ABC kaskadowych arkuszy stylów (CSS)

Autor: Bartosz Danowski
ISBN: 83-7197-747-6
Format: B5, stron: 300

Dynamiczny rozwój technologii przesyłu danych prowadzący do zwiększenia przepustowości internetu był kołem zamachowym powstawania nowych stron WWW opartych na coraz to wymyślniejszych rozwiązaniach. Jednym z nich są kaskadowe arkusze stylów (CSS), które obecnie są doskonałą alternatywą dla żmudnego procesu przygotowania stron WWW. Pozwalają nadać wspólny wygląd dokumentom tworzącym stronę WWW bez większego wysiłku i przy ogromnej oszczędności czasu.

Autor w książce „ABC kaskadowych arkuszy stylów (CSS)" stworzył możliwie kompletne, polskie opracowanie poświęcone roli CSS w tworzeniu stron WWW. Postawił sobie jeszcze jeden cel, zgodnie z którym książka miała być przeznaczona zarówno dla Czytelników rozpoczynających przygodę z tworzeniem stron WWW, jak i dla „starych wyjadaczy", mających w tej dziedzinie sporo doświadczenia. Zgodnie z sugestiami czytelników, którzy zapoznali się już z poprzednimi książkami Bartosza Danowskiego (np. „HTML. Ćwiczenia praktyczne", „Kaskadowe arkusze stylów. Ćwiczenia praktyczne"), niniejsza publikacja zawiera jeszcze więcej praktycznych przykładów.

Ze względu na charakter tematyki założono, że Czytelnik posiada już niezbędną wiedzę do pracy z komputerem. Dokładniej mówiąc, chodzi o umiejętność pracy z dowolnym edytorem tekstowym.

Doskonały podręcznik dla początkujących twórców stron WWW!

Książka opisuje m.in.:

- formatowanie czcionek na stronie WWW;
- osadzanie własnych czcionek;
- pozycjonowanie elementów;
- regulowanie marginesów;
- zmiana wyglądu okna przeglądarki i kursora myszki;
- formatowanie tabel;
- kontrola wyglądu bloków tekstu i nagłówków;
- kontrola wyglądu formularzy;
- tworzenie stron zgodnie z zaleceniami specyfikacji HTML 4.0.1.

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

Spis treści

Rozdział 3.

Dziedziczenie i kaskadowość

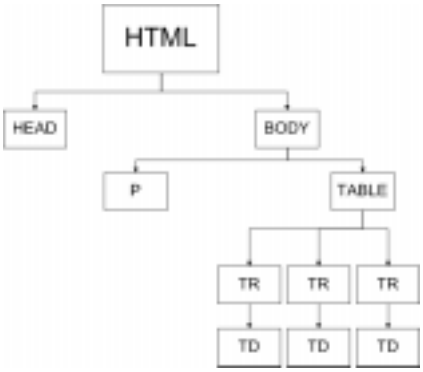
Dziedziczenie

Zrozumienie pojęcia dziedziczenia wymaga od nas zapoznania się z hierarchią ważności poszczególnych znaczników wewnątrz dokumentu. Kaskadowe arkusze stylów wprowadzają pojęcie *drzewa* i na jego przykładzie doskonale widać te zależności. Poniżej zamieszczam listing prostego kodu strony oraz rozrysowane dla niego drzewo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
  <HTML>
    <HEAD>
    <TITLE>Przykład drzewa</TITLE>
    </HEAD>
       <BODY>
         <P>Przykład tekstu.</P>
         <TABLE>
           <TR><TD></TD></TR>
           <TR><TD></TD></TR>
           <TR><TD></TD></TR>
         </TABLE>
    </BODY>
</HTML>
```

Przykładowe drzewo dla powyższego kodu wyglądałoby tak jak na rysunku 3.1.

Analizując rysunek, widzimy, że określenie np. koloru tekstu jako niebieski bezpośrednio dla pary znaczników <BODY></BODY> spowoduje jego użycie dla każdego elementu naszej strony znajdującego się niżej w hierarchii. Jeżeli dodatkowo zdefiniujemy kolor listy <UL></UL> jako zielony, to tekst całego dokumentu będzie niebieski poza listą, która zostanie sformatowana za pomocą koloru zielonego. Idąc dalej tym tokiem rozumowania, dopiszmy do stylu odpowiedzialnego za formatowanie <BODY> </BODY> definicję wielkości czcionki równą 14 punktów. Strona wynikowa będzie



Rysunek 3.1. *Przykładowe drzewo dziedziczenia zbudowane na podstawie opisanego kodu HTML*

sformatowana za pomocą czcionki o wielkości 14 punktów i w kolorze niebieskim. Natomiast lista będzie miała kolor zielony i identyczną wielkość czcionki, podobnie jak reszta dokumentu. Innymi słowy, styl odpowiedzialny za definicję czcionki będzie dziedziczony z nadrzędnego w hierarchii znaczników <BODY></BODY>.

Innym prostszym przykładem dziedziczenia może być deklaracja stylu dla odnośników, która opiera się na pseudoklasach. Mam tutaj na myśli przykład z definicją dwóch rodzajów odsyłaczy.

