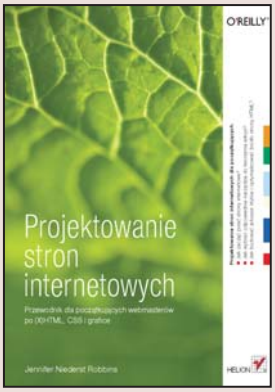
Projektowanie stron internetowych. Przewodnik dla początkujących webmasterów po (X)HTML, CSS i grafice



Autor: Jennifer Niederst Robbins
Tłumaczenie: Anna Trojan
ISBN: 978-83-246-1375-5
Tytuł oryginału: Learning Web Design: A Beginner
Format: 180x235, stron: 488

- Jak zacząć pisać strony internetowe?
- Jak wybrać odpowiednie narzędzia do tworzenia witryn?
- Jak budować arkusze stylów i optymalizować źródło strony HTML?

Od czego mam zacząć? Czy ja się do tego nadaję? Nie mam na to czasu... Wielu z nas właśnie z takim nastawianiem zabiera się do pisania swojej pierwszej strony internetowej. Takie i podobne wątpliwości rozwiewa właśnie ta książka, przeznaczona dla osób niemających żadnej wiedzy na temat tworzenia stron internetowych, a które chciałyby taką stronę wykreować. Czytając ją i pracując nad wieloma przykładami, nauczysz się, jak opracować swoją pierwszą stronę internetową i stopniowo odkryjesz w sobie pasję webmastera!

Jennifer Niederst Robbina, bazując na swoim kilkunastoletnim doświadczeniu w dziedzinie tworzenia stron internetowych, udowadnia, że pisać strony może każdy, należy mu tylko wskazać drogę. Książka „Projektowanie stron internetowych" jest trzecią edycją przewodnika dla początkujących, lecz napisana została całkowicie od początku, z uwzględnieniem najnowszych technologii i trendów w tej dziedzinie. Dodatkowym atutem książki jest przejrzystość i łatwo przyswajalny język oraz liczne przykłady i ćwiczenia, które pozwalają lepiej zrozumieć i przyswoić materiał.

- Struktura i znaczniki HTML
- Elementy struktury (X)HTML
- Tabele, obrazki, odnośniki, animacje i inne elementy stron
- Formatowanie tekstu
- Formularze i pola edycji
- Pływanie oraz pozycjonowanie elementów stron
- Arkusze stylów CSS
- Układ strony oparty na arkuszach stylów CSS
- Techniki CSS
- Grafika stron internetowych i jej optymalizacja
- Umieszczanie stron w Internecie

Poznaj techniki tworzenia, napisz i umieść własną stronę internetową w sieci!

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

Spis treści

ROZDZIAŁ 11

Zorientowanie na kaskadowe arkusze stylów

W TYM ROZDZIALE

Zalety oraz możliwości kaskadowych arkuszy stylów (CSS)

W jaki sposób znaczniki (X)HTML tworzą strukturę dokumentu

Pisanie reguł stylów CSS

Dołączanie stylów do dokumentów (X)HTML

Najważniejsze koncepcje z CSS: dziedziczenie, kaskada, specyficzność, kolejność reguł oraz model pojemnika

O arkuszach stylów wspomnieliśmy już kilka razy, a teraz w końcu się nimi zajmiemy, by nadać tworzonym stronom tak potrzebny styl. Kaskadowe arkusze stylów (Cascading Style Sheets, CSS) to standard W3C dotyczący definiowania prezentacji dokumentów napisanych w językach HTML, XHTML, a nawet XML. Prezentacja odnosi się do sposobu wyświetlania dokumentu lub dostarczania go do użytkownika — obojętnie czy na ekranie monitora komputera, wyświetlaczu telefonu komórkowego, czy też za pomocą odczytania na głos przez odpowiedni program. Kiedy CSS zajmuje się prezentacją, (X)HTML może powrócić do swojej pierwotnej roli — definiowania struktury dokumentu oraz znaczenia.

CSS jest osobnym językiem z własną składnią. Niniejszy rozdział omawia terminologię CSS wraz z najważniejszymi koncepcjami, które pomogą nam zrozumieć kolejne rozdziały, gdzie nauczymy się, w jaki sposób zmienia się style tekstu oraz czcionek, dodaje kolory i tła, a nawet tworzy podstawowy układ graficzny strony w oparciu o CSS. Czy każdy po lekturze części III książki będzie ekspertem od arkuszy stylów? Najprawdopodobniej nie. Na pewno jednak każdy będzie miał solidne podstawy oraz sporo praktyki.

UWAGA

W podrozdziale „Dalsza nauka CSS" na końcu rozdziału znajduje się lista książek i stron, które pomogą Czytelnikom kontynuować edukację w zakresie kaskadowych arkuszy stylów.

Zalety CSS

Oczywiście nikt już raczej nie potrzebuje, żeby go przekonywać, że arkusze stylów to dobra droga, ale mimo wszystko warto krótko podsumować zalety korzystania z CSS.

- Lepsza kontrola nad czcionkami oraz układem strony. Prezentacyjny (X)HTML nie jest w stanie zaoferować tego rodzaju kontroli nad czcionkami, tłem czy układem strony, jaka możliwa jest za pomocą CSS.
- Mniej pracy. Wygląd całej witryny internetowej można zmienić za jednym razem, edytując jeden arkusz stylów. Dokonywanie drobnych poprawek, a nawet przeprojektowanie całej strony za pomocą arkuszy stylów jest o wiele łatwiejsze, niż kiedy instrukcje prezentacyjne pomieszane są z kodem.

- Potencjalnie mniejsze dokumenty i krótszy czas pobierania. Stosowana niegdyś praktyka wykorzystywania zbędnych elementów `font` oraz zagnieżdżonych tabel powodowała rozdęcie dokumentów do niepotrzebnie dużych rozmiarów. Ale to nie wszystko: można także zastosować jeden arkusz stylów do wszystkich stron dla jednej witryny, oszczędzając kolejnych kilka bajtów.
- Bardziej dostępne strony. Kiedy za kwestie związane z prezentacją odpowiedzialny jest CSS, można oznaczyć zawartość dokumentu w sposób znaczący, semantyczny, czyniąc go bardziej dostępnym dla przeglądarek niewizualnych lub urządzeń mobilnych.
- Stabilna obsługa w przeglądarkach. Prawie każda przeglądarka będąca aktualnie w użyciu obsługuje wszystkie właściwości CSS Level 1 i większość CSS Level 2 (w ramce „Spotkanie ze standardami" wyjaśniono, czym są owe „poziomy").

Kiedy o tym pomyśleć, wykorzystywanie arkuszy stylów nie ma tak naprawdę żadnych wad. Istnieje kilka pozostałości po czasach niespójnej obsługi CSS w przeglądarkach, ale albo można ich uniknąć, albo spróbować je obejść, jeśli wie się już, gdzie się ich spodziewać. Na pewno nie może to być powód do zrezygnowania z CSS.

