# CSS. Kaskadowe arkusze stylów.

## Przewodnik encyklopedyczny


Autor: Eric A. Meyer
ISBN: 83-7197-520-1
Tytuł oryginału: Cascading Style Sheets. The Definitive Guide

Jest to najnowsza książka z serii przewodników encyklopedycznych O'Reilly o tworzeniu stron internetowych, w skład której wchodzą również takie pozycje, jak: „HTML: The Definitive Guide", „JavaScript: The Definitive Guide", „Dynamic HTML: The Definitive Guide", „Web Design in a Nutshell"
Kaskadowe arkusze stylów (CSS) z pewnością zaznaczą swoje istnienie w Internecie. Przy dobrych implementacjach w Internet Explorerze 5.0 i Operze 3.6 i obsłudze CSS1 w przeglądarce Netscape Mozilla, CSS szybko stanie się użytecznym, niezawodnym i mocnym narzędziem dla autorów stron WWW.
CSS jest zaakceptowaną przez W3C metodą wzbogacania wyglądu witryn. Książka zawiera pełny, szczegółowy przegląd aspektów CSS1 i pozycjonowania CSS, a także przedstawia pokrótce CSS2. Dokładnie zbadano każdą właściwość i opisano jej współdziałanie z innymi właściwościami; podano także informacje o tym, jak uniknąć częstych pomyłek interpretacyjnych.
Jest to pierwsza pozycja, w której opisano obsługę CSS w poszczególnych przeglądarkach, nie poprzestając na ogólnych i teoretycznych wyjaśnieniach. Zarówno dla autorów stron, jak i programistów książka ta jest pełnym przewodnikiem efektywnego stosowania CSS.
Książka jest także adresowana do początkujących autorów stron, którzy już rozpoczęli naukę znaczników i atrybutów HTML. Pozwoli im nie powtarzać błędów poprzedników i osiągnąć wymierne korzyści z poprawnego implementowania CSS.

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

# *Spis treści*

# 4

# Właściwości tekstu

Chociaż nasz wysiłek podczas projektowania stron WWW może skupiać się głównie na wybieraniu właściwych kolorów i nadawaniu stronie najlepszego wyglądu, to jednak najwięcej czasu poświęcamy martwiąc się o wygląd i rozmieszczenie tekstu. Obawy te spowodowały wprowadzenie do HTML, znaczników takich jak `<FONT>` i `<CENTER>`, które pozwalają na pewną formę kontroli tego, jak tekst będzie wyglądał i jak będzie rozmieszczony.

Właśnie dlatego duża część CSS dotyczy właściwości wpływających na tekst w taki czy inny sposób. Właściwości w CSS1 podzielone są na dwie części: *właściwości tekstu* i *właściwości czcionek*. Ten rozdział poświęcony jest pierwszej grupie. W następnym rozdziale podejmiemy temat czcionek — są one na swój sposób dosyć skomplikowane, więc zasługują na osobne omówienie.

## Manipulowanie tekstem

Można zastanawiać się, jaka jest różnica pomiędzy tekstem a czcionkami. Ujmując to prosto — tekst jest treścią. Czcionka użyta do jego wyświetlenia jest tylko jednym ze sposobów na zmianę wyglądu tekstu. Zanim zajmiemy się w czcionkami, powinniśmy omówić kilka prostszych sposobów wpływania na wygląd tekstu. Poza tym pewne rzeczy, którymi zajmiemy się tutaj, przydadzą się przy omawianiu właściwości czcionek, więc większy sens ma omówienie właściwości tekstu w pierwszej kolejności.

Dzięki właściwościom tekstu można wpłynąć na pozycję tekstu względem reszty linijki, ustawić indeks górny, podkreślenie i zmienić wielkość liter. Można nawet w pewnym stopniu symulować użycie klawisza tabulatora z maszyny do pisania.

### Akapity i wyrównanie poziome (horyzontalne)

Najlepiej zacząć od tego, jak można wpłynąć na poziome umiejscowienie tekstu w linii. Nie jest to tym samym, czym jest rzeczywiste pozycjonowanie, w którym tekst ustawiany jest względem samej strony. O tych właściwościach myślmy raczej jak o sposobach wpłynięcia na rozmieszczenie linii tekstu, tak jak robimy to pisząc raport lub tworząc biuletyn.

## Wcięcia

Przy formatowaniu tekstu w Internecie jednym z najbardziej poszukiwanych efektów jest wcinanie pierwszej linii akapitu (kolejnym jest chęć wyeliminowania pustej linii pomiędzy akapitami, co omówimy w rozdziale 7.). Niektórzy autorzy stron rozwiązują ten problem poprzez wstawienie małego przezroczystego obrazka przed pierwszą literą akapitu (popychając w ten sposób tekst dalej). Inni używają zupełnie niestandardowego znacznika `SPACER`. Dzięki CSS istnieje lepszy sposób na uzyskanie podobnego efektu.

### text-indent

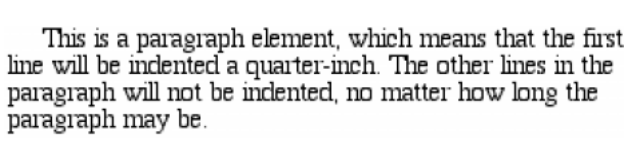
| | |
|---|---|
| **Wartości** | *<długość>* \| *<procenty>* |
| **Wartość domyślna** | 0 |
| **Dziedziczona** | tak |
| **Odnosi się do** | elementów blokowych |

*Uwaga*: wartości procentowe odnoszą się do szerokości elementu rodzica.

Każdy element używający wcięcia tekstu (`text-indent`) może mieć wciętą pierwszą linię o daną długość (nawet ujemną). Najczęstszym użyciem tej właściwości jest wcięcie pierwszej linii akapitów:

```
P {text-indent: 0.25in;}
```

Powyższa reguła mówi, że wcięcie pierwszej linii każdego akapitu będzie wynosiło ćwierć cala, tak jak to przedstawiono na rysunku 4.1.

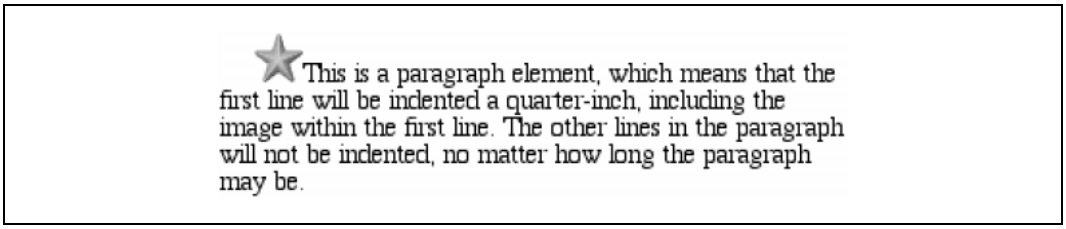



*Rysunek 4.1. Wcinanie tekstu*

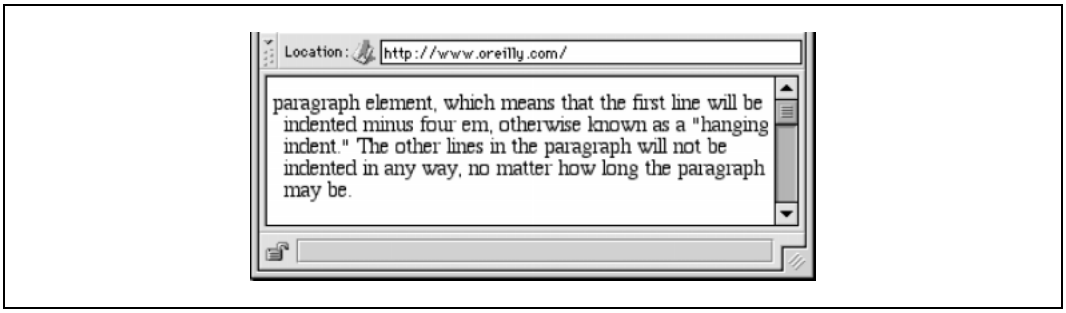
Właściwość `text-indent` może być nadana każdemu elementowi blokowemu, takiemu jak `PRE`, `H1` czy `BLOCKQUOTE`. Nie można nadać jej elementom stanowiącym część ciągu tekstu (*inline elements*), takim jak `STRONG` lub `A`, tak samo jak nie można używać jej w przypadku elementów podmienianych, takich jak obrazki (co oczywiście ma sens). Jeśli jednak w pierwszej linii elementu blokowego (np. akapitu) znajdzie się obrazek, zostanie on przesunięty razem z tekstem, tak jak to przedstawiono na rysunku 4.2.

Można zastosować ujemne wartości wcięcia tekstu, co może zostać użyte na wiele interesujących sposobów. Najbardziej popularnym jest „wiszące wcięcie", gdzie pierwsza linia jest wysunięta z lewej strony akapitu, tak jak to przedstawiono na rysunku 4.3.

```
P {text-indent: -4em;}
```



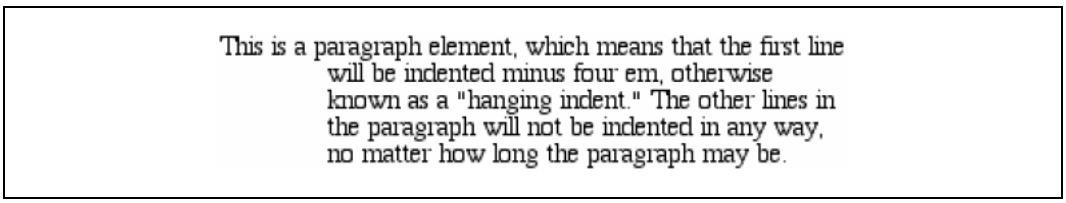

*Rysunek 4.2. Wcięcia tekstu i obrazków*




*Rysunek 4.3. Ujemne wcięcie tekstu*

Jak widać na rysunku 4.3, istnieje niebezpieczeństwo przy ustawianiu ujemnych wartości dla wcięcia tekstu. Pierwsze dwa słowa („This is") zostały obcięte przez lewy brzeg okna przeglądarki. Aby przy wyświetlaniu zapobiec takim problemom, doradza się użycie marginesu, który pomieści ujemne wcięcie, tak jak zrobiono to w kolejnym przykładzie (rysunek 4.4):

```
P {text-indent: -4em; padding-left: 4em;}
```

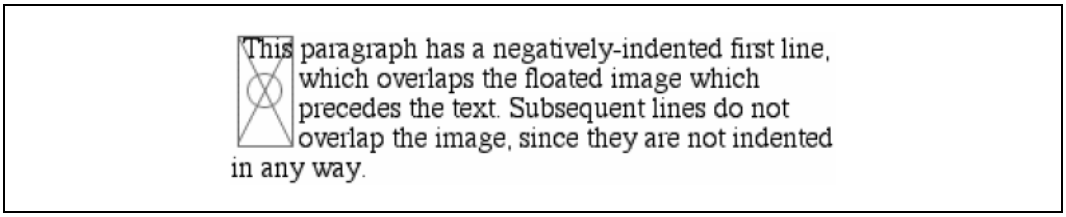



*Rysunek 4.4. Wyjaśnienie ujemnych wcięć*

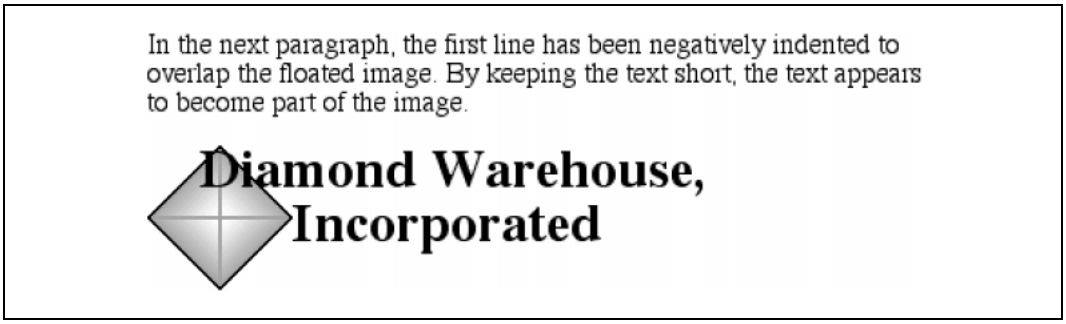
Ujemne wcięcie czasami może zostać użyte w korzystny sposób. Rozważmy następujący przykład, zademonstrowany na rysunku 4.5. Dodajmy obrazek do tej mieszanki:

```
P.hang {text-indent: -30px;}

<P CLASS="hang"><IMG SRC="floater.gif" WIDTH="30px" HEIGHT="60px"
ALIGN="left" ALT="Floated">This paragraph has a negatively indented
first line, which overlaps the floated image which precedes the text.
Subsequent lines do not overlap the image, since they are not
indented in any way.</P>
```

Przy użyciu tej prostej techniki można zrealizować wiele ciekawych projektów. Rysunek 4.6 przedstawia przykład tekstu, w którym pierwsza linia została wcięta o `-40px`.



