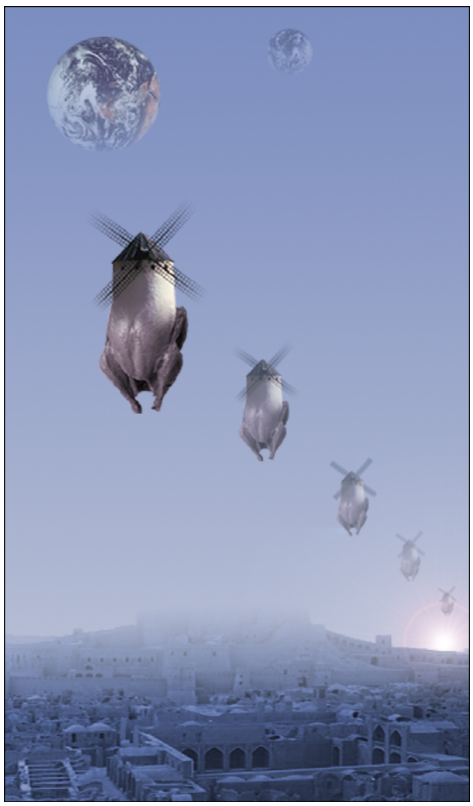
Arkadiusz Curulak

Wybrane zagadnienia programowania dla World Wide Web

część I
Wprowadzenie do HTML i CSS



Olsztyn 2001

Wybrane zagadnienia programowania dla World Wide Web
część I
Wprowadzenie do HTML i CSS

Olsztyn 28-12-2001

Ewie i Oli

Spis treści

Od autora

Niniejsze opracowanie powstało z myślą o wykorzystaniu jego jako materiału uzupełniającego oraz wspierającego naukę programowania dla World Wide Web, będącego elementem nauczania na studiach dziennych, zaocznych i podyplomowych w Wyższej Szkole Informatyki i Ekonomii Towarzystwa Wiedzy Powszechnej w Olsztynie, przedmiotów traktujących o sieciach komputerowych oraz ich administrowaniu, a w szczególności poruszających problematykę organizowania i zarządzania serwisami WWW.
Zdając sobie sprawę z faktu silnie rozwijającego się kierunku programowania dla WWW, autor traktuje to opracowanie również jako niezbędny wstęp do zagadnień związanych z bardziej zaawansowanymi technikami programowania dla WWW opartymi na językach XML, Perl, PHP, ASP, JavaScript czy JAVA, a które autor ma nadzieję przedstawić i przybliżyć studentom w przyszłości.

Ze względu na obszerność poruszanej tematyki oraz problemy wynikające z braku pełnej zgodności dostępnego oprogramowania klienckiego (UA) z opisywanymi specyfikacjami, pewne zagadnienia zostały opisane hasłowo. Dotyczy to w szczególności elementów CSS ze specyfikacji 2.0.
Całość materiału została podzielona na dwie części, z czego pierwsza opisuje język HTML (z elementami specyfikacji 4.01), a druga CSS (z elementami specyfikacji 2.0). W każdej z części autor starał się umieszczać jak największą liczbę przykładów, lepiej pokazujących działanie opisywanych elementów specyfikacji.
Wszystkie przykłady, a w szczególności zawierające arkusze stylów CSS, testowane były w najpopularniejszych systemach oraz przeglądarkach internetowych: MSIE 5.x (Windows9x/NT/2000, MacOS 8.x/9.x), Netscape 4.x (Windows9x/NT/2000, MacOS 8.x/9.x, Linux 2.x[1]), Opera 5.x (Windows9x/NT/2000), Konqueror 2.x (Linux 2.x) oraz Mozilla 0.7 (Linux 2.x).
Większość przykładów do testów pobierana była z serwera WWW. Jako serwer WWW wykorzystano darmowy serwer Apache[2] (Linux 2.x – RH7) oraz komercyjny produkt MS IIS 4.0 (Windows NT 4.0 Server).

[1] Do testów zastosowano Linux 2.x – dystrybucja RedHat 7.0/7.1 + KDE 2.0/2.1
[2] Apache, jest najpopularniejszym na świecie, darmowym, serwerem WWW – http://www.apache.org

Wiadomości wstępne

Protokół HTTP

Wszystkie przeglądarki stron WWW porozumiewają się z serwerami WWW za pomocą protokołu HTTP. Hypertext Transfer Protokol jest więc protokołem World Wide Web, wykorzystywanym podczas każdej transakcji w sieci WWW. Za jego pomocą przesyła się żądania udostępniania dokumentów WWW, informacje o kliknięciu odsyłacza (linku), czy dane pochodzące z formularzy.
Protokół HTTP określa formę żądań klienta dotyczących danych oraz formę odpowiedzi serwera na te żądania.

Języki wspierające programowanie dla WWW

Coraz rzadziej możemy spotkać w pełni statyczne serwisy WWW, zbudowane jedynie w oparciu o czysty język HTML z ewentualnymi dodatkami CSS. W chwili obecnej twórcy (programiści) serwisów internetowych (lub ich części) coraz częściej sięgają do silniejszych narzędzi (języków programowania), pozwalających m.in. dynamicznie generować dokumenty po stronie serwera, bądź też uruchamiać wbudowane w dokument programy. Najpopularniejszymi językami wspierającymi programowanie dla WWW są niewątpliwie JavaScript, VBScript, Perl, PHP, ASP czy też Java.
O sile i znaczeniu języka Java nie trzeba dzisiaj już nikogo przekonywać, dlatego w tym miejscu szczególną uwagę pragnę zwrócić na dwa wspaniałe narzędzia, niezwykle przydatne dla zaawansowanego programisty stron internetowych, jakimi są języki PERL i PHP. Niebagatelne znaczenie ma fakt, że oba języki doskonale pracują w darmowym środowisku Linux (czego nie można powiedzieć o ASP, który zorientowany jest na produkty z rodziny Windows). Pozwolę zatem sobie powiedzieć o nich jeszcze kilka słów.

PERL
Practical Extraction and Report Language[1]

Twórcą języka Perl oraz głównym opiekunem, czuwającym nad jego rozwojem, jest Larry Wall. Perl jest rozpowszechniany na licencji GNU General Public License, jest więc rozpowszechniany zarówno w postaci binarnej, jak i źródłowej za darmo. Sprzyja to dynamicznemu rozwojowi tego języka – i tak się właśnie dzieje. W chwili obecnej język Perl występuje na wielu platformach, m.in.: UNIX, Linux, MacOS, OS/2, MSDOS, Windows 9x/NT/2000, VMS, etc. Świadczy to niewątpliwie o jego bardzo silnej pozycji na rynku. Perl jest przeznaczony głównie do przetwarzania plików, łańcuchów oraz manipulowania procesami. Z języka Perl bardzo chętnie korzystają również programiści tworzący skrypty CGI na potrzeby dynamicznych stron WWW i WAP. Nie jest to wprawdzie głównym przeznaczeniem tego języka, niemniej jednak PERL bardzo silnie wspiera programowanie dla WWW i WAP.
Bardzo ważną cechą języka Perl jest to, że przy jego pomocy możemy dynamicznie generować nagłówek HTTP. Dzięki temu możemy tworzyć skrypty generujące dynamicznie dokumenty WWW (i WAP), budujące grafiki (np. wykresy),

[1] PERL – http://www.perl.com

przetwarzające bazy danych, analizujące dane przekazywane przez użytkownika za pomocą formularzy, etc.

PHP
Personal Home Page Tool

Powstanie języka PHP zawdzięczamy Rasmusowi Lerdorfowi., który za namową społeczności internetowej, opublikował pierwszą wersję języka PHP. Dość duże zainteresowanie nowym narzędziem przyczyniło się do dalszego jego rozwoju, co zaowocowało wersją PHP 2.0, znacznie rozwiniętą w stosunku do wersji poprzedniej. Dalszy rozwój tego języka oraz jego obecny kształt[1], to już praca szerszego grona programistów.
PHP jest doskonałym narzędziem do tworzenia dynamicznych serwisów internetowych, ponieważ z założenia jest do tego celu stworzony. Dzięki możliwości zintegrowania języka PHP z serwerem WWW Apache[2], skrypty wykonują się szybciej, niż w przypadku tradycyjnych skryptów CGI. Dodatkowym atutem PHP jest możliwość umieszczania kodu języka PHP bezpośrednio w dokumentach HTML czy WML. Taki sposób tworzenia dynamicznych stron internetowych znacznie przyspiesza ten proces. Język PHP, podobnie jak PERL, jest wieloplatformowy i jest w całości za darmo[3].
Podobnie jak w języku PERL, za pomocą PHP możemy dynamicznie wygenerować nagłówek HTTP, co daje nam możliwość wykorzystania siły PHP do dynamicznego tworzenia serwisów WWW (i WAP).
PHP od samego początku swego istnienia bardzo silnie wspiera obsługę baz danych[4]. Dzięki temu wygenerowanie strony WWW czy WAP zasilanej z bazy danych nie stanowi najmniejszego problemu. Dodatkowo w języku PHP zaimplementowano ogromną liczbę różnego przeznaczenia funkcji, a wśród nich m.in. pozwalające tworzyć oraz przetwarzać dynamicznie obrazy we wszystkich najpopularniejszych formatach graficznych wykorzystywanych w internecie.

[1] W chwili pisania tego skryptu bieżącą wersją PHP jest 4.0.6 (na podstawie http://www.php.org).
[2] Wprawdzie PHP najbardziej „związany" jest z serwerem Apache, jednak współpracuje również z innymi serwerami, m.in. z Xitami czy Internet Information Server firmy Microsoft.
[3] PHP rozprowadzane jest na licencji GNU General Public License.
[4] PHP wspiera obsługę ponad 20 baz danych (Adabas D, dBase, IBM DB2, Informix, InterBase, MS SQL, MySQL, ODBC, Oracle, PostgreSQL, Sybase, i inne).

I. HyperText Markup Language

1. Struktura dokumentu HTML

HyperText Markup Language został zdefiniowany w języku SGML (Standard Generalized Markup Language) i jest językiem zorientowanym specjalnie na potrzeby publikowania dokumentów w World Wide Web.

Strukturę dokumentu HTML można przedstawić ogólnie tak:

```
Informacja o wersji języka HTML
<HTML>
  Nagłówek dokumentu
  Ciało dokumentu
</HTML>
```

1.1. Informacja o wersji HTML

Dokument HTML powinien zawierać informację o wersji języka HTML, w jakiej został on utworzony. Informacja taka powinna być umieszczona na początku dokumentu.
W języku HTML 4.0 wyspecyfikowano trzy rodzaje definicji, które stosujemy w zależności od budowy dokumentu:

1. ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
        "http://www.w3.org/TR/REC-html40/strict.dtd">
   ```

   HTML 4.0 Strict DTD – wykorzystywany w dokumentach nie zawierających elementów i parametrów niezalecanych[*] oraz ramek.

2. ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
        "http://www.w3.org/TR/REC-html40/loose.dtd">
   ```

   HTML 4.0 Transitional DTD – wykorzystywany w dokumentach zawierających elementy i parametry niezalecane.

3. ```
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
        "http://www.w3.org/TR/REC-html40/frameset.dtd">
   ```

   HTML 4.0 Frameset DTD – wykorzystywany w dokumentach zawierających elementy i parametry niezalecane oraz ramki.

Dla dokumentów utworzonych przy użyciu języka zgodnego ze specyfikacją HTML 3.2 powinniśmy użyć następującej konstrukcji:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
  ... treść dokumentu ...
</HTML>
```

[*] Lista elementów i parametrów niezalecanych znajduje się w Dodatku I-A.
```

natomiast dla wersji HTML 4.0 na przykład takiej:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
         "http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
   ... treść dokumentu ...
</HTML>
```

1.2. Nagłówek dokumentu HTML (znacznik HEAD)

Jak już wspomniałem, każdy dokument HTML powinien posiadać nagłówek. W nagłówku możemy umieścić wiele cennych informacji o dokumencie, tj. jego tytuł, dane o autorze, krótką informację o jego treści. W tym miejscu możemy również zdefiniować sposób kodowania polskich liter, wczytywać skrypty i style umieszczone w zewnętrznych plikach, jak również możemy użyć konstrukcji automatyzujących czynności związane z odświeżaniem dokumentu czy wczytywaniem kolejnych dokumentów.

Nagłówek naszego dokumentu definiujemy przez umieszczenie jego treści między znacznikami `<HEAD>` i `</HEAD>`. Poniżej przedstawiamy sposób umieszczenia nagłówka w dokumencie:

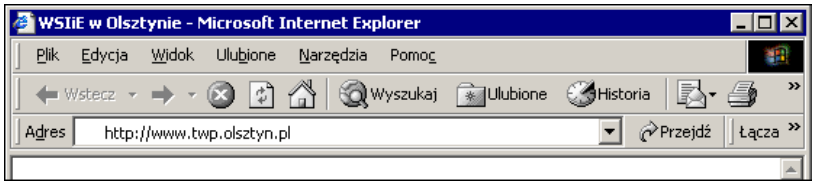
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
   <HEAD>
      ... treść nagłówka
   </HEAD>
   ... ciało dokumentu ...
</HTML>
```

Wprawdzie umieszczenie nagłówka w dokumencie HTML jest opcjonalne, jednak prawidłowo przygotowany dokument powinien zawierać tak zorganizowany blok nagłówkowy.

1.2.1. Znacznik TITLE

Każdy dokument HTML powinien posiadać tytuł. Tworzymy go przez umieszczenie treści tytułu między znacznikami `<TITLE>` i `</TITLE>`. Wymagane jest użycie obydwu znaczników. Wyspecyfikowany między znacznikami tytuł pojawi się w pasku tytułowym okna przeglądarki.

```
<HTML>
   <HEAD>
      <TITLE>WSIiE w Olsztynie</TITLE>
   </HEAD>
   ... ciało dokumentu ...
</HTML>
```



1.2.2. Znacznik META

Przy pomocy znacznika <META> możemy umieścić w nagłówku dokumentu wiele cennych informacji, np. dane o autorze, datę utworzenia dokumentu, określenie strony kodowej, informacje dla wyszukiwarek internetowych i wiele innych.
Znaczniki <META> należy umieszczać w nagłówku dokumentu bezpośrednio po znaczniku <HEAD>.

Z omawianym znacznikiem związane są cztery parametry:

- name - nazwa elementu meta,
- content - wartość elementu meta,
- http-equiv - informacja, która ma być zawarta w nagłówku HTTP dołączanym przez serwer do dokumentu,
- scheme - dodatkowe informacje dotyczące sposobu interpretacji danych zawartych w znaczniku meta.

Stosowanie znacznika </META> jest niezalecane.

Oto najważniejsze zastosowania tego znacznika.

- Umieszczenie informacji o autorze:

```
<HEAD>
   <META name=”author” content=”Jan Naparstek”>
   <TITLE>Wielki błękit</TITLE>
</HEAD>
```

- Określenie strony kodowej dla dokumentu:

```
<META http-equiv="Content-Type"
      content="text/html; charset=iso-8859-2">
```

Powyższa linia definiuje sposób kodowania polskich liter (ą, ć, ę, ł, ...) w naszym dokumencie zgodnie z normą ISO-8859-2. Norma ta jest najbardziej rozpowszechniona, chociaż stosowana jest również czasami strona kodowa Windows 1250. Dla tego przypadku parametrowi content powinniśmy przypisać wartość:

```
"text/html; charset=windows-1250".
```

- Umieszczenie w dokumencie informacji wykorzystywanych przez wyszukiwarki internetowe:

  słowa kluczowe
  ```
  <META name=”keywords” content=”auto, samochód, motoryzacja”>
  ```

  opis strony
  ```
  <META name="description" content="Znajdziesz tu informacje o...”>
  ```

- Odświeżanie strony oraz sterowanie jej prezentacją:

```
<META http-equiv=”refresh” content=”20”>
```

Umieszczenie powyższego wiersza w nagłówku spowoduje, że załadowany do przeglądarki dokument będzie co 20 sekund odświeżany. Oznacza to, że jeżeli w treści dokumentu pojawią się zmiany, to przy kolejnym odświeżeniu przeglądający stronę zobaczy jej nową zawartość. Z kolei wiersz:

```
<META http-equiv=”refresh” content=”30; url=strona2.html”>
```

spowoduje, że po upływie 30 sekund zostanie automatycznie załadowany do okna przeglądarki dokument `strona2.html`. Umieszczenie podobnych wierszy w kolejnych dokumentach da ciekawy efekt prezentacji.

1.2.3. Znacznik LINK

Znacznik `<LINK>` definiuje powiązania z innymi dokumentami, tzn. że dla każdego dokumentu możemy określić dokument główny, poprzedni czy następny. Przy pomocy tego znacznika możemy również dołączyć zewnętrzny plik z arkuszami stylów.
Znacznik ten może występować wyłącznie w nagłówku dokumentu.

Oto przykładowe konstrukcje znacznika `<LINK>` :

```
<LINK rel="Index" href="../index.html">
<LINK rel="Prev" href="poprzedni.html">
<LINK rel="Next" href="nastepny.html">

<LINK rel="stylesheet" href="mojestyle.css">
```

Ze względu na to, że większość możliwości znacznika `<LINK>` nie jest wykorzystywana przez przeglądarki, szczególną uwagę zwrócę na ostatnią konstrukcję, za pomocą której możemy wczytywać zewnętrzne pliki ze stylami, a którą bardziej szczegółowo opiszę w części poświęconej stylom ”*Arkusze stylów CSS (Cascading Style Sheets)*”.

Stosowanie znacznika `</LINK>` jest niezalecane.

1.2.4. Znacznik BASE – adresowanie względne i bezwzględne

Zanim omówimy znacznik `<BASE>` powiemy sobie kilka słów na temat sposobów adresowania.
Wyróżniamy dwa sposoby adresowania: względne i bezwzględne.
Adresowanie względne charakteryzuje się tym, że nazwa pliku nie zawiera nazwy serwera (domeny serwera), np.:

```
<LINK href="artac/artac.html">
```

Fakt ten wymusza obowiązek znajdowania się takiego pliku na serwerze, na którym znajduje się nasza strona.

Z kolei **adresowanie bezwzględne** polega na tym, że nazwa pliku zawiera również nazwę serwera (domenę), dzięki czemu plik taki może znajdować się na innym serwerze. Przykładem adresowania bezwzględnego może być np.:

```
<LINK href="http://www.uninet.pl/artac/artac.html">
```

Za pomocą znacznika <BASE> można określić ścieżkę, która dla adresów względnych służyć będzie za bazę do adresu danego pliku.

Jeżeli więc umieścimy w nagłówku tak zdefiniowany adres bazowy:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Strona firmy UNINET</TITLE>
    <BASE href="http://www.uninet.pl">
  </HEAD>
  ... ciało dokumentu ...
</HTML>
```

to wszystkie adresy względne w dokumencie będą odwoływały się do serwera o adresie www.uninet.pl.
Oznacza to, że podany adres względny artac/artac.html będzie w rezultacie odwoływał się do pliku o adresie http://www.uninet.pl/artac/artac.html.

Stosowanie znacznika </BASE> jest niezalecane.

1.3. Ciało dokumentu HTML

Główną część dokumentu, zawierającą prezentowane przez nas informacje, stanowią dane umieszczone pomiędzy znacznikami <BODY> i </BODY>. Blok ten nazywamy ciałem dokumentu. Ciało dokumentu umieszczamy pod jego nagłówkiem.

```
<HTML>
  <HEAD>
    <META name="author" content="Marian Pianka">
    <TITLE lang="pl">Coś na ząb</TITLE>
  </HEAD>
  <BODY>
    ... treść dokumentu ...
  </BODY>
</HTML>
```

Wprawdzie stosowanie znaczników informujących o początku i końcu ciała dokumentu jest opcjonalne, to jednak zaleca się ich stosowanie.

Znacznik <BODY> posiada wiele parametrów, z których najważniejsze to:

- background - nazwa pliku z rysunkiem będącego tłem dla dokumentu (URI),
- bgcolor - kolor tła dokumentu,
- text - kolor tekstu dokumentu,
- link - kolor odsyłacza,

- `vlink` - kolor odwiedzonego odsyłacza,
- `alink` - kolor aktywnego odsyłacza,
- `style` - jednowierszowy styl dla dokumentu.

Ustawienie koloru czarnego dla tła dokumentu i koloru białego dla tekstu opiszemy następująco:

```
<BODY bgcolor="#000000" text="white">
   ... ciało dokumentu ...
</BODY>
```

Wypełnienie tła dokumentu obrazkiem `logo.gif` oraz określenie kolorów dla odsyłaczy zdefiniujemy tak:

```
<BODY background="logo.gif" link="navy" vlink="blue" alink="teal">
   ... ciało dokumentu ...
</BODY>
```

Poza wymienionymi wyżej parametrami, znacznik `<BODY>` posiada kilkanaście dodatkowych parametrów. Są nimi m.in. dwa zdarzenia występujące podczas ładowania dokumentu i jego opuszczania. Mowa tu o `onLoad` i `onUnLoad`.

```
<BODY onLoad="alert('Witaj na stronie WSIiE TWP w Olsztynie')"
      onUnLoad="alert('Zapraszamy ponownie...')" >

   ... ciało dokumentu ...
</BODY>
```

Powyższy zapis spowoduje, że w trakcie ładowania strony zostanie wyświetlone okienko z komunikatem „*Witaj na stronie WSIiE TWP w Olsztynie*", natomiast w trakcie opuszczania strony okienko z komunikatem „*Zapraszamy ponownie...*".

Bardzo ciekawym parametrem znacznika `<BODY>` jest `style`. Pozwala on na definiowanie jednowierszowego stylu dla dokumentu, np.:

```
<BODY style="font-size: 60pt; color: red">
```

Więcej na ten temat znajdziesz w części poświęconej stylom CSS.

2. Elementy dokumentu HTML

2.1. Teksty, nagłówki, czcionki

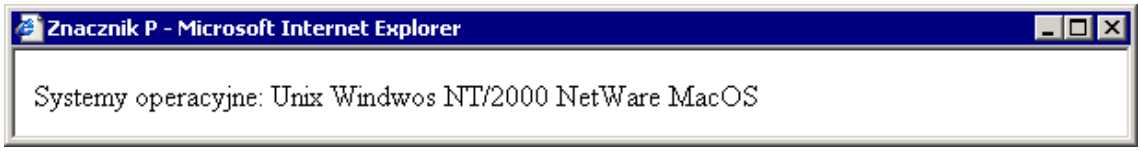
Na początku zajmiemy się znacznikami odpowiedzialnymi za organizację tekstu oraz definiowanie podstawowych parametrów czcionek. Omówimy sposoby wyróżniania fragmentów tekstu, wpływania na przejrzystość i czytelność dokumentów HTML.

2.1.1. Znaczniki akapitu P

Do formatowania akapitów używa się znaczników `<P>` i `</P>`. Zasadę ich działania opiszemy na przykładzie:

```
<HTML>
   <HEAD>
      <TITLE>Systemy operacyjne</TITLE>
   </HEAD>
   <BODY>
      Systemy operacyjne:
      Unix
      Windows NT
      NetWare
      MacOS
   </BODY>
</HTML>
```

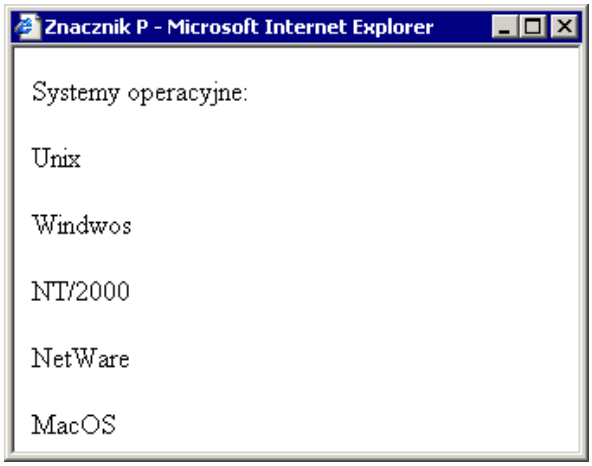
Mimo, iż tekst między znacznikami `<BODY>` i `</BODY>` został rozpisany w kolejnych wierszach, to znaki końca wiersza nie są poprawnie interpretowane przez przeglądarki, tj. w ich miejsce wstawiane są znaki odstępu. Tak więc w oknie przeglądarki zobaczymy tekst w postaci ciągu wyrazów:



Aby kolejne wiersze rozmieścić w kolejnych akapitach, musimy posłużyć się znacznikami `<P>` i `</P>` :

```
   <BODY>
      <P>Systemy operacyjne: </P>
      <P>Unix</P>
      <P>Windows NT</P>
      <P>NetWare</P>
      <P>MacOS</P>
   </BODY>
```

Efekt w przeglądarce będzie zatem taki:



Umieszczanie znacznika `</P>` jest opcjonalne.

Tekst umieszczany między znacznikami akapitów możemy dodatkowo formatować. Szczególnie przydatne do tego celu są parametry `align` i `style`. Przy pomocy pierwszego tekst w akapicie możemy wyrównywać do lewej i prawej, centrować czy

justować. Drugi parametr daje dużo większe możliwości i jest opisany szczegółowo w części poświęconej stylom.

```
<P align="center">...</P>
<P align="justify">...</P>

<P style="font-size: 30pt; color: green; text-align: right">...</P>
```

2.1.2. Łamanie wiersza (znacznik BR)

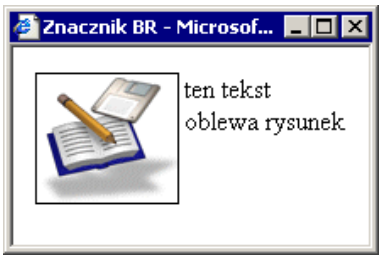
Jak już wcześniej wspomnieliśmy, znaki końca wiersza są ignorowane przez przeglądarki dokumentów HTML. Jeżeli chcemy przejść do nowego wiersza ("złamać" wiersz) musimy użyć znacznika <BR>.

```
<BODY>
  Wiersz pierwszy<BR>Wiersz drugi<BR>
  Wiersz trzeci
</BODY>
```

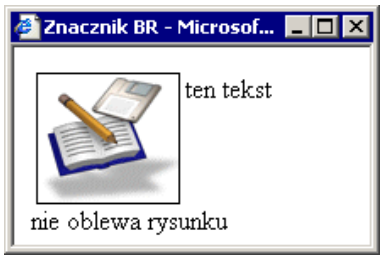
Stosowanie znacznika </BR> jest niezalecane.

Wraz ze znacznikiem łamania wiersza możemy dodatkowo stosować m.in. parametry clear i style. Za pomocą clear możemy przykładowo wymusić, aby tekst złamany znacznikiem <BR>, nie oblewał rysunku (lub innego obiektu).

```
<img src="pic.gif" align="left">ten
tekst<br>oblewa rysunek
```




```
<img src="pic.gif" align="left">ten
tekst<br clear="left">nie oblewa rysuneku
```



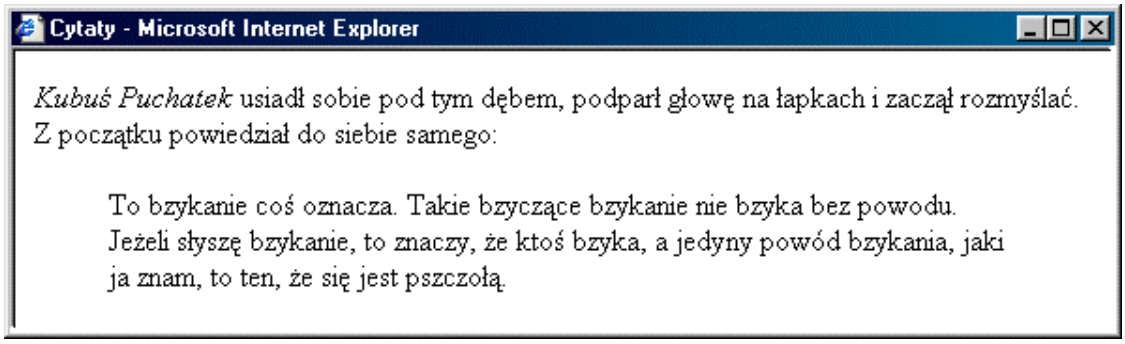
2.1.3. Cytaty (znaczniki BLOCKQUOTE i Q)

Do wyszczególniania dłuższych cytatów używa się znaczników <BLOCKQUOTE> i </BLOCKQUOTE>. Tekst wyszczególniony w taki sposób jest wcięty, a w niektórych przeglądarkach może być również pochylony.

```
<BODY>
  <CITE>Kubuś Puchatek</CITE> usiadł sobie pod tym dębem, podparł
  głowę na łapkach i zaczął rozmyślać.<BR>
  Z początku powiedział do siebie samego:

  <BLOCKQUOTE>
    To bzykanie coś oznacza. Takie bzyczące bzykanie nie bzyka bez
    powodu. Jeżeli słyszę bzykanie, to znaczy, że ktoś bzyka,
    a jedyny powód bzykania, jaki ja znam, to ten, że się jest
    pszczołą.
  </BLOCKQUOTE>
</BODY>
```

W przeglądarce zobaczymy:



Jak widać w powyższym przykładzie, autora wypowiedzi - Kubusia Puchatka - wyszczególniliśmy przy użyciu znacznika `<CITE>`, o którym dalej jeszcze powiemy, a który stosuje się właśnie do tego celu.

Możemy również tworzyć krótkie cytaty za pomocą znacznika `<Q>`. Znacznik ten działa tak, że na początku i końcu cytatu umieszcza cudzysłowy. Przykładowo, dla fragmentu kodu:

```
Steve powiedział, <Q>Relax baby...</Q>
```

zobaczymy tekst

```
Steve powiedział, ”Relax baby...”
```

Znaczniki `<Q>` możemy zagnieżdżać. Niektóre przeglądarki nie potrafią interpretować tego znacznika.

Tworzenie cytatów za pomocą opisanych znaczników jest niezalecane. Cytaty powinno tworzyć się za pomocą arkuszy stylów.

2.1.4. Preformatowany tekst (znacznik PRE)

Ciągi spacji w tekście traktowane są przez przeglądarkę jako pojedyncze odstępy. Jeżeli więc tekst w naszym dokumencie będzie wstępnie sformatowany w taki sposób:

```
<BODY>
  <P>
    int Liczba;
    if(Liczba % 2)
      printf("Liczba %d jest nieparzysta.", Liczba);
    else
      printf("Liczba %d jest parzysta.", Liczba);
  </P>
</BODY>
```

to w oknie przeglądarki i tak zobaczymy całkiem inne rozmieszczenie, ponieważ wszystkie ciągi spacji zostaną skrócone do pojedynczych odstępów:

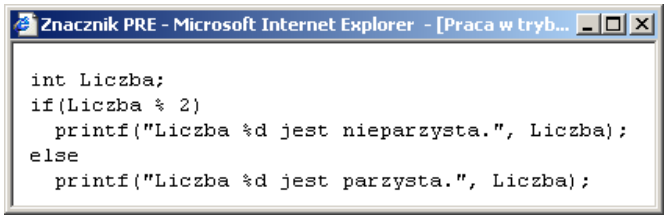
```
int Liczba; if(Liczba % 2) printf("Liczba %d jest nieparzysta.",
Liczba); else printf("Liczba %d jest parzysta.", Liczba);
```

Efekt jest nieprzyjemny szczególnie w sytuacjach, gdy tekst jest sformatowany, tak jak np. w źródłach programów. W takich sytuacjach stosujemy przeznaczony do tego celu znacznik <PRE>, który wstawiamy w miejsce znacznika akapitu.

Dokument nasz będzie zatem wyglądał tak:

```
<BODY>
  <PRE>
    int Liczba;
    if(Liczba % 2)
      printf("Liczba %d jest nieparzysta.", Liczba);
    else
      printf("Liczba %d jest parzysta.", Liczba);
  </PRE>
</BODY>
```

W oknie przeglądarki zobaczymy:



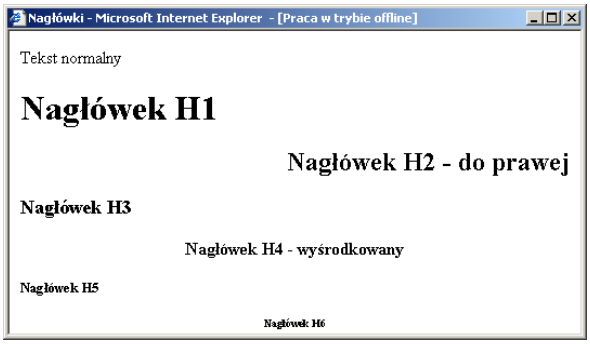
Stosowanie znacznika </PRE> jest obowiązkowe.

