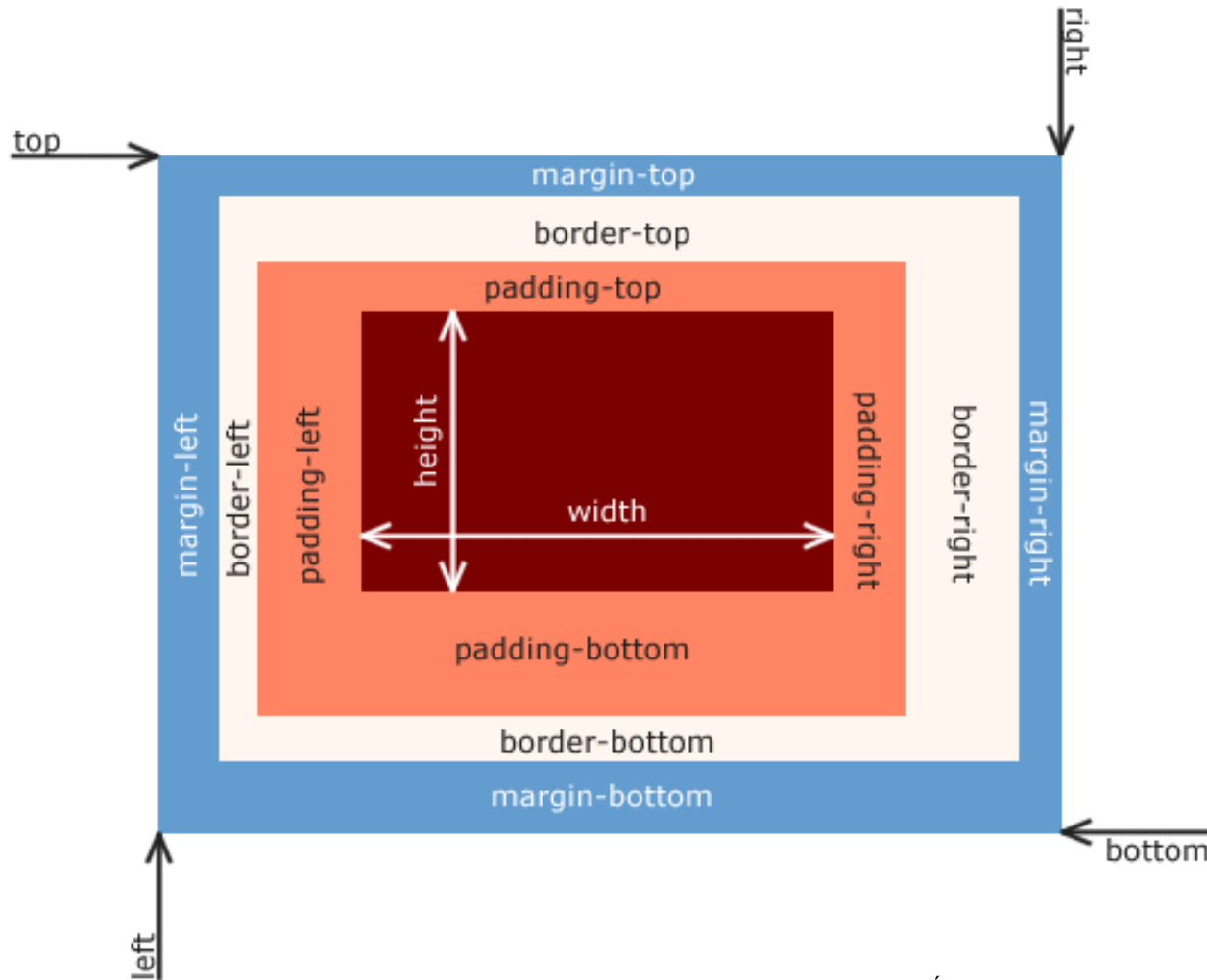


Model pudełkowy

Model pudełkowy (CSS box model)



Źródło: Wikipedia

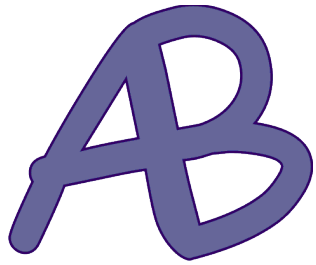
Do ustalania rozmiarów pudełka służą atrybuty **width** (szerokość) i **height** (wysokość). Określają one rozmiar zarezerwowany dla zawartości.