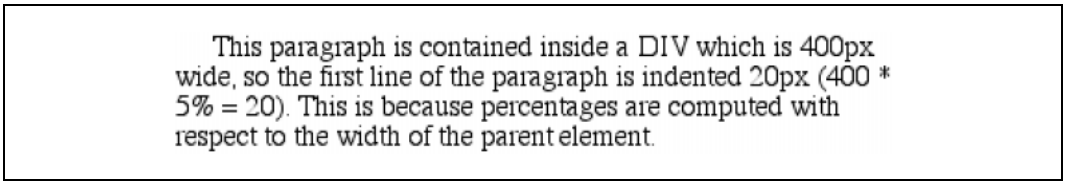

*Rysunek 4.5. Pływający obrazek i ujemne wcięcie tekstu*




*Rysunek 4.6. Ujemne wcięcia i pływające obrazki*

Do określenia głębokości wcięcia pierwszej linii akapitu można stosować dowolne jednostki. Dodatkowo można używać wartości procentowych. W takim przypadku procenty odnoszą się do szerokości elementu rodzica wcinanego elementu. Jeśli zatem ustawimy wartość wcięcia na `5%`, to pierwsza linia odnośnego elementu zostanie wcięta na 5% szerokości elementu rodzica, tak jak to przedstawiono na rysunku 4.7.

```
DIV {width: 400px;}
P {text-indent: 5%;}

<DIV>
<P>This paragraph is contained inside a DIV which is 400px wide,
so the first line of the paragraph is indented 20px (400 * 5% = 20).
This is because percentages are computed with respect to the width
of the parent element.</P>
</DIV>
```

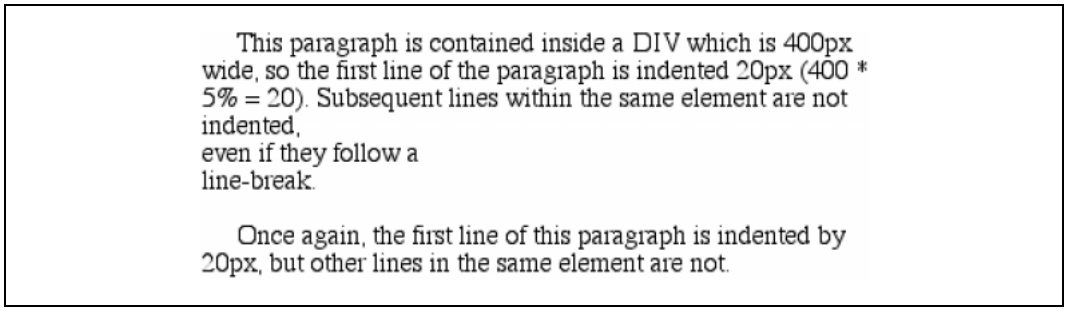



*Rysunek 4.7. Wcięcie tekstu określone za pomocą procentów*

Wcięcia będą nadane jedynie pierwszej linii elementu, nawet jeśli wstawimy znaczniki końca linii (`BR`). Przedstawia to rysunek 4.8:

```
DIV {width: 400px;}
P {text-indent: 5%;}
<DIV>
```

```
<P>This paragraph is contained inside a DIV which is 400px wide,
so the first line of the paragraph is indented 20px (400 * 5% = 20).
Subsequent lines within the same element are not indented,<BR>
even if they follow a<BR>
line-break.</P>
<P>Once again, the first line of this paragraph is indented by 2s0px,
but other lines in the same element are not.</P>
</DIV>
```

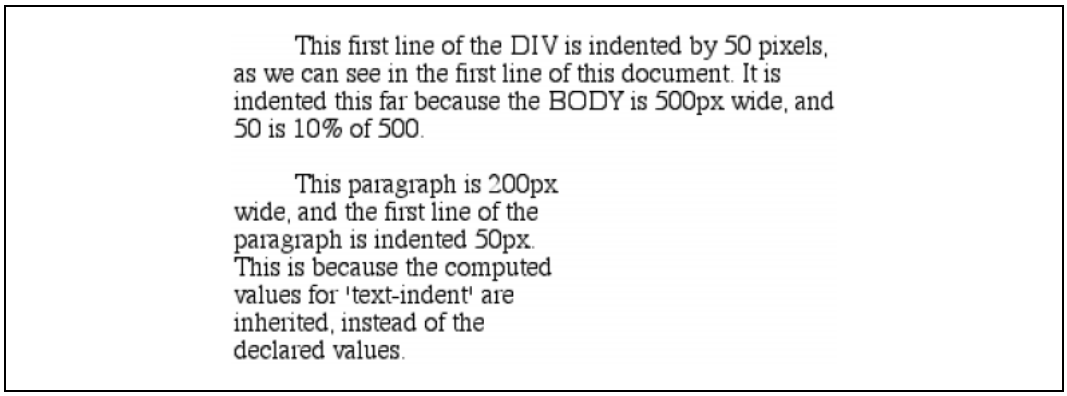



*Rysunek 4.8. Wcięcia i znaki końca linii*

Ciekawą cechą `text-indent` jest to, że jest właściwością dziedziczoną, ale dziedziczona jest wyliczona wartość, a nie wartość zadeklarowana. Weźmy za przykład następujący kod:

```
BODY {width: 500px;}
DIV {width: 500px; text-indent:10%;}
P {width: 200px;}

<DIV>
This first line of the DIV is indented by 50 pixels.
<P>This paragraph is 200px wide, and the first line of the paragraph
is indented 50px.  This is because computed values for 'text-indent'
are inherited, instead of the declared values.</P>
</DIV>
```

Pomimo że akapit ma szerokość tylko 200 pikseli, jego pierwsza linia jest wcięta o 50 pikseli (tak jak na rysunku 4.9), czyli odziedziczoną wartość `text-indent`.




*Rysunek 4.9. Odziedziczone wcięcia*

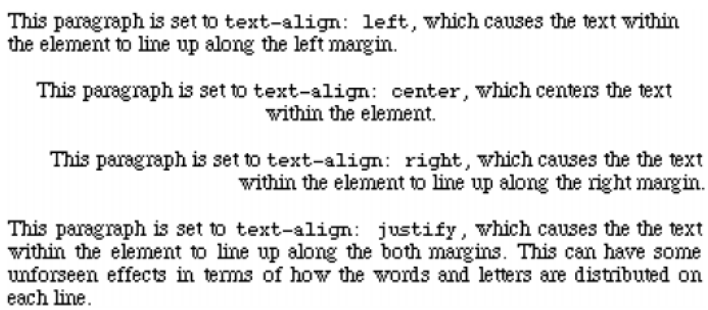
### *Wyrównywanie tekstu*

Jeszcze bardziej podstawową właściwością niż `text-indent` jest właściwość `text-align` (wyrównanie tekstu), która określa, w jaki sposób linie tekstu w elemencie mają być wyrównane względem siebie. Istnieją cztery wartości; pierwsze trzy są dość proste, ale czwarta mieści w sobie parę zawiłości.

**text-align**

| | |
|---|---|
| **Wartości** | `left\|center\|right\|justify` |
| **Wartość domyślna** | zależy od przeglądarki |
| **Dziedziczona** | tak |
| **Odnosi się do** | elementów blokowych |

Najszybszym sposobem na zrozumienie tego, jak te wartości działają, jest prześledzenie rysunku 4.10.




*Rysunek 4.10. Działanie właściwości text-align*

Wyrównanie tekstu jest kolejną właściwością nadawaną wyłącznie elementom blokowym, takim jak akapity. Nie można wyśrodkować odnośnika w linii, w której się znajduje, bez wyrównania pozostałej części linii (na czym nam nie zależy).

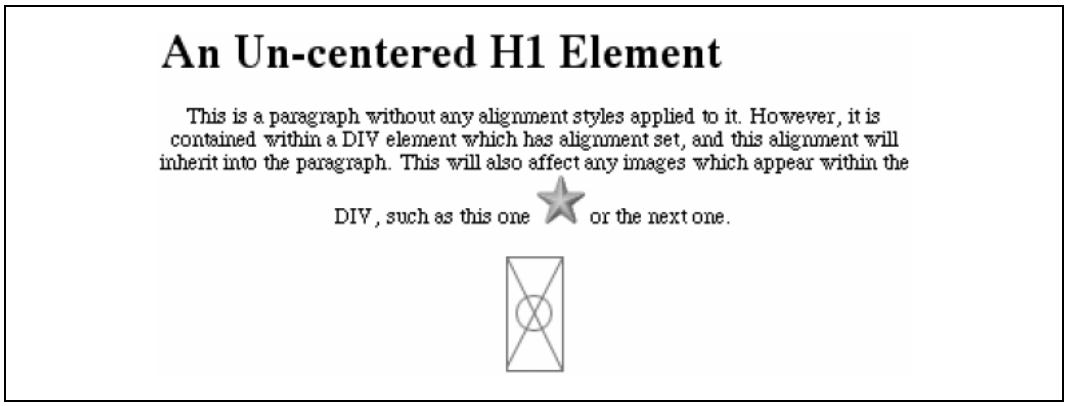
Reguła `text-align: center` może być użyta do zastąpienia operacji wykonywanych przez znaczniki `CENTER`, tak jak przedstawiono to na rysunku 4.11.

```
H1 {text-align: center;}
```


*Rysunek 4.11. Wyśrodkowanie tekstu przy użyciu właściwości text-align*

Można także spowodować wyśrodkowanie elementów z umieszczonym wewnątrz nich tekstem i rysunkami. Zadziała to dokładnie tak jak znacznik `CENTER`, w taki sposób, jaki przedstawia następujący kod i rysunek 4.12:

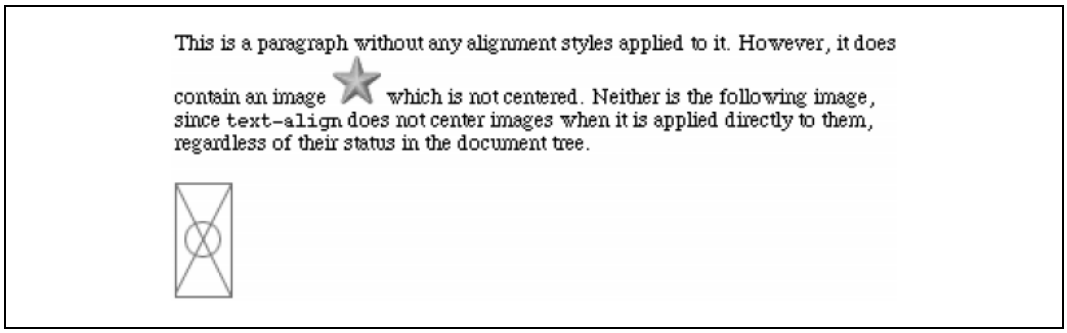
```
DIV.first {text-align: center;}

<H1>An Un-centered H1 Element</H1>
<DIV CLASS="first">
<P>
This is a paragraph without any alignment styles applied to it.
However, it is contained within a DIV element which has alignment
set, and this alignment will inherit into the paragraph.  This will
also affect any images which appear within the DIV, such as this one
<IMG SRC="star.gif"> or the next one.
</P>
<IMG SRC="floater.gif">
</DIV>
```




*Rysunek 4.12. Wyśrodkowanie właściwością text-align tekstu i wchodzących w ciąg tekstu rysunków*

Jeśli jednak chcemy wyśrodkować sam obrazek, `text-align` nie jest poprawnym sposobem. Mając poniższy kod uzyskamy rezultaty widoczne na rysunku 4.13:

```
IMG {text-align: center;}
```




*Rysunek 4.13. Właściwość text-align i rysunki blokowe*

Obrazek nie został wyśrodkowany, gdyżnie jest elementem blokowym. Jedynym sposobem na to, aby wyśrodkować obrazek przy użyciu `text-align` jest objęcie go elementem `DIV`, który ma ustawione wyśrodkowanie zawartości:

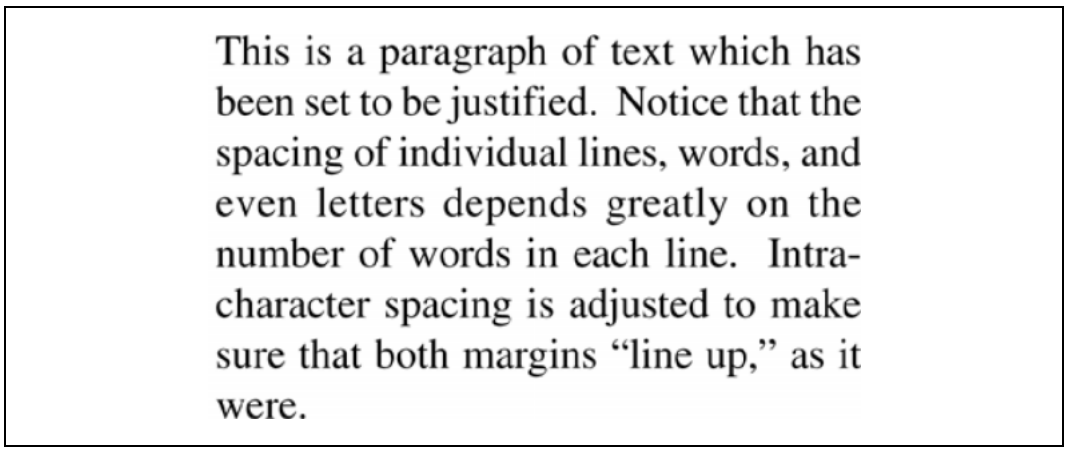
```
<DIV STYLE="text-align: center;">
   <IMG SRC="blyszczacy.gif" ALT="Błyszczący obiekt">
</DIV>
```

Jak widać na rysunku 4.14, taka deklaracja działa poprawnie.


*Rysunek 4.14. Zagmatwane wyśrodkowywanie rysunków*

Dla języków zachodnich (czytanych od lewej do prawej strony) wartością domyślną `text-align` jest `left` (z tekstem wyrównanym do lewego marginesu i „postrzępionym" prawym marginesem — inaczej znane jako „tekst z lewej do prawej"). Inne języki, takie jak arabski czy hebrajski, domyślnie będą wyrównane do prawej, ponieważ pisze się w nich z prawej do lewej strony. `center` daje oczekiwany efekt — powoduje wyśrodkowanie każdej linii wewnątrz elementu, któremu jest nadane.

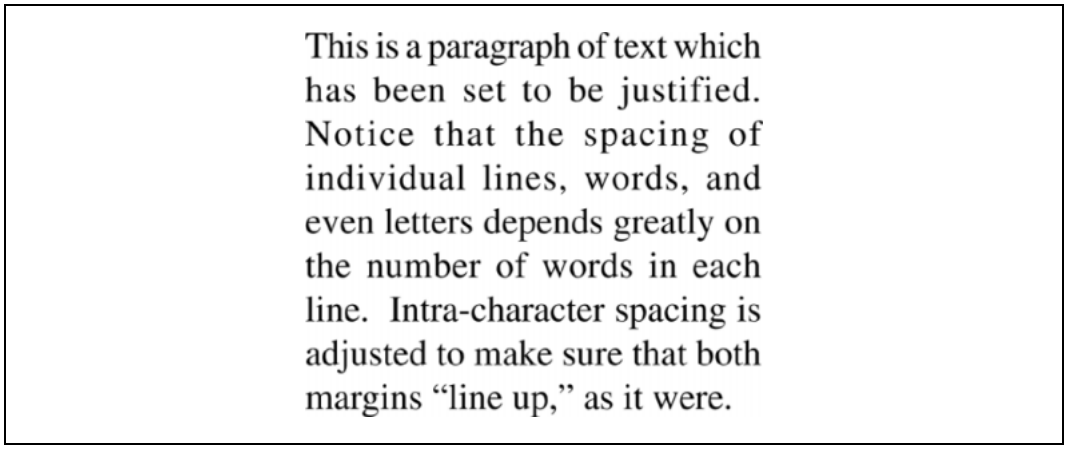
Przy `justify` (*justowanie*) należy rozważyć kilka spraw. Jak to przedstawia rysunek 4.15, wyjustowany tekst jest tak sformatowany, że końce każdej linii znajdują się na wewnętrznych brzegach elementu rodzica. Jest to osiągnięte poprzez zmianę odstępów pomiędzy słowami i literami w taki sposób, aby każda linia była dokładnie tej samej długości. Efekt ten spotykany jest często w świecie druku (np. książkach), ale przy CSS wchodzą w grę jeszcze inne zastosowania.