Spotkanie ze standardami

Pierwsza oficjalna wersja CSS (CSS Level 1 Recommendation, w skrócie CSS1) opublikowana została w 1996 roku i zawierała właściwości służące do dodawania instrukcji dotyczących czcionek, koloru oraz odstępu do elementów strony. Niestety, brak obsługi w przeglądarkach sprawił, że przyjmowanie tego standardu rozciągnęło się na kilka lat.
Specyfikacja CSS Level 2 (CSS2) opublikowana została w 1998 roku. Dodała przede wszystkim właściwości służące do pozycjonowania, które pozwoliły na wykorzystanie CSS w tworzeniu układu strony. Wprowadziła również style dla innych typów mediów (jak druk, urządzenia trzymane w dłoni czy urządzenia służące do odsłuchiwania treści), a także bardziej zaawansowane metody wybierania elementów do stylizacji. Specyfikacja CSS Level 2, Revision 1 (CSS2.1) wprowadza pewne drobne poprawki do CSS2 i w momencie pisania niniejszej książki ma status Candidate Recommendation. Na szczęście większość aktualnych przeglądarek obsługuje większą część specyfikacji CSS1, CSS2 oraz CSS2.1. Tak naprawdę niektóre przeglądarki obsługują już nawet opcje ze specyfikacji CSS Level 3 (CSS3), która nadal jest rozwijana. Ta wersja dodaje obsługę tekstu pionowego, poprawia obsługę tabel oraz języków międzynarodowych, zapewnia lepszą integrację z innymi technologiami XML i inne drobnostki, takie jak wstawianie wielu obrazków tła do jednego elementu oraz możliwość ustalenia dłuższej listy nazw kolorów.
Aktualny rozwój standardu CSS przez W3C można śledzić na stronie *www.w3.org/Style/CSS*.

Siła CSS

Nie mówimy tutaj o drobnych sztuczkach w zakresie wyglądu stron, takich jak zmiana koloru nagłówków czy dodawanie wcięcia do tekstu. Kiedy używa się CSS w pełni jego możliwości, jest to potężne narzędzie do projektowania o wielu możliwościach. Ja przekonałam się do CSS po tym, jak zobaczyłam różnorodność oraz bogactwo projektów z CSS Zen Garden (*www.csszengarden.com*). Na rysunku 11.1 widać tylko kilka moich ulubionych. Wszystkie projekty wykorzystują dokładnie ten sam dokument źródłowy XHTML. Co więcej, żaden nie zawiera ani jednego elementu `img` (wszystkie obrazki wykorzystane zostały jako tło). Warto zauważyć, jak bardzo różnią się od siebie te projekty i jak są przy tym wyszukane; wszystko to zostało osiągnięte za pomocą arkuszy stylów.

Oczywiście tworzenie układów strony opartych na CSS i widocznych na rysunku 11.1 wymaga sporo praktyki. Zaawansowane umiejętności w projektowaniu grafiki także będą przydatne (niestety, ich opisu nie znajdzie się w niniejszej książce). Pokazuję te przykłady, ponieważ chcę, by każdy był świadom potencjału projektów stron opartych na CSS, szczególnie że przykłady z książki przeznaczonej dla osób początkujących siłą rzeczy muszą być proste. Należy się uczyć, jednak nie można zapominać o celu.

Jak działają arkusze stylów

To proste jak liczenie od 1 do 3!

1. Należy rozpocząć od dokumentu, który został oznaczony za pomocą HTML bądź XHTML.
2. Teraz trzeba napisać reguły stylów określające, jak powinny wyglądać poszczególne elementy.
3. Należy dołączyć reguły stylów do dokumentu. Kiedy przeglądarka wyświetli dokument, będzie stosowała się do reguł wyświetlania elementów z arkusza stylów (o ile użytkownik nie zastosował własnych stylów, do czego dojdziemy za chwilę).

Oczywiście tak naprawdę to nie wszystko. Rozważmy każdy z tych kroków nieco bardziej szczegółowo.

Faded Flowers
autor: Mani Sheriar

CSS Zen Dragen
autor: Matthew Buchanan

Shaolin Yokobue
by Javier Cabrer

By the Pier
autor: Peter OngKelmscott

Organica Creativa
autor: Eduardo Cesario

Kelmscott
autor: Bronwen Hodgkinson

Contemporary Nouveau
autor: David Hellsing

Rysunek 11.1. *Wszystkie strony z CSS Zen Garden wykorzystują ten sam dokument źródłowy XHTML, jednak projekt zmieniany jest za pomocą samego CSS (obrazki wykorzystane za zgodą CSS Zen Garden oraz poszczególnych autorów).*

ćwiczenie 11.1.

Twój pierwszy arkusz stylów

W tym ćwiczeniu dodamy kilka prostych stylów do krótkiego artykułu. Dokument XHTML zatytułowany *twenties.html* wraz z powiązanym z nim obrazkiem *twenty_20s.jpg* dostępne są w materiałach do książki na stronie http://helion.pl/ksiazki/prstin.htm. Należy najpierw otworzyć stronę w przeglądarce, by zobaczyć, jak wygląda ona domyślnie (powinna wyglądać mniej więcej tak, jak na rysunku 11.2). Można również otworzyć dokument w edytorze tekstu i przygotować się do śledzenia pracy nad nim w kolejnym podrozdziale.

Rysunek 11.2. *Tak wygląda artykuł bez żadnych instrukcji dotyczących stylów. Choć nie zrobimy z niego cudu, zobaczymy, jak style działają w praktyce.*

1. Oznaczanie dokumentu

Dzięki lekturze poprzednich rozdziałów wiemy już sporo o oznaczaniu zawartości dokumentu. Wiemy na przykład, jak ważne jest wybieranie elementów (X)HTML, które w adekwatny sposób opisują znaczenie tej zawartości. Pisałam również, że znaczniki tworzą strukturę dokumentu, czasami nazywaną warstwą strukturalną, na bazie której stosuje się warstwę prezentacji.

W tym rozdziale oraz w kolejnych zobaczymy, że zrozumienie struktury dokumentu oraz relacji między elementami jest kluczowym elementem pracy twórcy arkuszy stylów.

By poczuć, jak łatwo można zmienić wygląd strony dokumentu za pomocą arkuszy stylów, warto spróbować pobawić się ćwiczeniem 11.1. Dobra wiadomość jest taka, że przygotowałam już niewielki dokument XHML, z którym będziemy pracować.

2. Pisanie reguł

Arkusz stylów składa się z jednej lub większej liczby instrukcji stylów (nazywanych regułami) opisujących, w jaki sposób element lub grupa elementów powinny być wyświetlane. Pierwszym krokiem w nauce CSS jest zapoznanie się z częściami reguły. Jak widać, są one dość intuicyjne. Każda reguła wybiera element i deklaruje, w jaki sposób powinien on działać.

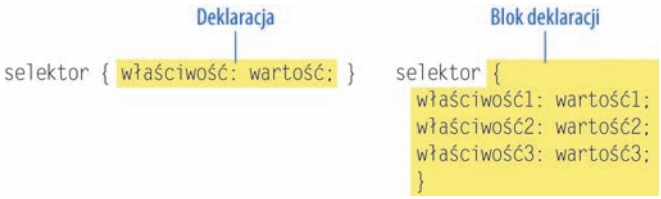
Poniższy przykład zawiera dwie reguły. Pierwsza z nich sprawia, że wszystkie elementy `h1` w dokumencie będą zielone, natomiast druga wskazuje, że akapity powinny być napisane małą czcionką bezszeryfową.

```
h1 { color: green; }
p { font-size: small; font-family: sans-serif; }
```

UWAGA

Czcionki bezszeryfowe nie mają niewielkich kresek (szeryfów) na zakończeniach kresek głównych i wyglądają bardziej elegancko oraz nowocześnie. Więcej informacji na temat czcionek znajduje się w rozdziale 12., „Formatowanie tekstu”.

W terminologii CSS dwie główne części reguły to selektor (ang. *selector*) identyfikujący element lub elementy, na które reguła ma wpływ, oraz deklaracja zawierająca instrukcje dotyczące wyświetlania. Deklaracja z kolei składa się z właściwości (takiej jak `color`) oraz jej wartości (jak `green`) rozdzielonych dwukropkiem i spacją. Deklaracje umieszcza się wewnątrz nawiasów klamrowych, jak widać na rysunku 11.3.