2.1.5. Nagłówki: H1, H2, H3, H4, H5, H6

W HTML wyróżniamy sześć poziomów nagłówków, od H1 do H6. Nagłówki wykorzystuje się na przykład do wyróżnienia tytułów lub podtytułów. Nagłówek poziomu pierwszego charakteryzuje się największym rozmiarem czcionki, a szóstego poziomu najmniejszym. Należy pamiętać o tym, że stosowanie znaczników końcowych </H1>, </H2>, </H3>, </H4>, </H5> i </H6> jest obowiązkowe.

```
<BODY>
  <P>Tekst normalny</P>
  <H1>Nagłówek H1</H1>
  <H2 align=”right”>Nagłówek H2 – do prawej</H2>
  <H3>Nagłówek H3</H3>
  <H4 align="center">Nagłówek H4 - wyśrodkowany</H4>
  <H5>Nagłówek H5</H5>
  <H6 style=”text-align: center”>Nagłówek H6</H6>
</BODY>
```

W oknie przeglądarki zobaczymy:



Podobnie jak w przypadku znacznika <P>, również tekst nagłówków możemy dodatkowo formatować, np. przy użyciu parametrów align i style. Możliwość tę wykorzystaliśmy w celu wyrównania nagłówka h2 do prawej strony i wycentrowania nagłówka H4 i H6.

2.1.6. Znaczniki wyróżniające tekst: EM, STRONG, DFN, CODE, SAMP, KBD, VAR, CITE, ABBR, ACRONYM, ADDRESS

Omawiane w tym punkcie znaczniki wykorzystywane są do zmiany kroju czcionki dla fragmentów tekstu w zależności od charakteru informacji. Stosowanie takich znaczników pozwala uczynić nasz dokument bardziej przejrzystym i wygodniejszym do czytania.

Znaczniki wyróżniające fragmenty tekstu zostały wyszczególnione poniżej:

- EM - pochylenie tekstu,
- STRONG - pogrubienie tekstu,
- CITE - wzmianki o autorze cytatu lub tytule,
- DFN - definicje,
- CODE - fragmenty źródeł programów,
- SAMP - dane programu, skrypty,
- KBD - dane przetwarzane (wprowadzane) przez użytkownika,
- VAR - zmienne, argumenty programu,
- ADDRESS - adresy,
- ABBR - skróty,
- ACRONYM - akronimy.

Wszystkie z wymienionych znaczników muszą być zakończone znacznikiem końcowym </EM>, </STRONG> itd.

```
<BODY>
  Pytania kierujcie na adres <ADDRESS>jn@hmmm.com</ADDRESS> lub
  pod numer <STRONG>555-323-343</STRONG>.
</BODY>
```

2.1.7. Znaczniki ustalające czcionkę dla tekstu: FONT i BASEFONT

Za pomocą znacznika <FONT> możemy definiować parametry wyświetlanego tekstu. Znacznik ten posiada kilka parametrów, za pomocą których możemy określać rodzaj czcionki, jej kolor oraz rozmiar.

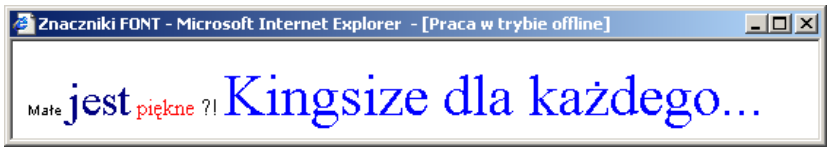
- face - Rodzaj czcionki, np. Arial, Courier, MS Sans Serif etc. Dopuszczalne jest podanie kilku nazw czcionek, rozdzielonych przecinkami, z których obowiązującym jest pierwszy znaleziony w systemie.
- color - Kolor czcionki, określony nazwą koloru lub jego reprezentacją szesnastkową, np. color="green" lub color="#008000".
- size - Rozmiar czcionki, określany przez podanie wartości z zakresu 1÷7, przy czym ustawieniem domyślnym jest 3. Dopuszczalne jest również

podawanie rozmiaru w odniesieniu do rozmiaru bieżącego przez umieszczenie przed liczbą znaku "+" lub "-".

Dodatkowo możemy stosować inne parametry, z czego najciekawszym jest `style`, za pomocą którego tworzymy jednowierszowe style. Taki właśnie sposób wpływania na właściwości czcionki jest zalecany.

```
<BODY>
   <FONT face="Arial" size="1">Małe</FONT>
   <FONT size="+3" color="navy"> jest</FONT>
   <FONT color="#FF0000"> piękne</FONT> ?!
   <FONT style="color: blue; font-size: 30pt;">
      Kingsize dla każdego...
  </FONT>
</BODY>
```

W oknie przeglądarki zobaczymy:



Za pomocą znacznika `<BASEFONT>` możemy zdefiniować domyślną czcionkę dla całego dokumentu. Podobnie jak w przypadku znacznika `<FONT>` również i tu możemy definiować atrybuty tekstu za pomocą takich samych parametrów.
Definicję domyślnej czcionki możemy umieścić zarówno w nagłówku dokumentu, jak również w jego ciele. Znaczniki nagłówków `<H1>`, ..., `<H6>` ignorują definicję domyślnej czcionki.

```
<basefont size="4" color="navy" face="Verdana">
<basefont style="font-family: verdana; font-size: 14pt">
```

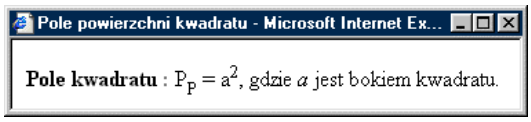
2.1.8. Znaczniki ustalające styl czcionki: TT, I, B, BIG, SMALL, STRIKE, S, U, SUB, SUP

Za pomocą dość okazałej grupy znaczników możemy definiować styl czcionki. Poniżej znajduje się opis znaczenia poszczególnych znaczników:

- `TT` - czcionka o stałej szerokości,
- `I` - czcionka pochylona (kursywa),
- `B` - czcionka pogrubiona,
- `BIG` - czcionka powiększona o jeden stopień,
- `SMALL` - czcionka pomniejszona o jeden stopień,
- `STRIKE` - czcionka przekreślona,
- `S` - jak `STRIKE`,
- `U` - czcionka nad linią,
- `SUB` - indeks dolny,
- `SUP` - indeks górny.

Wraz z wymienionymi wyżej znacznikami należy stosować znaczniki końca `</TT>`, `</I>`, `</B>`, `</BIG>`, `</SMALL>`, ... itd.

Dzięki przedstawionym znacznikom utworzenie prostego wzoru matematycznego nie stanowi już większego problemu.




```
<B>Pole kwadratu :</B>
P<SUB>P</SUB> = a<SUP>2</SUP>,
gdzie <I>a</I> jest bokiem
kwadratu.
```

Również z powyższymi znacznikami możliwe jest używanie bardzo wygodnego do definiowania stylu czcionki parametru `style`.

```
<b style="font-size: 30pt;">Wielki</b>
<b style="color: blue"> błękit</b>
```

2.2. Wyrównywanie elementów w dokumencie, linie poziome

2.2.1. Wyrównanie (parametr ALIGN)

Przy pomocy parametru `align` możemy wyrównywać bloki tekstów w dokumencie względem marginesów.
Parametr ten może występować wraz z wieloma znacznikami przyjmując jedną z poniższych wartości:

- `left` - wyrównanie do lewego marginesu,
- `right` - wyrównanie do prawego marginesu,
- `center` - wyśrodkowanie,
- `justify` - wyjustowanie do obydwu marginesów.

Żeby wyjaśnić zasadę stosowania parametru `align` posłużymy się do tego celu znacznikiem akapitu, z którym jest on najczęściej używany.

Na przykład wyrównanie akapitu do prawego marginesu będzie wyglądało tak:

```
<P align="right"> zawartość akapitu...
```

I jeszcze kilka przykładów wykorzystania parametru `align` :

```
<H3 align="left"> treść nagłówka...
<P align="justify"> zawartość akapitu...
<DIV align="center"> zawartość bloku...
```

Najciekawszą i bardzo wygodną metodą formatowania zawartości dokumentu jest metoda oparta na stylach:

```
<HTML>
  <HEAD>
    <TITLE>Wyrównywanie</TITLE>
    <STYLE type="text/css">
      P.akapitpr {text-align: right}
    </STYLE>
  </HEAD>
```

```
  <BODY>
     <P class="akapitpr">Treść akapitu</P>
  </BODY>
</HTML>
```

ale o tym więcej w dalszej części pracy.

2.2.2. Grupowanie elementów (znaczniki DIV i SPAN)

Bardzo przydatnym znacznikiem wykorzystywanym do wyrównywania fragmentów dokumentu jest `<DIV>`. Przy jego pomocy możemy wyrównywać względem marginesów kilka elementów jednocześnie przez ich zgrupowanie.

```
<BODY>
  <P align=”right”> tekst pierwszego akapitu...</P>
  <P align=”right”> tekst drugiego akapitu...</P>
  <P align=”right”> tekst trzeciego akapitu...</P>
</BODY>
```

Jak widzimy, każdy akapit wyrównywany jest do prawego marginesu. W każdym znaczniku `<P>` wpisano parametr `align=”right”`. Możemy jednak uprościć zapis używając znacznika `<DIV>`. Nasz dokument będzie wyglądał wtedy tak:

```
<BODY>
  <DIV align=”right”>
     <P>tekst pierwszego akapitu...</P>
     <P>tekst drugiego akapitu...</P>
     <P>tekst trzeciego akapitu...</P>
  </DIV>
</BODY>
```

Znacznik `<DIV>` wykorzystujemy do grupowania wieloelementowych bloków. Do grupowania jednoelementowych fragmentów dokumentu używamy znacznika `<SPAN>`.

Wartość tych znaczników szczególnie wzrasta, gdy połączymy je z arkuszami stylów.

Znacznik `<DIV>` i style:

```
<DIV style="color: red; font-family: arial; font-size: 14pt">
  <P>tekst pierwszego akapitu...</P>
  <P>tekst drugiego akapitu...</P>
</DIV>
```

Teraz użyjemy znacznika `<SPAN>` do zmiany koloru i rozmiaru czcionki środkowego wyrazu:

```
<P>Bardzo <SPAN style=”color: blue; font-size: 24pt”>ważna</SPAN>
   wiadomość</P>
```

I wszystko razem:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
  <HEAD>
    <TITLE>Grupowanie</TITLE>
```

```
    <STYLE type="text/css">
      .klasa { font-family: arial; font-size: 14pt }
      DIV { color: red }
    </STYLE>
  </HEAD>

  <BODY>
    <P>World <SPAN class="klasa">Wide</SPAN> Web</P>
    <DIV>
      <P>Piszę sobie jakiś tekst.</P>
      <P>Piszę sobie dalej jakiś tekst</P>
      <DIV class="klasa">
        <P>Kolejny wiersz tekstu.</P>
        <P>Kolejny <SPAN style="color: blue">wiersz</SPAN> tekstu.</P>
      </DIV>
      <P>Piszę sobie dalej jakiś tekst</P>
      <P>I to <SPAN style="font-size: 24pt; color: green">już
         ostatni wiersz</SPAN> tekstu.</P>
    </DIV>
  </BODY>
</HTML>
```

Stosowanie znaczników końca `</DIV>` i `</SPAN>` jest wymagane.

2.2.3. Linia pozioma (znacznik HR)

Znacznik `<HR>` odpowiedzialny za wyrysowywanie poziomej linii jest zaliczany do grupy znaczników wpływających na strukturyzację dokumentu. Dodatkowo powoduje on zakończenie bieżącego akapitu.

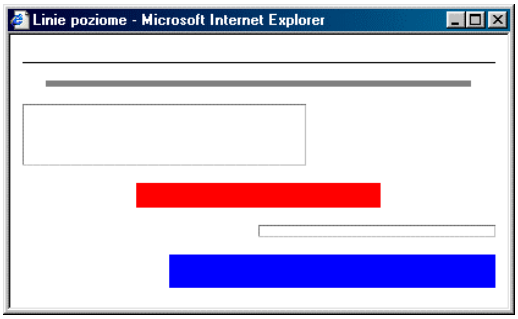
Znacznik ten posiada cztery główne parametry:

- `align` - Wyrównanie linii względem marginesów (`left`, `right`, `center`).
- `noshade` - Linia bez cieniowania.
- `size` - Wysokość linii wyrażona w pikselach.
- `width` - Szerokość linii wyrażona w pikselach lub procentach szerokości okna przeglądarki.

Z dodatkowych parametrów warto wspomnieć o wyróżnianym już `style`.

```
<HR noshade size="1" color="black">
<HR width="90%" size="5" color="#808080">
<HR align="left" size="50" width="60%">
<HR width="200" size="20" color="red">
<HR align="right" width="50%" size="10">

<HR style="text-align: right;
           color: blue;
           width: 200pt; height: 20pt;">
```



Stosowanie znacznika `</HR>` jest niezalecane.

2.3. Listy

Język HTML zawiera mechanizm tworzenia list i wyliczeń. Wśród nich możemy wyróżnić trzy rodzaje:

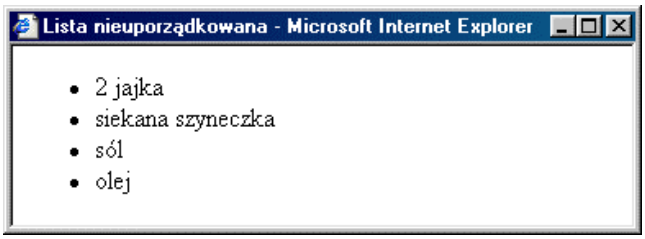
- listę nieuporządkowaną (wypunktowanie)
- listę uporządkowaną
- listę definicyjną

Zastosowanie list jest oczywiste, dlatego też od razu przejdziemy do ich szczegółowego scharakteryzowania.

2.3.1. Listy nieuporządkowane (UL), uporządkowane (OL) i elementy listy (LI)

Listę nieuporządkowaną tworzy się za pomocą znaczników `<UL>` i `</UL>`, z których pierwszy oznacza początek, a drugi koniec listy. Przed poszczególnymi składnikami listy stawia się znacznik `<LI>`.

Poniżej widzimy przykład prostej listy nieuporządkowanej:



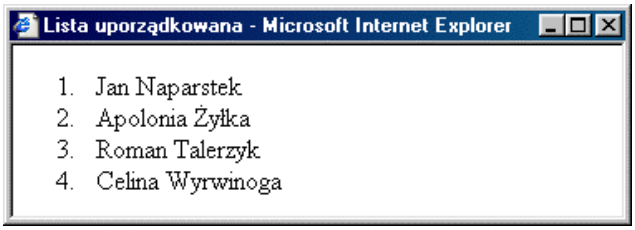
Przed składnikami listy przeglądarka automatycznie robi wcięcie i stawia symbol wyliczenia (w większości przeglądarek domyślnie jest to czarna kropka "•").
Definicja listy takiej jak wyżej w HTML będzie wyglądała następująco:

```
<UL>
   <LI>2 jajka</LI>
   <LI>siekana szyneczka</LI>
   <LI>sól</LI>
   <LI>olej</LI>
</UL>
```

Stosowanie znacznika `</UL>` jest obowiązkowe, natomiast `</LI>` jest opcjonalne.

W podobny sposób definiuje się listy uporządkowane. Tu jednak zamiast symboli wyliczenia, elementy listy są automatycznie ponumerowane.

Oto przykładowa lista uporządkowana:

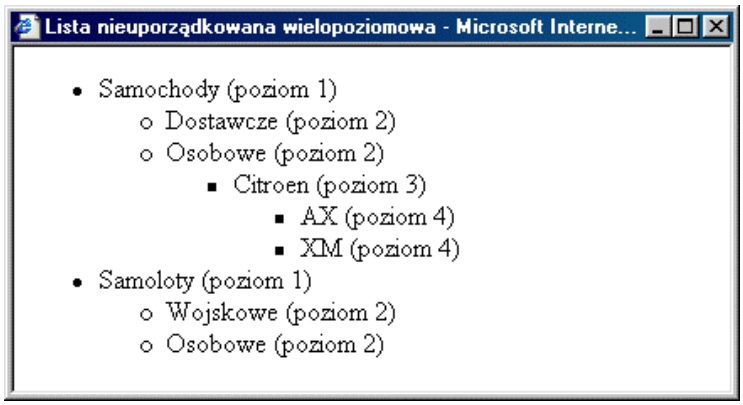


i jej definicja w HTML-u:

```
<OL>
  <LI>Jan Naparstek</LI>
  <LI>Apolonia Żyłka</LI>
  <LI>Roman Talerzyk</LI>
  <LI>Celina Wyrwinoga</LI>
</OL>
```

Jak widać w tym przypadku do określania początku i końca listy uporządkowanej używa się znaczników `<OL>` i `</OL>`. Stosowanie znacznika `</OL>` jest obowiązkowe.

Listy nieuporządkowane i uporządkowane mogą być wielopoziomowe. Oto przykład takiej listy:



W HTML-u powyższą listę definiujemy następująco:

```
<UL>
  <LI>Samochody (poziom 1)</LI>
  <UL>
    <LI>Dostawcze (poziom 2)</LI>
    <LI>Osobowe (poziom 2)</LI>
    <UL>
      <LI>Citroen (poziom 3)</LI>
      <UL>
        <LI>AX (poziom 4)</LI>
        <LI>XM (poziom 4)</LI>
      </UL>
    </UL>
  </UL>
  <LI>Samoloty (poziom 1)</LI>
  <UL>
    <LI>Wojskowe (poziom 2)</LI>
    <LI>Osobowe (poziom 2)</LI>
  </UL>
</UL>
```

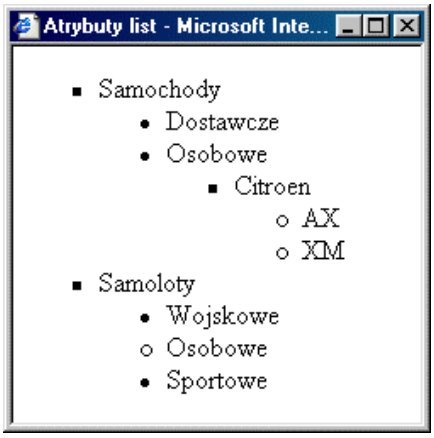
Definicja wielopoziomowej listy uporządkowanej będzie wyglądała niemal identycznie jak ta powyżej. Musimy jedynie znaczniki `<UL>` i `</UL>` zastąpić znacznikami `<OL>` i `</OL>`.

Opisane listy mogą być dowolnie zagnieżdżane.

Wyglądem list możemy sterować za pomocą kilku parametrów:

♦ Lista nieuporządkowana `<UL>`

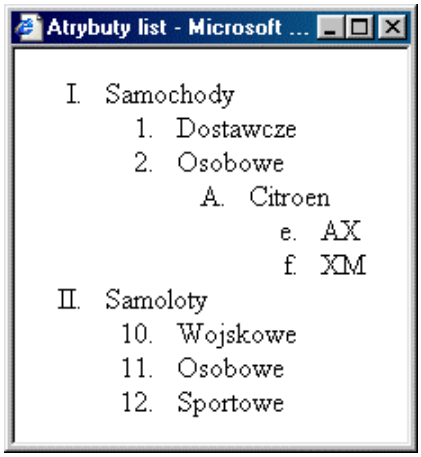
- `type` Określenie symbolu stawianego przed elementami listy:
  - `disc` wypełnione koło (symbol domyślny dla 1-go poziomu)
  - `circle` okrąg (symbol domyślny dla 2-go poziomu)
  - `square` wypełniony kwadrat (symbol domyślny dla 3-go poziomu)




```
<UL type="square">
  <LI>Samochody</LI>
  <UL type="disc">
    <LI>Dostawcze</LI>
    <LI>Osobowe</LI>
    <UL>
      <LI>Citroen</LI>
      <UL type="circle">
        <LI>AX</LI>
        <LI>XM</LI>
      </UL>
    </UL>
  </UL>
  <LI>Samoloty</LI>
  <UL type="disc">
    <LI>Wojskowe</LI>
    <LI type="circle">Osobowe</LI>
    <LI>Sportowe</LI>
  </UL>
</UL>
```

♦ Lista uporządkowana `<OL>`

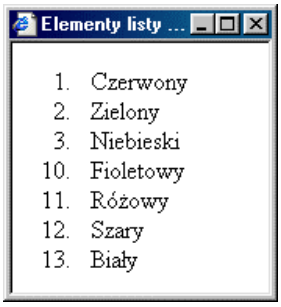
- `type` Określenie symbolu numerowania elementów listy:
  - `1` cyfry arabskie, 1, 2, 3, ...
  - `a` małe litery, a, b, c, ...
  - `A` wielkie litery, A, B, C, ...
  - `i` małe cyfry rzymskie, i, ii, iii, ...
  - `I` wielkie cyfry rzymskie, I, II, III, ...
- `start` Rozpoczęcie numerowania elementów listy począwszy od podanej wartości.




```
<OL type="I">
  <LI>Samochody</LI>
  <OL>
    <LI>Dostawcze</LI>
    <LI>Osobowe</LI>
    <OL type="A">
      <LI>Citroen</LI>
      <OL type="a" start="5">
        <LI>AX</LI>
        <LI>XM</LI>
      </OL>
    </OL>
  </OL>
  <LI>Samoloty</LI>
  <OL type="1" start="10">
    <LI>Wojskowe</LI>
    <LI>Osobowe</LI>
    <LI>Sportowe</LI>
  </OL>
</OL>
```

- Element list `<LI>`

- `value` - numerowanie następnych elementów listy od podanej wartości,
- `compact` - zmniejszenie odstępu między liniami w każdym elemencie.

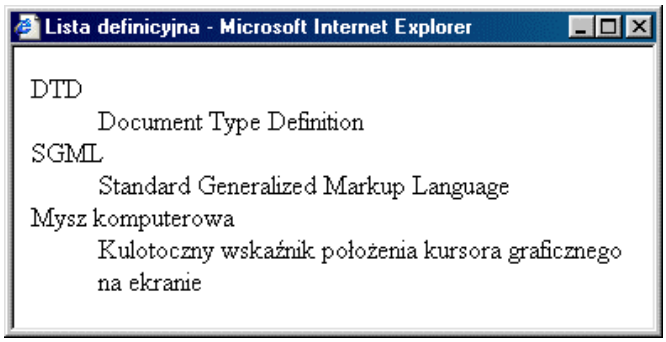



```
<OL>
  <LI>Czerwony</LI>
  <LI>Zielony</LI>
  <LI>Niebieski</LI>
  <LI value="10">Fioletowy</LI>
  <LI>Różowy</LI>
  <LI>Szary</LI>
  <LI>Biały</LI>
</OL>
```

Również w przypadku znaczników dot. list możemy, a nawet powinniśmy, stosować parametr `style`.

2.3.2. Listy definicyjne: znaczniki DL, DT i DD

Lista definicyjna wykorzystywana jest najczęściej do tworzenia definicji, opisów słownikowych lub encyklopedycznych. Listę taką umieszcza się między znacznikami `<DL>` i `</DL>`. Każdy element listy definicyjnej składa się z dwóch części: hasła `<DT>` i opisu `<DD>`. Oto przykład takiej listy:



i w HTML-u:

```
<DL>
  <DT>DTD
  <DD>Document Type Definition
  <DT>SGML
  <DD>Standard Generalized Markup Language
  <DT>Mysz komputerowa
  <DD>Kulotoczny wskaźnik położenia kursora graficznego na ekranie
</DL>
```

Stosowanie znacznika `</DL>` jest obowiązkowe, natomiast znaczniki `</DT>` i `</DD>` są opcjonalne.

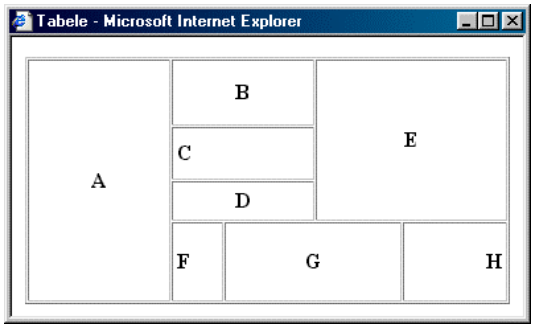
W języku HTML występują dodatkowo dwa znaczniki, za pomocą których możemy tworzyć listy. Są to `<MENU>` i `<DIR>`. Jednak w praktyce zamiast tych znaczników stosuje się `<UL>`.

```
<DIR>
  <LI>URI</LI>
  <LI>URL</LI>
  <LI>URN</LI>
</DIR>
```

```
<MENU>
  <LI>HTML</LI>
  <LI>XML</LI>
  <LI>XHTML</LI>
</MENU>
```

2.4. Tabele

Język HTML daje możliwość definiowania tabel. Poza typowym ich zastosowaniem, tabele znakomicie nadają się do formatowania wyglądu ekranu. W komórkach tabeli możemy umieszczać dowolne obiekty, np. tekst, grafikę, odsyłacze, formularze, etc. Tabele możemy zagnieżdżać.



Podobnie jak w edytorach tekstu, komórki możemy scalać zarówno w pionie, jak i w poziomie.

2.4.1. Znacznik definiujący tabelę TABLE

Tabelę tworzy się przez umieszczenie jej definicji między znacznikami `<TABLE>` i `</TABLE>`. Wraz ze znacznikiem `<TABLE>` możemy stosować m.in. parametry:

- `border` - grubość ramki otaczającej komórki, podawana w pikselach, np. `border="3"`. Komórki puste nie są otaczane ramką. Jeżeli chcemy ukryć obramowanie tabeli, musimy parametrowi `border` przypisać wartość `0`,
- `width` - szerokość tabeli, podawana w procentach szerokości okna (lub pikselach), np. `width="90%"`,
- `height` - wysokość tabeli, podawana w procentach wysokości okna (lub pikselach), np. `height="50%"`,
- `cellspacing` - odległość między komórkami tabeli, podawana w pikselach (wartością domyślną jest `cellspacing="2"`),
- `cellpadding` - odległość zawartości komórki od jej obramowania, podawana w pikselach (wartością domyślną jest `cellpadding="1"`),
- `align` - wyrównanie tabeli w oknie (`left`, `right` i `center`), np. `align="center"`,
- `bgcolor` - kolor tła dla wszystkich komórek tabeli, określony nazwą koloru lub jego reprezentacją szesnastkową, np. `bgcolor="#CDB6F1"`,
- `background` - plik z grafiką, która będzie tłem dla wszystkich komórek, np. `background="marmurek.gif"`,

- `hspace` - wstawia odstęp o zadanej szerokości po obu stronach tabeli, np. `hspace="10"`,
- `vspace` - wstawia odstęp o zadanej wysokości nad i pod tabelą, np. `vspace="20"`,
- `summary` - opis zawartości tabeli wyświetlany przez niewizualne przeglądarki, np. `summary="Tabela zawiera dane ..."`,
- `frame` - informacja o układzie zewnętrznego obramowania tabeli. Parametr może przyjąć jedną z następujących wartości:
  - `void` - brak obramowania (wartość domyślna),
  - `above` - obramowanie występuje tylko u góry,
  - `below` - obramowanie występuje tylko u dołu,
  - `hsides` - obramowanie występuje tylko na krawędziach poziomych,
  - `vsides` - obramowanie występuje tylko na krawędziach pionowych,
  - `lhs` - obramowanie występuje tylko po lewej stronie,
  - `rhs` - obramowanie występuje tylko po prawej stronie,
  - `box` - obramowanie występuje na wszystkich krawędziach,
  - `border` - obramowanie występuje na wszystkich krawędziach,
- `rules` - informacja o układzie linii rozdzielających komórki tabeli. Parametr może przyjąć jedną z następujących wartości:
  - `none` - brak linii (wartość domyślna),
  - `groups` - linie umieszczone są pomiędzy grupami wierszy (`thead`, `foot`, `tbody`) i kolumn (`colgroup`),
  - `rows` - linie umieszczone są pomiędzy wierszami,
  - `cols` - linie umieszczone są pomiędzy kolumnami,
  - `all` - linie umieszczone są pomiędzy kolumnami i wierszami.

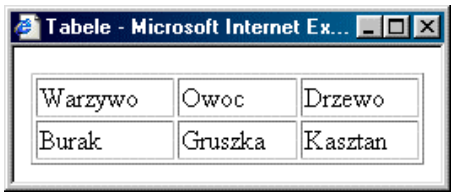
Przykładowa definicja tabeli:

```
<TABLE border="5" width="100%" bgcolor="gray" cellspacing="3">
   ... definicja tabeli ...
</TABLE>
```

Stosowanie znacznika `</TABLE>` jest obowiązkowe.

2.4.2. Znaczniki TR, TD i TH

Wiersze w tabeli definiuje się za pomocą znacznika `<TR>`, natomiast komórki w wierszu za pomocą `<TD>`.




```
<table border="1" width="90%">
  <tr>
    <td>Warzywo</td>
    <td>Owoc</td>
    <td>Drzewo</td>
  <tr>
    <td>Burak</td>
    <td>Gruszka</td>
    <td>Kasztan</td>
</table>
```

Jak widać na powyższym przykładzie, stosowanie znacznika `</TR>` nie jest obowiązkowe. Również umieszczanie znacznika `</TD>` jest opcjonalne.

Nagłówki kolumn tworzy się używając znacznika <TH>. Znacznik ten jest praktycznie taki sam jak <TD>. Różnica dotyczy tekstów, które w komórkach nagłówkowych domyślnie drukowane są pogrubioną czcionką.

```
<TABLE border="1" width="100%">
  <TR>
    <TH>Dane techniczne / Model</TH>
    <TH>1,4 T.Spark 16V</TH>
    <TH>1,6 T.Spark 16V</TH>
  <TR>
    <TD>Pojemność (cm<sup>3</sup>)</TD>
    <TD>1370</TD>
    <TD>1568</TD>
  <TR>
    <TD>Moc max. KM (kW) przy obr./min.</TD>
    <TD>103(76) 6300</TD>
    <TD>120(88) 6300</TD>
</TABLE>
```

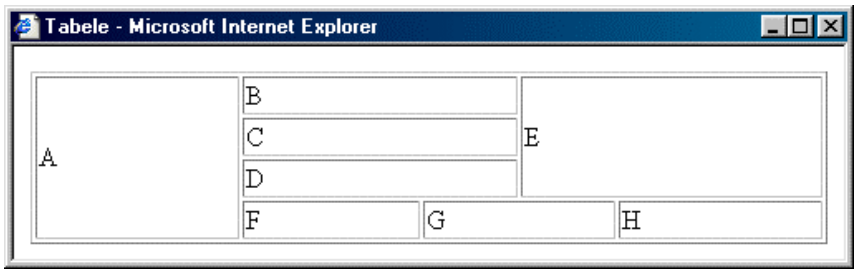
Stosowanie znacznika </TH> jest opcjonalne.

Do najważniejszych parametrów znacznika <TR> należą:

- `align` - wyrównanie poziome tekstu w komórkach bieżącego wiersza (`left`, `right` i `center`),
- `valign` - wyrównanie pionowe tekstu w komórkach bieżącego wiersza (`top`, `middle`, `bottom` i `baseline`),
- `bgcolor` - kolor tła bieżącego wiersza, określony nazwą koloru lub jego reprezentacją szesnastkową, np. `bgcolor="navy"`,
- `style` - jednowierszowy styl dla dokumentu.

Do najważniejszych parametrów znacznika <TD> i <TH> należą:

- `align` - wyrównanie poziome tekstu w komórce (`left`, `right` i `center`),
- `valign` - wyrównanie pionowe tekstu w komórce (`top`, `middle`, `bottom` i `baseline`),
- `colspan` - połączenie ze sobą określonej liczby komórek w poziomie znajdujących się w jednym wierszu, np. `colspan=2`,
- `rowspan` - połączenie ze sobą określonej liczby komórek w pionie znajdujących się w jednej kolumnie, np. `rowspan=3`,
- `bgcolor` - kolor tła bieżącego wiersza, określony nazwą koloru lub jego reprezentacją szesnastkową, np. `bgcolor="navy"`,
- `background` - nazwa pliku z rysunkiem będącego tłem dla komórki (URI),
- `width` - szerokość komórki wyrażona w pikselach lub jako procent szerokości całej tabeli, np. `width="60%"`,
- `height` - wysokość komórki wyrażona w pikselach lub jako procent wysokości całej tabeli, np. `height="20"`,
- `nowrap` - wyłączenie możliwości łamania wierszy dla bieżącej komórki,
- `abbr` - skrócony opis zawartości komórki,
- `scope` - określenie grupy komórek, dla których ma zastosowanie bieżąca komórka nagłówka (`row`, `col`, `rowgroup` i `colgroup`),
- `style` - jednowierszowy styl dla dokumentu.

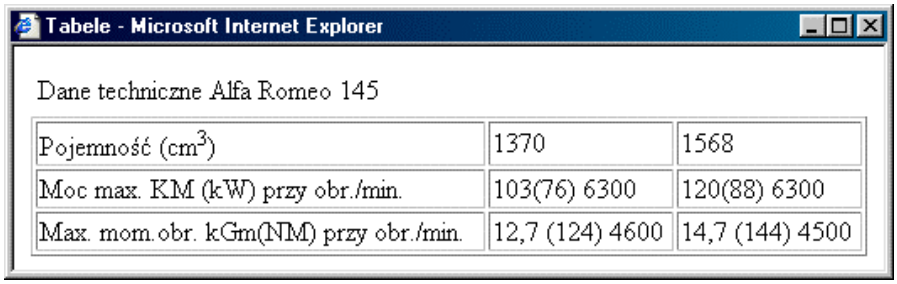



```
<TABLE border="1" width="100%">
  <TR>
    <TD rowspan="4">A</td>
    <TD colspan="2">B</td>
    <TD colspan="2" rowspan="3">E</td>
  <TR>
    <TD colspan="2">C</td>
  <TR>
    <TD colspan="2">D</td>
  <TR>
    <TD>F</td>
    <TD colspan="2">G</td>
    <TD>H</td>
</TABLE>
```

2.4.3. Nagłówek tabeli (znacznik CAPTION)

Tabele możemy podpisywać za pomocą znaczników `<CAPTION>` i `</CAPTION>`. Domyślnie podpis taki umieszczany jest centralnie nad tabelą. Przy użyciu parametru `align` możemy jednak samodzielnie określić sposób rozmieszczenia podpisu względem tabeli. Parametr `align` może przyjąć jedną z wartości:

- `top` - nad tabelą,
- `bottom` - pod tabelą,
- `left` - nad tabelą z lewej strony,
- `right` - nad tabelą z prawej strony.