Obszar ten może mieć tło (**background-color**) i obramowanie (**border**).

Aby określić całkowity rozmiar zajmowany przez pudełko należy więc uwzględnić:

- ✓ marginesy wewnętrzne (**padding**),
- ✓ obramowanie (**border**),
- ✓ marginesy (**margin**).

(Pamiętaj o deklaracji typu dokumentu — jest ona potrzebna, aby przeglądarka poprawnie zinterpretowała model pudełkowy).



Klasy i selektory

Selektor potomka

- Jeśli selektorem w regule CSS jest nazwa znacznika, to będzie jej podlegał każdy znacznik tego typu.
- Na przykład, poniższa reguła sprawi, że kolor tekstu we wszystkich akapitach będzie czerwony:

```
p {color:red;}
```

- Co jednak zrobić, jeśli chcemy, aby tylko jeden, określony akapit miał czerwony tekst?
- Do bardziej precyzyjnej selekcji znaczników służy selektor potomka. Oto przykład takiego selektora:

```
div p {color: red;}
```

- Teraz czerwony kolor tekstu będą miały akapity znajdujące się wewnątrz znacznika `div`.

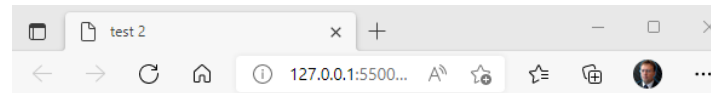
Selektor potomka (przykład)

<h1>Selektor potomka jest bardzo selektywny.</h1>

<p>Niniejszy przykład pokazuje sposób selekcji określonego znacznika w hierarchii dokumentu.</p>

<p>Znaczniki muszą tylko być potomkami w kolejności określonej w selektorze; Pomiędzy nimi mogą znajdować się inne znaczniki i nie mają one wpływu na działanie selektora.</p>

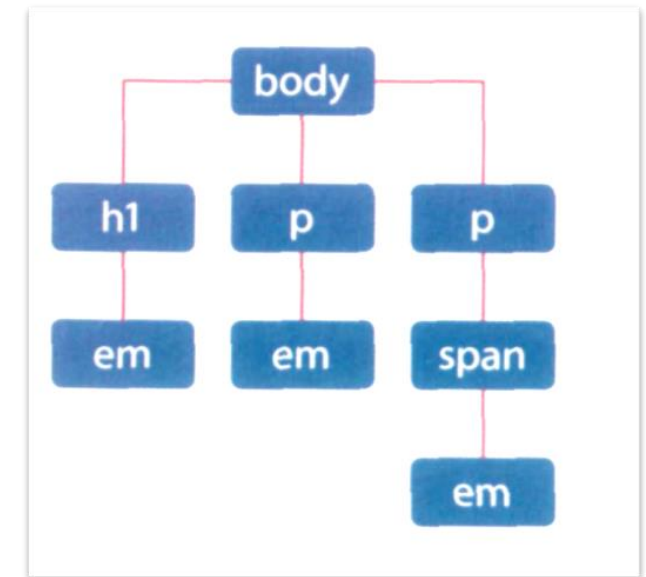
```
em {font-size: 1.5em;}  
p em {color: blue;}  
h1 em {color: green;}  
p span em {color: red;}
```



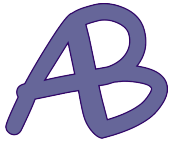
Selektor potomka jest *bardzo* selektywny.

Niniejszy przykład pokazuje sposób selekcji *określonego* znacznika w hierarchii dokumentu.