Rysunek 11.3. Części reguły arkusza stylów.

Selektory

W poprzednim przykładzie arkusza stylów elementy `h1` oraz `p` zostały wykorzystane jako selektory. Najbardziej podstawowy typ selektora nazywany jest selektorem typu elementu (ang. *element type selector*). Właściwości zdefiniowane powyżej będą miały zastosowanie do każdego elementu `h1` oraz `p` w dokumencie. W kolejnych rozdziałach przedstawione zostaną bardziej zaawansowane selektory, które mogą być wykorzystywane do wybierania elementów, w tym ich grup, oraz elementów pojawiających się w określonym kontekście.

Opanowanie selektorów — to znaczy wybieranie najlepszego typu selektora i wykorzystywanie go w sposób strategiczny — jest ważnym celem w staniu się Mistrzem Jedi CSS.

Deklaracje

Deklaracja składa się z pary właściwość-wartość. W pojedynczej regule może znajdować się większa liczba deklaracji — na przykład reguła dla elementu `p` zaprezentowana wcześniej zawiera właściwości `font-size` oraz `font-family`. Każda deklaracja musi się kończyć średnikiem w celu oddzielenia jej od kolejnej deklaracji (zobacz uwagę). Nawiasy klamrowe oraz deklaracje, które one zawierają, często określa się mianem bloku deklaracji (ang. *declaration block*), jak na rysunku 11.3.

Ponieważ CSS ignoruje białe znaki oraz znaki powrotu karetki wewnątrz bloku deklaracji, autorzy stron internetowych zazwyczaj piszą każdą deklarację z bloku w osobnym wierszu, jak w poniższym przykładzie. Dzięki temu łatwiej jest odnaleźć właściwości stosujące się do selektora; łatwiej jest też stwierdzić, gdzie reguła stylu się kończy.

```
p {
 font-size: small;
 font-family: sans-serif;
}
```

Warto zauważyć, że nic się właściwie nie zmieniło — nadal mamy jeden zestaw nawiasów klamrowych, średniki po każdej deklaracji i tak dalej. Jedyną różnicą jest wstawienie nowych wierszy oraz znaków spacji w celu wyrównania zapisu.

Sercem arkuszy stylów jest zbiór właściwości (ang. *properties*) standardowych, które mogą być stosowane do wybranych elementów. Pełna specyfikacja CSS definiuje dziesiątki właściwości dla wszystkiego, od wcięcia tekstu aż po sposób odczytywania nagłówków tabel. W niniejszej książce omówiono właściwości najczęściej wykorzystywane oraz najlepiej obsługiwane (zobacz uwagę).

UWAGA

Z technicznego punktu widzenia średnik nie jest wymagany po ostatniej deklaracji w bloku, jednak lepiej jest wyrobić w sobie nawyk kończenia każdej deklaracji tym znakiem. Dzięki temu łatwiej będzie później dodawać kolejne deklaracje do danej reguły.

UWAGA

Kompletna lista właściwości z aktualnego standardu CSS2.1 znajduje się w rekomendacji W3C dostępnej pod adresem www.w3.org/TR/CSS21/propidx.html. Można także zajrzeć do którejś z książek poświęconych CSS, na przykład CSS. Kaskadowe arkusze stylów. Przewodnik encyklopedyczny. Wydanie III autorstwa Erica Meyera lub Web Design in a Nutshell autorstwa Jennifer Robbins (to ja). Obie książki opublikowane zostały przez wydawnictwo O'Reilly, a w Polsce — Helion.

Podawanie wartości miar

Kiedy podaje się wartości miar, jednostka musi się znajdować bezpośrednio po liczbie, jak poniżej:

```
{ margin: 2em; }
```

Dodanie spacji przed jednostką sprawia, że właściwość nie będzie działała.

NIEPOPRAWNIE: `{ margin: 2 em; }`

Dopuszczalne jest pominięcie jednostki w miarach równych zero:

```
{ margin: 0; }
```

Wartości (ang. *values*) uzależnione są od właściwości. Niektóre właściwości przyjmują wartości długości, inne wartości koloru, a jeszcze inne mają zdefiniowaną pewną liczbę słów kluczowych. Kiedy wykorzystuje się właściwość, należy wiedzieć, jakie wartości ona przyjmuje. W wielu prostych przypadkach wystarczy rozsądek (oraz czasami znajomość języka angielskiego).

Zanim przejdziemy dalej, warto nabrać trochę wprawy przy kontynuacji ćwiczenia 11.1.

ćwiczenie 11.1.

Twój pierwszy arkusz stylów (kontynuacja)

Należy otworzyć plik *twenties.html* w edytorze tekstu. W nagłówku dokumentu można zauważyć, że wstawiłam tam element `style`, w którym będziemy wpisywać reguły stylów. Element `style` wykorzystywany jest do osadzania reguł stylów w elemencie `head` dokumentu (X)HTML.

Na początek dodamy niewielki arkusz stylów, który omawialiśmy w niniejszym podrozdziale. Należy wpisać poniższe reguły do dokumentu, tak jak poniżej:

```
<style type="text/css">
h1 {
 color: green;
}
p {
 font-size: small;
 font-family: sans-serif;
}
</style>
```

Teraz trzeba zapisać plik i zobaczyć go w przeglądarce. Powinno dać się zauważyć pewne zmiany (jeśli nasza przeglądarka wykorzystuje już czcionki bezszeryfowe, widoczna będzie tylko zmiana wielkości tekstu). Jeśli nie widać żadnych zmian, należy wrócić do dokumentu i sprawdzić, czy w kodzie znajdują się zawsze pary nawiasów klamrowych (otwierające oraz zamykające) oraz końcowe średniki. Łatwo jest przypadkowo pominąć te znaki, sprawiając jednocześnie, że arkusze stylów przestaną działać.

Teraz zmienimy arkusz stylów oraz dodamy do niego kolejne reguły, żeby zobaczyć, jak łatwe jest ich pisanie, a także zobaczyć efekty naszej pracy. Poniżej znajduje się kilka pomysłów, które można wykorzystać (należy pamiętać o każdorazowym zapisywaniu dokumentu, by zmiany były widoczne, kiedy zostanie on ponownie załadowany w przeglądarce).

- Nadajmy elementowi `h1` kolor szary (`gray`) i zobaczmy, jak wygląda w przeglądarce. Teraz zmieńmy kolor na niebieski (`blue`). Na koniec niech zostanie on czerwony (`red`). Pełna lista dostępnych nazw kolorów zostanie przedstawiona w rozdziale 13.„Kolory i tła”.
- Dodajmy nową regułę stylu, która sprawi, że również elementy `h2` będą czerwone.
- Dodajmy lewy margines o szerokości stu pikseli do elementów akapitów (`p`) za pomocą następującej deklaracji:

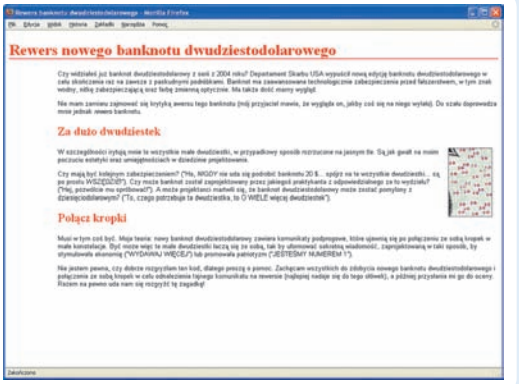
  ```
  margin-left: 100px;
  ```

  Należy pamiętać, że tę nową deklarację można dodać do istniejącej reguły dotyczącej elementu `p`.
- Teraz należy również dodać taki sam margines do nagłówków `h2`.
- Dodajmy czerwone obramowanie o szerokości jednego piksela na dole elementu `h1` za pomocą następującej deklaracji:

  ```
  border-bottom: 1px solid red;
  ```
- Należy przesunąć obrazek do prawego marginesu i pozwolić tekstowi na „opływanie” go za pomocą właściwości `float`. Skrótowa właściwość `margin` pokazana w poniższej regule dodaje odstęp o wartości zero pikseli na górze i dole obrazka, a o wartości dwunastu pikseli po jego prawej i lewej stronie (wartości są kopiowane zgodnie z metodą wyjaśnioną w rozdziale 14.„Model pojemnika”).

  ```
  img {
   float: right;
   margin: 0 12px;
  }
  ```

Kiedy wszystko jest gotowe, dokument powinien wyglądać mniej więcej jak ten pokazany na rysunku 11.4.



