Zaawansowany HTML i CSS - dzień 2

v. 1.8

Coders Lab
SZKOŁA PROGRAMOWANIA

Plan

1. Box model w CSS
2. Zaawansowany box model
3. Border i box-shadow
4. Pozycjonowanie elementów
5. Zaawansowane Selektory CSS

Box model w CSS

Jak elementy są wyświetlane?

Ważne

W HTML mamy dwa typy znaczników:

- **blokowe** – zajmują całą dostępną przestrzeń
- i zaczynają się na początku nowej linii (chyba że stylowanie CSS to zmienia),
- **inline** – zajmują tylko tyle przestrzeni, ile potrzebują do wyświetlenia swojej zawartości.

Dzięki właściwości **display** możemy zmieniać podstawowe wyświetlanie danego elementu.

```
p {
  display: block;
}

div {
  display: inline;
}

h1 {
  display: none;
  /* element nie będzie wyświetlany */
}
```

Jak elementy są wyświetlane?

Zmiana wyświetlania – inline-block

Właściwość **display** może przyjąć wartość **inline-block**. Wtedy taki element zachowuje wszystkie właściwości **blokowe**, a zarazem jest wyświetlany w linii z innymi.

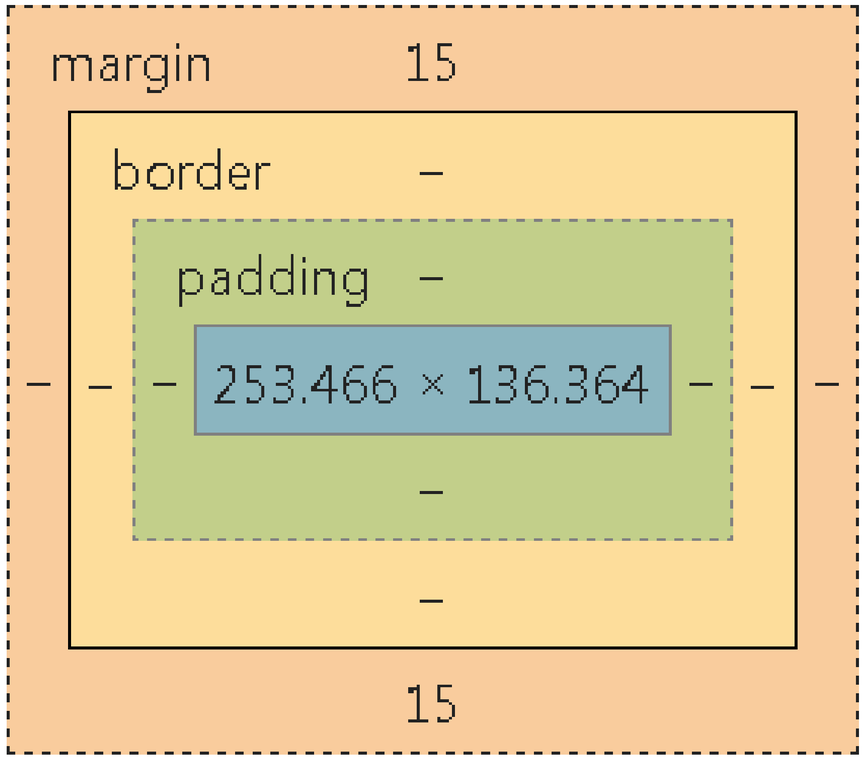
```
p {
  display: inline-block;
}
```

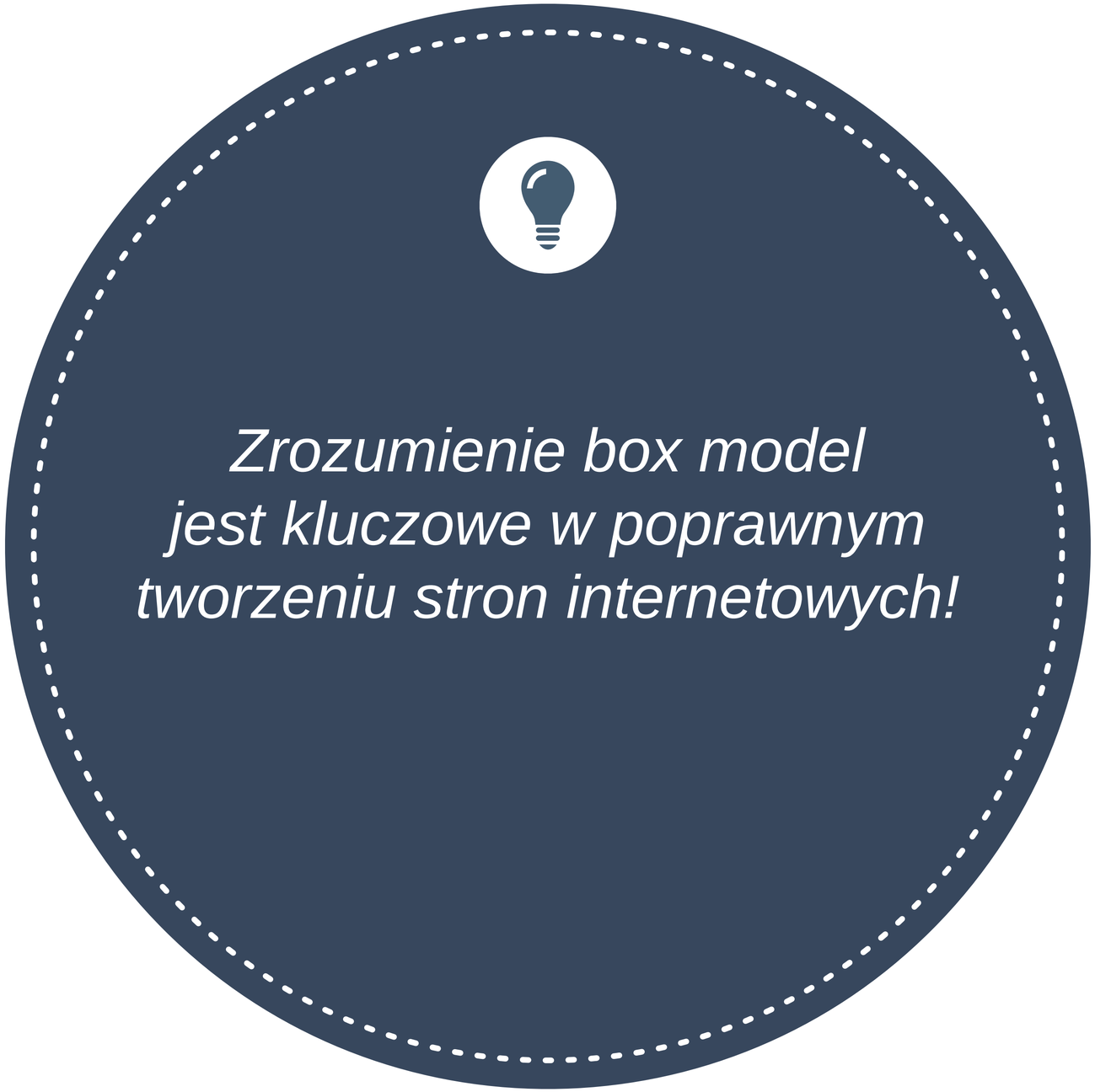
box model

Box-model

Każdy element w CSS można uznać za pudełko, któremu możemy nadać wielkość i pozycję oraz ustawić tło.

Każdy box składa się z następujących warstw:
- **width** (całkowita szerokość elementu),
- **height** (całkowita wysokość elementu),
- **padding** (odległość od zawartości do bordera),
- **border** (obramowanie zawartości),
- **margin** (odległość od obramowania do następnego boksa).





Zrozumienie box model
jest kluczowe w poprawnym
tworzeniu stron internetowych!

Atrybuty width oraz height

Wielkości

width

