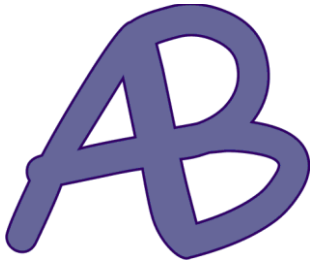


II. Pseudoklasy i pseudoelementy

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 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

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-of-type

Pseudoklasa rozpoznający pierwszy element określonego typu (x)
będący pierwszym elementem podrzednym

```
p:first-of-type {color:blue}
```

Wykrywa **pierwszy** akapit <p> zawarty w jakimś bloku

```
article p:first-child {color:blue}
```

Wykrywa **pierwszy** akapit <p> zawarty w artykule

x : last-of-type

Pseudo klasa rozpoznający ostatni element dziecko o nazwie x

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

Wykrywa **ostatni** znacznik <p> zawarty w bloku

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

x : last-child

Pseudo klasa rozpoznający ostatni element dziecko o nazwie x

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

Wykrywa **ostatni** znacznik
zawarty w znaczniku

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

x : last-child

Pseudo klasa rozpoznający ostatni element dziecko o nazwie x

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

Wykrywa **ostatni** znacznik
zawarty w znaczniku

Inne przydatne pseudoklasy

x : nth-of-type(n)

Pseudo klasa rozpoznający n-ty element określonego typu (x) będący elementem podrzędnym

```
p:nth-of-type(2) {color:blue}
```

Wykrywa **drugi** akapit należący do większego bloku

Słowne wartości parametru:

- even – elementy parzyste
- odd – elementy nieparzyste

Numer elementów podrzędnych możemy określać też za pomocą „n” (oznacza dowolny element)

- n + 5 – wszystkie elementy począwszy od piątego elementy
- 2n – co drugi element
- 3n – co trzeci element
- 2n_5 – co drugi element począwszy od piątego

Inne przydatne pseudoklasy

:not(x)

Pseudoklasa wykluczająca element określonego typu

```
:not(div) {color:blue}
```

Wykrywa wszystkie elementy nie będące elementem

x:not(.klasa)

Pseudoklasa wykluczająca elementy typu „x” ale należące do klasy „klasa”

```
p:not(.klasa) {color:blue}
```

Wykrywa wszystkie elementy typu <p> nie należące do klasy „klasa”

Inne przydatne pseudoklasy

x:only-child

Jedyny element podrzędny

```
:only-child {color:blue}
```

Wykrywa wszystkie elementy które są jedynym elementem podrzędnym w jakimś kontenerze

x:only-of-type(

Jedyny element podrzędny danego typu

```
a:only-of-type {color:blue}
```

Wykrywa wszystkie elementy typu <a> które są jedynym elementem podrzędnym danego typu w kontenerze

Inne przydatne pseudoklasy

x:empty

Pusty element typu x

```
div:empty {color:blue}
```

Wykrywa wszystkie puste kontenery

:empty

Pusty element typu

```
:empty {color:blue}
```

Wykrywa wszystkie puste elementy

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 lnia”

```
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:

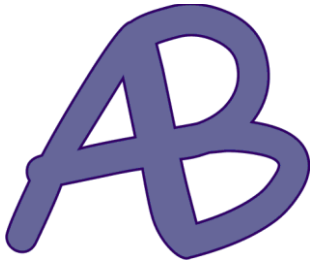
```
<hl class="age">25</hl>
```

w połączeniu z poniższymi regułami stylistycznymi:

```
hl.age::before {content:"Wiek;  "}
hi.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.

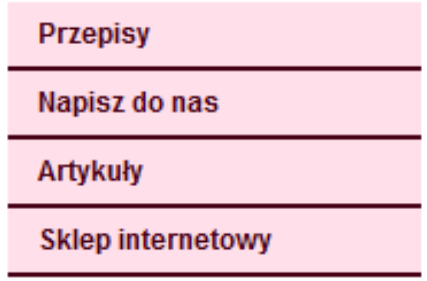
System nawigacji

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;
}
```



System nawigacji - analiza

Na początek nadajmy tabeli nawigacyjnej ID, co pozwoli na jej identyfikację w dokumencie. Następnie zdefiniujmy 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: nonę:
font-weight: bold; }
```

Rozwiązanie w pliku zastąp_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 zależeć 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.

Literatura:

W prezentacji wykorzystano fragmenty i zadania z książek:

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