```
<TABLE border="1" width="100%">
  <CAPTION align="left">Dane techniczne Alfa Romeo 145</CAPTION>
  <TR>
    <TD>Pojemność (cm3)</TD>
    <TD>1370</TD>
    <TD>1568</TD>
  <TR>
    <TD>Moc max. KM (kW) przy obr./min.</TD>
    <TD>103(76) 6300</TD>
    <TD>120(88) 6300</TD>
   <TR>
```

```
    <TD>Max. mom.obr. kGm(NM) przy obr./min.</TD>
    <TD>12,7 (124) 4600</TD>
    <TD>14,7 (144) 4500</TD>
</TABLE>
```

Stosowanie znacznika </CAPTION> jest obowiązkowe.

2.4.4. Grupowanie komórek (znaczniki THEAD, TBODY i TFOOT)

Grupowanie komórek (wierszy) polega na tworzeniu pewnych bloków, którym możemy następnie przypisywać pewne wspólne właściwości. Ogólnie jest stosowany podział na trzy bloki: nagłówek, ciało i stopkę tabeli. Poszczególne bloki opisywane są znacznikami:

- <THEAD> ... </THEAD> - nagłówek tabeli,
- <TBODY> ... </TBODY> - ciało tabeli,
- <TFOOT> ... </TFOOT> - stopka tabeli.

```
<TABLE border="1" width="100%">
<THEAD>
  <TR>
    <TD>A1...</TD>
    <TD>A2...</TD>
<TBODY style="color: blue;" >
  <TR>
    <TD>B1...</TD>
    <TD>B2...</TD>
  <TR>
    <TD>B11...</TD>
    <TD>B12...</TD>
<TFOOT align="center">
  <TR>
    <TD>C1...</TD>
    <TD>C2...</TD>
</TABLE>
```

Stosowanie znaczników </THEAD>, </TBODY> i </TFOOT> nie jest obowiązkowe.

Należy dodać, że grupowanie komórek odnosi się głównie do MS Internet Explorera i nie jest akceptowane przez niektóre przeglądarki.

2.4.5. Grupowanie kolumn: COLGROUP, COL

Za pomocą znacznika <COLGROUP> możemy zdefiniować właściwości dla jednej lub pewnej grupy kolumn. Właściwości każdej kolumny opisuje jeden znacznik <COLGROUP>. Możemy jednak za pomocą parametru span, którego wartością jest liczba określająca ilość kolumn, przypisać te same właściwości dla kilku kolumn jednocześnie.

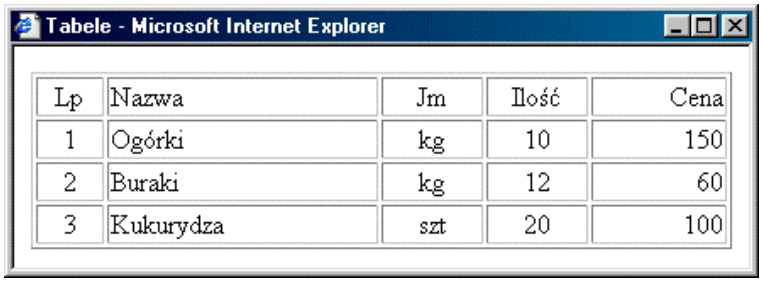
Załóżmy, że mamy tabelę posiadającą cztery kolumny. Chcemy, aby zawartość komórek w pierwszych trzech kolumnach tabeli była wyśrodkowana, natomiast zawartość czwartej kolumny wyrównana do prawej. Używając do tego celu znaczników <COLGROUP> definicja naszej tabeli będzie wyglądała tak:

```
<TABLE border="1" width="100%">
  <COLGROUP align="center" span="3">
  <COLGROUP align="right">

  <TR><TD>... <TD>... <TD>... <TD>...
  <TR><TD>... <TD>... <TD>... <TD>...
  <TR><TD>... <TD>... <TD>... <TD>...
  <TR><TD>... <TD>... <TD>... <TD>...
</TABLE>
```

Wiersz `<COLGROUP align="center" span="3">` możemy oczywiście zastąpić trzema wierszami `<COLGROUP align="center">`, ale czy warto?

Właściwości kolumn możemy również ustalać (zmieniać) wewnątrz definicji grupy kolumn `<COLGROUP>`. Dokonujemy tego przy pomocy znacznika `<COL>`. Znacznik ten umieszczony w definicji grupy po znaczniku `<COLGROUP>` anuluje jego działanie.



Lp	Nazwa	Jm	Ilość	Cena
1	Ogórki	kg	10	150
2	Buraki	kg	12	60
3	Kukurydza	szt	20	100

```
<TABLE border="1" width="100%">

<COLGROUP align="center" span="4">
  <COL width="10%">
  <COL align="left" width="40%">
  <COL width="15%" span="2">
</COLGROUP>
<COLGROUP align="right" width="20%">

  <TR>
    <TD>Lp<TD>Nazwa<TD>Jm<TD>Ilość<TD>Cena
  <TR>
    <TD>1<TD>Ogórki<TD>kg<TD>10<TD>150
  <TR>
    <TD>2<TD>Buraki<TD>kg<TD>12<TD>60
  <TR>
    <TD>3<TD>Kukurydza<TD>szt<TD>20<TD>100
</TABLE>
```

Stosowanie znacznika `</COLGROUP>` jest opcjonalne, natomiast `</COL>` jest niezalecane.

2.5. Odsyłacze hipertekstowe

W tym punkcie zajmiemy się omówieniem prostego, a zarazem bardzo pożytecznego mechanizmu, który powoduje, że nasz dokument może stać się w pełni interaktywny. Mowa o odsyłaczach hipertekstowych, z którymi każdy z pewnością już się spotkał. Odsyłaczem hipertekstowym jest fragment wyróżnionego tekstu lub grafiki, któremu

przypisany jest nowy adres internetowy lub miejsce na stronie. Po wybraniu takiego odsyłacza, np. przez kliknięcie na nim myszą, następuje załadowanie dokumentu, którego adres został przypisany temu odsyłaczowi lub wykonywany jest skok do podanego miejsca na stronie.
Dzięki odsyłaczom poruszanie się po stronie staje się przyjemne, intuicyjne i niejednokrotnie zaoszczędza nam dużo czasu podczas szukania konkretnych informacji. Ponadto dzięki „linkom" pewne elementy naszego dokumentu, np. mapy czy przyciski graficzne, zdecydowanie poprawiają atrakcyjność dokumentu oraz podnoszą jego funkcjonalność. O takim wykorzystaniu odsyłaczy hipertekstowych powiemy szerzej w dalszej części pracy.

2.5.1. Znacznik A

Odsyłacze hipertekstowe tworzy się przez umieszczenie elementu (fragment teksu, grafiki, etc.) pomiędzy znacznikami `<A>` i `</A>`, a którego kliknięcie będzie przenosiło nas pod wskazany adres.

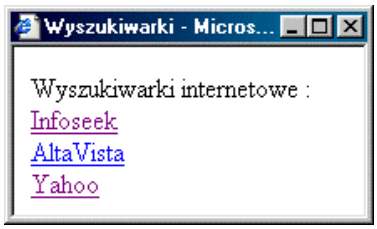
```
<A href="adres_celu"> element_do_kliknięcia </A>
```

Najważniejszymi parametrami znacznika `<A>` są m.in.:

- `href` - zdefiniowanie celu skoku dla odsyłacza hipertekstowego, np. `href="http://www.uninet.pl/artac/artac.html"` ,
- `name` - utworzenie „kotwicy", tj. miejsca, do którego możliwe będą skoki z odsyłacza (w obrębie tego samego dokumentu), np. `name="spis treści"`,
- `target` - określenie nazwy ramki, do której będzie załadowany dokument określony w `href`, przy czym nazwa ramki może być dowolnym tekstem lub jedną z predefiniowanych nazw:
  - `_blank` - nowe okno bez nazwy,
  - `_self` - aktywna ramka (okno),
  - `_top` - cały obszar roboczy przeglądarki (zignorowanie ramek),
  - `_parent` - ramka (okno) znajdująca się w kolejnym wyższym, poziomie zagnieżdżenia,

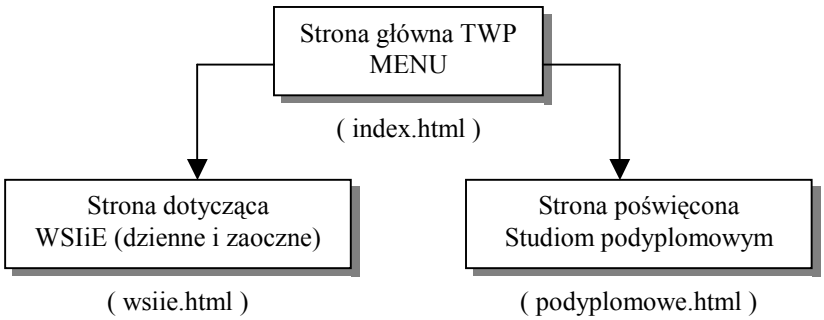
  np. `<A href="http://www.twp.olsztyn.pl" target="_top"> ...`
- `title` - opisanie obiektu wskazywanego przez `href` (stosowany wyłącznie do celów informacyjnych), np. `<A href="http://www.w3.org" title="Strona W3C"> ...`
- `rel` - opisanie relacji od `href` do kotwicy (stosowany wyłącznie do celów informacyjnych),
- `rev` - opisanie relacji od kotwicy do `href` (stosowany wyłącznie do celów informacyjnych),
- `link` - kolor odsyłacza, który nie był dotychczas odwiedzony,
- `alink` - kolor aktywnego odsyłacza (podczas trzymania na nim wciśniętego przycisku myszy),
- `vlink` - kolor odwiedzonego wcześniej odsyłacza, np. `<A href="#poz1" link="blue" alink="red" vlink="#005500">`.

Na początek prosty przykład pokazujący odsyłacze do stron umieszczonych na odległych serwerach:




```
<HTML>
<HEAD><TITLE>Wyszukiwarki</TITLE></HEAD>
<BODY>
 <P>Wyszukiwarki internetowe :<BR>
 <A href="http://www.infoseek.com">Infoseek</A>
 <BR>
 <A href="http://www.altavista.com">AltaVista</A>
 <BR>
 <A href="http://www.yahoo.com">Yahoo</A>
 </P>
</BODY>
</HTML>
```

Tak jak w powyższym przykładzie widać, adres celu musi być adresem bezwzględnym. Kliknięcie na jednej z nazw serwisów, np. na napisie *Infoseek* spowoduje odwołanie się do strony o adresie `www.infoseek.com`, a tym samym wczytanie jej do okna naszej przeglądarki.
Za pomocą znacznika `<A>` możemy również tworzyć odwołania hipertekstowe do plików należących do tej samej strony WWW (znajdujących się na tym samym serwerze). Przyjmijmy, że nasza strona posiada taką budowę:



Strona główna TWP (pełniąca dodatkowo funkcję MENU) wczytuje się jako pierwsza, a z niej dopiero możemy przejść do podstrony poświęconej studiom dziennym i zaocznym na WSIiE lub studiom podyplomowym.
Strona główna TWP (`index.html`) będzie poza różnymi dodatkowymi elementami posiadała dwa odsyłacze hipertekstowe do plików `wsiie.html` i `podyplomowe.html` (podstron) zdefiniowane przykładowo tak:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
  <TITLE>TWP w Olsztynie</TITLE>
</HEAD>
<BODY>
  ... inne elementy strony ...
  <P><STRONG>TWP w Olsztynie</STRONG></P>
  <P>
     <A href="wsiie.htm">Studia dzienne i zaoczne</A><BR>
     <A href="podyplomowe.html">Studia podyplomowe</A>
  </P>
  ... inne elementy strony ...
</BODY>
</HTML>
```

W sytuacji takiej jak ta powyżej, do opisywania celu możemy używać adresów względnych (nie zawierających domeny serwera), ponieważ zakładamy, że dokumenty znajdują się na tym samym serwerze.

Wspomnieliśmy wcześniej również o tym, że odsyłacze hipertekstowe mogą odwoływać się do jakiegoś miejsca na stronie. W takiej sytuacji celem dla naszego odsyłacza jest tzw. **kotwica**, którą definiujemy w naszym dokumencie według schematu:

```
<A name="nazwa"></A>
```

Definicja odsyłacza hipertekstowego w tym przypadku będzie taka:

```
<A href="#nazwa"> ...element... </A>
```

Przykładem może być szybki powrót (skok) z końca dokumentu do jego początku.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
  <TITLE>Ryby</TITLE>
</HEAD>
<BODY>
  <A name="lista"></A>
  <P><STRONG>Ryby słodkowodne</STRONG>
  <P>
    Szczupak<BR>
    Okoń<BR>
    Sandacz<BR>
    Karp<BR>
    Lin<BR>
    Leszcz<BR>
    Płoć<BR>
    Karaś<BR>
    ... cd. listy ...
  <P><A href="#lista">Początek listy</A>
</BODY>
</HTML>
```

Kotwic w dokumencie może być wiele, należy jedynie pamiętać o tym, że wielkość liter w nazwach kotwic nie jest rozróżniana. Z punktu widzenia języka HTML definicje:

```
<A name="IDENTYFIKATOR"></A>
<A name="identyfikator"></A>
```

są identyczne.

Do tej pory mówiąc o odsyłaczach hipertekstowych ograniczaliśmy się jedynie do dokumentów HTML. W rzeczywistości „linki” mogą wskazywać również na pliki innego typu. Na przykład poprzez naszą stronę WWW uzbrojoną w odpowiednie odsyłacze możemy udostępnić pliki z programami, grafikami, dźwiękiem, etc.

```
<BODY>
 <P>Mapa Olsztyna</P>
 <P>
  <A href="mapa.jpg">Plik graficzny wczytywany do okna
                     przeglądarki</A><BR>
  <A href="mapa.zip">Skompresowana mapa do pobrania</A>
 </P>
</BODY>
```

Na zakończenie przedstawiamy sytuację, w której odsyłacz hipertekstowy spowoduje wysłanie wiadomości pod wskazany adres pocztowy, a dokładniej spowoduje wyświetlenie czystego okna programu pocztowego z wpisanym adresem docelowym:

```
<A href="mailto:jan.kowalski@olsztyn.mail.pl">Napisz do mnie...</A>
```

2.6. Obrazy, grafika, mapy

Współcześnie niemal wszystkie tworzone strony WWW zawierają elementy graficzne. Są nimi wykresy, schematy, zdjęcia czy wreszcie grafiki wpływające na atrakcyjność wizualną naszego dokumentu.

2.6.1. Wstawianie grafiki (znacznik IMG)

Za pomocą znacznika `<IMG>` w naszym dokumencie możemy osadzić dowolną grafikę zapisaną w pliku graficznym, której format jest rozpoznawany przez przeglądarkę. Za standard plików graficznych używanych na stronach WWW przyjęto pliki zapisane w formacie GIF, JPEG i PNG.

```
<IMG src="adres_pliku_z_grafiką">
```

Znacznik `<IMG>` posiada szereg parametrów, z których najważniejsze przedstawiamy poniżej:

- `src` - adres URI pliku z grafiką wstawianą do dokumentu,
  np. `<IMG src="http://www.uninet.pl/artac/acpic202.gif">`
- `alt` - tekst wyświetlany w trakcie ładowania grafiki, braku możliwości jej wyświetlenia lub po najechaniu kursorem myszy (na grafikę),
  np. `<IMG src="mapa.gif" alt="Mapa miasta Olsztyn">`
- `border` - grubość ramki otaczającej rysunek (grafikę) wyrażana w pikselach,
  np. `<IMG src="rys1.jpg" border=1>`, przy czym `border=0` oznacza brak ramki,
- `height` - wysokość obrazka w pikselach, przy czym jeżeli nie określono szerokości za pomocą `width`, obrazek jest automatycznie skalowany przy jednoczesnym zachowaniu proporcji,
- `width` - szerokość obrazka w pikselach, przy czym jeżeli nie określono wysokości za pomocą `height`, obrazek jest automatycznie skalowany przy jednoczesnym zachowaniu proporcji,
  np. `<IMG src="ryba.gif" height="100" width="200">`
- `align` - wyrównywanie grafiki, gdzie `align` może przyjmować m.in. następujące wartości:
  - `top` - jedna linia tekstu wypisana u góry grafiki, reszta pod nią,
  - `middle` - jedna linia tekstu wypisana pośrodku grafiki, reszta pod nią,
  - `bottom` - jedna linia tekstu wypisana u dołu grafiki, reszta pod nią,
  - `left` - tekst będzie oblewał grafikę z prawej strony,
  - `right` - tekst będzie oblewał grafikę z lewej strony,

  np. `<IMG src="foto.jpg" align="left">`
- `hspace` - odstęp pomiędzy grafiką a pozostałą treścią dokumentu w poziomie,

- `vspace` - odstęp pomiędzy grafiką a pozostałą treścią dokumentu w pionie, np. `<IMG src="foto.jpg" align="left" hspace="5" vspace="5">`
- `ismap` - grafika będzie opisywała mapę graficzną,
- `usemap` - grafika będzie opisywała mapę graficzną o podanej nazwie, (więcej na ten temat w dalszej części).

Stosowanie znacznika `</IMG>` nie jest zalecane.

Warto pamiętać o tym, że animację zapisaną w formacie GIF (animowany GIF) również wstawimy do naszego dokumentu za pomocą znacznika `<IMG>`. Bardzo często większe grafiki dzieli się na kilka mniejszych części, dzięki czemu w prosty sposób możemy przypisywać im różne akcje (np. menu graficzne),

```
<BODY>
<IMG src="start.gif" width="68" height="58">
<A href="mailto:apolonia@poczta.pl">
  <IMG src="op_0.gif" alt="E-mail" border="0" width="88" height="58"></A>
<A href="menu1.html">
  <IMG src="op_1.gif" alt="Opcja 1" border="0" width="84" height="58"></A>
<A href="menu2.html">
  <IMG src="op_2.gif" alt="Opcja 2" border="0" width="110" height="58"></A>
</BODY>
```

Jak wynika z powyższego kodu, kliknięcie na grafice `op_0.gif` spowoduje uruchomienie programu pocztowego dającego możliwość wysłania listu na adres `apolonia@olsztyn.mail.pl`. Z kolei wybranie grafiki `op_1.gif` lub `op_2.gif` wczyta do okna przeglądarki odpowiednio dokument `menu1.html` lub `menu2.html`.

Ciekawymi parametrami opisującymi zdarzenia, które możemy dodatkowo stosować ze znacznikiem `<IMG>` są m.in.: `onclick`, `onmouseover` i `onmouseout`. Pierwszy spowoduje wywołanie odpowiedniego skryptu, w momencie kliknięcia myszą na obrazku, np.:

```
<IMG src="info.png" onclick="alert('Autor: Jan Naparstek')">
```

Parametr `onmouseover` spowoduje wywołanie odpowiedniego skrypt w momencie, gdy najedziemy na obrazek kursorem myszy. Z kolei `onmouseout` wywoła skrypt, gdy wyjdziemy kursorem myszy poza obrazek. Te dwa parametry są często wykorzystywane do tworzenia animowanych przycisków czy menu.

```
<A href="strona2.html">
<IMG src="start.gif" border="0" name="btn" onmouseover="isover()"
     onmouseout="isout()">
</A>
```

2.6.2. Mapy graficzne (znaczniki MAP i AREA)

Czym jest właściwie graficzna mapa odsyłaczy ?
Jest to zwykły obrazek (grafika), na którym zdefiniowane są pewne obszary będące zwykłymi odsyłaczami hipertekstowymi, reagującymi na kliknięcia myszą. Wyjaśnimy to na przykładzie poniższej grafiki:

Na rysunku tym zdefiniowaliśmy trzy obszary, po wybraniu których załadowane zostaną odpowiednie dokumenty. Obszary te to: prostokąt (wyraz „DOWNLOAD"), okrąg (okrągły przycisk z jabłuszkiem) i wielokąt o kształcie elipsy („nadgryziona" elipsa z napisem „WSIIE ON-LINE" bez fragmentu przykrytego przez przycisk z jabłkiem). Reszta to ozdobniki.

Jak utworzyć mapę graficzną?
Definicję całej mapy odsyłaczy rozpoczynamy od nadania jej nazwy:

```
<MAP name="nazwa_mapy">
```

Następnie określamy obszary odsyłaczy. Do tego celu przeznaczony jest znacznik `<AREA>`, który posiada następujące parametry:

- `shape` - kształt obszaru:
  - `rect` - prostokąt,
  - `circle` - koło,
  - `poly` - wielobok,

  np. `<AREA shape="rect" ...>`
- `coords` - współrzędne punktów wyznaczających figurę (obszar),
  - `rect` - `coords="x1,y1,x2,y2"` – (x1,y1) współrzędne lewego górnego, (x2,y2) prawego dolnego rogu prostokąta,
  - `circle` - `coords="x,y,r"` – (x,y) współrzędne środka okręgu, r jego promień,
  - `poly` - `coords="x1,y1,x2,y2,x3,y3,..."` – (x1,y1), (x2,y2), (x3,y3), ..., (xN,yN) współrzędne kolejnych punktów,

  np. `<AREA shape="rect" coords="10,10,200,150">`
- `href` - adres URI odsyłacza lub kotwicy,
- `nohref` - określenie obszaru wewnątrz mapy, który nie będzie reagował na kliknięcia myszą,
- `alt` - tekst wyświetlany pod kursorem myszy, gdy staniemy nim w obszarze odsyłacza,
- `target` - określenie nazwy ramki, do której będzie załadowany dokument określony w `href`.

Po określeniu wszystkich obszarów odsyłaczy kończymy definicję mapy odsyłaczy znacznikiem `</MAP>`.

Teraz już tylko pozostaje nam wstawienie rysunku (grafiki) do dokumentu, któremu przypiszemy definicję mapy. Zrobimy to przy pomocy znacznika <IMG> z parametrem usemap :

```
<IMG usemap="nazwa_mapy" src="URL_obrazka">
```

A oto fragment w HTML tworzący mapę odsyłaczy na rysunku wcześniej przedstawionym:

```
<BODY bgcolor="#FFFFFF">
<P align="center">
  <MAP name="TWP">
    <AREA shape="rect" coords="119,19,266,51" href="download.html"
          alt=" Download ">
    <AREA shape="poly" coords="33,86,7,105,8,131,57,159,152,171,248,154,
                      285,131,288,106,247,80,159,66,103,68,91,83,68,93"
          href="wsiie.html" alt=" WSIiE ">
    <AREA shape="circle" COORDS="59, 46, 43, 0" HREF="apple.html"
          alt=" Apple ">
  </MAP>
  <IMG src="mapamenu.jpg" usemap="#twp" width="319" height="218" border="0">
</P>
</BODY>
```

Warto pamiętać również o tym, że w znaczniku <AREA>, podobnie jak w <IMG>, możemy umieszczać parametry opisujące zdarzenia. Są nimi m.in.: onclick, onmouseover, onmouseout, onmousedown, onmouseup, np.:

```
<MAP name="TWP">
  <AREA shape="rect" coords="119,19,266,51" href="download.html"
        alt=" Download " onclick="NoweOkno('info.html');">
  <AREA shape="poly" coords="33,86,7,105,8,131,57,159,152,171,248,154,285,
        131,288,106,247,80,159,66,103,68,91,83,68,93" href="wsiie.html"
        alt=" WSIiE ">
  <AREA shape="circle" COORDS="59,46,43,0" HREF="apple.html" alt=" Apple ">
</MAP>
```

Po naciśnięciu przycisku myszy w obszarze opisanym współrzędnymi 119, 19, 266, 51, wywołana zostanie funkcja NoweOkno, którą możemy przygotować np. w języku JavaScript czy VBScript.

2.7. Multimedia (dźwięk, animacje, filmy)

Poza typowymi elementami, jakimi są tekst czy grafika, nasz dokument może również zawierać inne obiekty multimedialne. Mogą być nimi dźwięk, animacje czy sekwencje wideo.

2.7.1. Obiekty multimedialne (znacznik EMBED)

Za pomocą znacznika <EMBED> możemy w naszym dokumencie umieścić plik dźwiękowy, sekwencję wideo czy obiekt Flash. Znacznik ten posiada kilka parametrów:

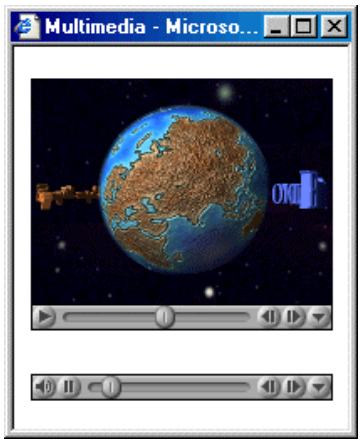
- src - nazwa pliku (adres URI pliku do odtworzenia),
- name - nazwa osadzonego obiektu,
- width - szerokość obrazu wyrażona w pikselach,
- height - wysokość obrazu wyrażona w pikselach,

np. `<EMBED src="HyperStudio.mov" width="180" height="150">`

- `hidden` - ukrycie interfejsu wstawionego obiektu. Parametr `hidden` może przyjąć jedną z dwóch wartości: `FALSE` lub `TRUE`, np.

```
<EMBED src="music.mov" hidden="true" width="150"
       height="20">
```

Poniżej, przy użyciu znacznika `<EMBED>` wstawiliśmy sekwencję wideo oraz utwór dźwiękowy:




```
<BODY>
  <P>
    <EMBED src="HyperStudio.mov" width="180"
           height="150">
  </P>
  <P>
    <EMBED src="Charriot.mov" width="180"
           height="20">
  </P>
</BODY>
```

Za pomocą znacznika `<EMBED>` możemy wstawiać pliki zapisane w wielu formatach:

```
<BODY>
  <P><EMBED src="movie01.mpg" width="355" height="300" hidden="true"></P>
  <P><EMBED src="hi.au"></P>
  <P><EMBED src="tada.wav" width="300" height="25" hidden="false"></P>
</BODY>
```

przy czym niektóre formaty plików multimedialnych wymagają zainstalowania w przeglądarce (lub systemie) odpowiednich plug-inów. W specyfikacji HTML 4.0 znacznik ten zastąpiono znacznikiem `<OBJECT>`, o którym piszę dalej.

2.7.2. Dźwięk w tle: znacznik BGSOUND

W poprzednim punkcie opisaliśmy m.in. w jaki sposób wstawić plik dźwiękowy do dokumentu HTML. Dodatkowo przeglądarka Internet Explorer ma wbudowane polecenie umożliwiające odtwarzanie muzyki (dźwięku) w tle, w trakcie przeglądania dokumentu. Mowa tu o znaczniku `<BGSOUND>`. Za jego pomocą możemy obsługiwać pliki dźwiękowe zapisane w formacie WAV, AU i MID.

Znacznik `<BGSOUND>` posiada następujące parametry:

- `src` - adres URI pliku do odtworzenia, np. `<BGSOUND src="canyon.mid">`
- `loop` - liczba powtórzeń, przy czym `loop="-1"` lub `loop="infinite"` oznacza pętlę nieskończoną, np. `<BGSOUND src="ooh.au" loop="3">`

```
<BODY>
  <BGSOUND src="janosik.mid" loop="-1">
  ... treść dokumentu ...
</BODY>
```

2.8. Ramki

Do tej pory zajmowaliśmy się dokumentami HTML, które wypełniały w całości ekran przeglądarki internetowej. Istnieje jednak możliwość podzielenia ekranu na kilka części, z których każda wypełniana jest treścią pochodzącą z innego pliku/dokumentu HTML. Daje nam to ogromne możliwości manipulowania kompozycją strony.

Przykładem może być często spotykana konstrukcja polegająca na podzieleniu ekranu na dwie części, z których pierwsza wyświetla menu wyboru, natomiast w drugiej, w zależności od wybranej opcji z menu, wyświetlane są kolejne dokumenty HTML.

2.8.1. Znaczniki FRAMESET i FRAME

Definicję ramek umieszcza się w oddzielnym pliku HTML, w którym nie występuje sekcja `<BODY> ... </BODY>`. Ciałem takiego dokumentu jest blok zawierający definicję ramek, rozpoczynający się znacznikiem `<FRAMESET>` i zamykany `</FRAMESET>`. Ogólną budowę pliku zawierającego definicje ramek przedstawiamy poniżej :

```
<HTML>
  <HEAD> ... treść nagłówka ... </HEAD>
  <FRAMESET>
    ... definicja poszczególnych ramek ...
  </FRAMESET>
</HTML>
```

Pomiędzy znacznikami `<FRAMESET>` i `</FRAMESET>` mogą występować jedynie znaczniki `<FRAME>`, definiujące kolejne ramki oraz zagnieżdżenie znacznika `<FRAMESET>` .

Znacznik `<FRAMESET>` posiada następujące parametry:

- `cols` - pionowy podział ekranu (liczba kolumn), określający szerokość poszczególnych kolumn w pikselach lub procentach zajętości ekranu,
  np. `<FRAMESET cols="15%,80%,*">`, utworzy 3 kolumny o szerokościach wyrażonych w procentach zajętości ekranu,

gdzie 1 kolumna = 15%, 2 kolumna = 80% i 3 kolumna = 100% - (15% + 80%) = 5%,

- `rows` - poziomy podział ekranu (liczba wierszy), określający wysokość poszczególnych wierszy w pikselach lub procentach zajętości ekranu,
  np. `<FRAMESET rows="50,*">` , utworzy 2 wiersze, gdzie wiersz 1 będzie miał wysokości 50 pikseli, natomiast 2 wiersz zajmie pozostałą część ekranu zależną od rozmiaru okna przeglądarki,
- `border` - grubość obramowania (separatora) oddzielającego poszczególne ramki podawana w pikselach,
  np. `<FRAMESET border="0" cols="10%,*">`, spowoduje ukrycie separatora,
- `bordercolor` - kolor obramowania, podawany w wartościach szesnastkowych lub za pomocą predefiniowanych nazw,
  np. `<FRAMESET bordercolor="#8800A7" rows="100,200,*">`
- `frameborder` - włączenie lub wyłączenie obramowania ramek, przy czym 0 oznacza jego wyłączenie, a 1 włączenie,
  np. `<FRAMESET frameborder="0" cols="40%,*">`
- `framespacing` - odstęp występujący pomiędzy ramkami podawany w pikselach, przy czym parametr ten występuje jak na razie jedynie w IE,
  np. `<FRAMESET framespacing="10" cols="40%,*">`

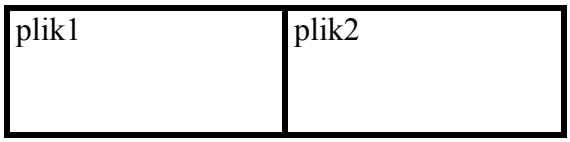
Znacznik `<FRAME>` posiada następujące parametry:

- `src` - Adres dokumentu, który zostanie załadowany do ramki,
  np. `<FRAME src="opis.html">`
  czy `<FRAME src="http://www.uninet.pl/index.html">`
- `name` - Nazwa ramki stosowana do komunikowania się z pozostałymi ramkami, np. `<FRAME name="menu" src="mojemenu.html">`, poprzez nazwę możemy wskazać na konkretną ramkę, do której będzie załadowany dokument HTML,
  np. `<A href="menu2.html target="menu">Zmiana menu</A>`
- `border` - Grubość obramowania (separatora) oddzielającego ramki podawana w pikselach.
- `bordercolor` - Kolor obramowania pojedynczej ramki, podawany w wartościach szesnastkowych lub za pomocą predefiniowanych nazw, np. `<FRAME bordercolor="blue" src="plik.html">`
- `frameborder` - Włączenie lub wyłączenie obramowania pojedynczej ramki, gdzie 0 oznacza jego wyłączenie, a 1 włączenie.
- `marginheight` - Odstęp pomiędzy górną i dolną krawędzią ramki a zawartością dokumentu podawany w pikselach.
- `marginwidth` - Odstęp pomiędzy lewą i prawą krawędzią ramki a zawartością dokumentu podawany w pikselach.
- `noresize` - Zablokowanie możliwości zmiany rozmiarów ramki.
- `scrolling` - Określenie trybu wyświetlania pasków przewijania w ramce:
  - `auto` paski pojawiają się automatycznie w momencie, gdy dokument nie mieści się w ramce,

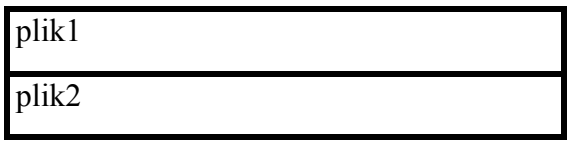
- `yes` paski przewijania są stale widoczne,
- `no` brak pasków przewijania w ramce,

np. `<FRAME src="a.htm" noresize scrolling="auto">`

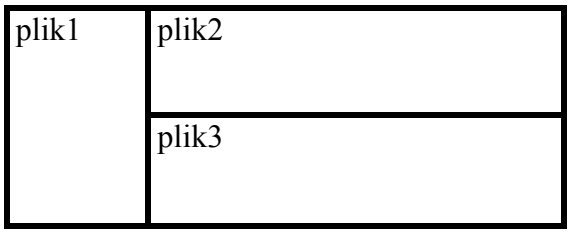
Poniżej przedstawiamy kilka przykładowych kompozycji:



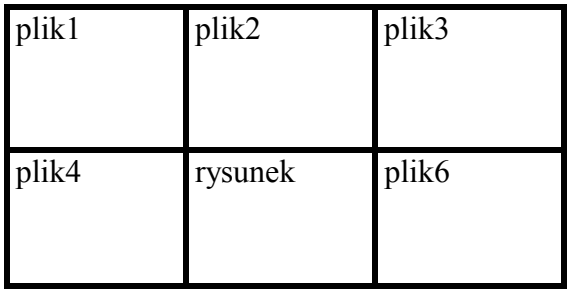

```
<FRAMESET cols="50%,*">
  <FRAME src="plik1.html">
  <FRAME src="plik2.html">
</FRAMESET>
```




```
<FRAMESET rows="50%,*">
  <FRAME src="plik1.html">
  <FRAME src="plik2.html">
</FRAMESET>
```




```
<FRAMESET cols="25%,*">
  <FRAME src="plik1.html">
  <FRAMESET rows="50%,*">
    <FRAME src="plik2.html">
    <FRAME src="plik3.html">
  </FRAMESET>
</FRAMESET>
```




```
<FRAMESET rows="50%,50%"
          cols="33%,34%,33%">
  <FRAME src="plik1.html">
  <FRAME src="plik2.html">
  <FRAME src="plik3.html">
  <FRAME src="plik4.html">
  <FRAME src="rysunek.gif">
  <FRAME src="plik6.html">
</FRAMESET>
```

I na zakończenie przedstawiamy treść dokumentu HTML tworzącego konstrukcję jak na rysunku przedstawionym wcześniej (acPlamka!):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN"
                      "http://www.w3.org/TR/REC-html40/frameset.dtd">
<HTML>
<HEAD>
  <META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">
  <TITLE>acPlamka!</TITLE>
</HEAD>

<FRAMESET border="0" cols="180,*">
  <FRAME src="menu.html" name="menu" noresize scrolling="no">
  <FRAME src="graf.html" name="main" scrolling="auto">
</FRAMESET>
</HTML>
```

2.8.2. Znacznik NOFRAMES

Wszystkie współczesne i popularne programy umożliwiające przeglądanie dokumentów HTML, bezproblemowo obsługują ramki. Jednak wszystkie przeglądarki, które nie obsługują znacznika `<FRAMESET>` pominą całą definicję, a tym samym nie wyświetlą dokumentów zdefiniowanych w znacznikach `<FRAME>`.