Rysunek 11.4. *Wygląd przykładowego artykułu po dodaniu niewielkiego arkusza stylów. Jak wspomniałam wcześniej, nie jest piękny, ale po prostu inny.*

3. Dołączanie stylów do dokumentu

W poprzednich ćwiczeniach osadziliśmy arkusz stylów bezpośrednio w dokumencie XHTML za pomocą elementu `style`. Jest to tylko jeden z dostępnych sposobów przekazania informacji o stylach do dokumentu (X)HTML. Z czasem omówimy każdy z nich z osobna, ale na razie dobrze będzie przyjrzeć się przeglądowi dostępnych metod oraz terminologii.

Zewnętrzne arkusze stylów

Zewnętrzny arkusz stylów jest odrębnym dokumentem tekstowym, który zawiera pewną liczbę reguł stylów. Musi mieć rozszerzenie *.css*. Odnośnik do tego dokumentu *.css* podawany jest następnie w dokumentach (X)HTML lub też jest do nich w jakiś sposób importowany (zostanie to omówione w rozdziale 13.). W ten sposób wszystkie pliki witryny internetowej mogą korzystać z jednego arkusza stylów. To rozwiązanie daje największe możliwości i jest też preferowaną metodą dołączania arkuszy stylów do treści dokumentów.

Osadzone arkusze stylów

To typ arkuszy stylów, z jakim pracowaliśmy w ostatnim ćwiczeniu. Umieszczane są w dokumencie za pomocą elementu `style`, a ich reguły odnoszą się tylko do danego dokumentu. Element `style` musi zostać umieszczony w części `head` dokumentu i musi też zawierać atrybut `type` identyfikujący zawartość elementu `style` jako `text/css` (aktualnie jest to jedyna dopuszczalna wartość). Poniższy przykład zawiera również komentarz (informacje na temat komentarzy znajdują się w ramce „Komentarze w arkuszach stylów”).

```
<head>
 <title>Miejsce na wymagany tytuł dokumentu</title>
 <style type="text/css">
 /* miejsce na reguły stylów */
 </style>
</head>
```

Element `style` może także zawierać atrybut `media` wykorzystywany w odniesieniu do określonych rodzajów mediów, takich jak ekran komputera, druk czy urządzenia mobilne trzymane w dłoni. Kwestie te również omówione są w rozdziale 13.

Style wewnętrzne

Właściwości oraz wartości można również zastosować do pojedynczego elementu za pomocą atrybutu `style` umieszczonego w tym elemencie, jak poniżej:

```
<h1 style="color: red">Wprowadzenie</h1>
```

By dodać kilka właściwości, wystarczy rozdzielić je średnikami, jak w poniższym przykładzie:

```
<h1 style="color: red; margin-top: 2em">Wprowadzenie</h1>
```

Style wewnętrzne (ang. *inline styles*) mają zastosowanie tylko do elementu, w którym się pojawiły. Powinno się unikać ich stosowania, o ile nadpisanie stylów pochodzących z osadzonego lub zewnętrznego arkusza stylów nie jest absolutnie konieczne. Style wewnętrzne są dość problematyczne, ponieważ umieszczają one informacje o prezentacji wewnątrz kodu nadającego dokumentowi strukturę. Sprawiają także, że wprowadzanie zmian jest o wiele trudniejsze, ponieważ w kodzie źródłowym trzeba w takim przypadku odnaleźć każde wystąpienie atrybutu `style`. Brzmi to nieco jak wady związane z używaniem przestarzałego i niezalecanego elementu `font`, prawda?

Komentarze w arkuszach stylów

Czasami przydaje się możliwość pozostawienia w arkuszu stylów komentarzy. Specyfikacja CSS ma własną składnię komentarzy, zaprezentowaną poniżej:

```
/* miejsce na komentarz */
```

Treść znajdująca się pomiędzy znakami `/*` oraz `*/` zostanie zignorowana podczas analizy arkusza stylów, co oznacza, że komentarz można zostawić w dowolnym miejscu arkusza — nawet wewnątrz reguły:

```
body { font-size: small;
/* tymczasowo */ }
```

ćwiczenie 11.2.

Zastosowanie stylu wewnętrznego

Należy otworzyć artykuł *twenties.html* w stanie, w jakim zostawiono go ostatnio po ćwiczeniu 11.1. Każdy, kto wykonał całe ćwiczenie, powinien mieć w kodzie regułę dodającą kolor do elementów h2.

Teraz należy napisać kod stylu wewnętrznego, który sprawia, że drugi element h2 stanie się purpurowy. Robi się to, dodając atrybut `style` bezpośrednio do znacznika otwierającego elementu h2, jak poniżej:

```
<h2 style="color: purple">Połączenie kropek</h2>
```

Teraz ten nagłówek staje się purpurowy, zmieniając kolor, jaki miał poprzednio. Pozostałe nagłówki h2 pozostają bez zmian.

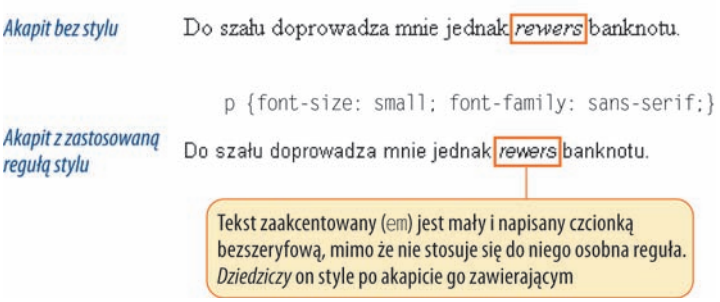
W ćwiczeniu 11.2 jest okazja do napisania kodu stylu wewnętrznego i sprawdzenia, w jaki sposób on działa. Nie będziemy już więcej pracować ze stylami wewnętrznymi, więc to jedyna taka szansa.

Najważniejsze koncepcje

Istnieje kilka podstawowych koncepcji, które należy zrozumieć, by w pełni pojąć sposób działania kaskadowych arkuszy stylów. Wprowadzę je teraz, by nie musieć zatrzymywać się później przy okazji przeglądu właściwości stylów. Każda z tych idei z pewnością pojawi się i zostanie zilustrowana przykładami w kolejnych rozdziałach.

