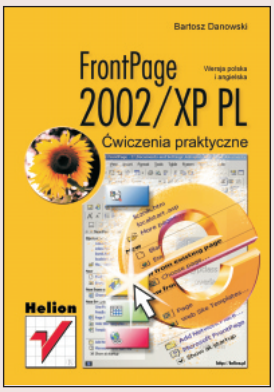
FrontPage 2002/XP PL.
Ćwiczenia praktyczne



Autor: Bartosz Danowski
ISBN: 83-7197-660-7
Format: B5, stron: 112

Od pojawienia się i spopularyzowania Internetu minęło już sporo czasu. Obecna sieć bardzo różni się od tej, jaką znamy sprzed lat. Postęp przejawia się niemal w każdym aspekcie: zarówno nowe technologie, usługi, jak i programowanie zaskakują nas codziennie. Projektowanie pierwszych stron WWW wymagało od nas znajomości tajemnego kodu i wielu wyrzeczeń. W chwili premiery najnowszej wersji pakietu MS Office XP oraz edytora MS FrontPage XP wchodzącego w jego skład, wszystkie dawne problemy odejdą w zapomnienie. Strona tworzy się na naszych oczach niemal sama, przy udziale różnego rodzaju kreatorów i innych udogodnień.

Dzięki tej książce poznacie najnowszą wersję tego znanego i kontrowersyjnego edytora stron WWW, pracującego w trybie graficznym. Praca z nowym edytorem stała się prawdziwą przyjemnością, a wszelkiego rodzaju wady znane ze starszych wersji zostały poprawione. Budowa książki pozwoli szybko i łatwo poznać program i zbudować własną witrynę WWW. Szereg dokładnych opisów oraz ilustracji będzie dodatkowym ułatwieniem. Myślę, że cenny dla niemal każdego jest oddzielny rozdział poświęcony kaskadowym arkuszom stylów, które pozwalają na zastosowanie rewolucyjnych rozwiązań na stronach WWW. Dodatkowo wiele przykładów zawiera odniesienie do czystego języka HTML, dzięki czemu będzie łatwiej zrozumieć pewne pojęcia, co z pewnością zwiększy również zainteresowanie samym językiem HTML, który nadal daje nam lepszą kontrolę nad projektem.

Dla początkujących projektantów książka jest doskonałym wprowadzeniem do pracy z edytorem. Przyda się również tym, którzy znają już MS FrontPage i chcieliby jedynie zapoznać się z nowościami oferowanymi w najnowszej wersji. Mam nadzieję, że zawarte tu przykłady i ćwiczenia okażą się pomocne w poznaniu programu i jego możliwości.

W książce opisano opcje zarówno dla polskiej, jak i angielskiej wersji MS FrontPage 2002/XP.

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

Spis treści

Rozdział 3.

Kaskadowe arkusze stylów na przykładzie MS FrontPage 2002/XP PL

Długo się zastanawiałem, jak bez sięgania do teorii opisać tak ważny element, jak kaskadowe arkusze stylów zaimplementowane w *MS FrontPage 2002/XP PL*. Doszedłem do wniosku, że nie jest to możliwe, dlatego w dalszej części rozdziału zamieściłem dość dokładny opis zasad konstrukcji i działania arkusza *CSS*, a następnie zanalizowałem, jak cała ta teoria ma się do *MS FrontPage 2002/XP PL*. Mam nadzieję, że takie rozwiązanie będzie najlepsze. Wydaje mi się, że dokładne przeczytanie poniższych podrozdziałów będzie pomocne w zrozumieniu zasady działania implementacji *CSS* w edytorze.

Krótkie wprowadzenie do CSS

Wyjaśnijmy najpierw, czym są *kaskadowe arkusze stylów*. Pojęcie to pojawiało się w poprzednim rozdziale prawie na każdej stronie, ale nie zostało dokładnie omówione. Od czasu pojawienia się wersji *HTML 3.2* wprowadzono do struktury języka pewne innowacje i zaczęto część znaczników zastępować innymi. Nowe znaczniki nazwano kaskadowymi arkuszami stylów i zaczęto stopniowo je rozbudowywać. Dzięki stylom możemy mieć pełną kontrolę nad formatowaniem dokumentu. Żadne z poleceń języka *HTML* nie pozwalało nam na regulowanie odstępów pomiędzy blokami oraz nakładanie ich na siebie. Warto wspomnieć, że style pozwalają na kontrolę tła poszczególnych części dokumentu, właściwości stosowanych na stronie czcionek, tabel, formularzy i wielu innych elementów.