Znaczniki muszą tylko być potomkami w *kolejności określonej* w selektorze; Pomiędzy nimi mogą znajdować się inne znaczniki i nie mają one wpływu na działanie selektora.



Selektor dziecka

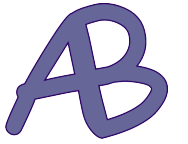


- Istnieje także możliwość napisania selektora odnoszącego się tylko do elementów będących dziećmi (bezpośrednimi potomkami) innego elementu. Służy do tego znak >

```
p > em {color: green;}
```

Uwaga:

Przeglądarka Internet Explorer 6 całkowicie je ignoruje (natomiast w Internet Explorerze 7 i nowsze są już obsługiwane)



Wspólne selektory dla wielu znaczników

- Możliwe jest zdefiniowanie stylu dla grupy znaczników

Na przykład poniższy styl może zostać użyty do określenia domyślnej czcionki witryny:

```
body, p, td, th, div, dl, ul, ol
{
    font-family: Tahoma, Verdana, Arial, Helvetica,
    sans-serif;
    font-size: 1em;
    color: #000000;
}
```

SELEKTOR UNIWERSALNY

Selektor uniwersalny `*` (potocznie nazywany gwiazdką) oznacza wszystko. Dlatego jeśli w arkuszu stylów użyjemy następującej reguły:

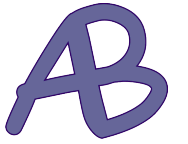
```
* {color:green;}
```

tekst na całej stronie będzie zielony, chyba że zostaną utworzone dodatkowe reguły określające inaczej

Interesującym zastosowaniem tego selektora jest użycie go jako odwrotności selektora dziecka:

```
p * em {font-weight:bold;}
```

Powyższy selektor wybiera wszystkie znaczniki `em`, które są nie-bezpośrednimi potomkami znacznika `p`.



Selektor braci

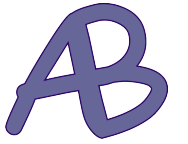
Selektor ten pozwala nadać formatowanie znacznikowi znajdującemu się za określonym znacznikiem mającym tego samego rodzica (czyli znajdującym się na tym samym poziomie w hierarchii dokumentu). Oto przykład jego użycia:

```
h1 + p (font-variant: small-caps;)
```

Jest to dobry sposób na pogrubienie pierwszego elementu listy. Na przykład:

```
ul + li (font-weight: bold;)
```

Selektor braci działa w przeglądarkach zgodnych ze standardami, przykładowo działa w IE 7 i nowszych, nie działa natomiast w IE 6.



Identyfikatory (wprowadzenie)

- Identyfikatory pisze się podobnie jak klasy, z tą różnicą, że zamiast kropką oznacza się je znakiem # w arkuszu stylów.

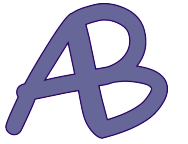
Jeśli akapit ma identyfikator jak poniżej:

```
<p id="specialtext">To jest specjalny tekst.</p>
```

odpowiadający mu selektor potomka wygląda następująco:

```
p#specialtext {reguły stylistyczne}
```

- Poza tym identyfikatory działają tak samo jak klasy — wszystko co zostało powiedziane do tej pory na temat klas, ma także zastosowanie do identyfikatorów. Co je zatem różni?



