ABC języka HTML i XHTML

Autor: Maria Sokół
ISBN: 83-246-0753-6
Format: B5, stron: 256
Przykłady na ftp: 28 kB

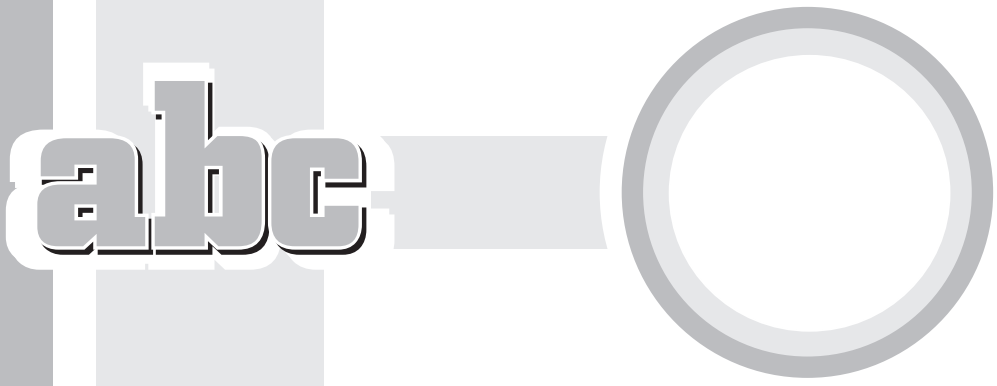
Dynamiczny rozwój internetu sprawia, że pojawia się w nim coraz więcej witryn WWW. Swoją „wizytówkę" w internecie chcą mieć nie tylko przedsiębiorstwa i organizacje, ale również użytkownicy domowi, dla których strona WWW może być miejscem do zaprezentowania swoich fotografii, podzielenia się wspomnieniami z podróży lub po prostu przedstawienia się. Wiele firm oferujących konta WWW udostępnia również mechanizmy do tworzenia własnych stron opartych na gotowych szablonach, jednak przygotowanie naprawdę „własnej" witryny internetowej wymaga przynajmniej elementarnej znajomości jej podstawowego budulca – języka HTML.

Książka „ABC języka HTML i XHTML" to wprowadzenie do tematu całkowicie samodzielnego tworzenia własnych witryn WWW. Przedstawia najnowszą specyfikację języka HTML i jego wersji rozwojowej, XHTML. Czytając tę książkę, poznasz strukturę dokumentów HTML i znaczniki wykorzystywane do określania wyglądu strony. Dowiesz się, w jaki sposób umieszczać na stronie elementy tekstowe i formatować je, tworzyć hiperłącza i mechanizmy nawigacyjne oraz wstawiać do dokumentów obrazy w formatach: GIF, JPG i PNG. Nauczysz się korzystać z kaskadowych arkuszy stylów oraz tworzyć tabele i ramki.

- Elementy dokumentu HTML
- Tytuł, słowa kluczowe i opis strony
- Wprowadzanie i formatowanie tekstu
- Adresy URL względne i bezwzględne
- Hiperłącza
- Wstawianie obrazów na strony
- Stosowanie kaskadowych arkuszy stylów
- Tabele
- Witryny oparte na ramkach

Zaznacz swoją obecność w sieci – stwórz własną stronę WWW

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl



SPIS TREŚCI

abc

6

PODSTAWY KASKADOWYCH ARKUSZY STYLÓW

Ogólne informacje na temat kaskadowych arkuszy stylów

Wprowadzenie przez konsorcjum W3C mechanizmu **kaskadowych arkuszy stylu** (ang. *Cascade Style Sheets — CSS*) zakończyło epokę niezgodnych ze sobą rozszerzeń języka HTML, wprowadzanych przez wszystkich liczących się producentów przeglądarek WWW, dając twórcom stron internetowych eleganckie i spójne narzędzie umożliwiające precyzyjne określenie wyglądu strony.

Cóż takiego specjalnego jest w kaskadowych arkuszach stylu? Przede wszystkim ich stosowanie jest niezwykle jednolite — prawie każdy element języka XHTML możesz rozbudować o atrybut `style`, a wewnątrz niego umieścić ściśle określony zestaw atrybutów stylu. Arkusze stylu pozwalają też ujednolicić wygląd serwisu internetowego oraz ograniczyć ilość danych pobieranych z internetu podczas oglądania stron WWW.

Jeśli przygotowujesz prezentacje internetowe i przewidujesz możliwość ich drukowania, zainteresować Cię może opcja przygotowania dwóch odrębnych arkuszy stylu dla tego samego dokumentu: jednego zoptymalizowanego pod kątem ekranu komputera, a drugiego — pod kątem drukarki.

Kaskadowe arkusze stylu możesz wykorzystać na trzy sposoby: stosując styl wpisany, styl osadzony lub zewnętrzne arkusze stylu.

Z kaskadowymi arkuszami stylów wiążą się dwa istotne pojęcia: dziedziczenie stylów i ich kaskadowy charakter.

Dlaczego arkusze stylów są kaskadowe?

Kaskadowy arkusz stylów to mechanizm definiujący formatowanie dla danej strony przy zastosowaniu stylów złożonych, które przeglądarka zinterpretuje w określonym porządku, zwanym kaskadą.

Prawie każdy element podrzędny (a więc na przykład element `span` zawarty wewnątrz elementu `p`) dziedziczy atrybuty stylu elementu nadrzędnego, a wybrane z nich może samodzielnie nadpisywać. W efekcie powstaje swoista kaskada definicji stylu, której wyższe stopnie narzucają wygląd stopniom niższym, a te z kolei mogą anulować wybrane elementy definicji i zastąpić je własnymi, które z kolei spróbują narzucić własnym elementom podrzędnym.

Na czym polega dziedziczenie stylu?

Dziedziczenie polega na tym, że elementy niższe w hierarchii drzewa dokumentu dziedziczą formatowanie elementów leżących wyżej w hierarchii, chyba że wyraźnie nadamy im inne formatowanie. Jeśli więc do tekstu tabeli zastosowano czcionkę pogrubioną o określonym kroju, to pojawi się ona we wszystkich komórkach tej tabeli, gdyż element `td` leży niżej w hierarchii drzewa dokumentu niż element `table`. Możemy jednak zdefiniować odrębne formatowanie dla wybranej komórki i w ten sposób uniknąć efektu dziedziczenia stylu. Ustawienia zdefiniowane bezpośrednio w wybranym elemencie mają bowiem pierwszeństwo nad dziedziczonymi. Ogólna zasada jest więc taka — jeśli wyraźnie nie zdefiniujemy formatowania dla jakiegoś elementu, dziedziczy on własności po swoich „przodkach", czyli elementach wyższych w hierarchii.

