2

Kaskadowe arkusze stylów (CSS)

W początkowej fazie rozwoju język HTML był rzeczywiście strukturalnym językiem znaczników. W kolejnych jego wersjach dodano elementy opisujące wygląd strony (np. zmianę koloru czy wielkości tekstu). Z czasem pomieszanie warstwy strukturalnej dokumentu z warstwą prezentacyjną zaczęło stwarzać coraz większe problemy.

Rozwiązanie tych problemów stało się możliwe po oddzieleniu warstwy strukturalnej i treści od warstwy prezentacyjnej. Za warstwę strukturalną, semantyczną i treść wciąż odpowiedzialny jest język HTML, natomiast za warstwę prezentacyjną — język CSS (ang. *Cascading Style Sheets*).

Język CSS określa układ graficzny dokumentu HTML: parametry czcionki, wysokość i szerokość obrazków, ich położenie, rodzaj tła itp. Wszystkie polecenia dotyczące formatowania powinny zostać umieszczone w oddzielnym pliku (arkuszu) i być powiązane z elementami zdefiniowanymi w kodzie HTML.

Cechy języka CSS:

- Oddziela strukturę informacyjną witryny od jej warstwy prezentacyjnej.
- Daje większe możliwości formatowania tekstu.
- Pozwala zapisać wszystkie informacje dotyczące wyglądu dokumentu w oddzielnym pliku tekstowym dołączanym później do dokumentów HTML.
- Umożliwia formatowanie wielu dokumentów przy użyciu jednego arkusza stylów.
- Umożliwia stosowanie różnych układów graficznych w zależności od typu urządzenia, na którym zamierzamy wyświetlać projektowaną stronę.

Formatowanie z wykorzystaniem CSS stało się podstawą tworzenia stron internetowych. Należy pamiętać również o tym, że w języku HTML występują przestarzałe znaczniki i atrybuty formatowania (elementy zdeprecjonowane), które będą stopniowo wycofywane z przeglądarek internetowych i zastępowane nowymi metodami formatowania dostępnymi w języku CSS.

Język CSS oferuje nowe możliwości formatowania, które były niedostępne w HTML, np. dotyczące formatowania tekstu, formatowania tła, definiowania obramowania, dodatkowe właściwości definiowania list, pozycjonowania, zmianę wyglądu odsyłaczy, stosowanie filtrów.

2.1. Wstawianie stylów

Kaskadowe arkusze stylów definiują jedynie sposób formatowania elementów dokumentu HTML, ale ich nie tworzą. Elementy te muszą zostać zdefiniowane za pomocą znaczników w dokumencie HTML.

W podstawowej składni kaskadowych arkuszy stylów jest kilka stałych elementów:

```
selektor {właściwość: wartość;}
```

- *selektor* — określa, do jakich znaczników języka HTML odnosi się dalsza część definicji stylu. Selektorem może być dowolny znacznik.
- *właściwość* — określa, jaki atrybut znacznika będzie ustawiany.
- *wartość* — określa wartość atrybutu.

Przykład 2.1

```
body {background-color: #dfdfdf; }
```

Przykład pokazuje ustawienie koloru tła strony. To samo polecenie w języku HTML miałoby postać:

```
<body bgcolor="#DFDFDF">
```

Dla jednego selektora można zdefiniować kilka atrybutów. W definicji stylu należy je oddzielić średnikami.

W dokumencie HTML istnieje kilka sposobów na dołączanie stylów CSS:

- styl lokalny (w linii),
- wewnętrzny arkusz stylów,
- zewnętrzny arkusz stylów.

Sposób wstawiania stylów zależy od konkretnej sytuacji. Podczas projektowania witryny najczęściej stosuje się różne sposoby osadzania arkuszy stylów.

2.1.1. Styl lokalny (w linii)

Korzystając ze stylu lokalnego, można zdefiniować formatowanie pojedynczego elementu strony. Taki styl jest definiowany w tej samej linii, w której znajduje się element formatowany, dlatego nazywany jest on również stylem w linii (ang. *inline*). Umieszczany jest bezpośrednio w dokumencie HTML.

Przykład 2.2

```
<html>
<head>
<title> Kaskadowe arkusze stylów</title>
```

```
</head>
<body style="background-color: #DFDFDF;">
<p style="color: red">Definiowanie stylów</p>
</body>
</html>
```

Znacznik — rozciąganie stylu

Znacznik `<span>` służy do grupowania kilku elementów strony (np. słów, obrazków, akapitów) w celu nadania im określonego stylu. Ten znacznik sam w sobie nie ma narzuconych żadnych właściwości, nie wywiera żadnego wpływu na zgrupowane elementy, dlatego elementy te będzie określał wyłącznie styl. Styl dla znacznika `<span>` ustala się przy użyciu stylu CSS.

```
<span style="właściwość: wartość";> .... </span>
```

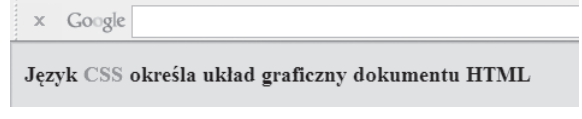
Znacznik `<span>` zwykle jest wykorzystywany wtedy, gdy trzeba inaczej sformatować kilka znaków w obszarze o określonym sposobie formatowania.

Przykład 2.3

```
<html>
<head>
<title> Kaskadowe arkusze stylów</title>
</head>
<body style="background-color: #DFDFDF;">
<h3>Język <span style="color: red">CSS</span> określa układ graficzny
dokumentu HTML</h3>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.1.

Rysunek 2.1. Zastosowanie znacznika `<span>` do fragmentu tekstu



Znacznik div — wydzielone bloki

Używając znacznika `<div>`, elementy strony można grupować w bloki i formatować za pomocą stylów. Zdefiniowane bloki mogą zawierać m.in. akapity, listy oraz inne bloki. Znaczniki `<div>` umożliwiają wydzielanie na stronie większych logicznych fragmentów w celu nadania im specyficznego formatowania z użyciem stylów.

Metoda definiowania stylu dla bloków jest podobna jak w przypadku znacznika `<span>`. Ustala się go za pomocą stylu lokalnego.

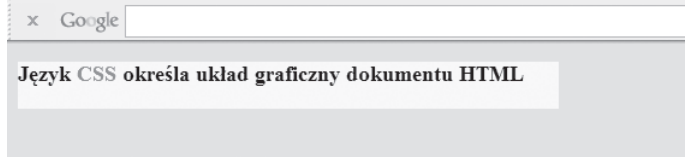
```
<div style="właściwość: wartość";> .... </div>
```

Przykład 2.4

```
<html>
<head>
<title> Bloki div</title>
```

```
</head>
<body style="background-color: #DFDFDF;">
<div style="width: 500px; height: 50px; background-color: yellow;">
<h3>Język <span style="color: red">CSS</span> określa układ graficzny
dokumentu HTML</h3></div>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.2.