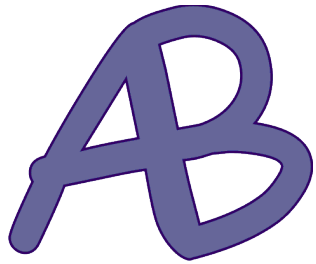
Identyfikatory (wprowadzenie)

- Zgodnie z zasadami XHTML dany identyfikator (np. id="mainmenu") może wystąpić w dokumencie **tylko raz**, natomiast klasa — dowolną liczbę razy.
- Aby zatem oznaczyć **unikatową część strony**, na przykład menu nawigacyjne, do którego ma być zastosowany specjalny zestaw reguł CSS, można użyć znacznika div z identyfikatorem.
- Do identyfikacji kilku specjalnych akapitów na stronie, które wymagają tego samego stylu różniącego się od stylu głównego akapitów, należy użyć klasy
- Dodatkowo identyfikatory służą do identyfikacji znaczników w języku JavaScript

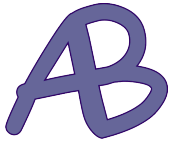
Identyfikatory (wprowadzenie)

- Selektory ID mogą być stosowane w połączeniu z innymi rodzajami selektorów. Na przykład w poniższym stylu selektor ID odnosi się do nieodwiedzonych łączy występujących w akapicie `pasek1`:

```
#pasek1 a:link {  
  font-weight: bold;  
  color: #FFFFFF: }
```



II. Pseudoklasy i pseudoelementy



Czym są pseudoklasy?

- ✓ Nazwa pseudoklasy wzięła się stąd że są to klasy niezwiązane z żadnym konkretnym znacznikiem.
- ✓ Pseudoklasy powodują zastosowanie reguł CSS do elementów, jeśli nastąpią określone zdarzenia.

Pseudoklasy odnośników

Istnieją cztery pseudoklasy, których można używać do formatowania łączy, ponieważ łącza mogą znajdować się w jednym z czterech stanów:

- **Link** — zwykły odnośnik, wyglądający jak odnośnik i czekający, aż go ktoś kliknie.
- **Visited** — odnośnik, który został już wcześniej kliknięty przez użytkownika.
- **Hover** — kursor znajduje się nad odnośnikiem.
- **Active** — odnośnik jest właśnie klikany



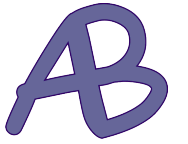
Pseudoklasy odnośników

Istnieją cztery pseudoklasy, których można używać do formatowania łączy, ponieważ łączy mogą znajdować się w jednym z czterech stanów:

- **Link** — zwykły odnośnik, wyglądający jak odnośnik i czekający, aż go ktoś kliknie.
- **Visited** — odnośnik, który został już wcześniej kliknięty przez użytkownika.
- **Hover** — kursor znajduje się nad odnośnikiem.
- **Active** — odnośnik jest właśnie klikany

Składnia:

```
a:link {color:black;}  
a:visited {color:gray;}  
a:hover {text-decoration:none;}  
a:active {color:navy;}
```

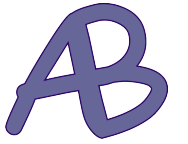



Pseudoklasa HOVER

Za pomocą tej klasy można tworzyć efekty rollover także dla innych elementów, nie tylko dla odnośnika.

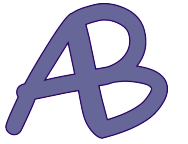
Na przykład:

```
p:hover {background-color:gray;}
```



Inne przydatne pseudoklasy

Zadaniem pseudoklas jest symulacja klas dodawanych do kodu XHTML w chwili wystąpienia określonych warunków. Poza tym, że mogą być stosowane w odpowiedzi na konkretne zdarzenia, jak najeżdżanie kursorem czy klikanie, mogą także być włączane na podstawie spełnienia określonych warunków w kodzie XHTML



Inne przydatne pseudoklasy

`x : first-child`

Pseudo klasa rozpoznający pierwszy element dziecko o nazwie x

```
div strong:first-child {color:blue}
```

Wykrywa **pierwszy** znacznik `` zawarty w znaczniku `<div>`

`x : last-child`

Pseudo klasa rozpoznający ostatni element dziecko o nazwie x

```
div strong:last-child {color:red}
```

Wykrywa **ostatni** znacznik `` zawarty w znaczniku `<div>`

Inne przydatne pseudoklasy

Pseudoklasa :focus

Kiedy użytkownik klika na element taki jak, np. pole tekstowe formularza element staje się aktywny, czyli uzyskuje Focus.