Kaskadowe arkusze stylów, w skrócie *CSS*, możemy określić jako narzędzie formatowania wyglądu dokumentów. Nie możemy jednak stworzyć za ich pomocą strony. Podobnie jak język *HTML*, kaskadowe arkusze stylów są standaryzowane przez *W3* i tam należy szukać odpowiedniej dla nich specyfikacji. Na dzień dzisiejszy dostępne są specyfikacje w wersji 1. oraz 2. Konsorcjum W3 pracuje aktualnie nad trzecią specyfikacją, która nie została jeszcze zatwierdzona, ale jej projekt jest dostępny razem z obowiązującymi dokumentami.

Używanie stylów wiąże się z pewnym ryzykiem, gdyż przeglądarki interpretują tak formatowaną stronę na różne sposoby. Poczynając od wersji 3. przeglądarek, specyfikacja pierwsza jest obsługiwana w mniejszym lub większym stopniu. Czwarta generacja przeglądarek jest bardziej zgodna ze specyfikacją. Podobnie jak w przypadku języka *HTML*, przeglądarka MS Internet Explorer o wiele lepiej radzi sobie z *CSS* niż Netscape Navigator.

Nasze arkusze możemy, podobnie jak język HTML, opatrywać komentarzem wewnątrz. Komentarz musi być umieszczony w następujący sposób:

```
/* komentarz dla dokumentu css */
```

Atrybuty stylów

Jak wspominałem we wprowadzeniu, style mogą formatować wiele elementów strony, ale w tym opracowaniu skupimy się jedynie na formatowaniu czcionek, tekstu, kolorów, tła, marginesów oraz pozycjonowaniu (*MS FrontPage 2002/XP PL* obsługuje tylko małą tego część).

Zacznijmy od umieszczania stylów na naszej stronie. Pierwszą możliwością jest deklaracja stylów wewnątrz dokumentu:

```
<html>
<head>
   <STYLE TYPE="text/css">
   <!--
 Deklaracje stylów
    -->
   </STYLE>
   </head>
   <body>
   </body>
   </html>
```

Jak widać na przykładzie, deklaracja stylów jest umieszczana w nagłówku dokumentu pomiędzy znacznikami *<style> </style>*. Takie rozwiązanie jest stosunkowo wygodne i ma ogromną przewagę nad umieszczaniem deklaracji stylów bezpośrednio w kodzie HTML, gdyż posiadamy wtedy stosunkowo prostą możliwość formatowania dokumentu.

Kolejną możliwością umieszczania stylów w naszym dokumencie jest przygotowanie ich w oddzielnym pliku.

Plik ten musi mieć rozszerzenie *CSS*: *nazwa.css*. Tak przygotowany plik wywołujemy w naszym dokumencie *HTML* w następujący sposób:

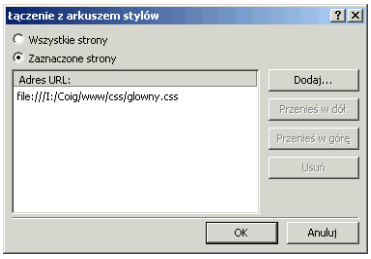
```
<html>
<head>

<LINK REL=STYLESHEET HREF="nazwa.css" TYPE="text/css">
</head>
<body>
</body>
</html>
```

Rozwiązanie to ma wiele zalet. Tak przygotowany arkusz może być zastosowany w każdym pliku *HTML* naszej strony; ułatwia to ingerowanie w wygląd tejże strony.

Wszystkie opisy i ćwiczenia przedstawione w dalszej części rozdziału opierać się będą na tym rozwiązaniu.

