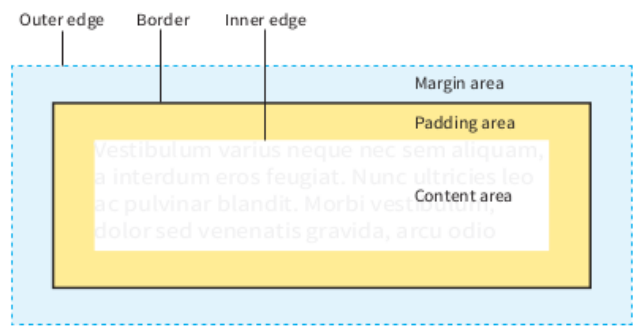


# 5.1 Teoria

## Step 1

Jak widzieliśmy, każdy element w dokumencie, zarówno na poziomie bloku, jak i w wierszu, generuje prostokątne pole elementu. Elementy składowe skrzynki elementów są przedstawione na poniżej. Zwróćmy uwagę na nową terminologię – będzie to pomocne w utrzymaniu porządku w dalszej części.



- **Obszar zawartości** (ang. **Content area**) - Rdzeniem pudełka elementu jest sama zawartość. Na rysunku obszar zawartości jest oznaczony białą ramką.
- **Wewnętrzne krawędzie** (ang. **Inner edges**) - Krawędzie obszaru zawartości są określane jako wewnętrzne krawędzie pudełka elementu. Chociaż na rysunku wewnętrzne krawędzie są wyróżnione przez zmianę koloru, na rzeczywistych stronach krawędź obszaru zawartości jest niewidoczna.
- **Wypełnienie** (ang. **padding**) - Wypełnienie to obszar między obszarem zawartości a opcjonalnym obramowaniem. Na diagramie obszar wypełnienia jest oznaczony kolorem żółto-pomarańczowym. Wypełnienie jest opcjonalne.
- **Obramowanie** (ang. **Border**) - Obramowanie to linia (lub linia stylizowana), która otacza element i jego dopełnienie. Są również opcjonalne.
- **Margines** (ang. **Margin**) - Margines to opcjonalna ilość miejsca dodawanego na zewnątrz obramowania. Na diagramie margines jest oznaczony jasnoniebieskim cieniowaniem, ale w rzeczywistości marginesy są zawsze przezroczyste, dzięki czemu widać tło elementu nadrzędnego.
- **Krawędź zewnętrzna** (ang. **Outer Edge**) - Zewnętrzne krawędzie obszaru marginesu tworzą zewnętrzną krawędź pudełka elementu. Jest to całkowity obszar zajmowany przez element na stronie, obejmujący szerokość obszaru zawartości oraz całkowitą ilość wypełnienia, obramowania i marginesów zastosowanych do elementu. Zewnętrzna krawędź na diagramie jest oznaczona linią przerywaną, ale na rzeczywistych stronach internetowych krawędź marginesu jest niewidoczna.

Wszystkie elementy mają te komponenty pudełkowe; jednak, jak zobaczymy, niektóre właściwości zachowują się inaczej w zależności od tego, czy element jest blokowy, czy wbudowany. W rzeczywistości niektóre z tych różnic zobaczymy od razu, gdy spojrzymy na wymiary pudełka.

<div>width</div> <div>Wartości: length   percentage   auto</div> <div>Domyślna wartość: auto</div> <div>Dotyczy: elementy blokowe i zastąpione elementy wbudowane (takie jak obrazy)</div> <div>Dziedziczy: no</div>
<div>height</div> <div>Wartości: length   percentage   auto</div> <div>Domyślna wartość: auto</div> <div>Dotyczy: elementy blokowe i zastąpione elementy wbudowane (takie jak obrazy)</div> <div>Dziedziczy: no</div>

## box-sizing

Wartości: content-box | border-box

Domyślna wartość: content-box

Dotyczy: wszystkich elementów

Dziedziczy: no

Domyślnie szerokość i wysokość elementu blokowego są obliczane automatycznie przez przeglądarkę (stąd domyślna wartość `auto`). Pudełko będzie tak szerokie jak okno przeglądarki lub inny blok zawierający element i tak wysokie, jak to konieczne, aby zmieściło się w nim zawartość. Możemy jednak użyć właściwości `width` i `height`, aby obszar zawartości elementu miał określoną szerokość lub wysokość.

Niestety, ustawienie wymiarów pola nie jest tak proste, jak po prostu umieszczenie tych właściwości w arkuszu stylów. Istnieją dwa sposoby określenia rozmiaru elementu. Domyślna metoda – wprowadzona dawno temu w `CSS1` – stosuje wartości szerokości i wysokości do pola zawartości. Oznacza to, że wynikowy rozmiar elementu będzie się składał z wymiarów określonych przez użytkownika plus ilość wypełnienia i obramowań, które zostały dodane do elementu. Druga metoda – wprowadzona jako część właściwości `box-sizing` w `CSS3` – stosuje wartości szerokości i wysokości do obramowania, które obejmuje zawartość, dopełnienie i obramowanie. Dzięki tej metodzie wynikowe widoczne pole elementu, w tym dopełnienie i obramowanie, będzie miało dokładnie takie wymiary, jakie określimy.

Bez względu na wybraną metodę szerokość i wysokość można określić tylko dla elementów blokowych i nietekstowych elementów śródliniowych, takich jak obrazy. Właściwości `width` i `height` nie mają zastosowania do elementów tekstu śródliniowego (niezastąpionych) i są ignorowane przez przeglądarkę. Innymi słowy, nie można określić szerokości i wysokości na przykład kotwicy (`a`) ani elementu `strong`.

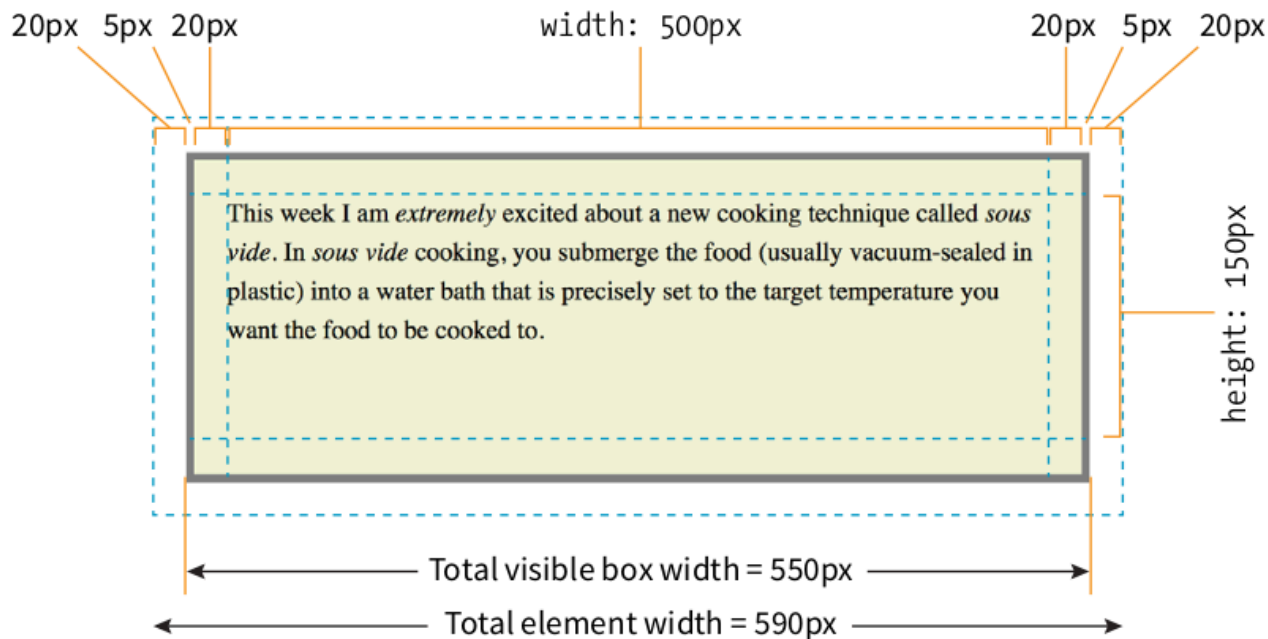
Domyślnie (to znaczy, jeśli w stylach nie uwzględnimy reguły określania rozmiaru pola), właściwości `width` i `height` są stosowane do pola zawartości. W ten sposób wszystkie obecne przeglądarki interpretują wartości szerokości i wysokości, ale możemy wyraźnie określić to zachowanie, ustawiając `box-sizing: content-box`. W poniższym przykładzie proste pudełko ma szerokość `500` pikseli i wysokość `150` pikseli, z `20` pikselami wypełnienia, `5`-pikselową ramką i `20`-pikselowym marginesem dookoła. W domyślnym modelu pola zawartości wartości szerokości i wysokości są stosowane tylko do obszaru zawartości.

```
p {
  background: #f2f5d5;
  width: 500px;
  height: 150px;
  padding: 20px;
  border: 5px solid gray;
  margin: 20px;
}
```