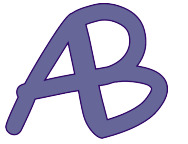
```
input:focus {background-color: burlywood;}
```

Gdy element typu input otrzyma focus i jego tło zmieni kolor

Zastosowanie:

- element button
- element input
- element select
- element textarea
- element a
- element area
- element HTML posiadający w danym momencie atrybut contenteditable o wartości true

Pseudoelementy



Pseudoelementy pozwalają uzyskać efekt pojawiania się dodatkowych elementów w dokumencie, pomimo, że w rzeczywistości nic nie dodano.

Pseudoelement „pierwsza litera”.

x:first-letter

Na przykład:

```
p:first-letter {font-size:300%; float: left;}
```

Powyższy kod powoduje, że pierwsza litera na początku akapitu będzie dużo większa od pozostałych.

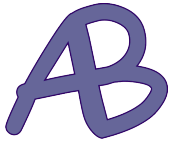
Pseudoelement „pierwsza linia”

x:first-line

Poniższy pseudoelement pozwala stylizować pierwszą linię tekstu (zazwyczaj) akapitu. Na przykład:

```
p:first-line{font-variant: small-caps;}
```

Powyższy kod spowoduje, że pierwsza linia akapitu będzie napisana kapitalikami.



Pseudoelementy

Pseudoelementy **:before** i **:after** pozwalają na wstawianie tekstu przed i za elementem.

Na przykład kod XHTML:

```
<h1 class="age">25</h1>
```

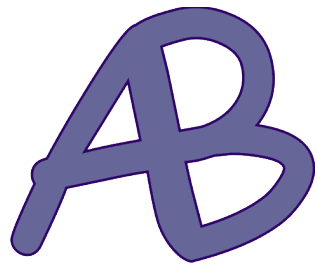
w połączeniu z poniższymi regułami stylistycznymi:

```
h1.age:before {content:"Wiek;  "}
```

```
h1.age.-after {content:" lat.  " }
```

da w rezultacie tekst Wiek: 25 lat.

Ponieważ wyszukiwarki nie potrafią dotrzeć do treści pseudoelementów (nie ma ich w kodzie XHTML), nie należy za ich pomocą wstawiać ważnej treści, która powinno zostać zaindeksowana przez wyszukiwarki.



III. Przykłady

Przykład I: System nawigacji

Jak za pomocą CSS zastąpić system nawigacji oparty na rysunkach?

Wykorzystanie plików graficznych do tworzenia przycisków nawigacji to wciąż bardzo popularna metoda tworzenia systemu nawigacji. Z takim rozwiązaniem wiąże się jednak szereg problemów:

- ✓ Dodanie nowego przycisku oznacza konieczność utworzenia nowej grafiki.
- ✓ Kłopoty w sytuacji gdy menu nawigacji tworzone jest dynamicznie, na przykład na podstawie bazy danych.
- ✓ Użytkownicy, którzy używają aplikacji do odczytywania zawartości witryny na głos, nie będą w stanie odczytać tekstu umieszczonego na przycisku.
- ✓ Każdy dodatkowy rysunek umieszczony na stronie wydłuża czas jej ładowania.

System nawigacji

HTML

```

<body>
<table id="nawigacja">
  <tr>
    <td>
      <a href="#">Przepisy</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#">Napisz do nas</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#">Artykuły</a>
    </td>
  </tr>
  <tr>
    <td>
      <a href="#">Sklep internwtow</a>
    </td>
  </tr>
</table>
</body>
</html>

```

CSS

```

#nawigacja {
  width: 180px;
  padding: 0;
  margin: 0;
  border-collapse: collapse;
}
#nawigacja td {
  height: 26px;
  border-bottom: 2px solid #460016;
  background-color: #FFDFEA;
  color: #460016;
}
#nawigacja a:link, #nawigacja a:visited {
  margin-left: 12px;
  color: #460016;
  background-color: transparent;
  font-size: 12px;
  font-family: Arial, Helvetica, sans-serif;
  text-decoration: none;
  font-weight: bold;
}

```

Przepisy
Napisz do nas
Artykuły
Sklep internetowy

System nawigacji - analiza

Na początek nadajmy tabeli nawigacyjnej ID, co pozwoli na jej identyfikację w dokumencie. Następnie zdefiniujemy selektory CSS dla poszczególnych elementów tabeli.