```
A:link
      {
      color: navy;
      font-size: 18pt;
      text-decoration: none;
      }
A.maly:link
      {
      font-size: 12pt;
      }
```

Zwróć uwagę na to, że pierwszy selektor określa rodzaj domyślnego odnośnika, któremu przypisuje kolor granatowy, wysokość 18 punktów oraz brak podkreślenia. Drugi selektor jest definicją klasy o nazwie `.maly` i odpowiada za utworzenie hiperłącza o takich samych właściwościach jak główny odsyłacz, a jedyna różnica ma polegać na wysokości tekstu, która tym razem wynosi zaledwie 12 punktów. Dlatego też wystarczy w przypadku mniejszego odsyłacza zadeklarować jedynie rozmiar czcionki, gdyż pozostałe elementy są dziedziczone z wyżej stojącego w hierarchii odsyłacza podstawowego.

Nic nie stoi na przeszkodzie, by poza wielkością zmienić również kolor mniejszego odsyłacza. W takiej sytuacji odpowiedni kod powinien przybrać następującą formę:

```
A:link
      {
      color: navy;
      font-size: 18pt;
      text-decoration: none;
      }
```

```
A.maly:link
      {
      font-size: 12pt;
      color: red;
      }
```

Duży, domyślny odsyłacz, będzie granatowy, natomiast małe hiperłącze określone za pomocą klasy powinno przybrać kolor czerwony.

Niestety, od zasady dziedziczenia zdarzają się wyjątki wynikające z niedoskonałości implementacji *CSS* przez producentów przeglądarek. Przykładem może być dokument, gdzie w znacznikach <BODY></BODY> zdefiniujemy kolor tekstu, a następnie wstawimy tabelę — przeglądarka Netscape Navigator nie będzie umiała zastosować naszej reguły dla zawartości komórek tabeli. Oczywiście w Mozilli, na której bazuję w niniejszej książce, wspomniany problem z dziedziczeniem nie występuje. Również Opera w miarę poprawnie dziedziczy style z elementów nadrzędnych.

Idealnym testem na sprawdzenie, czy nasza przeglądarka poprawnie dziedziczy style przypisane elementom nadrzędnym, są poniższy arkusz *CSS*:

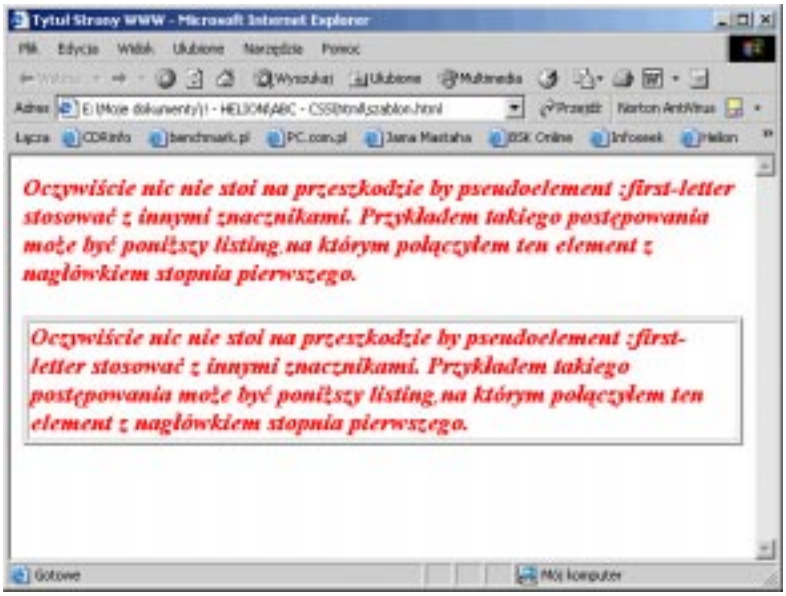
```
BODY
      {
      background-color: white;
      font-weight: bold;
      font-size: 15pt;
      font-style: italic;
      color: red;
      }
```

oraz kod HTML.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/
➥REC-html40/strict.dtd">
  <html>
    <head>
      <meta http-equiv="Content-type" content="text/html;charset=ISO-8859-2">
      <meta name="Keywords" content="słowa, kluczowe, dla, strony www">
      <meta name="Description" content="Krótki opis strony WWW">
      <meta name="Author" content="Bartosz Danowski">
      <meta name="Copyright" content="Wydawnictwo HELION">
      <LINK href="arkusz.css" rel="stylesheet" type="text/CSS">
      <title>Tytuł Strony WWW</title>

    </head>
    <body>
         Oczywiście nic nie stoi na przeszkodzie, by pseudoelement :first-letter
         stosować z innymi znacznikami. Przykładem takiego postępowania może być
         poniższy listing, na którym połączyłem ten element z nagłówkiem stopnia
         pierwszego.

      <table border=2>
         <tr>
              <td>
                  Oczywiście nic nie stoi na przeszkodzie, by pseudoelement
                  :first-letter stosować z innymi znacznikami. Przykładem takiego
                  postępowania może być poniższy listing, na którym połączyłem ten
                  element z nagłówkiem stopnia pierwszego.
```

```
                </td>
            </tr>
        </table>

    </body>
</html>
```

