z innych formatów rzadko zawierają grafikę. Materiały referencyjne i inna „poważna" zawartość często wystarczająco dobrze jest reprezentowana jako „czysty" tekst.

Jeśli bardzo zależy nam na szybkim ładowaniu strony, warto poprzestać na zawartości tekstowej. Kiedy wiadomo, że stronę będzie pobierać wiele osób, nie można przeładowywać dokumentu grafiką — czytelnikom trudniej będzie „dostać się" do takiej strony. W ekstremalnych przypadkach

można wstawić stronę wprowadzającą, na której czytelnik będzie mógł wybrać wersję naszej strony zawierającą obrazki lub nie. Popularne przeglądarki wstawiają specjalne ikonki w miejscach, gdzie powinny zostać załadowane elementy graficzne — mogą one spowodować bałagan i spadek czytelności dokumentu.

Jeżeli dokument ma zostać poprawnie zindeksowany przez wyszukiwarki WWW, powinien zawierać większość tekstu i tylko uzasadnione elementy graficzne — bez niepotrzebnych ozdóbek. Wyszukiwarki takie niemal zawsze ignorują pliki graficzne. Jeśli większość strony zawiera grafikę, wyszukiwarki internetowe być może nie będą potrafiły uzyskać z takiego dokumentu żadnych sensownych informacji.

## 5.2.4. Przyspieszanie pobierania grafiki

Istnieje kilka sposobów przyspieszenia ładowania elementów graficznych (oczywiście, poza samą powściągliwością przy wstawianiu ich do dokumentu):

*Uprościć grafikę*

Pełnoekranowa grafika w 24-bitowym kolorze, nawet skompresowana do postaci pliku GIF lub JPEG, i tak spowoduje zapchanie połączenia. Warto zdobyć narzędzia do optymalizacji rozmiarów obrazka i liczby kolorów i korzystać z nich. Należy w miarę możliwości upraszczać elementy graficzne. Unikać panoramicznych fotografii i dużych obszarów pustych, a także pokaźnych obramowań i innych zajmujących wiele miejsca komponentów. Trzeba również z rezerwą stosować dithering (łączenie przylegających punktów w różnych kolorach w celu uzyskania trzeciego); technika ta zmniejsza możliwości kompresji. Nie stosować dużych obszarów o jednolitym kolorze — kiepsko się kompresują zarówno w formacie GIF, jak i JPEG.

*Używać wielokrotnie tych samych elementów*

Szczególnie dotyczy to ikon i animowanych „GIF-ów". Większość przeglądarek przechowuje raz pobrane elementy strony w pamięci podręcznej (ang. *cache*), dzięki czemu ich kolejne ładowanie nie wymaga łączenia z siecią i odbywa się bardzo szybko. W animacjach GIF kolejne „klatki" należy budować poprzez zmianę tylko fragmentu poprzednich, a nie przerysowywać cały obrazek (to również przyspiesza samą animację).

*Dzielić duże dokumenty na części*

Ta ogólna zasada dotyczy również elementów graficznych. Dokumenty rozdzielone na wiele małych segmentów i połączone za pomocą odsyłaczy oraz spisów treści są lepiej przyjmowane przez czytelników niż strony bardzo duże. Odbiorca woli zazwyczaj „przerzucić" kilka stron niż czekać na załadowanie jednej, ale długiej (to jest jak przełączanie kanałów w pilocie — syndrom choroby telewizyjnej). Często przytacza się praktyczną zasadę, mówiącą, że jeden dokument nie powinien przekroczyć objętości 50 kilobajtów — wtedy odbiorca korzystający nawet z wolnego połączenia nie zniechęci się długim czasem ładowania.

*Oddzielać duże elementy graficzne*

Duże elementy graficzne warto oddzielić od właściwego dokumentu i zastąpić je odsyłaczem do pliku (np. w postaci miniaturki obrazka). Wtedy czytelnik sam zadecyduje, czy pobrać taki element graficzny. A ponieważ pobrana w ten sposób grafika nie jest „wymieszana" z innymi elementami strony (np. obrazkami wplecionymi w tekst), łatwiej jest ją zachować i przejrzeć w późniejszym czasie (więcej o pobieraniu takich elementów graficznych w punkcie 5.6.2).

*Podawać rozmiary obrazka*

Jeszcze inny sposób zwiększenia wydajności to określenie wysokości i szerokości obrazka w jego znaczniku. W ten sposób eliminuje się dodatkowe czynności, jakie musi wykonać przeglądarka w celu zaplanowania rozmieszczenia elementów strony; takie postępowanie ma jednak również wady — o nich w podpunkcie 5.2.6.12.

## 5.2.5. JPEG czy GIF?

Jeśli posiadamy już gotowe obrazki w jednym tylko formacie, lub korzystamy z narzędzia zapisującego tylko GIF lub tylko JPEG, możemy wzbogacać stronę elementami graficznymi tylko tego jednego typu. Po stronie czytelnika nie powinno być natomiast żadnych problemów z odczytaniem i jednego, i drugiego formatu.

Tym niemniej zaleca się zdobycie narzędzi pozwalających na zapis lub konwersję do obu tych formatów; każdy z nich ma charakterystyczne cechy. Na przykład, w ikonach i symbolach przydaje się możliwość uzyskania przezroczystości (GIF), a w dużych i kolorowych zdjęciach specjalna kompresja stratna (JPEG).

## 5.2.6. Znacznik <img>

Znacznik `<img>` umożliwia wstawienie elementu graficznego w bieżącym miejscu dokumentu. Ani przed, ani po tym znaczniku nie jest wstawiane domyślnie przełamanie wiersza, tak więc domyślnie wszystkie obrazki zostają „zagnieżdżone" w tekście, czy innej zawartości.

Format samego pliku graficznego nie jest zdefiniowany w standardach HTML i XHTML. Popularne przeglądarki obsługują dwa formaty: GIF i JPEG. Standardy nie opisują ani nie ograniczają również rozmiarów obrazka. Grafika może mieć dowolną liczbę kolorów, ale sposób ich wyświetlania w dużym stopniu zależy od przeglądarki.

Prezentacja grafiki w ogóle bardzo zależy od „browsera". Przeglądarki tekstowe mogą ignorować grafikę, a te pracujące w ograniczonym środowisku mogą zaś przedstawiać ją w zmodyfikowany sposób. Część użytkowników, szczególnie tych korzystających z powolnych połączeń, może w ogóle wyłączyć pobieranie obrazków. Trzeba więc zaplanować stronę tak, aby miała ona dla czytelnika sens nawet bez żadnych elementów graficznych.

### 5.2.6.1. Atrybut src

Atrybut `src` jest w znaczniku `<img>` wymagany (chyba że zastosowano atrybut `dynsrc` obsługiwany przez Internet Explorer i wykorzystywany do prezentacji filmów). Wartość atrybutu `src` to adres URL pliku, absolutny albo względny w stosunku do bieżącego dokumentu. Aby nie zaśmiecać katalogu strony, autorzy zazwyczaj przenoszą wszystkie pliki graficzne do oddzielnego folderu o odpowiedniej nazwie, np. „obrazki" czy „pics". *[adresy URL, 6.2]*

## <*img*>

***Funkcja:***

Powoduje wstawienie do dokumentu elementu graficznego

***Atrybuty:***

| | |
|---|---|
| ALIGN | ONDBLCLICK |
| ALT | ONERROR |
| BORDER | ONKEYDOWN |
| CLASS | ONKEYPRESS |
| CONTROLS O | ONKEYUP |
| DIR | ONLOAD |
| DYNSRC O | ONMOUSEDOWN |
| HEIGHT | ONMOUSEMOVE |
| HSPACE | ONMOUSEOUT |
| ID | ONMOUSEOVER |
| ISMAP | ONMOUSEUP |
| LANG | SRC |
| LONGDESC | START O |
| LOOP O | STYLE |
| LOWSRC N | TITLE |
| NAME N | USEMAP |
| ONABORT | VSPACE |
| ONCLICK | WIDTH |

***Znacznik zamykający:***

w HTML-u brak, w XHTML-u `</img>` lub `<img ... />`

***Zawiera:***

Nic

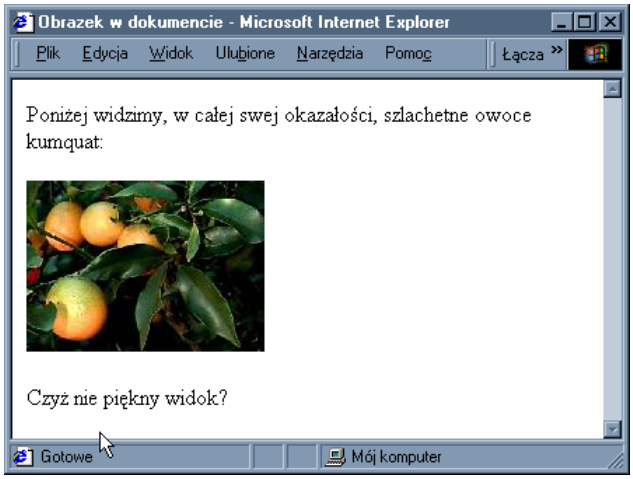
***Stosowany wewnątrz:***

text

Na przykład, poniższy fragment kodu HTML powoduje umieszczenie w dokumencie zdjęcia kumkwatów (rysunek 5.8):

```
Poniżej widzimy, w całej swej okazałości, szlachetne owoce kumquat:
<p>
<img src="kumquat.jpg">
<p>
Czyż nie piękny widok?
```

W przykładzie tym wokół znacznika `<img>` wstawiono znaczniki akapitów, dzięki czemu przed obrazkiem i po nim przeglądarka dodała nieco pustego miejsca. Jak opisano to w podpunkcie 5.2.6.4, tekst może także „przystawać" z boku obrazka.




*Rysunek 5.8. Obrazek w dokumencie*

### 5.2.6.2. Atrybut lowsrc

Przeglądarka Netscape umożliwia uzupełnienie atrybutu `src` atrybutem `lowsrc`, za którego pomocą można przyspieszyć ładowanie dokumentu. Wartością `lowsrc`, podobnie jak `src`, jest adres URL pliku graficznego ładowanego przez przeglądarkę po napotkaniu znacznika `<img>`. Plik ten jest ładowany natychmiast; dopiero po załadowaniu całej strony, gdy może ona już być odczytana przez użytkownika, ładowany jest plik podany jako wartość atrybutu `src`.

Obrazek określany atrybutem `lowsrc` ma niską rozdzielczość i jest „uboższą" wersją pliku właściwego. Użytkownik poznaje zawartość obrazka nie czekając długo na pobranie. Atrybut ten może jednak służyć również do uzyskiwania efektów specjalnych.

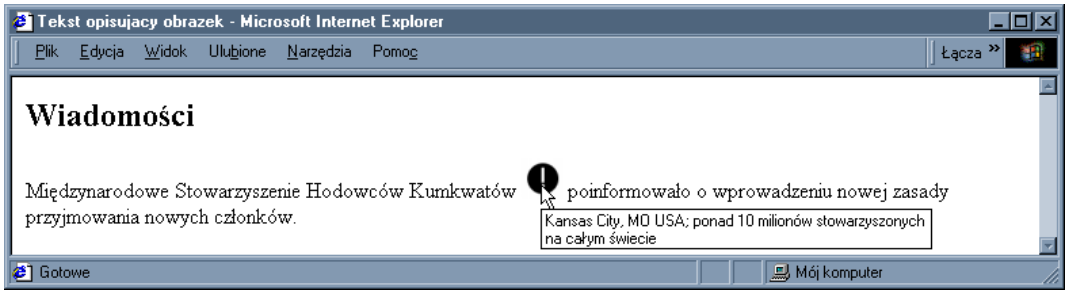
Przeglądarka Netscape rezerwuje pewien obszar dokumentu na element graficzny, zgodnie z rozmiarami obrazka podanego jako wartość `lowsrc`, chyba że wielkości te (szerokość i wysokość) podane zostaną jawnie za pomocą atrybutów `height` i `width`. Jeśli więc rozmiary obrazka określonego w atrybucie `src` są różne od tego w `lowsrc`, albo jeśli jawnie podano atrybuty wysokości i szerokości, obrazek podany jako `src` zostanie wyświetlony jako zmniejszony, powiększony, rozciągnięty lub ściśnięty — tak, aby pasował do przydzielonego miejsca. Co więcej, obrazki określone w atrybutach `lowsrc` i `src` nie muszą przedstawiać tego samego, a więc opóźnione wyświetlanie można zamienić w prostą animację.