Dziedziczenie

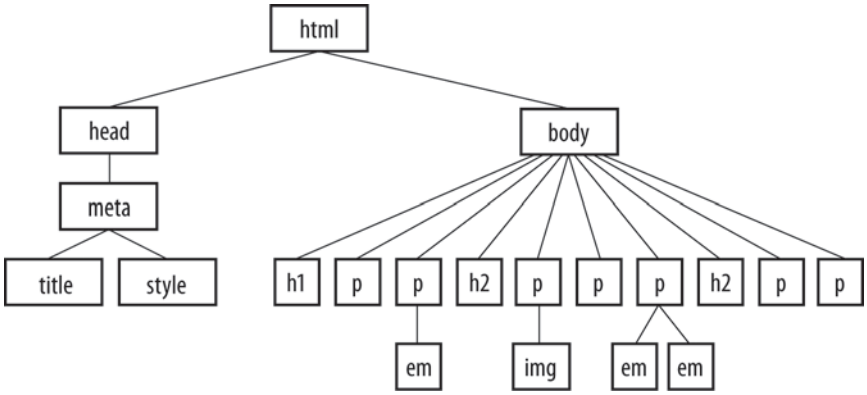
Czy oczy dziecka mają ten sam kolor co oczy rodziców? Czy dziecko dziedziczy ich kolor włosów? Unikalny uśmiech? Dokładnie tak jak rodzice przekazują pewne cechy dzieciom, elementy (X)HTML przekazują pewne właściwości stylów innym elementom, które zawierają. Można sobie przypomnieć, że kiedy w ćwiczeniu 11.1 elementy p otrzymały styl czcionki określający ich wielkość oraz typ, element em z drugiego akapitu również stał się mały i bezszeryfowy, mimo że nie napisaliśmy dla niego osobnej reguły (rysunek 11.5). Dzieje się tak, ponieważ element ten odziedziczył style po akapicie, w którym się znajduje.



Rysunek 11.5. *Element* em *dziedziczy style zastosowane do akapitu.*

Struktura dokumentu

Tutaj do gry wkracza zrozumienie struktury dokumentu. Jak wspomniałam wcześniej, dokumenty (X)HTML mają pewną strukturę czy, inaczej, hierarchię. Przykładowo artykuł, nad którym pracujemy, zawiera element `html`, w którym znajdują się elementy `head` oraz `body`. Element `body` zawiera z kolei elementy nagłówków oraz akapitów. W kilku z akapitów znajdują się za to elementy wewnętrzne, takie jak obrazki (`img`) czy tekst zaakcentowany (`em`). Strukturę dokumentu można sobie wyobrazić jako odwrócone drzewo rozgałęziające się od korzenia (elementu głównego), jak widać na rysunku 11.6.



Rysunek 11.6. Struktura drzewa dokumentu dla przykładowej strony twenties.html.

Rodzice i dzieci

Drzewo dokumentu staje się także drzewem rodzinnym (genealogicznym), jeśli chodzi o określanie związków pomiędzy elementami. Wszystkie elementy znajdujące się wewnątrz danego elementu określane są mianem jego potomków (ang. *descendant*). Przykładowo elementy `h1`, `h2`, `p`, `em` oraz `img` w dokumencie oraz na rysunku 11.6 są potomkami elementu `body`.

Element zawarty bezpośrednio wewnątrz innego elementu (bez poziomów pośrednich w hierarchii) określany jest mianem dziecka (ang. *child*) tego elementu. I odwrotnie: element go zawierający jest rodzicem (ang. *parent*). W tym przypadku element `em` jest dzieckiem elementu `p`, natomiast element `p` jest rodzicem `em`.

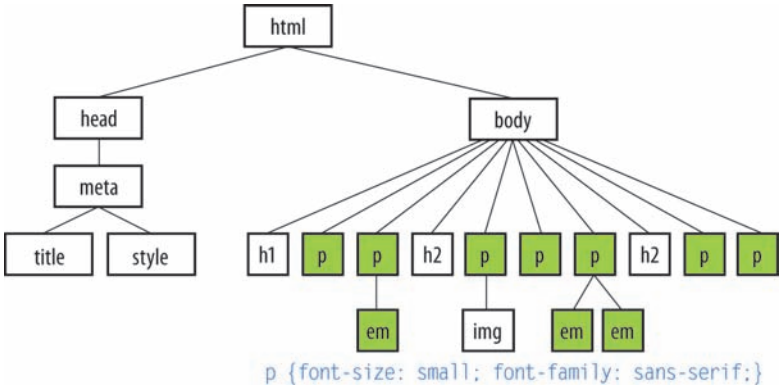
Wszystkie elementy znajdujące się w hierarchii wyżej od określonego elementu są jego przodkami. Dwa elementy mające tego samego rodzica są rodzeństwem. Nie mówimy jednak o „ciotkach" czy „kuzynach", więc na tym analogie się kończą. Ten wykład może wydawać się dość akademicki, ale przyda się, kiedy będzie mowa o pisaniu selektorów CSS.

Przekazywanie

Kiedy pisze się regułę stylu dotyczącą czcionki, wykorzystując `p` jako selektor, reguła ta odnosi się do wszystkich akapitów dokumentu, a także wszystkich tekstowych elementów wewnętrznych, jakie one zawierają. Dowód prawdziwości tego stwierdzenia można było zaobserwować jeszcze na rysunku 11.5 na podstawie elementu `em` dziedziczącego właściwości stylów zastosowane do elementu `p`. Na rysunku 11.7 zaprezentowano, co się dzieje, na diagramie przedstawiającym strukturę dokumentu. Warto zauważyć, że nie uwzględniono elementu `img`, ponieważ właściwości związane z czcionkami nie mają zastosowania do obrazków.

WSKAZÓWKA DOTYCZĄCA CSS

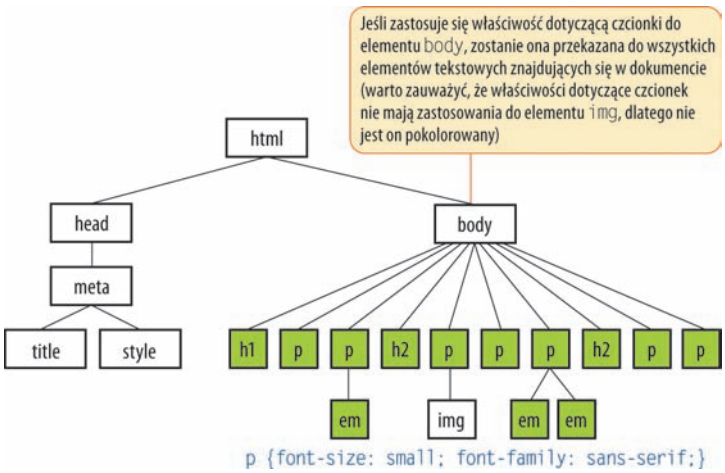
Kiedy poznajemy nową właściwość stylu CSS, dobrze jest zapamiętać, czy jest ona dziedziczona. Kwestia dziedziczenia wymieniona jest w przypadku każdej właściwości omówionej w książce. W większości przypadków dziedziczenie zachowuje się zgodnie z oczekiwaniami.



Rysunek 11.7. *Niektóre właściwości mające zastosowanie do elementu* `p` *są dziedziczone przez dzieci tego elementu.*

Warto zwrócić uwagę na to, że napisałam, iż dziedziczone są niektóre właściwości. Należy pamiętać, że pewne właściwości arkuszy stylów są dziedziczone, podczas gdy inne nie. Generalnie właściwości związane ze stylizacją tekstu — jak rozmiar czcionki, jej kolor czy styl — przekazywane są do elementów potomnych. Właściwości takie, jak marginesy, obramowanie, tło i podobne, wpływające na zamknięty obszar znajdujący się wokół elementu, zazwyczaj nie są przekazywane. Jeśli się nad tym zastanowić, ma to sens. Jeżeli na przykład wokół akapitu umieści się obramowanie, nie chce się przecież zobaczyć tego obramowania wokół każdego elementu wewnętrznego (takiego, jak `em`, `strong` czy `a`), który on zawiera.