Atrybut **width** ustawia bezwzględną szerokość boksu. Przy elementach typu **inline** nie jest brany pod uwagę.
Dla atrybutów typu **block** domyślnie jest ustawiony na 100%.

height

Atrybut **height** ustawiający bezwzględną wysokość boksa. Przy elementach typu **inline** nie jest brany pod uwagę.

```
div {
   width: 350px;
}

aside {
   height: 200px;
}
```

Atrybut margin – margines zewnętrzny

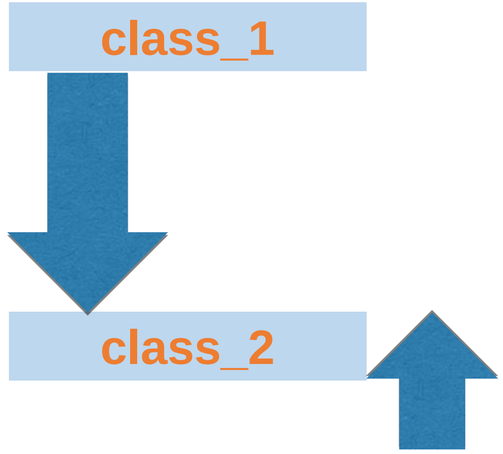
Odsuwanie

Atrybut ustawiający odległość od obramowania boxa do następnego elementu. Przy elementach typu **inline** działa tylko na szerokość.

Jeżeli obok siebie są dwa różnie wystylowane elementy, to odległość między nimi będzie równa większej wartości **margin**.

```
.class_1 {
   margin-bottom: 30px;
}

.class_2 {
   margin-top: 15px;
}
```



Możliwe deklaracje atrybutu margin

Odsuwanie

W pierwszym przykładzie jest zdefiniowana taka sama odległość dla wszystkich kierunków.

W drugim są osobno zdefiniowane odległości horyzontalne i wertykalne.

W trzecim i czwartym – odległości dla każdej ze stron.

```
.class_1 {
   margin: 12em;
}
.class_2 {
   margin: 12em 6em;
}
.class_3 {
   margin: 12em 5em 3em 6em;
}
.class_4 {
   margin-top: 12em;
   margin-right: 5em;
   margin-bottom: 3em;
   margin-left: 6em;
}
```

Atrybut padding – margines wewnętrzny

Rozpychanie

Atrybut ustawiający odległość od zawartości boksu do jego obramowania.

Często dla tego atrybutu ustawiamy odległości wyznaczone w **em**.

```
div {
   padding: 12em;
}

div {
   padding: 12em 5em 3em 6em;
}
```

Możliwe deklaracje atrybutu padding

Rozpychanie

W pierwszym przykładzie jest zdefiniowana taka sama odległość dla wszystkich kierunków.