Wynikowa szerokość pola widocznego elementu wynosi `550` pikseli: zawartość plus `40` pikseli wypełnienia (po `20` pikseli po lewej i prawej stronie) i `10` pikseli obramowania (po `5` pikseli po lewej i prawej stronie).

Gdy dorzucimy `40` pikseli marginesu, szerokość całego pola elementu wynosi `590` pikseli. Znajomość wynikowego rozmiaru elementów ma kluczowe znaczenie dla przewidywalnego zachowania układów.

## content-box model

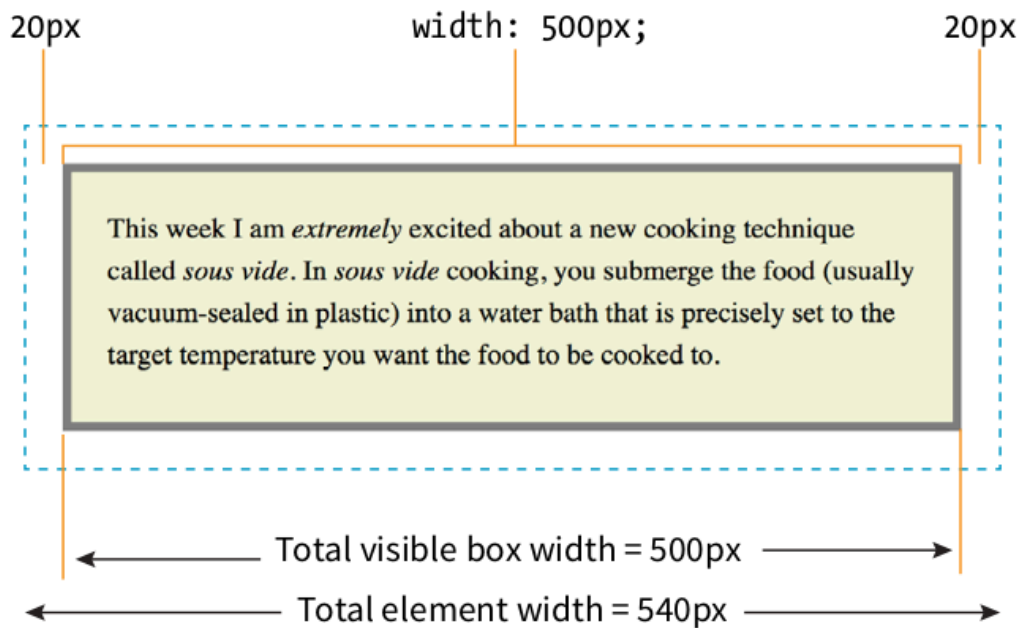


Innym sposobem określenia rozmiaru elementu jest zastosowanie wymiarów szerokości i wysokości do całego widocznego pola, w tym dopełnienia i obramowania. Ponieważ nie jest to domyślne zachowanie przeglądarki, musimy jawnie ustawić `box-sizing: border-box` w arkuszu stylów. Przyjrzyjmy się temu samemu przykładowi akapitu i zobaczmy, co się stanie, gdy zmniejszymy go do 500 pikseli za pomocą metody `border-box`. Wszystkie inne deklaracje stylu dla pudełka pozostają takie same.

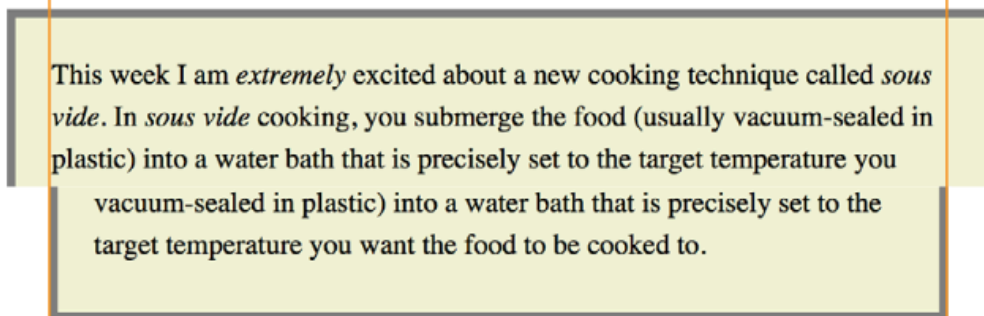
```
p {  
  background: #f2f5d5;  
  box-sizing: border-box;  
  width: 500px;  
  height: 150px;  
  padding: 20px;  
  border: 5px solid gray;  
  margin: 20px;  
}
```

Teraz szerokość widocznego pola wynosi 500 pikseli (w porównaniu z 550 pikselami w modelu `content-box`), a całkowita szerokość elementu wynosi 540 pikseli. Wielu programistów uważa model `border-box` za bardziej intuicyjny sposób określania rozmiaru elementów. Jest to szczególnie przydatne przy określaniu szerokości w procentach, co jest podstawą projektowania responsywnego. Na przykład możemy utworzyć dwie kolumny o szerokości 50% i wiedzieć, że będą pasować obok siebie, bez konieczności dodawania obliczonych szerokości wypełnienia i obramowania do miksu (choć nadal musimy uwzględnić marginesy).