Atrybut `lowsrc` jest obsługiwany wyłącznie przez Netscape. Inne przeglądarki ignorują go i ładują tylko ten element, który określono jako wartość atrybutu `src`. Jeśli użytkownik wyłączy pobieranie obrazków w Netscape, przeglądarka ta nie załaduje żadnej z tych dwóch wersji pliku. W takim przypadku obrazki te zostaną pobrane w odpowiedniej kolejności dopiero wtedy, gdy użytkownik kliknie na ikonie wstawionej w miejsce grafiki. Żadna przeglądarka nie ładuje tylko elementu opisanego jako `lowsrc`; zawarcie atrybutu `src` jest więc konieczne — inaczej w oknie przeglądarki nie pojawi się żaden obrazek poza ikonką-„wypełniaczem".

### 5.2.6.3. Atrybuty alt i longdesc

Atrybut `alt` umożliwia określenie alternatywnego tekstu, jaki zostanie wyświetlony przez przeglądarkę nie obsługującą grafiki lub, w której użytkownik wyłączył pobieranie obrazków. Znacznik ten jest opcjonalny, ale naprawdę warto korzystać z niego w większości przypadków. Jeśli element graficzny nie będzie dostępny, użytkownik zobaczy chociaż co *miało* się tam znaleźć.

Ponadto, najnowsze przeglądarki wyświetlają tekst podany jako wartość argumentu `alt` w ramce tekstowej, gdy użytkownik umieści wskaźnik myszy nad obrazkiem. Można więc umieścić tam informacje dodatkowe, wyświetlane np. po najechaniu myszą na małą ikonkę (rysunek 5.9).




*Rysunek 5.9. Współczesne przeglądarki wyświetlają zawartość atrybutu alt w okienku tekstowym*

Wartość atrybutu `alt` to łańcuch tekstowy o długości do 1024 znaków, w tym spacje i znaki przestankowe. Zawsze wstawiany jest w cudzysłowach. Może zawierać encje, ale nie znaczniki; nie jest możliwe sterowanie stylem takiego tekstu.

Jeśli obrazek jest dostępny, a użytkownik włączył pobieranie grafiki, przeglądarki graficzne zazwyczaj nie wyświetlają wartości atrybutu `alt`. W przeciwnym razie wartość ta pokazywana jest obok ikonki wstawianej w miejscu obrazka. Dobrze dobrane etykiety `alt` mogą więc bardzo pomóc użytkownikom nie pobierającym grafiki z powodu posiadania powolnego łącza.

Przeglądarki tekstowe, takie jak Lynx, umieszczają zawartość atrybutu `alt` bezpośrednio w tekście, tak jak każdy inny element zawartości dokumentu. Odpowiedni tekst takiego atrybutu może więc z powodzeniem zastąpić grafikę (użytkownicy przeglądarek tekstowych bardzo to docenią — nie lubią, kiedy na każdym kroku udowadnia się im, że są internautami drugiej kategorii). Na przykład, w poniższym przykładzie użytkownik przeglądarki graficznej zobaczy kulkę służącą do wypunktowania, a użytkownik przeglądarki tekstowej — gwiazdkę:

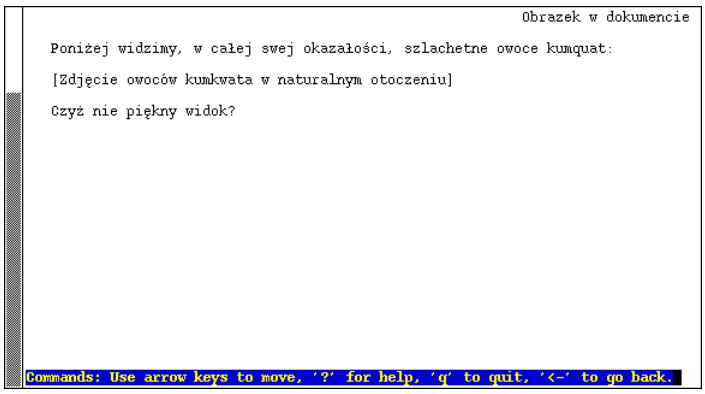
```
<h3>img src="obrazki/kulka.gif" alt="*">Wprowadzenie</h3>
```

Natomiast w poniższym przykładzie tekst zastępuje symboliczną ikonkę:

```
<ul>
   <li> Przepisy na przyrządzanie kumkwatów
        <img src="obrazki/nowosc.gif" alt="(Nowość!)">
   <li> Okresy zbiorów
</ul>
```

W przeglądarce tekstowej zamiast ikonki *nowosc.gif* pojawi się napis „(Nowość!)”. W atrybucie `alt` może się znaleźć nawet dłuższy tekst (patrz rysunek 5.10):

```
Poniżej widzimy, w całej swej okazałości, szlachetne owoce kumquat:
<p>
<img src="kumquat.jpg" alt="[Zdjęcie owoców kumkwata w naturalnym otoczeniu]">
<p>
Czyż nie piękny widok?
```




*Rysunek 5.10. Przeglądarki tekstowe zamiast obrazka wyświetlają tekst zawarty w atrybucie alt*

Atrybut `longdesc` jest podobny do atrybutu `alt`, ale umożliwia stosowanie dłuższych opisów. Wartością tego atrybutu jest adres URL dokumentu zawierającego opis obrazka. Jeśli chcemy umieścić opis dłuższy niż 1024 znaki, to robimy to właśnie za pomocą `longdesc`. Ani HTML 4, ani XHTML nie wyszczególniają, jaka może być zawartość takiego opisu. Obecnie żadna przeglądarka nie obsługuje także tego atrybutu, trudno więc przytoczyć jakąkolwiek praktyczną radę na jego temat.

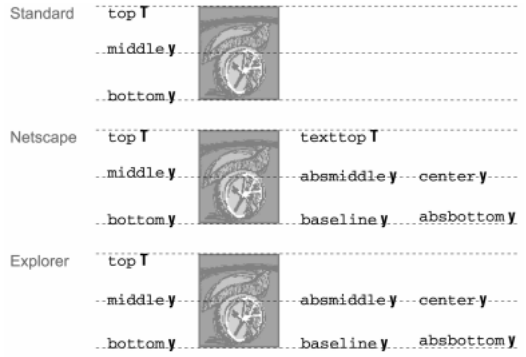
### 5.2.6.4. Atrybut align

Standardy nie definiują domyślnego wyrównania elementów graficznych względem pozostałego tekstu oraz innych obrazków w tym samym wierszu — nigdy nie można dokładnie przewidzieć, jak elementy te zostaną rozłożone na stronie[4]. Obrazki zazwyczaj wstawiane są wraz z tekstem w jednej linijce. Z innych mediów, np. z gazet, znamy natomiast odmienny sposób rozkładania elementów: tekst „opływa" ilustrację i poszczególne wiersze przystają do jej boku.

Na szczęście twórcy dokumentów HTML są w stanie uzyskać pewną kontrolę nad wyrównaniem ilustracji względem otaczającego tekstu. Służy do tego atrybut `align` znacznika `<img>`. Standardy HTML i XHTML definiują pięć wartości tego atrybutu: `left`, `right`, `top`, `middle` i `bottom`. Pierwsze dwie wartości powodują, że następujący po obrazku tekst „opływa" obrazek; sam obrazek jest zaś przesuwany do, odpowiednio, lewego lub prawego marginesu. Pozostałe trzy wartości sterują wyrównaniem pionowym względem otaczającego tekstu. Netscape obsługuje jeszcze cztery inne wartości związane z wyrównaniem pionowym: `texttop`, `absmiddle`, `baseline` oraz `absbottom`, Internet Explorer obsługuje wartość `center`.

Poniżej przedstawiono opisy wartości mówiących o wyrównaniu zagnieżdżonego w tekście obrazka; przykłady przedstawiono na rysunku 5.11.

[4] Większość popularnych przeglądarek wstawia element graficzny tak, że spód obrazka zrównany jest z podstawą linijki tekstu — czyli tak, jakby podano wartość wyrównania `bottom`. Nie można jednak być całkowicie pewnym takiego zachowania i zawsze należy jednoznacznie określać wyrównanie elementów graficznych.




| Alignment | Standard | Netscape | Explorer |
|---|---|---|---|
| top | ● | ● | ● |
| texttop | | ● | |
| middle | ● | ● | ● |
| absmiddle | | ● | ● |
| center | | ● | ● |
| bottom | ● | ● | ● |
| baseline | | ● | ● |
| absbottom | | ● | ● |

*Rysunek 5.11. Standardowe i niestandardowe atrybuty opisujące wyrównanie obrazka względem tekstu*

`top`

Szczyt obrazka jest wyrównywany ze szczytem najwyższego elementu w bieżącej linijce tekstu. Jeśli w bieżącym wierszu nie ma innych elementów graficznych, obrazek jest wyrównywany do górnej granicy tekstu.

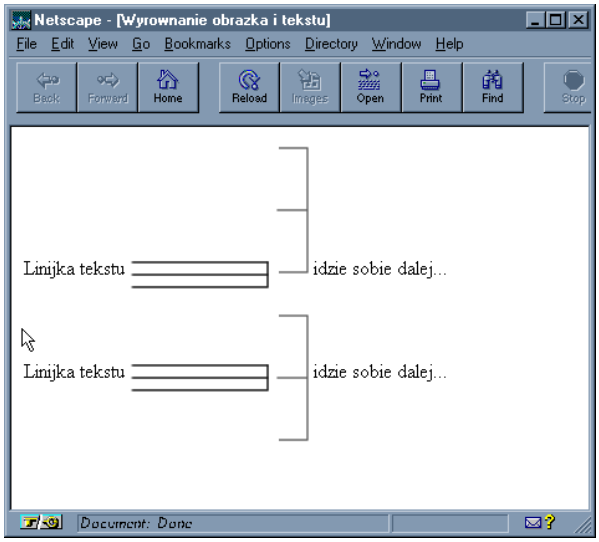
`texttop`

Atrybut `align=texttop` powoduje, że w przeglądarce Netscape szczyt obrazka jest zrównywany ze szczytem najwyższego elementu tekstowego w bieżącej linijce. Różni się od wartości `top` tym, że ta ostatnia powoduje wyrównanie do najwyższego elementu bez względu na to, czy jest to tekst czy element graficzny. Jeśli linijka nie zawiera innych elementów graficznych „wystających" ponad tekst, `texttop` i `top` działają identycznie.

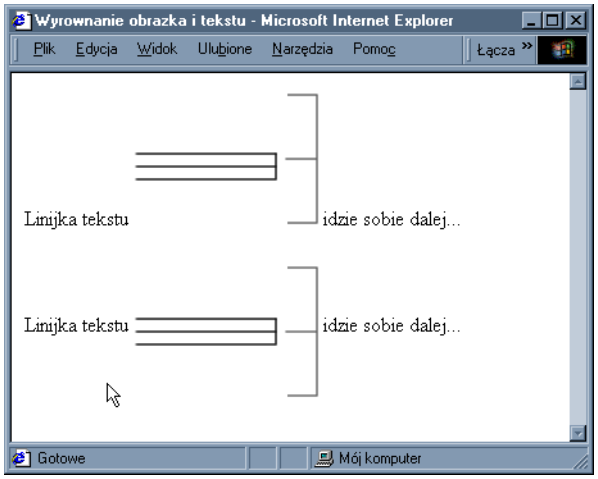
`middle`

Netscape i Internet Explorer traktują wartość `middle` na różne sposoby. Netscape zrównuje środek wysokości obrazka z podstawą linijki tekstu, bez względu na inne elementy zagnieżdżone (np. inny obrazek — rysunek 5.12). Internet Explorer zrównuje środek wysokości obrazka ze środkiem najwyższego elementu w bieżącej linijce — tekstowego czy też graficznego (rysunek 5.13). Spójrzmy na różnice w sposobie wyrównania na rysunkach 5.12 i 5.13, widoczne szczególnie wtedy, gdy tylko jeden obrazek jest opisany atrybutem `align`. Na obu rysunkach przedstawiono sposób interpretacji poniższego fragmentu kodu:

```
Linijka tekstu
<img src="poziomy.gif" align=middle>
<img src="pionowy.gif">
idzie sobie dalej...
<br clear=left>
<p>
Linijka tekstu
<img src="poziomy.gif" align=middle>
<img src="pionowy.gif" align=middle>
idzie sobie dalej...
```




*Rysunek 5.12. Netscape wyrównuje środek wysokości obrazka ze spodem tekstu*




*Rysunek 5.13. Internet Explorer wyrównuje środek wysokości obrazka ze środkiem najwyższego elementu w wierszu*

Należy również zauważyć, że przeglądarka Internet Explorer w wersji 3 oraz późniejszych traktuje `middle`, `absmiddle` i `center` identycznie. Wcześniejsze wersje oraz Netscape rozróżniają pomiędzy wyrównaniem `middle` a `absmiddle` (osoby, którym wszystkie te wartości zaczynają się właśnie mylić, proszone są o podniesienie ręki).