Rysunek 2.2. Definiowanie bloku

2.1.2. Wewnętrzny arkusz stylów

Wewnętrzny arkusz stylów jest umieszczany w dokumencie HTML dzięki zastosowaniu znacznika `<style>`, który zawsze występuje w części nagłówkowej dokumentu (`<head>`). Tę metodę wstawiania do dokumentu arkusza stylów stosuje się, gdy elementy formatowane pojawiają się na stronie wielokrotnie i wszystkie powinny mieć takie same atrybuty formatowania. Przyjmijmy, że chcemy, żeby wszystkie teksty zdefiniowane jako akapity miały kolor niebieski. Po wpisaniu odpowiedniej deklaracji stylów wszystkie akapity będą napisane czcionką niebieską bez definiowania jej koloru przy każdym elemencie. Zmiana koloru czcionki w deklaracji stylów zmieni kolor czcionki wszystkich akapitów.

Przykład 2.5

```
<html>
<head>
<title> Kaskadowe arkusze stylów</title>
<style type="text/css">
p { color: blue; }
</style>
</head>
<body>
<p>Definiowanie stylów</p>
<p>Wewnętrzny arkusz stylów</p>
</body>
</html>
```

Przykład 2.6

```
<html>
<head>
<title> Wewnętrzny arkusz stylów</title>
```

```
<style type="text/css">
body {background-color: yellow;}
h2 {font-size: 32pt;}
h3 {color: green; font-size: 24pt;}
p {color: blue; text-align: center; }
</style>
</head>
<body>
<h2> Wewnętrzne arkusze stylów</h2>
<h3>Definiowanie stylów</h3>
<p>Arkusz stylów</p>
</body>
</html>
```

Zadanie 2.1
Sprawdź działanie kodu podanego w przykładzie 2.6.

Zadanie 2.2
Używając stylu lokalnego, zdefiniuj rodzaj, kolor i rozmiar czcionki dla tekstu wyświetlanego na stronie. To samo wykonaj, używając wewnętrznego arkusza stylów. Kiedy warto zastosować styl lokalny, a kiedy wewnętrzny arkusz stylów?

2.1.3. Zewnętrzny arkusz stylów

Największą z zalet stosowania CSS jest możliwość wstawiania zewnętrznych arkuszy stylów. Polega ona na utworzeniu pliku tekstowego z rozszerzeniem *.css*, który będzie zawierał definicję wszystkich stylów używanych w projektowanej witrynie. W dokumencie HTML powinien znaleźć się odnośnik do tzw. zewnętrznego arkusza stylów, czyli do pliku z rozszerzeniem *.css*.

Odnośnik umieszczony w dokumencie HTML powinien mieć postać:

```
<link rel="stylesheet" type="text/css" href="arkusz.css">
```

Wartością atrybutu `href` powinna być ścieżka dostępu do pliku zawierającego zdefiniowane style. Aby utrzymać porządek w strukturze plików strony, warto umieszczać pliki stylów w odrębnych folderach, np. *css*.

Odnośnik do arkusza stylów powinien zostać umieszczony w części nagłówkowej dokumentu (`<head>`).

Przykład 2.7

```
<html>
<head>
<title> Kaskadowe arkusze stylów</title>
<link rel="stylesheet" type="text/css" href="arkusz.css">
</head>
```

```
<body>
<p>Definiowanie stylów</p>
<p>Wewnętrzny arkusz stylów</p>
</body>
</html>
```

Zaletą tak zdefiniowanych arkuszy stylów jest to, że do wielu dokumentów HTML można dołączyć ten sam plik arkusza stylów, czyli za pomocą jednego pliku CSS można formatować w taki sam sposób wiele stron WWW (np. kilka stron internetowych tej samej witryny).

Można również w jednym dokumencie HTML umieszczać dowolną liczbę zewnętrznych arkuszy stylów. W dokumencie tym za każdym razem musi wystąpić osobny element `<link rel="stylesheet">` odwołujący się do arkusza stylów. Jeżeli pojawi się konflikt dotyczący sposobu formatowania tego samego elementu w różnych arkuszach stylów, wiążące będą deklaracje z arkusza dołączonego najpóźniej.

Zadanie 2.3
Utwórz plik z rozszerzeniem *.css*. Zdefiniuj w nim styl tekstu wyświetlanego na stronie oraz styl tła strony. Dołącz go do dokumentu HTML opisującego przykładową stronę internetową.

2.1.4. Import arkusza stylów

Do zewnętrznego lub wewnętrznego arkusza stylów można zaimportować zewnętrzny arkusz stylów. Plik z zaimportowanym arkuszem stylów może znajdować się w dowolnym miejscu; należy tylko prawidłowo zdefiniować ścieżkę dostępu do niego. Importowanie arkusza stylów odbywa się podobnie jak dołączanie zewnętrznego arkusza. Priorytet dołączonego arkusza jest taki sam jak priorytet arkusza, do którego został on zaimportowany.

Polecenie importowania arkusza ma postać:

```
<style type="text/css">
/* <! [CDATA[ */
@import url (adres importowanego arkusza stylów.css);
/* ]]> */
</style>
```

Adres arkusza stylów powinien zostać umieszczony w nawiasach okrągłych. Przed poleceniem `import` musi zostać umieszczony znak `@`, natomiast na końcu adresu arkusza stylów **średnik**.

Oprócz importowania arkusza można między znacznikami `<style>` i `</style>` umieszczać inne deklaracje stylów. Polecenia importowania arkusza muszą występować na początku arkusza stylów, przed właściwymi regułami CSS.