*Rysunek 4.15. Wyjustowany tekst*

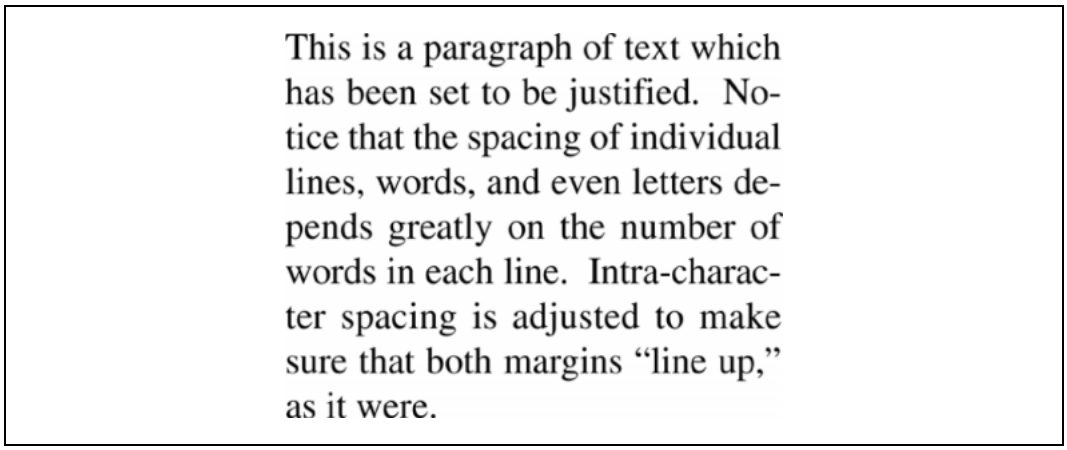
CSS nie określa sposobu rozciągania justowanego tekstu w celu wypełnienia przestrzeni pomiędzy lewym i prawym brzegiem elementu rodzica, dlatego niektóre przeglądarki mogą dodawać odstępy tylko pomiędzy słowami, a inne mogą dodawać odstępy między literami. Możliwe jest także to,

że niektóre przeglądarki zmniejszą odstępy w niektórych liniach i zagęszczą tekst bardziej niż zwykle. Wszystko to może wpłynąć na wygląd elementu i może nawet zmienić jego wysokość — zależy to od tego, na ile linii tekst zostanie podzielony przez przeglądarkę (rysunek 4.16).




*Rysunek 4.16. Możliwości justowania*

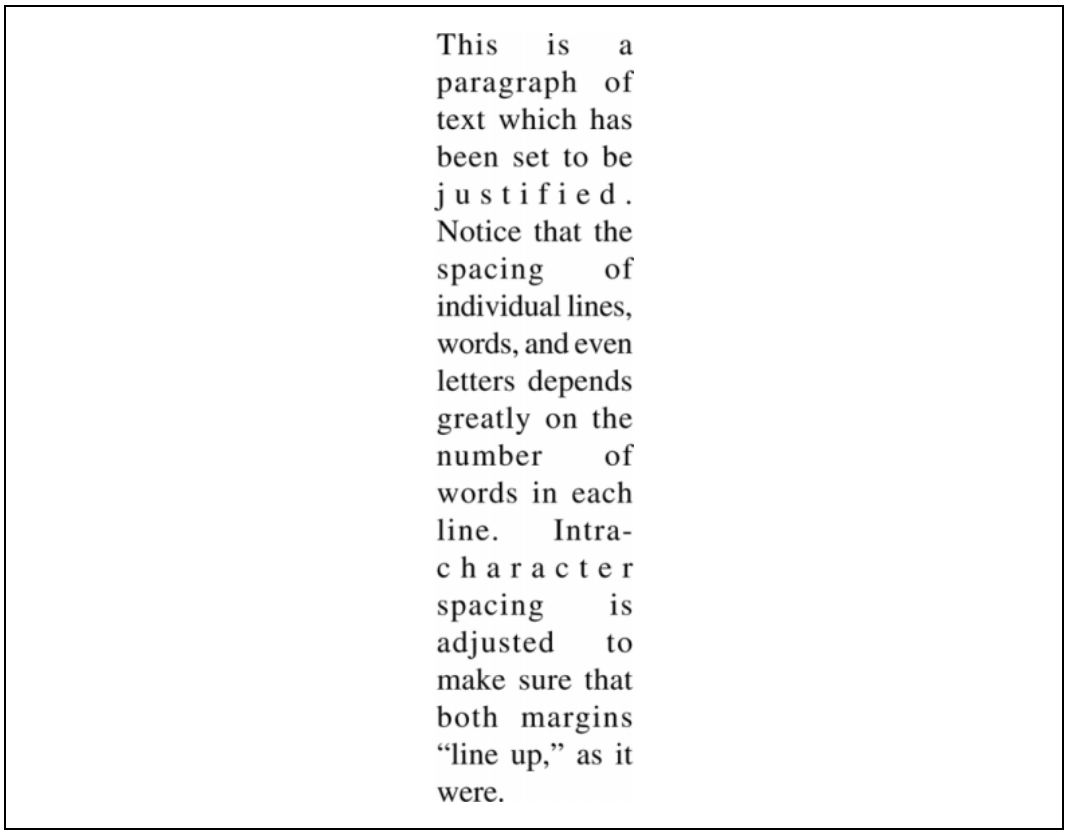
Pojawia się również inny problem — CSS przemilcza sposób dzielenia wyrazów. Dzielenie tekstu, czyli użycie łącznika do podzielenia wyrazu pomiędzy dwie linie, pozwala na zmniejszenie odstępów między wyrazami, poprawiając tym samym wygląd linii (rysunek 4.17).




*Rysunek 4.17. Justowanie z dzieleniem wyrazów*

Dzielenie wyrazów nie jest zdefiniowane w CSS, ponieważ różne języki mają różne zasady dzielenia wyrazów. Zamiast próbować improwizować tworząc zestaw reguł, który byłby zapewne i tak niekompletny, specyfikacja po prostu unika tego problemu. Pozwala to przeglądarkom na używanie własnych reguł dzielenia wyrazów i — z czasem — ulepszanie ich bez przeszkód w postaci specyfikacji CSS.

Skoro w CSS nie ma funkcji odpowiadającej za dzielenie wyrazów, przeglądarki przeważnie nie będą wykonywać automatycznego dzielenia wyrazów. A zatem wyjustowany tekst będzie raczej mniej atrakcyjny w CSS niż w druku. Zwłaszcza wtedy, gdy elementy są tak wąskie, że w każdej linii jest tylko kilka słów, tak jak to przedstawiono na rysunku 4.18. Oczywiście można justować tekst, ale należy pamiętać o wadach tej metody.




*Rysunek 4.18. Justowanie w wąskim akapicie bez dzielenia wyrazów*

Prawie każda przeglądarka obsługująca CSS poradzi sobie z większością wartości `text-align`, jednak często zawodzi przy `justify`. Netscape Navigator 4 obsługuje justowanie, ale popełnia przy tym dość dużo błędów. Najczęściej zawodzi wewnątrz tabelek. Internet Explorer 4.x nie obsługuje justowania, ale Internet Explorer 5 i Opera 3.6 już je obsługują.

## *Jak postępować z odstępami*

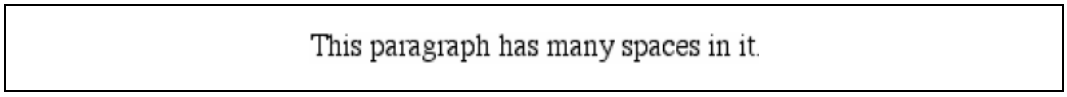
Aby zmienić tempo, porozmawiajmy o właściwości `white-space`, która może mieć ogromny wpływ na sposób wyświetlania tekstu.

## white-space

| | |
|---|---|
| **Wartości** | `pre | nowrap | normal` |
| **Wartość domyślna** | `normal` |
| **Dziedziczona** | nie |
| **Odnosi się do** | elementów blokowych |

Używając tej właściwości można wpłynąć na to, jak przeglądarka traktuje odstępy między słowami i liniami tekstu. HTML robi to w pewnym stopniu — zamienia każdy odstęp na jedną spację. A zatem następujący kod zostanie wyświetlony tak, jak to przedstawiono na rysunku 4.19 (z tylko jedną spacją pomiędzy każdym słowem):

```
<P>This    paragraph   has     many
    spaces        in it.</P>
```




*Rysunek 4.19. Wyświetlanie akapitu z wieloma spacjami*

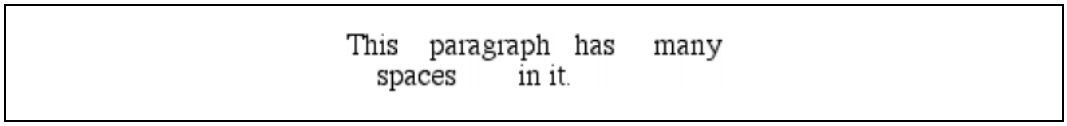
Takie zachowanie możesz wymusić następującą deklaracją:

```
P {white-space: normal;}
```

Reguła ta „powie" przeglądarce, że ma postępować tak, jak zawsze to robiły przeglądarki (powinna zignorować dodatkową pustą przestrzeń). Wszystkie zbyteczne spacje i znaki końca wiersza zostaną przez przeglądarkę pominięte.

Jeśli jednak zmienimy wartość `white-space` na `pre`, pusta przestrzeń będzie traktowana tak, jakby elementy były elementami `PRE`, w których pusta przestrzeń nie jest ignorowana, jak to przedstawiono na rysunku 4.20.

```
P {white-space: pre;}

<P>This    paragraph   has     many
    spaces        in it.</P>
```

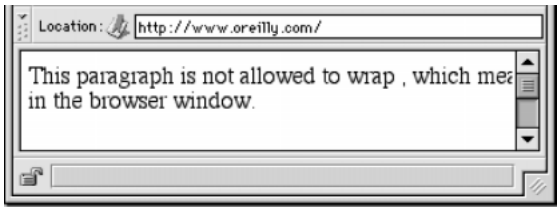



*Rysunek 4.20. Honorowanie spacji w kodzie*

Z właściwością `white-space` ustawioną na `pre`, przeglądarka uwzględni dodatkowe spacje i znaki końca wiersza. Pod tym względem — i tylko pod tym względem — każdy element może zachowywać się jak element `PRE`.

Istnieje jeszcze wartość `nowrap`, która chroni tekst wewnątrz elementu blokowego przed zawijaniem do nowej linii (za wyjątkiem użycia znaczników końca linii `<BR>`). Jest to podobne do ustawienia w komórce tabelki niezawijania wierszy poprzez `<TD NOWRAP>` z tym, że właściwość `white-space` może być nadana każdemu elementowi blokowemu, nie tylko tabelkom. Dlatego możemy uzyskać efekty, takie jak przedstawiono na rysunku 4.21:

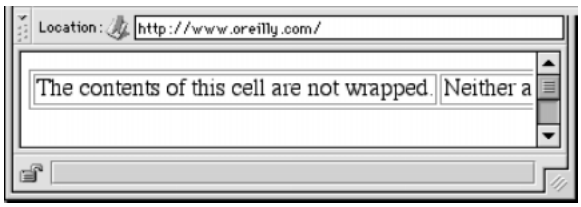
```
<P STYLE="white-space: nowrap;">This paragraph is not allowed to
wrap,which means that the only way to end a line is to insert a line-
break element.  If no such element is inserted, then the line will go
forever,forcing the user to scroll horizontally to read whatever
can't be initially displayed <BR>in the browser window.</P>
```




*Rysunek 4.21. Rezygnacja z zawijania wierszy przy użyciu właściwości white-space*

Właściwie `white-space` można użyć po to, aby zastąpić atrybut `nowrap` w komórkach tabelki, jak zademonstrowano poniżej (rysunek 4.22):

```
TD {white-space: nowrap;}

<TABLE><TR>
<TD>The contents of this cell are not wrapped.</TD>
<TD>Neither are the contents of this cell.</TD>
<TD>Nor this one, or any after it, or any other cell in this
table.</TD>
<TD>CSS prevents any wrapping from happening.</TD>
</TR></TABLE>
```




*Rysunek 4.22. Użycie właściwości white-space: nowrap w celu ominięcia zawijania wierszy w tabelce*

Chociaż `white-space` może wydawać się właściwością niesłychanie użyteczną, nie jest obsługiwana przez żadną przeglądarkę poza IE5 dla MacOS i pierwszymi próbnymi wersjami Netscape 6. W innych przeglądarkach nawet nie ma możliwości użycia atrybutu `nowrap` w komórkach tabelek, pomimo, że wydaje się to zachowaniem bardzo prostym w obsłudze.

## *Wysokość linii*

W przeciwieństwie do rozmiaru czcionki (zagadnienie to omówimy w rozdziale 5.) wysokość linii (`line-height`) odnosi się mniej więcej do dystansu występującego między bazowymi liniami pisma. Faktycznie właściwość ta ustala wielkość, o którą zwiększy się lub zmniejszy wysokość pudełka każdego elementu się. W najprostszych przypadkach jest to sposób na zwiększenie (lub zmniejszenie) pionowej przestrzeni występującej pomiędzy liniami tekstu, ale jest to myląco prosty sposób widzenia zasad działania właściwości `line-height`. Jeśli ktoś zna pakiety DTP, to można powiedzieć, że `line-height` kontroluje interlinię (ang. *leading*), będącą dodatkową przestrzenią pomiędzy liniami tekstu nad i pod polem czcionki. Różnica pomiędzy wartością `line-height` i rozmiarem czcionki to interlinia.

### line-height

| | |
|---|---|
| **Wartości** | *<długość>* \| *<procenty>* \| *<liczba>* \| `normal` |
| **Wartość domyślna** | `normal` |
| **Dziedziczona** | tak |
| **Odnosi się do** | wszystkich elementów |

*Uwaga*: wartości procentowe obliczane są względem rozmiaru czcionki elementu.

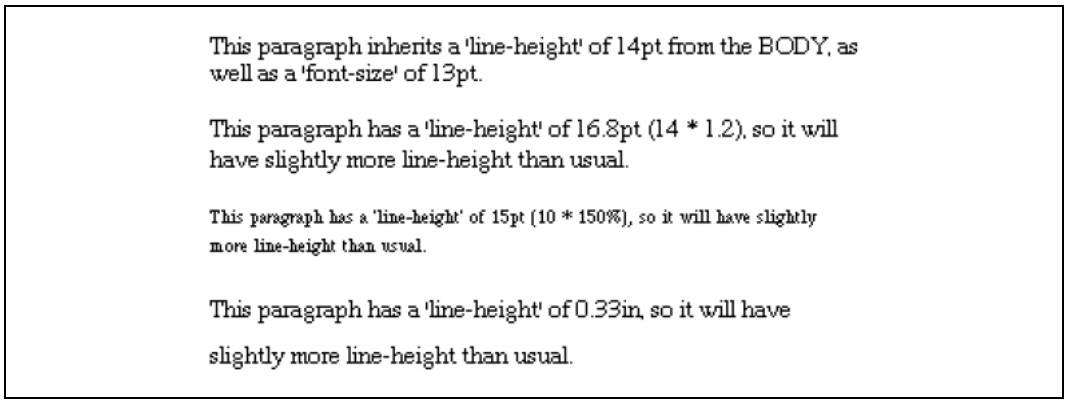
Mówiąc językiem technicznym — każdy element w linii generuje *obszar elementu* (ang. *content area*), który zależy od rozmiaru czcionki. Obszar elementu generuje także pudełko, które — przy braku innych czynników — powinno być dokładnie równe obszarowi elementu. Jednak interlinia tworzona przez `line-height` może zwiększyć lub zmniejszyć wysokość pudełka. Wielkość interlinii dzieli się przez 2 i przesuwa górną i dolną granicę pudełka o otrzymaną wartość. W ten sposób otrzymujemy pudełko o nowych wymiarach. Na przykład jeśli obszar elementu ma wysokość 14 punktów, a `line-height` ustawione jest na 18 punktów, wtedy różnica (4 punkty) jest dzielona na pół i dodana na górę i dół pudełka, co zwiększa jego rozmiar do wysokości 18 punktów. Brzmi to jak okrężna droga prowadząca do opisania działania wysokości linii, ale istnieją realne przyczyny takiego opisu. W rozdziale 8. znajdziemy jeszcze bardziej szczegółowy opis.