`absmiddle`

Jeśli atrybut `align` przyjmie wartość `absmiddle`, przeglądarka dopasuje bezwzględny środek (ang. *absolute middle*) wysokości obrazka do bezwzględnego środka wysokości bieżącej linijki. W Netscape oraz wczesnych wersjach Internet Explorera wartość ta działa inaczej niż `middle` — ta ostatnia zrównuje środek wysokości obrazka ze spodem bieżącej linijki tekstu (podstawą znaków). Przeglądarka Internet Explorer w wersjach 3 i późniejszych traktuje `absmiddle` dokładnie tak jak `middle` i `center`.

`center`

Wartość `center` jest traktowana przez Internet Explorer i przez Netscape dokładnie tak jak `absmiddle`, pamiętajmy jednak, że obie te przeglądarki inaczej traktują `absmiddle`.

`bottom` i `baseline` (ustawienie domyślne)

W przypadku Netscape oraz wczesnych wersji Internet Explorera wartości `bottom` i `baseline` miały takie samo działanie: tak jakbyśmy nie wstawili w ogóle atrybutu `align` — spód obrazka był na tej samej wysokości, co podstawa linijki tekstu. Nie jest to jednak to samo, co `absbottom` — tutaj „spód tekstu" to miejsce, do którego sięgają „ogonki" liter, np. małej litery „y". Internet Explorer w wersji 3 i późniejszych traktuje natomiast `bottom` tak jak `absbottom`.

`absbottom`

Atrybut `align=absbottom` informuje przeglądarkę, że spód obrazka ma zostać zrównany z faktycznym spodem bieżącej linijki tekstu. „Faktyczny spód" to najniższe miejsce w tekście, przy czym brane są pod uwagę „ogonki" liter takich jak „y" (nawet, jeśli nie występują w danej linijce). *Podstawa* linijki tekstu przystaje natomiast do dołu części „v" w znaku „y".

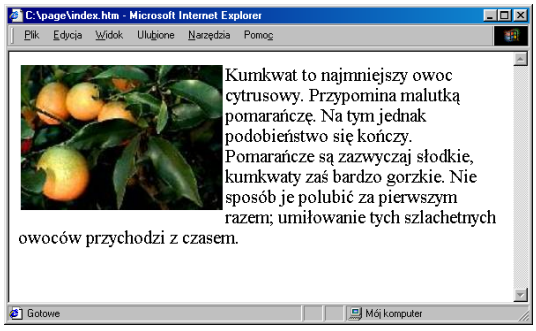
Przy wstawianiu ikon, symboli lub innych specjalnych znaków „scalanych" z otaczającym tekstem najlepiej używać wartości `top` lub `middle`. W innych przypadkach najlepsze efekty daje zazwyczaj zapis `align=bottom` (czyli ustawienie domyślne). Przy wyrównywaniu jednego lub więcej obrazków w jednym wierszu, najlepiej poeksperymentować z wyrównaniami i wybrać to prezentujące się najkorzystniej.

### 5.2.6.5. Tekst „opływający" ilustrację

Określenie wartości atrybutu `align` jako `left` lub `right` powoduje, że przeglądarka umieszcza obrazek odpowiednio przy lewym lub prawym marginesie. Następnie dalsza część zawartości dokumentu jest wyświetlana obok obrazka. Efekt jest taki, że treść znajdująca się za obrazkiem „opływa" go.

```
<img src="kumquat.jpg" align=left>
Kumkwat to najmniejszy owoc cytrusowy. Przypomina malutką pomarańczę.
Na tym jednak podobieństwo się kończy. Pomarańcze są zazwyczaj słodkie,
kumkwaty zaś bardzo gorzkie. Nie sposób je polubić za pierwszym razem;
umiłowanie tych szlachetnych owoców przychodzi z czasem.
```

Na rysunku 5.14 przedstawiono sposób interpretacji powyższego fragmentu w HTML-u.




*Rysunek 5.14. Tekst opływający ilustrację wyrównaną do lewej strony*

Ilustracje można umieszczać jednocześnie przy obu marginesach (rysunek 5.15); tekst zostanie wtedy wstawiony pomiędzy obrazkami:

```
<img src="kumquat.jpg" align=left>
<img src="kumquattree_maly.jpg" align=right>
Kumkwat to najmniejszy owoc cytrusowy. Przypomina malutką pomarańczę.
Na tym jednak podobieństwo się kończy. Pomarańcze są zazwyczaj słodkie,
kumkwaty zaś bardzo gorzkie. Nie sposób je polubić za pierwszym razem;
umiłowanie tych szlachetnych owoców przychodzi z czasem.
```


*Rysunek 5.15. Tekst pomiędzy wyrównanymi do marginesów rysunkami*

Na całej wysokości obrazka rolę marginesu dla zawartości pełni nie granica okna przeglądarki, ale właśnie krawędź obrazka. Kolejne ilustracje wyrównane w ten sam sposób będą przylegały jedna do drugiej. Oto przykładowy kod wywołujący taki efekt (patrz rysunek 5.16):

```
<img src="obrazki/linux.gif" align=left>
Penguin!
<br>
<img src="obrazki/livre.gif" align=left>
Pingwin!
<br>
<img src="obrazki/linlogo.gif" align=left>
Tux!
```


*Rysunek 5.16. Trzy piecieszne pingwiny*

Kiedy tekst „przelewa" się poza dolną krawędź obrazka, margines powraca do dawnej pozycji — zazwyczaj do krawędzi okna przeglądarki.

### 5.2.6.6. Wyśrodkowanie obrazka

Czy Czytelnik zauważył, że za pomocą atrybutu `align` nie jest możliwe wyśrodkowanie elementu graficznego? Wartości `middle` oraz `absmiddle` wyśrodkowują obrazek pionowo względem bieżącej linijki, poziome wyrównanie zależy od wcześniejszej zawartości oraz rozmiarów okna przeglądarki.

Poziome wyśrodkowanie obrazka zagnieżdżonego w zawartości strony jest możliwe, ale tylko jeśli obrazek ten jest odizolowany od otaczającej treści — np. za pomocą znacznika akapitu, działu lub przełamania wiersza. Wtedy wystarczy użyć znacznika `<center>` lub atrybutu `align=center`, ewentualnie odpowiedniego stylu opisującego użyty znacznik. Na przykład:

```
Szlachetne owoce kumkwaty
<br>
<center>
<img src="kumquat.gif">
</center>
- posiłek w witaminy bogaty!
```

Jeśli trzeba uzyskać nieco dodatkowej przestrzeni ponad oraz pod wyśrodkowanym obrazkiem, używamy atrybutu `align=center`:

```
Szlachetne owoce kumkwaty
<p align=center>
<img src="kumquat.gif">
</p>
- posiłek w witaminy bogaty!
```

### 5.2.6.7. Atrybut align nie jest zalecany

Standardy HTML 4 oraz XHTML nie zalecają już stosowania atrybutu `align` w żadnym znaczniku, a więc także w `<img>`; mają go zastąpić style. Tym niemniej atrybut ten jest wciąż bardzo popularny wśród autorów HTML i jest ciągle obsługiwany przez najpopularniejsze przeglądarki. Tak więc można się spodziewać, że kiedyś atrybut ten zostanie zupełnie zaniechany, ale nie nastąpi to jeszcze tak szybko.

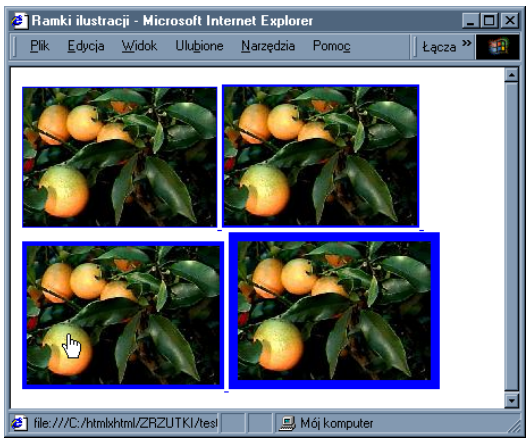
### 5.2.6.8. Atrybut border

Obrazki będące jednocześnie odsyłaczami (umieszczone wewnątrz znacznika `<a>`) są zazwyczaj przez przeglądarki opatrywane kolorową ramką o grubości dwóch pikseli — użytkownik dowiaduje się w ten sposób, że obrazek można kliknąć i w ten sposób przejść do innego dokumentu. Atrybut `border` pozwala zmienić grubość takiej ramki lub ją usunąć (`border=0`). Ten atrybut także nie jest już zalecany przez standardy HTML 4 i XHTML, wciąż jednak jest powszechnie stosowany.

Na rysunku 5.17 przedstawiono sposób, w jaki grubość ramek interpretuje Internet Explorer:

```
<a href="test.html">
<img src="kumquat.jpg" border="1" />
</a>
<a href="test.html">
<img src="kumquat.jpg" border="2" />
</a>
<a href="test.html">
```

```
<img src="kumquat.jpg" border="4" />
</a>
<a href="test.html">
<img src="kumquat.jpg" border="8" />
</a>
```




*Rysunek 5.17. Grube i cienkie ramki wokół obrazków*

### 5.2.6.9. Usuwanie ramki obrazka

Ramkę obrazka-odsyłacza można w ogóle usunąć, stosując w znaczniku `<img>` atrybut `border=0`. W przypadku niektórych ilustracji, szczególnie map odsyłaczy, taki zabieg stanowczo poprawia wygląd strony. Obrazki będące już na pierwszy rzut oka odsyłaczami do innych stron najlepiej wyglądają bez żadnych dodatkowych „ozdobników".

Tym niemniej przy usuwaniu obramowania takiej ilustracji trzeba uważać, aby nie spadła użyteczność strony. Brak ramki to brak jakiejkolwiek wskazówki, że dany element graficzny jest odsyłaczem — czytelnikowi automatycznie trudniej jest znaleźć „linki" na stronie. Co prawda, po najechaniu myszą na taki obrazek-odsyłacz wskaźnik zmienia wygląd, ale raczej nie powinno się oczekiwać od czytelników „badania" w ten sposób wszystkich obrazków na stronie.

Stanowczo należy wyraźnie zaznaczać, że określony obrazek pozbawiony ramki jest odsyłaczem. Wystarczy choćby krótka informacja tekstowa.

### 5.2.6.10. Atrybuty height i width

Na pewno nieraz Czytelnik zaobserwował efekt polegający na przesuwaniu się w poziomie różnych elementów strony w czasie jej ładowania. Dzieje się tak, gdy przeglądarka dopasowuje rozkład strony do już pobranych ilustracji. Po określeniu rozmiaru obrazka — co jest możliwe dopiero po pobraniu go i odczytaniu zawartych w pliku informacji — rezerwowana jest prostokątna przestrzeń na stronie[5]. Nie jest to najwydajniejszy sposób interpretacji dokumentu — przeglądarka musi zbadać każdy plik graficzny i obliczyć przestrzeń, jaką on zajmie, dopiero potem może zostać wstawiona dalsza treść. W ten sposób wydłuża się czas pobierania całej strony — to może zniechęcić czytelnika.

[5] To kolejny naoczny dowód, że obrazki są oddzielnymi plikami, ładowanymi niezależnie od strony.

Ten proces można jednak przyspieszyć, stosując atrybuty `height` i `width` znacznika `<img>`. Dzięki nim przeglądarka potrafi zarezerwować odpowiednią przestrzeń jeszcze przed pobraniem elementu graficznego, co przyspiesza wyświetlanie dokumentu i eliminuje efekt „przesuwania". Oba atrybuty przyjmują wartości w postaci liczb całkowitych odpowiadających wysokości (ang. *height*) oraz szerokości (ang. *width*) obrazka w pikselach. Nie jest istotna kolejność tych atrybutów.

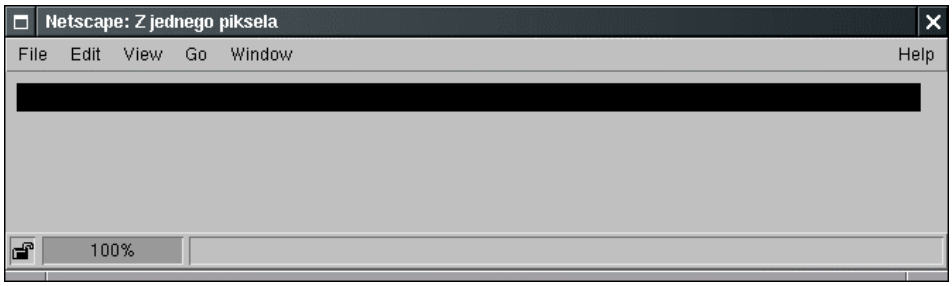
### 5.2.6.11. Zmiana wielkości ilustracji oraz wypełnianie

Atrybuty `height` i `width` pozwalają na zastosowanie pewnego triku, polegającego na podaniu innych rozmiarów obrazka niż są faktyczne. W takiej sytuacji przeglądarka automatycznie skaluje ilustrację tak, aby pasowała do określonej przestrzeni. W ten sposób można łatwo tworzyć miniaturki dużych ilustracji oraz powiększać małe obrazki. Trzeba jednak zachować tutaj ostrożność: bez względu na to, jakie rozmiary zostaną podane, przeglądarka i tak musi pobrać cały plik; jeśli natomiast zostaną zachwiane proporcje pomiędzy wysokością a szerokością, zaburzy to wygląd elementu graficznego.