W drugim są osobno zdefiniowane odległości horyzontalne i wertykalne.

W trzecim i czwartym – odległości dla każdej ze stron.

```
.class_1 {
   padding: 12em;
}
.class_2 {
   padding: 12em 6em;
}
.class_3 {
   padding: 12em 5em 3em 6em;
}
.class_4 {
   padding-top: 12em;
   padding-right: 5em;
   padding-bottom: 3em;
   padding-left: 6em;
}
```

Atrybut border

Obramowanie

Atrybut **border** opisuje, jak zachowuje się obramowanie elementu.

Atrybut ten przyjmuje po kolei wartości:
- grubość obramowania,
- styl obramowania,
- kolor obramowania.

http://jsfiddle.net/CodersLab/ukhkuewc

Możemy naszemu elementowi nadać następujące obramowanie:
- **dotted** (kropkowane),
- **dashed** (kreskowane),
- **solid** (ciągłe),
- **double** (dwie linie).

Dostępne są też różne kształty 3D, których jednak nie stosuje raczej przy płaskim designie.

Przykładowe deklaracje border

Obramowanie

W pierwszym przykładzie jest zdefiniowane obramowanie o grubości 2px, ciągłym wypełnieniu i czerwonym kolorze

W drugim przykładzie zdefiniowane jest w podobny sposób obramowanie dolne

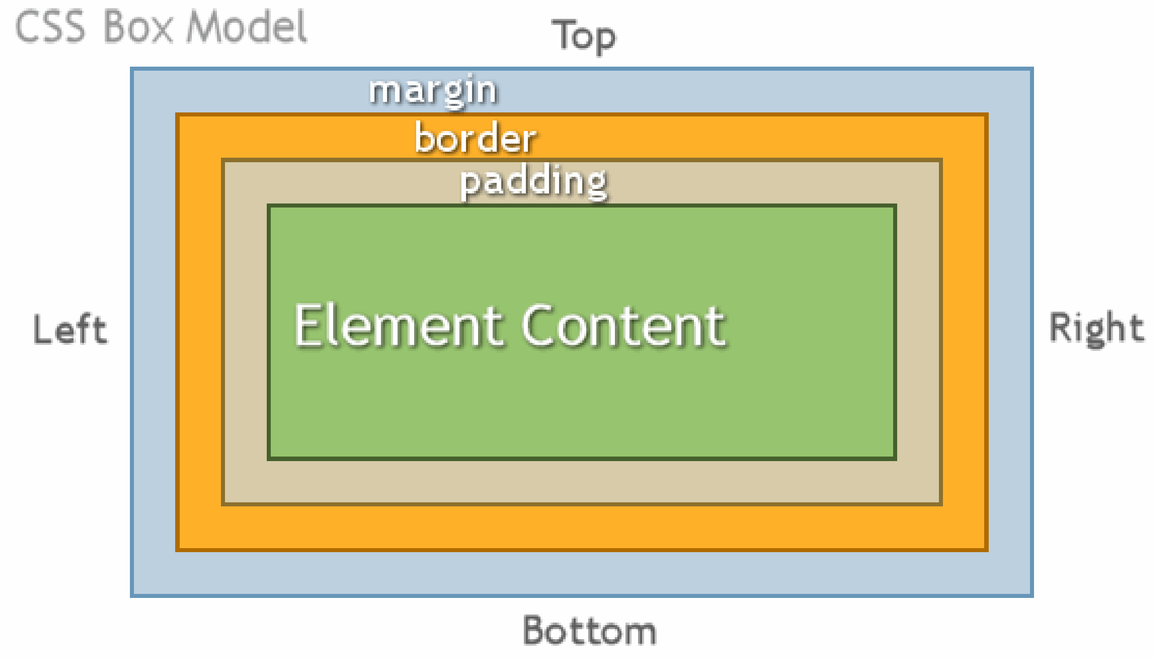
W trzecim przykładzie każda opcja zapisana jest osobno.

```
.class_1 {
  border: 2px solid red;
}

.class_2 {
  border-bottom: 2px solid #949599;
}

.class_3 {
  border-style: solid;
  border-width: 4px;
  border-bottom-width: 12px;
  border-left-color: green;
}
```

Zrozumienie wielkości boxa




```
div {
    height: 150px;
    width: 400px;
    border: 2px solid #949599;
    padding: 10px;
    margin: 15px;
}
```

- http://codepen.io/carolineartz/full/ogVXZj

Całkowita wielkość boxa

Wcale nie takie skomplikowane

Całkowitą szerokość możemy obliczyć wzorem:

margin-right + **border-right** + **padding-right** + **width** + **padding-left** + **border-left** + **margin-left**

Całkowitą wysokość możemy obliczyć wzorem:

margin-top + **border-top** + **padding-top** + **height** + **padding-bottom** + **border-bottom** + **margin-bottom**

Przepełnienie – overflow

Możliwość tworzenia tzw. przycinania

W CSS możemy kontrolować sposób, w jaki dany element radzi sobie z zawartością, w wypadku gdy przekracza ona dostępną wielkość elementu.