## border-box model



`box-sizing: content-box;`  
`width: 500px;`



`box-sizing: border-box;`  
`width: 500px;`

W rzeczywistości wielu programistów po prostu ustawia wszystko w dokumencie tak, aby korzystało z modelu obramowania, ustawiając go w elemencie głównym ( `html` ), a następnie ustawiając dziedziczenie wszystkich innych elementów, tak jak poniżej:

```
html {box-sizing: border-box;}
*, *:before, *:after {box-sizing: inherit;}
```

Właściwość `height` działa tak samo jak `width`. W praktyce rzadziej określa się wysokość elementów. Bardziej zgodne z charakterem nośnika jest umożliwienie automatycznego obliczania wysokości, co pozwala na zmianę pola elementu w zależności od rozmiaru czcionki, ustawień użytkownika lub innych czynników. Jeśli określimy wysokość elementu zawierającego tekst, pamiętajmy również o tym, co się stanie, jeśli zawartość się nie zmieści.

Kiedy rozmiar elementu jest zbyt mały w stosunku do jego zawartości, możemy określić, co zrobić z treścią, która się nie mieści, używając właściwości `overflow`.

`overflow`

Wartości: `visible` | `hidden` | `scroll` | `auto`

Domyślna wartość: `visible`

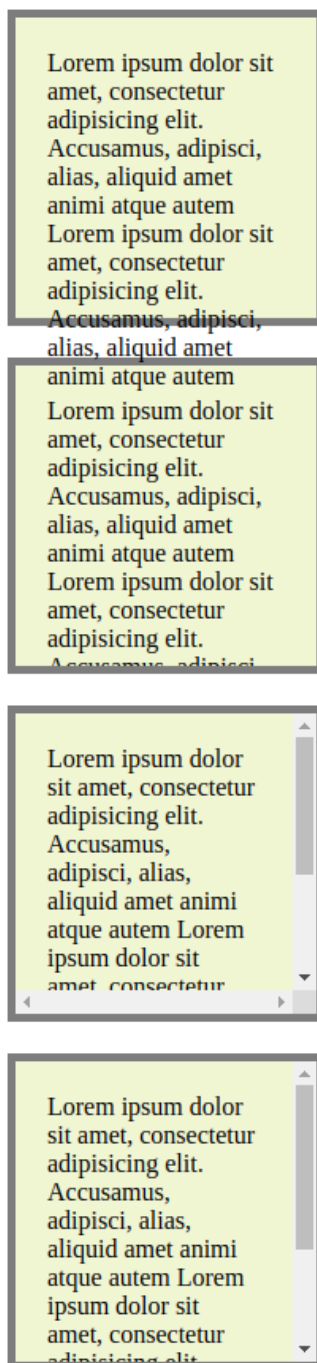
Dotyczy: elementy blokowe i zastąpione elementy wbudowane (takie jak obrazy)

Dziedziczy: `no`

Przykład poniżej pokazuje predefiniowane wartości dla `overflow`. Na rysunku różne wartości są stosowane do elementu o boku 150 pikseli. Kolor tła sprawia, że krawędzie obszaru zawartości są widoczne.

- `visible` - Wartość domyślna to `visible`, co pozwala zawartości zawisnąć nad polem elementu, dzięki czemu można ją zobaczyć.
- `hidden` - Gdy `overflow` jest ustawione na `hidden`, treść, która nie pasuje, zostaje obcięta i nie pojawia się poza krawędziami obszaru zawartości elementu.
- `scroll` - Gdy określono `scroll`, paski przewijania są dodawane do pola elementu, aby umożliwić użytkownikom przewijanie zawartości. Pamiętajmy, że mogą stać się widoczne dopiero po kliknięciu elementu, aby go przewinąć.
- `auto` - Wartość `auto` pozwala przeglądarce zdecydować, jak obsłużyć `overflow`. W większości przypadków paski przewijania są dodawane tylko wtedy, gdy treść nie pasuje i są potrzebne.

```
<!--overflow_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Overflow example</title>
  <style>
    p {
      background: #f2f5d5;
      width: 150px;
      height: 150px;
      padding: 20px;
      border: 5px solid gray;
      margin: 20px;
    }
  </style>
</head>
<body>
  <p style="overflow: visible">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
  <p style="overflow: hidden">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
  <p style="overflow: scroll">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
  <p style="overflow: auto">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
</body>
</html>
```



## Step 2

**Wypełnienie** (ang. **padding**) to przestrzeń między obszarem zawartości a obramowaniem (lub miejscem, w którym byłoby obramowanie, gdyby nie zostało określone). Pomocne jest dodawanie wypełnienia do elementów podczas używania koloru tła lub obramowania. Daje to trochę swobody treści i zapobiega uderzaniu obramowania lub krawędzi tła w tekst.

Możemy dodać wypełnienie do poszczególnych boków dowolnego elementu (na poziomie bloku lub w wierszu). Istnieje również skrócona właściwość `padding`, która umożliwia jednoczesne dodanie dopełnienia ze wszystkich stron.

padding-top, padding-right, padding-bottom, padding-left

Wartości: length | percentage

Domyślnie: 0

Dotyczy: wszystkich elementów

Dziedziczny: nie

padding

Wartości: length | percentage

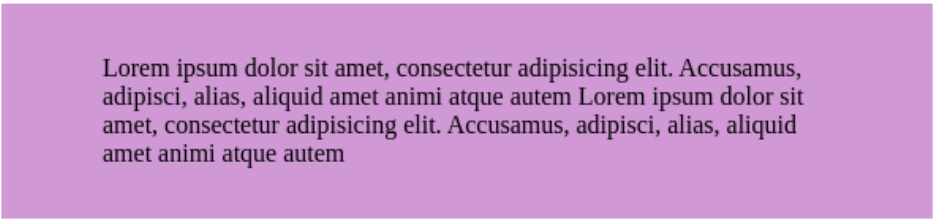
Domyślnie: 0

Dotyczy: wszystkich elementów

Dziedziczy: nie

Właściwości `padding-top`, `padding-right`, `padding-bottom` i `padding-left` określają wielkość wypełnienia dla każdej strony elementu, jak pokazano w poniższym przykładzie:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Padding example</title>
  <style>
    blockquote {
      padding-top: 2em;
      padding-right: 4em;
      padding-bottom: 2em;
      padding-left: 4em;
      background-color: #D098D4;
    }
  </style>
</head>
<body>
  <blockquote>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </blockquote>
</body>
</html>
```



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

`Padding` możemy określić w dowolnej jednostce długości `CSS` (najczęściej `em` i `px`) lub jako procent szerokości elementu nadrzędnego. Tak, szerokość rodzica jest używana jako podstawa, nawet dla wypełnienia górnego i dolnego. Jeśli zmieni się szerokość elementu nadrzędnego, zmieniają się również wartości dopełnienia ze wszystkich stron elementu podrzędnego, co sprawia, że zarządzanie wartościami procentowymi jest nieco trudne.

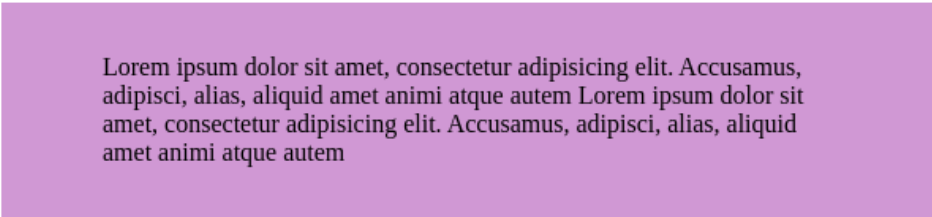
Jako alternatywę dla ustawiania dopełnienia po jednej stronie na raz, możemy użyć skróconej właściwości `padding`, aby dodać dopełnienie wokół elementu. Składnia jest interesująca; można określić cztery, trzy, dwa lub jedną wartość dla pojedynczej właściwości dopełnienia. Zobaczmy, jak to działa, zaczynając od czterech wartości.

Jeśli podamy cztery wartości dopełnienia, zostaną one zastosowane po każdej stronie w kolejności zgodnej z ruchem wskazówek zegara, zaczynając od góry. Niektórzy używają mnemonicznego urządzenia `TRouBL` (`Top` `Right` `Bottom` `Left`). Jest to powszechna składnia stosowania skrótów w `CSS`.