Kolejny trik związany z opisywanymi atrybutami polega na wypełnianiu obszarów strony i zwiększaniu wydajności pobierania. Wyobraźmy sobie, że zamierzamy wstawić na stronie kolorową belkę biegnącą przez całą szerokość dokumentu[6]. Zamiast wstawiać obrazek pełnych wymiarów, wystarczy utworzyć „GIF-a" o rozmiarach jeden na jeden piksel i w żądanym kolorze; następnie określić atrybuty `height` i `width` zgodnie z wymaganymi rozmiarami:

```
<img src="obrazki/jeden_piksel.gif" width=640 height=20>
```

Mały, jednopikselowy obrazek ładuje się bardzo szybko, a dzięki opisywanym atrybutom „rozciąga się" do pożądanych rozmiarów (rysunek 5.18).




*Rysunek 5.18. Ten pasek utworzono za pomocą jednopikselowego obrazka*

Ostatni trik z atrybutem `width` polega na użyciu wartości procentowej zamiast wielkości w pikselach. Przeglądarka skaluje wtedy obrazek tak, by zajmował określoną procentowo część szerokości okna przeglądarki. Tak więc, aby stworzyć pasek o wysokości 20 pikseli i szerokości okna przeglądarki, użyjemy zapisu:

```
<img src="obrazki/jeden_piksel.gif" width="100%" height=20>
```

Po zmianie wielkości okna przeglądarki, wielkość obrazka zostanie odpowiednio dopasowana.

[6] Między innymi w ten sposób można tworzyć kolorowe linie poziome w przeglądarce Netscape 3 lub wcześniejszej, nie obsługującej atrybutu `color` w znaczniku `<hr>`.

Jeśli podamy szerokość procentową, a pominiemy wysokość, przeglądarka zachowa procentowy stosunek wielkości boków przy zmianie wielkości elementu graficznego. Oznacza to, że wysokość będzie zawsze proporcjonalna do szerokości i zawartość obrazka nie ulegnie zburzeniu.

### 5.2.6.12. Problemy z wysokością i szerokością

Atrybuty `height` i `width` w znaczniku `<img>` zwiększają prędkość ładowania obrazka i umożliwiają zastosowanie opisanych powyżej trików. Ale wiąże się z nimi również pewien problem: przeglądarka rezerwuje określoną przestrzeń nawet wtedy, gdy użytkownik wyłączył automatyczne pobieranie obrazków. W takim przypadku często okazuje się, że strona pełna jest pustych ramek z nic nie znaczącymi ikonkami w środku. Wygląda to bardzo nieciekawie i zazwyczaj utrudnia zorientowanie się w dokumencie. Jeśli nie poda się rozmiarów obrazka, przeglądarka po prostu wstawia na stronie samą ikonkę i przynajmniej nie ma trudności z odczytaniem tekstu.

Nie czas tutaj rozważać, które wyjście jest lepsze; warto natomiast stosować atrybut `alt` i tym samym dostarczać czytelnikowi opis tego, czego ewentualnie nie może zobaczyć (patrz punkt 5.2.6.3). W sumie, raczej należałoby skłaniać się do stosowania atrybutów `width` i `height` — każdy sposób przyspieszenia ładowania strony jest dobry.

### 5.2.6.13. Atrybuty hspace i vspace

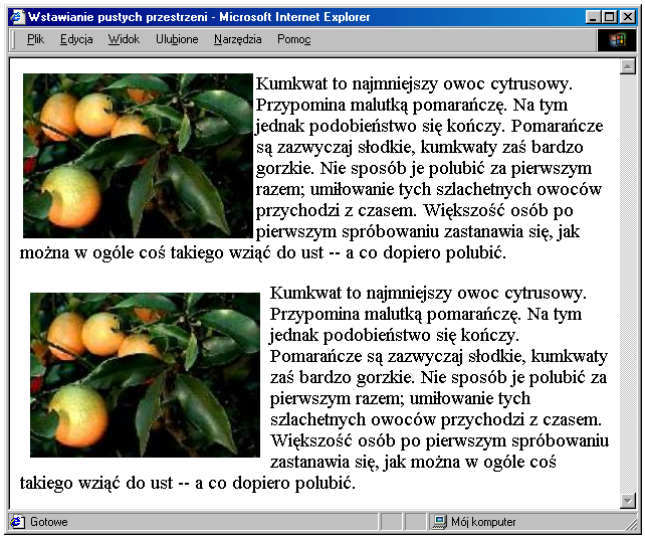
Przeglądarki graficzne zazwyczaj wyświetlają grafikę tak, że pomiędzy obrazkiem a tekstem nie pozostaje zbyt wiele miejsca. Jeśli nie umieścimy w obrazku przezroczystej ramki zwiększającej tę przestrzeń, to zazwyczaj dwupikselowy odstęp jest stanowczo za mały i wygląda nieestetycznie. A po wstawieniu obrazka w odsyłacz nawet przezroczysta ramka zostanie przesłonięta kolorowaną otoczką dołączaną przez przeglądarkę.

Przestrzeń wokół obrazka można jednak określić samodzielnie, za pomocą atrybutów `hspace` i `vspace`. Pierwszy umożliwia podanie w pikselach odległości pomiędzy obrazkiem a tekstem po jego prawej i lewej stronie, drugi — pozwala określić, jak duża przestrzeń ma zostać zachowana nad i pod ilustracją:

```
<img src="kumquat.jpg" align=left>
Kumkwat to najmniejszy owoc cytrusowy. Przypomina malutką pomarańczę.
Na tym jednak podobieństwo się kończy. Pomarańcze są zazwyczaj słodkie,
kumkwaty zaś bardzo gorzkie. Nie sposób je polubić za pierwszym razem;
umiłowanie tych szlachetnych owoców przychodzi z czasem. Większość osób
po pierwszym spróbowaniu zastanawia się, jak można w ogóle coś takiego
wziąć do ust -- a co dopiero polubić.
<p>
<img src="kumquat.jpg" align=left hspace=10 vspace=10>
Kumkwat to najmniejszy owoc cytrusowy. Przypomina malutką pomarańczę.
Na tym jednak podobieństwo się kończy. Pomarańcze są zazwyczaj słodkie,
kumkwaty zaś bardzo gorzkie. Nie sposób je polubić za pierwszym razem;
umiłowanie tych szlachetnych owoców przychodzi z czasem. Większość osób
po pierwszym spróbowaniu zastanawia się, jak można w ogóle coś takiego
wziąć do ust -- a co dopiero polubić.
```

Na rysunku 5.19 pokazano, jak powyższy kod jest interpretowany.

Chyba wszyscy się zgodzą, że dodatkowa przestrzeń wokół obrazka sprawia lepsze wrażenie i zwiększa czytelność strony.




*Rysunek 5.19. Dodanie przestrzeni wokół obrazka zwiększa czytelność strony*

### 5.2.6.14. Atrybuty ismap i usemap

Atrybuty `ismap` i `usemap` informują przeglądarkę, że element graficzny ma postać specjalnej mapy wizualnej zawierającej jeden lub więcej odsyłaczy, ilustrację taką najczęściej określa się mianem *mapy odsyłaczy*. Mapy odsyłaczy opisywane atrybutem `ismap` działają *po stronie serwera* (ang. *server-side*) i mogą być obsługiwane tylko poprzez znacznik `<a>`. *[<a>, 6.3.1]*

Na przykład:

```
<a href="/cgi-bin/obrazki/mapa2">
   <img src="mapa2.gif" ismap>
</a>
```

Kiedy użytkownik kliknie taki obrazek, przeglądarka automatycznie wysyła współrzędne x i y wskaźnika myszy (względem górnego lewego rogu obrazka) do serwera. Specjalne oprogramowanie po stronie serwera (w tym przykładzie program */cgi-bin/obrazki/mapa2*) po przeanalizowaniu otrzymanych danych podejmuje właściwe działanie.

Atrybut `usemap` służy do obsługi map odsyłaczy *po stronie klienta* (ang. *client-side*) i pozwala obyć się bez jakichkolwiek zabiegów wykonywanych na serwerze; w ten sposób eliminowane są również przestoje związane z opóźnieniami przesyłania danych przez sieć. Za pomocą specjalnych znaczników `<map>` i `<area>` określa się współrzędne „aktywnych" regionów ilustracji opisanej atrybutem `usemap` oraz podaje adres URL wywoływany po kliknięciu na takim regionie. Wartość atrybutu `usemap` to adres URL wskazujący na tę specjalną sekcję `<map>`.*[<map>, 6.5.3] [<area>, 6.5.4]*

Na przykład, w poniższym fragmencie opisano element graficzny `mapa2.gif` o rozmiarach 100×100 pikseli. Zdefiniowano cztery segmenty, które po kliknięciu odsyłają czytelnika do innych dokumentów. Zauważmy, że w znaczniku `<img>` umieszczono również atrybut `ismap`. Dzięki niemu można wprowadzić taką samą obsługę na serwerze, jak po stronie klienta i użytkownicy nie posiadający przeglądarek obsługujących `usemap` nie odczują żadnych niedogodności:

```
<a href="/cgi-bin/obrazki/mapa2">
   <img src="mapa2.gif" ismap usemap="#mapa2">
</a>

...
<map name="map2">
<area coords="0,0,49,49" href="link1.html">
<area coords="50,0,99,49" href="link2.html">
<area coords="0,50,49,99" href="link3.html">
<area coords="50,50,99,99" href="link4.html">
</map>
```

Rozwiązania tego typu świetnie nadają się do prezentacji map geograficznych — na przykład, firma może zamieścić na stronie mapkę z rozmieszczonymi oddziałami lokalnymi, które wystarczy kliknąć, aby przejść do strony najbliższego geograficznie punktu obsługi klienta. Zaletą atrybutu `usemap` jest fakt, że przetwarzanie obrazka nie wymaga oprogramowania ani jakichkolwiek zabiegów po stronie serwera, można więc go wykorzystać w stronach prezentowanych poza Internetem — np. na dysku lokalnym czy CD-ROM-ie.

Więcej informacji o punktach zakotwiczenia oraz odsyłaczach, a także o mapach odsyłaczy znajduje się w rozdziale 6.5.

### *5.2.6.15. Atrybuty class, dir, event, id, lang, style i title*

W wielu znacznikach opisujących zawartość obsługiwany jest pewien zestaw wspólnych atrybutów. Pozwalają one na identyfikację (`title`) oraz oznaczenie (`id`) zawartości znacznika w celu późniejszego odwołania się do danego elementu lub uproszczenia automatycznego przetwarzania; inne umożliwiają zmianę wyglądu elementu (`class`, `style`) oraz określenie języka i kierunku wyświetlania tekstu (`lang` i `dir`). *[atrybut `style`, 8.1.1] [atrybut `class`, 8.3]*

Ponadto istnieją jeszcze atrybuty opisujące reakcję na różne zdarzenia związane z danym elementem i wymagające pewnych zabiegów programistycznych (atrybuty `on...`). *[procedury obsługi zdarzeń JavaScript, 12.3.3]*

Z atrybutów tych, w kontekście ilustracji najważniejszy jest `id`. Umożliwia on oznaczenie obrazka i późniejsze odwołanie się do niego z programu lub poprzez funkcję przeglądarki (patrz rozdział 12.). *[atrybut `id`, 4.1.1.4]*

Pozostałe atrybuty raczej w nikły sposób oddziałują na znacznik `<img>`. Są pewne aspekty stylów, które mogą wpłynąć na jego zachowanie, można też podać tytuł (`title`), choć lepiej stosować atrybut `alt`. Trudno wyobrazić sobie, jak w kontekście grafiki mogą działać atrybuty języka (`lang`) lub kierunku tekstu (`dir`). *[atrybut `dir`, 3.6.1.1] [atrybut `lang`, 3.6.1.2] [atrybut `title`, 4.1.1.5]*

### *5.2.6.16 Atrybuty name, onAbort, onError, onLoad i inne związane ze zdarzeniami*

Przeglądarka Netscape obsługuje obecnie cztery atrybuty znacznika `<img>`, umożliwiające wykonywanie na ilustracji różnych czynności za pomocą programów JavaScript. Pierwszy z nich to atrybut `name`. Atrybut ten można zastąpić standardowym `id`[7], a umożliwia on oznaczenie elementu graficznego w celu późniejszego odwołania się do niego poprzez aplet JavaScript. Na przykład:

[7] Standard HTML 4.01 definiuje atrybut `name`, choć obecnie jest on obsługiwany tylko przez Netscape.

```
<img src="kumquat.jpg" name="kumquat">
```

Do obrazka opisanego tak jak powyżej można się potem odwołać z apletu JavaScript poprzez nazwę „kumquat" — np. w celu jego usunięcia lub zmodyfikowania. Nie jest możliwe operowanie na elemencie graficznym z poziomu JavaScript, jeśli nie został on nazwany poprzez atrybut `name` lub `id`.