Do tego celu służy nam właściwość **overflow**.

Właściwość **overflow** przyjmuje następujące wartości:

- **visible**,
- **auto**,
- **scroll**,
- **hidden**,
- **inherit**.

```
div {
  overflow: hidden;
}
```

Zadania

Zaawansowany box model

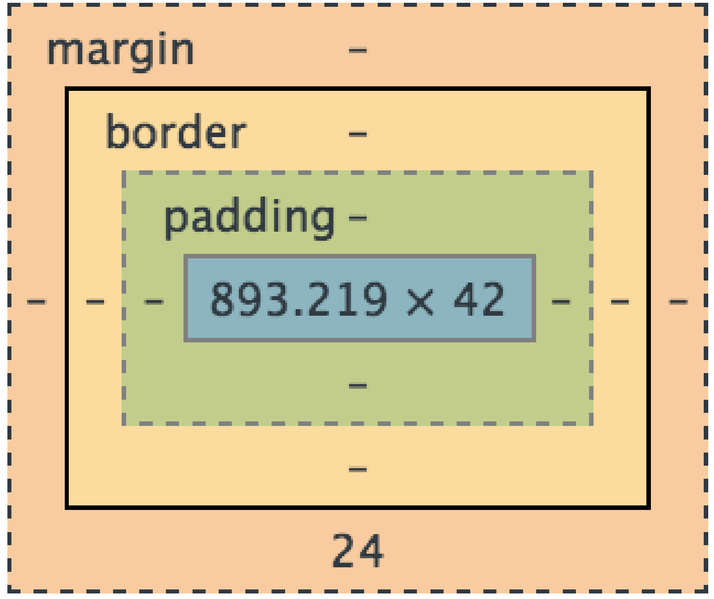
box-sizing

Ważne

Dzięki atrybutowi **box-sizing** możemy sterować tym, jak wyliczana jest długość i szerokość elementu, który definiujemy.

Atrybut **box-sizing** przyjmuje dwie wartości:

- **content-box - domyślna**,
- **border-box.**



*Wartość **content-box***
jest domyślną wartością
dla wszystkich elementów.

*Tylko **width** i **height***
ustawione przez nas
odpowiadają
wielkości elementu.

border-box

border-box

Wartość **border-box** jest najwygodniejszą możliwością ustawienia atrybutu **box-sizing**.

Oznacza to, że **width** i **height** wraz z **paddingiem i borderem** odpowiadają wielkości elementu.

Szerokość i wysokość zawartości możemy wyliczyć następująco:

width – **padding** – **border**,
height – **padding** – **border**.

Zadania

Wykonaj zadania z folderu

07.Zaawansowany_box_model

Border
i box-shadow

ZAAWANSOWANY ATRYBUT BORDER

Atrybut border-radius

Zaokrąglone rogi

Dzięki temu atrybutowi możemy ustawiać zaokrąglenie całego boksu.

Przykłady zastosowania atrybutu:

- http://jsfiddle.net/CodersLab/307c4z1v

Prezentacja Lei Verou:

- http://lea.verou.me/humble-border-radius

```
.class_1 {
  border-radius: 50%;
}

.class_2 {
  border-radius: 20em 10em 1em 20em;
}

.class_3 {
  border-top-left-radius: 20em;
  border-top-right-radius: 10em;
  border-bottom-right-radius: 1em;
  border-bottom-left-radius: 20em;
}
```

ATRYBUT BOX-SHADOW

Atrybut box-shadow

Cień pod elementem

Atrybut **box-shadow** służy do tworzenia cieni znajdujących się za elementem.
Przyjmuje on od dwóch do czterech wartości.
Dwa pierwsze podają położenie cienia. Trzeci – rozmycie, a czwarty rozszerzanie się cienia.