2.1.5. Kaskadowość arkuszy stylów

Zdarza się, że w dokumencie są umieszczone odwołania do kilku zewnętrznych arkuszy stylów lub na stronie używane są zewnętrzny arkusz stylów, deklaracja stylów w nagłówku strony oraz style lokalne. Wtedy może się pojawić konflikt dotyczący sposobu formatowania tego samego elementu w różnych arkuszach stylów. Zawsze pierwszeństwo mają style, które są umieszczone bliżej formatowanego elementu. Kaskadowość arkuszy stylów polega na ścisłym określeniu priorytetów stylów i przestrzeganiu zasad formatowania zgodnie z priorytetami. Ważność stylów została ustalona w następujący sposób:

1. Styl lokalny.
2. Rozciąganie stylu.
3. Wydzielone bloki.
4. Wewnętrzny arkusz stylów.
5. Zewnętrzny arkusz stylów.
6. Import stylów do zewnętrznego arkusza.
7. Atrybuty definiowane w dokumencie HTML.

Styl z numerem 1 ma najwyższy priorytet, a styl z numerem 7 — najniższy. Styl z najwyższym priorytetem ma pierwszeństwo w formatowaniu elementów strony. Style o niższym priorytecie formatują element tylko wtedy, gdy style o wyższym priorytecie nie dotyczą tego elementu. Należy zauważyć, że atrybuty definiowane dla elementu w dokumencie HTML mają najniższy priorytet, a więc style CSS definiowane w dowolny sposób zawsze są ważniejsze od tych atrybutów.

Postępując zgodnie z zasadami kaskadowości stylów, w dokumencie HTML polecenie dołączenia zewnętrznego arkusza stylów trzeba umieścić przed definicją wewnętrznego arkusza stylów.

Zasady kaskadowości można zmieniać. Do tego celu służy polecenie `!important` umieszczone w deklaracji stylu po wartości, której dotyczy. Takie umiejscowienie tego polecenia spowoduje, że właściwość zdefiniowana dla elementu będzie miała pierwszeństwo, nawet jeżeli ma niższy priorytet. Polecenie `!important` dotyczy tylko właściwości, dla której zostało użyte; pozostałe właściwości elementu będą obsługiwane zgodnie z ogólnymi zasadami kaskadowości stylów.

Przykład 2.8
Użycie polecenia `!important` w definicji stylu:

```
p {font-size: 20pt !important; color: green;}
```

Fragment kodu:

```
<p style= "font-size: 12pt; color: red"> Kaskadowe arkusze stylów</p>
```

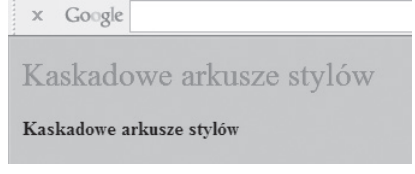
Przykład 2.9

```
<html>
<head>
<title>Arkusz stylów</title>
```

```
<style type="text/css">
body {background-color: #c0c0c0;}
p {font-size: 20pt !important; color: green;}
</style>
</head>
<body>
<p style= "font-size: 12pt; color: red"> Kaskadowe arkusze stylów</p>
<h2 style= "font-size: 12pt;"> Kaskadowe arkusze stylów</h2>
</body>
</html>
```

Na rysunku 2.3 został pokazany efekt działania polecenia `!important`. Dla znacznika `<p>` zastosowano kolor czcionki zdefiniowany za pomocą stylu lokalnego, natomiast rozmiar czcionki pochodzi z wewnętrznego arkusza stylów.

Rysunek 2.3. Efekt działania polecenia !important



Zadanie 2.4

W zewnętrznym arkuszu stylów został zdefiniowany znacznik `h2`:

```
h2 {color: blue; text-align: center; font-size: 12pt;}
```

W wewnętrznym arkuszu stylów również został zdefiniowany znacznik `h2`:

```
h2 {text-align: right; font-size: 24pt;}
```

Jeżeli strona ma powiązanie z zewnętrznym arkuszem stylów, to jakie właściwości będzie miał znacznik `h2`? Sprawdź swoją odpowiedź, pisząc kod przykładowej strony internetowej.

2.1.6. Dziedziczenie

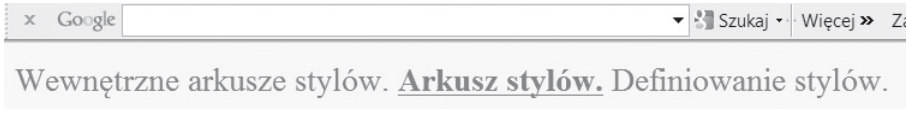
W języku HTML często występuje zagnieżdżanie jednego elementu wewnątrz drugiego. Jeżeli dla elementu nadrzędnego w arkuszach stylów zostały zdefiniowane właściwości, to w większości przypadków elementowi podrzędnemu zostaną przypisane te same właściwości, nawet jeżeli nie zostały dla niego zdefiniowane. Mechanizm ten nazywamy **dziedziczeniem**.

Przykład 2.10

```
<html>
<head>
<title> Wewnętrzny arkusz stylów</title>
<style type="text/css">
body {background-color: yellow;}
p {font-size: 20pt; color: green;}
```

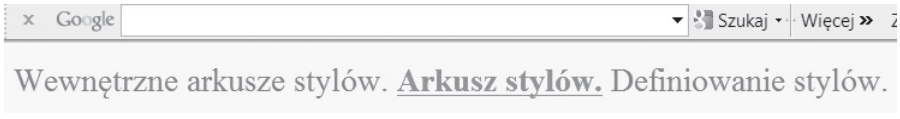
```
</style>
</head>
<body>
<p> Wewnętrzne arkusze stylów. <u><b>Arkusz stylów.</b></u> Definiowanie
stylów. </p>
</body>
</html>
```

W podanym przykładzie dla akapitu <p> został zdefiniowany kolor zielony. Wewnątrz akapitu wystąpiło pogrubienie i podkreślenie, którym nie nadano żadnego stylu, więc dziedziczą go one po akapicie (rysunek 2.4).



Rysunek 2.4. Dziedziczenie stylu dla tekstu „Arkusz stylów."

Na rysunku 2.5 widzimy efekt zdefiniowania atrybutu koloru dla fragmentu tekstu wewnątrz akapitu. Ten fragment nie odziedziczył stylu po akapicie.



Rysunek 2.5. Brak dziedziczenia stylu dla tekstu „Arkusz stylów."

Zadanie 2.5
Kod podany w przykładzie 2.10 zmodyfikuj tak, aby uzyskać wynik pokazany na rysunku 2.5.