Pozostałe trzy atrybuty umożliwiają przypisanie elementowi graficznemu procedur obsługi JavaScript. Wartość tych atrybutów to kod JavaScript w cudzysłowach. Kod ten może mieć postać jednego lub więcej wyrażeń JavaScript oddzielonych średnikami.

Przeglądarka Netscape uruchamia procedurę obsługi `onAbort` kiedy użytkownik zatrzyma pobieranie obrazka (zazwyczaj klikając przycisk „stop"). Za pomocą komunikatu wywoływanego poprzez `onAbort` można na przykład ostrzec użytkownika, że zatrzymuje on właśnie ładowanie bardzo ważnego obrazka — choćby mapy odsyłaczy (patrz punkt 6.5):

```
<img src="kumquat.jpg" usemap=#map1"
onAbort="window.alert('Uwaga: ten obrazek zawiera ważne odsyłacze.
Proszę pobrać go w całości.')">
```

Atrybut `onError` „dochodzi do głosu" wtedy, gdy w czasie pobierania obrazka nastąpi błąd, ale inny niż ten spowodowany brakiem pliku graficznego lub przerwaniem ładowania przez użytkownika. Atrybut taki można wykorzystać na przykład do podjęcia specjalnych kroków związanych z błędem lub załadowania zapasowego obrazka.

Zawartość atrybutu `onLoad` wykonywana jest natychmiast po udanym pobraniu i wyświetleniu obrazka.

Więcej informacji o programach JavaScript i procedurach obsługi można znaleźć w rozdziale 13.3.3.

#### *5.2.6.17. Łączenie atrybutów <img>*

Atrybuty standardowe i niestandardowe znacznika `<img>` można łączyć tam, gdzie tylko ma to sens. Kolejność atrybutów nie ma znaczenia. Trzeba tylko pamiętać, aby nie wstawiać ich nadmiarowo — trudno wtedy przewidzieć, jak przeglądarka się zachowa.

### *5.2.7. Rozszerzenia wideo*

Specjalne atrybuty niestandardowe znacznika `<img>`, jak `controls`, `dynsrc`, `loop` i `start`, działają tylko w przeglądarce Internet Explorer i nie zostały zdefiniowane w standardach HTML 4 ani XHTML. Umożliwiają zagnieżdżenie filmu w zawartości strony — podobnie jak w przypadku ilustracji.

Identyczny efekt w przeglądarce Netscape uzyskuje się poprzez program określany mianem modułu dodatkowego (ang. *plug-in*). Moduł to jednak rozwiązanie bardziej kłopotliwe dla użytkownika — aby obejrzeć film, musi on takiego „plug-ina" pobrać i zainstalować. W przeglądarce Internet Explorer mechanizm przeglądania filmów jest wbudowany i obsługiwany poprzez rozszerzenia znacznika `<img>`. *[Zawartość zagnieżdżona, 12.2]*

Ale z rozszerzeniem Internet Explorera do obsługi filmów wiążą się poważne ograniczenia: nie jest ono obsługiwane przez żadną inną przeglądarkę i obsługuje jedynie filmy w formacie Audio

Video Interleave (AVI) poprzez oprogramowanie wbudowane w system Microsoft Windows. Co więcej, dzięki najnowszym rozwiązaniom zastosowanym w przeglądarkach (szczególnie chodzi tu o obsługę obiektów i apletów), te rozszerzenia znacznika `<img>` mogą stać się zbędne.

### *5.2.7.1. Atrybut dynsrc*

Atrybut `dynsrc` znacznika `<img>` służy do wstawienia do strony filmu w formacie AVI; film taki może być wyświetlony jedynie przez przeglądarkę Internet Explorer. Wartość atrybutu to adres URL pliku filmowego, umieszczony w cudzysłowach. Na przykład, w poniższym znaczniku odwołujemy się do filmu *intro.avi*:

```
<img dynsrc="filmy/intro.avi">
```

Przeglądarka rezerwuje w oknie wewnętrzne „okienko", w którym odgrywany jest film (wraz z dźwiękiem, o ile taki został zarejestrowany w filmie i o ile potrafi odgrywać go nasz komputer). Przeglądarka Internet Explorer traktuje filmy `dynsrc` podobnie jak obrazek o określonych rozmiarach. I tak jak w przypadku ilustracji, film jest wyświetlany natychmiast po załadowaniu z serwera. Można to zachowanie zmienić, dodając możliwość sterowania wyświetlaniem przez użytkownika.

Ponieważ wszystkie inne przeglądarki ignorują te specjalne atrybuty służące do wyświetlania filmów, mogą „buntować się", gdy nie znajdą w znaczniku `<img>` wymaganego atrybutu `src` z adresem obrazka. Zaleca się więc — nawet w znacznikach odwołujących się do filmu — wstawianie tego atrybutu i podawanie jako jego wartość istniejącego pliku graficznego. Po napotkaniu takiego znacznika Internet Explorer wyświetli film, a nie wyświetli obrazka, inne przeglądarki postąpią odwrotnie. Kolejność atrybutów nie ma znaczenia. Na przykład:

```
<img dynsrc="filmy/intro.avi" src="obrazki/jedna_klatka.gif">
```

Internet Explorer pobiera i odgrywa film AVI *intro.avi*, inne przeglądarki graficzne załadują obrazek *jedna_klatka.gif*.

### *5.2.7.2. Atrybut controls*

Internet Explorer zazwyczaj odgrywa film w wewnętrznym „okienku" jeden raz i nie wyświetla żadnych dodatkowych elementów sterujących. Możliwe jest zatrzymanie, zrestartowanie i kontynuacja oglądania filmu poprzez kliknięcie myszą wewnątrz okienka. Za pomocą atrybutu `controls` (bez wartości) można do okienka podglądowego filmu dodać elementy sterujące, za pomocą których użytkownik może film odgrywać, przewijać, zatrzymywać i pauzować — zupełnie jak w magnetowidzie. Jeśli klip filmowy zawiera ścieżkę dźwiękową, pojawia się również „potencjometr" głośności. Na przykład, taki zapis:

```
<img dynsrc="filmy/intro.avi" controls src="obrazki/jedna_klatka.gif">
```

powoduje dodanie elementów sterujących do okienka podglądowego klipu filmowego tak, jak to pokazano na rysunku 5.20.


*Rysunek 5.20. Atrybut controls powoduje dodanie do okienka podglądowego klipu filmowego elementów sterujących*

### 5.2.7.3. Atrybut loop

Standardowo Internet Explorer po pobraniu odgrywa film jeden raz, od początku do końca. Atrybut `loop` znacznika `<img>` powoduje, że film będzie odgrywany tyle razy, ile wynosi wartość tego atrybutu (liczba całkowita) lub w nieskończoność, jeśli jako wartość atrybutu podamy `infinite`. Użytkownik może przerwać tę pętlę klikając na okienku filmu, wciskając przycisk stop (o ile podano atrybut `controls` — patrz 5.2.7.2) lub przechodząc do innego dokumentu.

W poniższym przykładzie film *intro.avi* zostanie odegrany od początku do końca, a następnie zrestartowany i odegrany w ten sposób jeszcze dziewięć razy:

```
<img dynsrc="filmy/intro.avi" loop=10 src="obrazki/jedna_klatka.gif">
```

Tutaj natomiast film będzie odgrywany raz po raz, w nieskończoność:

```
<img dynsrc="filmy/intro.avi" loop=infinite src="obrazki/jedna_klatka.gif">
```

Zapętlanie filmów niekoniecznie musi służyć do denerwowania oglądającego. Niektóre specjalne animacje to właśnie powtarzające się „na okrągło" sekwencje klatek. Zamiast tworzyć cały ciąg powtarzających się fragmentów, film wystarczy zapętlić i uzyska się taki sam efekt.

### 5.2.7.4. Atrybut start

Standardowo odgrywanie filmów w przeglądarce Internet Explorer rozpoczyna się po pobraniu pliku AVI. Zachowanie to można zmienić poprzez dodanie do znacznika `<img>` atrybutu `start`. Jeśli ustawimy jego wartość na `mouseover`, odgrywanie filmu zostanie odłożone aż do momentu najechania przez użytkownika myszką na okienko podglądowe. Inna wartość, `fileopen`, jest równoznaczna z ustawieniem domyślnym: odgrywanie ma się rozpocząć po pobraniu. Jeśli połączyć te dwie wartości w atrybucie `start`, odgrywanie filmu rozpocznie się natychmiast po pobraniu, a potem będzie powtarzane dopiero po najechaniu myszką na okienko podglądu. Aby połączyć wartości atrybutu `start`, wpisujemy je oddzielone przecinkami bez spacji; jeśli chcemy użyć spacji, musimy ująć całość w cudzysłowy.

Na przykład, kiedy przeglądarka zinterpretuje poniższy kod, plik *intro.avi* zostanie odegrany raz po załadowaniu dokumentu, a potem za każdym razem, gdy użytkownik przesunie wskaźnik myszy na okienko podglądu:

```
<img dynsrc="filmy/intro.avi" start="fileopen,mouseover"
     src="obrazki/jedna_klatka.gif">
```

### 5.2.7.5. Łączenie atrybutów <img> związanych z odgrywaniem filmu

Możliwe jest łączenie atrybutów znacznika <img>, zarówno tych specyficznych dla filmów, jak i standardowych. Na przykład, możliwe jest wyrównanie okienka podglądu filmu (lub jego zamiennika w postaci obrazka) do prawej strony okna przeglądarki:

```
<img dynsrc="filmy/intro.avi" src="obrazki/jedna_klatka.gif" align=right>
```

Łączenie atrybutów bywa bardzo przydatne. Zalecane jest także, gdzie tylko to możliwe, wstawianie atrybutu powodującego wyświetlenie elementów sterujących. Na przykład, jeśli zapętlamy film w nieskończoność, powinniśmy także dodać do znacznika <img> atrybut controls, tak aby użytkownik mógł zatrzymać film bez konieczności opuszczania dokumentu.

Jak już zostało to opisane w punkcie 5.2.7.4, łącząc atrybuty można również spowodować, że odgrywanie zostanie opóźnione aż do momentu, w którym użytkownik przesunie wskaźnik myszy nad okienko podglądowe. Wtedy „w zaczarowany sposób" film ożywia się i odgrywany jest w nieskończoność:

```
<img dynsrc="filmy/intro.avi" start=mouseover
     loop=infinite src="obrazki/jedna_klatka.gif">
```

# 5.3. Kolory dokumentu i grafika w tle

W standardach HTML 4 i XHTML zdefiniowano specjalne atrybuty znacznika <body>, umożliwiające definiowanie kolorów tekstu, odsyłaczy i tła dokumentu. Możliwe jest również określenie pliku graficznego, który będzie służył jako tło strony. Internet Explorer udostępnia jeszcze rozszerzenia pozwalające zdefiniować szerokości marginesów oraz zapewniające lepszą kontrolę nad sposobem wyświetlania grafiki w tle. Oczywiście, wszystkimi tymi parametrami można również sterować poprzez arkusze stylów obsługiwane przez współczesne przeglądarki.

## 5.3.1. Atrybuty dodatkowe i rozszerzające znacznika <body>

Atrybuty sterujące tłem, kolorem tekstu oraz marginesami dokumentu wstawiane są do znacznika <body>. *[<body>, 3.8.1]*

### 5.3.1.1. Atrybut bgcolor

Jednym ze standardowych sposobów zmiany domyślnego koloru tła okna przeglądarki, jest użycie atrybutu bgcolor znacznika <body>. Podobnie jak atrybut color znacznika <font>, bgcolor może przyjmować wartości dwóch rodzajów: liczbę określającą ilość barw czerwonej, zielonej i niebieskiej (RGB) lub standardową nazwę koloru. Omówienie kodowania RGB wraz z tabelą nazw kolorów można znaleźć w dodatku G.