Porównując rysunki 3.2 oraz 3.3 widzimy, że w przypadku Internet Explorera tekst wewnątrz tabeli niczym nie różni się do tego na zewnątrz, czyli ma wysokość 15 punktów, kolor czerwony oraz został pogrubiony i pochylony. Natomiast na rysunku 3.3 od razu widać, że zawartość tabeli jest wyświetlona za pomocą domyślnej czcionki przeglądarki.



Rysunek 3.2. *MS Internet Explorer — przykład poprawnego dziedziczenia. Tekst w tabeli jest dokładnie taki sam jak na zewnątrz*

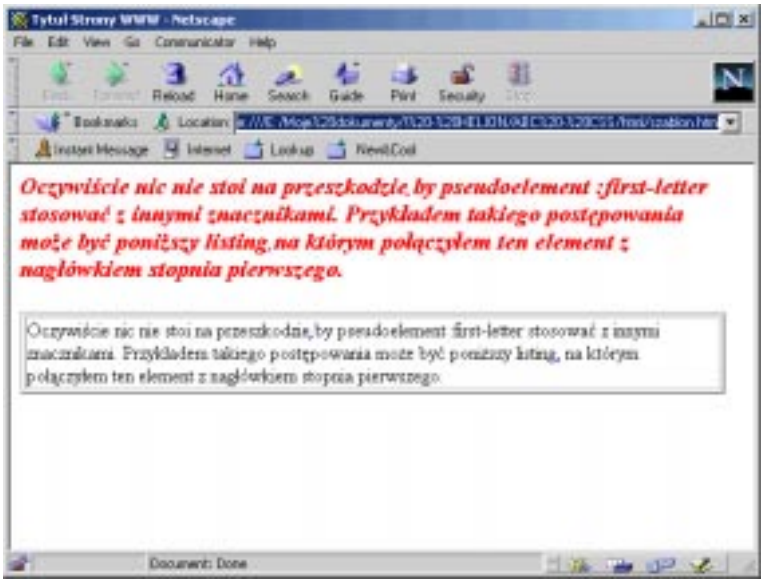
Rozwiązaniem tego problemu jest odpowiednie przygotowanie arkusza stylów (patrz rysunek 3.4).

```
BODY, TD
      {
      background-color: white;
      font-weight: bold;
      font-size: 15pt;
      font-style: italic;
      color: red;
      }
```