2.2. Składnia języka CSS

Arkusz stylów jest zwykłym plikiem tekstowym i można go utworzyć za pomocą dowolnego edytora tekstu. Można również skorzystać ze specjalnego edytora CSS.

Edytory CSS przyspieszają i ułatwiają pracę, ponieważ automatyzują niektóre działania, na przykład można za ich pomocą automatycznie zdefiniować wielkość i kolor czcionki, kolor tła czy kolor odsyłaczy.

Plik będący zewnętrznym arkuszem stylów musi zostać zapisany z rozszerzeniem *.css*.

Jeżeli do arkusza stylów będą importowane inne arkusze stylów, to polecenia importowania `@import` muszą znaleźć się na początku tego arkusza w postaci:

```
@import url (ścieżka/plik.css);
```

W języku CSS parametry formatowania definiuje się za pomocą **reguł stylów**. Każda reguła odnosi się do określonego elementu i składa się z dwóch części: **selektora**

i **deklaracji**. Selektor określa, jakiego elementu ma dotyczyć formatowanie. Deklaracja jest umieszczana w nawiasach klamrowych `{ . . . }` i definiuje formatowanie elementu. Deklaracja musi zawierać przynajmniej jedną właściwość i przypisaną do niej wartość. Poza tym jednym ograniczeniem deklarowanych właściwości może być dowolna liczba. Deklaracja zawsze kończy się średnikiem.

Podstawowa składnia wygląda następująco:

```
selektor {właściwość: wartość;}
```

Powyższa składnia dotyczy stylów wewnętrznych, definiowanych w części nagłówkowej dokumentu HTML, oraz stylów zewnętrznych. Natomiast dla stylów lokalnych składnia ma postać:

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

W definicji stylu można grupować selektory i przypisywać im te same wartości:

```
selektor1, selektor2, selektor3 {właściwość: wartość;}
```

Przykład 2.11

```
p, h3, h4 {font-family: calibri; color: blue;}
```

Do dokumentu HTML można dołączyć arkusz stylów zapisany w pliku *arkusz.css* z podaną przykładową zawartością.

Przykład 2.12

```
body {background-color: #DFDFDF; }
p {font-color: blue; font-size: 5}
```

Jeżeli dokument HTML i plik z arkuszem stylów zostaną umieszczone w odpowiednich lokalizacjach i do dokumentu HTML przy użyciu elementu `<link>` zostanie dołączony zewnętrzny plik CSS, to po otwarciu dokumentu zdefiniowane w arkuszu stylów ustawienia dotyczące tła strony oraz wyglądu tekstu będą widoczne na tej stronie.

W pliku CSS można umieszczać komentarze. Komentarze zaczynają się znakami `/*`, a kończą się znakami `*/`.

Przykład 2.13

```
/*To jest komentarz*/
body {background-color: #DFDFDF; }/*tutaj też został dodany komentarz*/
p {color: blue; font-size: 5; font-family: courier new; }
```

2.2.1. Wartości i jednostki

Każda właściwość używana do definiowania selektora zawiera zbiór dopuszczalnych wartości oraz dopuszczalne rodzaje wartości.

Mogą to być:

Liczby — całkowite i rzeczywiste. Liczba całkowita składa się z cyfr od `0` do `9`. Liczba rzeczywista może zawierać część całkowitą i część ułamkową. Część ułamkowa zapisana jest po kropce.

Procenty. Wartość procentowa jest zawsze określana względem innej wartości. Może zostać użyta na przykład do określenia wielkości bloków. Podana w procentach wielkość bloku będzie się zmieniała wraz ze zmianą rozmiaru bloku, który go zawiera (rozmiar ten zwykle zależy od rozmiaru okna przeglądarki).

Słowa kluczowe. Nie zawsze wartości wyrażone za pomocą słów kluczowych są poprawnie interpretowane przez przeglądarki.

Jednostki długości. Mogą być zapisywane jako wartości bezwzględne i wartości względne.

Wartości bezwzględne: `in` (cal, 1 in = 2,54 cm), `cm` (centymetr), `mm` (milimetr), `pt` (punkty, 1 cal = 72 punkty), `pc` (pica, 1 pc = 12 pt), `px` (piksele, 1 px = 0,75 pt). Jednostki bezwzględne są rzadko stosowane. Należy ich używać wtedy, gdy chcemy, aby wybrany element miał taki sam rozmiar niezależnie od rozdzielczości monitora.

Wartości względne: `em` (1 em jest wielkością rozmiaru czcionki; wartość 0,1 em jest najmniejszą wartością, którą przeglądarki są w stanie rozpoznać), `ex` (wysokość małej litery *x* w użytej czcionce). Jednostki względne są stosowane, gdy odwołujemy się do rozmiaru innego elementu.

URL — np. `url(zdjecia/obraz.jpg)`. Błędem jest wstawienie spacji przed nawiasem otwierającym.

W języku HTML przyjmuje się, że wartości liczbowe bez podanych jednostek mają jednostkę `px`, natomiast w języku CSS brak jednostki jest traktowany jako błąd. Niektóre przeglądarki ignorują ten błąd i przyjmują jednostkę `px`. Gdy podana wartość jest równa 0, jednostka nie ma znaczenia.

Kolory. Stosuje się je na takich samych zasadach jak w języku HTML. W wersji CSS 2.2 oprócz 16 podstawowych kolorów został zdefiniowany dodatkowy kolor podstawowy `orange` (`#FFA500`) — pomarańczowy. W CSS 3 zostało zdefiniowanych dodatkowo 130 kolorów rozszerzonych.