Alternatywą dla takich przeglądarek jest rozszerzenie definicji zawartej pomiędzy znacznikami <FRAMESET> i </FRAMESET> o blok, który będzie obsługiwany również i przez nie.
Blok taki tworzymy pomiędzy znacznikami <NOFRAMES> i </NOFRAMES>. W bloku tym umieszczamy kompletne ciało dokumentu pomiędzy znacznikami <BODY> i </BODY>, np. :

```
<FRAMESET cols="20%,*">
  <FRAME src="menu.html" name="menu" noresize scrolling="no">
  <FRAME src="pierwszy.html" name="main" scrolling="auto">

  <NOFRAMES>
  <BODY>
    <P>Menu:</P>
    <P><A href="pierwsza.html">Strona 1</A></P>
    <P><A href="druga.html">Strona 2</A></P>
    <P><A href="trzecia.html">Strona 3</A></P>
  </BODY>
  </NOFRAMES>

</FRAMESET>
```

Jeżeli więc nasz dokument zostanie załadowany do przeglądarki nie obsługującej ramek, to w jej oknie zostanie wyświetlona treść podana pomiędzy znacznikami <BODY> i </BODY> znajdującymi się w bloku <NOFRAMES>, a więc trzy odsyłacze do kolejnych stron.

2.8.3. Ramki wbudowane: znacznik IFRAME

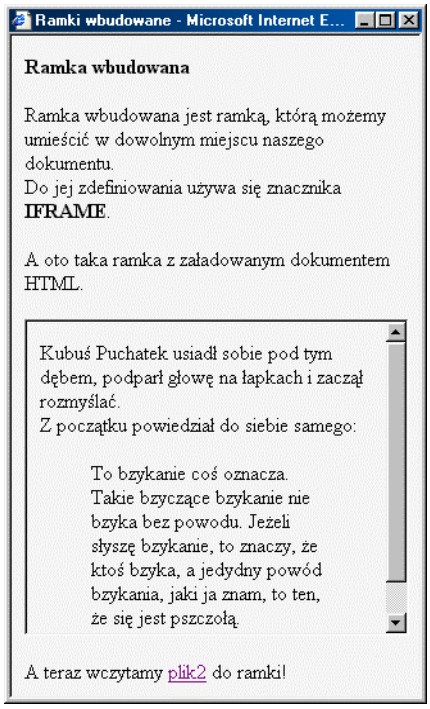
Bardzo ciekawym mechanizmem związanym z ramkami jest ramka wbudowana w dokument, nazywana również ramką pływającą.
Typowe ramki, o których pisałem do tej pory, pozwalają dzielić ekran przeglądarki na odrębne części. Ramka wbudowana jest ramką, którą możemy umieścić w dowolnym miejscu naszego dokumentu. Do jej zdefiniowania używa się znaczników <IFRAME> i </IFRAME>.

Ważniejsze parametry znacznika <IFRAME> to:

- src - Adres URI dokumentu, który zostanie załadowany do ramki, np. <IFRAME src=”opis.html” width=”200” height=”90”>
- width - Szerokość ramki podawana w pikselach lub procentach zajętości ekranu.
- height - Wysokość ramki podawana w pikselach lub procentach zajętości ekranu,
  np. <IFRAME width="80%" height="200" src="p1.html">
- name - Nazwa ramki, poprzez którą możemy kierować do niej ładowane dokumenty,
  np. <A href=”opis.html” target=”ifrm”>Opis</A>
- noresize - Wyłączenie możliwości zmiany rozmiaru ramki.
- frameborder - Włączenie lub wyłączenie trójwymiarowego obramowania ramki, np. <IFRAME src="ppp.html" frameborder=”1” ...

Dodatkowo występuje wiele innych parametrów, m.in.: `hspace`, `vspace`, `marginwidth`, `marginheight`, `scrolling`, które mają takie same znaczenie jak w przypadku ramek typowych oraz `align`, za pomocą którego możemy umieścić ramkę po prawej stronie dokumentu (`align="right"`).

Oto przykład ramki wbudowanej w dokument:




```
<BODY>
 <P><B>Ramka wbudowana</B></P>
 <P>Ramka wbudowana jest ramką, którą
    możemy umieścić w dowolnym
    miejscu naszego dokumentu.<BR>
    Do jej zdefiniowania używa się
    znacznika <B>IFRAME</B>.</P>
 <P>A oto taka ramka z załadowanym
   dokumentem HTML.</P>
 <P>

 <IFRAME width="300" height="250"
    name="ramka" src="plik1.html"
    scrolling="auto">

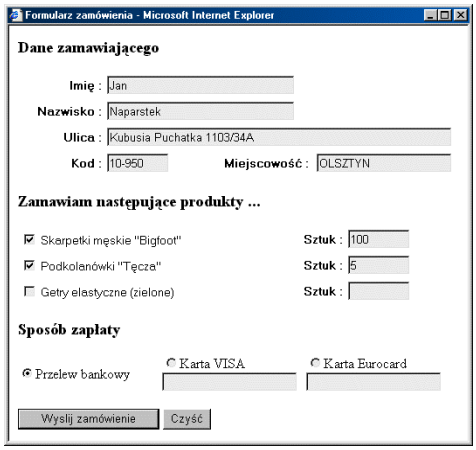
    Ta przeglądarka nie obsługuje
    ramek wbudowanych...
 </IFRAME>

 <P>
  A teraz wczytamy
 <A href="plik2.html" target="ramka">
    plik2
 </A>
  do ramki!</P>
</BODY>
```

2.9. Formularze

W zasadzie każdy większy serwis internetowy posiada elementy typowe dla formularzy. Przykładami wykorzystania elementów formularzy są np. księgi gości, fora, serwisy wyszukujące oraz strony, na których możemy robić zakupy. Ze względu jednak na ich użyteczność, bardzo często wykorzystywane są również do sterowania stroną, co pokażemy w dalszej części na kilku przykładach.

Oto wygląd przykładowego formularza:



Sposób wysyłania danych z formularza możemy podzielić na dwa główne rodzaje:

- dane wysyłane są do serwera (w tej sytuacji dane obsługiwane są najczęściej przez specjalne skrypty CGI czy PHP),
- dane wysyłane są pod konkretny adres e-mail.

2.9.1. Znacznik FORM

Definicję całego formularza rozpoczynamy znacznikiem `<FORM>`, a kończymy `</FORM>`, tak jak poniżej:

```
<FORM>
  ... treść formularza ...
</FORM>
```

Znacznik `<FORM>` posiada kilka parametrów:

- `action` - Adres URI skryptu CGI lub PHP obsługującego formularz lub adres e-mail, pod który przesłane będą dane z formularza.
- `method` - Metoda przesyłania danych z formularza, która może posiadać wartości `GET` lub `POST`.
- `enctype` - Określenie programu obsługującego odpowiedni typ danych pochodzących z formularza (pliki dźwiękowe, graficzne itp.), np. `<FORM ENCTYPE="image/jpeg" ...>`, wymusi przesłanie danych z formularza i uruchomienie programu obsługującego pliki typu `jpg`.
- `name` - Nazwa formularza (wykorzystywana przez skrypty do jego identyfikacji).
- `target` - Określenie nazwy ramki, do której zostanie załadowany rezultat wykonania skryptu CGI, np. `<FORM TARGET="ramka_prawa" ... >`
- `accept` - Określenie listy typów zawartości (rozdzielonych przecinkami), które może przyjąć i przetworzyć serwer. Przykładowe wartości tego parametru to np.: `"text/html"`, `"image/png"`, `"image/gif"`, `"video/mpeg"`, `"text/css"`, `"audio/basic"`.
- `accept-charset` - Określenie listy alfabetów akceptowanych przez serwer przyjmujący dane z formularza, np. `iso-8859-2`.

Treść formularza, znajdująca się pomiędzy wspomnianymi znacznikami składa się z obiektów formatujących jego wygląd. Obiekty te, które nazywać będziemy również kontrolkami, prezentujemy poniżej.

2.9.2. Znacznik INPUT

Za pomocą znacznika `<INPUT>`, którego używamy w obrębie formularza, możemy tworzyć różne kontrolki. Rodzaj kontrolki ustala się za pomocą parametru `type`. Może to być pole tekstowe, pola opcji, przełączników czy przyciski.

Znacznik <INPUT> posiada następujące parametry:

- type - Typ (rodzaj) tworzonej kontrolki:
  - text
  - password
  - checkbox
  - radio
  - submit
  - reset
  - button
  - file
  - image
  - hidden

  np. <INPUT type="button" name="btnStart" value="Start">
- name - Nazwa kontrolki.
- value - Wartość domyślna kontrolki, która jest zależna od rodzaju kontrolki, tj. dla kontrolek typu button, reset i submit, parametr value określa napis, jaki będzie umieszczony na przycisku, natomiast dla kontrolek checkbox i radio parametr value określa, jaka wartość ma być przesyłana wraz z formularzem,

  np.
  ```
  <INUPT type="button" value="Start" name="B1">
  <INPUT type="radio" value="off" name="R1">
  ```
- size - Długość pola tekstowego określana dla kontrolek text i password,

  np. <INPUT type="text" name="nazwa" size="20">
- maxlength - Maksymalna długość wprowadzanego tekstu w kontrolkach typu text i password,

  np. <INPUT type="password" name="pwd" maxlength="8">
- checked - Określenie stanu kontrolek typu checkbox i radio na „włączony" (zaznaczony),

  np. <INPUT type="radio" name="R1" checked>
- src - Dla kontrolek typu image określa adres URI rysunku,

  np. <INPUT type="image" name="rys" src="bocian.jpg">
- align - Dla kontrolek typu image określa położenie wiersza tekstu względem rysunku, gdzie align może przyjąć wartości top, middle i bottom,

  np. <INPUT type="image" src="mapa.jpg" align="top">

Stosowanie znacznika </INPUT> jest niezalecane.

W odniesieniu do znacznika <INPUT> również możemy używać parametru style.

Teraz zajmiemy się omówieniem kolejnych rodzajów kontrolek. Jak już wiemy, rodzaj kontrolki ustalamy za pomocą parametru type.

- **text**

Jednowierszowe pole do wprowadzania tekstu.

Jeżeli wprowadzony tekst będzie dłuższy niż szerokość pola określona parametrem `size`, to tekst ten będzie przewijany w poziomie.

```
<INPUT type=”text” name=”nazwisko” size=”20” maxlength=”40”>
```

- **password**

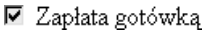
Jednowierszowe pole do wprowadzania tekstu, przy czym zamiast liter wyświetlane są gwiazdki (*).

Jeżeli wprowadzony tekst będzie dłuższy niż szerokość pola określona parametrem `size`, to tekst ten będzie przewijany w poziomie.

```
<INPUT type=”password” name=”pwd” size=”8”>
```

- **checkboxes**

Pole wyboru, które możemy ustawić na jedną z dwóch wartości: włączony lub wyłączony.

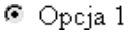


W grupie wielu takich pól wartość „włączony” może posiadać kilka elementów jednocześnie.

```
<INPUT type="checkbox" name="C1" value="Gotowka"> Zapłata gotówką
```

- **radio**

Pole wyboru, które możemy ustawić na jedną z dwóch wartości: włączony lub wyłączony.



W grupie wielu takich pól wartość „włączony” może posiadać wyłącznie jeden element.

```
<INPUT type="radio" name="R1" value="op1"> Opcja 1
```

- **submit**

Przycisk, którego naciśnięcie powoduje wysłanie danych z formularza, do serwera lub pod ustalony wcześniej adres e-mail.


```
<input type="submit" value="Wyślij formularz">
```

- **reset**

Przycisk, którego naciśnięcie powoduje przywrócenie wartości domyślnych dla wszystkich pól formularza.


```
<INPUT type="reset" value="Czyść formularz">
```

- **button**

Standardowy przycisk.


```
<INPUT type="button" name="btn1" value="Poprzednia" onClick="Prev()">
```

- **file**

Plik, który możemy dołączyć do danych wysyłanych z formularza.

Gdy dane z formularza wysyłane są pocztą elektroniczną, wskazany plik staje się zwykłym załącznikiem.

```
<INPUT type="file" name="plik">
```

- **image**

Obiekt graficzny (rysunek).

Kliknięcie na rysunku powoduje wysłanie z formularzem współrzędnych pozycji kursora myszki.

```
<INPUT type="image" name="rys" src="apple.gif">
```

- **hidden**

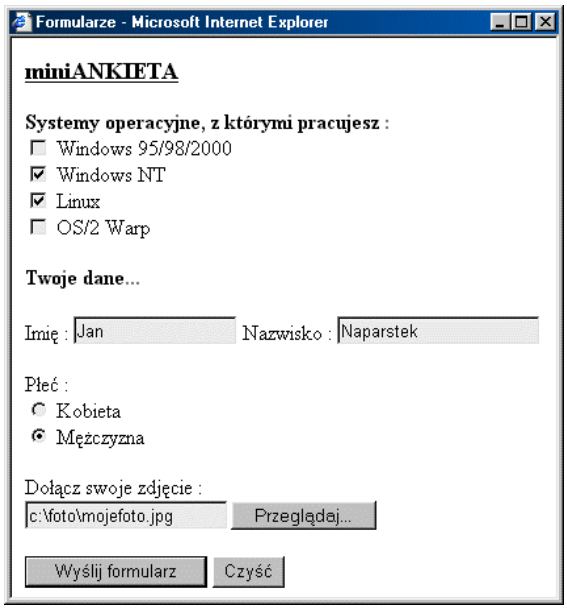
Kontrolka tego typu nie jest wyświetlana na ekranie, jednak jej wartość jest wysyłana razem z formularzem do serwera.

Wykorzystanie tego typu kontrolek przedstawimy na przykładzie poniższego formularza:

```
<FORM method="POST" action="http://www.musc.edu/cgi-bin/formmail.pl">
  <INPUT type="hidden" name="recipient" value="user@olsztyn.mail.pl">
```

```
  <INPUT type="hidden" name="required" value="realname,email">
  <INPUT type="hidden" name="redirect"
         value="http://www.twp.olsztyn.pl/yourdirectory/thanks.html">
  <INPUT type="text" name="realname">
  <INPUT type="text" name="email">
  <INPUT type="text" name="subject">
  <INPUT type="submit" value="Send It">
  <INPUT type="reset">
</FORM>
```

A teraz przedstawiamy formularz, który zbudowany jest z wyżej opisanych kontrolek. Dane z formularza wysyłane będą pod adres `ankieta@firma.pl` :



Ten sam formularz w HTML-u wygląda tak:

```
<FORM action="mailto:ankieta@firma.pl" method="post">
 <P><B>Systemy operacyjne, z którymi pracujesz :<BR></B>
  <INPUT type="hidden" name="autor" value="artac">
  <INPUT type="checkbox" name="C1" value="Win"> Windows 95/98/2000<BR>
  <INPUT type="checkbox" name="C2" value="Winnt"> Windows NT<BR>
  <INPUT type="checkbox" name="C3" value="Linux"> Linux<BR>
  <INPUT type="checkbox" name="C4" value="OS2"> OS/2 Warp
 </P>
 <P><B>Twoje dane...</B></P>
 <P>
   Imię : <INPUT type="text" name="imie" size="15" maxlength="20">
   Nazwisko : <INPUT type="text" name="nazwisko" size="20" maxlength="30">
 </P>
 <P>
  Płeć :<BR>
  <INPUT type="radio" name="R1" value="op1"> Kobieta<BR>
  <INPUT type="radio" name="R2" value="op2"> Mężczyzna
 </P>
 <P>
  Dołącz swoje zdjęcie : <BR>
  <INPUT type="file" name="foto">
 </P>
 <P>
  <INPUT type="submit" value="Wyślij formularz">
  <INPUT type="reset" value="Czyść">
 </P>
</FORM>
```

2.9.3. Znaczniki SELECT i OPTION

Za pomocą znacznika `<SELECT>` tworzymy listę, z której możemy wybrać jedną lub kilka pozycji. Listę kończymy znacznikiem `</SELECT>` i jest on wymagany. Kolejne opcje (pozycje) dodajemy do listy za pomocą znacznika `<OPTION>`.
Zanim przedstawię jakąś przykładową listę, opiszę parametry wspomnianych znaczników.

Zacznę od znacznika `<SELECT>` :

- `name` - Nazwa listy.
- `size` - Rozmiar czcionki używanej do opisywania elementów listy, który określany jest liczbą od 1 do 7 (wartością domyślną jest 3),
  np. `<SELECT name="modele" size="4">`
- `multiple` - Użycie tego parametru pozwala na wybieranie z listy więcej niż jednego elementu,
  np. `<SELECT name="wyszukiwarki" multiple>`

Z kolei znacznik `<OPTION>` posiada następujące parametry:

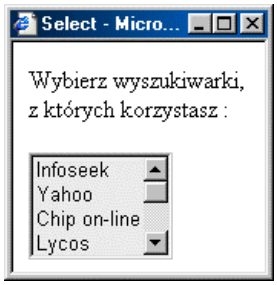
- `value` - Wartość przesyłana razem z formularzem w przypadku zaznaczenia elementu (opcji),
  np. `<OPTION value="op1">Opcja 1`
- `selected` - Automatyczne zaznaczenie elementu (opcji) przy wejściu do formularza,
  np. `<OPTION value="pl" selected>Polska`
- `label` - Treść etykiety definiowanej opcji.

Lista z możliwością wybrania tylko jednej opcji:


```
<P>Wybierz model :</P>

<SELECT name="model">
  <OPTION value="m1">Citroen AX
  <OPTION value="m2">Citroen ZX
  <OPTION value="m3">Citroen XM
  <OPTION value="m4">Citroen Saxo
</SELECT>
```

Lista z możliwością wybrania kilku opcji:




```
<P>Wybierz wyszukiwarki, z których korzystasz :</P>

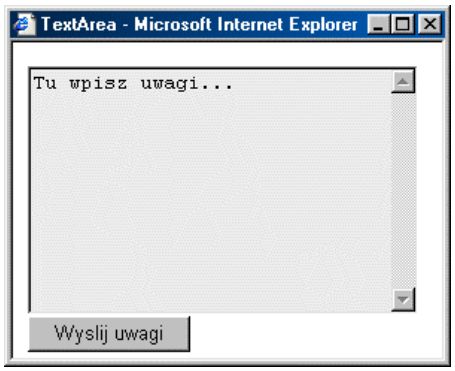
<SELECT name="s1" multiple>
  <OPTION value="infoseek">Infoseek
  <OPTION value="yahoo">Yahoo
  <OPTION value="chip">Chip on-line
  <OPTION value="lycos">Lycos
  <OPTION value="hotbot">HotBot
  <OPTION value="inne">i inne...
</SELECT>
```

2.9.4. Znacznik TEXTAREA

Za pomocą znaczników `<TEXTAREA>` i `</TEXTAREA>` definiujemy wielowierszowe pole tekstowe, do którego możemy wprowadzać stosunkowo długie łańcuchy.
Znacznik ten posiada kilka parametrów. Najważniejsze z nich to:

- `name` - Nazwa pola tekstowego używana do identyfikacji.
- `rows` - Liczba wierszy w polu tekstowym.
- `cols` - Liczba kolumn w polu tekstowym,
  np. `<TEXTAREA cols=”25” rows=”10” name=”opis2”>...`

Znacznik `</TEXTAREA>` jest wymagany.



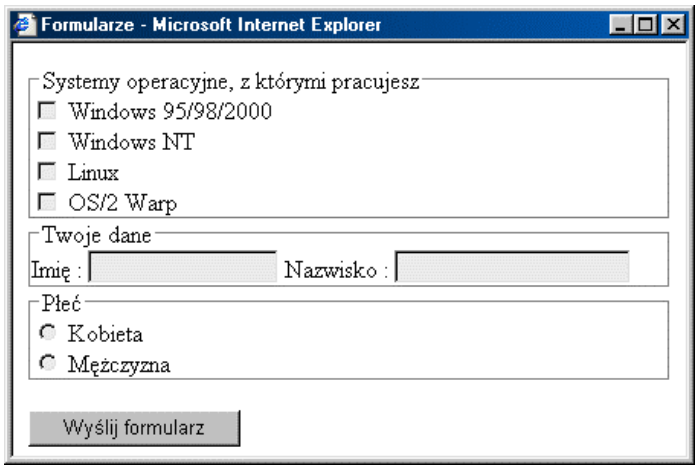

```
<FORM action="mailto:info@firma.pl"
      method="post">
  <TEXTAREA name="opis1" rows="10"
            cols="30">
    Tu wpisz uwagi...
  </TEXTAREA>
  <BR>
  <INPUT type="submit"
         value="Wyślij uwagi">
</FORM>
```

Bardzo ciekawą opcją jest możliwość zablokowania zmian w obiekcie `textarea`, dając tym samym jedynie prawa do odczytu. Robimy to przez użycie wraz ze znacznikiem `<TEXTAREA>` parametru `readonly`.

```
<TEXTAREA name=”umowa” cols=”40” rows=”20” readonly>
  ... Informacja tylko do odczytu ...
</TEXTAREA>
```

2.9.5. Nadawanie struktury formularzom (znaczniki FIELDSET i LEGEND)

Pola formularza możemy grupować tematycznie, dzięki czemu cały formularz staje się bardziej przejrzysty. Poniższy rysunek przedstawia formularz, w którym utworzono trzy grupy:



Każda z grup otoczona jest ramką i posiada tytuł. Grupę pól definiuje się znacznikami <FIELDSET> i </FIELDSET>, natomiast tytuł grupy znacznikami <LEGEND> i </LEGEND>.

W HTML-u wygląda to tak:

```
<FORM action="mailto:ankieta@firma.pl" method="post">
  <FIELDSET>
    <LEGEND> Systemy operacyjne, z którymi pracujesz </LEGEND>
      <INPUT type="checkbox" name="C1" value="Win"> Windows 95/98/2000<br>
      <INPUT type="checkbox" name="C2" value="Winnt"> Windows NT<br>
      <INPUT type="checkbox" name="C3" value="Linux"> Linux<br>
      <INPUT type="checkbox" name="C4" value="OS2"> OS/2 Warp
  </FIELDSET>

  <FIELDSET>
    <LEGEND> Twoje dane </LEGEND>
      Imię : <INPUT type="text" name="imie" size="15">
      Nazwisko : <INPUT type="text" name="nazwisko" size="20">
  </FIELDSET>

  <FIELDSET>
    <LEGEND> Płeć </LEGEND>
      <INPUT type="radio" name="R1" value="op1"> Kobieta<br>
      <INPUT type="radio" name="R2" value="op2"> Mężczyzna
  </FIELDSET>

  <P><INPUT type="submit" value="Wyślij formularz">
</FORM>
```

2.9.6. Parametr accesskey, kontrolki nieaktywne i tylko do odczytu

Z formularzami związane są jeszcze trzy ciekawe parametry, o których nie sposób nie wspomnieć. Oto one:

- accesskey
- disabled
- readonly

Parametrów związanych z kontrolkami jest zdecydowanie więcej, choćby cała grupa parametrów opisujących zdarzenia (onfocus, onblur, onselect, onclick, onmouseover, onkeypress ... etc.) czy też tabindex, my jednak w tym opisie ograniczymy się do trzech powyższych.

- **accesskey**

Parametr accesskey wykorzystuje się do zdefiniowania „gorących klawiszy", dzięki którym możemy szybko poruszać się po formularzu za pomocą klawiatury. Oto przykład użycia parametru accesskey :

```
<FORM name="formularz" ...>
  <P>
  <LABEL for="txt1" accesskey="U">
    Użytkownik :
  </LABEL>
  <INPUT type="text" name="t1" size="10" id="txt1">
```

```
  <P>
  <LABEL for="txt3" accesskey="H">
    Hasło :
  </LABEL>
  <INPUT type="text" name="t2" size="10" id="txt2">
  <P>
  <LABEL for="txt3" accesskey="G">
    Grupa :
  </LABEL>
  <INPUT type="text" name="t3" size="10" id="txt3">
</FORM>
```

W powyższym przykładzie parametr `accesskey` został użyty w znacznikach `<LABEL>`, które powiązane są z polami tekstowymi za pomocą identyfikatorów `txt1`, `txt2` i `txt3`.

Również znacznik `<A>`, tworzący odsyłacze hipertekstowe, może zawierać ten parametr:

```
<A href=”wyszukiwarki.html” accesskey=”W”>
  Wyszukiwarki internetowe
</A>
```

Parametr `accesskey`, poza znacznikami `<LABEL>` i `<A>` może wystąpić również w `<AREA>`, `<BUTTON>`, `<INPUT>`, `<LEGEND>` i `<TEXTAREA>`.

- **disabled**

Parametrem `disabled` możemy unieaktywnić kontrolkę, przez co użytkownik nie może zmieniać jej wartości:

```
<INPUT type=”text” name=”txt1” size=”20” disabled>
```

Parametr `disabled` może wystąpić w `<BUTTON>`, `<INPUT>`, `<OPTION>`, `<OPTGROUP>`, `<SELECT>` oraz `<TEXTAREA>`.

Dla kontrolek nieaktywnych nie można uzyskać fokusu. Nie można również przejść do nich klawiszem tabulacji.

- **readonly**

Umieszczenie parametru `readonly` w znacznikach `<INPUT>` lub `<TEXTAREA>` tworzy kontrolki, których wartości można tylko czytać:

```
<INPUT type=”text” name=”ro” value=”const” readonly>
```

Dla kontrolek ustawionych tylko do odczytu można uzyskać fokus oraz przejść do nich klawiszem tabulacji.

Na zakończenie omawiania formularzy warto jeszcze wspomnieć o sposobie rozmieszczania elementów formularza na ekranie. Uzyskanie jednolitego poziomego wyrównania pól wprowadzania danych jest bardzo trudne, a czasami wręcz niemożliwe. Jednak używając do tego celu tabel sprawa staje się prosta, a nasze formularze

wyglądają bardzo elegancko, jak przykładowy forumlarz przedstawiony na początku rodziału.

2.10. Aplety i obiekty

Coraz częściej na stronach internetowych możemy spotkać obiekty, które są niezależnie działającymi aplikacjami, dającymi twórcom stron nieograniczone wręcz możliwości. W szczególności mowa o apletach napisanych przy pomocy języka zorientowanego obiektowo, jakim jest JAVA, ale nie tylko.

W języku HTML występują odpowiednie znaczniki, za pomocą których możemy wstawiać takie programy do naszego dokumentu (strony).

Obok widać przykładowy aplet napisany w języku JAVA (pomijając grafikę rzecz jasna), który w rzeczywistości jest w pełni interaktywną stroną, na której niemal każdy jej fragment reaguje na ruchy naszej myszy. Poza ciekawymi efektami wizualnymi dodano również efekty dźwiękowe.

2.10.1. Znaczniki APPLET i PARAM

Aplet możemy wstawić do dokumentu HTML używając jednego z dwóch znaczników `<APPLET>` lub `<OBJECT>`.
Jako pierwszy omówię znacznik `<APPLET>`, który używany był już w poprzedniej specyfikacji języka HTML, a od którego odchodzi się na rzecz znacznika `<OBJECT>`.