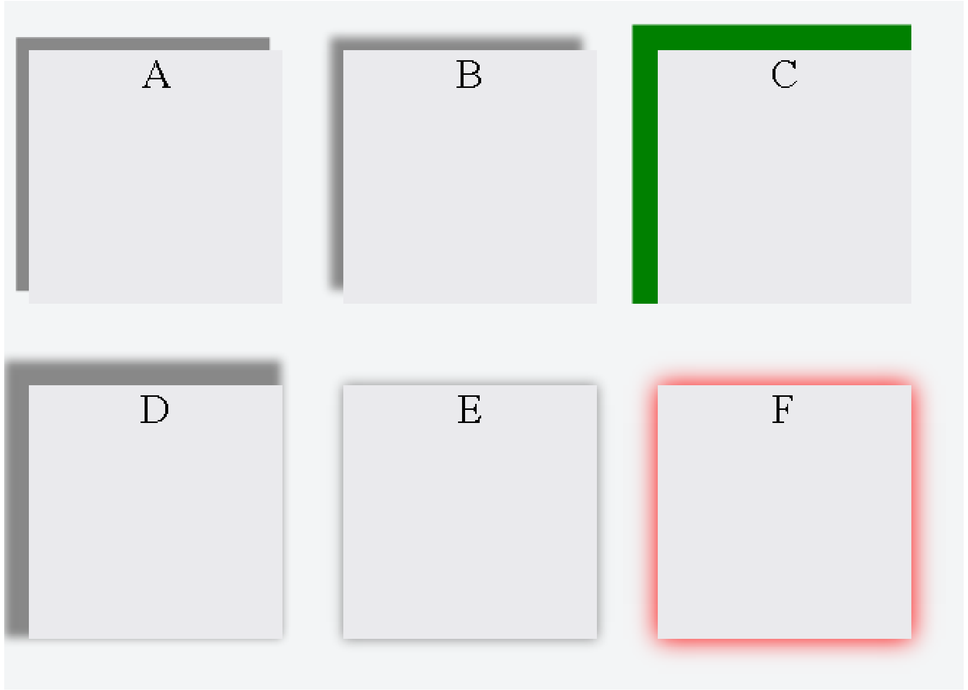
Przykłady zastosowania atrybutu

- http://jsfiddle.net/CodersLab/fwd35nxr

Generator cienia

- http://www.cssmatic.com/box-shadow

```
div {
  box-shadow: -5px -5px 10px 10px #888;
}
```



box-shadow a border-radius

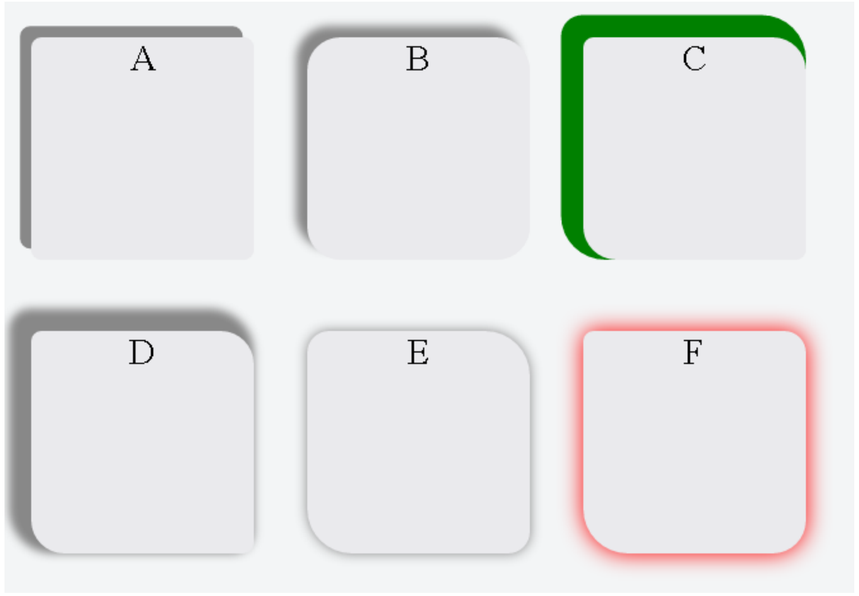
Cień pod elementem i zaokrąglone rogi

Na cień naszego elementu ma wpływ też kształt krawędzi.
Jeżeli zatem je zaokrąglimy, nasz cień też przyjmie taki kształt.

Przykład zastosowania parametru:

- http://jsfiddle.net/CodersLab/ts40jppy

```
div {
  border-radius: 5%;
  box-shadow: -5px -5px 10px 10px #888 inset;
}
```



Filtry graficzne

Sprawdź jak wspierane są filtry w przeglądarkach

Ciekawą regułą CSS są filtry graficzne, które mogą zastąpić podstawowe funkcje programów do edycji zdjęć. Do dyspozycji mamy m.in. następujące efekty:

- rozmycie (**blur**),
- jasność (**brightness**) oraz kontrast (**contrast**),
- skala szarości (**greyscale**) oraz sepia (sepia).

Cień możemy zatem uzyskać również dzięki odpowiedniemu filtrowi.

Więcej o filtrach:

- http://www.cssreflex.com/css-generators/filter/
- http://www.html5rocks.com/en/tutorials/filters/understanding-css

```
div {
   -webkit-filter:
   drop-shadow(10px 10px 8px rgba(0, 0, 0, .5));
   filter:
   drop-shadow(10px 10px 8px rgba(0, 0, 0, .5));
}
img {
   -webkit-filter: sepia(80%);
   filter: sepia(80%);
}

img#top {
   -webkit-filter: blur(10px);
   filter: blur(10px);
}
```

Zadania

Wykonaj zadania z folderu

08.Border_i_box_shadow

Pozycjonowanie elementów

Opływanie elementów – float

Ustawiamy elementy obok siebie

Atrybut **float** pozwala ustawiać elementy blokowe w jednej linii.

Możemy ustawiać je z lewej strony lub z prawej.

```
.box-left {
  float: left;
}

.box-right {
  float: right;
}
```

Opływanie elementów – float

Ustawiamy elementy obok siebie

Element, który ma ustawiony **float**, opływa pozostałe elementy.

float: left; **float: left;** **float: right;**

Zapobieganie opływaniu – clear

Powrót do naturalnego biegu dokumentu

Do naturalnego biegu dokumentów powrócimy za pomocą właśności **clear**:

- **left** – zsuniemy elementy poniżej wszystkich, które mają być ustawione **float: left** (ale nie **float: right**),
- **right** – zsuniemy elementy poniżej wszystkich, które mają ustawione **float: right** (ale nie **float: left**),
- **both** – zsuniemy elementy poniżesz wszystkich z właściwością **float**,
- **none** – jest to domyślna wartość, która wyłącza działanie **clear** .