Są jednak odstępstwa od tej zasady. Na przykład zdefiniowanie stylu czcionki w sekcji `body` (a więc na szczycie hierarchii) nie wpływa na postać czcionki w komórkach tabeli, jeśli stronę oglądamy w przeglądarce Netscape Communicator. Z tego względu należy sprawdzać wygląd strony przynajmniej w dwóch przeglądarkach.

Co daje stosowanie arkuszy CSS?

Podstawowe zalety stylów CSS to możliwość szybkiej i prostej modyfikacji stylu oraz błyskawiczna wręcz aktualizacja postaci dokumentu w przypadku takich zmian. One naprawdę zaoszczędzą Twój czas! Wyobraź sobie, że w rozbudowanym serwisie internetowym trzeba zmienić sposób formatowania dziesiątków nagłówków czy odnośników! To wiele godzin pracy, jeśli będziesz ręcznie wyszukiwał elementy i zmieniał ich atrybuty, lub kilka chwil, jeśli zastosujesz w dokumencie arkusze CSS — wówczas modyfikacja stylu to parę stuknięć w klawisze, a aktualizacja odbywa się automatycznie.

Jakie są rodzaje arkuszy stylów?

Arkusze stylów dają wiele możliwości stosowania stylów. Definicja stylu może pojawić się w konkretnym elemencie XHTML — wówczas mówimy o *stylu wpisanym*; w obrębie elementu `head` strony HTML (to znaczy między znacznikami `<head>` `</head>`) — takie arkusze stylów nazywa się *osadzonymi*; lub może zostać pobrana z pliku zewnętrznego — jest to wtedy *zewnętrzny* lub *łączony arkusz stylów*.

Wszystkie typy arkuszy CSS — *wpisane*, *osadzone* i *zewnętrzne* — można stosować jednocześnie.

Łączone arkusze stylów są przechowywane w zewnętrznym pliku o rozszerzeniu nazwy *.css*. Składnia takiego arkusza jest podobna do składni arkusza osadzonego, a sformatowanie strony wymaga jedynie umieszczenia odnośnika do pliku zawierającego definicję stylu.

Arkusze stylów można także importować. Opcję tę obsługuje na razie tylko Internet Explorer (od wersji 4). Polecenie importu umieszcza się w obrębie elementu `head`:

```
<style>
<!--
@import url("adres URL arkusza");
-->
</style>
```

W wyrażeniu `@import url("adres URL arkusza")` można korzystać zarówno z adresu względnego, jak i bezwzględnego.

Importowane style ułatwiają zachowanie jednolitego wyglądu witryn. Importowany arkusz stylów można także wzbogacać o własne definicje stylów dla różnych elementów. W przypadku konfliktu pierwszeństwo będą miały ustalenia własne. Oto przykład takiej definicji:

```
<style>
<!--
@import url("http://www.strona.com/style/styl.css");
H1 {font-size: 30pt; font-family: Helvetica}
-->
</style>
```

W definicji możemy podać kilka kolejnych adresów importowanych arkuszy (polecenia `@import` muszą się znajdować na początku). W przypadku konfliktu definicji pierwszeństwo mają ostatnie w kolejności definicje `@import`.

Jaki styl zostanie zastosowany, jeśli w dokumencie zdefiniowano kilka arkuszy stylów (czyli jak działa kaskada stylów)?

Ponieważ źródła stylów mogą być różne — definicje można umieszczać w nagłówku strony, bezpośrednio w dokumencie, mogą one też pochodzić z zewnętrznych źródeł — konieczne jest więc ustalenie hierarchii ważności w przypadku konfliktu stylów. Może się bowiem zdarzyć, że zewnętrzny arkusz definiuje akapit za pomocą czcionki `Times 12pt`, w nagłówku strony akapit jest reprezentowany przez czcionkę `Helvetica 11pt`, a w samym dokumencie pojawia się akapit zdefiniowany za pomocą czcionki `Times 11pt`. Musi więc istnieć sposób rozwiązania takiego konfliktu.

Przyjęto, że oddziaływanie stylów z arkuszy zewnętrznych może być modyfikowane przez style zdefiniowane w nagłówku dokumentu, to zaś może być modyfikowane przez style zdefiniowane bezpośrednio w treści dokumentu. Pierwszeństwo mają zatem style zdefiniowane „bliżej" konkretnego elementu. Przeglądarka sprawdza więc najpierw, czy istnieją jakieś arkusze zewnętrzne, i stosownie do ich definicji formatuje stronę. Następnie sprawdza, jakie są definicje stylów w nagłówku strony, i modyfikuje wygląd zgodnie z ich ustaleniami. Następnie sprawdza style w samym dokumencie i ponownie modyfikuje fizyczną postać strony. Ten model działania pokazuje, jak działa kaskada stylów. Między stylami z różnych źródeł nie muszą zresztą wcale występować żadne konflikty — style mogą uzupełniać się, tworząc styl wypadkowy.

Ta zasada pozwala też wygodnie manipulować postacią całych kompleksów stron. Można na przykład ustalić pewne ogólne cechy całej witryny firmy i zawrzeć je w zewnętrznym arkuszu stylów. Następnie można zbudować odrębne arkusze dla wydziałów firmy i zawrzeć w nich bardziej precyzyjną informację. Po ustaleniu hierarchii arkuszy można łatwo definiować style dla dziesiątek i setek stron, a jedna drobna zmiana w arkuszu spowoduje zmiany we wszystkich objętych nimi dokumentach.

Jak z tego widać, kaskadowy charakter arkuszy stylów i zasada dziedziczenia stylu uzupełniają się, tworząc zwarty system ogólnych zasad sterujących działaniem stylów. Kaskada ustala hierarchię źródeł stylów, a dziedziczenie wpływa na wygląd strony z punktu widzenia hierarchii elementów w danym dokumencie.

Tak więc w przypadku obecności różnych arkuszy stylów na jednej stronie stopień ich ważności rośnie w następującej kolejności:

1. Domyślne ustawienia przeglądarki.
2. Zewnętrzny arkusz stylów.
3. Osadzony arkusz stylów (umieszczony między znacznikami `<head> </head>`).
4. Styl wpisany (dotyczący konkretnego elementu HTML).

Najwyższy priorytet ma styl wpisany, co oznacza, że jego ustawienia są dominujące względem zadeklarowanych w sekcji `head` oraz pobranych z pliku zawierającego zewnętrzny arkusz stylów. Ustawienia stylu wpisanego dominują także nad domyślnymi ustawieniami przeglądarki.

Jaka jest ogólna postać kaskadowego arkusza stylów?