Dziedziczenie można wykorzystać, kiedy pisze się arkusze stylów. Jeśli na przykład chce się, by wszystkie elementy tekstowe były napisane czcionką Verdana, można zapisać osobne reguły stylów dla każdego elementu w dokumencie i ustawić właściwość `font-face` na `Verdana`. Zdecydowanie lepszym sposobem będzie jednak napisanie jednej reguły stylu, która przypisze właściwość `font-face` do elementu `body` i pozwoli, by wszystkie elementy tekstowe w nim się znajdujące odziedziczyły ten styl (rysunek 11.8).



Rysunek 11.8. *Wszystkie elementy dokumentu dziedziczą pewne właściwości stylu zastosowane do elementu* `body`.

Każda właściwość zastosowana do określonego elementu powoduje zastąpienie wartości odziedziczonych dla tej właściwości. Powracając do przykładowego artykułu, można na przykład określić, że element `em` powinien być napisany czcionką szeryfową, co nadpisałoby odziedziczone ustawienie czcionki na bezszeryfową.

Konflikt stylów i kaskada

Każdy z pewnością kiedyś się zastanawiał, dlaczego arkusze stylów nazywane są kaskadowymi. CSS pozwala na zastosowanie kilku arkuszy stylów do jednego dokumentu, co oznacza, że mogą wystąpić konflikty pomiędzy nimi. Co na przykład powinna zrobić przeglądarka, kiedy zaimportowany arkusz stylów mówi, że elementy `h1` powinny być czerwone, podczas gdy osadzony arkusz stylów zawiera regułę ustawiającą je na kolor purpurowy?

Osoby tworzące specyfikację arkuszy stylów przewidziały ten problem i wymyśliły system hierarchiczny, który przypisuje różne wagi do różnych źródeł informacji dotyczących stylów. Kaskada odnosi się do sytuacji, kiedy kilka źródeł informacji dotyczących stylów rywalizuje o kontrolę nad elementami znajdującymi się na stronie — informacje o stylach przekazywane są w dół, dopóki nie zostaną nadpisane przez polecenie dotyczące stylu mające większą wagę.

Jeśli na przykład nie zastosuje się żadnych reguł dotyczących stylów dla strony internetowej, zostanie ona wyświetlona zgodnie z wbudowanym arkuszem stylów przeglądarki (nazywamy to wyświetlaniem domyślnym). Jeśli jednak autor witryny udostępnia arkusz stylów dla tego dokumentu, ma on większą wagę i przeważy nad stylami przeglądarki. Również użytkownicy mogą stosować własne style do dokumentów, co omówione jest w ramce „Arkusze stylów użytkownika".

Jak wskazano wcześniej, istnieją trzy sposoby dołączania informacji o stylu do dokumentu źródłowego i one także mają odpowiednią kolejność w tej kaskadzie. Ogólnie rzecz ujmując, im bardziej arkusz stylów zbliżony jest do zawartości dokumentu, tym większą ma wagę. Osadzone arkusze stylów, które znajdują się w elemencie `style` w dokumencie, mają większą wagę od arkuszy zewnętrznych. W przykładzie rozpoczynającym niniejszy podrozdział elementy `h1` stałyby się purpurowe, tak jak określono to w osadzonym arkuszu stylów, a nie czerwone, jak podano w zewnętrznym pliku *.css*, który ma mniejszą wagę. Style wewnętrzne mają z kolei większą wagę od osadzonych arkuszy stylów, ponieważ nie można bardziej zbliżyć się do zawartości elementu, niż wstawiając styl bezpośrednio do znacznika otwierającego elementu.

Żeby zapobiec nadpisaniu określonej reguły, należy przypisać jej ważność za pomocą wskaźnika `!important`, jak zostanie to wyjaśnione w ramce „Przypisywanie ważności".

W ramce „Hierarchia arkuszy stylów" znajduje się przegląd porządku kaskady od najbardziej ogólnego do najbardziej szczegółowego.

Arkusze stylów użytkownika

Użytkownicy mogą pisać swoje własne arkusze stylów i stosować je do stron oglądanych w ich przeglądarce. Rekomendacja CSS określa je mianem arkuszy stylów czytelnika (ang. *reader style sheets*), choć w praktyce częściej stosuje się nazwę arkusze stylów użytkownika.

Zazwyczaj reguły stylów znajdujące się w arkuszu stylów autora strony (zewnętrznym, osadzonym lub wewnętrznym) wygrywają z arkuszem stylów użytkownika. Jeśli jednak użytkownik oznaczy styl jako ważny, będzie on triumfował nad wszystkimi innymi stylami dostarczonymi przez autora strony, a także wbudowanymi arkuszami stylów przeglądarki (więcej informacji na ten temat znajduje się w ramce „Przypisywanie ważności"). Jeśli zatem na przykład użytkownik z upośledzeniem wzroku nadpisze regułę nakazującą wyświetlanie każdego tekstu bardzo dużą czarną czcionką na białym tle, będzie miał gwarancję, że tak właśnie się stanie. Dokładnie o to chodziło W3C, kiedy zaproponowano wprowadzenie arkuszy stylów użytkownika i tym samym umożliwienie użytkownikowi nadpisywania wszelkich innych stylów.

Specyficzność

Po wybraniu arkusza stylów, który ma zastosowanie, nadal mogą istnieć konflikty, dlatego kaskada kontynuowana jest na poziomie reguł. Kiedy dwie reguły w jednym arkuszu stylów znajdują się w konflikcie, do wyboru zwycięzcy wykorzystywany jest typ selektora. Im bardziej specyficzny (szczegółowy) jest selektor, tym większą wagę otrzymuje przy zastępowaniu deklaracji będących z nim w konflikcie.

Przypisywanie ważności

Jeśli chce się, by reguła nie mogła zostać nadpisana przez kolejną regułę będącą z nią w konflikcie, należy dodać wskaźnik ważności `!important` (od angielskiego ***important*** — ważny) tuż po wartości właściwości, a przed średnikiem dla tej reguły. Żeby na przykład tekst akapitów zawsze był niebieski, należy skorzystać z poniższej reguły:

```
p {color: blue !important;}
```

Nawet jeśli przeglądarka napotka styl wewnętrzny w dalszej części dokumentu (co w normalnych warunkach spowodowałoby nadpisanie reguły z arkusza stylów dokumentu), jak poniżej:

```
<p style="color: red">
```

akapit ten nadal będzie niebieski, ponieważ reguła oznaczona jako ważna za pomocą wskaźnika `!important` nie może być nadpisana przez inne reguły arkusza stylów autora strony.

Jedyną sytuacją, w jakiej reguła `!important` może być nadpisana, jest taka, w której mamy do czynienia z będącą z nią w konflikcie regułą z arkusza stylów użytkownika również oznaczoną jako ważna. Taka hierarchia ma na celu zapewnienie, że specjalne wymagania użytkownika, na przykład duża czcionka w przypadku osób niedowidzących, nigdy nie zostaną zastąpione.

W oparciu o poprzednie przykłady można stwierdzić, że gdyby arkusz stylów użytkownika zawierał następującą regułę:

```
p {color: black;}
```

tekst nadal byłby niebieski, ponieważ wszystkie style autora strony (nawet te nieoznaczone jako ważne) mają pierwszeństwo przed regułami użytkownika. Jeśli jednak reguła będąca w konflikcie oznaczona jest w arkuszu stylów użytkownika jako `!important`, jak poniżej:

```
p {color: black !important;}
```

akapity pozostaną czarne i nie może to zostać zmienione za pomocą stylów dostarczonych przez autora strony.

Hierarchia arkuszy stylów

Informacje dotyczące stylów mogą pochodzić z różnych źródeł, wymienionych poniżej od najbardziej ogólnych do najbardziej szczegółowych. Elementy znajdujące się niżej na liście będą wygrywały z elementami znajdującymi się wyżej.