Identyfikator tabeli nazwiemy nawigacja, co dodatkowo oszczędzi nam określania atrybutów znacznika `<table>`.

```
<table id="nawigacja">
```

System nawigacji - analiza

Listing poniżej przedstawia kod CSS odpowiedzialny za wygląd całej tabeli.

```
#nawigacja {  
width: 180px;  
padding: 0;  
margin: 0;  
border-collapse: collapse: }
```

Ustawienie właściwości **border-collapse** na **collapse** spowoduje, że komórki będą ze sobą złączone tak, że pomiędzy nimi widoczna będzie tylko jedna ramka. Domyślnie bowiem każda komórka posiada osobną ramkę, a między poszczególnymi komórkami widoczne są niewielkie marginesy.

System nawigacji - analiza

Teraz musimy zdefiniować styl znaczników tabeli `<td>`. Każda komórka ma mieć określony kolor wypełnienia i widoczną dolną krawędź

```
#nawigacja td {  
height: 26px;  
border-bottom: 2px solid #460016;  
background-color: #FFDFEA;  
color: #460016; }
```

System nawigacji - analiza

Nadszedł czas na zdefiniowanie stylu łączy w komórkach tabeli. Musimy wprowadzić lewy margines, aby odsunąć tekst od krawędzi ramek, a także określić kolor, rozmiar i rodzinę czcionek oraz ewentualne efekty dla czcionki wykorzystanej w łączach. Dodatkowo z łączy usuniemy podkreślenie.

```
#nawigacja a:link, #nawigacja a:visited {  
margin-left: 12px;  
color: #460016;  
background-color: transparent;  
font-size: 12px;  
font-family: Arial, Helvetica, sans-serif;  
text-decoration: none;  
font-weight: bold; }
```

Rozwiązanie w pliku zastap_obrazki.html

Przykład II: Efekt najazdu na rysunkach

System nawigacji zbudowany z wykorzystaniem CSS może oferować wiele interesujących efektów, jednak wciąż istnieją takie, które wymagają zastosowania rysunków. Jak zatem stosować pliki graficzne, nie rezygnując z zalet nawigacji opartej na zwykłym tekście?

Oto menu, do którego dodamy grafikę:

```
<ul id="naw">
  <li><a href="#">Przepisy</a></li>
  <li><a href="#">Napisz do nas</a></li>
  <li><a href="#">Artykuły</a></li>
  <li><a href="#">Sklep internetowy</a></li>
</ul>
```

Efekt najazdu na rysunkach

Do wykonania efektu potrzebny będzie obrazek, a właściwie trzy obrazki odpowiadające trzem stanom łącza, ale wstawione w jeden rysunek.

Rozwiązanie:

```
ul#naw {  
    list-style-type: none;  
    padding: 0;  
    margin: 0;}  
#naw a:link, #naw a:visited {  
    display: block;  
    width: 150px;  
    padding: 10px 0 16px 32px;  
    font: bold 80% Arial, Helvetica, sans-serif;  
    color: #FF9900;  
    background: url("papryki.gif") top left no-repeat;  
    text-decoration: none;}  
#naw a:hover {  
    background-position: 0 -69px;  
    color: #B51032;}  
#naw a:active {  
    background-position: 0 -138px;  
    color: #006E01;}
```