Przykłady podawania wartości kolorów w CSS 3:

1. *predefiniowane* — podawanie kolorów w postaci ich nazw.
```
p {background-color: orange; }
```
2. *#RRGGBB* — podawanie wartości każdej składowej kolorów podstawowych (`RR` — czerwony, `GG` — zielony, `BB` — niebieski). Wszystkie liczby są dwucyfrowe i zapisane w systemie szesnastkowym, np. `#FF00CC`.
```
p {background-color: #ff0000; }
```
3. `#`*RGB* — jako składowe kolorów trzeba podawać liczby jednocyfrowe w systemie szesnastkowym, np. `#F0C`.
```
p {background-color: #f00; }
```
4. `rgb (`*R*`, `*G*`, `*B*`)` — można podać oddzielnie każdą składową koloru w systemie dziesiętnym.
```
p {background-color: rgb (255,0,0); }
```

5. `rgb (R%, G%, B%)` — można podać oddzielnie każdą składową koloru w postaci wartości procentowej.
6. *28 kolorów systemowych* — kolory pochodzą z systemu operacyjnego (kolor pulpitu, okien, menu). Gdy użytkownik zmieni kolory w swoim systemie, zmienią się również kolory zdefiniowane tą metodą.
7. `hsl (H, S%, L%)` — odmienny od RGB sposób opisu kolorów. `H` (*Hue*) określa barwę, `S` (*Saturation*) — nasycenie, `L` (*Lightness*) — jasność. Kolor powstaje przez wybranie określonej barwy (`H`), a następnie sterowanie nasyceniem (`S`) i jasnością (`L`) aż do uzyskania pożądanego efektu.

```
p {background-color: hsl (120,65%,75%); }
```

8. `rgba (R, G, B, A)`, `rgba (R%, G%, B%, A%)`, `hsla (H, S%, L%, A)` — uzyskiwanie koloru z przezroczystością. `A` (*alpha*) określa wartość przezroczystości z zakresu od `0` (całkowita przezroczystość) do `1` (brak przezroczystości).

```
p {background-color: rgba (255,0,0,0.5); }
p {background-color: hsla (120,65%,75%,0.3); }
```

2.3. Selektory

Selektorem może być dowolny element języka HTML, dla którego chcemy zdefiniować parametry formatowania. W zależności od tego, w jaki sposób odwołujemy się w definicji reguły do formatowanych elementów, wyróżniamy następujące rodzaje selektorów:

- selektory elementów,
- selektory atrybutów,
- selektory specjalne,
- selektory pseudoklas,
- selektory pseudoelementów.

2.3.1. Selektory elementów

Składnia dla selektorów elementów to podana już podstawowa składnia języka CSS:

```
selektor {właściwość: wartość;}
```

Selektor typu

Jest to podstawowy typ selektora. Służy do definiowania formatowania znaczników występujących na stronie (np.: `p`, `h3`, `table`, `img`).

Przykład 2.14

```
p {color: blue; font-size: 4pt;}
h3 {color: green; font-size: 24pt;}
```

Selektor uniwersalny

Selektor uniwersalny (inaczej: ogólny) to selektor pasujący do wszystkich znaczników. Jest oznaczany gwiazdką `*`.

Zamiast grupować elementy, np.:

```
p, h1, h2, h3, h4, table {font-family: courier new; color: green;}
```

można użyć selektora uniwersalnego:

```
* {font-family: courier new; color: green;}
```

Selektor potomka

Przy użyciu selektora potomka można formatować elementy, które są zawarte wewnątrz innych znaczników, czyli leżą niżej w hierarchii drzewa dokumentu.

Jeżeli w dokumencie HTML wewnątrz znacznika znajduje się inny znacznik, np. wewnątrz znacznika <h2> zostały umieszczone znaczniki <u> oraz <i>:

```
<h2> Wewnętrzne arkusze stylów. <u><i>Definiowanie stylów. </i></u>Arkusz
stylów. </h2>
```

to definicja stylu dla znacznika <i> będzie wyglądać następująco:

```
h2 i { font-size: 20pt; color: blue; }
```

Przykład 2.15

```
<html>
<head>
<title> Selektor potomka</title>
<style type="text/css">
body {background-color: grey;}
h2 i {font-size: 20pt; color: blue;}
</style>
</head>
<body>
<h2> Wewnętrzne arkusze stylów. <u><i> Definiowanie stylów. </i></u>
Arkusz stylów.</h2>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.6.

Rysunek 2.6. Selektor potomka

Jak widać, potomek nie musi być umieszczony bezpośrednio w znaczniku, którego jest potomkiem. W powyższym przykładzie w znaczniku <h2> znajduje się znacznik <u>, a dopiero w nim znacznik <i>, którego dotyczy definiowany styl.

Selektor potomka jest często wykorzystywany do definiowania odsyłaczy występujących wewnątrz bloku.

Selektor dziecka

Selektor dziecka służy do definiowania formatowania elementów, które znajdują się o jeden rząd niżej w hierarchii drzewa dokumentu. Ma on postać:

```
rodzic > dziecko {właściwość: wartość;}
```

gdzie symbol > oznacza bezpośredni związek między elementami.

Znacznik będący dzieckiem musi wystąpić bezpośrednio wewnątrz znacznika nadrzędnego.

Przykład 2.16

```
<html>
<head>
<title> Selektor dziecka</title>
<style type="text/css">
body {background-color: #c0c0c0;}
p>u {font-size: 20pt; color: yellow;}
</style>
</head>
<body>
<p> Wewnętrzne arkusze stylów. <u><i>Definiowanie stylów.</i></u> Arkusz
stylów.</p>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.7.

Rysunek 2.7. Selektor dziecka

Kolor *żółty* i wielkość liter `20pt` zostały nadane tylko tekstom otoczonym znacznikiem `<u>` i umieszczonym bezpośrednio w akapicie (znacznik `<p>`).

Selektor braci

Dla elementów znajdujących się w tym samym rzędzie hierarchii można zdefiniować selektor braci. Umożliwia on nadanie drugiemu z sąsiadujących elementów zdefiniowanych atrybutów formatowania. Selektor braci przyjmuje postać:

```
brat1 + brat2 {właściwość: wartość;}
```

gdzie symbol + oznacza połączenie elementów.