Ustawienie koloru tła jest proste. Aby uzyskać tło barwy czystej czerwieni za pomocą kodowania RGB, wpisujemy:

```
<body bgcolor="#FF0000">
```

A teraz tło bardziej subtelne:

```
<body bgcolor="peach">
```

### 5.3.1.2. Atrybut background

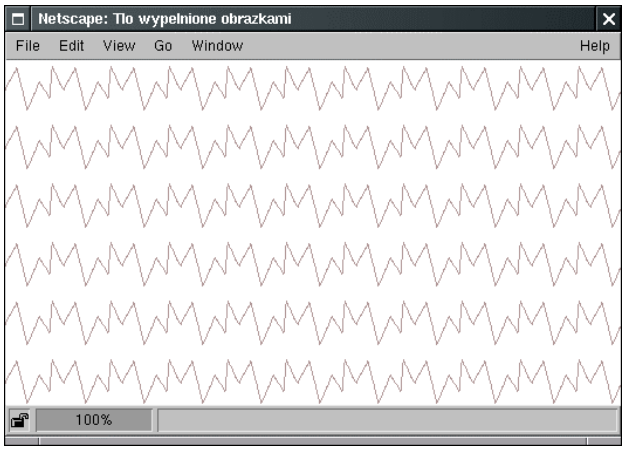
Jeśli nie wystarczy nam jednolity kolor, w tle możemy również umieścić obrazek. W tym celu użyjemy atrybutu `background`.

Wymaganą wartością tego atrybutu jest adres URL obrazka. Przeglądarka automatycznie powiela obrazek poziomo i pionowo, tak aby zapełnił całe okno.

Zazwyczaj powinno się wybierać niewielkie, stonowane elementy graficzne — tak, aby tło było interesujące, ale jednocześnie nie przeszkadzało w przeglądaniu strony. No i plik niewielkich rozmiarów szybciej przemierzy bezkresy Internetu niż ilustracja zajmująca cały ekran.

Na rysunku 5.21 przedstawiono sposób, w jaki przeglądarka o rozszerzonych możliwościach wyświetla jeden niewielki obrazek, powielając go tak, by zapełnione zostało całe tło:

```
<body background="obrazki/tlo.gif">
```

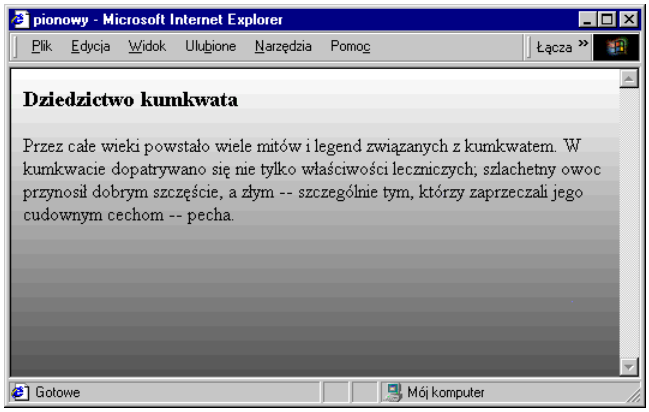



*Rysunek 5.21. Całe tło z jednego obrazka w wykonaniu Netscape*

Za pomocą elementów graficznych o różnych rozmiarach można tworzyć bardzo interesujące efekty w poziomie i w pionie. Na przykład, wąski i wysoki plik graficzny może służyć do oddzielenia nagłówka dokumentu:

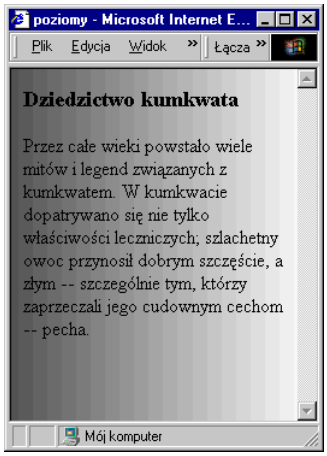
```
<body background="obrazki/pionowy.gif">
<h3>Dziedzictwo kumkwata</h3>
Przez całe wieki powstało wiele mitów i legend związanych z kumkwatem.
W kumkwacie dopatrywano się nie tylko właściwości leczniczych;
szlachetny owoc przynosił dobrym szczęście, a złym -- szczególnie tym,
którzy zaprzeczali jego cudownym cechom -- pecha.
```

Jeśli obrazek *pionowy.gif* jest wysoki i wąski, jaśniejszy u góry i ciemniejszy u dołu, uzyskamy efekt podobny do tego na rysunku 5.22.




*Rysunek 5.22. Efekt uzyskany po wstawieniu wysokiego i wąskiego tła w przeglądarce Internet Explorer*

Inny efekt da nam tło w postaci obrazka szerokiego, ale bardzo niskiego (rysunek 5.23).




*Rysunek 5.23. Tło szerokie, ale niskie*

Atrybut `background` według standardów HTML 4 i XHTML nie jest już zalecany — podobne efekty można uzyskać za pomocą stylów.

### *5.3.1.3. Atrybut bgproperties*

Atrybut `bgproperties` jest rozszerzeniem znacznika `<body>` obsługiwanym tylko przez przeglądarkę Internet Explorer. Przyjmuje tylko jedną wartość: `fixed`. Powoduje, że tło zostaje zablokowane i nie jest przewijane wraz z tekstem. Tak więc w poniższym przykładzie plik *znakH2O.gif* może służyć jako „znak wodny" strony:

```
<body background="obrazki/znakH2O.gif" bgproperties=fixed">
```

### 5.3.1.4. Atrybut text

Kiedy już zmienimy kolor lub grafikę służącą jako tło strony, możemy zająć się kolorem samego tekstu — czasem (np. gdy tło określimy jako czarne) jest to wręcz niezbędne. Standardowy atrybut `text` znacznika `<body>` służy właśnie do tego celu: ustawia kolor tekstu nie pełniącego roli odsyłacza dla całego dokumentu.

Wartość atrybutu `text` to kolor wyrażony podobnie jak w przypadku tła — za pomocą wartości RGB lub nazwy (patrz podpunkt 6.3.1.1 oraz dodatek G). Na przykład, aby uzyskać tekst niebieski na bladożółtym tle, napiszemy:

```
<body bgcolor="#777700" text="blue">
```

Oczywiście, najlepiej wybrać kolor kontrastowy względem koloru lub grafiki tła.

Standardy HTML 4 i XHTML nie zalecają już korzystania z atrybutu `text`, podobne efekty można uzyskać za pomocą stylów.

### 5.3.1.5. Atrybuty link, vlink oraz alink

Atrybuty `link`, `vlink` oraz `alink` znacznika `<body>` służą do określania koloru odsyłaczy (elementów opisanych znacznikiem `<a>`) w całym dokumencie. Wszystkie trzy przyjmują wartość w postaci kodu RGB lub nazwy koloru, podobnie jak `bgcolor` i `text`.

Atrybut `link` określa kolor wszystkich odsyłaczy, z których użytkownik jeszcze nie skorzystał. Atrybut `vlink` definiuje kolor odsyłaczy już wcześniej odwiedzonych (ang. *visited link*), zaś `alink` — odsyłaczy aktywnych (ang. *active link*), czyli właśnie wybranych (gdy użytkownik przesunie nad taki odsyłacz wskaźnik myszy i kliknie).

Podobnie jak kolor tekstu, kolor odsyłaczy musi być dobrze widoczny na tle strony, odsyłacz musi także wyróżniać się z otaczającego tekstu.

Standardy HTML 4 i XHTML nie zalecają już korzystania z tych atrybutów, podobne efekty można uzyskać za pomocą stylów.

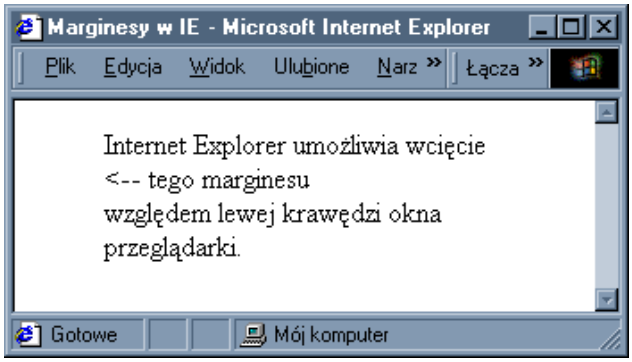
### 5.3.1.6. Atrybut leftmargin

Atrybut `leftmargin`, obsługiwany tylko przez przeglądarkę Internet Explorer, pozwala określić wcięcie lewego marginesu względem lewej krawędzi okna przeglądarki (podobnie jak na kartkach książki). Inne przeglądarki ignorują ten atrybut i wyrównują zawartość strony do lewego brzegu okna przeglądarki.

Wartość atrybutu `leftmargin` to liczba całkowita wyrażająca głębokość wcięcia lewego marginesu w pikselach, wartość domyślna to 0. Margines wypełniany jest kolorem lub grafiką tła.

Na przykład, Internet Explorer wyświetla poniższy tekst, jako wyrównany do lewego marginesu szerokiego na 50 pikseli (rysunek 5.24):

```
<body leftmargin=50>
Internet Explorer umożliwia wcięcie<br>
&lt;-- tego marginesu<br>
względem lewej krawędzi okna przeglądarki.
</body>
```




*Rysunek 5.24. Atrybut leftmargin w przeglądarce Internet Explorer pozwala uzyskać wcięcie określonej szerokości*

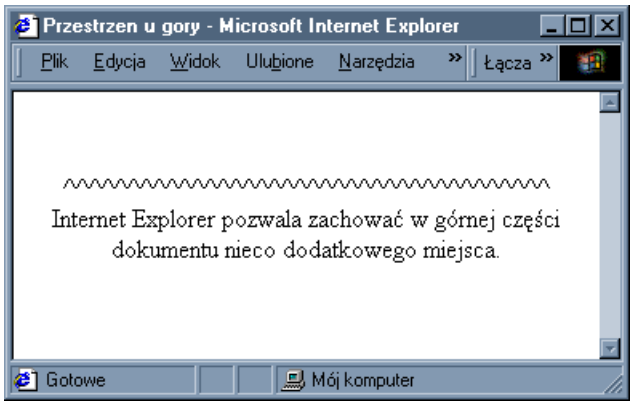
### 5.3.1.7. Atrybut topmargin

Podobnie jak `leftmargin`, atrybut `topmargin` jest obecnie obsługiwany wyłącznie przez przeglądarkę Internet Explorer. Wstawiany jest do znacznika `<body>` i definiuje przestrzeń, jaka zostanie pozostawiona u góry dokumentu. Margines taki jest wypełniany kolorem lub grafiką tła.

Zawartość strony jest wyświetlana dopiero poniżej marginesu szerokiego na liczbę pikseli podaną jako wartość `topmargin`, ustawienie domyślne to 0.

Na przykład, poniższy tekst zostanie wyświetlony przynajmniej o 50 pikseli niżej niż górna krawędź okna przeglądarki (patrz rysunek 5.25):

```
<body topmargin=50>
<p align=center>
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Internet Explorer pozwala zachować w górnej części dokumentu
nieco dodatkowego miejsca.
</p>
</body>
```




*Rysunek 5.25. Działanie atrybutu topmargin w przeglądarce Internet Explorer*

#### 5.3.1.8. Atrybuty style i class

Określenie stylu związanego ze znacznikiem `<body>` możliwe jest również poprzez kaskadowe arkusze stylów CSS. Jednym ze sposobów jest określenie stylu wewnątrz znacznika, poprzez atrybut `style`; jednak zalecane jest umieszczenie stylu w nagłówku, poprzez znacznik `<style>` lub w zewnętrznym arkuszu. Odwołanie się z poziomu stylów do konkretnego znacznika `<body>` umożliwia atrybut `class` (choć jeśli w danym dokumencie może być tylko jeden znacznik `<body>`, to jaki jest cel określania nazwy klasy?). O stylach i definicjach klas — w rozdziale 8.

#### 5.3.1.9. Mieszanie i dopasowywanie atrybutów znacznika <body>

Atrybuty `background` i `bgcolor` mogą wystąpić w jednym znaczniku `<body>`, ale obrazek pełniący rolę tła zasłoni określony przez nas kolor — chyba że w pliku graficznym określono obszary przezroczyste, o których było powiedziane wcześniej. Ale nawet jeśli grafika całkowicie zasłania barwne tło, warto umieścić atrybut `bgcolor` w znaczniku `<body>` i ustawić go na odpowiednią wartość. Przecież użytkownik może wyłączyć pobieranie grafiki, a więc także grafiki tła. To może spowodować, że strona będzie wyglądała bardzo nieatrakcyjnie. Ponadto, jeśli nie wstawimy atrybutu `bgcolor` lub nie określimy grafiki tła, przeglądarki zignorują atrybuty opisujące kolor tekstu i odsyłaczy, zastępując je wartościami domyślnymi lub tymi określonymi przez użytkownika.

### 5.3.2. Jeszcze ostrzeżenie

Różne zabiegi z kolorami powiodą się tylko wtedy, gdy użytkownik posiada w komputerze wyświetlacz obsługujący przynajmniej 256 kolorów, dużą ilość pamięci, nieograniczoną przepustowość łącza sieciowego i dobrze widzi obraz. W rzeczywistości jednak wielu użytkowników ma monitory monochromatyczne lub oferujące tylko ograniczoną liczbę kolorów, niewiele pamięci RAM na buforowanie grafiki, łącze o bardzo niskiej przepustowości i... słaby wzrok.