Znacznik `<APPLET>` posiada m.in. następujące parametry:

- `code` - Nazwa pliku (klasy) apletu,
  np. `<APPLET code="baner.class" width="200" height="40">`
- `codebase` - Adres bazowy URI dla klas i apletów.
- `name` - Nazwa apletu wykorzystywana przez inne aplety na stronie.
- `width` - Początkowa szerokość powierzchni zajmowanej przez aplet.
- `height` - Początkowa wysokość powierzchni zajmowanej przez aplet.
- `alt` - Tekst wyświetlany zamiast apletu.
  Dot. przeglądarek nie obsługujących aplety napisane w języku JAVA.
- `align` - Położenie obszaru zajmowanego przez aplet względem otaczających go elementów. Parametr `align` może przyjąć wartość: `top`, `middle`, `bottom`, `left`, `right`, `texttop`, `absmiddle`, `absbottom`, `baseline`,
  np. `<APPLET code="hop.class" ... align="left">`

Fragment w HTML-u zawierający definicję apletu może wyglądać tak:

```
<BODY>
  <HR>
  <APPLET code="Duke.class" width="150" height="100" align="bottom">
  </APPLET>
  Aplet "Under Construction"
  <HR>
</BODY>
```

Powyższa konstrukcja spowoduje wstawienie do dokumentu apletu `Duke.class`, który zajmie powierzchnię o szerokości 150 i wysokości 100 pikseli. Dodatkowo tekst `Aplet „Under Construction”` ustawiony będzie u dołu, po prawej stronie apletu.
Do apletów możemy również przekazywać pewne parametry (dane), których ilość i rodzaj zależny jest od budowy apletu. Parametry takie definiujemy pomiędzy znacznikami `<APPLET>` i `</APPLET>` za pomocą znacznika `<PARAM>`. Znacznik ten posiada następujące parametry:

- `name` - Nazwa parametru.
- `value` - Wartość parametru, przekazywana do apletu (zawsze jako łańcuch znakowy).
- `type` - Rodzaj zasobu internetowego (określany wyłącznie, gdy `valuetype` ustawiony jest na `ref`).
- `valuetype` - Określenie sposobu interpretowania wartości podanej w `value`.
  Parametr `valuetype` może przyjąć wartości:
  - `data` - wartość domyślna, określająca typowe dane łańcuchowe,
  - `ref` - wartość będąca adresem URI,
  - `object` - wartość identyfikująca obiekt w tym samym dokumencie,

  np. `<PARAM name=”txt” value=”Hello” valuetype=”data”>`

A oto definicja apletu tworzącego migające napisy, do którego przekazujemy dwa parametry: pierwszy o nazwie `lbl`, określający wyświetlane (migoczące) napisy oraz drugi o nazwie `speed`, określający szybkość migotania.

```
<HTML>
<TITLE>Blinking Text</TITLE>
<BODY>
  <HR>
  <APPLET code="Blink.class" width="250" height="80">
    <PARAM name="lbl" value="Blinking Text, hoo hoo, ...">
    <PARAM name="speed" value="4">
    Aplet BLINKING TEXT
  </APPLET>
  <HR>
  Aplet "Blinking Text"
</BODY>
</HTML>
```

2.10.2. Wstawianie obiektów (znacznik OBJECT)

Aplety napisane w języku JAVA możemy również wstawiać do dokumentu HTML za pomocą znaczników `<OBJECT>` i `</OBJECT>`. W rzeczywistości za pomocą tych znaczników możemy oprócz apletów wstawiać inne obiekty.
Znacznik `<OBJECT>` posiada m.in. następujące parametry:

- `classid` - Lokalizacja implementacji obiektu (klasy) – nazwa klasy.
  Składnia wartości tego parametru zależy od rodzaju obiektu,
  np. `<OBJECT classid="java:MojAplet.class" ...>`
- `codetype` - Parametr opcjonalny, określający rodzaj danych pobieranych podczas ładowania obiektu wyspecyfikowanego w `classid`.
- `codebase` - Adres bazowy URI dla obiektów.
  Składnia wartości tego parametru zależy od rodzaju obiektu.
- `data` - Adres URI danych dla obiektu.
  Składnia wartości tego parametru zależy od rodzaju obiektu.
- `type` - Rodzaj obiektu (typ MIME), np.:
  - `"text/html"`,
  - `"image/png"`,
  - `"image/gif"`,
  - `"video/mpeg"`,
  - `"text/css"`,
  - `"audio/basic"`.

  np. `<OBJECT data="picture.png" type="image/png">`
- `width` - Początkowa szerokość powierzchni zajmowanej przez obiekt.
- `height` - Początkowa wysokość powierzchni zajmowanej przez obiekt.
- `align` - Jak w znaczniku `<APPLET>`.
- `standby` - Komunikat, który ma być wyświetlany w trakcie ładowania obiektu.
- `declare` - Deklaracja obiektu bez jego tworzenia.

Poza wymienionymi parametrami, wraz ze znacznikiem `<OBJECT>`, możemy stosować dość pokaźną grupę parametrów opisujących zdarzenia (`onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup`), jak również `tabindex`, `usemap`, `style`.

Przykładowe zastosowanie znacznika `<OBJECT>` do wstawienia apletu prezentuję poniżej:

```
<BODY>
   ...

   <OBJECT codetype="application/java" classid="java:Blink.class"
           width="250" height="80">
     <PARAM name="lbl" value="Blinking Text, hoo hoo, ...">
     <PARAM name="speed" value="4">
   </OBJECT>
   Aplet "Blinking Text"

   ...
</BODY>
```

Konstrukcja wstawiająca aplety, wykorzystująca znaczniki <OBJECT> i </OBJECT> jest bardziej zalecana od stosowanych do tej pory znaczników <APPLET> i </APPLET>.

Innymi obiektami, bardzo popularnymi, często goszczącym na stronach WWW, są obiekty Flash'a czy QuickTime'a.

2.11. Arkusze stylów

Ze względu na to, że arkusze stylów są bardzo ciekawym narzędziem, omówimy je w dalszej części pracy dokładniej. W tym miejscu zostaną omówione jedynie podstawowe zagadnienia związane bezpośrednio z HTML-em.

Kaskadowe arkusze stylów, zdefiniowane na potrzeby naszego dokumentu HTML, pozwalają w prosty sposób wpływać na parametry elementów umieszczanych w dokumencie. Zdefiniowane na początku każdej strony (ale nie tylko), dają możliwość zachowania w nich jednolitego stylu. Dzięki nim, podczas tworzenia dokumentu nie musimy już po każdym wprowadzonym elemencie definiować jego wszystkich parametrów. Jest to bardzo wygodny mechanizm i przez to jest on powszechnie zalecany i wykorzystywany do formatowania stron WWW.

2.11.1. Definiowanie domyślnego języka dla arkuszy stylów

Zanim przystąpimy do korzystania z arkuszy stylów musimy zdefiniować dla nich domyślny język. Możemy to zrobić za pomocą znacznika <META>. Jeżeli w naszym dokumencie HTML domyślnym stylem jest CSS, to w cześci nagłówkowej (pomiędzy <HEAD> i </HEAD>) musimy umieścić następujący wiersz:

```
<META http-equiv="Content-Style-Type" content="text/css">
```

Zdefiniowanie stylu domyślnego jest szczególnie ważne, gdy nasz dokument zawiera elementy, w których parametrowi `style` przypisujemy jakieś wartości.

2.11.2. Tworzenie jednowierszowych stylów

Możliwość tworzenia jednowierszowych stylów daje nam jeszcze większą kontrolę nad parametrami elementów. Jednowierszowe style definiujemy za pomocą parametru `style`. Ustawienie tego parametru wpływa na parametry pojedynczego elementu. Parametr może być umieszczany w niemal wszystkich opisanych wyżej znacznikach, dając tym samym bardzo duże możliwości wpływania na ich konstrukcję. Język arkusza stylów obowiązujący dla stylu jednowierszowego, zdefiniowanego za pomocą parametru `style`, jest zgodny z językiem domyślnym.

Sposób tworzenia jednowierszowego stylu wyjaśnimy na przykładzie akapitu :

```
<P style="font-weight: bold; font-size: 10pt; color: blue">
  Czyż style nie są wspaniałe ?
</P>
```

Jak widać, deklaracja parametru `style` składa się z pojedynczych składników `"nazwa_artybutu : wartość"` oddzielonych średnikami.
Dzięki parametrowi `style` bez potrzeby korzystania z dodatkowych znaczników zdefiniowaliśmy parametry dla tekstu całego akapitu. Czyż nie jest warto stosować jednowierszowe style ? Z pewnością tak !

2.11.3. Definiowanie arkusza stylów (znacznik STYLE)

Innym sposobem tworzenia arkuszy stylów jest umieszczanie ich treści w nagłówku dokumentu HTML pomiędzy znacznikami `<STYLE>` i `</STYLE>`.

Znacznik `<STYLE>` posiada następujące parametry:

- `type` - Określenie języka stylów dla arkuszy stylów,
  np. `<STYLE type="text/css">`
- `media` - Określenie docelowego medium dla stylu.
  Wartością domyślną jest `screen`.

A teraz przykład tworzenia stylu:

```
<HTML>
<HEAD>
  <STYLE type="text/css">
    P {border-width: 1; border: solid; font-size: 18pt;}
  </STYLE>
</HEAD>

<BODY>
  <P>Style są extra</P>
</BODY>
</HTML>
```

Załadowanie do przeglądarki powyższego dokumentu spowoduje wypisanie tekstu `Style są extra` domyślnym krojem czcionki o rozmiarze 18 punktów, otoczonego ramką narysowaną linią ciągłą.

Możemy również zdefiniować style dla kilku elementów jednocześnie:

```
<STYLE type="text/css">
   P {font-size: 12pt; font-family: arial;}
   H1 {color: red;}
</STYLE>
```

Należy pamiętać o tym, że CSS jest bardzo wrażliwy na składnię, dlatego też należy bardzo uważnie tworzyć arkusze.

2.11.4. Zewnętrzne pliki z arkuszami stylów (znacznik LINK)

Arkusze stylów nie muszą być zdefiniowane w tym samym pliku, w którym znajduje się nasz dokumentu HTML. Ich treść możemy umieścić w osobnym pliku, który następnie będzie ładowany do przeglądarki razem z konkretnym dokumentem.

Przykładowa treść pliku ze stylami, który nazwiemy np. `mojestyle.css` może wyglądać tak:

```
P.male {font-size:  8pt; font-family: verdana;}
P.duze {font-size: 20pt; font-family: verdana;}
H1 {color: green;}
H2 {color: gray;}
H3 {color: silver;}
```

Do włączenia tego pliku do naszego dokumentu HTML użyjemy znacznika `<LINK>`, który należy umieścić w nagłówku dokumentu:

```
<HEAD>
  ...
  <LINK rel="stylesheet" type="text/css" href="mojestyle.css">
  ...
</HEAD>
```

Znacznik `<LINK>` posiada następujące parametry:

- `href` - Lokalizacja pliku zwierającego arkusze stylów (nazwa pliku).
- `type` - Określenie języka przyłączanego pliku z arkuszami stylów.

Arkusz stylów możemy uczynić stałym, preferowanym lub alternatywnym. Do tego celu używa się dwóch parametrów `rel` i `title` :

- aby arkusz stylów uczynić stałym, należy parametrowi `rel` przypisać wartość `"stylesheet"` oraz nie ustawiać parametru `title`, np.:
  ```
  <LINK href="styl.css" type="text/css" rel="stylesheets">
  ```
- aby arkusz stylów uczynić preferowanym, należy parametrowi `rel` przypisać wartość `"stylesheet"` a parametrowi `title` nazwę, np.:
  ```
  <LINK href="styl.css" type="text/css" rel="stylesheets"
        title="compact">
  ```
- aby arkusz stylów uczynić alternatywnym, należy parametrowi `rel` przypisać wartość `"alternate stylesheet"` a parametrowi `title` nazwę, np.:
  ```
  <LINK href="styl.css" type="text/css" rel="alternate stylesheets"
        title="compact">
  ```

2.12. Skrypty

W zasadzie do tworzenia prostych stron WWW wystarczy nam czysty język HTML. Problem pojawia się w momencie, gdy chcemy naszą stronę uczynić w pełni dynamicznym dokumentem, obsługującym m.in. zdarzenia pochodzące od myszy czy klawiatury. W takiej sytuacji HTML jest niewystarczający i dlatego musimy posłużyć się językiem skryptowym np. językiem JavaScript czy VBScript, który wspomaga tworzenie stron w języku HTML.

2.12.1. Znaczniki SCRIPT i NOSCRIPT

Programy napisane w języku skryptowym możemy umieścić w dokumencie HTML za pomocą znaczników `<SCRIPT>` i `</SCRIPT>`.

Najważniejszymi parametrami znacznika <SCRIPT> są :

- language - Rodzaj skryptowego języka programowania, np. JavaSript czy VBScript, np. <SCRIPT language="JavaScript">
- src - Adres URI zewnętrznego dokumentu (pliku) zawierającego kod źródłowy napisany w języku skryptowym, np. <script language="JavaScript" src="funkcje.js">
- type - Określenie rodzaju języka skryptowego. Parametr ten nie posiada wartości domyślnej, należy wiec ją przypisać samodzielnie, np. <SCRIPT type="text/javascript">

W pierwszym przykładzie treść skryptu umieścimy w nagłówku. Zdefiniujemy tam funkcję zmieniającą kolor tła dla naszego dokumentu. Funkcja wywoływana będzie po wciśnięciu przycisku „Zmiana koloru":

```
<html>
<head>
  <title>Skrypty...</title>
  <script type="text/vbscript">
    Sub Kolor()
      document.BGCOLOR = "AA99AA"
    End Sub
  </script>
</head>
<body>
  <form>
    <input type="button" value="Zmiana koloru" name="B1"
           onClick="Kolor()">
  </form>
</body>
</html>
```

W kolejnym przykładzie skrypt umieścimy w ciele dokumentu:

```
<html>
<head><title>Skrypty</title></head>
<body>
  <p>
  Wypunktowanie :<br>
  <script type="text/javascript">
    for(i = 1; i <= 5; i++) {
      document.write(i,".<br>");
    }
  </script>
  Koniec listy.
  </p>
</body>
</html>
```

W trzecim przykładzie fragment kodu napisanego w JavaScript umieścimy bezpośrednio w znaczniku tworzącym przycisk. Miniprogramik zadziała jak naciśniemy przycisk myszką:

```
<html>
<head><title>Skrypty</title></head>
<body>
```

```
  <p>Komunikat:
  <input type="button" value="Start" name="b1"
         onClick="alert('Hello !');">
  </p>
</body>
</html>
```

Podobnie jak to miało miejsce w przypadku arkuszy stylów, również dla skryptów możemy określić język domyślny. Robimy to poprzez umieszczenie w nagłówku następującego polecenia:

```
<META http-equiv="Content-Script-Type" content="typ">
```

gdzie parametr `content` może przyjąć następujące wartości (zamiast wpisanego ogólnie słowa `typ`): `text/tcl`, `text/vbscript` lub `text/javascript`, np.

```
<META http-equiv="Content-Script-Type" content="text/tcl">
```

Na zakończenie tego punktu powiemy sobie jeszcze o znacznikach `<NOSCRIPT>` i `</NOSCRIPT>`.

Wszystkie popularne, współczesne przeglądarki obsługują poprawnie skrypty, jednak należy liczyć się z tym, że nasz dokument może być przeglądany również w przeglądarkach nie obsługujących skryptów. W takich sytuacjach z pomocą idą nam wspomniane znaczniki, za pomocą których możemy wykonać pewne czynności, o ile przeglądarka nie będzie potrafiła wykonać naszych skryptów.

```
<SCRIPT type="text/javascript">
... treść skryptu ...
</SCRIPT>
<NOSCRIPT>
... treść alternatywna nie zawierająca skryptów ...
</NOSCRIPT>
```

Często stosuje się również ukrywanie treści skryptu przed przeglądarkami nie potrafiącymi obsługiwać ich. Robi się to za pomocą komentarzy, np. tak:

- dla **JavaScript**

```
<SCRIPT type="text/javascript">
<!-- ukrywamy skrypt przed starszymi przeglądarkami
... treść skryptu ...
// koniec ukrywania -->
</SCRIPT>
```

- dla **VBScript**

```
<SCRIPT type="text/vbscript">
<!— ukrywamy skrypt przed starszymi przeglądarkami
... treść skryptu ...
‘ koniec ukrywania -->
</SCRIPT>
```

- dla **TCL**

```
<SCRIPT type="text/tcl">
<!-- ukrywamy skrypt przed starszymi przeglądarkami
... treść skryptu ...
# koniec ukrywania -->
</SCRIPT>
```

2.12.2. Dołączanie zewnętrznych plików ze skryptami

Podobnie jak to miało miejsce w przypadku arkuszy stylów, również skrypty możemy umieszczać w oddzielnych plikach, które możemy następnie dołączyć do dokumentu (o ile zajdzie taka konieczność).
Aby treść skryptu włączyć do dokumentu musimy posłużyć się następującą konstrukcją:

```
<SCRIPT type="typ" src="adres_pliku"></SCRIPT>
```

gdzie `typ` oznacza rodzaj języka (`text/tcl`, `text/vbscript`, `text/javascript`), natomiast `adres_pliku` adres URI zewnętrznego dokumentu (pliku) zawierającego kod źródłowy napisany w języku skryptowym, np.:

- treść zewnętrznego pliku `data.js` zawierającego skrypt napisany w JavaScript:

```
var Data  = new Date();
var Rok   = Data.getYear();
    Rok = Rok < 2000 ? 1900 + Rok : Rok;
    document.write(Data.getDate(),"-",Data.getMonth()+1,"-",Rok);
```

- treść głównego dokumentu HTML, do którego włączamy plik `data.js` :

```
<head>
  <title>Skrypty</title>
  <script type="text/javascript" src="data.js"></script>
</head>

  ... treść dokumentu ...
</body>
```

Dodatek I – A
Indeks znaczników HTML 4.0

Stosowanie znacznika: O = Opcjonalne, Z = Zabronione, N = Niezalecane

Nazwa znacznika	Znacznik początkowy	Znacznik końcowy	Znacznik niezalecany
A			
ABBR			
ACRONYM			
ADDRESS			
APPLET			N
AREA		Z	
B			
BASE		Z	
BASEFONT		Z	N
BDO			
BIG			
BLOCKQUOTE			
BODY	O	O	
BR		Z	
BUTTON			
CAPTION			
CENTER			N
CITE			
CODE			
COL		Z	
COLGROUP		O	
DD		O	
DEL			
DFN			
DIR			N
DIV			
DL			
DT		O	
EM			
FIELDSET			
FONT			N
FORM			
FRAME		Z	
FRAMESET			
H1			
H2			
H3			
H4			
H5			
H6			
HEAD	O	O	
HR		Z	
HTML	O	O	
I			

IFRAME			
IMG		Z	
INPUT		Z	
INS			
ISINDEX		Z	N
KBD			
LABEL			
LEGEND			
LI		O	
LINK		Z	
MAP			
MENU			N
META		Z	
NOFRAMES			
NOSCRIPT			
OBJECT			
OL			
OPTGROUP			
OPTION		O	
P		O	
PARAM		Z	
PRE			
Q			
S			N
SAMP			
SCRIPT			
SELECT			
SMALL			
SPAN			
STRIKE			N
STRONG			
STYLE			
SUB			
SUP			
TABLE			
TBODY	O	O	
TD		O	
TEXTAREA			
TFOOT		O	
TH		O	
THEAD		O	
TITLE			
TR		O	
TT			
U			N
UL			
VAR			

Dodatek I – B
Znaki specjalne poprzedzone & (ampersand)

Kod	Encja	Znak	Opis
			Znak tabulacji
		Znak nowego wiersza
		Powrót karetki
			Spacja
!		!	Wykrzyknik
"	"	″	Cudzysłów prosty
#		#	Krzyżyka (hash)
$		$	Dolara
%		%	Procenta
&	&	&	Znak ampersand
'		′	Apostrof prosty
(		(	Lewy nawias okrągły
)		)	Prawy nawias okrągły
*		*	Gwiazdka
+		+	Plus
,		,	Przecinek
-		-	Minus (łącznik)
.		.	Kropka
/		/	Ukośnik prawy (slash)
0 ÷ &#057		0 ÷ 9	Cyfry 0 ÷ 9
:		:	Dwukropek
;		;	Średnik
<	<	<	Znak mniejszości
=		=	Znak równości
>	>	>	Znak większości
?		?	Znak zapytania
@		@	Znak *at* („małpka”)
A ÷ &#090		A ÷ Z	Litery A ÷ Z
[		[	Lewy nawias kwadratowy
\		\	Ukośnik lewy (backslash)
]		]	Prawy nawias kwadratowy
^		^	Daszek (dash, control)
_		_	Kreska podkreślenia
`		`	Akcent słaby (grave)
a ÷ z		a ÷ z	Litery a ÷ z
{		{	Lewy nawias klamrowy
			|	Kreska pionowa
}		}	Prawy nawias klamrowy
~		~	Tylda
‚		‚	Przecinek
ƒ		*ƒ*	Symbol florena
„		”	Cudzysłów górny zamykający
…		...	Wielokropek
†		†	Znak krzyża (dagger)
‡		‡	Znak podwójnego krzyża (double dagger)
ˆ		ˆ	Akcent cyrkumfleksowy (circumflex)
‰		‰	Promil
Š		_	Podkreślenie
‹		<	Znak mniejszości
Œ		Œ	Ligatura OE
‘		‘	Lewy apostrof drukarski
’		’	Prawy apostrof drukarski

&#147;		“	Lewy cudzysłów drukarski
&#148;		”	Prawy cudzysłów drukarski
&#149;		•	Znak wypunktowania
&#150;		–	Półpauza (n-dash)
&#151;		—	Pauza, myślnik (m-dash)
&#152;		˜	Tylda
&#153;		™	Znak towarowy (trademark)
&#154;			Podkreślenie
&#155;		>	Znak większości
&#156;			Ligatura oe
&#159;			Duże Y z diaerezą (umlaut)
			Twarda spacja
&#161;	&iexcl;	¡	Odwrócony wykrzyknik
&#162;	&cent;	¢	Symbol centa
&#163;	&pound;	£	Symbol funta szterlinga
&#164;	&curren;	¤	Ogólny symbol waluty
&#165;	&yen;	¥	Symbol jena
&#166;	&brvbar;	¦	Przerwana linia pionowa
&#167;	&sect;	§	Znak sekcji lub paragrafu
&#168;	&uml;	¨	Diaereza (umlaut)
&#169;	&copy;	©	Symbol *copyright*
&#170;	&ordf;	ª	Liczebnik porządkowy żeński
&#171;	&laquo;	«	Lewy cudzysłów ostrokątny
&#172;	&not;	¬	Znak negacji
&#173;	&shy;	-	Łącznik opcjonalny
&#174;	&reg;	®	Zastrzeżony znak towarowy
&#175;	&macr;	¯	Kreska górna (macron)
&#176;	&deg;	°	Znak stopnia
&#177;	&plusmn;	±	Plus-minus
&#178;	&sup2;	²	2 w indeksie górnym
&#179;	&sup3;	³	3 w indeksie górnym
&#180;	&acute;	´	Akcent silny
&#181;	&micro;	µ	Grecka litera *mi* (mikro)
&#182;	&para;	¶	Znak akapitu
&#183;	&middot;	·	Punkt środkowy
&#184;	&cedil;	¸	Cedilla
&#185;	&sup1;	¹	1 w indeksie górnym
&#186;	&ordm;	º	Liczebnik porządkowy męski
&#187;	&raquo;	»	Prawy cudzysłów ostrokątny
&#188;	&frac14;	¼	Ułamek jedna czwarta
&#189;	&frac12;	½	Ułamek jedna druga
&#190;	&frac34;	¾	Ułamek trzy czwarte
&#191;	&iquest;	¿	Odwrócony znak zapytania
&#192;	&Agrave;	À	Duże A z akcentem słabym
&#193;	&Aacute;	Á	Duże A z akcentem silnym
&#194;	&Acirc;	Â	Duże A z daszkiem (akcentem cyrkumfleksowym)
&#195;	&Atilde;	Ã	Duże A z tyldą
&#196;	&Auml;	Ä	Duże A z diaerezą (umlaut)
&#197;	&Aring;	Å	Duże A z kółkiem
&#198;	&Aelig;	Æ	Ligatura AE
&#199;	&Ccedil;	Ç	Duże C, cedilla
&#200;	&Egrave;	È	Duże E z akcentem słabym
&#201;	&Eacute;	É	Duże E z akcentem silnym
&#202;	&Ecirc;	Ê	Duże E z daszkiem
&#203;	&Euml;	Ë	Duże E z diaerezą (umlaut)
&#204;	&Igrave;	Ì	Duże I z akcentem słabym
&#205;	&Iacute;	Í	Duże I z akcentem silnym
&#206;	&Icirc;	Î	Duże I z daszkiem

&#207;	&Iuml;	Ï	Duże I z diaerezą (umlaut)
&#208;	&ETH;	Đ	Duże islandzkie *eth*
&#209;	&Ntilde;	Ñ	Duże N z tyldą
&#210;	&Ograve;	Ò	Duże O z akcentem słabym
&#211;	&Oacute;	Ó	Duże O z akcentem silnym
&#212;	&Ocirc;	Ô	Duże O z daszkiem
&#213;	&Otilde;	Õ	Duże O z tyldą
&#214;	&Ouml;	Ö	Duże O z diaerezą (umlaut)
&#215;	&times;	×	Znak iloczynu
&#216;	&Oslash;	Ø	Duże O z przekreśleniem
&#217;	&Ugrave;	Ù	Duże U z akcentem słabym
&#218;	&Uacute;	Ú	Duże U z akcentem silnym
&#219;	&Ucirc;	Û	Duże U z daszkiem
&#220;	&Uuml;	Ü	Duże U z diaerezą (umlaut)
&#221;	&Yacute;	Ý	Duże Y z akcentem silnym
&#222;	&THORN;	Þ	Duże islandzkie *thorn*
&#223;	&szlig;	ß	Małe *scharfes s*
&#224;	&agrave;	à	Małe a z akcentem słabym
&#225;	&aacute;	á	Małe a z akcentem silnym
&#226;	&acirc;	â	Małe a z daszkiem
&#227;	&atilde;	ã	Małe a z tyldą
&#228;	&auml;	ä	Małe a z diaerezą (umlaut)
&#229;	&aring;	å	Małe a z kółkiem
&#230;	&aelig;	æ	Mała ligatura ae
&#231;	&ccedil;	ç	Małe c z cedillą
&#232;	&egrave;	è	Małe e z akcentem słabym
&#233;	&eacute;	é	Małe e z akcentem silnym
&#234;	&ecirc;	ê	Małe e z daszkiem
&#235;	&euml;	ë	Małe e z diaerezą (umlaut)
&#236;	&eth;	ì	Małe i z akcentem słabym
&#237;	&igrave;	í	Małe i z akcentem silnym
&#238;	&iacute;	î	Małe i z daszkiem
&#239;	&iuml;	ï	Małe i z diaerezą (umlaut)
&#240;		ð	Małe islandzkie *eth*
&#241;	&ntilde;	ñ	Małe n z tyldą
&#242;	&ograve;	ò	Małe o z akcentem słabym
&#243;	&oacute;	ó	Małe o z akcentem silnym
&#244;	&ocirc;	ô	Małe o z daszkiem
&#245;	&otilde;	õ	Małe o z tyldą
&#246;	&ouml;	ö	Małe o z diaerezą (umlaut)
&#247;	&divide;	÷	Znak ilorazu
&#248;	&oslash;	ø	Małe o z przekreśleniem
&#249;	&ugrave;	ù	Małe u z akcentem słabym
&#250;	&uacute;	ú	Małe u z akcentem silnym
&#251;	&ucirc;	û	Małe u z daszkiem
&#252;	&uuml;	ü	Małe u z diaerezą (umlaut)
&#253;	&yacute;	ý	Małe y z akcentem silnym
&#254;	&thorn;	þ	Małe islandzkie thorn
&#255;	&yuml;	ÿ	Małe y z diaerezą (umlaut)

Dodatek I – C
Kolory

Nazwa koloru	Wartości szesnastkowe			Wartości dziesiętne		
	R	G	B	R	G	B
ALICEBLUE	F0	F8	FF	240	248	255
ANTIQUEWHITE	FA	EB	D7	250	235	215
AQUA	00	FF	FF	0	255	255
AQUAMARINE	7F	FF	D4	127	255	212
AZURE	F0	FF	FF	240	255	255
BEIGE	F5	F5	DC	245	245	220
BISQUE	FF	E4	C4	255	228	196
BLACK	00	00	00	0	0	0
BLANCHEDALMOND	FF	EB	CD	255	235	205
BLUE	00	00	FF	0	0	255
BLUEVIOLET	8A	2B	E2	138	43	226
BROWN	A5	2A	2A	165	42	42
BURLYWOOD	DE	B8	87	222	184	135
CADETBLUE	5F	9E	A0	95	158	160
CHARTREUSE	7F	FF	00	127	255	0
CHOCOLATE	D2	69	1E	210	105	30
CORAL	FF	7F	50	255	127	80
CORNFLOWER	64	95	ED	100	149	237
CORNSILK	FF	F8	DC	255	248	220
CRIMSON	DC	14	3C	220	20	60
CYAN	00	FF	FF	0	255	255
DARKBLUE	00	00	8B	0	0	139
DARKCYAN	00	8B	8B	0	139	139
DARKGOLDENROD	B8	86	0B	184	134	11
DARKGRAY	A9	A9	A9	169	169	169
DARKGREEN	00	64	00	0	100	0
DARKKHAKI	BD	B7	6B	189	183	107
DARKMAGENTA	8B	00	8B	139	0	139
DARKOLIVEGREEN	55	6B	2F	85	107	47
DARKORANGE	FF	8C	00	255	140	0
DARKORCHID	99	32	CC	153	50	204
DARKRED	8B	00	00	139	0	0
DARKSALMON	E9	96	7A	233	150	122
DARKSEAGREEN	8F	BC	8B	143	188	139
DARKSLATEBLUE	48	3D	8B	72	61	139
DARKSLATEGREY	2F	4F	4F	47	79	79
DARKTURQUOISE	00	CE	D1	0	206	209
DARKVIOLET	94	00	D3	148	0	211
DEEPPINK	FF	14	93	255	20	147
DEEPSKYBLUE	00	BF	FF	0	191	255
DIMGRAY	69	69	69	105	105	105
DODGERBLUE	1E	90	FF	30	144	255
FIREBRICK	B2	22	22	178	34	34
FLORALWHITE	FF	FA	F0	255	250	240
FORESTGREEN	22	8B	22	34	139	34
FUCHIA	FF	00	FF	255	0	255
GAINSBORO	DC	DC	DC	220	220	220
GHOSTWHITE	F8	F8	FF	248	248	255
GOLD	FF	D7	00	255	215	0

GOLDENROD	DA	A5	20	218	165	32
GRAY	80	80	80	128	128	128
GREEN	00	80	00	0	128	0
GREENYELLOW	AD	FF	2F	173	255	47
HONEYDEW	F0	FF	F0	240	255	240
HOTPINK	FF	69	B4	255	105	180
INDIANRED	CD	5C	5C	205	92	92
INDIGO	4B	00	82	75	0	130
IVORY	FF	FF	F0	255	255	240
KHAKI	F0	E6	8C	240	230	140
LAVENDER	E6	E6	FA	230	230	250
LAVENDERBLUSH	FF	F0	F5	255	240	245
LAWNGREEN	7C	FC	00	124	252	0
LEMONCHIFFON	FF	FA	CD	255	250	205
LIGHTBLUE	AD	D8	E6	173	216	230
LIGHTCORAL	F0	80	80	240	128	128
LIGHTCYAN	E0	FF	FF	224	255	255
LIGHTGOLDENRODYELLOW	FA	FA	D2	250	250	210
LIGHTGREEN	90	EE	90	144	238	144
LIGHTGREY	D3	D3	D3	211	211	211
LIGHTPINK	FF	B6	C1	255	182	193
LIGHTSALMON	FF	A0	7A	255	160	122
LIGHTSEAGREEN	20	B2	AA	32	178	170
LIGHTSKYBLUE	87	CE	FA	135	206	250
LIGHTSLATEGRAY	77	88	99	119	136	153
LIGHTSTEELBLUE	B0	C4	DE	176	196	222
LIGHTYELLOW	FF	FF	E0	255	255	224
LIME	00	FF	00	0	255	0
LIMEGREEN	32	CD	32	50	205	50
LINEN	FA	F0	E6	250	240	230
MAGENTA	FF	00	FF	255	0	255
MAROON	80	00	00	128	0	0
MEDIUMAQUAMARINE	66	CD	AA	102	205	170
MEDIUMBLUE	00	00	CD	0	0	205
MEDIUMORCHID	BA	55	D3	186	85	211
MEDIUMPURPLE	93	70	DB	147	112	219
MEDIUMSEAGREEN	3C	B3	71	60	179	113
MEDIUMSLATEBLUE	7B	68	EE	123	104	238
MEDIUMSPRINGGREEN	00	FA	9A	0	250	154
MEDIUMTURQUOISE	48	D1	CC	72	209	204
MEDIUMVIOLETRED	C7	15	85	199	21	133
MIDNIGHTBLUE	19	19	70	25	25	112
MINTCREAM	F5	FF	FA	245	255	250
MISTYROSE	FF	E4	E1	255	228	225
MOCCASIN	FF	E4	B5	255	228	181
NAVAJOWHITE	FF	DE	AD	255	222	173
NAVY	00	00	80	0	0	128
OLDLACE	FD	F5	E6	253	245	230
OLIVE	80	80	00	128	128	0
OLIVEDRAB	6B	8E	23	107	142	35
ORANGE	FF	A5	00	255	165	0
ORANGERED	FF	45	00	255	69	0
ORCHID	DA	70	D6	218	112	214
PALEGOLDENROD	EE	E8	AA	238	232	170
PALEGREEN	98	FB	98	152	251	152
PALETURQUOISE	AF	EE	EE	175	238	238

PALEVIOLETRED	DB	70	93	219	112	147
PAPAYAWHIP	FF	EF	D5	255	239	213
PEACHPUFF	FF	DA	B9	255	218	185
PERU	CD	85	3F	205	133	63
PINK	FF	C0	CB	255	192	203
PLUM	DD	A0	DD	221	160	221
POWDERBLUE	B0	E0	E6	176	224	230
PURPLE	80	00	80	128	0	128
RED	FF	00	00	255	0	0
ROSYBROWN	BC	8F	8F	188	143	143
ROYALBLUE	41	69	E1	65	105	225
SADDLEBROWN	8B	45	13	139	69	19
SALMON	FA	80	72	250	128	114
SANDYBROWN	F4	A4	60	244	164	96
SEAGREEN	2E	8B	57	46	139	87
SEASHELL	FF	F5	EE	255	245	238
SIENNA	A0	52	2D	160	82	45
SILVER	C0	C0	C0	192	192	192
SKYBLUE	87	CE	EB	135	206	235
SLATEBLUE	6A	5A	CD	106	90	205
SLATEGRAY	70	80	90	112	128	144
SNOW	FF	FA	FA	255	250	250
SPRINGGREEN	00	FF	7F	0	255	127
STEELBLUE	46	82	B4	70	130	180
TAN	D2	B4	8C	210	180	140
TEAL	00	80	80	0	128	128
THISTLE	D8	BF	D8	216	191	216
TOMATO	FF	63	47	255	99	71
TURQUOISE	40	E0	D0	64	224	208
VIOLET	EE	82	EE	238	130	238
WHEAT	F5	DE	B3	245	222	179
WHITE	FF	FF	FF	255	255	255
WHITESMOKE	F5	F5	F5	245	245	245
YELLOW	FF	FF	00	255	255	0
YELLOWGREEN	9A	CD	32	154	205	50

Przykład opisywania koloru w HTML-u :

```
<P>
  <FONT color="YELLOWGREEN"> Tekst żółto-zielony</FONT></BR>
  <HR color="SILVER">
  <FONT color="#9ACD32"> Tekst żółto-zielony</FONT></BR>
</P>
```

II. Arkusze stylów CSS (Cascading Style Sheets)

1. Co to jest CSS ?

CSS (Cascading Style Sheets) jest mechanizmem, za pomocą którego możemy tworzyć kaskadowe arkusze stylów na potrzeby dokumentów internetowych. Kaskadowe arkusze stylów dają możliwość definiowania wielu atrybutów dla elementów oraz określonych fragmentów dokumentu. Dzięki nim zachowanie jednolitego stylu dla całego serwisu WWW (składającego się nawet z wielu części) jest proste i wygodne.
Znaczenie arkuszy stylów podkreśla fakt, iż W3C[1] zaleca ich stosowanie w miejsce przestarzałych znaczników i parametrów języka HTML.