Podpinanie zewnętrznego arkusza w edytorze *MS FrontPage 2002/XP PL* jest możliwe dzięki opcji *Łącza arkusza stylów (Sheet Style Links)* w menu *Format (Format)*. Dodawanie arkusza odbywa się za pomocą przycisku *Dodaj (Add)* i typowego okna, znanego z wyszukiwania plików na dysku. Przydatną opcją jest również *Usuń formatowanie (Remove Formatting)* z menu *Format (Format)*, usuwa ona wszystkie wpisy formatujące znajdujące się w kodzie *HTML*.

Rysunek 3.1.
Podłączanie zewnętrznego arkusza



Definiowanie arkusza rozpoczynamy od nadania mu etykiety (w celach informacyjnych):

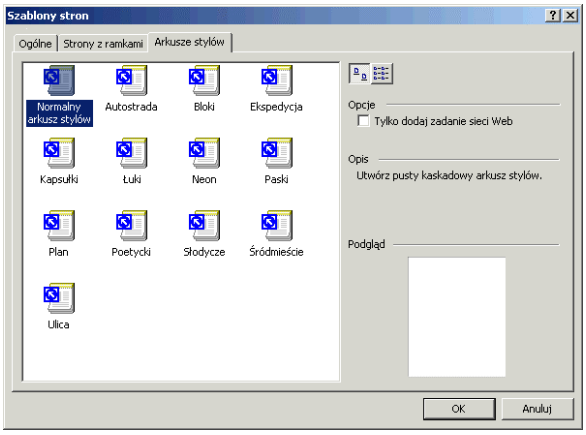
```
/*

Przykład arkusza
*/

body, p
    {
    Font-Size: 10pt;
    Font-Family: 'Times New Roman', Verdana;
    }
```

Jak widzimy na przykładzie, pierwsza część określa, który element języka *HTML* ma być formatowany przez *CSS* — w naszym przypadku zajmiemy się <*BODY*> oraz <*P*>. Opis formatowania znajduje się pomiędzy nawiasami klamrowymi *{}* i składa się z dwóch członów. Pierwszy oddzielony jest od drugiego dwukropkiem, a po ostatnim znajduje się średnik.

Przy formatowaniu kroju czcionki możemy zadeklarować kilka wartości, rozdzielając je przecinkami. Zalecany przez nas krój warto zamknąć pomiędzy znakami ' '.

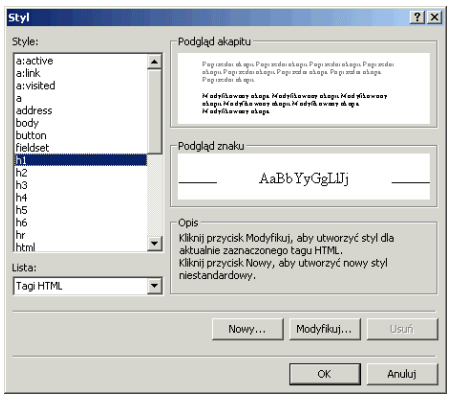
Definicja stylów w *MS FrontPage 2002/XP PL* polega na utworzeniu nowego, pustego pliku arkusza. Robimy to za pomocą menu *Plik/Nowy/Strona* lub *sieć Web (File/New/Page or Web)*, który otwiera *Okno zadań (Task Panel)*. Z *Okna zadań (Task Panel)* znajdującego się po prawej stronie wybieramy opcję *Szablony strony (Page Templates)*. W nowo otwartym oknie przechodzimy do zakładki *Arkusze stylów (Style Sheets)*, na której znajduje się kilkanaście zdefiniowanych typowych arkuszy oraz jeden pusty *Normalny arkusz stylów (Normal Style Sheet)*, z którego będziemy korzystać.