Jeśli używamy domyślnej wartości `normal`, ilość przestrzeni rozciągającej się pomiędzy liniami będzie wynikać z ustawień przeglądarki. Zazwyczaj jest to liczba mieszcząca się między 1,0 a 1,2 rozmiaru czcionki, ale może być różna w różnych przeglądarkach.

Dozwolone długości inne niż `em` i `ex` są zwykłymi wymiarami długości (np. `0.2cm`). Występują tu te same problemy, co zawsze: nawet jeśli użyje się poprawnej wartości (np. `4cm`), przeglądarka (lub system operacyjny) może niepoprawnie odwzorować rzeczywiste wartości. A zatem wyświetlając dokument na ekranie monitora, linijka zapewne nie wskaże, że wysokość linii ma dokładnie cztery centymetry. Żeby dowiedzieć się czegoś więcej, przeczytaj rozdział 3.

Wartości procentowe (tak samo jak `em` i `ex`) są obliczane w stosunku do wielkości czcionki elementu. Poniższy kod powinien być względnie prosty, a rezultat widoczny jest na rysunku 4.23:

```
BODY {line-height: 14pt; font-size: 13pt;}
P.one {line-height: 1.2em;}
P.two {font-size: 10pt; line-height: 150%;}
P.three {line-height: 0.33in;}

<P>This paragraph inherits a 'line-height' of 14pt from the BODY,
as well as a 'font-size' of 13pt.</P>
<P CLASS="one">This paragraph has a 'line-height' of 16.8pt (14 *
1.2), so it will have slightly more line-height than usual.</P>
<P CLASS="two">This paragraph has a 'line-height' of 15pt (10 *
150%), so it will have slightly more line-height than usual.</P>
<P CLASS="three">This paragraph has a 'line-height' of 0.33in, so
it will have slightly more line-height than usual.</P>
```




*Rysunek 4.23. Proste obliczenia związane z właściwością line-height(wysokość linii)*

Wyniki stają się mniej przewidywalne, kiedy wartość wysokości linii jest dziedziczona z jednego elementu blokowego na inny. Wyliczona wartość wysokości linii jest dziedziczona bez względu na to, jaki rozmiar czcionki może być ustawiony dla potomka. Może się to okazać problemem, tak jak to przedstawiono na rysunku 4.24:

```
BODY {font-size: 10pt;}
DIV {line-height: 12pt;}
P {font-size: 18pt;}

<DIV>
<P>This paragraph's 'font-size' is 18pt, but the inherited 'line-
height' is only 12pt.  This may cause the lines of text to overlap
each other by a small amount.</P>
</DIV>
```

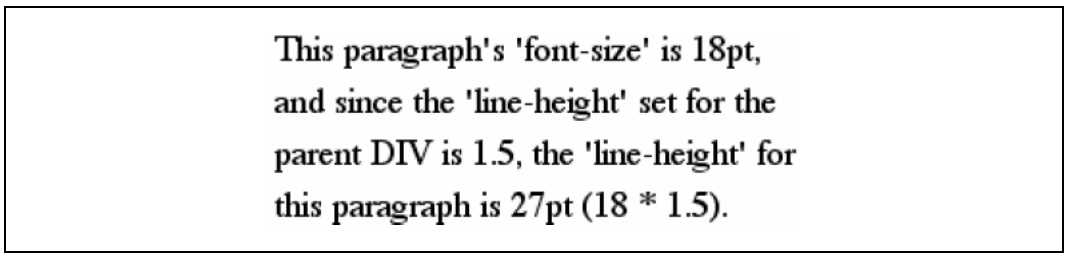

*Rysunek 4.24. Mała wysokość linii, duża wielkość czcionki — drobny problem*

Istnieją dwa rozwiązania. Można ustawiać wysokość linii dla każdego elementu, ale nie jest to rozwiązanie zbyt praktyczne tym bardziej, że może być wiele elementów wymagających takiego podejścia. Drugim rozwiązaniem jest ustawienie zwykłej liczby, która w rzeczywistości będzie stanowić współczynnik skalujący:

```
BODY {font-size: 10pt;}
DIV {line-height: 1.5;}
P {font-size: 18pt;}
```

Jeśli używamy liczby, dziedziczony jest współczynnik skalujący, a nie wyliczona wartość. Współczynnik nadany jest elementowi i wszystkim jego potomkom, a więc każdy element ma wysokość linii wyliczoną względem własnego rozmiaru czcionki, jak to przedstawiono na rysunku 4.25:

```
BODY {font-size: 10pt;}
DIV {line-height: 1.5;}
P {font-size: 18pt;}

<DIV>
<P>This paragraph's 'font-size' is 18pt, and since the 'line-height'
set for the parent DIV is 1.5, the 'line-height' for this paragraph
is 27pt (18 * 1.5).</P>
</DIV>
```




*Rysunek 4.25. Obejście problemów przy użyciu współczynnika skalującego*

Jak już pisałem wcześniej — chociaż wydaje się, że `line-height` rozdziela dodatkową przestrzeń nad i pod każdą linią tekstu — w rzeczywistości dodaje (lub odejmuje) pewną ilość na górze lub na dole obszaru elementu. To prowadzi do „dziwnych" rezultatów, gdy linia zawiera elementy z różną wielkością czcionki. Na razie jednak pozostańmy przy prostych przykładach. Załóżmy, że domyślna wielkość czcionki dla akapitu wynosi `12pt` i rozważmy następującą regułę:

```
P {line-height: 16pt;}
```

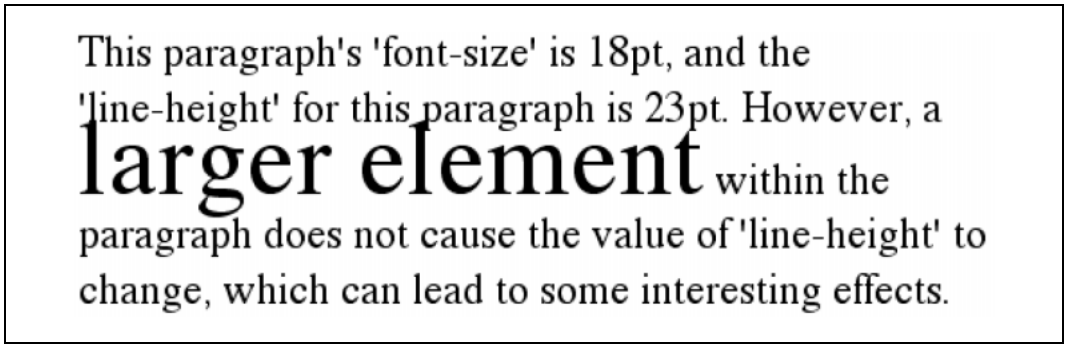
Skoro dziedziczona wysokość linii 12-punktowego tekstu wynosi 12 punktów, powyższa reguła oznajmiać będzie, że wokół każdej linii tekstu w akapicie będą dodatkowe 4 punkty przestrzeni. Te dodatkowe punkty, po podzieleniu na dwa, zostaną dodane nad i pod każda linię tekstu. Wynikiem rozdzielenia dodatkowej przestrzeni jest dystans pomiędzy liniami równy `16pt`.

Teraz przyjrzyjmy się trochę bardziej zawiłemu przypadkowi. Weźmy następujący przykład:

```
BODY {font-size: 10pt;}
P {font-size: 18pt; line-height: 23pt;}
BIG {font-size: 250%;}
```

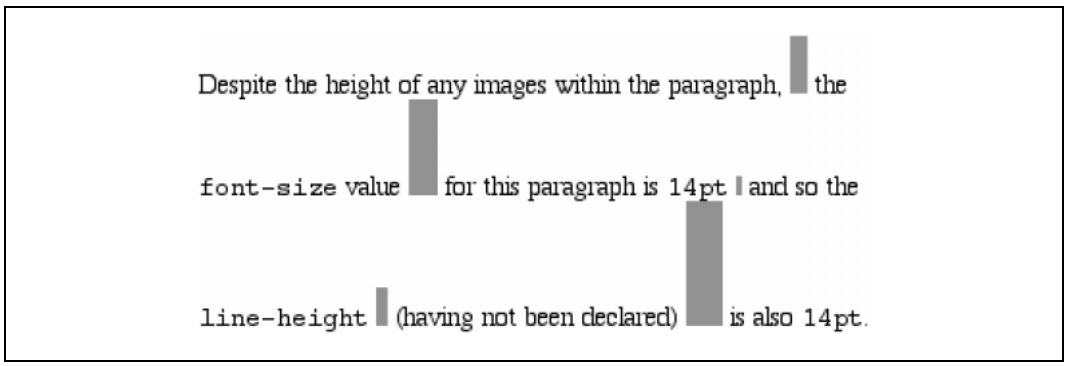
```
<P>This paragraph's 'font-size' is 18pt, and the 'line-height' for
this paragraph is 23pt. However, a <BIG>larger element</BIG> within
the paragraph does not cause the value of 'line-height' to change,
which can lead to some interesting effects.</P>
```

Efekt przedstawiony na rysunku 4.26 może wyglądać jak wynik błędu przeglądarki, ale tak nie jest. Jest dokładnie tym, co powinna wyświetlić przeglądarka.




*Rysunek 4.26. Wysokości linii a elementy wewnętrzne innego rozmiaru*

To jest jedyna dziwaczność wynikająca z użycia wysokości linii. Na rysunku 4.27 wartości `line-height` są dokładnie takie same dla każdej linii elementu (bez względu na to jak różne mogą się wydawać). Ten fakt zostanie omówiony już niedługo w podrozdziale —*Wyrównanie pionowe*.

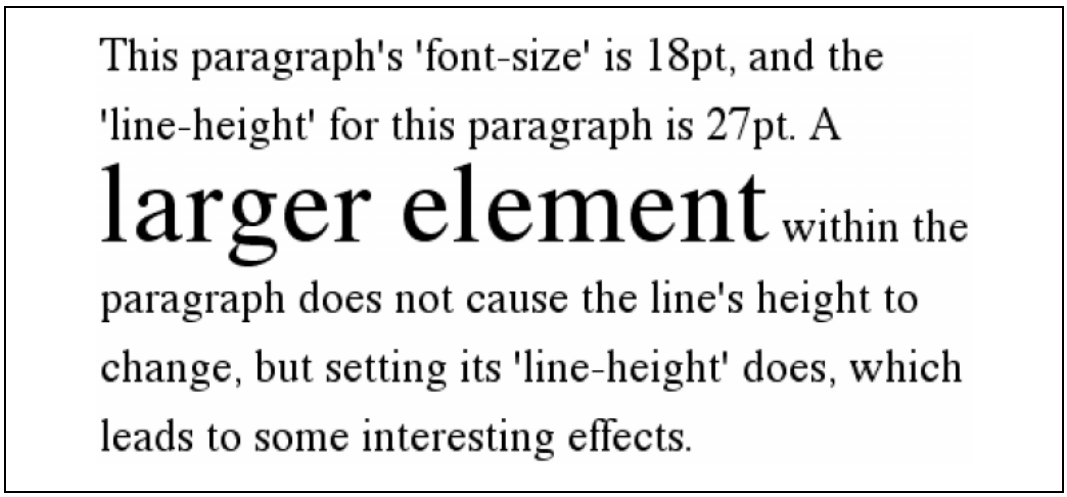



*Rysunek 4.27. Wysokie rysunki nie zmieniają wysokości linii*

Na rysunku 4.27 rzeczywista wartość wysokości linii jest dla każdej linii taka sama. Można wykorzystać fakt, że właściwość `line-height` może zostać ustawiona dla każdego elementu (nie wyłączając elementów wewnętrznych). Powróćmy do naszego poprzedniego przykładu i wprowadźmy do niego jedną małą zmianę poprzez dodanie `line-height` do stylów elementu `BIG` (zmienimy również wysokość linii dla `P` na `27pt`). Będzie to miało efekt widoczny na rysunku 4.28:

```
BODY {font-size: 10pt;}
P {font-size: 18pt; line-height: 27pt;}
BIG {font-size: 250%; line-height: 1em;}
```

```
<P>This paragraph's 'font-size' is 18pt, and the 'line-height' for
this paragraph is 27pt. A <BIG>larger element</BIG> within the
paragraph does not cause the line's height to change, but setting its
'line-height' does, which leads to some interesting effects.</P>
```




*Rysunek 4.28. Zmiana line-height dla elementu wewnętrznego*

Ustawienie wysokości linii na `1em` dla elementu `BIG` sprawi, że wysokość linii będzie taka sama jak wysokość tekstu elementu `BIG`. Da to taki sam efekt, jaki dawały obrazki: zwiększy się wysokość całej linii tekstu tak, by poprawnie wyświetlić element w niej zawarty. W tym przypadku efekt w dużym stopniu jest taki sam, ale zawdzięczamy go zwiększeniu wartości `line-height` wewnętrznego elementu `BIG` (w przeciwieństwie do optycznej zmiany wysokości linii ze względu na dopasowanie do rzeczywistego rozmiaru obrazka).

Powody takiego zachowania są raczej skomplikowane. Więcej informacji na temat sposobu formatowania elementów umieszczonych wewnątrz tekstu znajdziemy w rozdziale 8.

## *Z życia wzięte*

Jak wynika z poprzedniego podrozdziału, współczynnik skalujący jest najlepszym sposobem na uniknięcie problemów związanych z dziedziczeniem, które mogą pojawić się przy stosowaniu jednostek długości w wysokości linii. Dlatego może się wydawać, że użycie samej liczby jest zawsze lepsze. Niestety, niekoniecznie tak jest. Internet Explorer 3 zinterpretuje współczynnik skalujący jako ilość pikseli, a wynik będzie taki, jak to przedstawiono na rysunku 4.29.

*Rysunek 4.29. Współczynnik skalujący użyty z line-height oznacza duże kłopoty w Internet Explorer 3*

Zgodnie ze specyfikacją CSS przeglądarki mogą ustawić dowolną wartość, którą uważają za najlepszą dla domyślnego słowa kluczowego `normal`, ale sugerowany przedział zawiera się między

1,0 a 1,2 — w zależności od użytej czcionki i od urządzenia wyświetlającego. Większość przeglądarek używa wartości bliższej prawej granicy przedziału, ale oczywiście może to ulec zmianie wraz z pojawianiem się innych przeglądarek lub nowych wersji starszych przeglądarek.