Styl wpisany

Ten sposób wykorzystania kaskadowych arkuszy stylu jest Ci już doskonale znany. Polega on na rozbudowywaniu wybranego elementu języka HTML, którego wygląd chcesz zmienić, o atrybut `style`, którego wartością jest definicja stylu:

```
<znacznik style="właściwość: wartość;">
```

Na przykład:

```
<span style="color: blue; background-color: yellow;">Niebieski tekst na żółtym
tle</span>
```

Zaletą tej metody jest jej bezpośredniość — możesz zmienić wygląd dowolnego, najmniejszego choćby elementu strony całkowicie niezależnie od wyglądu pozostałych elementów. Z tej zalety wynika również największa wada tej metody — zmiana wyglądu większej liczby elementów wymaga wprowadzania olbrzymiej ilości kodu.

Stosowanie stylu wpisanego ma największy sens, gdy chcesz zmienić wygląd kilku wyrazów lub linijek tekstu. Zanim jednak zastosujesz styl wpisany, zastanów się, czy nie warto zastosować stylu osadzonego. W prezentowanych dotychczas przykładach pojawiał się styl wpisany, gdyż jest on najprostszą formą kaskadowych arkuszy stylu. Jednak w większości przypadków lepszym rozwiązaniem byłoby zastosowanie stylu osadzonego.

Styl osadzony

Styl osadzony jest najchętniej stosowanym elementem kaskadowych arkuszy stylu. Definicję stylu osadzonego tworzy się w całkowitym oderwaniu od rzeczywistego elementu — określamy rodzaj czcionki, kolor tekstu i szerokość marginesu, mając na myśli nie jakiś pojedynczy, konkretny element strony, a całą klasę elementów. Definicja stylu może określać wygląd wybranych elementów języka HTML lub też wyłącznie ich wydzielonych podklas.

Raz stworzona definicja stylu osadzonego może być wykorzystana później do zmiany wyglądu dowolnej liczby elementów strony WWW. Co więcej, jeśli nagle zapragniesz zmienić nieco tę definicję, automatycznie zmieni się wygląd wszystkich elementów, których wygląd ta definicja określa! Nie muszę mówić, jak wielkim jest to udogodnieniem.

Trudno się dziwić, że w zasadzie wygląd każdej większej strony WWW opisany jest właśnie za pomocą osadzonego arkusza stylu. Zmniejsza się w ten sposób rozmiar pliku HTML (definicja stylu umieszczona jest tyko w jednym miejscu kodu), zaś ewentualne zmiany wyglądu strony nie wymagają wprowadzania poprawek w dziesiątkach lub setkach miejsc kodu.

Ogólna postać osadzonego arkusza CSS jest następująca:

```
<style type="text/css">
<!--
selektor {właściwość:wartość [; właściwość:wartość] ...}
-->
</style>
```

Zawarta w obrębie elementu `style` definicja ma następującą składnię:

```
selektor{właściwość:wartość [; właściwość:wartość] ...}
```

Selektorem nazywa się znacznik lub element, który chcesz zdefiniować; *właściwość* to jego atrybut, który zmieniasz przypisując mu nową *wartość*. Właściwość i wartość rozdzielone są dwukropkiem oraz zawarte w nawiasach klamrowych:

```
body {color: black}
```

Jeśli wartość ma postać wielowyrazową — na przykład *sans serif* — umieszcza się ją w cudzysłowie:

```
p {font-family: "sans serif"}
```

Oto przykład rzeczywistej definicji:

```
<style type="text/css">
 <!--
  body        {margin:20px; color:black}
  h1          {color:blue; text-align:center}
  h2          {color:blue; text-align:left}
  p           {text-align:justify; text-indent:25px}
 -->
</style>
```

Między znacznikami <style> i </style> umieszczana jest lista elementów HTML wraz z właściwościami arkusza, które te elementy definiują. Jeśli właściwości jest kilka, wszystkie muszą być umieszczone w nawiasie klamrowym ({}) oraz muszą być oddzielone średnikami (;).

Atrybut type znacznika <style> ma wartość text/css. Jest to dla przeglądarki informacja, że następne instrukcje to arkusz stylów. Instrukcje te zawarte są w znaczniku komentarza, <!-- -->. Jest to zabezpieczenie na wypadek, gdyby strona trafiła do przeglądarki starego typu, która nie potrafi obsłużyć arkuszy stylów. Wówczas instrukcje formatowania zostaną zignorowane, a nie wyświetlone na ekranie jako tekst.

Zewnętrzny arkusz stylu

Zewnętrzny arkusz stylu nie różni się zasadniczo od osadzonego arkusza stylu; rozszerza tylko jego uniwersalność, przenosząc definicje stylów z wnętrza kodu jednej strony WWW do osobnego pliku, który może być wykorzystany w dziesiątkach, setkach lub nawet tysiącach stron składających się na serwis internetowy.

Zewnętrzny arkusz stylu jest tym dla serwisu internetowego, czym styl osadzony był dla pojedynczej strony — umożliwia scentralizowanie definicji stylu i uniknięcie konieczności wprowadzania poprawek w wielu plikach przy każdej najdrobniejszej zmianie Twojego projektu. Zewnętrzny arkusz stylu jeszcze bardziej ogranicza ilość danych, które czytelnik musi pobrać z Sieci — raz pobrany plik arkusza stylu pozostaje w pamięci podręcznej przeglądarki i jest dostępny natychmiast dla każdej kolejnej strony Twojego serwisu.

Czy jedną właściwość można przypisać kilku elementom?

W definicji stylu elementy (inaczej selektory) można *grupować*. Selektory w grupie oddzielane są przecinkami. Oto przykład definicji, w której wszystkim poziomom nagłówka (od h1 do h6) przypisany zostaje kolor czerwony:

```
h1, h2, h3, h4, h5, h6
{
color: red
}
```

Czy jednemu elementowi można przyporządkować różne style?

Klasę określa się w taki oto sposób:

```
element.nazwa_klasy{właściwość:wartość [; właściwość:wartość] ...}
```

lub

```
.nazwa_klasy{właściwość:wartość [; właściwość:wartość] ...}
```

W pierwszym przykładzie klasa jest powiązana z elementem danego typu, w drugim przypadku klasa jest niezależna od typu elementu.

Powiedzmy, że chcesz w swoim dokumencie zastosować do akapitów dwa różne sposoby wyrównania tekstu: pewne akapity chcesz dosunąć do prawego marginesu, a pozostałe wycentrować. W takim przypadku przydatny będzie atrybut `class`. Pozwala on zdefiniować różne style dla tego samego elementu — inaczej mówiąc, pozwala zdefiniować *klasy stylu*.