- Domyślne ustawienia przeglądarki.
- Ustawienia stylów użytkownika (ustawione w przeglądarce jako arkusze stylów użytkownika).
- Zewnętrzne arkusze stylów (dodane za pomocą elementu `link`).
- Zaimportowane arkusze stylów (dodane za pomocą funkcji `@import`).
- Osadzone arkusze stylów (dodane za pomocą elementu `style`).
- Informacje o stylach wewnętrznych (dodane za pomocą atrybutu `style` umieszczonego w znaczniku otwierającym elementu).
- Dowolna reguła oznaczona jako `!important` przez autora strony.
- Dowolna reguła oznaczona jako `!important` przez użytkownika (czytelnika) strony.

Na razie jest nieco za wcześnie na omawianie specyficzności, ponieważ póki co zapoznaliśmy się z jednym rodzajem selektorów (i na dodatek tym najmniej specyficznym czy też szczegółowym). Odłóżmy zatem pojęcie specyficzności (ang. *specificity*) oraz kwestię niektórych selektorów nadpisujących inne. Zagadnienia te omówione zostaną w rozdziale 12., kiedy będziemy już znali większą liczbę typów selektorów.

Kolejność reguł

I wreszcie, jeśli istnieją konflikty w ramach reguł stylów o identycznej wadze, wygrywa ta, która znajduje się na końcu listy. Weźmy pod uwagę na przykład trzy poniższe reguły:

```
<style type="text/css">
  p { color: red; }
  p { color: blue; }
  p { color: green; }
</style>
```

W tym scenariuszu tekst akapitu będzie zielony, ponieważ ostatnia reguła z arkusza stylów, czyli ta najbliżej treści dokumentu, nadpisuje wcześniejsze.

Model pojemnika

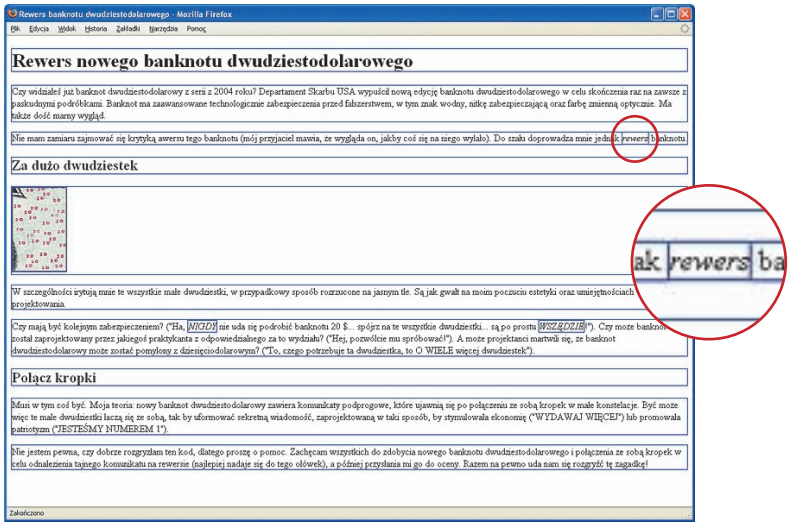
Skoro już omawiamy najważniejsze zagadnienia związane z kaskadowymi arkuszami stylów, należy wspomnieć o kluczowym elemencie systemu formatowania wizualnego CSS — modelu pojemnika (ang. *box model*). Najłatwiejszy sposób wyobrażenia sobie tego modelu to założenie, że przeglądarki widzą każdy element na stronie internetowej (zarówno blokowy, jak i wewnętrzny) jako zawarty w małym, prostokątnym pojemniku. Do tych pojemników mają zastosowanie właściwości dotyczące obramowania, marginesów, dopełnienia czy tła. Pojemniki te można nawet przemieszczać i pozycjonować na stronie.

Więcej szczegółów dotyczących modelu pojemnika znajduje się w rozdziale 14., jednak ogólne zrozumienie tego modelu przyda się nawet w dyskusji dotyczącej tekstów oraz tła w kolejnych dwóch rozdziałach.

By zobaczyć te elementy mniej więcej w sposób, w jaki widzi je przeglądarka, napisałam reguły stylów dodające obramowanie wokół każdego elementu zawierającego treść w naszym przykładowym artykule.

```
h1 { border: 1px solid blue; }
h2 { border: 1px solid blue; }
p { border: 1px solid blue; }
em { border: 1px solid blue; }
img { border: 1px solid blue; }
```

Na rysunku 11.9 zaprezentowano rezultat zastosowania tych reguł. Obramowanie ujawnia kształt każdego z pojemników elementów blokowych. Także wokół elementów wewnętrznych (`em` oraz `img`) znajdują się pojemniki. Warto zauważyć, że pojemniki elementów blokowych rozszerzają się, tak by wypełnić dostępną szerokość okna przeglądarki, co jest naturalne dla elementów blokowych w normalnym układzie dokumentu. Pojemniki wewnętrzne zawierają jedynie te znaki lub obrazki, jakie się w nich znajdują.



Rysunek 11.9. *Obramowanie znajdujące się wokół wszystkich elementów odkrywa pojemniki elementów.*

UWAGA

Reguła „ostatnia wymieniona wygrywa" ma także zastosowanie w innych kontekstach w CSS. Przykładowo późniejsze deklaracje z bloku deklaracji mogą nadpisywać deklaracje wcześniejsze. Dodatkowo zewnętrzne arkusze stylów wymienione w kodzie źródłowym później będą miały pierwszeństwo przed tymi wymienionymi przed nimi (nawet jeśli wcześniejsze reguły to arkusze stylów osadzone w elemencie `style`*).*

Grupowanie selektorów

SZYBKI QUIZ

To dobra okazja, żeby spróbować odpowiedzieć na następujące pytanie: „Dlaczego nie dodałam po prostu właściwości `border` do elementu `body` i nie pozwoliłam pozostałym elementom zebranym w zgrupowanym selektorze na odziedziczenie jej? ”.

Odpowiedź:

Ponieważ `border` jest jedną z właściwości, które nie są dziedziczone — jak wspomniano wcześniej w tekście rozdziału.

Wydaje się, że jest to właściwy moment, żeby zaprezentować użyteczny skrót stosowany w regułach stylów. Jeśli chce się zastosować tę samą właściwość stylu do większej liczby elementów, można pogrupować selektory w jedną regułę, rozdzielając je przecinkami. Jedna reguła daje wtedy ten sam efekt co pięć reguł wymienionych wcześniej. Pogrupowanie ich sprawia, że przyszłe edycje dokumentu wykonywane będą w sposób bardziej wydajny, a dodatkowo sam plik będzie miał mniejszy rozmiar.

```
h1, h2, p, em, img { border: 1px solid blue; }
```

Teraz znamy już dwa rodzaje selektorów — prosty selektor elementów oraz selektory pogrupowane.

Dalsza nauka CSS

W niniejszym rozdziale omówiono podstawy kaskadowych arkuszy stylów, w tym składnię reguł, sposoby zastosowania stylów do dokumentu, a także kluczowe zagadnienia dziedziczenia, kaskady oraz modelu pojemnika. Arkusze stylów nie powinny już być wielką zagadką i od teraz będziemy po prostu budować na tych podstawach, dodając do naszego arsenału kolejne właściwości oraz selektory, a także rozszerzając koncepcje wprowadzone tutaj.

CSS jest bardzo rozległym zagadnieniem, zdecydowanie wykraczającym poza zakres niniejszej książki. Księgarnie oraz Internet wypełnione są pozycjami i informacjami na temat arkuszy stylów przeznaczonymi dla użytkowników o różnym poziomie umiejętności. Poniżej zamieszczam listę zasobów, które w moim procesie uczenia się okazały się najbardziej przydatne. Dodałam także listę narzędzi wspomagających pisanie arkuszy stylów i zamieściłam ją w ramce „Narzędzia CSS”.