Możemy powrócić do naturalnego biegu dokumentu za pomocą **clear**.

```
float: left;
float: left;
float: right;
clear: both;
float: left;
```

float vs inline-block

Który sposób wybrać?

Jeżeli chcemy, aby elementy wyświetlały się obok siebie a nie opływały – używajmy **display: inline-block** zamiast **float**.

Właściwość **float** stosujemy, jeśli chcemy, aby wybrany element opływał inny.

Jakiś tekst, jakiś tekst, jakiś
Jakiś tekst, jakiś tekst, jakiś
Jakiś tekst, jakiś
Jakiś tekst, jakiś
Jakiś tekst, jakiś
Jakiś tekst, jakiś tekst
Jakiś tekst, jakiś tekst

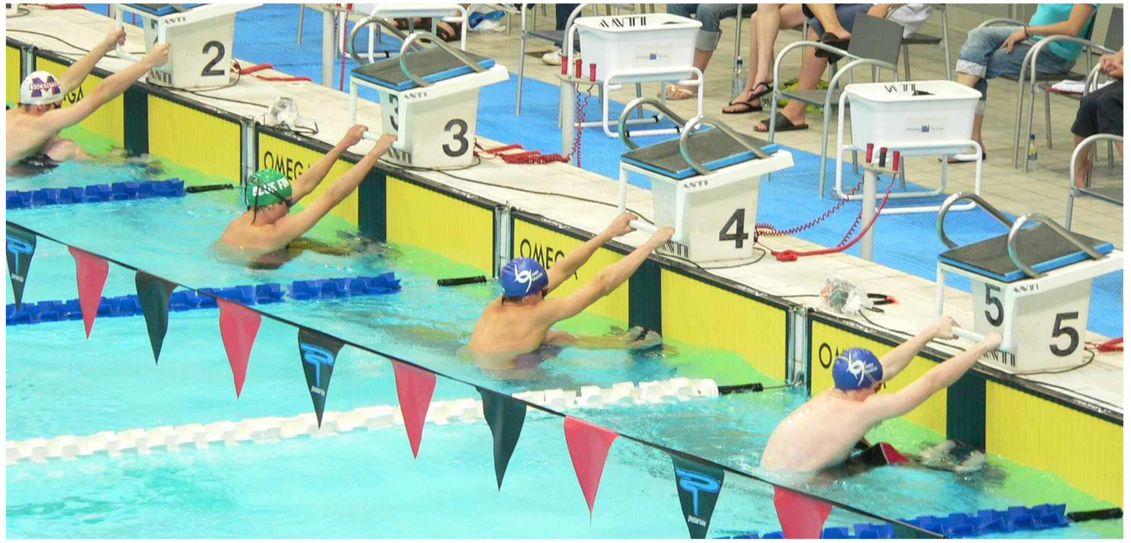
position

Pozycja

Domyślnie każdy element ma ustawioną pozycję statyczną – **static**.
Jest to naturalne ułożenie elementów na stronie, czyli od lewej do prawej (**inline**) i od góry do dołu (**block**).

Inne wartości **position** to:

- relative,
- absolute,
- fixed.



position: relative

Pozycja i przesuwanie

Jest to podobne ułożenie jak **static**, z tym że możemy przesuwać elementy o offset (box offset) za pomocą
własności:

- **top**,
- **right**,
- **bottom,**
- **left.**

```
.box-green {
  position: relative;
  top: 50px;
  left: 50px;
}
```

position: absolute

Pozycja i przesuwanie

Elementy z ustawionym **position absolute** są wyjęte z biegu dokument i akceptują przesunięcie o offset (box offset) tak jak relative:

- **top**,
- **right**,
- **bottom,**
- **left.**

```
.box-green {
  position: absolute;
  top: 0;
  left: 0;
}
```

position: fixed

Pozycja i przesuwanie

Elementy z ustawionym **position fixed** działają podobnie jak **absolute**, z tym że są pozycjonowane względem okna przeglądarki.
Podczas przewijania strony element z **position fixed** jest „przyczepiony” do okna.

- http://www.jsfiddle.net/CodersLab/eu3an0hL

```
.box-green {
  position: fixed;
  bottom: 0;
  right: 0;
}
```

position absolute + relative

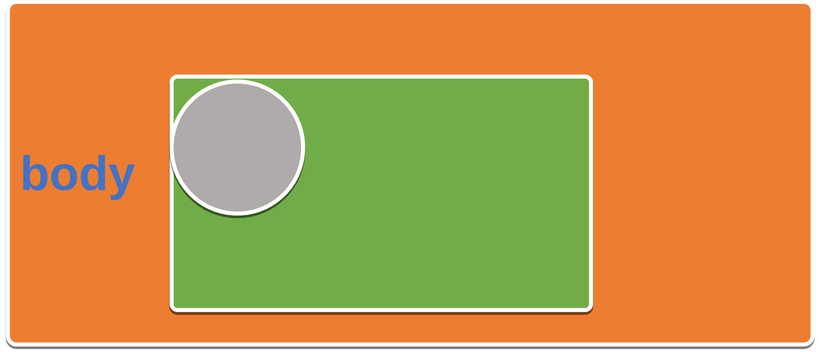
Pozycja i przesuwanie

Element, któremu nadamy **position relative** staje się kontenerem zawierającym.
Dzięki temu elementy w nim zawarte, którym nadaliśmy **position absolute**, będą ustawiane względem jego krawędzi, a nie krawędzi elementów nadrzędnych (np. body).
Rzadko używa się **position absolute** osobno.

```
<article class="box-green">
    <div class="circle-grey"></div>
</article>
```