Rysunek 3.2.
Nowy arkusz stylów



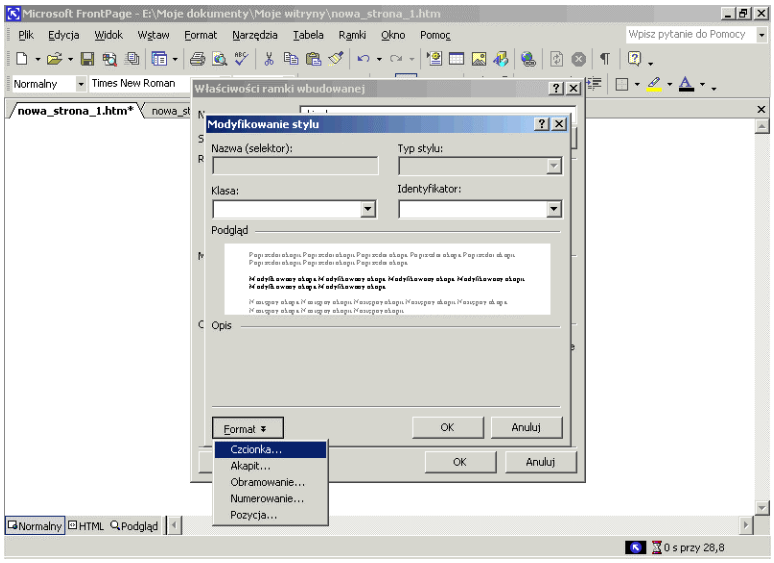
Został utworzony nowy dokument, który jest widoczny na pasku zakładek plików otwartych w *MS FrontPage 2002/XP PL*, oraz pasek z przyciskiem *Styl (Style)*, dzięki któremu możemy definiować nowe wpisy do arkusza.

Rysunek 3.3.
Definicja wpisów



Na powyższym rysunku dokładnie widać spis wszystkich dostępnych znaczników, które możemy dowolnie formatować za pomocą przycisku *Modyfikuj (Modify)*. Formatowanie odbywa się za pomocą nowo otwartego okna.

Rysunek 3.4.
Definicja wpisów — ciąg dalszy



Za pomocą przycisku *Format* możemy zmienić następujące atrybuty dla każdego z obiektów:

- *Czcionka (Font)* — krój, wygląd, wielkość, kolor i inne właściwości, omawiane przy okazji właściwości tekstu,
- *Akapit (Paragraph)* — wszystkie właściwości bloków tekstu omawiane na początku książki,
- *Obramowanie (Border)* — obramowania i ramki dowolnych obiektów,
- *Numerowanie (Numbering)* — właściwości list,
- *Pozycja (Position)* — pozycjonowanie elementów.

Dzięki opcjom *Klasa (Class)* i *Identyfikator (ID)* możemy w prosty sposób odwołać się do już istniejących klas lub ID (o których napiszę za chwilę).

Po zatwierdzeniu wszystkich deklaracji w naszym arkuszu powinny pojawić się pierwsze wpisy, podobne do tych z powyższych przykładów.

Selektory

Selektory to podstawa *CSS*. Ich zadaniem jest wskazywanie obiektu, któremu przypisujemy jakąś wartość. Przykłady zastosowania selektorów znajdowały się przy opisie poszczególnych elementów *CSS*. Tutaj postaram się jeszcze raz wyjaśnić dokładniej rolę selektorów. Przykładem może być <*P*>:

```
P {font-family: Arial;}
```

Selektor ten przypisze Arial jako czcionkę domyślną dla każdego znacznika <*P*>.

Specyfikacja *CSS* pozwala nam na grupowanie selektorów:

```
H1, H2, H3 {color: red;}
```

Dzięki temu wszystkie nagłówki *H1*, *H2*, *H3* będą miały kolor czerwony.

Grupowanie selektorów może wyglądać również następująco:

```
P, DIV,
{
color: navy;
}
```

Z selektorami spotkaliśmy się w *MS FrontPage 2002/XP PL* wcześniej, przy omawianiu przykładów, teraz jedynie dowiedzieliśmy się, że tak się nazywają i że można je dowolnie grupować.

Klasy

Prawie każdy znacznik języka *HTML* 4 zawiera atrybut o nazwie *CLASS=" "*, który odpowiada za jego formatowanie za pomocą *CSS*. Zasada jego działania jest bardzo prosta. W pliku arkusza stylów definiujemy jakąś klasę, którą potem możemy wywołać za pomocą atrybutu *CLASS=" "* w dowolnym znaczniku. Metoda wywołania nie jest skomplikowana:

```
<p class="nazwa_klasy"></p>
```

W *MS FrontPage 2002/XP PL* przypisywanie klas dla elementów *HTML* polega na przejściu do tekstowego trybu edycji i przypisaniu odpowiedniemu elementowi atrybutu *class=" "*. By ograniczyć zakres poszukiwań, możemy posłużyć się prostą sztuczką: w trybie graficznym ustawiamy kursor w miejscu, dla którego ma być przypisana klasa i dopiero wtedy przechodzimy do zakładki *HTML*. Kursor zostanie automatycznie umieszczony w odpowiednim miejscu.