## *Wyrównanie pionowe*

Wszystko wskazuje na to, że w pewnym stopniu znamy już pojęcie pionowego wyrównania tekstu. Jeśli kiedykolwiek używaliśmy elementów `SUP` i `SUB` (indeksu górnego i dolnego) lub — wyświetlając obrazek — posłużyliśmy się kodem `<IMG SRC="foo.gif" ALIGN="texttop">`, to można przyjąć, że mieliśmy już do czynienia z wyrównaniem pionowym. Właściwość CSS `vertical-align` (wyrównanie pionowe) pozwala na użycie wszystkich znanych nam już sposobów wyrównania stosowanych wewnątrz tekstu. Mamy też do wyboru kilka nowych możliwości.

Należy pamiętać o jednej rzeczy: `vertical-align` nie wpływa na wyrównanie zawartości komórek tabel lub tekstu znajdującego się wewnątrz elementów blokowych. CSS1 nie przewiduje duplikowania kodu, takiego jak `<TD valign="top">`. Zmienia się to w CSS2; więcej informacji znajduje się podrozdziale *Tabelki* rozdziału 10.

Sprawdźmy, czego możemy dokonać za pomocą `vertical-align` (wyrównanie pionowe). Właściwość ta nadawana jest tylko elementom wewnętrznym. Nie jest dziedziczona, chociaż obejmuje też elementy podmieniane, takie jak obrazki czy pola formularzy.

### vertical-align

| | |
|---|---|
| **Wartości** | `baseline` \| `sub` \| `super` \| `bottom` \| `text-bottom` \| `middle` \| `top` \| `text-top` \| *<procenty>* |
| **Wartość domyślna** | `baseline` |
| **Dziedziczona** | nie |
| **Odnosi się do** | elementów wewnętrznych |

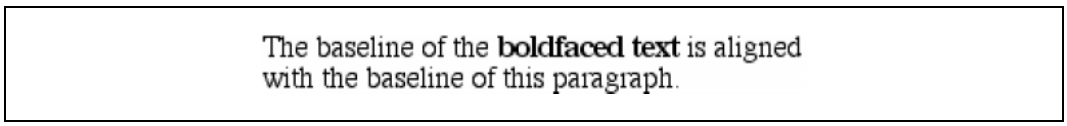
*Uwaga*: wartości procentowe odnoszą się do wysokości linii elementu.

Właściwość `vertical-align` akceptuje każde z ośmiu słów kluczowych lub wartości procentowe (ale nigdy nie robi tego jednocześnie). Słowa kluczowe to mieszanka znanych i nieznanych słów: `baseline` (wartość domyślna), `sub`, `super`, `bottom`, `text-bottom`, `middle`, `top` i `text-top`. Prześledzimy po kolei, jakie efekty daje stosowanie każdego z nich.

### *Wyrównanie do bazowej linii pisma*

Właściwość `vertical-align: baseline` zmusza element do wyrównania bazowych linii pisma rodzica i potomka, czyli w sumie odpowiada za to, co zawsze robią przeglądarki. Pokazuje to poniższy kod i rysunek 4.30:

```
B {vertical-align: baseline;}

<P>The baseline of the <B>boldfaced text</B> is aligned with the
baseline of this paragraph.</P>
```

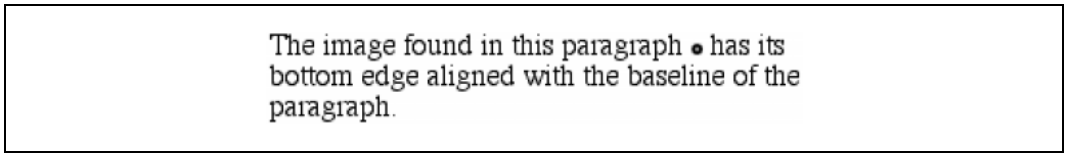



*Rysunek 4.30. Wyrównanie do bazowej linii pisma*

Oglądając rysunek 4.30 nie zobaczymy nic innego niż to czego się spodziewaliśmy.

Jeśli element, który wyrównujemy w pionie, nie posiada bazowej linii tekstu — gdy jest obrazkiem, polem formularza lub innym podmienianym elementem — to dolny brzeg elementu jest wyrównywany z bazową linią pisma rodzica, tak jak to przedstawiono na rysunku 4.31:

```
IMG {vertical-align: baseline;}

<P>The image found in this paragraph <IMG SRC="dot.gif" ALT="a dot">
has its bottom edge aligned with the baseline of the paragraph.</P>
```

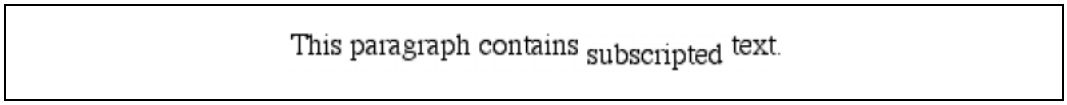



*Rysunek 4.31. Wyrównanie obrazka z bazową linią tekstu*

## *Indeks górny i indeks dolny*

Deklaracja `vertical-align: sub` powoduje, że element jest wyświetlony jako indeks dolny. Ogólnie oznacza to, że linia bazowa elementu (lub dolny brzeg w przypadku elementów podmienianych) jest położony niżej w stosunku do bazowej linii pisma elementu rodzica. Jednak wielkość przesunięcia nie jest określona w specyfikacji, dlatego może być inna w rożnych przeglądarkach. Jako że `sub` nie implikuje zmian w rozmiarze czcionki, nie powinien zmniejszać (lub powiększać) indeksowanego tekstu. Tekst w elemencie indeksu dolnego powinien być domyślnie tego samego rozmiaru co tekst elementu rodzica, tak jak to przedstawiono na rysunku 4.32.

```
SUB {vertical-align: sub;}

<P>This paragraph contains <SUB>subscripted</SUB> text.</P>
```

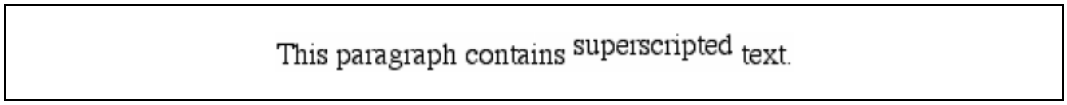



*Rysunek 4.32. Ustawienie indeksu dolnego*

Oczywiście można zmniejszyć tekst poprzez użycie deklaracji `font-size`, ale tego rozwiązania nie będziemy analizować w naszej książce.

Słowo `super` zachowuje się podobnie jak `sub`, ale w tym przypadku linia bazowa elementu (lub dolny brzeg elementu podmienianego) jest podnoszony w stosunku do bazowej linii pisma rodzica. Odcinek, o jaki tekst zostanie podniesiony, zależy od przeglądarki. Żadne zmiany w wielkości czcionki nie są wprowadzane. Zademonstrowano to na rysunku 4.33:

```
SUP {vertical-align: super;}

<P>This paragraph contains <SUP>superscripted</SUP> text.</P>
```

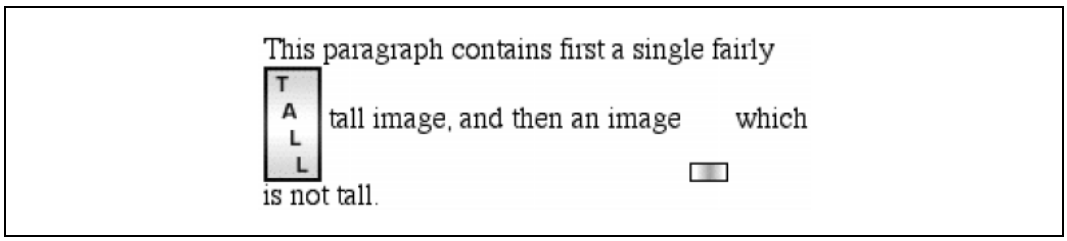



*Rysunek 4.33. Ustawienie indeksu górnego*

## *Na dno*

Działanie `vertical-align: bottom` wydaje się dosyć proste. Jest nim wyrównanie dolnych brzegów pudełka elementu i pudełka linii. Na przykład kod taki jak przedstawiono poniżej powinien dać efekt widoczny na rysunku 4.34:

```
IMG.feeder {vertical-align: bottom;}

<P>This paragraph contains first a single fairly <IMG SRC="tall.gif"
align="middle"
ALT="tall image"> which is tall, and then an image <IMG SRC="short.gif"
CLASS="feeder" ALT="short image"> which is not tall.</P>
```



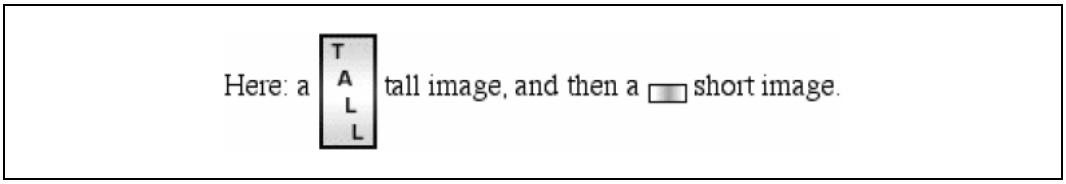

*Rysunek 4.34. Wyrównanie dolne*

Jak widać na rysunku 4.34, druga linia akapitu zawiera dwa obrazki, których dolne brzegi są ze sobą wyrównane. Dla pierwszego obrazka zostało ustawione wyrównanie środkowe przy użyciu atrybutu HTML `valign`. Nieco później — ale jeszcze w tym rozdziale — przyjrzymy się sposobom stosowanym w CSS do osiągnięcia podobnych efektów.

Fraza `vertical-align: text-bottom` odnosi się do dolnego brzegu linii tekstu. Dla spełnienia celów tej wartości ignorowane są elementy podmieniane i dowolnego typu elementy nietekstowe. Pod uwagę brane jest natomiast domyślne pole tekstowe. To domyślne pudełko wywodzi się z właściwości `font-size` elementu rodzica. Spód wyrównywanego elementu jest wyrównany ze spodem domyślnego pola tekstowego. A zatem — stosując poniższy kod — zobaczymy sytuację przedstawioną na rysunku 4.35:

```
IMG.tbot {vertical-align: text-bottom;}
```

```
<P>Here: a <IMG SRC="tall.gif" ALIGN="middle"
ALT="tall image"> tall image, and then a <IMG SRC="short.gif"
CLASS="tbot" ALT="short image"> short image.</P>
```

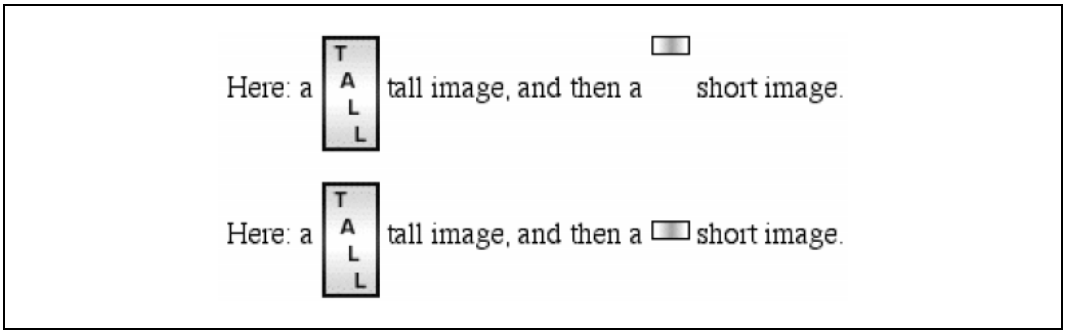



*Rysunek 4.35. Ustawienie text-bottom*

## *W górę*

Użycie `vertical-align: top` ma przeciwny efek niż wartość `bottom` (podobnie jak `vertical-align: text-top` jest odwrotnością `text-bottom`). Rysunek 4.36 przedstawia efekt, jaki da wykonanie poniższego kodu:

```
IMG.up {vertical-align: top;}
IMG.textup {vertical-align: text-top;}

<P>Here: a <IMG SRC="tall.gif" ALIGN="middle"
ALT="tall image"> tall image, and then a <IMG SRC="short.gif"
CLASS="up" ALT="short image"> short image.</P>
<P> Here: a <IMG SRC="tall.gif" ALIGN="middle"
ALT="tall image"> tall image, and then a <IMG SRC="short.gif"
CLASS="textup" ALT="short image"> short image.</P>
```




*Rysunek 4.36. Użycie właściwości top i text-top do wyrównania tekstu*

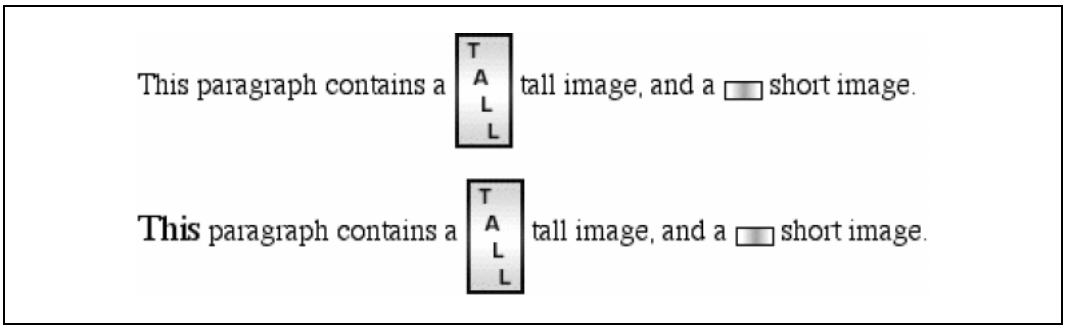
Dokładna pozycja wyrównania będzie zależała od tego, jakie elementy znajdą się w linii i od tego, jak będą wysokie.

## *Pośrodku*

A teraz `vertical-align: middle`. Wartość ta stosowana jest zazwyczaj w celu wyrówania rysunków, ponieważ powoduje ustawienie środka elementu na poziomie środka linii. Środek linii zdefiniowany jest jako punkt znajdujący się o połowę wartości `ex` nad bazową linią pisma, a wartość

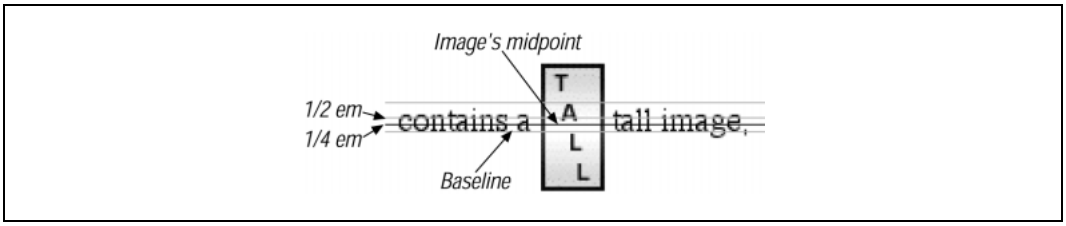
ex obliczana jest na podstawie wielkości czcionki elementu rodzica. Przykład przedstawiony jest na rysunku 4.37:

```
IMG {vertical-align: middle;}
```




*Rysunek 4.37. Wyrównanie środków*

Ponieważ większość przeglądarek wyświetla 1ex jako połowę em, wartość middle praktycznie powoduje ustawienie środka elementu na wysokości jednej czwartej wartości em nad bazową linią pisma rodzica. Obrazuje to dokładnie rysunek 4.38.