- http://www.jsfiddle.net/CodersLab/j9vL85rx/3

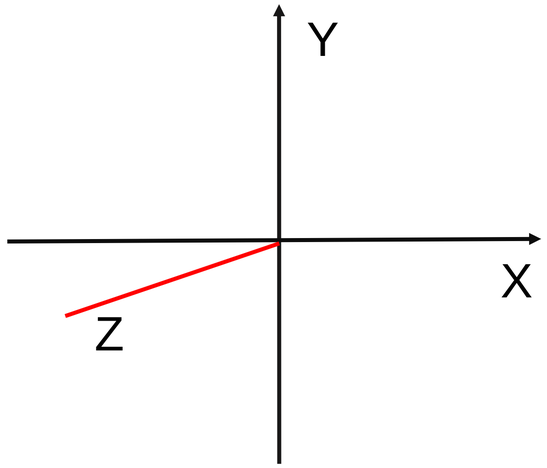
```
.box-green {
  position: relative;
  bottom: 10px;
  left: 80px;
}
.circle-grey {
   position: absolute;
}
```



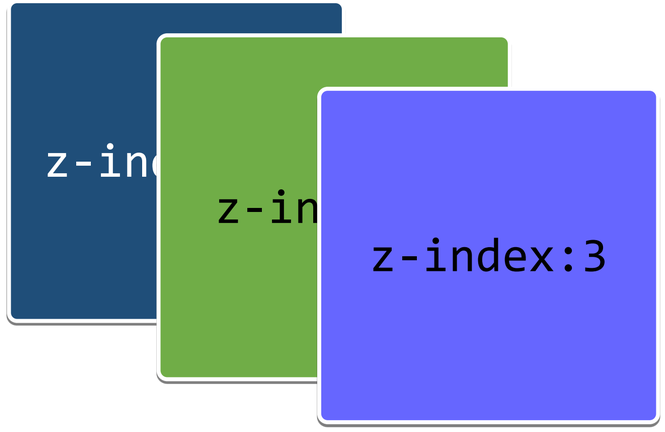
z-index

Trzeci wymiar

Do tej pory przesuwaliśmy elementy w lewo, prawo (oś X) oraz góra, dół (oś Y).
Możemy przesuwać elementy również po osi Z.



Dzięki *z-index* możemy układać elementy warstwowo. *z-index* działa dla elementów, które mają ustawioną własność *position* na *relative, fixed* lub *absolute*!



→ http://www.jsfiddle.net/CodersLab/c7f75511

Zadania

Wykonaj zadania z folderu

09.Pozycjonowanie_elementow

Zaawansowane selektory CSS

Pseudoklasy i pseudoelementy

Dodatkowe elementy

Chcesz ostylować kliknięty link czy np. zaznaczony przycisk wyboru?

Chcesz nadać kolor np. ostatniemu wierszowi w liście?

W takich przypadkach najlepiej użyć **pseudoklas** i **pseudoelementów**.

Selektor i **pseudoklasę** oddzielamy dwukropkiem.

```
a:hover {
    color: red;
}
a:visited {
    color: violet;
}
```

Dynamiczne pseudoklasy – hover

Efekt po najechaniu myszką

Dynamiczne pseudoklasy pozwalają wybrać elementy, z którymi następuje albo nastąpiła interakcja.

:hover

Dzięki ten pseudoklasie znajdziesz element, nad którym znajduje się kursor myszy.

Jest to bardzo przydatne m.in. podczas tworzenia animacji na stronie.

```
div:hover {
    background-color: red;
}
```

Dynamiczne pseudoklasy – active

Efekt po naciśnięciu

:active

Dzięki ten pseudoklasie wybierzesz element, który jest właśnie naciskany.

Używane jest to np. do stworzenia efektu wciskanego guzika (przez zmianę jego cienia).

```
button:active {
    border: 1px solid grey;
}
```

Pseudoklasy związane z linkami

Efekt po odwiedzeniu linka

:visited

Dzięki ten pseudoklasie znajdziesz wszystkie linki, które już użytkownik odwiedził.

```
a:visited {
    text-decoration: underline;
}
```

Pseudoklasy związane z linkami

Efekt wyróżniający

:focus

Dzięki tej pseudoklasie jesteśmy w stanie znaleźć element który ma **focus** (jest w jakiś sposób wybrany).

Ta pseudoklasa jest często używana do wyróżnienia wybranego elementu formularza.

```
input:focus {
    border: 2px solid red;
}
```

LoVe and HAte

Zasada kolejności

Przy stosowaniu pseudoklas dla linków należy trzymać się określonego porządku.

Prawidłowa kolejność użycia pseudoklas jest podana po prawej.

```
a:link { }
a:visited { }
a:hover { }
a:active { }
```

Pseudoklasy dla formularzy

:enabled

Dzięki tej pseudoklasie jesteśmy w stanie znaleźć element, który jest włączony.