2. Pojęcia podstawowe

2.1. Składnia CSS

Tworzenie prostych arkuszy stylów nie jest skomplikowane i wymaga jedynie zapamiętania kilku podstawowych zasad oraz podstaw z programowania w HTML.

Aby lepiej omówić składnię CSS zdefiniujemy przykładowy styl, w którym dla tekstu akapitu ustalimy kolor czerwony:

```
P { color: red }
```

Taką definicję możemy podzielić na dwie główne części: typ elementu `P` oraz umieszczoną w nawiasach klamrowych deklarację `color: red`, gdzie `color` jest właściwością elementu, a `red` jej wartością.

2.2. CSS w dokumencie HTML

Arkusze stylów możemy umieścić w dokumencie HTML na cztery sposoby:

- przez umieszczenie arkusza stylów za pomocą znacznika `<STYLE>` w nagłówku dokumentu HTML,
- przez włączenie zewnętrznego pliku ze stylami do dokumentu za pomocą znacznika `<LINK>` (również w nagłówku),
- przez włączenie zewnętrznego pliku ze stylami do dokumentu za pomocą polecenia `@import`,
- przez utworzenie jednowierszowego stylu za pomocą parametru `style`.

W poniższym przykładzie wykorzysam wszystkie wymienione metody umieszczania arkuszy stylów w dokumencie HTML:

```
<HTML>
  <HEAD>
    <TITLE>CSS w dokumencie</TITLE>
```

[1] W3C – World Wide Web Constortium

```
    <LINK rel="stylesheet" type="text/css" href="mojestyle.css">
    <STYLE type="text/css">
      @import url(akapit.css);
      H1 {color: silver}
    </STYLE>
  </HEAD>

  <BODY>
    <P>Akapit</P>
    <HR>
    <P style="color: green">Akapit - styl jednowierszowy</P>
    <H1>Znacznik H1</H1>
    <P>Akapit</P>
  </BODY>
</HTML>
```

- **Znacznik <STYLE>**

Wprawdzie znacznik `<STYLE>` został już omówiony w części poświęconej językowi HTML w pkt. 2.11.3, jednak przypomnę jego najważniejsze parametry:

- `type` - określenie języka stylów dla arkuszy stylów, np. `<STYLE type="text/css">`,
- `media` - określenie docelowego medium dla stylu, którym może być np. ekran telewizyjny, projektor, drukarka, przeglądarki wykorzystujące dźwięki itp. Wartością domyślną jest `screen`.

Dzięki parametrowi `media` możemy sterować właściwościami elementów prezentowanego dokumentu w zależności od rodzaju medium, na które dokument jest kierowany.

Przykładowo przyjmijmy, że dla dokumentu skierowanego do przeglądarki, nagłówki poziomu 1 będą czerwone i wyrównywane domyślnie (do lewej krawędzi), natomiast w przypadku skierowania tego samego dokumentu na drukarkę, nagłówki będą wyśrodkowane na stronie.

Całość zapiszemy tak:

```
<HEAD>
  <TITLE>CSS a media</TITLE>
  <STYLE type="text/css" media="screen">
    H1 {color: red}
  </STYLE>

  <STYLE type="text/css" media="print">
    H1 {text-align: center}
  </STYLE>
</HEAD>
```

przy czym w pierwszej definicji parametru `media="screen"` mogłoby nie być, gdyż jest on domyślny.

Za pomocą parametru `media` możemy również dodawać efekty dźwiękowe, jednak ma to sens jedynie dla przeglądarek obsługujących tę usługę:

```
<STYLE type=text/css" media="aural">
    A {cue-before: uri(tada.wav); cue-after: uri(plum.wav)}
</STYLE>
```

- **Dołączanie zewnętrznych arkuszy stylów znacznikiem <LINK>**

Znacznik `<LINK>` podobnie jak `<STYLE>` został opisany w części poświęconej językowi HTML w pkt. 2.11.4, jednak i w tym przypadku pozwolę sobie przypomnieć jego najważniejsze parametry:

- `href` - lokalizacja pliku zwierającego arkusze stylów (nazwa pliku),
- `type` - określenie języka przyłączanego pliku z arkuszami stylów.

Arkusz stylów możemy uczynić stałym, preferowanym lub alternatywnym. Do tego celu używa się dwóch parametrów `rel` i `title` :

- aby arkusz stylów uczynić stałym, należy parametrowi `rel` przypisać wartość `"stylesheet"` oraz nie ustawiać parametru `title`, np.:
  ```
  <LINK href="styl.css" type="text/css" rel="stylesheets">
  ```
- aby arkusz stylów uczynić preferowanym, należy parametrowi `rel` przypisać wartość `"stylesheet"` a parametrowi `title` nazwę, np.:
  ```
  <LINK href="styl.css" type="text/css" rel="stylesheets"
        title="compact">
  ```
- aby arkusz stylów uczynić alternatywnym, należy parametrowi `rel` przypisać wartość „`alternate stylesheet"` a parametrowi `title` nazwę, np.:
  ```
  <LINK href="styl.css" type="text/css" rel="alternate stylesheets"
        title="compact">
  ```

Arkusze stylów importowane za pomocą znacznika `<LINK>` są automatycznie spajane z arkuszami stylów zdefiniowanymi bezpośrednio w dokumencie. Jeżeli więc plik `style.css` zawierający definicję stylów o treści:

```
P {color: blue}
```

włączymy do dokumentu w następujący sposób,

```
<HEAD>
  <LINK href="style.css" type="text/css" rel="stylesheets">
  <STYLE type="text/css">
    H1 { color: green }
  </STYLE>
</HEAD>
```

to w efekcie nasz wynikowy arkusz stylów będzie wyglądał tak:

```
P {color: blue}
H1 {color: green}
```

Jeżeli natomiast plik ze stylami będzie następującej treści,

```
H1 {color: red}
```

to obowiązującym kolorem dla nagłówków pierwszego poziomu będzie kolor zielony, ponieważ ostatnia deklaracja `color:green` zasłoni deklarację z pliku `color:red`.

- **Dołączanie zewnętrznych arkuszy stylów poleceniem @import**

Zewnętrzne pliki z arkuszami stylów możemy również włączać do dokumentu HTML za pomocą polecenia `@import`, typowego dla CSS.

```
<HEAD>
  <STYLE type="text/css">
    @import url(style.css);
    ...
  </STYLE>
</HEAD>
```

Nazwa pliku z arkuszami stylów może być również pełnym adresem URL, np. `www.mojadres.pl/style.css`. Pozostałe zasady są takie same, jak w przypadku włączania stylów znacznikiem `<LINK>`.

- **Style jednowierszowe**

Style możemy również tworzyć bezpośrednio dla konkretnych elementów znajdujących się w ciele dokumentu (pomiędzy znacznikami `<BODY>` i `</BODY>`) za pomocą parametru `style`. Oto kilka przykładów :

```
<P style="color: blue">tekst akapitu...</P>
<HR style="color: red">
<H2 style="font-size: 20pt">Nagłówek</H2>
```

Ten sposób tworzenia styli jest bardzo szybki i wygodny, i dlatego warto pamiętać o parametrze `style`.

2.3. Komentarze

Komentarze w CSS składnią przypominają konstrukcje znane z języka programowania C. Komentarz taki rozpoczynamy znakami `/*` a kończymy `*/`, np.:

```
P { color: blue }  /* Kolor tekstu w akapicie będzie niebieski */
```

Komentarzy takich nie można zagnieżdżać.

Mówiąc o komentarzach warto również w tym miejscu wspomnieć o jeszcze jednej, bardzo istotnej rzeczy. Otóż starsze przeglądarki ignorują znaczniki, których nie znają. Również znacznik `<STYLE>` może zostać zignorowany. W takiej sytuacji treść umieszczona pomiędzy znacznikami `<STYLE>` i `</STYLE>` będzie traktowana jako część ciała dokumentu, co jest oczywiście bardzo niepożądane. Aby ustrzec się przed takimi błędami, treść arkusza stylów możemy ukryć przed starszymi przeglądarkami za pomocą komentarzy znanych z języka HTML:

```
<STYLE type="text/css">
<!--
  P {color: blue}
-->
</STYLE>
```

Jak to widać w powyższym przykładzie, komentarz rozpoczynamy łańcuchem `<!--` a kończymy `-->`. Wszystko, co znajduje się pomiędzy nimi, nie będzie interpretowane przez starsze przeglądarki, nie obsługujące znacznika `<STYLE>`.

2.4. Typy danych prostych (wartości)

2.4.1. Rozmiar

Rozmiar możemy określać używając jednostek zdefiniowanych w grupie jednostek relatywnych (względnych) lub absolutnych.

Jednostki relatywne (względne):
- **em** - wielokrotność obowiązującego (domyślnego) rozmiaru czcionki,
- **ex** - wielokrotność obowiązującego (domyślnego) rozmiaru *x-height*[1] czcionki,
- **px** - piksele.

Jednostki absolutne:
- **in** - cale (1 cal jest równy 2,54 cm),
- **cm** - centymetry,
- **mm** - milimetry,
- **pt** - punkty (punkt w CSS2 jest równy 1/72 cala),
- **pc** - picas (1 picas jest równy 12 punktom).

Poniższa deklaracja:

```
H1 { font-size: 1.5em }
```

ustala rozmiar czcionki dla nagłówków poziomu pierwszego o 50% większy od rozmiaru obowiązującego. Z kolei deklaracja

```
H1 { font-size: 12px }
```

ustali rozmiar czcionki na 12 pikseli.

Absolutne jednostki rozmiaru stosuje się wtedy, gdy znane są fizyczne parametry medium wyjściowego dla naszego dokumentu.

2.4.2. Procenty

Format wartości procentowej jest typowy, czyli najpierw podajemy liczbę, a następnie znak `%`, np. `120%`.

```
P { font-size: 200% }
```

Powyższa deklaracja ustali rozmiar czcionki tekstu akapitu na 2 razy większy od rozmiaru obowiązującego.

[1] *x-height* - porównywany jest z wysokością małej litery *x* (domyślnej czcionki).

2.4.3. Kolory

Kolory możemy opisywać za pomocą predefiniowanych nazw lub reprezentacji numerycznej składowych RGB. W HTML 4.0 zdefiniowano 16 nazw kolorów podstawowych: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white` i `yellow`.

```
H1 { color: silver }
BODY { color: black; background: white }
P { color: red }
```

Używając składowych RGB, kolor możemy opisać na cztery sposoby. Każdy z nich przedstawię na przykładzie koloru czerwonego (`red`):

```
P {color: #f00}              /* #rgb */
P {color: #ff0000}           /* #rrggbb */
P {color: rgb(255,0,0)}      /* zakres całkowity 0 – 255 */
P {color: rgb(100%,0%,0%)}   /* zakres rzeczywisty 0.0% - 100.0% */
```

Postać `#rgb` jest zawsze konwertowana do postaci `#rrggbb`, zatem przykładowa wartość `#bf0` zostanie rozszerzona do `#bbff00`.
Aby przedstawić zależności zachodzące pomiędzy wartościami wykorzystywanymi w notacjach jak wyżej, użyję zapisu: `F = FF = 255 = 100.0%`.

2.4.4. Łańcuchy (napisy)

Do wyróżnienia łańcuchów możemy używać cudzysłowów lub apostrofów. W łańcuchu umieszczonym pomiędzy cudzusłowami nie może występować swobodnie inny cudzysłów. Chcąc wstawić taki znak musimy poprzedzić go znakiem `\`. Ta sama zasada dotyczy apostrofów. Oto kilka przykładów:

```
"to jest napis"
'to jest napis'
"to jest 'napis'"
'to jets "napis"'
"to jest \"napis\""
'to jest \'napis\''
```

Aby łańcuch złamać i przenieść jego dalszą część do drugiego wiersza, musimy w miejscu złamania umieścić znak `\`, np.:

```
A[TITLE="to jest\
przykładowy napis"] {/*...*/}
```

Powyższy zapis jest równoważny zapisowi

```
A[TITLE="to jest przykładowy napis"] {/*...*/}
```

2.4.5. Adresy URL

URL (Uniform Resource Locators) dostarcza nam informacji o lokalizacji zasobu w sieci (jego adres).

Do wyznaczania adresu zasobu stosowana jest następująca notacja:

```
BODY { background: url(”http://www.mojadres.pl/marmur.gif”) }
```

Adres `http://www.mojadres.pl/marmur.gif` możemy umieścić pomiędzy apostrofami lub czudzysowami. Dopuszczalna jest również konstrukcja pozbawiona ich, np.:

```
<STYLE type=”text/css”>
  BODY { background: url('yellow.gif') }
  @import url(mojestyle1);
  @import url(”mojestyle2”);
</STYLE>
```

2.4.6. Czas

Jednostki czasu wykorzystywane są w dźwiękowych arkuszach stylów. Mamy do dyspozycji dwie jednostki:

- **ms** - milisekundy
- **s** - sekundy

Wartości nie mogą być ujemne.

2.4.7. Częstotliwość

Jednostki częstotliwości wykorzystywane są w dźwiękowych arkuszach stylów. Mamy do dyspozycji dwie jednostki:

- **Hz** - herce
- **kHz** - kilohece

Wartości nie mogą być ujemne.

2.4.8. Kąty

Jednostki wartości katów wykorzystywane są w dźwiękowych arkuszach stylów. Do wyrażania wartości kątów możemy używać jednostek:

- **deg** - stopień (jednostka kąta płaskiego ($1^o = \pi/180$ rad)
- **grad**- jednostka kąta płaskiego równa 0,01 kąta prostego ($1^g = 0^o = \pi/200$ rad)
- **rad** - radian (1 rad ≈ 57º14’14”)

Przykładowo kąt prosty zapiszemy jako `90deg`, `100grad` lub `1.57079632679489rad`.

Wartości kątów mogą być ujemne. Takie kąty są automatycznie konwertowane do wartości z zakresu `0÷360deg`. Przykładowo wartości `-30deg` i `330deg` są sobie równoważne.

2.5. Grupowanie

Jeżeli w arkuszu stylów kilka elementów posiada identycznie zdefiniowane właściwości, np.:

```
<STYLE type="text/css">
  H1 { color: blue }
  H2 { color: blue }
  P  { color: blue }
</STYLE>
```

to zapis ten możemy uprościć przez podanie po przecinkach wszystkich elementów, a za nimi w nawiasach klamrowych deklaracji właściwości, które są dla nich wspólne:

```
<STYLE type="text/css">
  H1, H2, P { color: red }
</STYLE>
```

Przez grupowanie możemy również rozumieć zebranie deklaracji wielu właściwości jednocześnie i przypisanie ich jednemu elementowi w sposób następujący:

```
<STYLE type="text/css">
  P {
      color       : red;
      font-family : sans-serif;
      font-size   : 14pt;
      font-weight : bold;
    }
</STYLE>
```

Poszczególne deklaracje oddzielamy średnikami. Jest to bardzo ważne, aby o nich nie zapominać, gdyż CSS jest bardzo wrażliwy na składnię.

2.6. Dziedziczenie

Z dziedziczeniem spotkaliśmy się w zasadzie już w trakcie omawiania języka HTML. Przykładem dzidziczenia właściwości po innym elemencie może być następujący fragment:

```
<P>Biedroneczki <B>są</B> w kropeczki!</P>
```

Tekst umieszczony pomiędzy znacznikami `<B>` i `</B>`, mimo iż jest pogrubiony, pozostałe atrybuty, tj. rozmiar, kolor czy krój czcionki, dziedziczy po `<P>`.
Mechanizm dziedziczenia właściwości elementów występuje również w części deklaracyjnej arkuszy stylów. Oto przykład:

```
<STYLE type="text/css">
  P { font-size: 14pt; background: yellow }
  P { line-height: 200% }
</STYLE>
```

W pierwszej deklaracji właściwości elementu `P`, zmieniamy domyślny rozmiar czcionki na 14pt i tło wiersza na kolor żółty. Kolejna deklaracja ustala wysokość wiersza. Dla tej

deklaracji istotne są dwie rzeczy. Po pierwsze – mimo, iż wartość właściwości `line-height` podana jest w procentach, to i tak zamieniana jest ona na jednostki rozmiaru. Po drugie – element „dziecko" (drugi element `P`) dziedziczy po swoim „rodzicu" (pierwszym elemencie `P`) wszystkie właściwości, a więc i rozmiar czcionki (wpływający na wysokość wiersza), co w konsekwencji spowoduje, że wartość właściwości `line-height` zostanie ustalona na 28pt (200% * 14pt).

W tym miejscu powiemy sobie również o sytuacjach, w których wartości właściwości elementu mogą być pokrywane. Wyjaśnię to na przykładzie:

```
<STYLE type=„text/css">
  P { color: red; font-size: 12pt }
  P { color: blue }
  ...
</STYLE>
```

W wierszu `P{color: red; font-size: 12pt}` ustalamy kolor tekstu akapitu na niebieski i rozmiar czcionki na 12pt. W kolejnym wierszu właściwości `color` elementu `P` ponownie przypisujemy pewną wartość – `blue`, która przykryje poprzednio zadeklarowaną wartość. W efekcie zapis ten będzie równoważny zapisowi:

```
P { color: blue; font-size: 12pt }
```

Te same zasady pokrywania wartości dot. importowanych oraz jednowierszowych arkuszy stylów. W uproszczeniu można powiedzieć, że wartościami obowiązującymi są te, które zostały zadeklarowane jako ostatnie. Wyjątkiem są deklaracje zawierające parametr `!important`, o którym za chwilę powiem więcej.

2.7. Klasy (parametr CLASS)

Wraz z arkuszami stylów, wprowadzono do języka HTML nowy parametr `CLASS`, dzięki któremu zyskujemy jeszcze większą kontrolę nad właściwościami elementów dokumentu. Parametr ten umożliwia przypisanie dowolnemu obiektowi (elementowi dokumentu) utworzonej przez nas klasy. W takiej klasie możemy zdefiniować interesujące nas właściwości danego obiektu. Dodatkową zaletą tego mechanizu jest fakt, iż dla danego obiektu (elementu dokumentu) możemy utworzyć wiele klas.

Utworzenie klasy w CSS wymaga użycia następującej składni:

```
Obiekt.NazwaKlasy {właściwość: wartość; ...}
```

Przykładowa definicja klasy `mojaKlasa` :

```
<STYLE type="text/css">
  P { color: black }
  P.mojaKlasa {
                color: blue;
                font-family: arial;
                font-size: 10pt;
              }
</STYLE>
```

w ciele dokumentu klasę wykorzystujemy tak:

```
<BODY>
  ...
  <P>Czarno to widzę</P>
  <P class="mojaKlasa">Teraz już na niebiesko...</P>
  ...
</BODY>
```

Możemy również utworzyć uniwersalną klasę, z której będą mogły korzystać różne elementy dokumentu HTML. Składnia definicji takiej klasy sugeruje brak powiązania z konkretnym obiektem (elementem dokumentu):

```
.uniwersalna { font-family: arial; font-size: 12pt }
```

Również do wywołania takich klas wykorzystujemy parametr `class` :

```
<P class="uniwersalna">Tak zadziałała uniwersalna klasa</P>
<H1 class="uniwersalna">Nagłówek też padł jej ofiarą.</H1>
```

2.8. Wyjątki (parametr ID)

Kolejnym parametrem, który pojawił się w języku HTML wraz ze stylami jest parametr `ID`. Dzięki niemu możemy obsługiwać tzw. wyjątki.

W CSS definiujemy je używając składni:

```
#NazwaWyjątku { właściwość: wartość; ... }
```

Korzystanie z wyjątku dla wybranego elementu dokumentu HTML sprowadza się do umieszczenia w tym elemencie parametru `id` i przypisanie jemu nazwy tego wyjątku:

```
  id="NazwaWyjątku"
```

Wyjatki wyjaśnię na przykładzie:

```
<HTML>
  <HEAD>
    <TITLE>Wyjątki</TITLE>
    <STYLE type="text/css">
      P.Klaseczka { color: red; font-family: arial; font-size: 14pt }
      #zielony { color: green }
    </STYLE>
  </HEAD>
  <BODY>
    <P class="Klaseczka">Pomidory są czerwone...</P>
    <P class="Klaseczka" id="zielony">Poza niedojrzałymi!</P>
  </BODY>
</HTML>
```

Wykorzystanie klasy `Klaseczka` spowoduje, że tekst `Pomidory są czerwone...` będzie wypisany czcionką `arial` o rozmiarze `14pt` w kolorze czerwonym. Z kolei użycie jednocześnie klasy `Klaseczka` i wyjątku `zielony` przyczyni się do tego, że

wartość red właściwości color, zdefiniowanej w klasie, zostanie zastąpiona wartością green ustaloną dla tej samej właściwości w wyjątku. W efekcie tekst Poza niedojrzałymi! będzie wypisany czcionką arial o rozmiarze 14pt w kolorze zielonym.

2.9. Kontekstowe arkusze stylów

Kontekstowe arkusze stylów odnoszą się do konkretnych przypadków występujących w zagnieżdżonej strukturze znaczników (elementów). Najlepiej wyjaśni to przykład. Załóżmy, że kolor tekstu w nagłówku stopnia pierwszego będzie zmieniony z niebieskiego na czerwony, ale wyłącznie w jego części znajdującej się pomiędzy znacznikami <EM> i </EM>.

```
<HTML>
  <HEAD>
    <TITLE>Kontekstowe arkusz stylów</TITLE>
    <STYLE type="text/css">
      H1 { color: blue }
      H1 EM { color: red }
    </STYLE>
  </HEAD>
  <BODY>
    <H1>Nagłówek do tego miejsca jest niebieski, <EM>ale tu jest już
        czerwony</EM> i znowu niebieski.</H1>
  </BODY>
</HTML>
```

Zgodnie z tym co założyliśmy, tekst nagłówka znajdujący się pomiędzy znacznikami <EM> i </EM> będzie w kolorze czerwonym, natomiast reszta nagłówka w kolorze niebieskim.

I jeszcze kilka przykładowych definicji:

```
P I { color: red }
UL LI { font-size: 10pt }
H1 B, H2 B, H1 EM { color: green }
```

2.10. Parametr !important

Wiemy już o tym, że w CSS wartości pewnych właściwości elementów dokumentu HTML mogą być przykrywane, np. w takiej sytuacji:

```
<STYLE type="text/css">
  P { font-size: 10pt }
  P { font: 18pt sans-serif }
  ...
  P { font-size: 20pt }
</STYLE>
```

Mimo, iż pierwsza deklaracja rozmiaru czcionki ustala wartość na 10pt, to i tak zostanie ona przykryta wartością 20pt, która została zadeklarowana w dalszej części arkusza stylów.

Jeżeli zależy nam na tym, aby wartości pewnych właściwości były stałe, tj. nie dawały się przykrywać, musimy do deklaracji takich właściwości dodać parametr `!important`.

Zatem rozmiar czcionki w naszym przykładzie będzie posiadał stałą wartość 10pt, jeżeli zostanie on zadeklarowany następująco:

```
<STYLE type="text/css">
  P { font-size: 10pt !important }
  P { font: 18pt sans-serif }
  ...
  P { font-size: 20pt }
</STYLE>
```

3. Pseudo-klasy

3.1. Pseudo-klasa ':first-child'

Za pomocą pseudo-klasy `:first-child` możemy wpływać na właściwości elementu będącego pierwszym potomkiem (dzieckiem) zadanego znacznika. Przykładowa definicja stylu dla pierwszego potomka znacznika `<DIV>` może wyglądać następująco:

```
DIV > P:first-child { color: blue }
```

Zapis ten oznacza, że pierwszy akapit (pierwszy potomek znacznika `<DIV>`) znajdujący się we wnętrzu bloku `<DIV> ... </DIV>` będzie wypisany w kolorze niebieskim, np.:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>

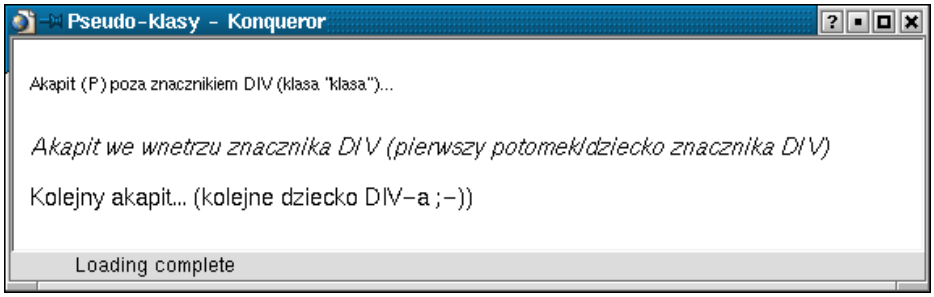
<HEAD>
  <TITLE>Pseudo-klasy</TITLE>

  <STYLE type="text/css">
     DIV > P:first-child { font-style: italic }
     .klasa {
              font-size: 14pt;
              font-style: normal;
            }
  </STYLE>
</HEAD>

<BODY>
  <P>Akapit (P) poza znacznikiem DIV (klasa "klasa")...
     <DIV class="mojaklasa">
     <P>Akapit we wnętrzu znacznika DIV (pierwszy potomek/dziecko
        znacznika DIV)
     <P>Kolejny akapit... (kolejne dziecko DIV-a ;-))
     </DIV>
    </P>
</BODY>

</HTML>
```

W przeglądarce zobaczymy:



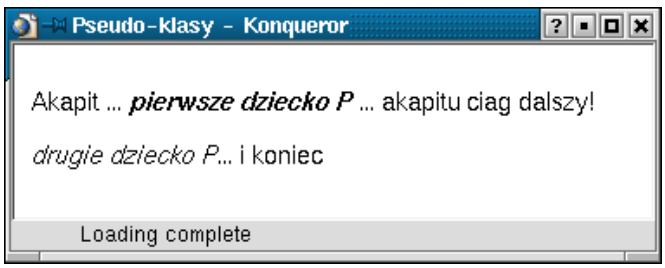
Z powyższego przykładu wynika, że właściwości akapitu `Akapit (P) poza znacznikiem DIV (klasa "klasa")...`, znajdującego się poza blokiem `<DIV>`, będą posiadały wartości domyślne. Wywołanie znacznika `<DIV>` z klasą `mojaklasa` spowoduje, że właściwości wszystkich akapitów znajdujących się we wnętrzu bloku `<DIV>` będą posiadały wartości zdefiniowane w klasie `mojaklasa`. Definicja ta pominie jednak pierwszy akapit, tj. `Akapit we wnętrzu znacznika DIV (pierwszy potomek/dziecko znacznika DIV)`, ponieważ będzie on pierwszym potomkiem znacznika `<DIV>` i zgodnie z wcześniejszą definicją, tekst znajdujący się w tym akapicie będzie pochylony.

Stosując pseudo-klasę `:first-child` możemy również w inny sposób definiować właściwości elementów będących pierwszymi potomkami. Sposób ten przedstawię w poniższym przykładzie:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <HEAD>
    <TITLE>Pseudo-klasy</TITLE>
    <STYLE type="text/css">
      P:first-child EM { font-weight: bold }
    </STYLE>
  </HEAD>

  <BODY>
    <P>Akapit ... <EM>pierwsze dziecko P</EM> ... akapitu ciąg dalszy!
    <P><EM>drugie dziecko P...</EM> i koniec
  </BODY>
</HTML>
```

Zapis `P:first-child EM {font-weight: bold}` oznacza, że tekst umieszczony po pierwszym znaczniku `<EM>` będzie pogrubiony. Wszystko dlatego, że dla pierwszego potomka/dziecka znacznika `<P>` (będącego znacznikiem `<EM>`) zostało zdefiniowane pogrubienie.



Na koniec jeszcze malutki przykład. Poniższe zapisy są równoważne:

```
* > A:first-child { ... }
A:first-child { ... }
```

i oznaczają, zdefiniowanie pierwszego potomka <A> dla dowolnego znacznika.

3.2. Pseudo-klasy ':link', ':visited', ':hover', ':active'

Najbardziej popularne pseudo-klasy, to te które związane są z elementem odsyłacza, który w HTML-u tworzymy za pomocą znacznika <A>.

Są nimi:

- :link - definiujący odsyłacz, który nie był do tej pory odwiedzony,
- :visited - definiujący odsyłacz już odwiedzony,
- :active - definiujący odsyłacz, który aktualnie jest uaktywniony.Sytuacja taka ma miejsce na przykład, gdy na odsyłaczu wciśniemy przycisk myszy i przytrzymamy go,
- :hover - definiujący odsyłacz, który został wybrany, ale nie uaktywniony. Sytuacja taka występuje na przykład, gdy przesuniemy kursor myszy na odsyłacz (nieuaktywniając go).

Przykładowe użycie pseudo-klas:

```
<STYLE type="text/css">
  A:link    { color: blue; text-decoration: none }
  A:visited { color: silver; text-decoration: none }
  A:active  { color: black; text-decoration: none }
  A:hover   { color: red; text-decoration: underline;
              background: yellow }
</STYLE>
```

użycie elementu A w ciele dokumentu HTML jest typowe:

```
<A href="http://www.w3.org">W3C</A>
```

Aby utworzyć klasę elementu A musimy posłużyć się następującą konstrukcją:

```
<STYLE type="text/css">
  A.klasa:link    { color: black }
  A.klasa:visited { color: silver }
  A.klasa:active  { color: black }
  A.klasa:hover   { color: red }
</STYLE>
```

Z klasy korzystamy zgodnie z wcześniej opisanymi zasadami, czyli:

```
<A href="http://www.w3.org" class="klasa">W3C</A>
```

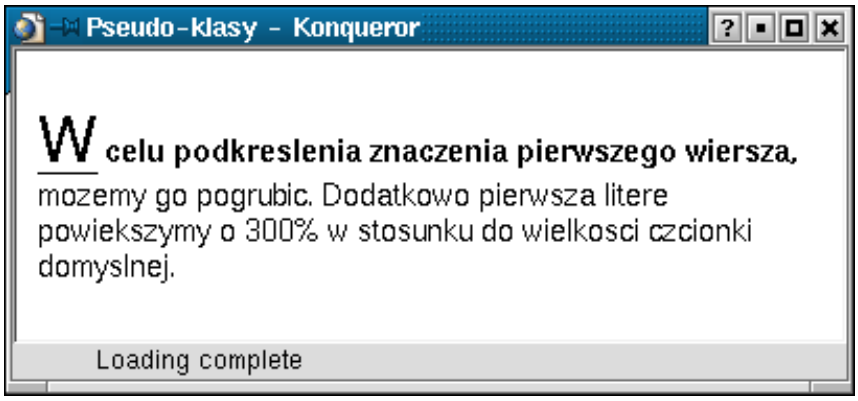
3.3. Pseudo-elementy ':first-line' i ':first-letter'

Za pomocą pseudo-elementu `:first-line`, możemy wpływać na zmianę właściwości pierwszego wiersza bloku tekstowego. Z kolei za pomocą `:first-letter` możemy sterować właściwościami pierwszego znaku bloku tekstowego. Wykorzystanie podanych pseudo-elementów przedstawię w poniższym przykładzie:

```
<HTML>
  <HEAD>
    <TITLE>Pseudo-klasy</TITLE>
    <STYLE type="text/css">
      P:first-line { font-weight: bold }
      P:first-letter { font-size: 300%; text-decoration: underline }
    </STYLE>
  </HEAD>

  <BODY>
   <P>W celu podkreślenia znaczenia pierwszego wiersza, możemy go
      pogrubić. Dodatkowo pierwszą literę powiększymy o 300%
      w stosunku do wielkości czcionki domyślnej.</P>
  </BODY>
</HTML>
```

W przeglądarce zobaczymy:



Jak widać powyżej, dla całego pierwszego wiersza akapitu `<P>` zdefiniowaliśmy pogrubienie. Dodatkowo pierwszy znak akapitu powiększyliśmy o 300% i dodaliśmy podkreślenie.