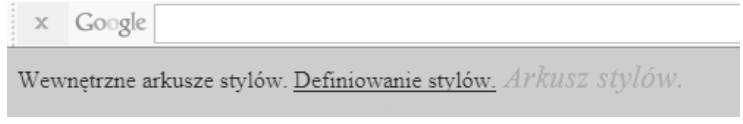
Przykład 2.17

```
<html>
<head>
<title> Selektor braci</title>
<style type="text/css">
```

```
body {background-color: #c0c0c0;}
u + i {font-size: 16pt; color: red;}
</style>
</head>
<body>
<p> Wewnętrzne arkusze stylów. <u>Definiowanie stylów.</u><i> Arkusz
stylów.</i> </p>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.8.

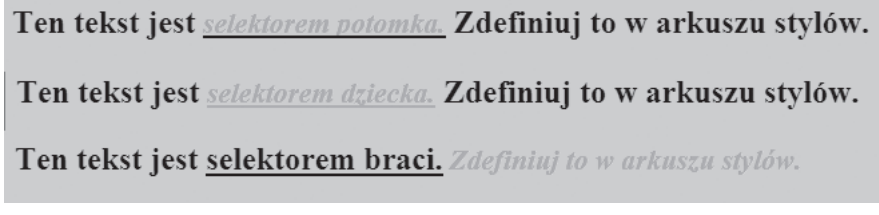
Rysunek 2.8. Selektor braci



W powyższym przykładzie znaczniki `<u>` oraz `<i>` znajdują się w tym samym rzędzie hierarchii, dlatego można dla nich ustawić selektor braci. Znacznik `<i>` występuje bezpośrednio po znaczniku `<u>` i tylko dla tekstu otoczonego tym znacznikiem jest realizowane formatowanie zdefiniowane dla selektora braci.

Zadanie 2.6

Wykorzystując arkusze stylów oraz selektory potomka, dziecka i braci, zdefiniuj style, które pozwolą na wyświetlenie tekstów sformatowanych tak jak na rysunku 2.9. Cały tekst powinien zostać zdefiniowany przy użyciu znacznika `<h2>`, natomiast wyróżnione fragmenty — za pomocą znaczników `<i>` oraz `<u>`.



Rysunek 2.9. Efekt użycia selektorów potomka (pierwszy wiersz), dziecka (drugi wiersz) i braci (trzeci wiersz)

2.3.2. Selektory atrybutów

W języku CSS można formatować znaczniki na podstawie posiadanych przez nie atrybutów. Na przykład jeżeli dla znacznika `<p>` (akapit) zostanie zdefiniowany atrybut `align` (wyrównanie), to inaczej będzie wyświetlany akapit wyrównany do lewej, a inaczej akapit wyśrodkowany.

Postać selektora atrybutów:

```
selektor [atrybut="wartość atrybutu"] {właściwość: wartość;}
```

Przykład 2.18

```
p [align="center"] {font-size: 16pt; color: yellow;}
```

Prosty selektor atrybutu

Prosty selektor atrybutu jest wykorzystywany dla elementów o nadanym określonym atrybucie, którego wartość nie ma znaczenia. Ma on postać:

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

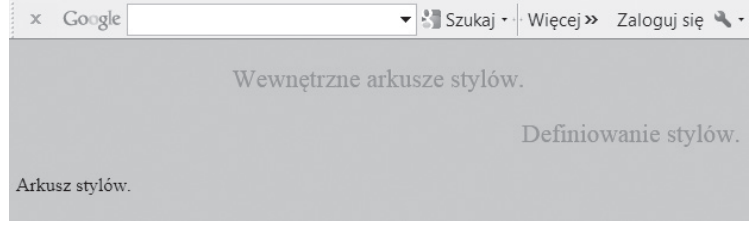
lub

```
[atrybut] {właściwość: wartość;}
```

Przykład 2.19

```
<html>
<head>
<title> Selektor atrybutu</title>
<style type="text/css">
body {background-color: #c0c0c0;}
p[align]{font-size: 16pt; color: red;}
</style>
</head>
<body>
<p align="center">Wewnętrzne arkusze stylów.</p>
<p align="right">Definiowanie stylów.</p>
<p> Arkusz stylów.</p>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.10.



Rysunek 2.10. Prosty selektor atrybutu

W powyższym przykładzie dla akapitu pierwszego i drugiego został zdefiniowany atrybut `align` o różnych wartościach. Niezależnie od wartości tego atrybutu akapity zostały sformatowane zgodnie z definicją stylu dla znacznika `<p>`. Ponieważ trzeci akapit nie ma zdefiniowanego atrybutu `align`, nie jest formatowany.

Selektor atrybutu o określonej wartości

Ten rodzaj selektora określa formatowanie, gdy atrybut ma określoną wartość. Ma on postać:

```
selektor [atrybut="wartość"] {właściwość: wartość;}
```

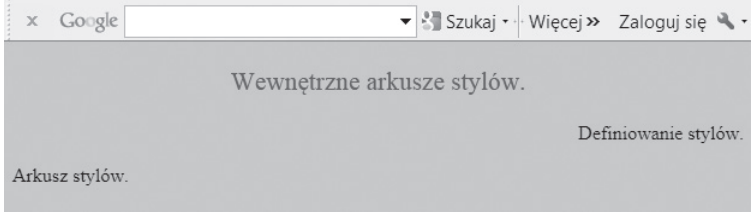
lub

```
[atrybut="wartość"] {właściwość: wartość;}
```

Przykład 2.20

```
<html>
<head>
<title> Selektor atrybutu</title>
<style type="text/css">
body {background-color: #c0c0c0;}
p[align="center"]{font-size: 16pt; color: blue;}
</style>
</head>
<body>
<p align="center" > Wewnętrzne arkusze stylów.</p>
<p align="right">Definiowanie stylów.</p>
<p> Arkusz stylów.</p>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.11.



Rysunek 2.11. Selektor atrybutu o określonej wartości

W tym przykładzie tylko pierwszy akapit został sformatowany zgodnie z definicją stylów, ponieważ tylko tu został zdefiniowany atrybut `align` z wartością `center`.

Selektor atrybutu zawierającego określony wyraz

Ten rodzaj selektora może zostać wykorzystany, gdy wartość atrybutu składa się z kilku wyrazów. Wystarczy, że w wartości atrybutu wystąpi podany wyraz, aby dany element został odpowiednio sformatowany. Postać selektora to:

```
selektor [atrybut~="wyraz"] {właściwość: wartość;}
```

lub

```
[atrybut~="wyraz"] {właściwość: wartość;}
```

Przykład 2.21

```
<html>
<head>
<title> Selektor atrybutu</title>
<style type="text/css">
body {background-color: #c0c0c0;}
p[title~="jest"]{font-size: 16pt; color: green;}
</style>
```