Konstrukcja klasy w arkuszu stylów również nie powinna sprawić problemu:

```
.nazwa_klasy
 { font-size: 10pt;}
```

Definiowanie klas może też przybierać następującą postać:

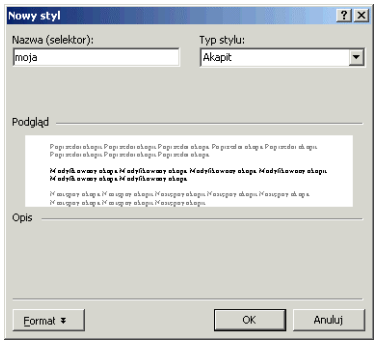
```
P {font-size: 10pt;}
P.maly {font-size: 8pt;}
```

Taka konstrukcja określa dla każdego znacznika <*P*> domyślną wielkość na 10 pt. Natomiast druga linia,*P.maly {font-size: 8 pt;}*, tworzy klasę definiującą tekst o wielkości 8 pt. W tworzonym dokumencie wszystkie elementy zamknięte w znaczniku <*P*> będą posiadały wielkość 10 pt, natomiast elementy zamknięte w <*P class="maly"*> pokażą tekst o wielkości 8 pt.

Jak widzimy, jest to rozwiązanie bardzo wygodne; pozwala ono różnicować wyglądy poszczególnych selektorów.

Edytor *MS FrontPage 2002/XP PL* pozwala nam definiować własne klasy. Proces definiowania polega na kliknięciu przycisku *Styl (Style)* i wybraniu przycisku *Nowy (New)* z nowo otwartego okna (pojęcie to jest nam dobrze znane).

Rysunek 3.5.
Definicja klasy



Proces definicji naszej klasy polega na wpisaniu w polu *Nazwa selektor (Name (selector))* dowolnej nazwy i określeniu odpowiednich reguł za pomocą przycisku *Format*.

Kaskadowe arkusze stylów posiadają pewne zdefiniowane, standardowe klasy — *pseudoklasy*. W pierwszej specyfikacji *CSS* zdefiniowano ich kilka; odnoszą się one do następujących elementów: tekstu i odnośników. Najpierw zajmiemy się najpopularniejszymi *pseudoklasami*, czyli odnośnikami.

Każdy na pewno widział na stronach odnośniki bez podkreśleń, które często zmieniają kolor w chwili przesuwania ponad nimi kursora myszy (efekt bajecznie prosty do uzyskania za pomocą *CSS* i *pseudoklas*). Za deklarowanie wyglądu odnośników odpowiadają następujące pseudoklasy:

```
A:LINK
A:VISITED
 A:ACTIVE
```

Pseudoklasy używa się następująco:

```
A:link /* standardowy odnośnik */
  {
  font-size: 12pt;
  color: navy;
  }
A:visited /* odnośnik odwiedzony */
  {
  font-size: 12pt;
  color: blue;
  }
A:hover /* odnośnik po najechaniu na niego myszką */
  {
  font-size: 12pt;
  color:red;
  }
```

Efektem działania takiej definicji będzie zmiana koloru wszystkich odnośników na stronie na NAVY — granat i wielkość 10 pt, a także zmiana odwiedzonych odnośników na BLUE — niebieski oraz zmiana odnośników, nad którymi znajduje się kursor myszy, na RED — czerwony.

Efekt działania definicji jest nam już znany; wiemy, że odnosi się on globalnie do każdego odnośnika na stronie, na której używamy tego arkusza. Na pewno wielu z Was nasuwa się pytanie, co zrobić, by zróżnicować odnośniki w ramach jednej strony. Sprawa wbrew pozorom jest stosunkowo prosta: musimy utworzyć dodatkowe klasy dla naszych odnośników.