Z powodu tych właśnie ograniczeń powinniśmy w miarę możliwości unikać opisywanych rozszerzeń. Podobnie jak za dawnych czasów użytkownicy macintoshy zmieniali w dokumentach kroje czcionek niczym szantażysta wyklejający list z wycinków gazet („Ja tu mam 40 fontów i mam zamiar użyć każdego z nich!"), wielu autorów nie może powstrzymać się przed dodaniem teksturowego tła w swoich stronach WWW („Ja tu mam 13 kor drzewnych i 22 marmury i mam zamiar użyć ich wszystkich!").

W rzeczywistości tła teksturowe wnoszą niewiele informacji do naszego dokumentu, chyba że całość została naprawdę dobrze przemyślana. Wartość dokumentu tak czy inaczej leży w jego treści — w tekście i odpowiednich ilustracjach, a nie w zachwycającym wzorze tła. Żeby nie wiem jak pięknie to wszystko wyglądało, czytelnikom takie ozdoby niewiele pomogą, a mogą jedynie zmniejszyć przejrzystość strony.

Lepiej więc nie korzystać z opisywanych „kolorowych" rozszerzeń języka, chyba że w celowo „frywolnych" miejscach sieci WWW lub tam, gdzie takie zabiegi rzeczywiście zwiększają wartość strony (np. na stronach reklamowych lub marketingowych).

### *5.3.2.1. Problemy z obrazkami w tle*

Oto problemy, na jakie można się natknąć w związku z obrazkami w tle:

- Czas ładowania dokumentu wydłuża się o czas potrzebny do załadowania obrazka. Wyświetlanie reszty dokumentu nie jest możliwe dopóty, dopóki nie zostanie pobrane tło.
- Obrazek w tle zajmuje miejsce w lokalnej pamięci podręcznej przeglądarki, być może kosztem innych, naprawdę użytecznych elementów graficznych. Przez to inne dokumenty, być może nawet nie posiadające żadnej grafiki w tle, pobierane są dłużej.
- W systemie czytelnika mogą być niedostępne kolory użyte w obrazku tła; spowoduje to dithering grafiki — duże obszary określonego koloru zamieniane są na powtarzające się wzorce kilku przybliżonych barw. To zmniejsza czytelność nie tylko samej grafiki, ale również tekstu.
- Ponieważ przeglądarka musi cały czas wyświetlać obraz w tle (a nie po prostu wypełnia tło jednym kolorem), przewijanie dokumentu może odbywać się bardziej „opornie".
- Nawet jeśli tego od razu nie zauważamy, to zawsze tekst na tle grafiki będzie trudniej przeczytać. Czasem może to być niemożliwe.
- W różnych komputerach są dostępne różne kroje czcionek; fonty użyte w przeglądarce autora mogą współgrać z tłem, fonty u czytelnika mogą się z tłem gryźć.

### *5.3.2.2. Problemy z tłem, tekstem i kolorami odsyłaczy*

Całkiem sporo problemów może również wystąpić w związku z kolorami tła. Oto niektóre:

- Wybrany kolor, według nas taki śliczny, może czytelnikowi wydawać się okropny. Po co denerwować odbiorcę i zmieniać kolory, które ustawił sobie w przeglądarce?
- Być może jesteśmy zwolennikami stylu „biało na czarnym"; wiele osób woli natomiast styl praktykowany od ponad trzech tysięcy lat, czyli „czarno na białym". Zamiast próbować trafić w czyjeś gusta, po prostu przyjmijmy, że użytkownik ustawił już sobie przeglądarkę według własnych upodobań.
- Są osoby nie potrafiące rozróżniać kolorów. My stworzymy stronę o wydumanej kombinacji barw, dla nich pozostanie ona całkowicie nieczytelna. Szczególnie trzeba uważać na to, by nie używać koloru zielonego do oznaczania odsyłaczy odwiedzonych lub nieodwiedzonych. Miliony mężczyzn nie potrafią odróżnić koloru zielonego od czerwonego.
- Godzinami dobierana barwa może nie być dostępna na wyświetlaczu czytelnika. Przeglądarka wybiera wtedy najbliższy możliwy kolor. W przypadku wyświetlaczy o bardzo ograniczonej liczbie kolorów (takich jak te w kilku milionach używanych wciąż systemów Windows z kartami VGA obsługującymi 16 kolorów), najbliższy kolor dla tła i tekstu może się okazać... tym samym kolorem!
- Z tych samych powodów identycznie mogą wyglądać odsyłacze odwiedzone, nieodwiedzone i aktywne.
- Zmieniając kolory tekstu, a szczególnie odsyłaczy odwiedzonych i nieodwiedzonych, możemy wprowadzić całkiem spore zamieszanie. Zmuszamy wtedy czytelnika do eksperymentowania, klikania to tu, to tam, i odkrywania „co jest czym" na naszej stronie.

- Większość projektantów stron nie ma wiedzy z zakresu psychologii poznawczej, sztuk pięknych czy projektowania przemysłowego, a jednak beztrosko „bawi się" kolorami. Czasem warto jednak spytać specjalistę, co sądzi o takim, a nie innym doborze kolorów.

#### 5.3.2.3. *A jednak...*

Nie można jednak zaprzeczyć, że dzięki tym rozszerzeniom można stworzyć naprawdę niesamowicie wyglądające strony. A eksperymentowanie z kolorami to naprawdę świetna zabawa. Należy więc testować i realizować ciekawe pomysły. Po prostu trzeba to robić z rozwagą.

# 5.4. Dźwięk w tle

Jest jeszcze jeden rodzaj „multimediów" dostępny dla dużej części internautów: dźwięk. Większość przeglądarek traktuje pliki dźwiękowe jak oddzielne dokumenty, pobierane i wyświetlane przez specjalne aplikacje pomocnicze, aplety lub moduły rozszerzające. Z drugiej strony, np. Internet Explorer posiada wbudowany dekoder dźwięku i obsługuje specjalny znacznik umożliwiający zintegrowanie z dokumentem pliku audio i odgrywanie go w tle jako „podkład" strony. *[aplety i obiekty, 12.1] [zawartość zagnieżdżona, 12.2]*

Trzeba pochwalić programistów Internet Explorera za zbudowanie mechanizmu spajającego dźwięk z dokumentami HTML i XHTML — tworzy on naprawdę ciekawe możliwości. Jednocześnie trzeba jednak przestrzec autorów wykorzystujących te specyficzne znaczniki i atrybuty: uzyskany w ten sposób efekt nie będzie widzialny (a raczej słyszalny) na innych przeglądarkach. I nie można zakładać, że w przyszłości w tych innych przeglądarkach dźwięk będzie obsługiwany właśnie w ten sposób. Miejmy się więc na baczności.

## 5.4.1. Znacznik <bgsound>

Za pomocą znacznika `<bgsound>` można stworzyć „podkład muzyczny" strony WWW. Znacznik działa tylko w przeglądarce Internet Explorer. Plik audio jest pobierany i odgrywany w czasie pierwszego pobrania i wyświetlania dokumentu. Po odświeżeniu strony plik odgrywany jest ponownie.

### <bgsound> 0

***Funkcja:***

Odgrywa podkład dźwiękowy strony WWW

***Atrybuty:***

LOOP SRC

***Znacznik zamykający:***

w HTML-u brak

***Zawiera:***

Nic

***Stosowany wewnątrz:***

body_content

### 5.4.1.1. Atrybut src

Atrybut `src` w znaczniku `<bgsound>` jest wymagany. Jego wartość to adres URL odpowiedniego pliku dźwiękowego. Na przykład, kiedy użytkownik pierwszy raz pobierze dokument zawierający znacznik:

```
<bgsound src="audio/powitanie.wav">
```

odegrany zostanie jeden raz plik *powitanie.wav* — prawdopodobnie zawierający komunikat powitalny.

Obecnie przeglądarka Internet Explorer obsługuje trzy różne formaty plików dźwiękowych: `wav` (podstawowy format obsługiwany przez komputery PC), `au` (podstawowy format dźwiękowy w systemach uniksowych) oraz MIDI (uniwersalny schemat kodowania zapisu muzycznego). Informacje o obsługiwanych formatach zawarto również w tabeli 5.1.

*Tabela 5.1. Popularne formaty danych multimedialnych i odpowiadające im rozszerzenia nazw plików*

| Format | Typ | Rozszerzenie | Rodzima platforma formatu |
|---|---|---|---|
| GIF | grafika | *gif* | wszystkie |
| JPEG | grafika | *jpg, jpeg, jpe* | wszystkie |
| XBM | grafika | *xbm* | Unix |
| TIFF | grafika | *tif, tiff* | wszystkie |
| PICT | grafika | *pic, pict* | wszystkie |
| Rasterfile | grafika | *ras* | Sun |
| MPEG | film | *mpg, mpeg* | wszystkie |
| AVI | film | *avi* | Microsoft |
| QuickTime | film | *qt, mov* | Apple |
| AU | dźwięk | *au, snd* | Sun |
| WAV | dźwięk | *wav* | Microsoft |
| AIFF | dźwięk | *aif, aiff* | Apple |
| MIDI | dźwięk | *midi, mid* | wszystkie |
| PostScript | dokument | *ps, eps, ai* | wszystkie |
| Acrobat | dokument | *pdf* | wszystkie |

### 5.4.1.2. Atrybut loop

Podobnie jak w przypadku zagnieżdżanych plików filmowych, atrybut `loop` znacznika `<bgsound>` powoduje, że podkład dźwiękowy jest odgrywany określoną liczbę razy (lub „na okrągło"), a przynajmniej do czasu, gdy użytkownik przejdzie do następnej strony lub zamknie przeglądarkę.

Wartość atrybutu `loop` to liczba całkowita oznaczająca liczbę odtworzeń pliku dźwiękowego lub `infinite` (plik odtwarzany w nieskończoność).

Na przykład:

```
<bgsound src="audio/tadum.wav" loop=10>
```

spowoduje odegranie melodyjki „tadum" dziesięć razy, natomiast:

```
<bgsound src="audio/halas.wav" loop=infinite>
```

oznacza odtwarzanie pliku *halas.wav* w nieskończoność.

### 5.4.2. Alternatywny sposób odgrywania dźwięków

Istnieją także inne sposoby dołączania dźwięku do dokumentów, na przykład za pomocą bardziej ogólnych mechanizmów obsługujących również inne typy zagnieżdżonej zawartości multimedialnej. Najbardziej znaną alternatywą znacznika `<bgsound>` jest `<embed>`, pierwotnie obsługiwany tylko przez Netscape; znacznik ten w standardach HTML 4 i XHTML został zastąpiony elementem `<object>`. Więcej informacji na ten temat można znaleźć w rozdziale 12.

## 5.5. Animacja tekstu

W przeglądarce Internet Explorer, chyba głównie z myślą o twórcach reklam, udostępniono obsługę animowanego tekstu. Animacja nie jest skomplikowana — tekst przewijany jest poziomo w oknie przeglądarki, ale dobrze nadaje się do uwydatnienia sloganów, haseł, nagłówków i innych tego typu informacji. Z drugiej strony, podobnie jak w przypadku znacznika `<blink>`, taki animowany tekst może szybko zdenerwować odbiorcę. Jeśli więc koniecznie musimy z takich urozmaiceń korzystać, róbmy to z rozwagą.

### 5.5.1. Znacznik <marquee>

Znacznik `<marquee>` opisuje tekst przewijany poziomo w oknie przeglądarki. Jest obsługiwany tylko przez Internet Explorera i nie wchodzi w skład standardów. Rozmiarem przewijanego obszaru, jego wyglądem, sposobem wyrównania oraz szybkością przewijania sterują odpowiednie atrybuty.

Znacznik `<marquee>` oraz jego atrybuty są ignorowane przez pozostałe przeglądarki, nie dotyczy to jednak samej zawartości tego elementu. Zawartość ta wyświetlana jest po prostu jako tekst statyczny, bez formatowania opisanego znacznikami.

#### 5.5.1.1. Atrybut align

Internet Explorer umieszcza tekst `<marquee>` w otaczającej zawartości podobnie jak elementy graficzne — można więc stosować tutaj takie same typy wyrównania. Atrybut `align` przyjmuje wartości `top`, `middle` lub `bottom` i powoduje odpowiednie wyrównanie animowanego tekstu względem otaczającej zawartości. Tak więc:

```
<marquee align=top>
```

spowoduje zrównanie obszaru przewijanego ze szczytem otaczającego tekstu. Wpływ na wygląd całości mają tutaj także opisane dalej atrybuty `height`, `width`, `hspace` i `vspace`, sterujące rozmiarami tego obszaru.