Efekt najazdu na rysunkach - analiza

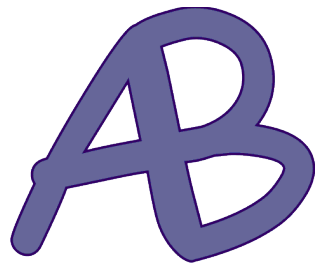
```
#naw a:hover {  
    background-position: 0 -69px;  
    color: #B51032;}  
#naw a:active {  
    background-position: 0 -138px;  
    color: #006E01;}
```



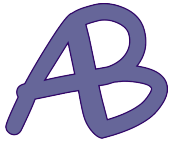
Stan **:hover** powoduje przesunięcie tła w górę o liczbę pikseli niezbędną do odsłonięcia czerwonej papryki. W moim przypadku musiałam dokonać przesunięcia o -69 pikseli, ale w innych zależy to będzie od rozmiarów grafiki. Wymaganą wartość możesz obliczyć lub znaleźć metodą prób i błędów.

Stan aktywny powoduje ponowne (-138) przesunięcie tła i odsłonięcie po kliknięciu łącza papryki w zielonej wersji.

Wykład 2 - część IV.



IV. Różne...



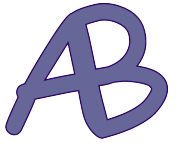
Jak zmienić lub usunąć znaki punktora na liście elementów?

Wygląd punktora listy elementów można zmodyfikować za pomocą właściwości

list-style-type

Oto dopuszczalne wartości właściwości list-style-type:

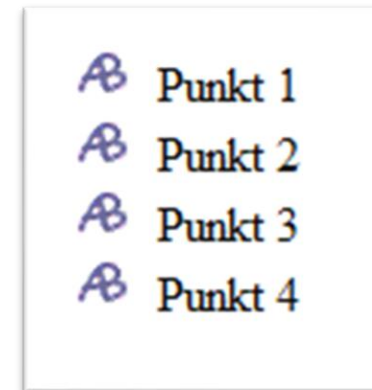
- disc,
- circle,
- decimal-leading-zero,
- decimal,
- lower-roman,
- upper-roman,
- lower-greek,
- lower-alpha,
- lower-latin,
- upper-alpha,
- upper-latin,
- Hebrew,
- Armenian,
- Georgian,
- none.
















Jak zdefiniować punktory użytkownika?

Aby zastosować własny obrazek jako punktory listy numerowanej, należy zamiast właściwości `list-style-type` użyć właściwości **`list-style-image`**, która pozwala podać adres URL zawierający ścieżkę i nazwę pliku graficznego.









```
ul{list-style-image: url(plik.gif);}
```



Właściwość **cursor** może przyjmować szereg wartości reprezentujących ikony

Wartość cursor	Wygląd kursora (IE 6)	Wersja IE (Win)	Wersja IE (Mac)	NS/Mozilla
auto	brak	4	4	6/1
crosshair	+	4	4	6/1
default		4	4	6/1
e-resize		4	4	6/1
help		4	4	6/1
move		4	4	6/1
n-resize		4	4	6/1
ne-resize		4	4	6/1
nw-resize		4	4	6/1
pointer		4	4	6/1
s-resize		4	4	6/1
se-resize		4	4	6/1
sw-resize		4	4	6/1
text	I	4	4	6/1
url	brak	6	—	—
w-resize		4	4	6/1
wait		4	4	6/1

Kursory dostępne wyłącznie w przeglądarce Internet Explorer

Wartość cursor	Wygląd (w IE 6)	Wersja IE (Win)	Wersja IE (Mac)
all-scroll		6	—
col-resize		6	—
hand		4	4
no-drop		6	—
not-allowed		6	—
progress		6	—
row-resize		6	—
vertical-text		6	—

Literatura:

W prezentacji wykorzystano fragmenty i zadania z książek:

- Andrew R.; *CSS.Antologia. 101 wskazówek i tricków*. Helion, Gliwice 2005.
- Wyke-Smith Ch.; *CSS Witryny szyte na miarę*. Helion, Gliwice 2008.
- Sokół M.; *CSS Ćwiczenia*. Helion, Gliwice 2007.