```
padding: top right bottom left;
```

Korzystając z właściwości `padding`, możemy odtworzyć wypełnienie określone czterema indywidualnymi właściwościami z poprzedniego przykładu w następujący sposób:

```
<!--padding_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Padding example 2</title>
  <style>
    blockquote {
      padding: 2em 4em 2em 4em;
      background-color: #D098D4;
    }
  </style>
</head>
<body>
<blockquote>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
  animi atque autem
  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
  animi atque autem
</blockquote>
</body>
</html>
```



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Jeśli lewe i prawe wypełnienie są takie same, możemy je skrócić, podając tylko trzy wartości. Wartość `right` (druga wartość w łańcuchu) zostanie odzwierciedlona i użyta również dla `left`. To tak, jakby przeglądarka zakładała, że brakuje wartości `left`, więc po prostu używa wartości `right` po obu stronach. Składnia dla trzech wartości jest następująca:

```
padding: top right/left bottom;
```

Ta reguła byłaby równoważna z poprzednim przykładem, ponieważ dopełnienie zarówno lewej, jak i prawej krawędzi elementu jest ustawione na `4em`:

```
<!--padding_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Padding example 2</title>
  <style>
    blockquote {
      padding: 2em 4em 2em;
      background-color: #D098D4;
    }
  </style>
</head>
<body>
<blockquote>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
  animi atque autem
  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
  animi atque autem
</blockquote>
</body>
</html>
```



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Kontynuując ten wzór, jeśli podamy tylko dwie wartości, pierwsza zostanie użyta dla górnej i dolnej krawędzi, a druga dla lewej i prawej krawędzi:

```
padding: top/bottom right/left;
```

Ponownie, ten sam efekt osiągnięty w poprzednich dwóch przykładach można osiągnąć za pomocą tej reguły:

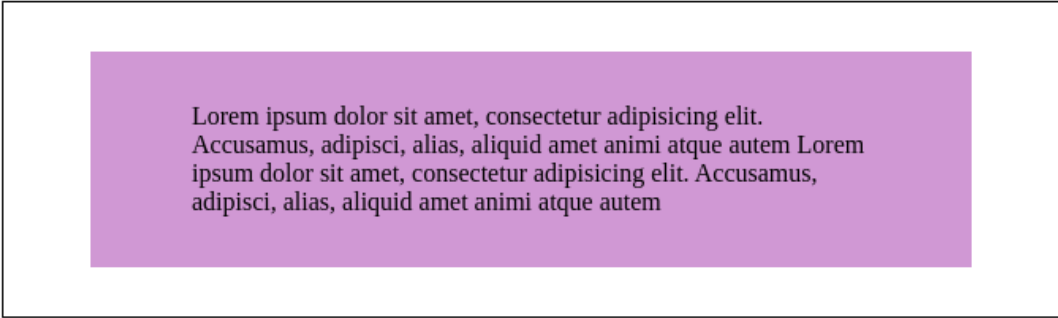
```
<!--padding_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Padding example 2</title>
  <style>
    blockquote {
      padding: 2em 4em;
      background-color: #D098D4;
    }
  </style>
</head>
<body>
<blockquote>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
  animi atque autem
  Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
  animi atque autem
</blockquote>
</body>
</html>
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Wreszcie, jeśli podamy tylko jedną wartość, zostanie ona zastosowana do wszystkich czterech stron elementu. Ta deklaracja stosuje

15 pikseli dopełnienia ze wszystkich stron elementu `div` :

```
<!--padding_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Padding example 2</title>
  <style>
    blockquote {
      padding: 2em 4em;
      background-color: #D098D4;
    }
    div#announcement {
      padding: 15px;
      border: 1px solid;
    }
  </style>
</head>
<body>
<div id="announcement">
  <blockquote>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </blockquote>
</div>
</body>
</html>
```



Lorem ipsum dolor sit amet, consectetur adipisicing elit.  
Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem  
ipsum dolor sit amet, consectetur adipisicing elit. Accusamus,  
adipisci, alias, aliquid amet animi atque autem

## Step 3

**Obramowanie** (ang. `border`) to po prostu linia narysowana wokół obszaru zawartości i jego (opcjonalnie) wypełnienia. Możesz wybierać spośród ośmiu stylów obramowania i nadać im dowolną szerokość i kolor. Obramowania można zastosować dookoła elementu lub tylko po określonej stronie lub bokach. CSS3 wprowadził właściwości zaokrąglania rogów lub nakładania obrazów na krawędzie. Rozpocznijmy naszą eksplorację granic od różnych stylów granic.

Style obramowania można stosować po jednej stronie naraz lub za pomocą skróconej właściwości `border-style`.

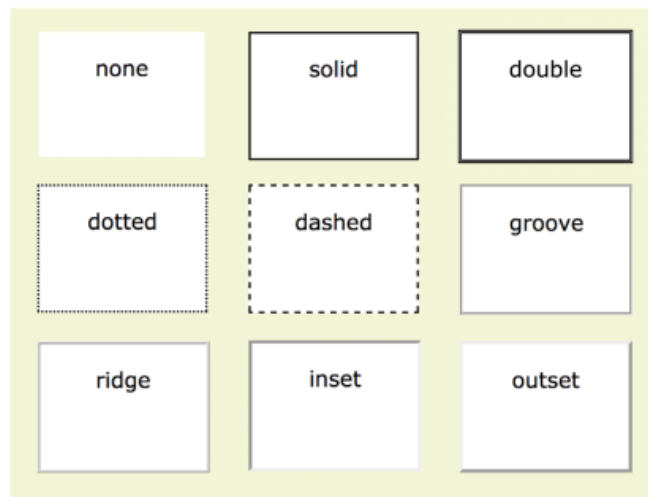
`border-top-style, border-right-style,  
border-bottom-style, border-left-style`

Wartości: `none | solid | hidden | dotted | dashed | double | groove | ridge | inset | outset`  
Domyślnie: `none`  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

`border-style`

Wartości: `none | solid | hidden | dotted | dashed | double | groove | ridge | inset | outset`  
Domyślnie: `none`  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

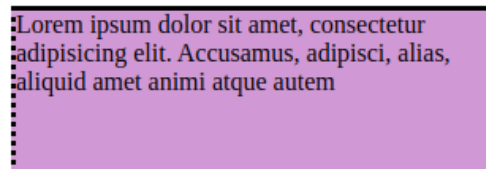
Wartość właściwości `border-style` jest jednym z 10 słów kluczowych opisujących dostępne style obramowania, jak pokazano poniżej. Ukryta wartość jest równoważna `none`.



Użycie właściwości stylu obramowania specyficznego dla boku ( `border-top-style` , `border-right-style` , `border-bottom-style` i `border-left-style` ), stosuje styl do jednej strony elementu. Jeśli nie określiliśmy szerokości, zostanie użyta domyślna średnia szerokość. Jeśli nie określono koloru, obramowanie używa koloru pierwszego planu elementu (takiego samego jak tekst).