:disabled

Dzięki tej pseudoklasie znajdziemy element, który jest wyłączony.

:checked

Dzięki tej pseudoklasie znajdziemy checkboksy, które są wybrane.

```
input:enabled {
}

input:disabled {
}

input:checked {
}
```

Pseudoklasy związane z położeniem

:first-child

Pseudoklasa, która wybiera pierwszy element potomny swojego rodzica. Przykład obok ustawi czcionkę na 1.5em elementowi p, który jest pierwszym dzieckiem elementu body

:last-child

Pseudoklasa, która wybiera ostatni element potomny swojego rodzica. Przykład obok ustawi czcionkę na 1.5em elementowi p, który jest ostatnim dzieckiem elementu body

```
<body>
    <p> paragraf 1 </p>
    <p> paragraf 2 </p>
    <p> paragraf 3 </p>
</body>
 p:first-child {
    font-size: 1.5em;
 }
 p:last-child {
    font-size:  1.5em;
 }
```

Pseudoklasy związane z położeniem

:nth-child()

Pseudoklasa, która wybiera n-ty element potomny swojego rodzica.
Przykład 1. obok ustawi czcionkę na 1.5em elementom p, które są nieparzystymi dziećmi elementu body

Przykład 2. obok ustawi czcionkę na 1.5em co czwartemu elementowi p, temu który jest dzieckiem elementu body.

W nawiasy wpisujemy formułę **an**+**b**, gdzie:

- **a** – liczba całkowita,
- **n** – określenie wielokrotności (jak w pętli),
- **b** – liczba całkowita.

```
p:nth-child(2n+1) {
    font-size: 1.5em;   /* 1., 3., 5. … dziecko */
}

p:nth-child(4n+4) {
    font-size: 1.5em; /* 4., 8., 12. … dziecko */
}
```

:nth-child

n	2n+1	4n+1	4n+4	4n	5n-2
0	1	1	4	–	–
1	3	5	8	4	3
2	5	9	12	8	8
3	7	13	16	12	13
4	9	17	20	16	18

- http://css-tricks.com/examples/nth-child-tester
- http://nth-test.com/

Pseudoklasy związane z położeniem

:nth-last-child()

Pseudoklasa, która wybiera pierwsze n-te dziecko od końca danego elementu.

W nawiasy wpisujemy formułę **an**+**b**, gdzie:

- **a** – liczba całkowita,
- **n** – określenie wielokrotności (jak w pętli),
- **b** – liczba całkowita.

```
p:nth-last-child(2n+1) {
    color: blue;   /* 1., 3., 5. … dziecko od końca */
}
```

Pseudoklasy – element danego typu

:first-of-type

Pseudoklasa, która wybiera pierwszy element danego typu.

:last-of-type

Pseudoklasa, która wybiera ostatni element danego typu.

```
p:first-of-type {
    font-size: 20px;
}

p:last-of-type {
    font-size: 20px;
}
```

Pseudoklasy – element danego typu

:nth-of-type

Pseudoklasa, która wybiera n-ty element danego typu.

:nth-last-of-type

Pseudoklasa, która wybiera n-ty element danego typu od końca.

```
p:nth-of-type(2n+1) {
    font-size: 20px; /* 1., 3., 5… element */
}

p:nth-last-of-type(2n+1) {
    font-size: 20px; /* 1., 3., 5… element od końca */
}
```

Pseudoelementy

Czym są pseudoelementy?

Poza pseudoklasami w CSS mamy też pseudoelementy.

Są one używane do stylowania części danego elementu albo treści wokoło elementów. Pseudoelementy są nieklikalne.

::before

Pseudoelement używany do dodania danej treści, obrazku przed danym elementem.

::after

Pseudoelement używany do dodania danej treści, obrazku po danym elemencie.

```
p::before {

}

ul::after {

}
```

Atrybut content

Ważne

Podczas używania pseudoelementów wymagany jest atrybut **content**.
Atrybut ten działa tylko dla pseudoelementów.

Wynik na stronie

- 1
- 2

Koniec listy!

```
<ul>
    <li> 1 </li>
    <li> 2 </li>
</ul>

ul::after {
    content: "Koniec listy!";
    color: red;
    font-weight: bold;
}
```

Pseudoelementy związane z tekstem

Pseudoelementy związane z tekstem

::first-letter

Dzięki temu pseudoelementowi wystylujemy pierwszą literę tekstu.

::first-line

Dzięki temu pseudoelementowi wystylujemy pierwszą linię tekstu.

```
p::first-letter {
    font-size: 2.5em;
}

p::first-line {
    text-decoration: underline;
}
```

Zaprzeczenie w CSS

Jeżeli nie chcemy czegoś wybrać

CSS pozwala nam na używanie zaprzeczeń dzięki selektorowi **:not**.

W pierwszym przykładzie wybieramy wszystkie elementy **div** oprócz elementu o klasie **content**.

W drugim przykładzie odwrotnie – wszystkie elementy o klasie **content**, ale nie elementy **div**.

```
div:not(.content) {
    color: #F00;
}

.content:not(div) {
    color: #F00;
}
```

Spis selektorów,
które warto pamiętać.

http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memorize--net-16048

Zadania