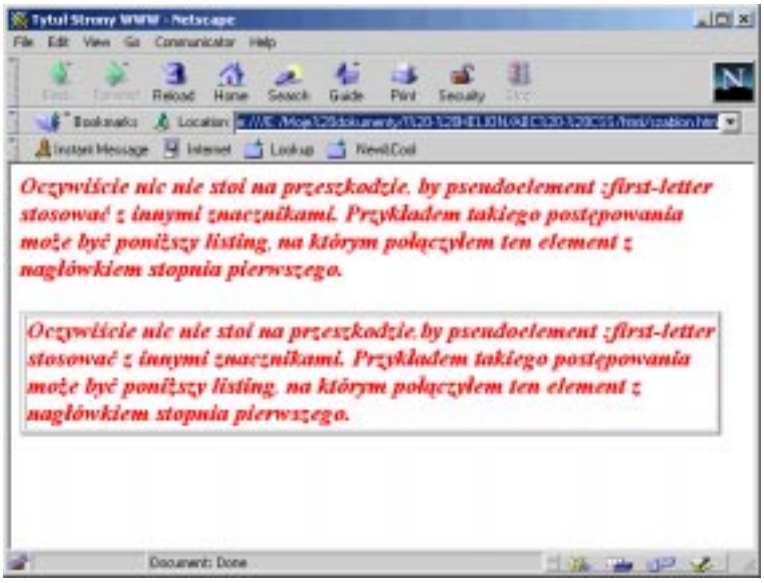
Ciekawym przykładem dziedziczenia jest niżej przedstawiony przypadek.

Zawartość arkusza stylów:

```
.czerwony
       {
font-size: 15pt;
```



Rysunek 3.3. *Netscape Navigator — przykład błędnego dziedziczenia. Tekst w tabeli jest inny niż na zewnątrz*



Rysunek 3.4. *Netscape Navigator po odpowiednich zmianach w kodzie arkusza CSS potrafi wyświetlić stronę zgodnie z naszymi oczekiwaniami — jednak nadal nie dziedziczy stylów nadrzędnych elementów*

```
color: red;
}
H1
      {
font-size: 30pt;
color: zielony;
}
```

Przykładowy kod HTML:

```
<h1 class="czerwony">Nagówek stopnia pierwszego</h1>
```

Analizując definicję stylów dochodzimy do wniosku, że nagłówek stopnia pierwszego powinien mieć wysokość tekstu równą 30 punktów oraz kolor zielony. Jeżeli przyjrzymy się samemu nagłówkowi w kodzie stronie, to widzimy, że nasz nagłówek ma przypisaną klasę o nazwie `.czerwony`.

O zgrozo, co tu teraz zadziała?... Otóż nie jest to takie skomplikowane, gdyż specyfikacja *CSS* określa *specyficzność* poszczególnych elementów. W moim przykładzie specyficzność kształtuje się następująco:

- `H1` — niższa specyficzność równa 1,
- `.czerwony` — wyższa specyficzność równa 10,

Zgodnie z założeniami zawartymi w specyfikacji realizowana jest reguła z większy numerem. Dlatego w tym konkretnym przypadku nagłówek zostanie wyświetlony czcionką o wysokości 15 punktów w kolorze czerwonym gdyż ta klasa ma wyższą specyficzność.

Czasami chcemy wyłączyć dziedziczenie pewnych stylów w czasie formatowania. W takim przypadku należy skorzystać z *ważności* danych stylów. Również tym razem oprę się na przykładzie z nagłówkiem. Chciałbym, aby mój nagłówek na stronie został wyświetlony czcionką o wysokości 15 punktów w kolorze zielonym. Niestety, moja zachcianka to połączenie wybranych właściwości klasy `.czerwony` oraz selektora `H1`. Najprostszym rozwiązaniem jest przygotowanie odpowiedniej klasy i przypisanie jej dla nagłówka `H1`. Niestety, tym razem nie możemy dodawać nowych definicji do arkusza, a jedynie dokonać kosmetycznej poprawki. Jak już wiesz, w poprzednim przykładzie nagłówek został sformatowany za pomocą klasy `.czerwony`, gdyż miała ona wyższą specyficzność, a styl dla selektora `H1` został zupełnie pominięty. Teraz skorzystamy z zaistniałej sytuacji i dla selektora `H1`, a dokładniej dla koloru zielonego, dodamy polecenie `!important`. Zmodyfikowany arkusz jest widoczny poniżej.

```
.czerwony
      {
      font-size: 15pt;
      color: red
      }
H1
      {
      font-size: 30pt;
      color: green !important;
      }
```

Zwróć uwagę na sposób, w jaki zostało dodane polecenie `!important` do konstrukcji stylu. Zawsze występuje po wartości przypisanej dla właściwości konkretnego selektora. Oczywiście sam kod HTML nie uległ żadnej zmianie i nadal ma następującą postać:

```
<h1 class="czerwony">Nagówek stopnia pierwszego</h1>
```

Przypisywanie ważności umożliwia zablokowanie dziedziczenia pewnych stylów z nadrzędnych elementów, dzięki czemu mamy jeszcze większe możliwości kontrolowania wyglądu strony WWW.