W poniższym przykładzie zastosowano inny styl do każdej strony elementu, aby pokazać właściwości obramowania jednostronnego w działaniu.

```
<!--border_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border example</title>
<style>
  div#silly {
    border-top-style: solid;
    border-right-style: dashed;
    border-bottom-style: double;
    border-left-style: dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
</style>
</head>
<body>
  <div id="silly">
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
</body>
</html>
```

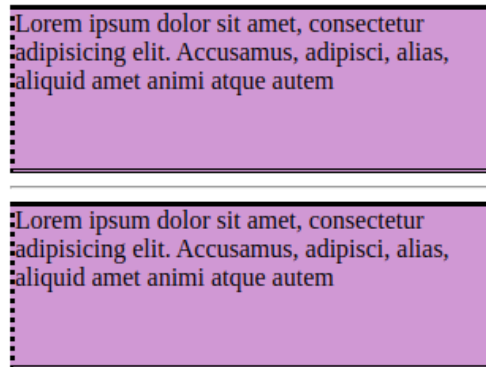


Właściwość `border-style` działa w systemie zgodnym z hasłem ( `TRouBLe` ) opisanym wcześniej dla wypełniania. Możemy podać cztery wartości dla wszystkich czterech stron lub mniej wartości, gdy lewe/prawe i górne/dolne obramowanie są takie same. Głupi efekt obramowania z poprzedniego przykładu można również określić za pomocą właściwości `border-style` , jak pokazano tutaj, a wynik byłby taki sam:

```

<!--border_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border example</title>
<style>
  div#silly {
    border-top-style: solid;
    border-right-style: dashed;
    border-bottom-style: double;
    border-left-style: dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
  div#silly2 {
    border-style: solid dashed double dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
</style>
</head>
<body>
  <div id="silly">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="silly2">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
</body>
</html>

```



Użycie jednej z właściwości `border-width` , określa grubość obramowania. Po raz kolejny możemy obrać za cel każdą stronę elementu za pomocą właściwości jednostronnej lub określić kilka stron jednocześnie w kolejności zgodnej z hasłem ( `TRouBLe` ) za pomocą skróconej właściwości `border-width` .

`border-top-width, border-right-width,`  
`border-bottom-width, border-left-width`

Wartości: `length` | `thin` | `medium` | `thick`  
Domyślnie: `medium`  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

`border-width,`

Wartości: `length` | `thin` | `medium` | `thick`  
Domyślnie: `medium`  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

Najpopularniejszym sposobem określania szerokości obramowań jest pomiar w pikselach lub `em`; jednak możemy również określić jedno ze słów kluczowych ( `thin` , `medium` lub `thick` ) i pozostawić renderowanie przeglądarce.

W tym przykładzie uwzględniono kombinację wartości. Uwzględniono również właściwość `border-style` , ponieważ obramowanie w ogóle by się nie renderowało:

```
<!--border_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border example</title>
<style>
  div#silly {
    border-top-style: solid;
    border-right-style: dashed;
    border-bottom-style: double;
    border-left-style: dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
  div#silly2 {
    border-style: solid dashed double dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
  div#help {
    border-top-width: thin;
    border-right-width: medium;
    border-bottom-width: thick;
    border-left-width: 12px;
    border-style: solid;
    width: 300px;
    height: 100px;
  }
  div#help2 {
    border-width: thin medium thick 12px;
    border-style: solid;
    width: 300px;
    height: 100px;
  }
</style>
</head>
<body>
  <div id="silly">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="silly2">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="help">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="help2">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
</body>
</html>
```

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Kolory obramowania są określane w ten sam sposób: poprzez właściwości specyficzne dla strony lub skróconą właściwość `border-color`. Określenie koloru obramowania zastępuje kolor pierwszego planu ustawiony przez właściwość `color` elementu.

`border-top-color, border-right-color,  
border-bottom-color, border-left-color`

Wartości: color name or RGB/HSL value | transparent  
Domyślnie: color elementu  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

`border-color`

Wartości: color name or RGB/HSL value | transparent  
Domyślnie: color elementu  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

Wiemy wszystko o określaniu wartości kolorów i powinniśmy przyzwyczaić się również do właściwości skrótowych, więc ten przykład będzie krótki i przyjemny. Tutaj podano dwie wartości właściwości `border-color`, aby góra i dół elementu `div` `maroon` oraz lewa i prawa strona były koloru `aqua`:

```
<!--border_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border example</title>
<style>
  div#silly {
    border-top-style: solid;
    border-right-style: dashed;
    border-bottom-style: double;
    border-left-style: dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
  div#silly2 {
    border-style: solid dashed double dotted;
    background-color: #D098D4;
    width: 300px;
    height: 100px;
  }
  div#help {
    border-top-width: thin;
    border-right-width: medium;
    border-bottom-width: thick;
    border-left-width: 12px;
    border-style: solid;
    width: 300px;
    height: 100px;
  }
  div#help2 {
    border-width: thin medium thick 12px;
    border-style: solid;
    width: 300px;
    height: 100px;
  }
  div#special {
    border-color: maroon aqua;
    border-style: solid;
    border-width: 6px;
    width: 300px;
    height: 100px;
  }
</style>
</head>
<body>
  <div id="silly">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="silly2">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="help">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="help2">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
  <hr/>
  <div id="special">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </div>
```

```
</body>  
</html>
```

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur  
adipiscing elit. Accusamus, adipisci, alias,  
aliquid amet animi atque autem

## Step 4

Innym rodzajem reguły, którą można narysować wokół elementu, jest kontur (ang. `outline`). Kontury wyglądają jak obramowania, a składnia jest taka sama, ale istnieje istotna różnica. Kontury, w przeciwieństwie do obramowań, nie są liczone na szerokość pola elementu. Po prostu leżą na wierzchu, nie ingerując w nic. Kontury są rysowane na zewnętrznej krawędzi obramowania (jeśli została określona) i zachodzą na marginesy.

Ponieważ kontury nie wpływają na układ, są doskonałym narzędziem do sprawdzania projektu. Możemy je włączać i wyłączać bez wpływu na pomiary szerokości, aby zobaczyć, gdzie i jak są ustawione pola elementów.

Właściwości `outline` są podobne do właściwości obramowania z jedną ważną różnicą: nie można określić obrysów dla poszczególnych boków elementu – wszystko albo nic.

`outline-style`

Wartości: `auto` | `solid` | `none` | `dotted` | `dashed` | `double` | `groove` | `ridge` | `inset` | `outset`  
Domyślnie: `none`

Są to te same wartości, co wartości w `border-style`, z dodatkem `auto`, który pozwala przeglądarce wybrać styl. Nie można również ustawić stylu konturu na `hidden`.