W naszym przykładzie musimy więc zdefiniować dwie klasy: pierwszą, nazwijmy ją `prawy`, w której ustawimy sposób wyrównania tekstu do prawego marginesu, i klasę `center`, w której tekst będzie wyśrodkowany:

```
p.prawy {text-align: right}
p.center {text-align: center}
```

Teraz wystarczy wstawić nazwę klasy do tych znaczników, w których chcemy mieć określony sposób formatowania, i gotowe:

```
<p class="prawy"> Ten akapit zostanie wyrównany do prawej. </p>
<p class="center"> Ten akapit zostanie wyśrodkowany.</p>
```

Nazwy klas mogą być dowolne, pamiętaj jednak, by *nie stosować w nich polskich liter*.

Jeśli pominiesz nazwę selektora w definicji klasy:

```
.nazwa_klasy{właściwość: wartość}
```

zdefiniowany zostanie styl, który można zastosować do wielu elementów. W przykładzie przedstawionym poniżej ta sama klasa `prawy` została zastosowana zarówno do elementu `h1`, jak i do akapitu `p`:

```
<h1 class="prawy">Ten nagłówek został wyrównany do prawego marginesu:</h1>
<p class="prawy">Ten akapit został wyrównany do prawego marginesu:</p>
```

Czy można narzucić styl pojedynczym wystąpieniom danego elementu?

Pojedynczym wystąpieniom danego typu można narzucić styl, korzystając z atrybutu `id`. Atrybut `id` pozwala oznaczać elementy, podobnie jak atrybut `class` — `id` jest czymś w rodzaju identyfikatora elementów.

Atrybut `id` można zdefiniować na dwa sposoby: tak, by styl stosowany był do dowolnego elementu o określonym `id`, lub tak, by styl był stosowany do określonego elementu o danym `id`.

Ogólna postać definicji jest następująca:

```
selektor#id {właściwość:wartość [; właściwość:wartość] ...}
```

lub

```
#id {właściwość:wartość [; właściwość:wartość] ...}
```

Nazwa atrybutu `id` musi być jednoznaczna. W dokumencie może istnieć *tylko jeden* element o danym `id`.

Oto przykłady dwóch definicji identyfikatora `id`. W pierwszym przypadku styl zostanie zastosowany do każdego elementu, który oznaczysz identyfikatorem `wstep`:

```
#wstep
{
font-size:110%;
font-weight:bold;
color:#0000ff;
background-color:transparent
}
```

A oto drugi przypadek. Tutaj styl będzie zastosowany tylko do elementu `p` o `id="wstep"`:

```
p#wstep
{
font-size:110%;
font-weight:bold;
color:#0000ff;
background-color:transparent
}
...
<p id="wstep">Do tego akapitu zostaną zastosowane ustawienia stylu zdefiniowane
w definicji id o nazwie wstep.</p>
```

Zwróć uwagę, że w obu przypadkach ustawienia stylu, które będą stosowane do elementów oznaczonych atrybutem `id`, identyfikuje się wartością atrybutu `id` poprzedzoną znakiem #.

Komentarze CSS

Definicje stylu często nie są proste, a nawet jeśli są, nie zawsze pamiętamy, dlaczego wybraliśmy takie, a nie inne właściwości. Dlatego stosuj komentarze. Będą bardzo przydatne przy przyszłej edycji arkusza stylów.

Jaką postać ma komentarz CSS?

Komentarz CSS rozpoczyna się od znaku prawego ukośnika, po nim następuje gwiazdka, potem tekst komentarza i wreszcie komentarz zostaje zamknięty gwiazdką i prawym ukośnikiem. Oto przykład:

```
/* To jest komentarz CSS */
p
{
text-align: right;
/* To też jest komentarz CSS */
color: green;
font-family: arial
}
```

Styl wpisany

Informacje ogólne na temat arkuszy stylów sprawdzimy teraz w praktyce. Zaczniemy od zdefiniowania stylu wpisanego.

Jak zmienić czcionkę, kolor tekstu i tła oraz marginesy tekstu, korzystając ze stylu wpisanego?

O tym, jak zmieniać czcionkę, którą złożony jest tekst, kolor tej czcionki, kolor tła tekstu oraz szerokość marginesów elementu języka HTML, dowiedziałeś się już w części tego kursu poświęconej formatowaniu akapitów i bloków tekstu. Tutaj omówię te zagadnienia jeszcze raz w wielkim skrócie, by przypomnieć Ci sposób stosowania stylu wpisanego.

Za zmianę kroju pisma, którym rysowany jest tekst zapisany wewnątrz elementu, odpowiada atrybut stylu `font-family`. W jego wnętrzu podaje się listę sugerowanych krojów pisma w kolejności od preferowanego aż po najbardziej ogólny. Jeżeli nazwa kroju zawiera znaki odstępu, należy zamknąć ją w pojedynczych znakach cudzysłowu:

```
font-family: 'Trebuchet MS',Arial,sans-serif;
```

Nazwy ogólnych krojów pisma to:

- *serif* — czcionka szeryfowa, np. Times New Roman,
- *sans-serif* — czcionka bezszeryfowa, np. Arial,
- *cursive* oraz *fantasy* — ozdobne czcionki pochyłe (rzadko używane, gdyż każda przeglądarka może podstawić w ich miejsce całkowicie inne kroje pisma — często w ogóle nieprzystające do zamysłu autora strony),
- *mono* — czcionka nieproporcjonalna (o stałej szerokości wszystkich znaków) używana najczęściej do zapisywania tekstu programów komputerowych, tworzenia prymitywnych tabel lub wyróżniania poleceń programów; np. Courier New.

Za rozmiar czcionki odpowiada atrybut stylu `font-size`. Rozmiar najczęściej podaje się w punktach drukarskich (`pt`; każdy punkt to 1/72 część cala):

```
font-size: 12pt;
```

Atrybut stylu `color` pozwala zdefiniować kolor wyświetlanego tekstu. Podana w nim wartość może być zestandaryzowaną nazwą barwy, opisem natężenia składowych RGB zapisanym szesnastkowo (`#RGB` lub `#RRGGBB`) lub opisem natężenia zapisanym w postaci `rgb(r,g,b)`:

```
color: blue;
color: #A05;
color: #0015A0;
color: rgb(100, 0, 255);
```

Atrybut stylu `background-color` określa kolor tła tekstu lub elementu. Oznaczenia koloru można podawać we wszystkich formach opisanych powyżej:

```
background-color: yellow;
```

Za pomocą atrybutu stylu `margin` możesz zdefiniować szerokość zewnętrznych marginesów akapitu tekstu lub elementu języka HTML. Wartość atrybutu stylu `margin` składa się z czterech elementów oddzielonych znakami spacji i podanych w następującej kolejności: szerokość górnego marginesu, szerokość prawego marginesu, szerokość dolnego marginesu i szerokość lewego marginesu:

```
margin: 1cm 1cm 1cm 1cm;
```