### <*marquee*> O

***Funkcja:***

Tworzy obszar tekstu przewijanego

***Atrybuty:***

| | |
|---|---|
| ALIGN | LOOP |
| BEHAVIOR | SCROLLAMOUNT |
| BGCOLOR | SCROLLDELAY |
| CLASS | STYLE |
| DIRECTION | VSPACE |
| HEIGHT | WIDTH |
| HSPACE | |

***Znacznik zamykający:***

`</marquee>`, nigdy nie jest pomijany

***Zawiera:***

plain_text

***Stosowany wewnątrz:***

body_content

### *5.5.1.2. Atrybuty behavior, direction oraz loop*

Te trzy atrybuty sterują stylem, kierunkiem oraz czasem przewijania tekstu.

Atrybut `behavior` przyjmuje jedną z trzech wartości:

`scroll` (domyślna)

Wartość `scroll` powoduje, że przewijany tekst wygląda jak reklama na placu Times Square: najpierw opisany obszar jest pusty, potem „wjeżdża" na niego tekst z jednej strony (z której — to zależy od atrybutu `direction`) i przewijany jest aż do „opuszczenia" tego obszaru.

`slide`

Ta wartość powoduje, że opisywany obszar jest początkowo pusty, potem tekst przewijany jest od jednej strony do drugiej. Po osiągnięciu drugiej krawędzi obszaru tekst zatrzymuje się i przewijany jest w przeciwnym kierunku. Teraz już przez cały czas pozostaje widoczny.

`alternate`

Tekst jest od początku w pełni widzialny na jednym z końców obszaru przewijania. Następnie przesuwany jest do drugiego końca, tam zatrzymuje się i „wraca".

Jeśli nie określimy atrybutu `behavior`, domyślnie przyjmowana jest wartość `scroll`.

Atrybut `direction` określa kierunek przewijania tekstu. Dopuszczalne wartości to `left` (domyślna) lub `right`. Zauważmy, że początkowa pozycja tekstu to przeciwieństwo kierunku przewijania: `left` oznacza, że tekst na początku znajduje się po prawej stronie, a potem jest przewijany w lewo. Pamiętajmy również, że osoby przyzwyczajone do czytania z lewej do prawej będą miały trudności z odczytaniem tekstu przewijanego w prawą stronę.

Atrybut `loop` przyjmuje wartość całkowitą i określa liczbę „przewinięć" tekstu. Można również określić jego wartość jako `infinite` — wtedy tekst przewijany będzie dopóty, dopóki użytkownik nie pobierze do przeglądarki nowego dokumentu.

Spójrzmy jak to wygląda w praktyce:

```
<marquee align=center loop=infinite>
  Kumkwaty nie są zapychające
  ..............      I mają doskonały smak!
</marquee>
```

W powyższym przykładzie tekst rozpoczyna „wędrówkę" po prawej stronie okna (ustawienie domyślne), jest przewijany do samego końca w lewo, a następnie znów „pojawia się" z prawej — i tak w kółko, dopóki użytkownik nie przejdzie do następnej strony. Zauważmy obecność kropek i spacji; nie jest możliwe dołączanie jednego przewijanego napisu do drugiego.

Przewijanie z atrybutem `slide` nie prezentuje się korzystnie przy włączonym powtarzaniu.

### *5.5.1.3. Atrybut bgcolor*

Atrybut `bgcolor` umożliwia zmianę koloru tła przewijanego tekstu. Dopuszczalne wartości to albo liczba wyrażająca natężenie barw RGB, albo standardowa nazwa kolorów (patrz dodatek G). Aby stworzyć obszar przewijania o żółtym kolorze, trzeba napisać:

```
<marquee bgcolor=yellow>
```

### *5.5.1.4. Atrybuty height i width*

Atrybuty `height` i `width` określają rozmiary przewijanego obszaru. Jeśli ich nie podamy, obszar ten rozciąga się na całą szerokość okna Internet Explorera i jest akurat na tyle wysoki, żeby zmieścił się w nim przewijany tekst.

Oba atrybuty przyjmują wartości numeryczne, oznaczające absolutną wielkość w pikselach, lub wartości procentowe, oznaczające rozmiar w stosunku do wysokości i szerokości okna przeglądarki.

Na przykład, aby stworzyć obszar przewijania o wysokości 50 pikseli i zajmujący jedną trzecią okna przeglądarki, napiszemy:

```
<marquee height=50 width="33%">
```

Zazwyczaj obszar powinien być na tyle wysoki, żeby tekst się w nim mieścił, często natomiast długość określa się krótszą niż tekst — powstaje wtedy jakby „okienko podglądowe", za którym przesuwane są słowa.

#### 5.5.1.5. Atrybuty hspace i vspace

Atrybuty `hspace` i `vspace` służą do definiowania przestrzeni pomiędzy przewijanym tekstem a otaczającą zawartością.

Oba atrybuty wymagają jako argumentu liczby całkowitej wyrażającej wymaganą przestrzeń w pikselach. Atrybut `hspace` tworzy przestrzeń z lewej i prawej strony przewijanego tekstu, `vspace` opisuje przestrzeń nad i pod nim. Aby pozostawić wokół całego tekstu 10-pikselowy „margines", należy napisać:

```
<marquee vspace=10 hspace=10>
```

#### 5.5.1.6. Atrybuty scrollamount i scrolldelay

Te atrybuty opisują prędkość i płynność przewijania.

Wartość atrybutu `scrollamount` to liczba pikseli, o jaką przesuwany jest tekst za każdym razem. Im niższa wartość, tym płynniejsze — ale również wolniejsze — przewijanie.

Wartość atrybutu `scrolldelay` to liczba milisekund pomiędzy kolejnymi przesunięciami tekstu. Im jest ona mniejsza, tym przewijanie jest szybsze.

Za pomocą niskiej wartości `scrolldelay` można przyspieszyć przesuwanie tekstu opisanego niską wartością `scrollamount`. Na przykład:

```
<marquee scrollamount=1 scrolldelay=1>
```

Tekst będzie przesuwany co jeden piksel, ale tak szybko jak tylko to możliwe. W tym przypadku szybkość przewijania będzie ograniczona tylko możliwościami komputera użytkownika.

# 5.6. Inna zawartość multimedialna

W sieci WWW można przesyłać i udostępniać zupełnie dowolną zawartość. W tej części rozdziału przyjrzymy się innym sposobom odwoływania się do grafiki, filmów, materiałów dźwiękowych i dokumentów w innych formatach.

## 5.6.1. Zawartość zagnieżdżona a odwołanie do zawartości

Pośród różnych formatów multimedialnych grafika jest traktowana specjalnie przez przeglądarki (poza nielicznymi): możliwe jest wstawianie obrazków bezpośrednio w dokument HTML lub XHTML i wyświetlanie w jednym oknie wraz z pozostałą zawartością. Czasem jednak, o czym już wspomniano, warto jest odwoływać się do zewnętrznych plików graficznych. Dotyczy to szczególnie plików dużych, w których liczą się detale, ale niekoniecznie obowiązkowo obecnych w samym dokumencie. Inne elementy multimedialne — np. cyfrowe audio i wideo, również dołączane są poprzez zewnętrzne odsyłacze.

Zazwyczaj do odwołania się do zewnętrznych elementów multimedialnych w bieżącym dokumencie stosujemy znacznik zakotwiczenia (<a>). Podobnie jak w przypadku innych odsyłaczy, po jego wybraniu przeglądarka pobiera obiekt multimedialny i prezentuje go użytkownikowi, być może za pośrednictwem zewnętrznej aplikacji lub modułu rozszerzającego. Proces ten zawsze składa się z dwóch etapów: przedstawienie dokumentu odwołującego się do obiektu multimedialnego, a dopiero potem przedstawienie samego obiektu (po wybraniu odsyłacza). *[<a>, 6.3.1]*

W przypadku obrazków można wybrać spośród dwóch sposobów prezentacji: jako element zagnieżdżony w dokumencie (znacznik <img>) lub jako element zewnętrzny (<a>). Jeśli elementy graficzne są niewielkie i stanowią integralną część dokumentu, powinno się je wstawiać bezpośrednio na stronie. Jeśli są duże i mają tylko dodatkowe znaczenie — poprzez znacznik <a>.

Jeśli wybierzemy ten drugi sposób, czasem warto poinformować czytelnika, jak duży jest plik, który ma zostać pobrany; można również przedstawić jego miniaturkę. Użytkownik sam zdecyduje, czy obrazek warto pobrać.

## 5.6.2. Odwołania do materiałów graficznych, audio i wideo

Do dowolnego zewnętrznego dokumentu, bez względu na jego format lub typ, odwołujemy się za pomocą konwencjonalnego znacznika <a>:

```
<a href="muzyka/hymn.au">Hymn Hodowców Kumkwatów</a> to hołd oddany
tysiącom hodowców szlachetnych owoców na całym świecie.
```

Podobnie jak w przypadku dowolnych innych dokumentów, do których się odwołujemy, serwer dostarcza obiekt multimedialny do przeglądarki kiedy tylko użytkownik wybierze odsyłacz. Jeśli przeglądarka „zauważy", że pobrany dokument nie jest w formacie HTML lub XHTML, automatycznie uruchamia odpowiednie narzędzie lub w inny sposób przedstawia pobraną zawartość użytkownikowi.

Możliwe jest skonfigurowanie przeglądarki tak, by do obsługi różnych formatów dokumentów wykorzystywała odpowiednie aplikacje pomocnicze. Na przykład, do odgrywania plików dźwiękowych może służyć program przetwarzający dane audio, do odgrywania filmów — odgrywarka wideo. Jeśli przeglądarka nie została skonfigurowana do obsługi określonego formatu dokumentu, poinformuje o tym użytkownika i będzie on mógł po prostu zachować taki plik na dysku. Potem z pobranych w ten sposób danych można skorzystać za pomocą oddzielnego narzędzia.

Przeglądarki określają format i specjalnie traktują pliki multimedialne, kierując się dwiema wskazówkami: albo typem MIME (Multipurpose Internet Mail Extension) opisującym plik i dostarczanym przez serwer, albo rozszerzeniem nazwy pliku. Preferowany sposób to MIME — za pomocą tego standardu można dość szczegółowo opisać plik i jego zawartość. Jednak w rozpoznawaniu pierwszorzędne znaczenie ma rozszerzenie — np. *.gif* lub *.jpg* (pliki GIF i JPEG) czy *.au* (specjalny plik dźwiękowy).

Ponieważ nie wszystkie przeglądarki obsługują typy MIME i nie we wszystkich skonfigurowano odpowiednie aplikacje pomocnicze, w nazwach obiektów multimedialnych zawsze należy stosować poprawne rozszerzenia (patrz tabela 5.1 we wcześniejszej części rozdziału).

## 5.6.3. Jak odwoływać się do obiektów zewnętrznych?

Bardzo ważne jest, aby dobrze przemyśleć sposób, w jaki odwołamy się do zewnętrznego obiektu multimedialnego. Użytkownik musi wiedzieć, jaki element kryje się za odsyłaczem oraz jaka aplikacja ma zostać wykonana. Ponadto większość obiektów multimedialnych to duże pliki, więc wypada jakoś zaznaczyć, jak długiego pobierania może się spodziewać użytkownik.

Oprócz odsyłacza do dużego pliku graficznego i opisującego go tekstu warto również wstawić miniaturkę obrazka, dzięki której czytelnik zorientuje się, czy duży plik może być dla niego przydatny.

## 5.6.4. Zagnieżdżanie innych typów dokumentów

W sieci WWW można zamieścić niemal dowolny typ dokumentu w postaci elektronicznej — nie tylko grafikę, dźwięk czy pliki wideo. Aby je jednak wyświetlić, przeglądarka musi skorzystać z aplikacji pomocniczej. Współczesne przeglądarki obsługują oprogramowanie w postaci wtyczek i jak opisano w rozdziale 12., mogą zostać rozszerzone tak, by na przykład wyświetlać obiekty multimedialne jako zagnieżdżone w tekście.

Na przykład, wyobraźmy sobie firmę, w której szeroka dokumentacja produktów przechowywana jest w formacie jakiejś popularnej aplikacji do składu tekstu, np. FrameMaker, Quark XPress czy PageMaker. Do dystrybucji takich dokumentów sieć WWW nadaje się wręcz doskonale, ale konwersja do HTML-a lub XHTML-a zajęłaby zbyt wiele czasu.

Rozwiązanie polega na przygotowaniu kilku dokumentów HTML lub XHTML służących jako katalogi plików we właściwych formatach i pobraniu tych plików, uruchamiających odpowiedni aplet. Można również założyć, że użytkownik zainstalował w przeglądarce odpowiednią wstawkę lub skonfigurował ją tak, by uruchamiana była aplikacja pomocnicza. Typową wstawką jest program Acrobat Reader. Kiedy przeglądarka pobiera dokument w formacie Acrobat (*.pdf*), wstawka ta jest automatycznie uruchamiana i dokument zostanie wyświetlony — często bezpośrednio w oknie przeglądarki.