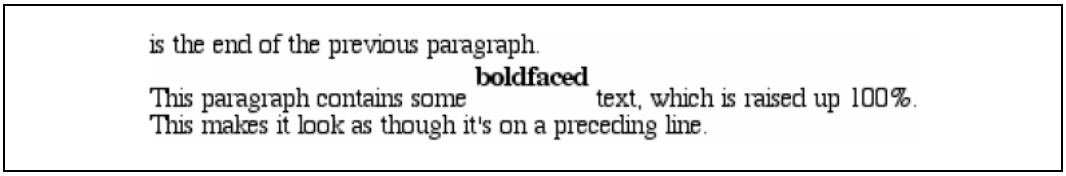

*Rysunek 4.38. Szczegóły wyrównania środkowego*

To posunięcie jest bliskie symulacji działania <IMG ALIGN="middle">; można spodziewać się, że wartości procentowe dadzą jeszcze lepsze wyniki.

## *Procenty*

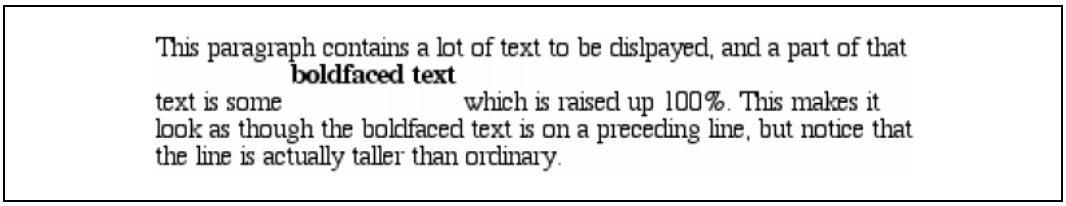
Jak się okazuje, procenty nie pozwalają symulować ALIGN="middle" w przypadku obrazków. Ustawienie wartości procentowej dla vertical-align powoduje podniesienie lub opuszczenie bazowej linii pisma (lub spodu elementu podmienianego) o podaną wielkość względem linii bazowej rodzica. Wielkość przesunięcia obliczana jest jako procent wartości wysokości linii. Dodatnie wartości powodują podniesienie elementu, a ujemne — obniżenie. Może to spowodować taki efekt jak ten, który jest widoczny na rysunku 4.39 (elementy wyglądają tak, jakby były umieszczone w osobnej linii):

```
B {vertical-align: 100%}

<P>This paragraph contains some <B>boldfaced</B> text, which is
raised up 100%.  This makes it look as though it's on a preceding
line.</P>
```




*Rysunek 4.39. Wyrównanie pionowe przy użyciu procentów*

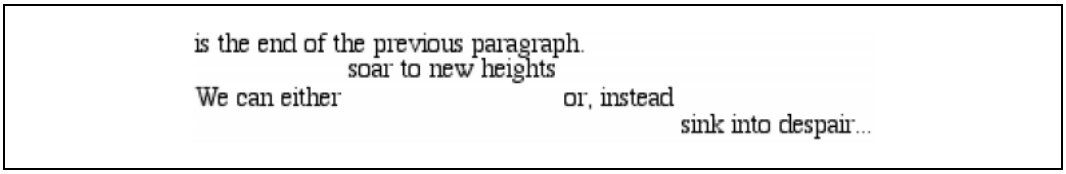
Należy jednak pamiętać, że wyrównywany pionowo tekst nie jest częścią innej linii. Wydaje się tak jedynie wtedy, gdy nie ma innego tekstu dookoła. Spójrzmy na rysunek 4.40, na którym w środku akapitu pojawia się pewien wyrównany pionowo tekst.




*Rysunek 4.40. Wyrównanie pionowe może wpłynąć na wysokość linii*

Można tę właściwość wykorzystać do uzyskania ciekawych efektów wizualnych jak te, które zostały przedstawione na rysunku 4.41:

```
SUB {vertical-align: -100%;}
SUP {vertical-align: 100%;}

<P>We can either <SUP>soar to new heights</SUP> or, instead,
<SUB>sink into despair...</SUB></P>
```

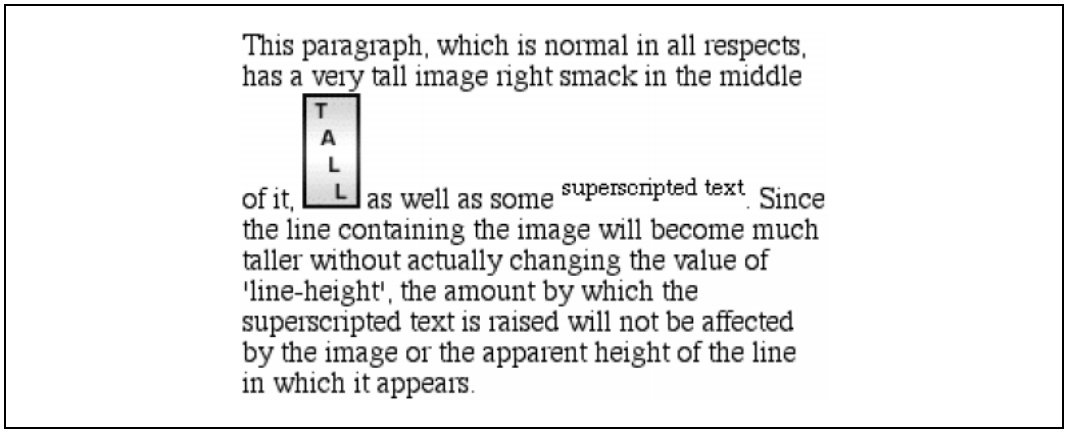



*Rysunek 4.41. Procenty i efektowne skutki*

Skoro wartości procentowe mają być procentami wysokości linii, można zapytać o to, co się stanie, jeśli szczególnie wysoki obrazek umieszczony wewnątrz linii zwiększy jej wysokość?

Jeśli przypomnimy sobie wcześniejsze rozważania, to odpowiedź już znamy — obrazek, bez względu na swoją wysokość, nie spowoduje powiększenia wartości `line-height`. Zwiększy się wysokość pudełka. Dlatego — jeśli wysokość linii wynosi `14px`, a element wewnątrz tej linii jest podniesiony o `50%`, a wewnątrz niej znajduje się także obrazek wysokości 50 pikseli — to otrzymany rezultat będzie taki, jak to przedstawiono na rysunku 4.42:

```
SUP {vertical-align: 50%;}
```

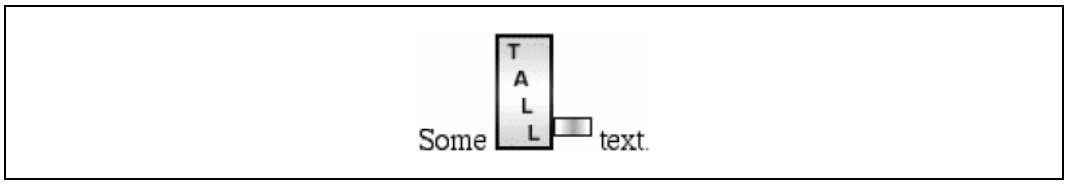



*Rysunek 4.42. Wyrównanie pionowe — procenty i wysoki obrazek*

Podniesiony o `50%` element będzie miał linię bazową podniesioną o 7 pikseli (połowa `14px`), a nie o 25 pikseli. Zauważmy także, że pudełko linii zostało odpowiednio zwiększone, aby mogło pomieścić rysunek. Jest to zgodne z modelem pudełka, ponieważ elementy podmieniane dają efekty tego rodzaju.

Możemy lepiej prześledzić operację wyrównania pionowego, jeśli mamy dwa obrazki, z których tylko jeden jest wyrównany pionowo za pomocą wartości procentowej. Rezultaty, które są widoczne na rysunku 4.43, są zależne od wysokości linii, którą wyraźnie ustawiliśmy na `14px`:

```
P {line-height: 14px;}
IMG.up {vertical-align: 50%;}

<P>Some <IMG SRC="tall.gif" alt="tall image">
<IMG SRC="short.gif" CLASS="up" ALT="short image"> text.</P>
```



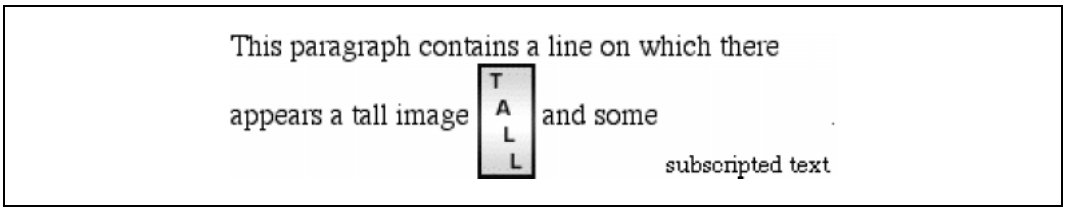

*Rysunek 4.43. Wyrównanie pionowe — procenty i dwa obrazki*

Spód (lub linia bazowa) mniejszego obrazka jest podniesiony o połowę (`50%`) wysokości linii lub o 7 pikseli. Jeśli ustawilibyśmy wyrównanie dla `IMG.up` na `-50%`, to niższy obrazek zostałby obniżony o 7 pikseli.

## Łączenie ustawień

Jak już widzieliśmy, sposób wyświetlenia danej linii tekstu zależy w dużym stopniu od wyrównania pionowego różnych elementów wewnątrz tej linii. Fakt ten, ze względu na swą wagę, zasługuje na ponowne podkreślenie. Na przykład załóżmy, że linia zawiera wyśrodkowany obrazek i wyrównany do dołu element normalnego ciągu tekstu. Rezultat widać na rysunku 4.44:

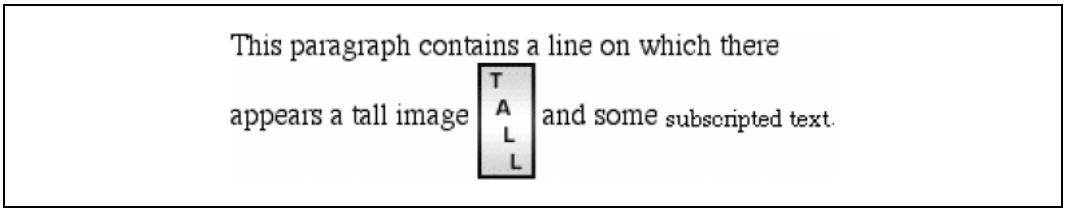
```
IMG {vertical-align: middle;}
SUB {vertical-align: bottom;}
```




*Rysunek 4.44. Łączenie wyrównań pionowych*

Jeśli zmienimy wyrównanie elementu tekstowego na `text-bottom`, sytuacja zmieni się dość radykalnie, jak to przedstawiono na rysunku 4.45:

```
IMG {vertical-align: middle;}
SUB {vertical-align: text-bottom;}
```




*Rysunek 4.45. Inny sposób łączenia wyrównań pionowych*

## *Odstępy między słowami i znakami*

Dosyć proste pojęcia — odstępy między słowami i znakami — to po prostu sposoby na zwiększenie lub zmniejszenie przestrzeni pomiędzy słowami lub literami. Jak zawsze do rozważenia jest parę nieintuicyjnych kwestii dotyczących tych właściwości. Na początek omówmy odstępy między słowami.

### *Odstępy między słowami*

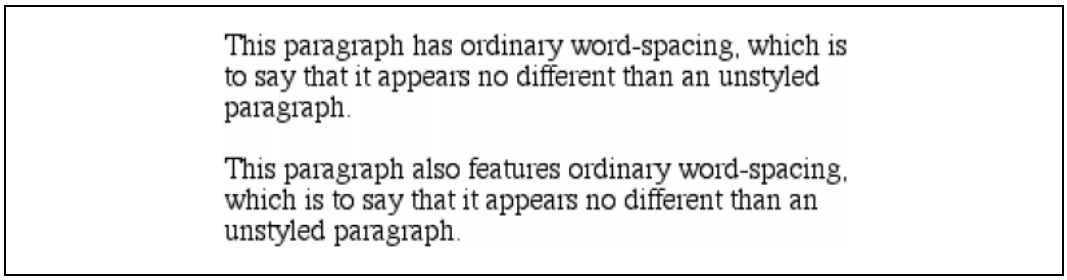
**word-spacing**

| | |
|---|---|
| **Wartości** | *<długość>* \| normal |
| **Wartość domyślna** | normal |
| **Dziedziczona** | tak |
| **Odnosi się do** | wszystkich elementów |

Właściwość `word-spacing` (*odstępy między słowami*) może przyjmować wartość dodatnią lub ujemną. Wartość jest *dodana* do normalnej przestrzeni między słowami, co prawdopodobnie nie

jest tym, czego moglibyśmy się spodziewać. W efekcie podanej w deklaracji `word-spacing` wartości używa się jako modyfikatora przestrzeni między słowami. Dlatego wartość domyślna — `normal` działa tak samo jak ustawienie wartości zerowej (`0`), co przedstawiono na rysunku 4.46:

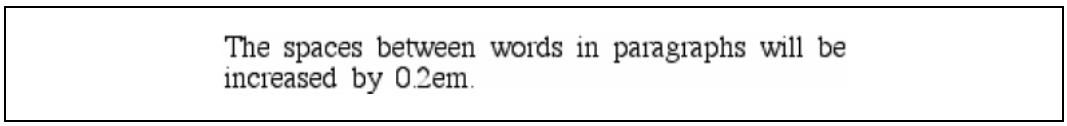
```
P.base {word-spacing: normal;}
P.norm {word-spacing: 0;}
```




*Rysunek 4.46. Dwa sposoby na osiągnięcie zwykłych odstępów między słowami*

Jeśli zatem wpiszemy dodatnią wartość długości, przestrzeń pomiędzy słowami zwiększy się, jak to przedstawiono na rysunku 4.47:

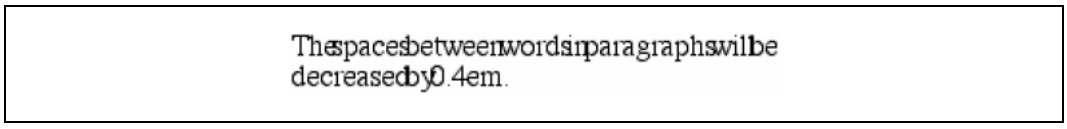
```
P {word-spacing: 0.2em;}

<P>The spaces between words in paragraphs will be increased by
0.2em.</P>
```




*Rysunek 4.47. Zwiększenie przestrzeni między słowami*

Słowa mogą zostać zbliżone poprzez ustawienie dla właściwości `word-spacing` wartości ujemnej. Da to taki efekt, jak widać na rysunku 4.48:

```
P {word-spacing: -0.4em;}

<P>The spaces between words in paragraphs will be decreased by
0.4em.</P>
```




*Rysunek 4.48. Zmniejszenie przestrzeni pomiędzy słowami*

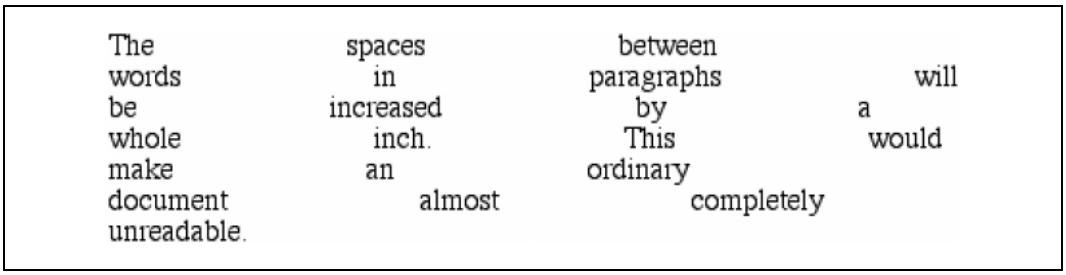
Jak na razie nie określiliśmy dokładnie, czym tak naprawdę jest „słowo". W najprostszych terminach CSS „słowo" jest ciągiem znaków bez przerw, otoczonym przez puste znaki jakiegokolwiek typu. Definicja nie ma znaczenia semantycznego — zakłada się jedynie, iż autor opracowuje dokument tak, że każde słowo jest oddzielone od drugiego przynajmniej jednym pustym znakiem. Nie można

jednak oczekiwać od przeglądarek rozumiejących CSS, aby decydowały, co jest poprawnym słowem w danym języku, a co nim nie jest.