Jeżeli chciałbyś wyzerować szerokości marginesów (usunąć je całkiem), możesz też użyć skróconej postaci:

```
margin: 0;
```

Za wewnętrzny margines (zawierający w sobie kolor tła elementu) odpowiada z kolei atrybut stylu padding, którego zasady stosowania są identyczne:

```
padding: 5pt 1cm 10px 0;
padding: 0;
```

Kompletny kod znacznika otwierającego element p wraz z definicją stylu może wyglądać więc mniej więcej następująco:

```
<p style="font-family: 'Trebuchet MS',Arial,sans-serif; font-size: 12pt;
color: blue; background-color: yellow; margin: 0; padding: 5pt 5pt 5pt 5pt;">
```

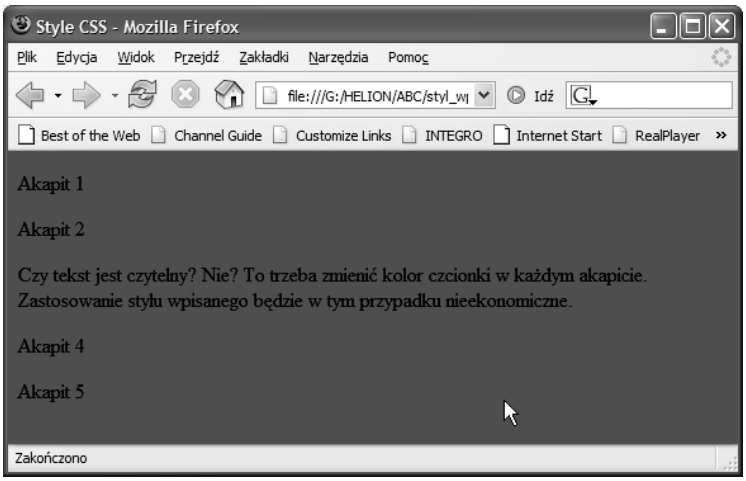
Wpisany arkusz stylów odnosi się tylko do tych akapitów, w znacznikach których umieszczona została definicja stylu. Pozostałe akapity zachowują domyślne formatowanie.

W ramach treningu zmieńmy kolor tła strony (elementu body) na niebieski, używając mechanizmu stylu wpisanego (listing 6.1, rysunek 6.1):

Listing 6.1. Styl wpisany

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
 <head>
  <title>Style CSS</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 </head>
 <body style="background-color: blue;">
  <p>Akapit 1</p>
  <p>Akapit 2</p>
  <p>Czy tekst jest czytelny? Nie? To trzeba zmienić kolor czcionki w każdym
  akapicie. Zastosowanie stylu wpisanego będzie w tym przypadku
  nieekonomiczne.</p>
  <p>Akapit 4</p>
  <p>Akapit 5</p>
 </body>
</html>
```

Ze stylu wpisanego korzystaj tylko tam, gdzie koniecznie zachodzi potrzeba zmiany wyglądu jednego tylko elementu strony.



Rysunek 6.1. Do określenia tła strony zastosowano styl wpisany. Na ciemnym tle tekst nie jest widoczny. Zmiana koloru tekstu we wszystkich akapitach byłaby bardzo czasochłonna — w tym przypadku nie ma sensu korzystać ze stylu wpisanego

Osadzony arkusz stylów

Tekst na rysunku 6.1 nie jest czytelny. Prezentowałby się o wiele lepiej, gdyby był zapisany kolorem białym lub żółtym. Gdybyś jednak chciał zmieniać kolor każdego akapitu, czekałoby Cię poprawianie deklaracji stylu w pięciu miejscach — a co począć z większymi stronami lub częstszymi zmianami kolorystyki?

Jak sformatować tekst i tło strony, korzystając z osadzonego arkusza stylów?

Deklaracja stylu osadzonego pozwala za jednym zamachem zmienić atrybuty stylu *wszystkich* elementów języka HTML danego typu. Wystarczy dodać do pliku (wewnątrz elementu `head`) następujący fragment:

```
<style type="text/css"><!--
 p {
  color: yellow;
 }
--></style>
```

a wszystkie akapity tekstu zmienią swój kolor na żółty dokładnie tak, jakbyś znacznik otwierający każdego z nich zapisał nie jako `<p>`, lecz jako `<p style="color: yellow;">`.

Osadzenie stylu elementu p na początku kodu strony umożliwia Ci teraz natychmiastową zmianę koloru tekstu na biały, turkusowy lub dowolny inny — możesz eksperymentować, za każdym razem zmieniając definicję koloru w jednym tylko miejscu!

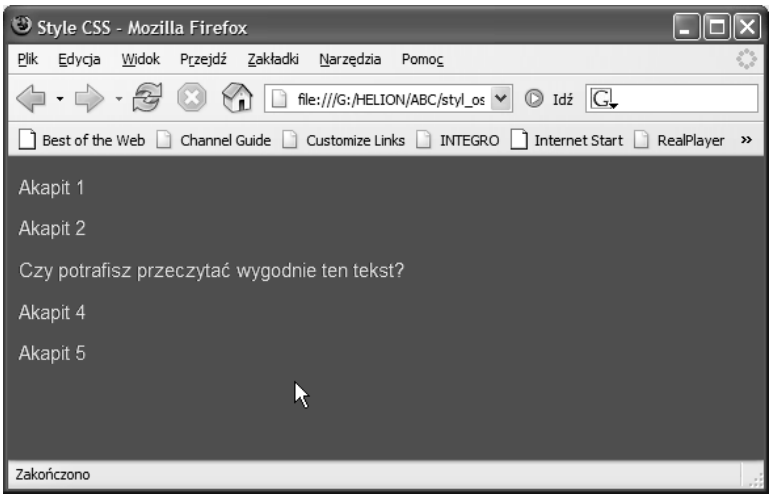
Rozszerzmy definicję stylu również na element body, dodatkowo zmieniając jeszcze dalej wygląd akapitów (listing 6.2, rysunek 6.2).

Listing 6.2. Styl osadzony

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
 <head>
  <title>Style CSS</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css"><!--
   body {
    background-color: blue;
   }
   p {
    color: yellow;
    font-family: Arial,sans-serif;
    font-size: 11pt;
   }
  --></style>
 </head>
 <body>
  <p>Akapit 1</p>
  <p>Akapit 2</p>
  <p>Czy potrafisz przeczytać wygodnie ten tekst?</p>
  <p>Akapit 4</p>
  <p>Akapit 5</p>
 </body>
</html>
```

Dzięki osadzonym definicjom stylu wszystko, co dotyczy wyglądu strony WWW, zgrupowane jest w jednym miejscu i może podlegać prostym zmianom, a właściwy kod HTML określa strukturę dokumentu, a nie jego wygląd.