```
</head>
<body>
<p title="To jest mój styl" > Wewnętrzne arkusze stylów.</p>
<p title="To jest mój akapit">Definiowanie stylów.</p>
<p title="Uwaga na style"> Arkusz stylów.</p>
</body>
</html>
```

Zadanie 2.7

Wykorzystując arkusze stylów oraz selektory atrybutów, zdefiniuj style, które pozwolą na wyświetlenie tekstów umieszczanych w akapicie w różnych kolorach w zależności od wyrównania tych tekstów na stronie (np. dla tekstów wyrównanych do lewej będzie to kolor niebieski, dla tekstów wyśrodkowanych kolor zielony, a dla tekstów wyrównanych do prawej kolor czerwony). Niech dla tekstów umieszczonych w znaczniku `<h2>` kolory będą różne w zależności od zastosowanej wielkości czcionki (np. czcionka o rozmiarze `16pt` — kolor żółty, czcionka o rozmiarze `20pt` — kolor granatowy).

2.3.3. Selektory specjalne

Selektor klasy

Selektory pozwalają na nadanie określonych atrybutów takim samym elementom występującym na stronie. Czasami jednak chcemy zastosować pewien styl formatowania do jednej grupy elementów, a do drugiej inny. Możemy wtedy dla każdej grupy utworzyć klasę o dowolnej nazwie i zdefiniować styl dla selektora klasy.

Selektor klasy ma postać:

```
selektor.klasa { właściwość: wartość; }
```

gdzie *klasa* to dowolna nazwa.

Nazwa klasy musi spełniać kilka warunków:

- Musi zaczynać się od kropki.
- Mogą być w niej użyte litery, cyfry, znak podkreślenia, łącznik.
- Po kropce musi zostać wpisana litera.
- Rozróżniane są małe i wielkie litery.

Przykład 2.22

```
p.tekst1 {color: red; font-size: 24pt;}
p.tekst2 {color: green; font-size: 20pt;}
p.tekst3 {font-family: courier; color: blue; font-size: 16pt;}
```

Odwołanie do klasy w dokumencie HTML ma postać:

```
<znacznik class="nazwa_klasy"> .... </znacznik>
```

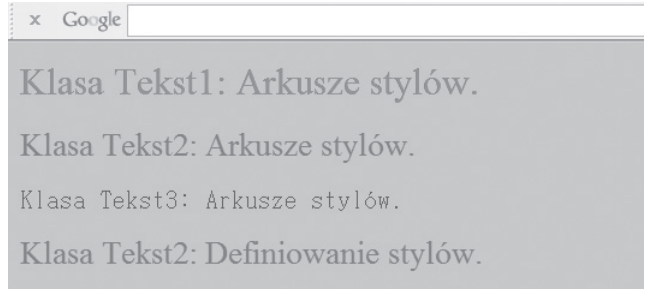
W odwołaniu nazwa klasy jest wartością atrybutu `class`. W nazwie klasy podawanej jako wartość atrybutu nie wstawia się kropki.

Przykład 2.23

```
<html>
<head>
<title> Selektor atrybutu</title>
<style type="text/css">
body {background-color: #c0c0c0;}
p.tekst1 {color: red; font-size: 24pt;}
p.tekst2 {color: green; font-size: 20pt;}
p.tekst3 {font-family: courier; color: blue; font-size: 16pt;}
</style>
</head>
<body>
<p class="tekst1"> Klasa Tekst1: Arkusze stylów.</p>
<p class="tekst2"> Klasa Tekst2: Arkusze stylów.</p>
<p class="tekst3"> Klasa Tekst3: Arkusze stylów.</p>
<p class="tekst2"> Klasa Tekst2: Definiowanie stylów.</p>
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.12.

Rysunek 2.12. Selektor klasy



Zdarza się, że definiując klasę, nie określamy, jakich znaczników będzie ona dotyczyła. Tak zdefiniowana klasa może zostać użyta do sformatowania dowolnego elementu bez względu na typ znacznika. Ma ona postać:

```
.klasa { właściwość: wartość; }
```

Każdy element, któremu nadamy klasę o podanej nazwie, zostanie sformatowany zgodnie z definicją stylu dla klasy.

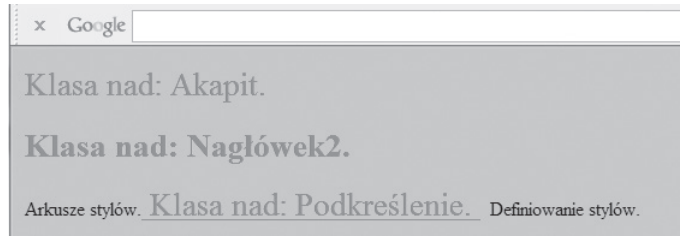
Przykład 2.24

```
<html>
<head>
<title> Selektor atrybutu</title>
<style type="text/css">
body {background-color: #c0c0c0;}
.nad {color: red; font-size: 20pt;}
</style>
```

```
</head>
<body>
<p class="nad"> Klasa nad: Akapit.</p>
<h2 class="nad"> Klasa nad: Nagłówek2.</h2>
Arkusze stylów.<u class="nad"> Klasa nad: Podkreślenie. </u>
Definiowanie stylów.
</body>
</html>
```

Wynik interpretacji kodu został pokazany na rysunku 2.13.

Rysunek 2.13. Efekt zastosowania klasy do formatowania różnych rodzajów znaczników



Deklaracja klasy jest przydatna zwłaszcza wtedy, gdy w różnych miejscach witryny mają występować elementy o takich samych parametrach formatowania, lecz nie możemy (lub nie chcemy) przy ich definiowaniu posługiwać się selektorem typu.

Selektor identyfikatora

Selektor identyfikatora jest stosowany, gdy chcemy nadać określone atrybuty formatowania elementowi, który ma przypisany jednoznaczny identyfikator. Selektor identyfikatora ma postać:

```
selektor#identyfikator {właściwość: wartość;}
```

lub

```
#identyfikator {właściwość: wartość;}
```

Deklaracja identyfikatora musi rozpoczynać się znakiem #.

Przykład 2.25

```
h3#naglo3 {color: yellow;}
```

Odwołanie do identyfikatora w kodzie ma postać:

```
<h3 id="naglo3"> Styl nagłówkowy</h3>
```

2.3.4. Pseudoklasy

W języku CSS style są dodawane do elementów lub grup elementów na podstawie ich nazw, atrybutów lub zawartości.