Oznacza to także, że nie każdy język, który używa piktogramów lub innych, niż rzymski sposobów pisania, będzie mógł korzystać z tej właściwości. Ponadto przeglądarki nie muszą uwzględniać podanej tu definicji słowa; to tylko najprostszy sposób zdefiniowania tego pojęcia. Przeglądarki mogą użyć bardziej zaawansowanych sposobów i decydować o tym, co słowem jest, a co nie.

Używając właściwości odstępów między słowami można stworzyć bardzo nieczytelny dokument, taki jak widzimy na rysunku 4.49:

```
P {word-spacing: 1in;}
```




*Rysunek 4.49. Naprawdę szerokie odstępy między słowami*

## *Kerning*

### letter-spacing

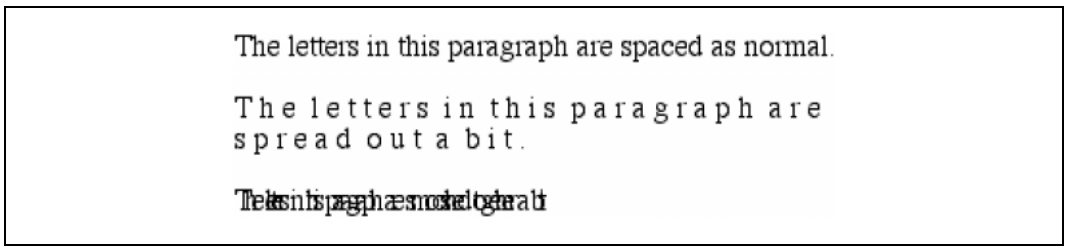
| | |
|---|---|
| **Wartości** | *<długość>* \| normal |
| **Wartość domyślna** | normal |
| **Dziedziczona** | tak |
| **Odnosi się do** | wszystkich elementów |

Wiele aspektów dotyczących odstępów między słowami obowiązuje także w przypadku odstępów między znakami. Jedyną rzeczywistą różnicą pomiędzy tymi właściwościami jest to, że `letter-spacing` jest modyfikatorem normalnego światła międzyliterowego.

Tutaj także dozwolone są wszystkie wartości długości i domyślne słowo kluczowe `normal` (wywołujące takie samo działanie jak `letter-spacing: 0`). Każda wartość długości zwiększy lub zmniejszy przestrzeń pomiędzy literami o zadeklarowaną wartość. Rysunek 4.50 przedstawia rezultaty poniższego przykładowego kodu:

```
P {letter-spacing: 0;}   /* identyczne działanie jak "normal" */
P.szerokie {letter-spacing: 0.25em;}
P.waskie {letter-spacing: -0.25em;}
<P>The letters in this paragraph are spaced as normal.</P>
```

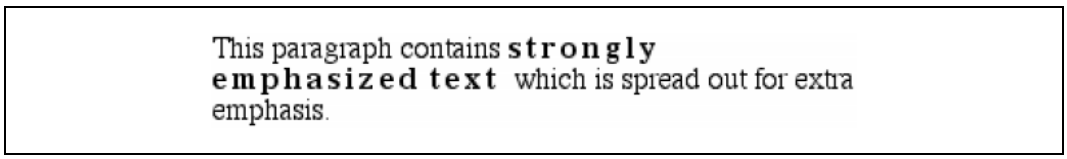
```
<P CLASS="szerokie">The letters in this paragraph are spread out a
bit.</P>
<P CLASS="waskie">The letters in this paragraph are smooshed together
a bit.</P>
```




*Rysunek 4.50. Rożne rodzaje kerningu*

Interesującym wykorzystaniem odstępów międzyliterowych jest zwiększenie ich w celu podkreślenia znaczenia wyrazu. Taka technika była powszechnie stosowana w poprzednich stuleciach. A zatem można zadeklarować poniższy kod, aby otrzymać rezultat widoczny na rysunku 4.51:

```
STRONG {letter-spacing: 0.2em;}

<P>This paragraph contains <STRONG>strongly emphasized text</STRONG>
which is spread out for extra emphasis.</P>
```



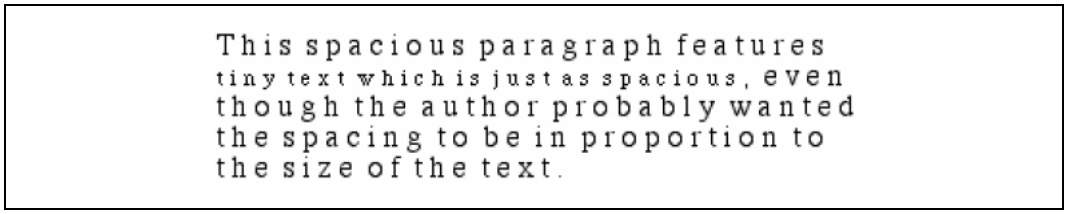

*Rysunek 4.51. Użycie odstępów między znakami dla szczególnego podkreślenia treści*

## *Odstępy, wyrównanie i rozmiar czcionki*

Wartość `text-align` może wpływać na odstępy między słowami i między znakami. Jeśli element jest wyjustowany, to odstępy między znakami i słowami mogą ulec zmianie. Umożliwienie pełnego justowania może zmienić odstępy zadeklarowane przez autora przez właściwości `word-spacing` i `letter-spacing`. Specyfikacja CSS nie określa w takich przypadkach sposobu obliczenia odstępów, a więc przeglądarki mogą robić to, co ich programiści uznali za najlepsze rozwiązanie.

Jak to zwykle bywa, obliczona wartość dziedziczona jest przez wszystkie dzieci elementu. W odróżnieniu od wysokości linii dla `word-spacing` i `letter-spacing` niemożliwe jest ustawienie współczynnika skalującego, który byłby dziedziczony zamiast wyliczonej wartości. A zatem pojawiają się problemy takie jak przedstawiony na rysunku 4.52:

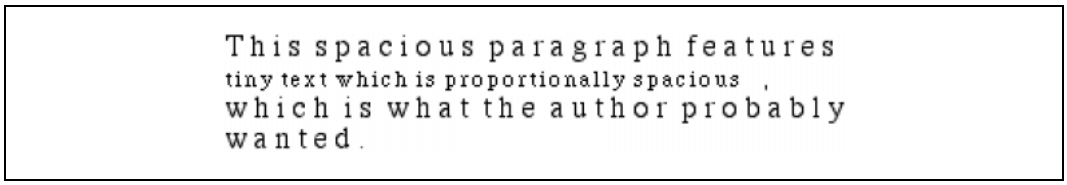
```
P {letter-spacing: 0.25em;}
SMALL {font-size: 50%;}

<P>This spacious paragraph features <SMALL>tiny text which is just
as spacious</SMALL>, even though the author probably wanted the
spacing to be in proportion to the size of the text.</P>
```




*Rysunek 4.52. Odziedziczone światła międzyliterowe*

Jedyną szansą na uzyskanie efektu proporcjonalności odstępów do wielkości tekstu jest wyraźne zadeklarowanie kerningu, co pokazano na rysunku 4.53:

```
P {letter-spacing: 0.25em;}
SMALL {font-size: 50%; letter-spacing: 0.25em;}

<P>This spacious paragraph features <SMALL>tiny text which is
proportionally spacious</SMALL>, which is what the author
probably wanted.</P>
```




*Rysunek 4.53. Odrzucenie dziedziczenia wartości kerningu*

## *Transformacja tekstu*

Teraz zajmijmy się kapitalizacją tekstu. Możemy to zrobić posługując się właściwością `text-transform` (*transformacja tekstu*).

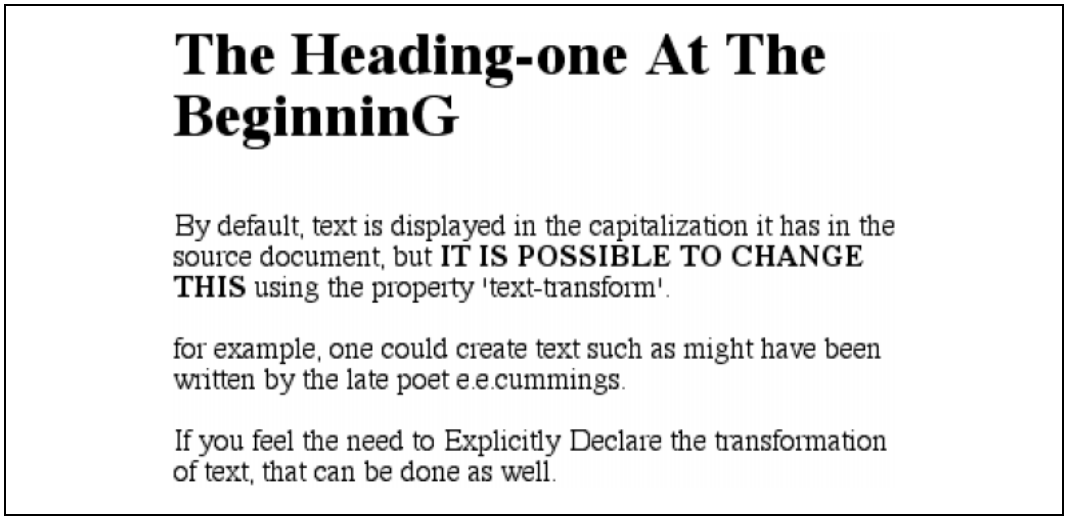
### text-transform

| | |
|---|---|
| **Wartości** | `uppercase \| lowercase \| capitalize \| none` |
| **Wartość domyślna** | `none` |
| **Dziedziczona** | tak |
| **Odnosi się do** | wszystkich elementów |

Domyślna wartość `none` pozostawia tekst bez zmian, gdyż stosuje taką kapitalizację, jaka istnieje w źródle dokumentu. Wartości, takie jak `uppercase` (*wersaliki*, czyli wielkie litery) i `lowercase` (*małe litery*), powodują zamianę tekstu tylko na wielkie lub tylko na małe litery zgodnie z tym, co sygnalizują nazwy. Wartość `capitalize` (*kapitalizacja*) zmienia pierwszą literę każdego wyrazu na wielką, a resztę liter pozostawia bez zmian. Zauważmy, że dla tej właściwości definicja słowa jest taka sama jak w podrozdziale o odstępach między słowami.

Rysunek 4.54 przedstawia wiele sposobów użycia ustawień.

```
STRONG {text-transform: uppercase;}
H1, H2, H3 {text-transform: capitalize;}
P.cummings {text-transform: lowercase;}
P.surowy {text-transform: none;}

<H1>The heading-one at the beginninG</H1>
<P>
By default, text is displayed in the capitalization it has in the
source document, but <STRONG>it is possible to change this</STRONG>
using the property 'text-transform'.
</P>
<P CLASS="cummings">
For example, one could Create TEXT such as might have been Written by
the late Poet E.E.Cummings.
</P>
<P CLASS="surowy">
If you feel the need to Explicitly Declare the transformation of
text, that can be done as well.
</P>
```




*Rysunek 4.54. Różne typy transformacji tekstu*

Pamiętajmy, że różne przeglądarki w różny sposób mogą decydować o tym, gdzie zaczyna się nowe słowo (czyli o tym, jakie litery mają być zamienione). Na przykład dla elementy `H1` z rysunku 4.54 tekst „heading-one” może być wyświetlony na jeden z dwóch sposobów: „Heading-One” lub „Heading-one”. CSS nie określa, który zapis jest poprawny, więc obydwa są możliwe.

Zauważmy także, że w nagłówku `H1` na rysunku 4.54 ostatnia litera jest nadal wielka. Taka sytuacja jest poprawna: nadając `text-transform: capitalize` CSS wymaga jedynie od przeglądarki zamiany pierwszych liter wyrazów na wielkie. W pozostałej części wyrazu nic nie zostało więc zmienione.

Może się wydawać, że właściwość `text-transform` nie ma zbyt dużych możliwości. Ale w rzeczywistości jest bardzo użyteczna. Jest tak na przykład wtedy, gdy nagle zdecydujemy, że wszystkie elementy `H1` powinny być w całości napisane wielkimi literami. Zamiast przepisywać wszystkie nagłówki, wystarczy posłużyć się transformacją tekstu:

```
H1 {text-transform: uppercase;}

<H1>This is an H1 element</H1>
```

Jak widać na rysunku 4.55, tekst jest napisany wielkimi literami.


*Rysunek 4.55. Transformowanie elementu H1*

Zalet takiego posunięcia jest wiele. Po pierwsze — aby dokonać zmiany, wystarczy napisać pojedynczą regułę, zamiast zmieniać wszystkie nagłówki `H1`. Po drugie — jeśli później zdecydujemy się na zamianę wielkich liter na kapitalizacje tylko pierwszej litery każdego wyrazu, zmiana jest jeszcze łatwiejsza, co pokazuje rysunek 4.56:

```
H1 {text-transform: capitalize;}

<H1>This is an H1 element</H1>
```


*Rysunek 4.56. Inny sposób transformowania elementu H1*

## *Ozdabianie tekstu*

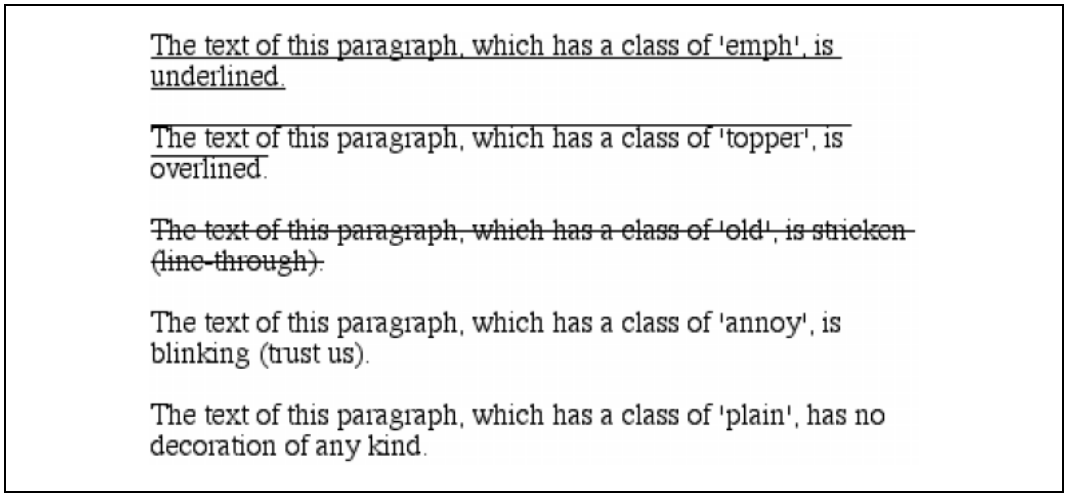
W ten sposób dochodzimy do dekoracji tekstu (`text-decoration`) będącej fascynującą właściwością, która daje szansę na wykorzystanie dziwnych efektów i nieścisłości w przeglądarkach. Jednak na początku omówmy sposób, w jaki właściwość ta powinna działać teoretycznie.