Globalne formatowanie oznacza, że styl jest stosowany do wszystkich instancji elementu, dla którego został zdefiniowany. Są, oczywiście, wyjątki od reguły. Już o nich wspominaliśmy przy okazji omawiania klas i atrybutów id. W dalszej części rozdziału zobaczysz, jak w praktyce poradzić sobie z odejściem od zasad globalnego formatowania w przypadku pewnych wystąpień elementu.



Rysunek 6.2. Strona została sformatowana przy użyciu osadzonego arkusza stylów

Skąd się wzięła taka postać strony?

Jak już wspomniałam, osadzony arkusz stylów definiuje style dla całego dokumentu. Definicja stylu z listingu 6.2:

```
<style type="text/css"><!--
   body {
    background-color: blue;
   }
   p {
    color: yellow;
    font-family: Arial,sans-serif;
    font-size: 11pt;
   }
  --></style>
```

stanowi, że:

- Tło strony będzie koloru niebieskiego — decyduje o tym wartość właściwości `background-color: blue`. `blue` to nazwa koloru. Zamiast niej możesz użyć kodu koloru, `#0000FF`. Kody kolorów znajdziesz w Dodatku A.
- Tekst dokumentu (akapitów) będzie koloru żółtego. Określa to para: `color: yellow`.
- Zastosowana zostanie czcionka szeryfowa, najlepiej taka jak Arial, (`font-family: Arial,sans-serif`) o rozmiarze 11 punktów (`font-size: 11pt`).

Jak widzisz, pomimo tajemniczego wyglądu, osadzony arkusz stylów nie jest wcale tak tajemniczy.

W jaki sposób zastosować inny styl do wybranej instancji elementu?

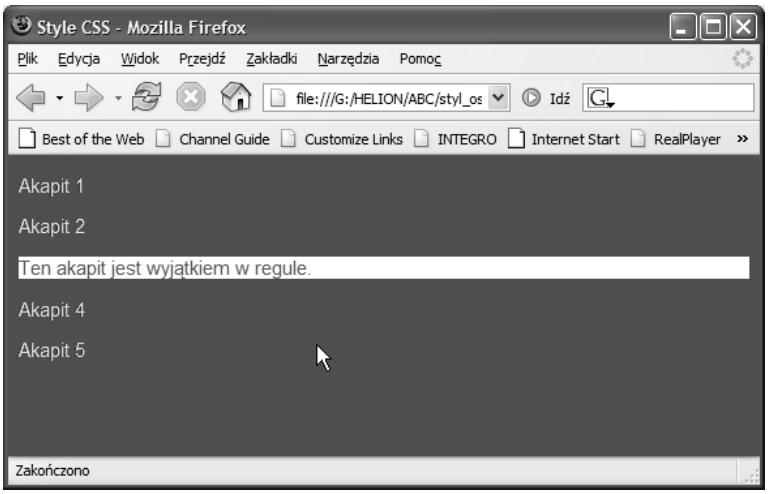
Powiedzmy, że chcesz, aby jeden z akapitów tekstu miał zupełnie inne formatowanie niż pozostałe. Okaże się to prostym zadaniem, jeśli skorzystamy z atrybutu `id` (patrz informacje wstępne do tego rozdziału). Wartość atrybutu może składać się z liter od a do z, cyfr, łączników i kropek, przy czym na początku musi znajdować się litera.

Na listingu 6.3 przedstawiony został kod źródłowy dokumentu z listingu 6.2, w którym jednemu z akapitów zmieniono styl. Akapitowi przypisano identyfikator `id="wyjatek"`, a styl `#wyjatek` dołączono do listy definicji stylów w elemencie `style`. Zgodnie z definicją, tekst pojawi się na białym tle i będzie koloru czerwonego. Pozostałe ustawienia stylu akapitu zostaną zachowane.