`outline-width`

Wartości: `length` | `thin` | `medium` | `thick`  
Domyślnie: `medium`



outline-color

Wartości: color name or RGB/HSL value | invert  
Domyślnie: invert

Domyślna wartość `invert` powoduje zastosowanie odwrotności koloru tła do konturu, ale jest bardzo słabo obsługiwana przez przeglądarkę.

outline-offset

Wartości: length  
Domyślnie: 0

Domyślnie kontur jest rysowany tuż poza krawędzią obramowania. przesunięcie konturu przesunę kontur poza granicę o określoną długość.

outline

Wartości: outline-style outline-width outline-color  
Domyślnie: Domyślne wartości powyższych właściwości

Skrócona właściwość `outline` łączy wartości dla `outline-style` , `outline-width` i `outline-color` . Pamiętajmy, że możemy je określić tylko dla wszystkich stron elementu na raz.

```
<!--outline_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Outline example</title>
  <style>
    div#story {
      outline: 2px dashed red;
      border: 2px dotted purple;
    }
  </style>
</head>
<body>
  <div id="story">
    Lorem ipsum
  </div>
</body>
</html>
```



## Step 5

Twórcy `CSS` nie skąpili, jeśli chodzi o skróty do właściwości `border` . Stworzyli również właściwości do dostarczania wartości stylu, szerokości i koloru w jednej deklaracji, po jednej stronie na raz. Możemy określić wygląd określonych boków lub użyć właściwości `border` , aby zmienić wszystkie cztery boki jednocześnie.

border-top, border-right, border-bottom, border-left

Wartości: border-style border-width border-color  
Domyślnie: domyślna wartość każdej właściwości  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

border

Wartości: border-style border-width border-color

Domyślnie: domyślna wartość każdej właściwości

Dotyczy: wszystkich elementów

Dziedziczy: nie

Wartości dla `border` i właściwości obramowania specyficzne dla strony mogą obejmować wartości stylu, szerokości i koloru w dowolnej kolejności. Nie musimy deklarować wszystkich trzech, ale jeśli wartość `border-style` zostanie pominięta, obramowanie nie zostanie wyrenderowane.

Skrócona właściwość `border` działa nieco inaczej niż inne omówione przez nas właściwości skrócone, ponieważ przyjmuje jeden zestaw wartości i zawsze stosuje je do wszystkich czterech stron elementu. Innymi słowy, nie używa zgodnego z hasłem `TRouBLe`, który widzieliśmy z innymi właściwościami skrótowymi.

Oto kilka prawidłowych przykładów właściwości `border`, które dają wyobrażenie o tym, jak działają:

```
<!--border_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border example 2</title>
  <style>
    h1 { border-left: red .5em solid; }
    h2 { border-bottom: 1px solid; }
    p.example { border: 2px dotted #663; }
  </style>
</head>
<body>
  <h1>This is a big heading</h1>
  <h2>This is another heading</h2>
  <p class="example">
    Lorem ipsum of my class example
  </p>
</body>
</html>
```

**This is a big heading**

**This is another heading**

Lorem ipsum of my class example

Właściwość `border-radius` powoduje zaokrąglanie krawędzi naszych pudełek. Istnieją indywidualne właściwości narożników, a także skrócony promień obramowania.

border-top-left-radius, border-top-right-radius,  
border-bottom-right-radius, border-bottom-left-radius

Wartości: length | percentage

Domyślnie: 0

Dotyczy: wszystkich elementów

Dziedziczy: nie

border-radius

Wartości: 1, 2, 3, 4 | length | percentage

Domyślnie: 0

Dotyczy: wszystkich elementów

Dziedziczy: nie

Aby zaokrąglić róg elementu, po prostu należy zastosować jedną z właściwości `border-radius` , ale pamiętajmy, że zobaczysz wynik tylko wtedy, gdy element ma obramowanie lub kolor tła. Wartości są zwykle podawane w `em` lub pikselach. Wartości procentowe są dozwolone i są przydatne do zachowania proporcjonalności krzywej do pola w przypadku zmiany rozmiaru, ale mogą wystąpić pewne niespójności w przeglądarce.

Możemy opisywać rogi indywidualnie lub użyć skróconej właściwości `border-radius` . Jeśli podamy jedną wartość dla promienia obramowania, zostanie ona zastosowana do wszystkich czterech rogów. Cztery wartości są stosowane zgodnie z ruchem wskazówek zegara, zaczynając od lewego górnego rogu ( `top-left` , `top-right` , `bottom-right` , `bottom-left` ). Gdy podamy dwie wartości, pierwsza jest używana dla lewego górnego i prawego dolnego rogu, a druga dla pozostałych dwóch rogów.

Porównajmy wartości promienia obramowania z wynikowymi polami poniżej. W zależności od tego, jak ustawimy wartości, możemy osiągnąć wiele różnych efektów, od lekko zmiękczonych narożników po kształt długiej kapsułki.

```
<!--border_radius_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border radius example</title>
<style>
  p {
    width: 200px;
    height: 100px;
    background: darkorange;
    text-align: center;
  }
  p.many {
    border-top-left-radius: 1em;
    border-top-right-radius: 2em;
    border-bottom-right-radius: 1em;
    border-bottom-left-radius: 2em;
  }
</style>
</head>
<body>
  <p style="border-radius: 1em;">border-radius: 1em</p>
  <p style="border-radius: 50px;">border-radius: 50px</p>
  <p style="border-top-right-radius: 50px;">border-top-right-radius: 50px</p>
  <p class="many">all four properties</p>
</body>
</html>
```



Do tej pory narożniki, które wykonaliśmy, są przekrojami idealnych okręgów, ale można również utworzyć narożnik eliptyczny, określając dwie wartości: pierwszą dla promienia poziomego i drugą dla promienia pionowego. Jeśli chcemy użyć właściwości skrótu, promienie poziome i pionowe zostaną oddzielone ukośnikiem (w przeciwnym razie zostałyby pomyłone dla różnych wartości narożników). W przykładzie ustawiamy promień poziomy we wszystkich rogach na 60 pikseli, a promień pionowy na 40 pikseli.

```
<!--border_radius_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Border radius example 2</title>
<style>
  p {
    width: 200px;
    height: 100px;
    background: darkorange;
    text-align: center;
  }
  p.many {
    border-top-right-radius: 50px 20px;
    border-top-left-radius: 50px 20px;
  }
</style>
</head>
<body>
border-top-right-radius: 100px 50px;
  <p style="border-top-right-radius: 100px 50px;"></p>
  border-radius: 60px / 40px;
  <p style="border-radius: 60px / 40px;"></p>
  border-radius: 36px 40px 60px 20px / 12px 10px 30px 36px;
  <p style="border-radius: 36px 40px 60px 20px / 12px 10px 30px 36px;"></p>
border-top-right-radius: 50px 20px;
border-top-left-radius: 50px 20px;
  <p class="many"></p>
</body>
</html>
```

border-top-right-radius: 100px 50px;



border-radius: 60px / 40px;



border-radius: 36px 40px 60px 20px / 12px 10px 30px 36px;



border-top-right-radius: 50px 20px; border-top-left-radius: 50px 20px;



## Step 6

Margines (ang. `margin`) to opcjonalna ilość miejsca, którą można dodać na zewnątrz obramowania. Marginesy zapobiegają wpadaniu elementów na siebie lub na krawędź okna przeglądarki lub widocznego obszaru.

Właściwości `margin` działają podobnie do właściwości `padding`, o których już mówiliśmy; jednak marginesy mają pewne specjalne zachowania, o których należy pamiętać.

```
margin-top, margin-right, margin-bottom, margin-left
```

```
Wartości: length | percentage | auto  
Domyślnie: auto  
Dotyczy: wszystkich elementów  
Dziedziczy: nie
```

```
margin
```

```
Wartości: length | percentage | auto  
Domyślnie: auto  
Dotyczy: wszystkich elementów  
Dziedziczy: nie
```

Właściwości marginesu są bardzo proste w użyciu. Możemy określić wielkość marginesu, który ma się pojawić po każdej stronie elementu, lub użyć właściwości `margin`, aby określić wszystkie boki jednocześnie.