Jest też inna możliwość dodawania stylu. Element może nabywać styl lub tracić go w związku z działaniem użytkownika lub zmieniać się w zależności od umiejscowienia. Przykładem takiego zachowania są linki (*odsyłacze*), które zmieniają swój wygląd po

najechaniu na nie kursorem myszy, ich kliknięciu lub otwarciu. Do takiej dynamicznej zmiany stylu elementu służą **pseudoklasy**.

Odsyłacze (linki) mają właściwości, które określają działanie użytkownika. Są to:

- `:active` — link odwiedzany, strona jest aktualnie wczytana;
- `:link` — link nieaktywny, nie został przez użytkownika odwiedzony;
- `:visited` — link odwiedzony, strona była otwierana;
- `:hover` — link gotowy do kliknięcia, kursor myszy ustawiony nad linkiem.

Przy użyciu specjalnych selektorów dla odsyłaczy można definiować style, które będą nadawane odsyłaczom w zależności od ich bieżącego stanu.

Odsyłacz podstawowy (pseudoklasa :link)
Nadaje atrybuty formatowania wszystkim jeszcze nieodwiedzonym odnośnikom.

```
a:link {właściwość: wartość;}
```

Przy deklarowaniu klasy można ustawić atrybuty formatowania dla wybranych odsyłaczy, które nie były jeszcze odwiedzane.

```
a.nazwa_klasy:link {właściwość: wartość;}
```

Przykład 2.26

```
a:link {color: green; background: yellow;}
a.nowa:link {color: orange;}
```

Odsyłacz odwiedzony (pseudoklasa :visited)
Nadaje atrybuty formatowania odsyłaczowi, jeśli ten był odwiedzony, a informacja o tym została umieszczona w pamięci przeglądarki.

```
a:visited {właściwość: wartość;}
```

Z deklaracją klasy ustawiane są atrybuty formatowania dla odsyłaczy wybranej klasy, które były otwierane.

```
a.nazwa_klasy:visited {właściwość: wartość;}
```

Przykład 2.27

```
a:visited {color: red;}
a.po_nowa:visited {color: red;}
```

Wskazanie kursorem myszy (pseudoklasa :hover)
Nadaje atrybuty formatowania odsyłaczowi, gdy mysz znajduje się nad nim, ale nie aktywuje go.

```
a:hover {właściwość: wartość;}
```

Z deklaracją klasy ustawiane są atrybuty formatowania dla odsyłaczy wybranej klasy, które są w danej chwili wskazane myszą.

```
a.nazwa_klasy:hover {właściwość: wartość;}
```

Przykład 2.28

```
a:hover {color: yellow;}
a.po_nowa:hover {color: yellow;}
```

UWAGA

Pseudoklasa `:hover` może być używana do definiowania stylów elementów innych niż odsyłacze.

Przykład 2.29

```
<html>
<head>
<title> Pseudoklasa :hover</title>
<style type="text/css">
body {background-color: #c0c0c0;}
.blok {height: 25px; width: 200px; background: aqua; color: black; }
.blok:hover { background: yellow; color: red; }
</style>
</head>
<body>
<div class="blok"> Strony internetowe</div>
</body>
</html>
```

Przykład pokazuje zastosowanie pseudoklasy `:hover` dla znacznika `<div>`.

Odsyłacz aktywny (pseudoklasa :active)

Nadaje atrybuty formatowania odsyłaczowi, który w danej chwili jest aktywny, np. gdy użytkownik naciśnie i przytrzyma przycisk myszy.

```
a:active {właściwość: wartość;}
```

Z deklaracją klasy ustawiane są atrybuty formatowania dla odsyłaczy wybranej klasy, które są w danej chwili aktywne.

```
a.nazwa_klasy:active {właściwość: wartość;}
```

Przykład 2.30

```
a:active {color: blue;}
a.po_nowa:active {color: blue;}
```

Kolejność deklarowania w arkuszu stylów pseudoklas przypisanych do odsyłaczy ma znaczenie dla ich prawidłowego działania.

Oto poprawna kolejność deklarowania pseudoklas:

```
a:link {właściwość: wartość; }
a:visited {właściwość: wartość; }
a:hover {właściwość: wartość; }
a:active {właściwość: wartość; }
```

Przykład 2.31

```
<html>
<head>
<title> Pseudoklasa odnośnik</title>
```

```
<style type="text/css">
body {background-color: #c0c0c0;}
a:link {color: green; }
a:visited {color: red; }
a:hover {color: yellow; }
a:active {color: blue; }
</style>
</head>
<body>
<p> Link do strony wydawnictwa: <a href="http://helion.pl/">HELION</a></p>
</body>
</html>
```

Pseudoklasa :focus

Nadaje atrybuty formatowania odsyłaczowi (wcześniej wybranemu) lub polu formularza, na którym został ustawiony kursor, na przykład gdy odsyłacz został wybrany za pośrednictwem klawisza *Tab* lub w polu formularza znalazł się kursor.

```
selektor :focus {właściwość: wartość; }
```

Zadanie 2.8

Wykorzystując arkusze stylów, zdefiniuj pseudoklasy, które umożliwią dynamiczną zmianę stylu linków znajdujących się na przykładowej stronie internetowej. Link podstawowy powinien mieć kolor czerwony, odwiedzany — kolor zielony, wskazany przez kursor myszy — kolor pomarańczowy, a link aktywny — kolor żółty.

2.3.5. Selektory pseudoelementów

W języku HTML nie ma mechanizmów dostępu do takich elementów strony jak pierwsza litera lub pierwsza linia akapitu bez otaczania ich znacznikami. Do formatowania tych specjalnych elementów można wykorzystać **pseudoelementy** dostępne w języku CSS.

Pierwsza linia (:first-line)

Pseudoelement `:first-line` nadaje określone formatowanie wszystkim pierwszym liniom znacznika, do którego odnosi się selektor.

```
selektor :first-line {właściwość: wartość; }
```

Selektorem może być dowolny znacznik języka HTML.

Przykład 2.32

```
p :first-line { color: red; font-size: 16pt;}
```

Pierwsza litera (:first-letter)

Pseudoelement `:first-letter` nadaje odrębne formatowanie pierwszej literze np. akapitu. Służy głównie do poprawienia wyglądu tekstu poprzez zaprojektowanie ozdobnej pierwszej litery (inicjału).

```
selektor :first-letter {właściwość: wartość; }
```

Przykład 2.33

```
p :first-letter { font-family: arial; font-size: 24pt; color: blue; }
```