Listing 6.3. Osadzony arkusz stylów, w którym wykorzystano atrybut id

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
 <head>
  <title>Style CSS</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css"><!--
   body {
    background-color: blue;
   }
   p {
    color: yellow;
    font-family: Arial,sans-serif;
    font-size: 11pt;
   }
  #wyjatek {
    color: #FF0000; background-color: #FFFFFF;
   }
  --></style>
 </head>
 <body>
  <p>Akapit 1</p>
  <p>Akapit 2</p>
  <p id="wyjatek">Ten akapit jest wyjątkiem w regule.</p>
  <p>Akapit 4</p>
  <p>Akapit 5</p>
 </body>
</html>
```

Wygląd strony pokazany został na rysunku 6.3.



Rysunek 6.3. Osadzony arkusz stylów określa style na całej stronie, lecz można spod jego działania wyłączyć określone instancje elementu

Atrybut `id` stosuje się jednak przede wszystkim w skryptach — do wskazywania z poziomu skryptu jednej konkretnej instancji obiektu znajdującej się na stronie, na przykład w celu zmiany stylu odpowiedniego pojedynczego elementu strony z poziomu skryptu.

Jak zastosować w dokumencie klasy?

Klasy także pozwalają na indywidualizację ustawień stylu — aby skorzystać z tej możliwości, musisz po prostu zdefiniować odpowiednią klasę i oznaczyć za pomocą atrybutu `class` konkretny element.

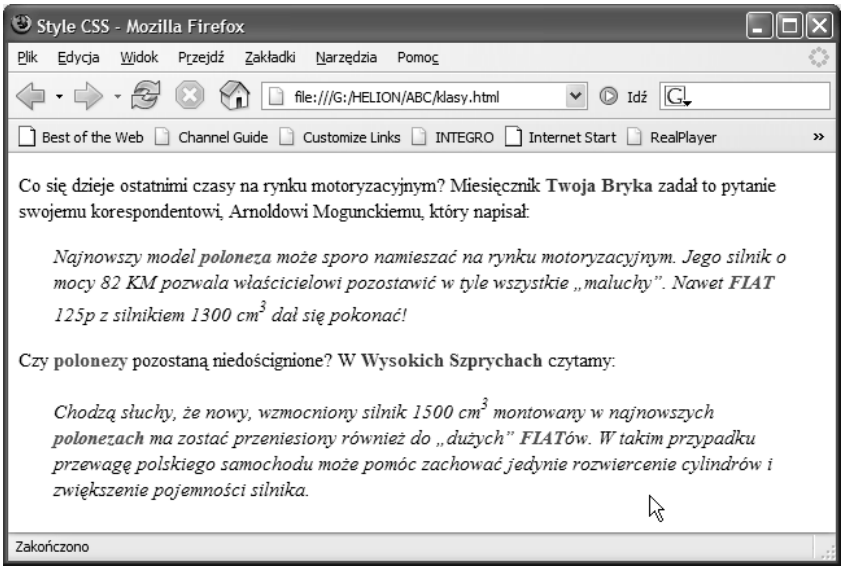
Nazwa klasy stylu powinna składać się ze znaków alfabetu łacińskiego oraz cyfr i nie powinna zaczynać się od cyfry. Stosowanie polskich znaków diakrytycznych jest zabronione.

Załóżmy, że na swojej stronie — poświęconej, powiedzmy, motoryzacji — chcesz cytować fragmenty obcych tekstów, każdy z fragmentów wyróżniając kursywą i szerszymi marginesami. Dodatkowo inną czcionką warto wyróżnić nazwy tytułów czasopism oraz marek samochodów. Aby za każdym razem nie wprowadzać wpisanej definicji stylu odróżniającej elementy `p` stanowiące cytat od elementów `p` tworzących Twój własny tekst, należy dokonać podziału: normalne elementy `p` będą tworzyły akapity własnego tekstu, a elementy `p` klasy `cytat` — cytaty. Z kolei elementy `span` klasy `czasopismo` będą służyły do zapisywania tytułów, a klasy `marka` — marek samochodów. Oto przykład (listing 6.4):

Listing 6.4. Zastosowanie klas stylów w dokumencie

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
 <head>
  <title>Style CSS</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css"><!--
   p {
    font-size: 11pt;
    line-height: 135%;
   }
   p.cytat {
    font-style: italic;
    margin: 0 20pt 0 20pt;
   }
   span.tytul {
    font-weight: bold;
    font-style: normal;
    color: brown;
   }
   span.marka {
    color: blue;
    font-weight: bold;
   }
  --></style>
 </head>
 <body>
  <p>Co się dzieje ostatnimi czasy na rynku motoryzacyjnym?
  Miesięcznik <span class="tytul">Twoja Bryka</span> zadał
  to pytanie swojemu korespondentowi, Arnoldowi Mogunckiemu,
  który napisał:</p>
  <p class="cytat">Najnowszy model <span class="marka">poloneza</span>
  może sporo namieszać na rynku motoryzacyjnym. Jego silnik
  o mocy 82 KM pozwala właścicielowi pozostawić w tyle
  wszystkie „maluchy". Nawet <span class="marka">FIAT</span> 125p
  z silnikiem 1300 cm<sup>3</sup> dał się pokonać!</p>
  <p>Czy <span class="marka">polonezy</span> pozostaną
  niedoścignione? W <span class="tytul">Wysokich Szprychach</span>
  czytamy:</p>
  <p class="cytat">Chodzą słuchy, że nowy, wzmocniony silnik
  1500 cm<sup>3</sup> montowany w najnowszych
  <span class="marka">polonezach</span> ma zostać przeniesiony
  również do „dużych" <span class="marka">FIAT</span>ów.
  W takim przypadku przewagę polskiego samochodu może pomóc zachować
  jedynie rozwiercenie cylindrów i zwiększenie pojemności silnika.</p>
 </body>
</html>
```

Efekty możesz podziwiać na rysunku 6.4.



Rysunek 6.4. Dzięki klasom te same zasady składu zostały zastosowane wielokrotnie wobec różnych elementów tego samego typu

Definicja klasy stylu elementu wyróżnia się dodaniem do nazwy elementu dowolnie wybranej nazwy klasy, oddzielonej kropką. Na przykład, aby założyć klasę `cytat` elementu `p`, utworzyliśmy wewnątrz elementu `style` następującą definicję:

```
p.cytat {
 .........
}
```

Aby skorzystać z utworzonej klasy, dodaliśmy do elementu atrybut `class` i podaliśmy w cudzysłowie nazwę założonej wcześniej klasy, na przykład:

```
<p class="cytat">.........</p>
```

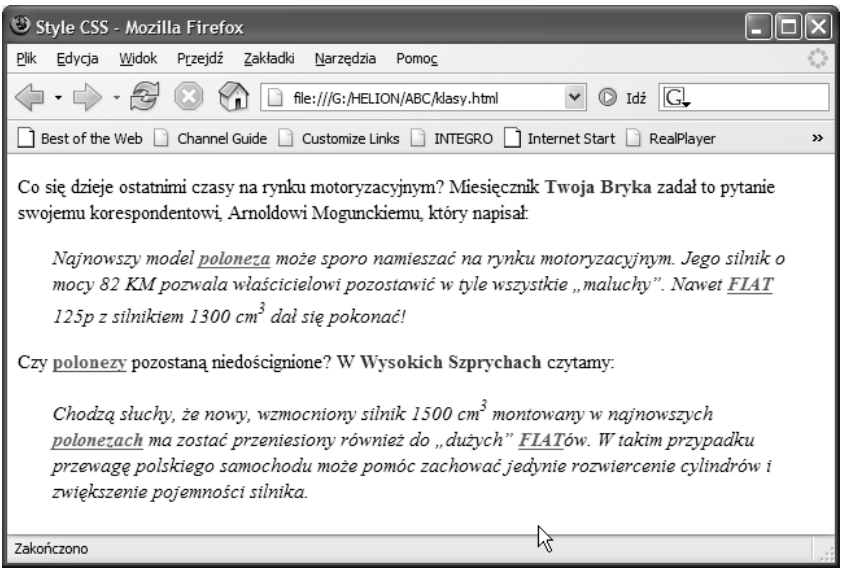
Wszystkie klasy dziedziczą atrybuty stylu elementu podstawowego. Widać to w powyższym przykładzie: mimo iż dla klasy `cytat` elementu `p` nie zadeklarowaliśmy rozmiaru czcionki, fakt, że dla samego elementu `p` została podana deklaracja `font-size: 11pt;`, wystarczył, by również cytaty były zapisane czcionką o rozmiarze `11` punktów.

Generalnie, dla podstawowej klasy elementu (czyli dla samego elementu) powinieneś nadawać wszystkie cechy wspólne klas pochodnych, by nigdy nie dopuszczać do duplikowania wpisów. Klasy powinny zawierać tylko deklaracje odróżniające je od klasy podstawowej.

Jakie są zalety tworzenia klas? Pomyśl sobie, że nie użyłeś ich, stosując w powyższym przykładzie za każdym razem styl wpisany modyfikujący wygląd tytułów i marek. Gdybyś teraz chciał zmienić wygląd marek samochodów, by były podkreślone i wypisane zieloną czcionką, czekałoby Cię poprawianie deklaracji w kilkunastu, kilkudziesięciu lub kilkuset miejscach! Używając klas, wystarczy wprowadzić poprawkę w deklaracji klasy stylu:

```
span.marka {
 color: green;
 font-weight: bold;
 text-decoration: underline;
}
```

a wszystkie miejsca, w których otoczyłeś tekst elementami `span` klasy `marka`, natychmiast po odświeżeniu strony zmienią swój wygląd (rysunek 6.5).