4. Media

Problem ten został już poruszony w pkt. 2.2, w którym przedstawiłem HTML-owe podejście do różnych mediów. W tym punkcie zajmiemy się z kolei tym samym problemem widzianym od strony kaskadowych arkuszy stylów.
CSS2 pozwalają na definiowanie wyglądu dokumentu w zależności od rodzaju medium, na jakie kierowany jest ten dokument. Dzięki takiemu podejściu, dokument oglądany w oknie przeglądarki, będzie mógł wyglądać nieco inaczej niż na wydruku czy projektorze.

4.1. Definiowanie styli dla mediów '@medium'

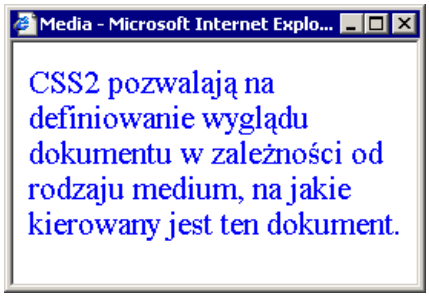
Aby lepiej wyjaśnić sposób definiowania styli dla różnych mediów, posłużę się przykładem. Właściwości znacznika akapitu `<P>` zdefiniuję pod kątem ekranu (to co zobaczymy w przeglądarce) oraz drukarki:

```
<STYLE type="text/css">
  @media screen {
    P { font-size: 16pt; font-style: normal; color: blue }
  }

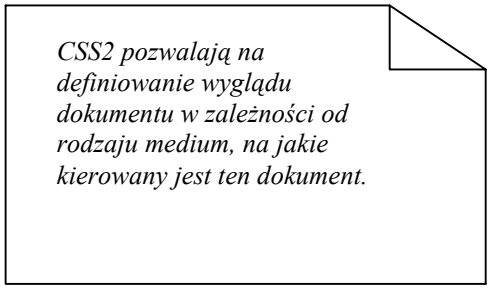
  @media print {
    P { font-size: 10pt; font-style: italic }
  }
</STYLE>
```

Dla takich definicji, tekst akapitu w oknie przeglądarki będzie wyświetlany czcionką normalną (prostą) o rozmiarze 16pt w kolorze niebieskim. Ten sam tekst na drukarce zostanie wydrukowany czcionką pochyloną o rozmiarze 10pt.

Okno przeglądarki:



Wydruk z drukarki:



W3C w specyfikacji CSS2 proponuje szereg urządzeń, dla których możemy definiować oddzielne style (tak jak w powyższym przykładzie). Są wśród nich m.in.:

- screen
- print
- aural
- projector
- all

W obecnej chwili nie wszystkie przeglądarki potrafią poprawnie interpretować podane urządzenia. Myślę jednak, że jest to kwestią czasu (jak zwykle ;-)).

5. Formatowanie zawartości dokumentu

5.1. Marginesy (margin)

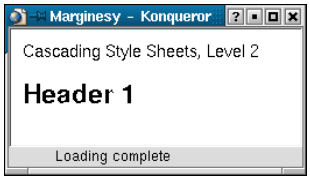
Do określania marginesów dokumentu wykorzystuje się następujące właściwości:

- `margin-top` - margines górny (odległość od górnej krawędzi dokumentu),
- `margin-bottom` - margines dolny,

- `margin-left` - margines lewy,
- `margin-right` - margines prawy,
- `margin` - właściwość określająca margines dla wszystkich czterech stron dokumentu jednocześnie.

Przykładowo, aby odsunąć zawartość dokumentu od krawędzi okna przeglądarki o 10 pikseli z każdej strony, powinniśmy zdefiniować następująco właściwość `margin` dla znacznika `<BODY>` :

```
<STYLE type="text/css">
  BODY { margin: 10px }
</STYLE>
```

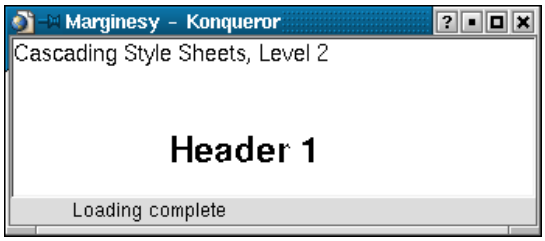


Marginesy możemy definiować dla różnych elementów dokumentu HTML. Przedstawię to na przykładzie znaczników `<BODY>` i `<H1>` posiadających własne marginesy.

```
<STYLE type="text/css">
  BODY {
    margin-top    : 0px;
    margin-right  : 0px;
    margin-bottom : 0px;
    margin-left   : 0px;
  }

  H1 {
    margin-left: 100px;
    margin-top :  40px;
  }
</STYLE>
```

W oknie przeglądarki zobaczymy:




```
<BODY>
  Cascading Style Sheets, Level 2 <h1>Header 1</h1>
</BODY>
```

Definicję powyższego stylu dla znacznika `<BODY>` można w tym przypadku uprościć. Korzystając z faktu, że dla każdej z czterech stron dokumentu margines jest jednakowy (0px), możemy zapisać to tak:

```
BODY { margin: 0px }
```

I jeszcze jeden przykład przypominający o możliwości definiowania bardzo wygodnych styli jednowierszowych:

```
<!DOCTYPE HTML PUBLIC „-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD><TITLE>Marginesy</TITLE></HEAD>

<BODY style=„margin: 5em">
  Cascading Style Sheets, Level 2
</BODY>
</HTML>
```

5.2. Odstęp pomiędzy elementem a jego obramowaniem (padding)

Do określania odstępu pomiędzy elementem a jego obramowaniem stosuje się następujące właściwości:

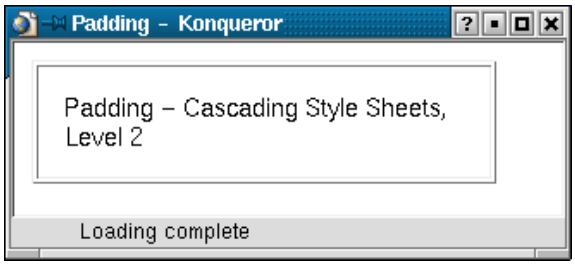
- `padding-top` - odstęp górny (odległość od górnej krawędzi elementu),
- `padding-bottom` - odstęp dolny,
- `padding-left` - odstęp lewy,
- `padding-right` - odstęp prawy,
- `padding` - właściwość określająca odstęp dla wszystkich czterech stron elementu jednocześnie.

Najbardziej typowym miejscem wykorzystania tych właściwości są tabele. Dlatego też pierwszym przykładem jaki przedstawię, będzie określenie odstępu zawartości komórek od obramowania za pomocą właściwości `padding`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
  <TITLE>Padding</TITLE>
  <STYLE type="text/css">
    TABLE { padding: 15px; }
  </STYLE>
</HEAD>

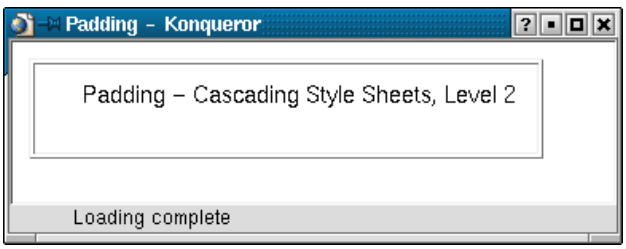
<BODY>
  <TABLE border="1" width="90%">
    <TR>
      <TD>Padding - Cascading Style Sheets, Level 2
  </TABLE>
</BODY>
</HTML>
```

W oknie przeglądarki zobaczymy:



Podobnie, jak w przypadku marginesów, również i w tym miejscu możemy oddzielnie dla każdej z czterech stron definiować odstępy.

```
<STYLE type=„text/css">
  TABLE {
    padding-left   : 30px;
    padding-right  :  5px;
    padding-top    : 10px;
    padding-bottom :  2em;
  }
</STYLE>
```

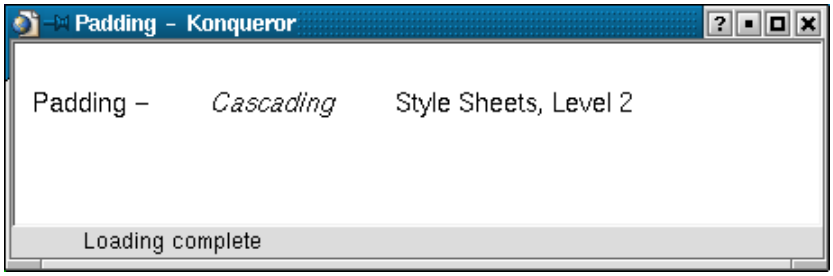


Jak już wspomniałem, właściwości `padding` nie odnoszą się wyłącznie do tabel. Możemy je również stosować w połączeniu z innymi elementami dokumentu. Poniżej przedstawiam przykład użycia ich wraz ze znacznikiem `<EM>`.

```
<HTML>
<HEAD>
  <TITLE>Padding</TITLE>
  <STYLE type=„text/css”>
    EM { padding: 30px; }
    P  { font-size: 14px; }
  </STYLE>
</HEAD>

<BODY>
  <P>Padding - <EM>Cascading</EM> Style Sheets, Level 2
</BODY>
</HTML>
```

W oknie przeglądarki zobaczymy:



5.3. Obramowanie (border)

Dla każdego elementu umieszczonego w dokumnecie możemy zdefiniować obramowanie. Dla tworzonego obramowania możemy określać grubość, jego kolor oraz styl.

5.3.1. Grubość obramowania

Do określania grubości krawędzi obramowania używamy właściwości:

- `border-top-width` - grubość górnej krawędzi,
- `border-bottom-width` - grubość dolnej krawędzi,
- `border-left-width` - grubość lewej krawędzi,
- `border-right-width` - grubość prawej krawędzi,
- `border-width` - grubość wszystkich czterech krawędzi.

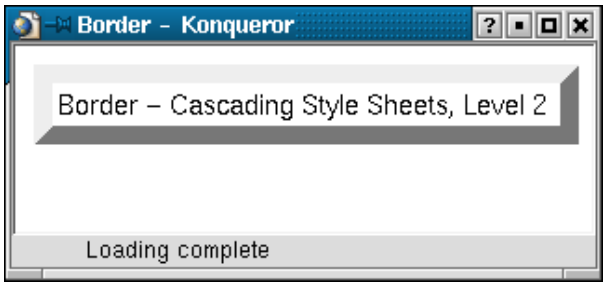
Powyższym właściwościom możemy przypisywać następujące wartości:

- `thin` - cienka krawędź,
- `medium` - średniej grubości krawędź,
- `thick` - gruba krawędź,
- konkretną wartość, np. `5px` czy `0.3em`.

Poniżej zdefiniujemy dla komórek tabeli obramowanie o grubości 10 pikseli (dla wszystkich krawędzi jednakowo).

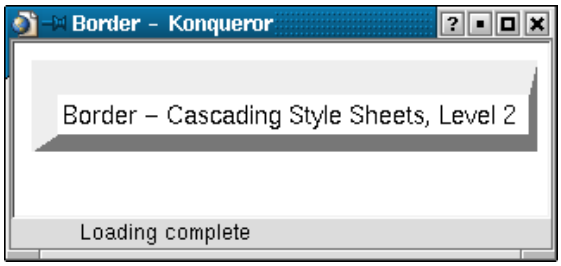
```
<HTML>
<HEAD>
 <TITLE>Border</TITLE>
 <STYLE type=„text/css”>
   TABLE {
     border-width: 10px;
     font-size: 14px;
   }
 </STYLE>
</HEAD>
<BODY>
 <TABLE>
 <TR>
   <TD>Border - Cascading Style
       Sheets, Level 2
 </TABLE>
</BODY>
</HTML>
```

Widok dokumentu w przeglądarce:



Jeżeli chcemy dla każdej krawędzi zdefiniować inną grubość obramowania, posłużymy się do tego celu następującą definicją:

```
<STYLE type=„text/css”>
 TABLE {
  border-left-width   : 15px;
  border-top-width    : 20px;
  border-right-width  :  5px;
  border-bottom-width : 10px;
 }
</STYLE>
```



Do określania grubości krawędzi możemy stosować zapis skrócony.
Jeżeli przykładowo będziemy chcieli krawędziom poziomym (`top` i `bottom`) obramowania akapitu przypisać wartość `thin`, natomiast pionowym (`right` i `left`) wartość `thick`, możemy zapisać to tak:

```
P { border-width: thin thick }
```

Dla takiej definicji poszczególnym krawędziom (`top`, `right`, `bottom`, `left`) zostaną przypisane odpowiednio (w nastepującej kolejności) wartości:

```
border-top-width    = thin
border-right-width  = thick
border-bottom-width = thin
border-left-width   = thick
```

I jeszcze kilka skrótów:

```
P { border-width: thick }              /* thick thick thick thick */
P { border-width: thin thick medium } /* thin thick medium thick */
```

5.3.2. Kolor obramowania

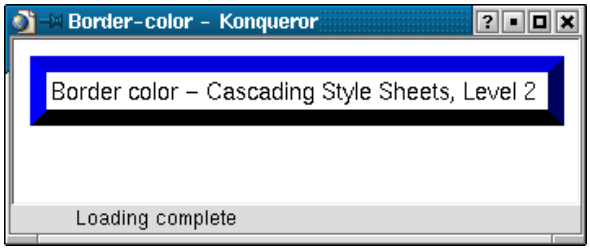
Do określania koloru krawędzi obramowania używamy właściwości:

- `border-top-color` - kolor górnej krawędzi,
- `border-bottom-color` - kolor dolnej krawędzi,
- `border-left-color` - kolor lewej krawędzi,
- `border-right-color` - kolor prawej krawędzi,
- `border-color` - kolor wszystkich czterech krawędzi.

Wymienionym właściwościom możemy przypisywać dowolne wartości definiujące kolor. Wartości koloru mogą być wyrażone w dowolnych jednostkach opisanych w rozdziale 2.4.3. Przykładowo:

```
<HTML>
<HEAD>
  <TITLE>Border-color</TITLE>
  <STYLE type=„text/css”>
    TABLE {
      border              : 10px;
      border-left-color   : rgb(0,0,255);
      border-top-color    : #00d;
      border-right-color  : #0000bb;
      border-bottom-color : black;
    }
  </STYLE>
</HEAD>
<BODY>
  <TABLE>
    <TR><TD>Border color - Cascading Style Sheets, Level 2</TD></TR>
  </TABLE>
</BODY>
</HTML>
```

W oknie przeglądarki zobaczymy:



Krawędź lewa, górna i prawa są w odcieniach niebieskich ;-) (coraz ciemniejszych), natomiast dolna jest czarna.

5.3.3. Styl obramowania

Krawędzie obramowania możemy wyrysowywać przy użyciu zadanego stylu, wzoru. Na ten użytek zdefiniowane zostały następujące właściwości:

- `border-top-style` - styl górnej krawędzi,
- `border-bottom-style` - styl dolnej krawędzi,

- `border-left-style` - styl lewej krawędzi,
- `border-right-style` - styl prawej krawędzi,
- `border-style` - styl wszystkich czterech krawędzi.

Powyższym właściwościom możemy przypisywać wartości:

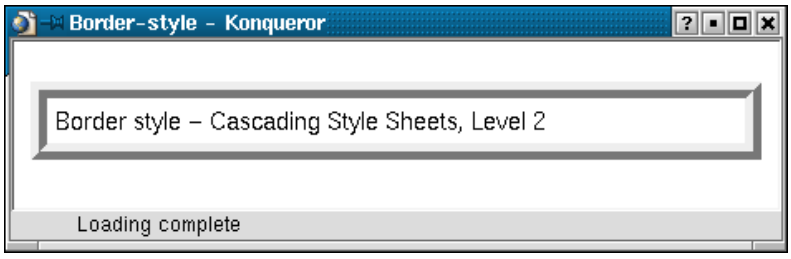
- `none` - brak krawędzi,
- `hidden` - krawędź ukryta,
- `dotted` - linia punktowa,
- `dashed` - linia przerywana,
- `solid` - linia ciągła,
- `double` - podwójna linia ciągła,
- `groove` - ramka 3D,
- `ridge` - ramka 3D (odwrotna do `groove`),
- `inset` - efekt wklęsłości,
- `outset` - efekt wypukłości.

Przykład wykorzystania opisanych wyżej elementów:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
  <TITLE>Border-style</TITLE>
  <STYLE type="text/css">
    P {
      border       : 10px;
      padding      : 5px;
      border-style : rigde;
    }
  </STYLE>
</HEAD>

<BODY>
  <P>Border style - Cascading Style Sheets, Level 2
</BODY>
</HTML>
```

W oknie przeglądarki zobaczymy następujący efekt:



W odniesieniu do `border-style`, podobnie jak miało to miejsce w przypadku właściwości `border-width`, możemy stosować zapis skrócony. Przykładowo:

```
H1 {border-style: solid dotted} /* solid dotted solid dotted */
H2 {border-style: dotted}       /* dotted dotted dotted dotted */
```

Należy pamiętać jedynie o zasadzie, że pierwsza wartość dotyczy górnej krawędzi (`top`), a kolejne określają krawędzie zgodnie z ruchem wskazówek zegara (`right`, `bottom`, `left`).

5.3.4. Definiowanie grubości, koloru i stylu obramowania jednocześnie

Powyżej omówione zostały trzy grupy właściwości dające możliwość definiowania odpowiednio grubości, koloru i stylu obramowania elementu. W specyfikacji CSS2 występuje dodatkowo jeszcze jedna grupa właściwości związanych z obramowaniem. Za ich pomocą możemy ustawić wszystkie opisane cechy obramowania jednocześnie. Do grupy tych właściwości należą:

- `border-top` - właściwości górnej krawędzi,
- `border-bottom` - właściwości dolnej krawędzi,
- `border-left` - właściwości lewej krawędzi,
- `border-right` - właściwości prawej krawędzi,
- `border` - właściwości wszystkich czterech krawędzi.

Właściwościom tym możemy przypisywać opisane w poprzednich rozdziałach wartości, przy czym jako pierwszą podajemy grubość, następnie styl i na końcu kolor krawędzi. Przykładowo:

```
<STYLE type="text/css">
  TABLE {
    border: 2em inset magenta;
    font-size: 14px;
  }
</STYLE>
```

Wszystkie krawędzie obramowania będą jednakowe i będą wyrysowane linią o grubości `2em`, wzorem (stylem) `inset` i w kolorze różowym (`magenta`).

Niektóre wartości możemy pomijać, np.:

```
border-top: thick solid;
border: dashed blue;
```

5.4. Określanie szerokości i wysokości

Domyślna wysokość prostokątnego obszaru z elementem określana jest na podstawie wysokości tego elementu. Z kolei szerokość obszaru określana jest na podstawie szerokości elementu znajdującego się w tym obszarze lub miejsca dostępnego na stronie.
Domyślne wartości wysokości i szerokości możemy zmieniać za pomocą następujących właściwości:

- `height` - wysokość,
- `width` - szerokość,
- `min-height` - najmniejsza wysokość, w jakiej element może zostać wyświetlony,
- `max-height` - największa wysokość, w jakiej element może zostać wyświetlony,

- `min-width` - najmniejsza szerokość, w jakiej element może zostać wyświetlony,
- `max-width` - największa szerokość, w jakiej element może zostać wyświetlony.

Przykładowo:

```
P { width  : 200px;
    height : 100px;
  }
```

5.5. Właściwość 'overflow'

Za pomocą właściwości `overflow` możemy określić, jak zachować ma się tekst w obszarze, którego wymiary określone zostały właściwościami `width` i `height`, a który to tekst nie mieści się w całości w tym obszarze. Właściwość overflow może przyjąć jedną z wartości:

- `auto` - dodanie paska przewijania, jeśli tekst nie mieści się w obszarze,
- `hidden` - nie wyświetlanie tekstu nie mieszczącego się w obszarze,
- `scroll` - ustawienie na stałe pasków przewijania, nawet gdy tekst mieści się,
- `visible` - wyświetlenie całego tekstu (wyjście poza określony obszar).

Przykładowo:

```
P { width : 200px; height : 100px;
    overflow : scroll;
  }
```

5.6. Właściwość 'clip'

Właściwość `clip` związana jest z obszarem obcinania zawartości obiektu i domyślnie wartości tej właściwości są zgodne z zewnętrznymi obrzeżami tego obszaru.
Właściwość ta dotyczy jedynie tych elementów, których właściwość `overflow` nie jest ustawiona na `visible`.
Jeżeli chcemy zmienić domyślne wartości obcinania (obszaru prostokatnego) wartość `clip` należy ustawić następująco: `rect(górny, dolny, lewy, prawy)`, gdzie `górny`, `dolny`, `lewy`, `prawy` to wartości odstępów od odpowiednich brzegów obszaru.

Przykładowo:

```
P { clip     : rect(10px, 10px, 15px, 15px);
    overflow : auto;
  }
```

5.7. Właściwość 'visibility'

Właściwość `visibilty` określa widoczność elementu. Może ona przyjąć nastepujące wartości:

- `visible` - zawartość obszaru wraz z wszystkimi obramowaniami jest widoczna,

- `hidden` - zawartość obszaru nie jest pokazywana, mimo iż nadal zajmuje ten obszar i wpływa na rozmieszczenie pozostałych elementów na stronie,
- `collapse` - działa jak `hidden`, z tym że wiersze i kolumny tabeli są całkowicie ukrywane,

Przykładowo:

```
P { width      : 150px;
    height     : 100px;
    visibility : hidden;
    overflow   : auto;
  }
```

5.8. Warstwy

W CSS2 dodano właściwości pozwalające manipulować prostokątnymi obszarami, a w szczególności umieszczać je w określonych miejscach na stronie, w oknie czy innym obszarze. Obszary możemy umieszczać na tzw. warstwach, które możemy swobodnie pozycjonować na stronie niezależnie od siebie.
Do pozycjonowania warstw wykorzystuje się właściwość `position`, która może przyjąć jedną z czterech wartości:

- `relative` - pozycjonowanie względne,
- `absolute` - pozycjonowanie bezwzględne,
- `static` - pozycjonowanie stałe,
- `fixed` - nakładanie elementów.

Przesunięcie warstwy określamy za pomocą właściwości `left`, `right`, `top` i `bottom`.

5.8.1. Pozycjonowanie względne

Warstwy pozycjonowane względnie tworzymy przez przypisanie dla właściwości `position` wartości `relative`.
Pozycje warstw pozycjonowanych względnie nie wpływają na siebie, mogą więc zachodzić jedna na drugą.

```
<HTML>
<HEAD>
  <TITLE>position</TITLE>
  <STYLE type="text/css">
    #warstwa { position: relative; top: 150px; left: 200px }
  </STYLE>
</HEAD>

<BODY>
  <DIV id=”warstwa”>Hello ... Tu warstwa 1!</DIV>
</BODY>
</HTML>
```

5.8.2. Pozycjonowanie bezwzględne

Warstwy pozycjonowane bezwzględnie tworzymy przez przypisanie dla właściwości `position` wartości `absolute`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>position</TITLE>
  <STYLE type="text/css">
    #w1 { position: absolute; top: 200px; left: 200px }
    #w2 { position: absolute; top: 195px; left: 220px }
  </STYLE>
</HEAD>

<BODY>
  <DIV id="w1">Hello ... Tu warstwa 1!</DIV>
  <DIV id="w2">A kuku ... tu warstewka 2!</DIV>
  ...
</BODY>
</HTML>
```

5.8.3. Pozycjonowanie stałe

Warstwy ze stałą pozycją tworzymy przez przypisanie dla właściwości `position` wartości `static`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>position</TITLE>
  <STYLE type="text/css">
    #warstwa {
       position: fixed;
       top: 10px; left: 20px; height: 50px; width: 100px;
       background: magenta;
    }
  </STYLE>
</HEAD>

<BODY>
  <DIV id="warstwa">Obszar warstwy</DIV>
  ...
</BODY>
</HTML>
```

5.8.4. Nakładanie elementów (właściwość ‘z-index’)

Za pomocą właściwości `z-index` możemy określić kolejność nakładania się na siebie warstw. Warstwy posiadające właściwość `z-index` z większą wartością (będącą liczbą całkowitą) będą przykrywały warstwy posiadające `z-index` z mniejszą wartością.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>z-index</TITLE>
  <STYLE type="text/css">
    #war1 { position: absolute;
            top: 100px; left: 100px; height: 50px; width: 50px;
            background: red;
            z-index: 2 }
    #war2 { position: absolute;
            top: 80px; left: 80px; height: 50px; width: 50px;
            background: yellow;
            z-index: 3 }
    #war3 { position: absolute;
            top: 60px; left: 120px; height: 50px; width: 50px;
            background: green;
            z-index: 1 }
  </STYLE>
</HEAD>

<BODY>
  ...
  <DIV id="war1">Jestem sobie ogrodniczka.</DIV>
  <DIV id="war2">Mam nasionek pół koszyczka.</DIV>
  <DIV id="war3">A w nim bratki i stokrotki.</DIV>
  ...
</BODY>
</HTML>
```

6. Kolor i tła elementów

6.1 Kolor elementów

Do określania koloru elementów występujących w dokumencie, stosujemy właściwość `color`. Właściwość ta może przyjmować wartości, które opisałem szczegółowo w rozdziale 2.4.3, dlatego w tym miejscu ograniczę się do podania kilku przykładów.

```
H1 { color: blue }
H2 { color: rgb(255,0,100) }
EM { color: rgb(100%,10%,40%) }
P  { color: #E855B3 }
P  { color: #F02 }
```

6.2. Tło elementów

Dla elementów występujących w dokumentach HTML możemy określać tło. Tłem może być wypełnienie w zadanym kolorze lub dowolny obrazek (jego powtórzenia) zgodny z formatem akceptowanym przez HTML-a (*gif*, *jpg* czy *png*).

Do określania parametrów tła mamy do dyspozycji kilka właściwości:

- `background-color` - kolor tła,
- `background-image` - adres URI obrazka będącego tłem,
- `background-repeat` - powtarzalności obrazka w tle,
- `background-posiotion` - pozycja obrazka względem krawędzi ekranu,
- `background-attachment` - określenie, czy obrazek tła będzie przewijany wraz z zawratością dokumentu czy też nie,
- `background` - zgrupowanie powyższych właściwości w jednej właściwości (forma skrótowa).

Właściwość `background-color` może przyjmować takie same wartości, jak właściwość `color`, np.:

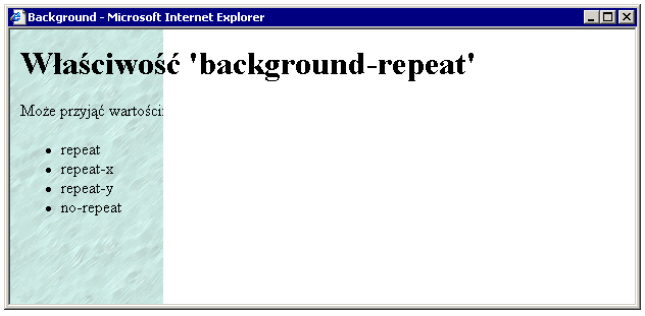
```
H1 { background-color: white }
```

Za pomocą właściwości `background-image` określamy, jakim obrazkiem będzie wypełnione tło elementu, np.:

```
BODY { background-image: url("marmur.gif") }
```

Przy tak zdefiniowanym tle dokumentu, obrazek `marmur.gif` wypełni całą powierzchnię tła dokumentu. W przypadku, gdy rozmiary obrazka będą mniejsze od aktualnego rozmiaru okna przeglądarki (powierzchni dokumentu), nastąpi automatyczne powielenie obrazka (w poziomie i pionie) w celu wypełnienia całej powierzchni tła. Jest to bardzo przydatny mechanizm :-), jednak czasami zależy nam na tym, aby obrazek nie był powtarzany lub też chcemy, aby powtarzanie przebiegało tylko w wybranym kierunku (x lub y). I tu z pomocą idzie nam kolejna właściwość `background-repeat`. Może ona przyjąć jedną z czterech wartości:

- `repeat` - powtarzanie w obu kierunkach,
- `repeat-x` - powtarzanie tylko w poziomie,
- `repeat-y` - powtarzanie tylko w pionie,
- `no-repeat` - brak powtarzania.



Efekt, jak z boku, otrzymamy definiując tło w następujący sposób:

```
<STYLE type="text/css">
  BODY {
    background-image  : url("b.gif");
    background-repeat : repeat-y;
  }
</STYLE>
```

Dla obrazka tła, za pomocą właściwości `background-position`, możemy określać pozycję. Do dyspozycji mamy pięć wartości:

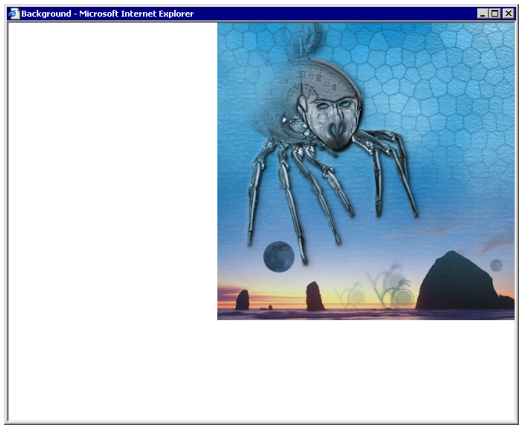
- `left` - `( 0%)` ściągnięcie obrazka do lewej krawędzi,
- `right` - `(100%)` ściągnięcie do prawej krawędzi,
- `top` - `( 0%)` ściągnięcie do górnej krawędzi,
- `bottom` - `(100%)` ściągnięcie do dolnej krawędzi,
- `center` - `( 50%)` rozmieszczenie centralne.

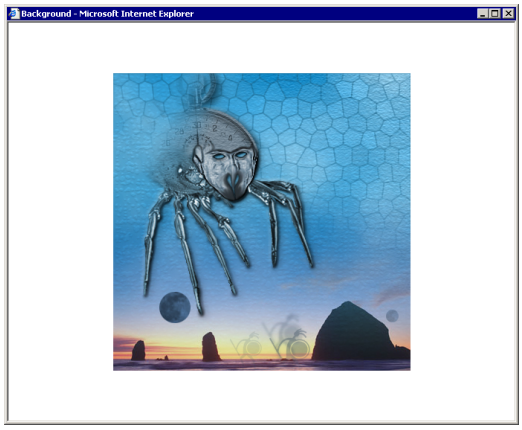
Dodatkowo wymienione pozycje możemy określać procentowo. Wartości procentowe odpowiadające pozycji podane zostały w nawiach. Przykład znajdziesz niżej :-).

Pierwsze cztery wartości możemy stosować parami, np. `top left` czy `bottom left`.