Przykład poniżej przedstawia, jak to powinno wyglądać.

```
A.maly:link /*odnośnik mniejszy od standardowego*/
  {
  font-size: 9pt;
  }
A.maly:visited /* odnośnik mniejszy od odwiedzonego */
  {
  font-size: 9pt;
  }
A.maly:hover /* odnośnik mniejszy po najechaniu na niego myszką */
  {
  font-size: 9pt;
  }
```

Zwróćmy uwagę, że przy deklaracji nowej klasy użyliśmy już tylko innych wielkości, gdyż chcieliśmy jedynie pomniejszyć nasz odnośnik. Pozostałe wartości zostały bez zmian przeniesione z głównej definicji wyglądu odnośników. O dziedziczeniu należy pamiętać i warto z niego korzystać w pracy nad dokumentami *HTML*.

MS FrontPage 2002/XP PL posiada pseudoklasy na liście dostępnych znaczników, dzięki czemu będziemy w stanie je sformatować w bardzo prosty sposób. Dodatkowo, w oknie właściwości strony znajduje się odpowiednia opcja, pozwalająca na włączenie zmian odnośników po najechaniu na nie kursorem myszy. Moim zdaniem lepiej jednak zdefiniować odpowiednie wpisy w arkuszu samodzielnie.

ID

W przeciwieństwie do klas, które są dziedziczone przez kolejne znaczniki, *ID* odnosi się tylko do jednego znacznika, któremu przypiszemy odpowiednie *ID*. Definiowanie *ID* w arkuszu stylów wygląda tak:

```
#zielony { color: green;}
```

Wywołanie takiego *ID* w dokumencie przedstawia się następująco:

```
<p ID="zielony"> Ten tekst będzie zielony </p>
```

Efektem działania takiej konstrukcji będzie wyświetlenie zawartości <*P*> w kolorze zielonym. Zakres działania *ID* ograniczy się wyłącznie do tego jednego <*P*>.

Oczywiście, działać będzie również taka deklaracja:

```
Definicja w arkuszu:
  .zielony { color: green;}
  Wywołanie w HTML
  <p CLASS="zielony"> Ten tekst będzie zielony </p>
```

dla tego jednego znacznika *<P>*, ponieważ zadeklarowaliśmy klasę, ale nie przypisaliśmy do niej żadnego selektora. Rozwiązanie to jest nieco mniej eleganckie, ale efektywne.

Jak na pewno zaobserwowałeś, przy formatowaniu jednego selektora lub klasy starałem się skrupulatnie mieszać różne elementy *CSS*. Chciałem uzmysłowić Wam, jak potężne możliwości daje *CSS*. Na koniec podam jeszcze przykład, który nie powinien być dla Was zaskoczeniem.

```
A:link /*odnośnik mniejszy od standardowego*/
   {
   font-size: 9pt;
   color: blue;
   }
A:visited /* odnośnik mniejszy od odwiedzonego */
   {
   font-size: 9pt;
   color: green;
   background-color: silver;
   }
A:hover /* odnośnik mniejszy po najechaniu na niego myszką */
   {
   font-size: 9pt;
   color: red;
   background-color: yellow;
   }
```

Jak widać, określiłem kolor dla tła pod naszymi odnośnikami. Domyślnie odnośnik nie posiada żadnego zdefiniowanego koloru dla tła. Odwiedzony odnośnik będzie miał tło zielone, a odnośnik, nad którym właśnie znajduje się kursor myszki, zmieni kolor tła na żółty, natomiast tekst będzie czerwony. Skoro potrafimy w tak prosty sposób zmieniać kolor tła odnośnika, to wydaje nam się, że nie musimy już nikogo przekonywać do stosowania *CSS*.

Niestety, klasa *:Hover* nie jest obsługiwana przez przeglądarkę Netscape Navigator, ale mam nadzieję, że zostanie ona szybko wprowadzona do użytku (wersja beta szóstego wydania już interpretuje tę klasę).

Warto wspomnieć, że w celu uzyskania podobnego efektu (bez użycia *CSS*) musielibyśmy sięgać po *Java Script* i pisać dość rozbudowany skrypt, który musiałby przewidywać dodatkowo rodzaj zastosowanej przeglądarki.

MS FrontPage 2002/XP PL nie obsługuje wielu elementów *CSS*, ale nic nie stoi na przeszkodzie, byśmy sami dopisali je do naszego arkusza ręcznie. Należy przy tym zachować konstrukcję stosowaną przez edytor.