Książki

Istnieje wiele dobrych książek na temat CSS, jednak poniżej znajduje się lista tych, z których sama się uczyłam, dlatego z czystym sumieniem mogę je polecić:

CSS. Kaskadowe arkusze stylów. Przewodnik encyklopedyczny. Wydanie III autorstwa Erica Meyera (Helion; oryginalne wydanie anglojęzyczne — O’Reilly),

Web Standards Solutions: The Markup and Style Handbook autorstwa Dana Cederholma (Friends of Ed),

Zen stosowania CSS. Źródło oświecenia dla projektantów stron WWW autorstwa Dave’a Shea oraz Molly E. Holzschlag (Helion; oryginalne wydanie anglojęzyczne — New Riders),

CSS według Erica Meyera. Sztuka projektowania stron WWW autorstwa Erica Meyera (Helion; oryginalne wydanie anglojęzyczne — New Riders).

Zasoby internetowe

Poniższe strony stanowią dobry punkt wyjścia do zdobywania wiedzy na temat kaskadowych arkuszy stylów w Internecie.

World Wide Web Consortium (*www.w3.org/Style/CSS*)
Konsorcjum W3C nadzoruje rozwój technologii webowych, w tym CSS.

A List Apart (*www.alistapart.com*)
Ten magazyn internetowy zawiera niektóre z najlepszych artykułów dotyczących myślenia oraz pisania w kategoriach wybitnego, zgodnego ze standardami projektowania stron internetowych. Został założony w 1998 roku przez Jeffreya Zeldmana oraz Briana Platza.

css-discuss (*www.css-discuss.org*)
To lista mailingowa oraz powiązana z nią strona poświęcona rozmowom na temat CSS oraz stosowaniu tego standardu.

Pokazowe strony oparte na CSS, źródła inspiracji

Jeśli szuka się doskonałych przykładów tego, co można osiągnąć za pomocą CSS, należy zapoznać się z poniższymi stronami.

CSS Zen Garden (*www.cssgarden.com*)
To strona pokazująca, co można osiągnąć za pomocą CSS, jednego pliku XHTML oraz twórczych pomysłów setek projektantów stron internetowych. Jej twórcą jest ekspert w dziedzinie standardów — Dave Shea. Wyżej wymieniono książkę będącą dodatkiem do tej strony.

CSS Beauty (*www.cssbeauty.com*)
Prezentacja doskonałych stron zaprojektowanych w CSS.

Ważne strony osobiste

Jednymi z najlepszych źródeł informacji dotyczących kaskadowych arkuszy stylów są blogi oraz strony osobiste osób pasjonujących się projektowaniem opartym na CSS. Poniżej znajduje się tylko kilka adresów, ale będą one dobrym punktem wyjścia do zapoznania się ze społecznością osób, dla których ważne są standardy webowe.

Stopdesign (*www.stopdesign.com*)
Douglas Bowman, guru CSS i projektowania grafiki, publikuje tutaj swoje artykuły oraz wyznaczające trendy przewodniki.

Mezzoblue (*www.mezzoblue.com*)
To strona osobista Dave'a Shea, twórcy CSS Zen Garden.

Meyerweb (*www.meyerweb.com*)
To strona osobista króla CSS — Erica Meyera.

Molly.com (*www.molly.com*)
To blog płodnej pisarki i aktywistki standardów webowych Molly E. Holzschlag.

Simplebits (*www.simplebits.com*)
To strona osobista guru standardów oraz pisarza Dana Cederholma.

Narzędzia CSS

Konsorcjum W3C utrzymuje dość aktualną listę narzędzi do tworzenia kodu CSS na stronie internetowej poświęconej kaskadowym arkuszom stylów — *www.w3.org/Style/CSS/#editors*. Poniżej znajduje się kilku moich faworytów.

Rozszerzenie Web Developer

Programiści kochają rozszerzenie Web Developer przeznaczone dla przeglądarek Firefox oraz Mozilla, napisane przez Chrisa Pedericka. Rozszerzenie to dodaje do przeglądarki pasek narzędzi, który pomaga w analizowaniu oraz przetwarzaniu dowolnej strony w oknie. Można edytować arkusz stylów oglądanej strony, a także otrzymać informacje na temat kodu (X)HTML oraz grafiki. Dodatek sprawdza również poprawność CSS, (X)HTML oraz dostępności strony internetowej. Dostępny jest na stronach *chrispederick.com/work/web-developer/* lub na stronie poświęconej dodatkom do przeglądarek z rodziny Mozilla znajdującej się pod adresem *addons.mozilla.org*.

Programy do tworzenia stron internetowych

Obecne programy typu WYSIWYG, takie jak Adobe Dreamweaver czy Microsoft Expression Web, mogą zostać tak skonfigurowane, by pisać arkusze stylów w sposób automatyczny, w miarę pracy nad projektem strony. Wadą tego rozwiązania jest to, że często taki kod nie jest napisany w najbardziej wydajny sposób (programy te mają na przykład tendencję do nadużywania atrybutu `class` do tworzenia reguł stylów). Nadal mogą jednak być dobrym punktem wyjścia dla arkusza stylów, który można potem ręcznie zmienić.

Sprawdź się!

Poniżej znajduje się kilka pytań sprawdzających znajomość podstaw kaskadowych arkuszy stylów. Odpowiedzi na nie znajdują się w dodatku A.

1. Należy zidentyfikować różne części tej reguły stylu:

```
blockquote { line-height: 1.5; }
selektor:_____________          wartość:_____________
właściwość:_____________        deklaracja:_____________
```

2. Jaki kolor będą miały akapity, jeśli poniższy osadzony arkusz stylów zostanie zastosowany do dokumentu? Dlaczego?

```
<style type="text/css">
  p { color: purple; }
  p { color: green; }
  p { color: gray; }
</style>
```

3. Należy przepisać każdy z poniższych fragmentów kodu CSS. Niektóre z nich są całkowicie niepoprawne, podczas gdy inne mogłyby być napisane w sposób bardziej wydajny.

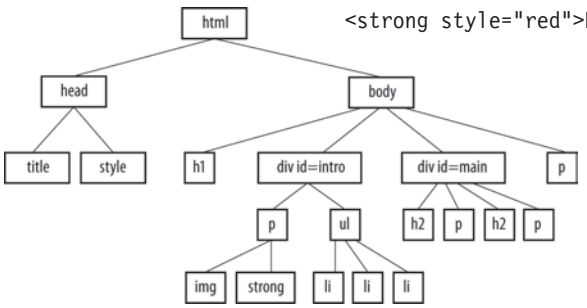
```
p {font-face: sans-serif;}
p {font-size: 1em;}
p {line-height: 1.2em;}
```

```
blockquote {
 font-size: 1em
 line-height: 150%
 color: gray }
```

```
body
 {background-color: black;}
 {color: #666;}
 {margin-left: 12em;}
 {margin-right: 12em;}
```

```
p {color: white;}
blockquote {color: white;}
li {color: white;}
```

```
<strong style="red">Działaj od razu!</strong>
```



Rysunek 11.10. Struktura przykładowego dokumentu.

4. W diagramie należy zakreślić wszystkie elementy, które powinny stać się czerwone, kiedy poniższa reguła stylów zostanie zastosowana do dokumentu XHTML ze strukturą przedstawioną na rysunku 11.10. Reguła ta wykorzystuje selektor, którego jeszcze nie omawialiśmy, ale wystarczy skorzystać z własnego rozsądku i dotychczasowej wiedzy.

```
div#intro { color: red; }
```