```
BODY {
 background-image: url("acpic.jpg");
 background-posiotion: right top;
}
```

```
BODY {
 background-image: url("acpic.jpg");
 background-posiotion: center;
}
```





Stosując zapis procentowy, dla lewego przypadku mógłbym zapisać:

```
BODY {
 background-image: url("acpic.jpg");
 background-posiotion: 100% 0%;
}
```

Kolejną właściwość – `background-attachment` – wyjaśnię na przykładzie tła zdefiniowanego dla elementu `BODY`.

Zakłóżmy, że nasz dokument posiada tło zbudowane z jakiegoś obrazka:

```
BODY {
  background-image  : url("acpic.jpg");
  background-repeat : repeat-y;
}
```

a treść dokumentu jest na tyle duża, że nie mieści się jednocześnie w oknie przeglądarki. Aby zatem przeczytać ją w całości, musimy dokument przewijać za pomocą pasków przewijania (pionowego). Domyślnie, wraz z przewijaną treścią dokumentu, przewijany będzie obrazek (lub obrazki w przypadku powielania) tła. Ta

sytuacja jest równoważna ustawieniu właściwości `background-attachment` na wartość `scroll`. Jeżeli jednak chcemy, aby nasz obrazek nie był przesuwany razem z treścią dokumentu (tworzył tzw. znak wodny) opisywanej właściwości musimy nadać wartość `fixed`, np.:

```
BODY {
  background-image      : url(”acpic.jpg”);
  background-repeat     : repeat-y;
  background-attachment : fixed;
}
```

Wszystkie opisane wyżej właściwości jednostkowe opisujące parametry tła, możemy ustawiać jednocześnie za sprawą właściwości `background`.

Poniższy zapis:

```
TABLE {
  background-image      : url("bruk.gif");
  background-repeat     : repeat-y;
  background-color      : rgb(100,200,250);
}
```

możemy skrócić do następującego:

```
TABLE { background: url("bruk.gif") repeat-y rgb(100,200,250) }
```

7. Czcionki i teksty

Niemal najważniejszymi elementami dokumentów HTML-owych są teksty. W tym punkcie zajmiemy się opisaniem najważniejszych cech czcionek oraz całych fragmentów tekstów. Zaczniemy od opisania elementarnych składników tekstów, czyli od czcionek.

7.1. Czcionki

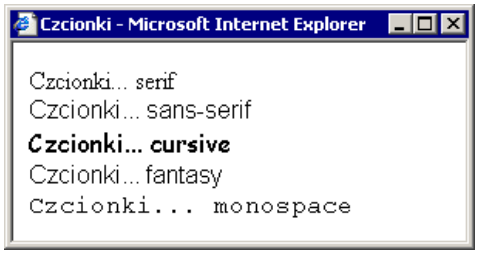
7.1.1. Krój czcionki

Teksty znajdujące się w dokumencie mogą być pisane różnymi krojami czcionek. Do ustalania kroju czcionki w CSS wykorzystuje się właściwość `font-family`. Właściwość ta może przyjmować wartości, będące konkretnymi nazwami krojów czcionek (zainstalowanych w systemie, bądź dostarczonych z dokumentem) lub nazwami ogólnymi. W przypadku nazw krojów czcionek sprawa jest otwarta :-). Co do nazw ogólnych, to CSS daje nam kilka propozycji: `'serif'`, `'sans-serif'`, `'cursive'`, `'fantasy'` i `'monospace'`.

Poniższy przykład prezentuje stosowanie ogólnych nazw krojów czcionek:

```
<SPAN style="font-family: serif">Czcionki... serif</SPAN><BR>
<SPAN style="font-family: sans-serif">Czcionki... sans-serif</SPAN><BR>
<SPAN style="font-family: cursive">Czcionki... cursive</SPAN><BR>
<SPAN style="font-family: fantasy">Czcionki... fantasy</SPAN><BR>
<SPAN style="font-family: monospace">Czcionki... monospace</SPAN>
```

W przeglądarce zobaczymy:



Z kolei stosowanie konkretnych krojów czcionek przedstawia następujący przykład:

```
<SPAN style="font-family: Lucida Console">Czcionka...Lucida Console</SPAN><BR>
<SPAN style="font-family: Verdana">Czcionka... Verdana</SPAN><BR>
<SPAN style="font-family: Symbol">Czcionka... Symbol</SPAN><BR>
<SPAN style="font-family: MS Sans Serif">Czcionka... MS Sans Serif</SPAN><BR>
<SPAN style="font-family: Courier New CE">Czcionka... Courier New CE</SPAN>
```

Tym razem przeglądarka pokaże nam to:

Stosowanie konkretnych nazw czcionek wiąże się z pewną niedogodnością. Otóż może zdarzyć się, że w definicji kroju wpiszemy nazwę czcionki, której nie będzie w systemie osoby przeglądającej nasz dokument. W takiej sytuacji przeglądarka posłuży się krojem domyślnym, co może skutecznie zepsuć estetykę dokumentu. Aby ustrzec się przed takim efektem, możemy zdefiniować pewną grupę czcionek alternatywnych. Jeżeli pierwsza na liście czcionka nie będzie występowała w systemie przeglądającego nasz dokument, przeglądarka spróbuje posłużyć się kolejną czcionką z listy, itd.

Przykładowa definicja:

```
<STYLE type="text/css">
  P { font-family: tahoma, verdana, arial, sans-serif }
</STYLE>
```

oznacza, że w naszym dokumencie domyślnym krojem czcionki dla akapitu będzie `tahoma`. Jeżeli przeglądarka nie znajdzie w systemie takiej czcionki spróbuje załadować kolejną, czyli `verdana`. Jeżeli i tej nie będzie – sięgnie po `arial`, etc.

7.1.2. Styl czcionki

Na styl czcionki możemy wpływać przy pomocy następujących właściwości:

- `font-style`
- `font-variant`

- font-weight
- font-stretch

Właściwość font-style może przyjąć jedną z trzech wartości: normal, italic i oblique.
Ustawienie powyższej właściwości na wartość normal (wartość domyślna) spowoduje przypisanie czcionce parametrów zdefiniowanych w przeglądarce jako *'normal'* (u większości przeglądarek będzie to tekst prosty). W przypadku wartości italic i oblique obowiązuje ta sama zasada jak dla normal, przy czym tekst będzie pochylony.

```
<STYLE type=„text/css”>
  P { font-family: snas-serif; font-style: italic }
</STYLE>
```

I jeszcze jeden przykład:

```
<BODY>
 <H1><I>Styl <span style="font-style: normal">czcionki</span></I></H1>
</BODY>
```

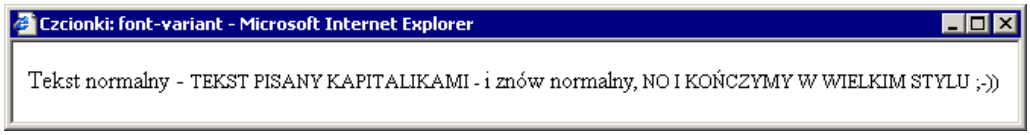
Dzięki właściwości font-variant możemy modyfikować czcionki z postaci normalnej do kapitalików i odwrotnie. Właściwość ta może przyjąć jedną z dwóch wartości: normal (wartość domyślna) i small-caps.

Oto przykład użycia właściwości font-variant :

```
<HTML>
<HEAD>
  <TITLE>Czcionki: font-variant</TITLE>
  <STYLE type="text/css">
    P { font-family: times, arial; font-style: normal }
    .klasa1 { font-variant: normal }
    .klasa2 { font-variant: small-caps }
  </STYLE>
</HEAD>

<BODY>
  <P>Tekst normalny - <SPAN class="klasa2">tekst pisany
     kapitalikami - <SPAN class="klasa1">i znów normalny,</SPAN>
     no i kończymy w wielkim stylu ;-))</SPAN>
</BODY>
</HTML>
```

W oknie przeglądarki zobaczymy:



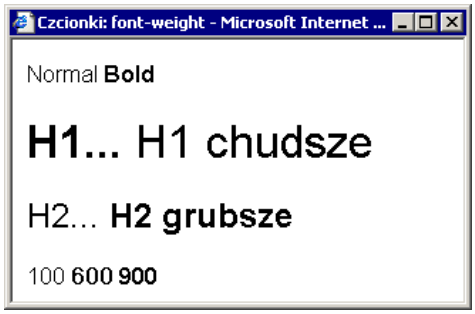
Właściwość font-weight pozwala określić grubość czcionki. Może ona przyjmować nastepujące wartości:

- `100÷900` - wartość `100` określa najmniejszą grubość, wartość `900` największą. Wartości pośrednie zmieniają się co `100`,
- `normal` - czcionka normalna (domyślna grubość czcionki), równa wartości `400`.
- `bold` - czcionka pogrubiona, równa wartości `700`,
- `bolder` - zwiększenie o `100` grubości obowiązującej dla elementu dokumnetu, dla którego przypisujemy właściwość `font-weight` z wartością `bolder`,
- `lighter` - zmniejszenie o `100` grubości obowiązującej dla elementu dokumnetu, dla którego przypisujemy właściwość `font-weight` z wartością `lighter`.

Przykład użycia właściwości `font-weight` :

```
<HTML>
<HEAD>
  <TITLE>Czcionki: font-weight</TITLE>
  <STYLE type="text/css">
    BODY { font-family: arial }
    H1 { font-weight: bold }
    H2 { font-weight: normal }
    P  { font-weight: normal }
  </STYLE>
</HEAD>
<BODY>
  <P>
  <SPAN style="font-weight: normal">Normal</SPAN>
  <SPAN style="font-weight: bold">Bold</SPAN><BR>
  <H1>H1... <SPAN style="font-weight: lighter"> H1 grubsze</SPAN></H1>
  <H2>H2... <SPAN style="font-weight: bolder"> H2 chudsze</SPAN></H2>
  <SPAN style="font-weight: 100">100</SPAN>
  <SPAN style="font-weight: 600">600</SPAN>
  <SPAN style="font-weight: 900">900</SPAN>
  </P>
</BODY>
</HTML>
```

I widok w przeglądarce:



Ostatnią właściwością wpływającą na styl czcionki jest `font-stretch`. Za jej pomocą możemy regulować zagęszczeniem czcionek tworzących napis. Właściwość ta może przyjmować następujące wartości:

- `ultra-condensed` - najmniejsze odstępy między czcionkami (literami),
- `extra-condensed` -
- `condensed` - ↓ zwiększające się odstępy między czcionkami,
- `semi-condensed` -

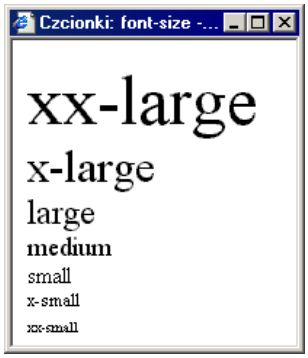
- `normal` - normalne odstępy między czcionkami (literami),
- `semi-expanded` -
- `expanded` - zwiększające się odstępy między czcionkami,
- `extra-expanded` -
- `ultra-expanded` - największe odstępy między czcionkami (literami).

Wartością domyślną jest `normal`. Dodatkowo właściwości `font-stretch` możemy przypisywać wartość `wider` i `narrower`. Pierwsza wartość określa kolejną (względem obowiązującej) wartość powiększenia odstępu między czcionkami. Druga z kolei kolejną (względem obowiązującej) wartość pomniejszenia odstępu między czcionkami.

7.1.3. Rozmiar czcionki

Do określania rozmiaru czcionki zdefiniowana została w CSS właściwość `font-size`. Może ona przyjąć wartość zdefiniowaną w jednej z czterech grup wartości:

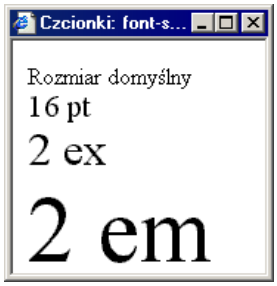
- wartości absolutne określające rozmiar
  - `xx-large`
  - `x-large`
  - `large`
  - `medium`
  - `small`
  - `x-small`
  - `xx-small`



- wartości relatywne (względne) określające rozmiar
  - `larger`
  - `smaller`

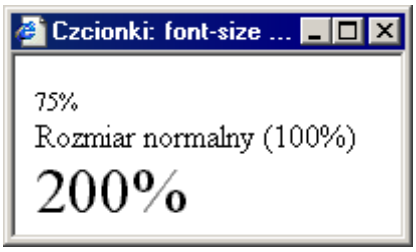
- wartości określające bezwzględny rozmiar czcionki (opisane w rozdziale 2.4.1), tj. `em`, `ex`, `px`, `pt`, `pc`, `in`, `mm` i `cm`.

```
<BODY>
 Rozmiar domyślny
 <BR>
 <SPAN style="font-size:16px">16 pt</SPAN>
 <BR>
 <SPAN style="font-size:4ex">2 ex</SPAN>
 <BR>
 <SPAN style="font-size:4em">2 em</SPAN>
</BODY>
```



- wartości procentowe (opisane w rozdziale 2.4.2), np.: `100%` czy `120%`.

```
<BODY>
 <SPAN style="font-size: 75%">75%</SPAN>
 <BR>
 Rozmiar normalny (100%)<BR>
 <SPAN style="font-size: 200%">200%</SPAN>
 <BR>
</BODY>
```



7.1.4. Właściwość 'font'

Ostatnią właściwością związaną z czcionkami, jaką opiszę jest właściwość `font`. Właściwość ta pełni funkcję podobną do właściwości `margin`, `padding` czy `border`, a konkretnie pozwala określić kilka parametrów czcionek jednocześnie (bez konieczności korzystania z właściwości jednostkowych, które wyżej opisałem). Przeznaczenie tej właściwości najlepiej zobrazuje przykład.

Zapis

```
P {
    font-weight : bold;
    font-size   : 14pt;
    font-family : verdana;
  }
```

możemy zastąpić uproszczoną formą

```
P { font: bold 14pt verdana }
```

Dla powyższej definicji parametrów czcionki akapitu, poniższy dokument

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>Czcionki: font</TITLE>
  <STYLE type="text/css">
    P { font: bold 14pt verdana }
  </STYLE>
</HEAD>

<BODY>
  <P>Czcionka: Bold Verdana 14pt</P>
</BODY>
</HTML>
```

w oknie przeglądarki będzie wyglądał następująco:

7.2. Teksty

7.2.1. Rozmieszczenie tekstu w kierunku poziomym i pionowym

- **'text-align'**

Właściwością odpowiedzialną za rozmieszczenie (wyrównanie) tekstu w dokumencie w kierunku poziomym jest `text-align` (dotyczy to również tekstu znajdującego się we wnętrzu tabel).

Właściwość ta może przyjąć jedną z czterech wartości:

- `left` - wyrównanie do lewej krawędzi,
- `right` - wyrównanie do prawej krawędzi,
- `center` - wyśrodkowanie względem obydwu krawędzi,
- `justify` - jednoczesne wyrównanie do lewej i prawej krawędzi.

Przykładowo:

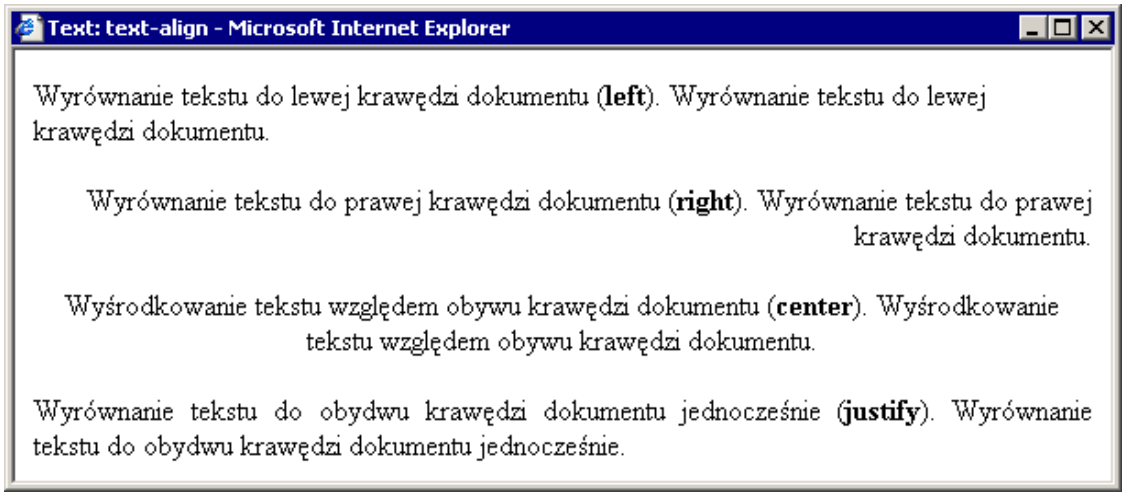
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>Text: text-align</TITLE>

  <STYLE type="text/css">
    .lewa { text-align: left }
    .praw { text-align: right }
    .cent { text-align: center }
    .just { text-align: justify }
  </STYLE>
</HEAD>

<BODY>
 <P class="lewa">Wyrównanie tekstu do lewej krawędzi dokumentu
    (<b>left</b>). Wyrównanie tekstu ...
 </P>
 <P class="praw">Wyrównanie tekstu do prawej krawędzi dokumentu
    (<b>right</b>). Wyrównanie tekstu ...
 </P>
 <P class="cent">Wyśrodkowanie tekstu względem obywu krawędzi
     dokumentu (<b>center</b>). Wyśrodkowanie tekstu ...
 </P>
 <P class="just">Wyrównanie tekstu do obydwu krawędzi dokumentu
     jednocześnie (<b>justify</b>). Wyrównanie tekstu ...
</P>
</BODY>

</HTML>
```

Efekt w oknie przeglądarki:



- **'vertical-align'**

Właściwością odpowiedzialną za rozmieszczenie tekstu w kierunku pionowym jest `vertical-align`. Właściwość ta może przyjmować wartości:

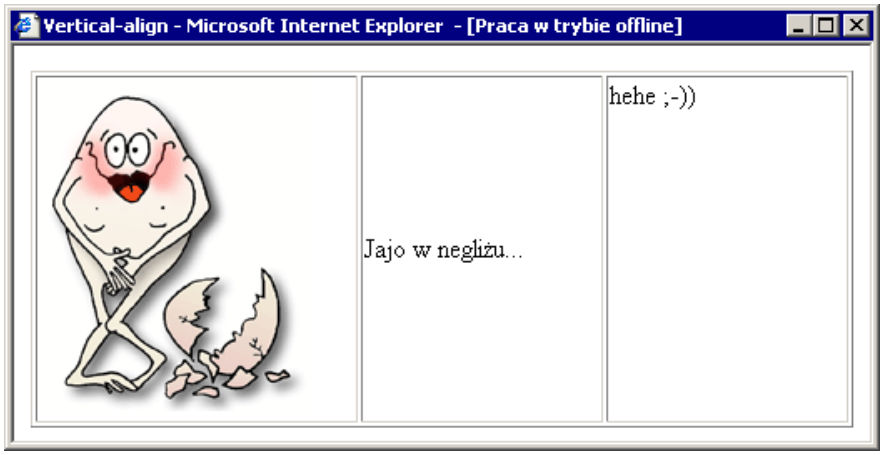
- `baseline` - wyrównanie tekstu do linii bazowej elementu rodzica (najczęściej do dolnej krawędzi),
- `text-top` - wyrównanie tekstu do górnej krawędzi elementu rodzica,
- `top` - wyrównanie tekstu do górnej krawędzi elementu rodzica,
- `middle` - wyśrodkowanie tekstu względem elementu rodzica,
- `text-bottom` - wyrównanie tekstu do dolnej krawędzi elementu rodzica,
- `bottom` - wyrównanie tekstu do dolnej krawędzi elementu rodzica,
- `sub` - indeks górny,
- `super` - indeks dolny.

Właściwość `vertical-align` jest wykorzystywana głównie do rozmieszczania fragmentów dokumentu w komórkach tabeli. Poniżej przedstawiam prosty przykład pokazujący rozmieszczanie obrazków i tekstu w tabeli.

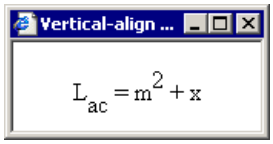
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>Text: text-align</TITLE>
</HEAD>

<BODY>
  <TABLE width="100%" border="1">
    <TR>
      <TD width="40%"><IMG src="acpic202.gif">
      <TD width="30%" style="vertical-align: middle">Jajo w negliżu...
      <TD width="30%" style="vertical-align: top">hehe ;-))
  </TABLE>
</BODY>
</HTML>
```

W przeglądarce zobaczymy:



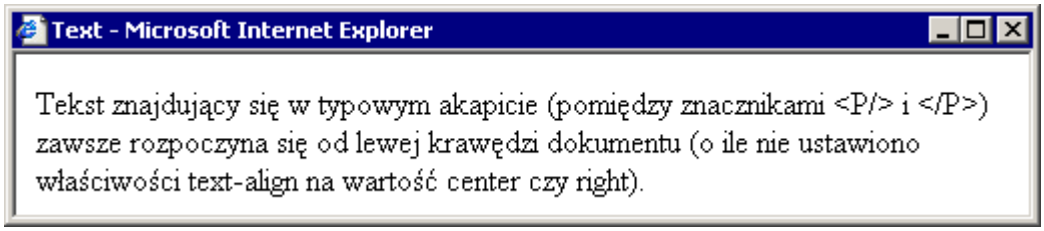
I jeszcze mały przykład:




```
<BODY>
  L<span style="vertical-align: sub">ac</span> = m
  <span style="vertical-align: super">2</span> + x
</BODY>
```

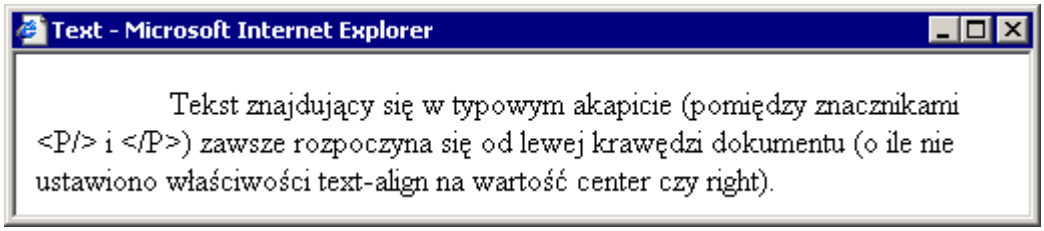
7.2.2. Wcięcie rozpoczynające akapit

Tekst znajdujący się w typowym akapicie (pomiędzy znacznikami `<P>` i `</P>`) zawsze rozpoczyna się od lewej krawędzi dokumentu lub elementu rodzica (o ile nie ustawiono właściwości `text-align` na wartość `center` czy `right`). Wygląda to np. tak:



Jeżeli zależy nam na tym, aby każdy akapit (czy też każdy inny blok tekstu) rozpoczynał się wcięciem, musimy zdefiniować dla niego właściwość `text-indent`. Właściwość `text-indent` może przyjmować wartości określające szerokość wcięcia opisane w pkt. 2.4.1 lub wartości procentowe opisane w pkt. 2.4.2 (związane z szerokością bloku tekstu).

Następujący efekt ...



uzyskamy definiując dla elementu `P` właściwość `text-indent`, na przykład w następujący sposób:

```
P { text-indent: 50pt }
```

I jeszcze kilka przykładów:

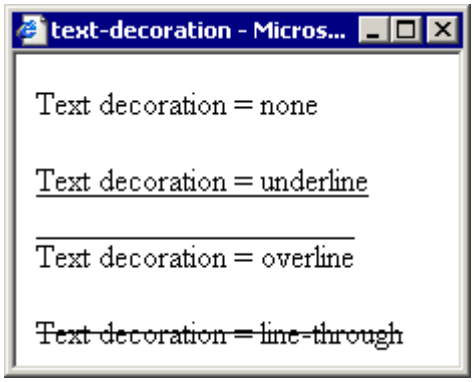
```
DIV { text-indent: 10% }
H1 { text-indent: 100px }
.klasa { text-indent: 3em }
```

7.2.3. Ozdobniki

Często zdarza się sytuacja, kiedy pewne fragmenty naszego dokumentu chcemy z różnych względów wyróżnić. Dla takich właśnie celów w CSS została zdefiniowana właściwość `text-decoration`, która może przyjmować następujące wartości:

- `none` - brak ozdobnika (zdjęcie ozdobnika),
- `underline` - tekst podkreślony (tekst nad linią),
- `overline` - tekst pod linią,
- `line-trough` - tekst przekreślony linią,
- `blink` - miganie tekstu.

Przykład zastosowania właściwości `text-decoration` z powyższymi wartościami:




```
<STYLE type="text/css">
 .none { text-decoration: none }
 .unln { text-decoration: underline }
 .ovln { text-decoration: overline }
 .lntr { text-decoration: line-through }
 .blnk { text-decoration: blink }
</STYLE>
```

Z oczywistych względów na rysunku obok nie widać wykorzystania ostatniej definicji (wartość `blink`).

Bardzo ciekawy efekt uzyskuje się definiując właściwość `text-decoration` dla pseudo-klas znacznika `<A>` :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>text-decoration</TITLE>
  <STYLE type="text/css">
    A:link { text-decoration: none }
    A:visited { text-decoration: none }
    A:active { text-decoration: none }
    A:hover { text-decoration: underline }
  </STYLE>
</HEAD>
```

```
<BODY>
  Odsyłacz hipertekstowy <A href="dalej.html"><B>[DALEJ]</B></A>
  zostaje podkreślony po najechaniu na niego kursorem myszy!
</BODY>
</HTML>
```

Dodanie różnych kolorów jeszcze bardziej uaktrakcyjni nasz odsyłacz hipertekstowy.

7.2.4. Transformacje tekstu

Kolejną właściwością pozwalającą modyfikować wygląd bloków tekstowych jest `text-transform`. Może ona przyjmować wartości:

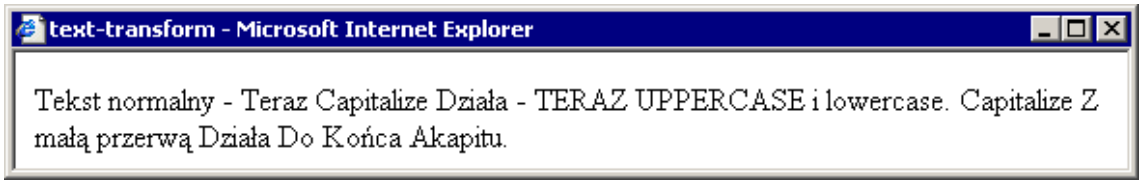
- `capitalize` - zamiana wszystkich pierwszych liter w bloku na wielkie,
- `uppercase` - zamiana wszystkich liter w bloku tekstowym na wielkie,
- `lowercase` - zamiana wszystkich liter w bloku tekstowym na małe,
- `none` - brak transformacji dla tekstu (zdjęcie transformacji).

Przykład wykorzystania właściwości `text-transform` :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>Text</TITLE>
  <STYLE type="text/css">
    .cp { text-transform: capitalize }
    .up { text-transform: uppercase }
    .lo { text-transform: lowercase }
    .no { text-transform: none }
  </STYLE>
</HEAD>

<BODY>
  Tekst normalny - <SPAN class="cp">teraz capitalize działa</SPAN> -
  <SPAN class="up">teraz uppercase</SPAN> i <SPAN class="lo">
  LOWERCASE</SPAN>. <SPAN class="cp">Capitalize z <SPAN class="no">
  małą przerwą</SPAN> działa do końca akapitu.</SPAN>
</BODY>
</HTML>
```

W oknie przeglądarki zobaczymy:



7.2.5. Odstępy między literami i wyrazami

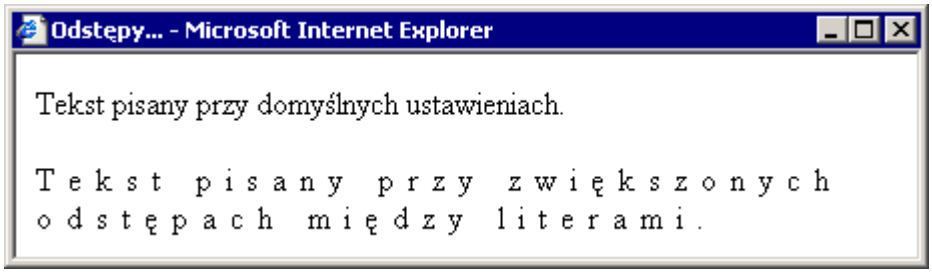
Jako ostatnie omówię właściwości pozwalające ustalać odstępy pomiędzy literami i wyrazami znajdującymi się w bloku tekstowym.

Najpierw zajmiemy się właściwością `letter-spacing`. Jak sama nazwa wskazuje, przy jej pomocy możemy określać odstępy znajdujące się pomiędzy literami. Właściwość ta może przyjmować wartości opisane w pkt. 2.4.1 lub predefiniowaną wartość `normal`, określającą odstęp domyślny dla danego kroju czcionki.

Przykładowo:

```
<HTML>
<HEAD>
  <TITLE>Odstępy...</TITLE>
  <STYLE type="text/css">
    P.szeroko { letter-spacing: 5pt }
  </STYLE>
</HEAD>
<BODY>
  <P>Tekst pisany przy domyślnych ustawieniach.</P>
  <P class="szeroko">Tekst pisany przy zwiększonych odstępach
   między literami.</P>
</BODY>
</HTML>
```

W przeglądarce zobaczymy:



Odstępy znajdujące się pomiędzy wyrazami pozwala definiować nam właściwość `word-spacing`. Właściwość ta może, podobnie jak `letter-spacing`, przyjmować wartości opisane w pkt. 2.4.1 lub predefiniowaną wartość `normal`, określającą odstęp charakterystyczny dla danego kroju czcionki.

8. Wygląd kursora

Kursor myszy, pokazujący bieżącą pozycję wskaźnika na ekranie może przyjmować różne kształty (ikonki). Domyślnie jego wygląd zależny jest od tego, na jaki element strony nim najedziemy, np. najechanie na odsyłacz hipertekstowy zmieni jego bieżący wygląd na rączkę. Dodatkowo dzięki właściwości `cursor` możemy samodzielnie określać jak ma zachowywać się kształt kursora.

Właściwość `cursor` może przyjmować wartości:

- `auto` - ustawienia domyślne,
- `crosshair` - kształt prostego krzyżyka,
- `default` - kształt zależny od używanej platformy (zwykle strzałka),
- `hand` - ręka,
- `move` - dwie skrzyżowane (prostopadłe) strzałki,

- `e-resize` - strzałka skierowana na wschód,
- `ne-resize` - strzałka skierowana na północny wschód,
- `nw-resize` - strzałka skierowana na północny zachód,
- `n-resize` - strzałka skierowana na północ,
- `se-resize` - strzałka skierowana na południowy wschód,
- `sw-resize` - strzałka skierowana na południowy zachód,
- `s-resize` - strzałka skierowana na południe,
- `w-resize` - strzałka skierowana na zachód,
- `text` - kształt wielkiej litery I,
- `wait` - kształt symbolizujący upływ czasu (zwykle klepsydra),
- `help` - znak zapytania.

Jeżeli chcemy, aby kursor zmienił się na rękę, gdy znajdzie się nad elementem `P` musimy właściwości `cursor` przypisać wartość `hand`, np.:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN">
<HTML>
<HEAD>
  <TITLE>Kursor</TITLE>
  <STYLE type="text/css">
    P { color  : red;
        cursor : hand;
      }
  </STYLE>
</HEAD>

<BODY>
  <H1>Kursory</H1>
  <P>Akapit - i kursor jest reką
  <P>Tu też...
</BODY>
</HTML>
```

Literatura

1. Lemay L., HTML 4 – vademecum profesjonalisty, Helion, Gliwice 1998
2. Campbell B., Damell R., Dynamic HTML, Helion, Gliwice 1998
3. Goodman D., Dynamic HTML: The Definitive Reference, O'Reilly, 1998
4. Romowicz W., HTML i JavaScript, , Helion, Gliwice 1998
5. Flanagan D., JavaScript: The Definitive Guide, 3rd Edition, O'Reilly, 1998
6. Musciano Ch., Kennedy B., HTML – podręcznik użytkownika, Wydawnictwo RM, W-wa 1999
7. Burns J., HTML Goodies, Rodzynki języka HTML, Mikom, W-wa 1999
8. Niederst J., HTML. Leksykon kieszonkowy, Helion, Gliwice 2000
9. Castro E., Po prostu HTML 4 (wydanie II), Helion, Gliwice 2000
10. Laurie B., Laurie P., „Apache – przewodnik encyklopedyczny", Helion, Gliwice 2000
11. Stigler M., Lisenbardt M., IIS 4 i Proxy Server 2, Mikom, W-wa 2000
12. Kelli A., Internet Information Services. Administracja, Wyd. Robomatic, Wrocław 2000
13. Harold E., XML. Księga eksperta, Helion, Gliwice 2000
14. Musciano Ch., Kennedy B., HTML & XHTML: The Definitive Guide 4th Edition, O'Reilly, 2001
15. Bradenbaugh J., JavaScript – Receptury, Helion, Gliwice 2001
16. Wong C., „HTTP. Leksykon kieszonkowy", Helion, Gliwice 2001

Internet

World Wide Web Consorcium - http://www.w3.org
HTML 4.01 Specification - http://www.w3.org/TR/html4/
CSS, Level 1.0 Specification - http://www.w3.org/TR/REC-CSS1/
CSS, Level 2.0 Specification - http://www.w3.org/TR/REC-CSS2/
HTTP Specification - http://www.w3.org/Protocols/
XHTML Specification - http://www.w3.org/TR/xhtml1/

Index