Poniżej, w tabelach, znajduje się opis poszczególnych deklaracji *CSS* i dostępnych wartości. Dodatkowo znajdują się w niej proste przykłady dla każdego z elementów.

Tabela 3.1. *Właściwości czcionek*

Deklaracja	Dostępne wartości
font-family	Dostępne kroje pisma: p {font-family: Verdana, Arial;}
font-style	normal | italic | oblique | inherit p {font-style: italic;}
font-variant	normal | small-caps | inherit p {font-variant: small-caps;}
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit p {font-weight: bold;}
font-size	absolute-size | relative-size | length | percentage | inherit p {font-size: 10pt;}

Przykładem definicji rozbudowanego arkusza, określającego parametry czcionki wewnątrz akapitu <*P*>, może być poniższy fragment listy. Efektem jej działania będzie wyświetlenie tekstu za pomocą pogrubionej czcionki VERDANA o wielkości 16 pt.

```
P
{
font-family: 'Verdana';
font-size: 16pt;
font-weight: bold:
}
```

Fragment ten może również dotyczyć innych elementów, np.:

```
P, DIV
{
font-family: 'Verdana';
font-size: 16pt;
font-weight: bold:
}
```

Ten wpis będzie formatował bloki tekstu zamknięte w <*P*> oraz <*DIV*>. O wygodzie przedstawionego powyżej wpisu nie muszę chyba nikogo przekonywać.

Tabela 3.2. *Właściwości tekstu*

Deklaracja	Dostępne wartości
text-decoration	none | underline | overline | line-through | blink p {text-decoration: underline;}
text-transform	capitalize | uppercase | lowercase | none p {text-transform: capitalize;}
text-align	left | right | center | justify p {text-align: left;}
text-indent	length | percentage p {text-indent: 3pt;}
vertical-align	baseline | sub | super | top | text-top | middle | bottom | text-bottom | percentage p {vertical-align: baseline;}
letter-spacing	normal | length p {letter-spacing: 3pt;}
word-spacing	normal | length p {word-spacing: 3pt;}

Nasz tekst możemy również dowolnie formatować za pomocą *CSS* — podkreślać, przekreślać, regulować odstępy pomiędzy słowami i znakami, równać do lewej, prawej, środkować i justować. Dodatkowo całość możemy również pozycjonować w pionie, co jest szczególnie przydatne przy pozycjonowaniu tekstu w tabeli. Przykładem zastosowania formatowania właściwości tekstu jest poniższy wpis:

```
TD
{
text-align: center;
vertical-align: bottom;
     }
```

Dzięki takiemu arkuszowi będziemy mogli wyśrodkować zawartość komórek w tabeli zarówno w pionie, jak i w poziomie.

Tabela 3.3. *Kolor i tło dokumentu*

Deklaracja	Dostępne wartości
color	deklaracje koloru w postaci: #FFFFFF p {color: #FFCCFF;}
background-color	color | transparent | inherit h1 {background-color: #FF00CC}
background-image	url | none | inherit body {background-image: url(obrazek.jpg)}
background-repeat	repeat | repeat-x | repeat-y | no-repeat | inherit body {background-repeat: repeat-y;}
background-attachment	scroll | fixed | inherit body {background-image: url(imagefilename);} background-attachment: fixed;}
background-position	top | center | bottom | left | center | right percentage | length | percentage | length background-position: 0% 3cm; background-position: 100%; background-position: 100% 100%;

Definicja kolorów i tła jest szczególnie ważna przy projektowaniu stron. Każdy chce mieć możliwość zdefiniowania tła strony, tła tabeli, tła będącego obrazkiem, a także koloru tekstu. Poniżej podaję przykład, w którym połączyłem kilka wcześniej poznanych właściwości:

```
BODY
   {
   background-color: white;
   background-image: url(images/nazwa.gif);
   background-repeat: no-repeat;
   color: black;
   }
P
   {
   font-style: Verdana;
   font-size: 12pt;
   text-align: justify;
   color: navy;
   }
```

Taka lista *CSS* pozwoliła nam określić następujące właściwości:

BODY

- kolor tła — biały;
- obrazek tła — nazwa.gif;
- brak powtarzania tła;
- domyślny kolor tekstu — czarny.