Polecenie !important jest poprawnie obsługiwane przez przeglądarki, na których oparłem się w niniejszej książce. Niestety, w przypadku Netscape Navigatora nie działa ono poprawnie i dany styl jest dziedziczony z elementu znajdującego się wyżej w hierarchii.

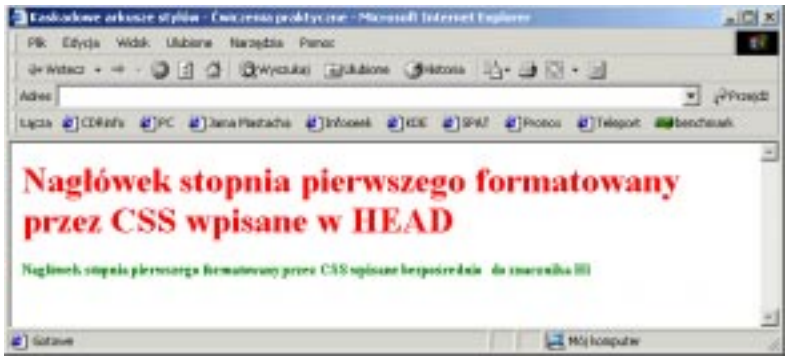
Kaskadowość

Kolejnym bardzo ważnym pojęciem stosowanym w kaskadowych arkuszach stylów, a przy tym występującym w samej nazwie, jest *kaskadowość*. Funkcja ta odpowiada za określenie hierarchii stosowanych stylów w dokumencie. Wiemy już, że style do dokumentu możemy wstawiać na kilka sposobów (bezpośrednio w kodzie strony jako atrybut dowolnego znacznika, w nagłówku <HEAD></HEAD>, globalnie dla danego dokumentu oraz przez dołączenie zewnętrznego arkusza). Mieszanie zastosowanych stylów jest jak najbardziej możliwe i często spotykane, dlatego konieczne stało się określenie ważności poszczególnych metod. Zasada kaskadowości przyjęta przez twórców wygląda następująco: najpierw ładowane i uwzględniane są zewnętrzne arkusze, następnie style wpisane do nagłówka <HEAD></HEAD>, a na samym końcu style wpisane bezpośrednio do znacznika. Takie rozwiązanie umożliwia pełną kontrolę nad dokumentem, a w przypadku sprzeczności zdefiniowanych stylów użyty zostanie ten, który jest najbliżej formatowanego dokumentu.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/
➥REC-html40/strict.dtd">
  <html>
    <head>
      <meta http-equiv="Content-type" content="text/html;charset=ISO-8859-2">
      <meta name="Keywords" content="słowa, kluczowe, dla, strony www">
      <meta name="Description" content="Krótki opis strony WWW">
      <meta name="Author" content="Bartosz Danowski">
      <meta name="Copyright" content="Wydawnictwo HELION">
      <title>Tytuł Strony WWW</title>
        <style type="text/CSS">
              H1
              {
              color: red;
              }
        </style>

    </head>
    <body>
<h1>Nagłówek stopnia pierwszego formatowany przez arkusz CSS wpisany w HEAD</h1>
    <h1 style="color: green; font-size: 10pt">Nagłówek stopnia pierwszego</h1>