Rysunek 6.5. Centralizacja definicji klas stylu elementów pozwala szybko zmieniać wygląd strony, dostosowując go do nowych wymagań

Zauważ też, że tworzenie klas stylów stanowi kolejny krok w kierunku rozdzielenia logicznej struktury dokumentu (strony WWW) od jego wyglądu. Nawet gdybyś kiedyś stwierdził, że nazwy marek samochodów nie wymagają specjalnego wyróżnienia na ekranie, pozostaną wyróżnione w kodzie i będziesz miał potencjalną możliwość zmiany ich wyglądu. Poza tym tworząc klasy elementów, unikasz umieszczania w kodzie strony (i w zapisywanym tekście) jakichkolwiek informacji o wyglądzie.

Przy okazji tego przykładu poznałeś kilka nowych atrybutów stylu elementów. Pierwszym z nich jest `line-height`, w którym podaje się odstęp międzywierszowy tekstu. Możesz podawać go w dowolnych jednostkach, jednak najczęściej używa się skali względnej (`%`), zmieniającej odstępy równomiernie w zależności od rozmiaru czcionki. Pamiętaj jednak, że mała czcionka wymaga szerokich odstępów między wierszami, podczas gdy duża pozostaje czytelna, nawet gdy kolejne wiersze prawie stykają się ze sobą.

Drugi z atrybutów to `font-style`. Wartość `normal` oznacza zwykły tekst, a `italic` — kursywę. Atrybut ten zastępuje element `i` — a więc możesz z niego w ogóle zrezygnować, jeżeli cały tekst pojawiający się w konkretnych elementach ma być zapisany kursywą.

Trzeci to `font-weight`, który może przybierać wiele wartości, jednak dwie najczęściej używane (i często jedyne dające widoczne na ekranie efekty) to `normal` i `bold`. Wartość `normal` oznacza tekst normalnie zapisany, zaś `bold` — pogrubiony. Jak w przypadku kursywy, atrybut `bold` zastępuje element `b`, umożliwiając automatyczne pogrubianie całych fragmentów tekstu bez używania znaczników `<b>` i `</b>`.

Ostatnim jest atrybut `text-decoration`. Może on przyjmować następujące wartości:

- `none` — brak dekoracji tekstu,
- `underline` — podkreślenie,
- `overline` — nakreślenie,
- `line-through` — przekreślenie.

Tego atrybutu używaj bardzo rozważnie. Podkreślony tekst może kojarzyć się z odnośnikami, zaś przekreślenia czynią go nieczytelnym. Najlepiej w ogóle zrezygnować z tego typu dekoracji, rezerwując ją dla odnośników. Podany powyżej przykład zamiany stylu tekstu reprezentującego markę samochodu jest właśnie przykładem, jak *nie* używać podkreślenia.

Jak zdefiniować niezależne klasy stylu?

Klasy stylu związane z elementami języka HTML pozwalają określić wygląd podzbioru elementów danego typu. Czasem jednak przydaje się możliwość określenia wyglądu szerszej grupy elementów, niekoniecznie jednolitych pod względem typu. W takich przypadkach mogą się okazać pomocne niezależne klasy stylu.

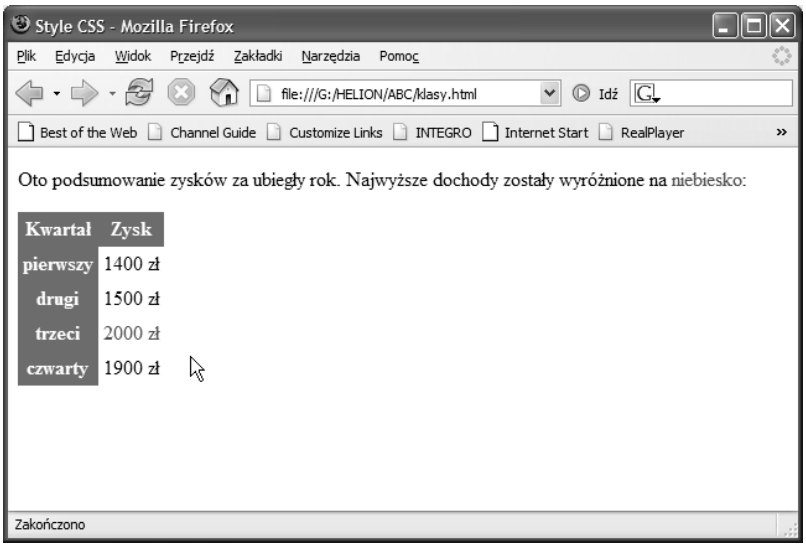
Na przykład, poniższy kod strony WWW definiuje niezależną klasę o nazwie *niebieski*, nadającą dowolnemu elementowi niebieski kolor.

W deklaracji niezależnej klasy stylu nie pojawia się nazwa elementu języka HTML, a od razu tylko kropka i nazwa klasy (listing 6.5, rysunek 6.6):

Listing 6.5. Niezależne klasy stylu

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
 <head>
  <title>Style CSS</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <style type="text/css"><!--
   th {
    background-color: gray;
    color: white;
   }
   .niebieski {
    color: blue;
   }
  --></style>
 </head>
 <body>
  <p>Oto podsumowanie zysków za ubiegły rok. Najwyższe dochody
  zostały wyróżnione na <span class="niebieski">niebiesko</span>:</p>
  <table cellpadding="4" cellspacing="0">
   <tr><th>Kwartał</th><th>Zysk</th></tr>
   <tr><th>pierwszy</th><td>1400 zł</td></tr>
   <tr><th>drugi</th><td>1500 zł</td></tr>
   <tr><th>trzeci</th><td class="niebieski">2000 zł</td></tr>
   <tr><th>czwarty</th><td>1900 zł</td></tr>
  </table>
 </body>
</html>
```

Mimo wygody, unikaj nadużywania uniwersalnych, niezwiązanych z żadnym konkretnym elementem języka HTML klas stylów. Tak naprawdę przydają się one niezwykle rzadko, a w większości przypadków wygodniejsze jest stworzenie kilku specjalizowanych klas stylów — po jednej dla każdego typu elementu — niż dostosowywanie różnych elementów do jednej, niezależnej klasy.



Rysunek 6.6. Za pomocą niezależnej klasy stylów można kształtować wygląd elementów różnego typu