P

- czcionka — Verdana;
- wielkość — 12 pt;
- tekst — wyjustowany;
- kolor tekstu akapitu — granatowy.

Tabela 3.4. *Marginesy*

Deklaracja	Dostępne wartości
margin-top margin-right margin-left margin-bottom	length | percentage | auto | inherit body {margin-top: 1em; margin-right: 2em; margin-bottom: 3em; margin-left: 2em;}
margin	length | percentage | auto | inherit BODY {margin: 1em, 2em;}
padding-top padding-right padding-bottom padding-left	length | percentage | auto | inherit H1 {margin-top: 2em;}
padding	length | percentage | length | percentage | inherit H1 {padding: 1em, 2em;}

Wprowadzenie formatowania marginesów za pomocą *CSS* było wielkim krokiem naprzód. Wyobraźmy sobie, że tworzymy stronę zawierającą kilka akapitów tekstu i sytuacja wymaga od nas, by co drugi akapit był w całości wcięty bardziej niż pozostałe. Wobec braku *CSS* musimy użyć tabeli. Poniżej przedstawiam przykład.

Treść pierwszego akapitu bez wcięcia...	
	Treść drugiego akapitu wciętego w całości w stosunku do pierwszego...
Treść trzeciego akapitu bez wcięcia....	

Dla takiej tabeli musieliśmy określić krawędzie jako niewidoczne i wówczas formatowanie zostało wykonane.

Dziś z pomocą przychodzi nam *CSS*, możemy zapomnieć o stosowaniu tabeli. Poniżej przedstawiam ten sam przykład zdefiniowany w oparciu o *CSS*.

Definicja w arkuszu stylów:

```
P
 {
 font-size: 10pt;
 margin-left: 0;
 }
  P.wciety
  {
  margin-left: 5%;
  }
```

Wygląd kodu HTML:

```
<P>Treść pierwszego akapitu bez wcięcia</p>
<P class="wciety">Treść drugiego akapitu wciętego w całości w stosunku do
pierwszego...</P>
<P>Treść trzeciego akapitu bez wcięcia</P>
```

Jak widać, definiowanie oparte na *CSS* jest znacznie wygodniejsze niż definiowanie tabeli. Warto zwrócić uwagę, że w przypadku pierwszego rozwiązania, formatowanie tekstu wewnątrz tabeli za pomocą *CSS* może nam przysporzyć wielu problemów z przeglądarką Netscape Navigator. Natomiast drugie rozwiązanie jest znacznie bardziej dla niej przyjazne.

Tabela 3.5. *Listy*

Deklaracja	Dostępne wartości
list-style-type	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none OL {list-style-type: lower-alpha;}
list-style-image	inside | outside UL {list-style-image: url(images/obrazek.gif)}
list-style-position	length | percentage | auto | inherit UL {list-style: outside}
list-style	[disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none] || [inside | outside] || [<url> | none] UL {list-style: upper-roman inside}

Przy okazji omawiania list w języku *HTML* wspominałem, że *CSS* oferuje dość rozbudowane wsparcie dla formatowania tego elementu. Domyślnie HTML może definiować listy posortowane *<OL>* oraz nieposortowane *<UL>*.

Jeśli chodzi o listy sortowane, domyślnie są one numerowane (1, 2, 3...), a uzyskanie list numerowanych za pomocą liczb było możliwe dzięki atrybutowi *TYPE=" "*. Jednak HTML 4 uznaje ten atrybut za przestarzały i sugeruje użycie do tego celu *CSS*. Definiowanie listy numerowanej za pomocą liter w *CSS* wygląda następująco:

```
OL
  {
  list-style-type: lower-alpha;
  color: red;
  text-align: left;
  margin-left: 5%;
  }
```

Tym wpisem do arkusza stylów określiliśmy, że nasza lista <*OL*> ma być numerowana za pomocą małych liter. Lista będzie miała kolor czerwony i zostanie wyrównana domyślnie do lewego marginesu, który ma być wcięty o 5% w stosunku do pozostałych elementów na stronie.

Jak zauważyliście, rozdział ten nie zawiera ćwiczeń, ale nie martwcie się, wszystko nadrobimy w trakcie realizacji projektu z kolejnego rozdziału.