    </body>
</html>
```

Rysunek 3.5 przedstawia przykład działania kaskady. Domyślnie dla każdego nagłówka stopnia pierwszego został zdefiniowany kolor czerwony bezpośrednio w <HEAD></HEAD> dokumentu. Następnie dla drugiego nagłówka w samym znaczniku <H1></H1> wpisałem dodatkowy styl określający kolor tekstu jako zielony o wielkości 10

punktów. Oczywiście każdy następny nagłówek stopnia pierwszego, wpisany do dokumentu i ograniczony tylko znacznikami <H1></H1> bez dodatkowych wpisów, również będzie sformatowany zgodnie z definicją znajdującą się w <HEAD></HEAD> strony.



Rysunek 3.5. *Przykład prostej kaskady*

Na początku tego rozdziału nie napisałem wszystkiego, gdyż chciałem w możliwie najłatwiejszy sposób pokazać zasadę działania kaskady. Otóż kaskady nie ograniczają się jedynie do trzech możliwości zdefiniowania stylów w dokumencie — jest jeszcze kilka innych poziomów. W praktyce wygląda to tak, że każda przeglądarka ma zdefiniowane swoje domyślne arkusze, za pomocą których formatuje znaczniki w dokumencie. Na przykład, jeżeli w kodzie strony znajduje się znacznik <H1></H1>, dla którego nie ustawiono żadnych stylów, to przeglądarka wyświetli taki nagłówek zgodnie z tym, jak ją zaprogramowano dla tego typu elementów.

Niektóre przeglądarki pozwalają na przypisanie swoich arkuszy stylów, które potrafią zastąpić domyślne formatowanie przeglądarki. W ten sposób zdefiniowany arkusz jest kolejnym poziomem kaskady.

Trzecim poziomem kaskady są style zdefiniowane przez projektanta strony WWW. Innymi słowy, są to te wszystkie polecenia, które zostały opisane w tej książce.

W praktyce wygląda to mniej więcej tak, że jeżeli internauta wejdzie na stronę, na której nie zastosowano żadnych stylów, to przeglądarka skorzysta ze swoich domyślnych ustawień. Jeżeli właściciel przeglądarki przygotuje własny arkusz stylów i wejdzie na tę samą stronę, na której nie ma zdefiniowanych żadnych stylów, to do wyświetlenia jej zawartości zostanie użyty arkusz użytkownika, gdyż jest on ważniejszy w hierarchii kaskad. Idąc dalej tym tokiem myślenia, rozpatrzmy trzeci przypadek, w którym internauta wejdzie na stronę, w której umieszczone dowolne style. Teraz załóżmy, że nadal korzysta z przeglądarki, w której zdefiniował swój własny arkusz stylów. W takim przypadku do wyświetlenia strony zostanie użyty styl przygotowany przez autora strony, gdyż stoi on wyżej w hierarchii niż domyślny styl przeglądarki oraz arkusz internauty. Na rysunku 3.6 dokładnie widać to, co starałem się opisać powyżej.

Oczywiście w przypadku arkusza autora mamy do czynienia z kolejnymi kaskadami, o których wspominałem na samym początku.

Rysunek 3.6. *W rywalizacji arkuszy wygrywa arkusz autora, zaraz za nim jest internauta, a na ostatnim miejscu znajduje się domyślny arkusz przeglądarki*

Wewnątrz samego arkusza stylów również mamy do czynienia z kaskadami, a idealnym przykładem takiej sytuacji był listing przedstawiony na początku tego rozdziału oraz rysunek 3.5. W jednym dokumencie starły się style wpisane do nagłówka <HEAD></HEAD> ze stylami wpisanymi bezpośrednio do danego znacznika. Pomimo tego, że obie deklaracje dotyczyły tego samego elementu, wygrał styl wpisany bezpośrednio do formatowanego akapitu. W razie potrzeby sytuację taką możemy zmienić poprzez zastosowanie polecenia `!important`.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "http://www.w3.org/TR/
➥REC-html40/strict.dtd">
  <html>
    <head>
      <meta http-equiv="Content-type" content="text/html;charset=ISO-8859-2">
      <meta name="Keywords" content="słowa, kluczowe, dla, strony www">
      <meta name="Description" content="Krótki opis strony WWW">
      <meta name="Author" content="Bartosz Danowski">
      <meta name="Copyright" content="Wydawnictwo HELION">
      <title>Tytuł Strony WWW</title>
        <style type="text/CSS">
             H1
             {
             color: red !important;
             }
        </style>

    </head>
    <body>
<h1>Nagłówek stopnia pierwszego formatowany przez arkusz CSS wpisany w HEAD</h1>
     <h1 style="color: green; font-size: 10pt">Nagłówek stopnia pierwszego</h1>

    </body>
</html>
```

Przykład, który jest widoczny powyżej, spowoduje wyświetlenie nagłówków o dwóch różnych wielkościach ale o identycznym kolorze. Stanie się tak pomimo tego, że bezpośrednio do <H1></H1> wpisałem styl nadający temu elementowi kolor zielony.

Podsumowanie

Zrozumienie pojęcia kaskadowość i dziedziczenie ma ogromne znaczenie dla świadomej pracy ze stylami dlatego jeżeli masz jakieś problemy z którymś z elementów opisanych w tym rozdziale to spróbuj jeszcze raz się z nim zapoznać. W przypadku

gdy nadal czegoś nie rozumiesz spróbuj przepisać poszczególne przykłady a jeżeli to nic nie da napisz do mnie. Pytania prześlij na adres *eathan@irc.pl* a w miarę swoich możliwości postaram Ci się pomóc.