Właściwość `margin` działa tak samo jak `padding`. Jeśli podamy cztery wartości, zostaną one zastosowane w kolejności zgodnej z ruchem wskazówek zegara (`top`, `right`, `bottom`, `left`) po bokach elementu. Jeśli podamy trzy wartości, środkowa wartość dotyczy zarówno lewej, jak i prawej strony. W przypadku podania dwóch wartości pierwsza jest używana dla górnej i dolnej krawędzi, a druga dla lewej i prawej krawędzi. Na koniec jedna wartość zostanie zastosowana do wszystkich czterech boków elementu.

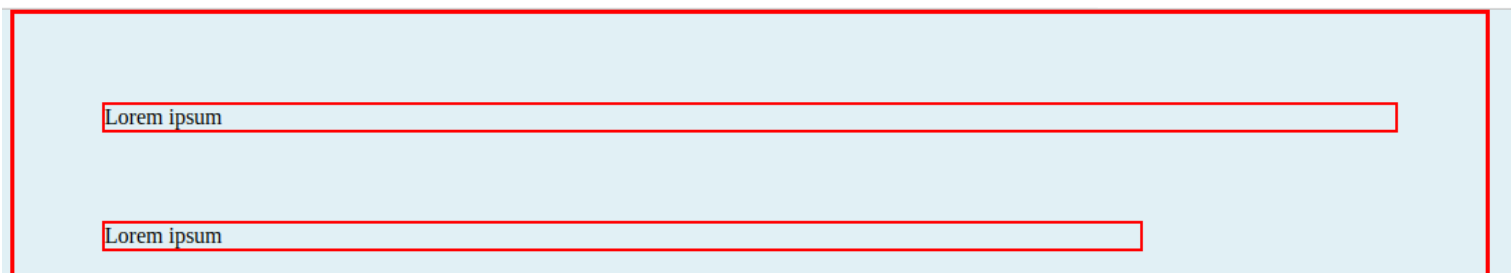
Podobnie jak w przypadku większości miar internetowych, `em`, piksele i wartości procentowe to najczęstsze sposoby określania marginesów. Pamiętajmy jednak, że jeśli określimy wartość procentową, zostanie ona obliczona na podstawie szerokości elementu nadrzędnego. Jeśli zmieni się szerokość elementu nadrzędnego, zmienią się również marginesy na wszystkich czterech bokach elementu podrzędnego (dopełnienie również ma takie zachowanie). Słowo kluczowe `auto` umożliwia przeglądarce wypełnienie marginesu niezbędnego do dopasowania lub wypełnienia dostępnego miejsca.

Poniżej znajduje się przykład pokazujący wyniki następujących przykładów marginesów. Dodano czerwoną ramkę do elementów w przykładach, aby ich granice były bardziej wyraźne. Linie kropkowane zostały dodane na rysunku, aby wskazać zewnętrzne krawędzie marginesów wyłącznie dla zachowania przejrzystości, ale nie są one czymś, co można zobaczyć w przeglądarce.

```

<!--margin_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Margin example</title>
<style>
  p#A {
    margin: 4em;
    border: 2px solid red;
    background: #e2f3f5;
  }
  p#B {
    margin-top: 2em;
    margin-right: 250px;
    margin-bottom: 1em;
    margin-left: 4em;
    border: 2px solid red;
    background: #e2f3f5;
  }
  body {
    margin: 0 20%;
    border: 3px solid red;
    background-color: #e2f3f5;
  }
</style>
</head>
<body>
  <p id="A">Lorem ipsum</p>
  <p id="B">Lorem ipsum</p>
</body>
</html>

```



Chociaż łatwo jest napisać reguły, które stosują kwoty marginesów wokół elementów `HTML`, ważne jest, aby znać niektóre dziwactwa związane z zachowaniem marginesów.

Najbardziej znaczącym zachowaniem marginesów, o którym należy pamiętać, jest zawijanie górnego i dolnego marginesu sąsiednich elementów. Oznacza to, że zamiast kumulować się, sąsiednie marginesy nakładają się i używana jest tylko największa wartość.

Używając dwóch akapitów z poprzedniego rysunku jako przykładu, jeśli górny element ma margines dolny `4em`, a następny element ma margines górny `2em`, wynikowy margines między elementami nie sumuje się do `6em`. Zamiast tego marginesy się zwiną, a wynikowy margines między akapitami wyniesie `4em`, największą określoną wartość.

Górny i dolny margines nie zawijają się tylko w przypadku elementów pływających lub pozycjonowanych bezwzględnie. Marginesy po lewej i prawej stronie nigdy się nie zawijają, więc są ładne i przewidywalne.

```
<!--collapse_margin_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Collapse margin example</title>
<style>
  p.one{
    border: 1px solid;
    margin: 4em;
  }
  p.two{
    border: 1px solid;
    margin: 2em;
  }
</style>
</head>
<body>
  <p class="one">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
  <p class="two">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
</body>
</html>
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Możemy zastosować margines górny i dolny do elementów tekstu w wierszu (lub „niezastąpionych elementów w wierszu”, by użyć właściwej terminologii `CSS`), ale nie spowoduje to dodania pionowej przestrzeni nad i pod elementem, a wysokość linii będzie się nie zmieniać. Jeśli jednak zastosujemy lewy i prawy margines do elementów tekstu w wierszu, odstępy między marginesami przed i za tekstem w przepływie elementu będą utrzymywane w czystości, nawet jeśli ten element zostanie podzielony na kilka wierszy.

Aby było ciekawiej, marginesy na zastąpionych elementach śródliniowych, takich jak obrazy, renderują się ze wszystkich stron, a zatem wpływają na wysokość linii.

```
<!--collapse_margin_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Collapse margin example</title>
<style>
  p.one{
    border: 1px solid;
    margin: 4em;
  }
  p.two{
    border: 1px solid;
    margin: 2em;
  }
  em {
    border: 1px solid;
    margin: 2em;
  }
  img {
    border: 1px solid;
    margin: 2em;
  }
</style>
</head>
<body>
  <p class="one">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
  <p class="two">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem
  </p>
  <p>
    Lorem ipsum dolor sit amet, consectetur <em>adipisicing</em> elit. Accusamus, adipisci, alias, aliquid
    amet animi atque autem
    Lorem ipsum dolor sit amet, consectetur  adipisicing elit.
    Accusamus, adipisci, alias, aliquid amet animi atque autem
  </p>
</body>
</html>
```



Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem Lorem ipsum dolor sit



amet, consectetur

adipiscing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem

Warto zaznaczyć, że istnieje możliwość określenia wartości ujemnych dla marginesów. Po zastosowaniu marginesu ujemnego zawartość, `padding` i `border` są przesuwane w przeciwnym kierunku, niż wynikałoby to z dodatniej wartości marginesu.

Wyjaśnimy to na przykładzie. Poniżej mamy dwa sąsiednie akapity z obramowaniami w różnych kolorach, aby pokazać ich granice. W lewym widoku dodano dolny margines `3em` do górnego akapitu, co powoduje przesunięcie następnego akapitu w dół o tę wartość. Jeśli określimy wartość ujemną ( `-3em` ), następny element przesunie się w górę o tę wartość i nałoży się na element z ujemnym marginesem.