### text-decoration

| | |
|---|---|
| **Wartości** | `none` \| [`underline` \|\| `overline` \|\| `line-through` \|\| `blink`] |
| **Wartość domyślna** | `none` |
| **Dziedziczona** | nie |
| **Odnosi się do** | wszystkich elementów |

Jak można się spodziewać, `underline` (*podkreślenie*) powoduje podkreślenie elementu. Działa dokładnie tak samo jak znacznik `U` w HTML. Wartość `overline` (*nadkreślenie*) jest „lustrzanym odbiciem" podkreślenia; linia jest rysowana wzdłuż górnej krawędzi tekstu „nadkreślanego" elementu. Wartość `line-through` (*linia poprzez*) przeprowadza linię prosto przez środek tekstu; jest znana także jako „przekreślenie" (ang. *striketrough*) tekstu i jest odpowiednikiem znaczników `S` i `STRIKE` w HTML. A `BLINK` (*mrugać*) powoduje oczywiście miganie tekstu (dokładnie tak samo jak znacznik `BLINK` obsługiwany przez program Netscape Navigator). Rysunek 4.57 przedstawia efekty działania każdej z tych wartości:

```
P.emph {text-decoration: underline;}
P.topper {text-decoration: overline;}
P.old {text-decoration: line-through;}
P.annoy {text-decoration: blink;}
P.plain {text-decoration: none;}
```



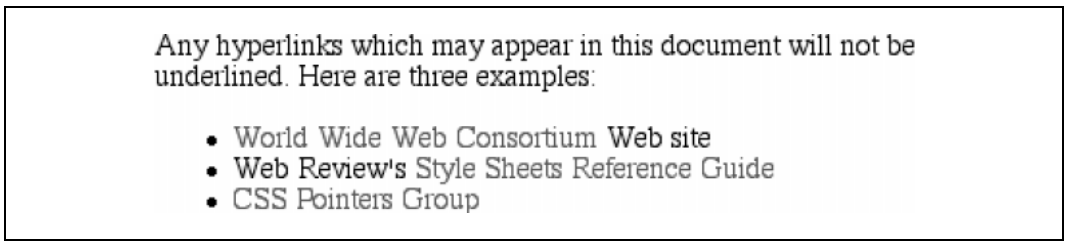

*Rysunek 4.57. Różne typy dekoracji tekstu*

Pokazanie efektu migania (`blink`) oczywiście nie jest możliwe w druku, ale jest łatwe do wyobrażenia. Przeglądarki nie muszą obsługiwać migania. Nawiasem mówiąc — w czasie, gdy pisałem tę książkę, miganie obsługiwał tylko Navigator 4.x.

Jak widać na ostatniej części rysunku 4.57, wartość `none` wyłącza wszelkie dekoracje, jakie mogły zostać nadane elementowi. Jest to zazwyczaj domyślny wygląd tekstu, ale nie zawsze tak się dzieje. Na przykład odnośniki domyślnie są tu podkreślone. Jeśli chcielibyśmy pozbyć się podkreślenia hiperłączy, to reguła CSS byłaby taka:

```
A:link {text-decoration: none;}
```

Tekst na rysunku 4.58 zawiera trzy odnośniki. Są to trzy elementy listy. Skoro wyłączyliśmy podkreślanie odnośników, to jedyną widoczną różnicą pomiędzy odnośnikami a normalnym tekstem jest kolor.




*Rysunek 4.58. Wyłączenie podkreślania hiperłączy*

Wielu użytkowników irytuje się, kiedy zdadzą sobie sprawę z tego, że autor wyłączył podkreślanie odnośników. Oczywiście *de gustibus non est disputandum*, pozwólmy więc gustowi być w tej kwestii naszym przewodnikiem. Pamiętajmy jednak, że jeśli kolory odnośników dostatecznie nie odróżniają się od normalnego tekstu, użytkownicy ze słabszym wzrokiem mogą mieć duże problemy z wyszukaniem hiperłączy w naszym dokumencie.

W jednej regule można połączyć kilka dekoracji. Załóżmy na przykład, że chcemy, aby wszystkie odnośniki były jednocześnie podkreślone oraz nadkreślone. Reguła, którą musimy zastosować wygląda następująco:

```
A:link, A:visited {text-decoration: underline overline;}
```

Dzięki temu `text-decoration` pozwala na skrócenie zapisu. W takim przypadku należy być ostrożnym, ponieważ może się zdarzyć, że mamy dwie różne dekoracje przyporządkowane temu samemu elementowi, a wtedy wartość wygrywającej reguły całkowicie zastąpi regułę przegraną. Spójrzmy:

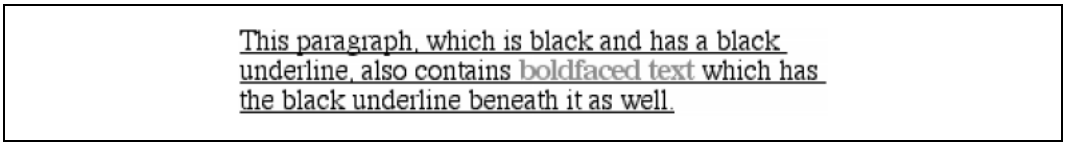
```
H2.przekreslony {text-decoration: line-through;}
H2 {text-decoration: underline overline;}
```

Przy zastosowaniu takich dwóch reguł każdy element `H2` o klasie „przekreślony" będzie miał dekorację tylko w postaci przekreślenia. Podkreślenie i nadkreślenie zostaną zgubione, ponieważ jedna deklaracja zastąpi drugą, a nie skumuluje się z nią.

## *Dziwne dekoracje*

Teraz zagłębmy się w paru dziwacznych aspektach dekoracji tekstu. Po pierwsze wartość — `text-decoration` nie jest dziedziczona. A to pociąga za sobą pewne wymaganie — każda linia dekoracji narysowana wzdłuż tekstu (obojętnie czy nad, pod czy poprzez) musi być tego samego koloru co element rodzic. Dzieje się tak nawet wtedy, gdy potomek jest innego koloru, jak przedstawiono na rysunku 4.59:

```
P {text-decoration: underline; color: black;}
B {color: gray;}

<P>This paragraph, which is black and has a black underline, also
contains
<B>boldfaced text</B> which has the black underline beneath it as
well.</P>
```




*Rysunek 4.59. Jednolitość podkreślenia*

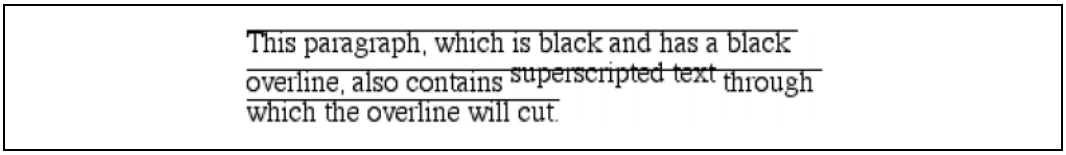
Dlaczego tak się dzieje? Wartość `text-decoration` nie jest dziedziczona, zatem element `B` ma domyślną wartość `none`. Dlatego element `B` *nie* ma podkreślenia. Oczywiście jest wyraźnie widoczna linia występująca pod elementem `B`, a więc idiotyzmem wydaje się uparte twierdzenie, że nie jest on podkreślony. Ale nie jest. To, co widzimy pod elementem `B` to podkreślenie akapitu, które obejmuje element `B`. Może stać się to bardziej wyraźne, jeśli zmienimy style pogrubionego elementu w taki sposób:

```
P {text-decoration: underline; color: black;}
B {color: gray; text-decoration: none;}

<P>This paragraph, which is black and has a black underline, also
contains
<B>boldfaced text</B> which has the black underline beneath it as
well.</P>
```

Rezultat jest taki sam; dokładnie już omówiliśmy, czemu się tak dzieje. Innymi słowy — nie ma możliwości wyłączenia podkreślenia (nadkreślenia lub przekreślenia) wewnątrz elementu. Jeśli dla jakiegoś elementu została ustawiona dekoracja, to dzieci tego elementu też będą miały tę dekorację nadaną, jeśli nawet nie rzeczywiście, to przynajmniej wizualnie.

Jeszcze dziwniejsze rzeczy będą się działy, gdy połączymy dekoracje z wyrównaniem pionowym. Rysunek 4.60 pokazuje nam jedną z nich. Skoro element `SUP` nie ma własnej dekoracji, ale jest podniesiony wewnątrz nadkreślonego elementu, to nadkreślenie przechodzi przez środek elementu `SUP`:

```
P {text-decoration: overline; font-size: 12pt;}
SUP {vertical-align: 50%; font-size: 12pt;}
```

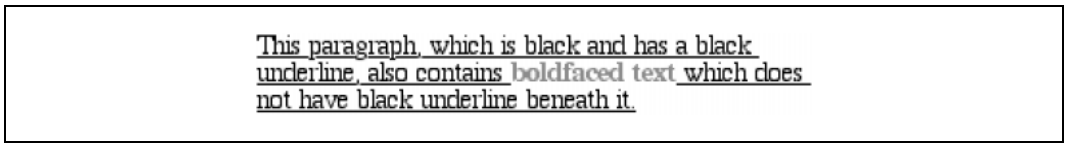



*Rysunek 4.60. Poprawny, ale dziwny efekt dekoracji*

Dekoracje tekstu powodują tak duże problemy, że możemy zwątpić w celowość ich użycia. Chociaż może być jeszcze gorzej (a może lepiej) — na razie omówiliśmy teoretyczne działanie właściwości. W rzeczywistości wiele przeglądarek wyłącza podkreślanie elementów potomków, chociaż nie powinno to być stosowane. Powodem, dla którego naruszają one specyfikację, są po prostu oczekiwania autora.

```
P {text-decoration: underline; color: black;}
B {color: gray; text-decoration: none;}

<P>This paragraph, which is black and has a black underline, also
contains
<B>boldfaced text</B> which does not have black underline beneath
it.</P>
```

Jak widać na rysunku 4.61 przeglądarka wyłączyła podkreślenie elementu `B`. Robi to Navigator, Explorer i Opera, jeśli wyraźnie zaznaczono `text-decoration: none`, aby wyłączyć dekorację. Jest to oczywiście to, czego spodziewałby się autor i dlatego przeglądarki spełniają jego oczekiwania.



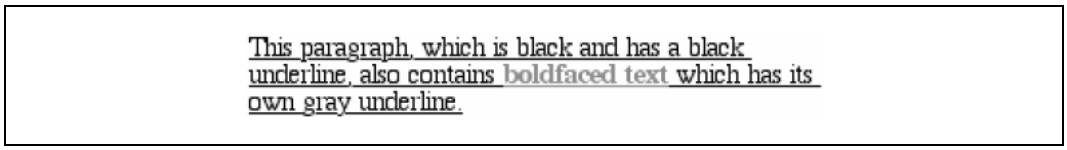

*Rysunek 4.61. Jak naprawdę zachowują się przeglądarki*

Jednak w przyszłości przeglądarki mogą zacząć precyzyjnie stosować się do specyfikacji. Jeśli zaczniemy korzystać z `none` w celu wyzwolenia się z dekoracji, należy pamiętać, że być może kiedyś będziemy musieli wrócić do tego problemu. Miejmy nadzieję, że przyszłe wersje CSS będą zawierać sposób na wyłączenie dekoracji bez używania `none`.

Ostatecznie istnieje sposób na to, by zmienić kolor dekoracji bez naruszania zasad specyfikacji. Jak sobie przypominamy — ustawienie dekoracji tekstu na danym elemencie oznacza, że cały element powinien mieć taki sam kolor dekoracji, nawet jeśli istnieją potomkowie o innym kolorze. Aby połączyć kolor dekoracji z kolorem każdego elementu, należy wyraźnie zadeklarować dekorację tych elementów tak, jak to widać poniżej:

```
P {text-decoration: underline; color: black;}
B {color: gray; text-decoration: underline;}

<P>This paragraph, which is black and has a black underline, also
contains
<B>boldfaced text</B> which has its own gray underline.</P>
```

Na rysunku 4.62 element `B` ustawiony jest jako szary i podkreślony. Szare podkreślenie zapisuje się na podkreśleniu rodzica, a więc kolor dekoracji pasuje do koloru elementu `B`.




*Rysunek 4.62. Zwalczenie domyślnego zachowania podkreśleń*

# *Podsumowanie*

Istnieje wiele sposobów zmiany wyglądu tekstu. Można je stosować nawet bez podejmowania prób zmiany użytych czcionek. Istnieje klasyczny efekt podkreślenia, ale CSS daje nam możliwość rysowania linii nad tekstem i poprzez tekst, daje możliwość zmiany wielkości odstępów między słowami i znakami, wcięcia pierwszej linii akapitu (lub innego elementu blokowego), wyrównania tekstu do lewej lub prawej strony itd. Można nawet zmienić odstęp występujący pomiędzy poszczególnymi liniami tekstu, chociaż operacja ta jest nadspodziewanie skomplikowana (została dokładnie opisana w rozdziale 8.).

Te zachowania obsługiwane są względnie dobrze albo nie są obsługiwane wcale. Jedną z tych nienajlepiej obsługiwanych rzeczy jest pełne justowanie tekstu. Większość wprowadzonych na rynek w dwudziestym wieku przeglądarek zawierała błędy w obsłudze dekoracji tekstu i wyrównania pionowego (podobnie jak w obliczaniu wysokości linii). Z drugiej strony — odstępy między słowami i znakami prawie zawsze działają poprawnie (gdy są obsługiwane), a wcięciom towarzyszy jedynie kilka bardzo małych błędów. To samo dotyczy możliwości zmiany kapitalizacji, obsługiwanej zazwyczaj poprawnie.

Następną rzeczą, którą autorzy zwykle chcą zrobić, jest zmiana czcionki użytej w tekście, zmiana jej rozmiaru, grubości i innych właściwości. O tym, jak to zrobić, przeczytamy w następnym rozdziale.