```
<!--negative_margins_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Negative margins example</title>
<style>
  p.one {
    margin-bottom: 3em;
    border: 2px solid;
  }
  p.two {
    margin-bottom: -3em;
    border: 2px solid;
  }
</style>
</head>
<body>
  <p class="one">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem.
  </p>
  <p class="one">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem.
  </p>
  <p class="two">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem.
  </p>
  <p class="two">
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet
    animi atque autem.
  </p>
</body>
</html>
```

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi atque autem.

Może się to wydawać dziwne, a tak naprawdę prawdopodobnie nie nakładalibyśmy bloków tekstu, jak pokazano. Chodzi o to, że możemy używać marginesów zarówno z wartościami dodatnimi, jak i ujemnymi, aby przenosić elementy na stronie. Jest to podstawa niektórych starszych technik układu `CSS`.

## Step 7

Skoro już mówimy o pudełkach i modelu układu `CSS`, to jest to dobry moment na wprowadzenie właściwości `display`. Powinniśmy już być zaznajomiony z zachowaniem wyświetlania elementów blokowych i wbudowanych. Choć `HTML` przypisuje określone zachowania (lub typy wyświetlania, używając najnowszego terminu `CSS`) do elementów, które definiuje, istnieją inne języki oparte na `XML`, które mogą używać `CSS` i nie robią tego samego. Z tego powodu właściwość `display` została stworzona, aby umożliwić autorom określenie, jak elementy mają zachowywać się w layoutach.

`display`

Wartości: `inline` | `block` | `run-in` | `flex` | `grid` | `flow` | `flow-root` | `list-item` | `table` | `table-row-group` | `table-header-group` | `table-footer-group` | `table-row` | `table-cell` | `table-column-group` | `table-column` | `table-caption` | `ruby` | `ruby-base` | `ruby-text` | `ruby-base-container` | `ruby-text-container` | `inline-block` | `inline-table` | `inline-flex` | `inline-grid` | `contents` | `none`  
Domyślnie: `inline`  
Dotyczy: wszystkich elementów  
Dziedziczy: tak

Właściwość `display` definiuje typ pudełka elementu generowanego przez element w układzie. Oprócz znanych typów wyświetlania śródliniowego ( `inline` ) i blokowego ( `block` ) można również wyświetlać elementy jako elementy listy lub różne części tabeli. Istnieje również szereg wartości adnotacji `ruby` dla języków wschodnioazjatyckich. Jak widać z listy wartości, rodzajów wyświetlania jest wiele, ale tylko kilka z nich jest używanych w codziennej praktyce.

Przypisanie właściwości `display` jest przydatne do uzyskiwania efektów układu przy jednoczesnym zachowaniu nienaruszonej semantyki źródła `HTML` . Na przykład powszechną praktyką jest wyświetlanie elementów `li` (które zwykle wyświetlają się z cechami elementów blokowych) jako elementów wbudowanych, aby zamienić listę w poziomy pasek nawigacyjny. Możemy także wyświetlić element `a` (kotwica) w innym przypadku jako blok, aby nadać mu określoną szerokość i wysokość:

```
<!--display_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Display example</title>
  <style>
    ul.navigation li { display: inline; }
    ul.navigation li a { display: block; }
  </style>
</head>
<body>
  <ul class="navigation">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
    <li>One</li>
    <li>Two</li>
    <li>Three</li>
  </ul>
</body>
</html>
```

[Home](#)  
[About](#)  
[Contact](#)  
One Two Three

Inną użyteczną wartością właściwości `display` jest `none` , która całkowicie usuwa zawartość z normalnego przepływu. W przeciwieństwie do komendy `visibility: hidden` , która po prostu sprawia, że element jest niewidoczny, ale pozostawia pustą przestrzeń, którą by zajmował, `display: none` usuwa zawartość, a miejsce, które by zajmował, zostaje zamknięte.

Jednym z popularnych zastosowań `display: none` jest zapobieganie wyświetlaniu określonej zawartości dokumentu źródłowego na określonych nośnikach, na przykład podczas drukowania strony lub wyświetlania na urządzeniach z małymi ekranami. Można na przykład wyświetlać adresy `URL` łączy w dokumencie podczas drukowania, ale nie podczas wyświetlania na ekranie komputera, gdzie łączy są interaktywne.

Należy pamiętać, że zawartość, której `display` jest ustawione na `none` , nadal jest pobierana wraz z dokumentem. Ustawienie niektórych treści do `display: none` dla urządzeń z małymi ekranami może spowodować, że strona będzie krótsza, ale nie zmniejszy wykorzystania danych ani czasu pobierania.

```
<!--display_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Display example</title>
<style>
  ul.navigation li { display: inline; }
  ul.navigation li a { display: block; }
  p {
    display: none;
  }
</style>
</head>
<body>
  <ul class="navigation">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
    <li>One</li>
    <li>Two</li>
    <li>Three</li>
  </ul>
  <p>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus, adipisci, alias, aliquid amet animi
    atque autem.
  </p>
</body>
</html>
```

[Home](#)  
[About](#)  
[Contact](#)  
One Two Three

Dotarliśmy do ostatniego przystanku na trasie elementu `box`. Poznaliśmy właściwość `text-shadow`, która dodaje cień do tekstu. Właściwość `box-shadow` nakłada cień wokół całego widocznego pola elementu (z wyłączeniem marginesu).

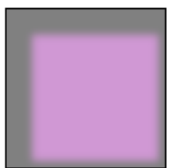
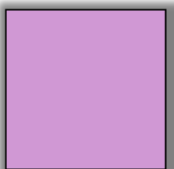
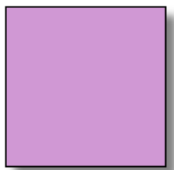
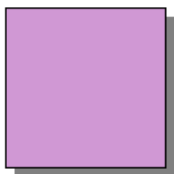
`box-shadow`

Wartości: 'horizontal offset' 'vertical offset' 'blur distance' 'spread distance' color  
inset | none  
Domyślnie: none  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

Wartość właściwości `box-shadow` powinna wydawać się znajoma po pracy z `text-shadow`: określamy odległość przesunięcia w poziomie i w pionie, stopień rozmycia cienia oraz kolor. W przypadku `box-shadow` można również określić wielkość rozrzutu, która zwiększa (lub zmniejsza przy wartościach ujemnych) rozmiar cienia. Domyślnie kolor cienia jest taki sam jak kolor pierwszego planu elementu, ale określenie koloru go zastępuje.

Możemy renderować cień wewnątrz krawędzi widocznego pola elementu, dodając słowo kluczowe `inset` do reguły. To sprawia, że wygląda to tak, jakby element był wciśnięty w ekran.

```
<!--box_shadow_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Box shadow example</title>
<style>
  div {
    width: 100px;
    height: 100px;
    background-color: #D098D4;
    border: 1px solid;
    margin-bottom: 2em;
  }
</style>
</head>
<body>
  <div style="box-shadow: 6px 6px gray;"></div>
  <div style="box-shadow: 6px 6px 5px gray;"></div>
  <div style="box-shadow: 6px 6px 5px 10px gray;"></div>
  <div style="box-shadow: inset 6px 6px 5px 10px gray;"></div>
</body>
</html>
```



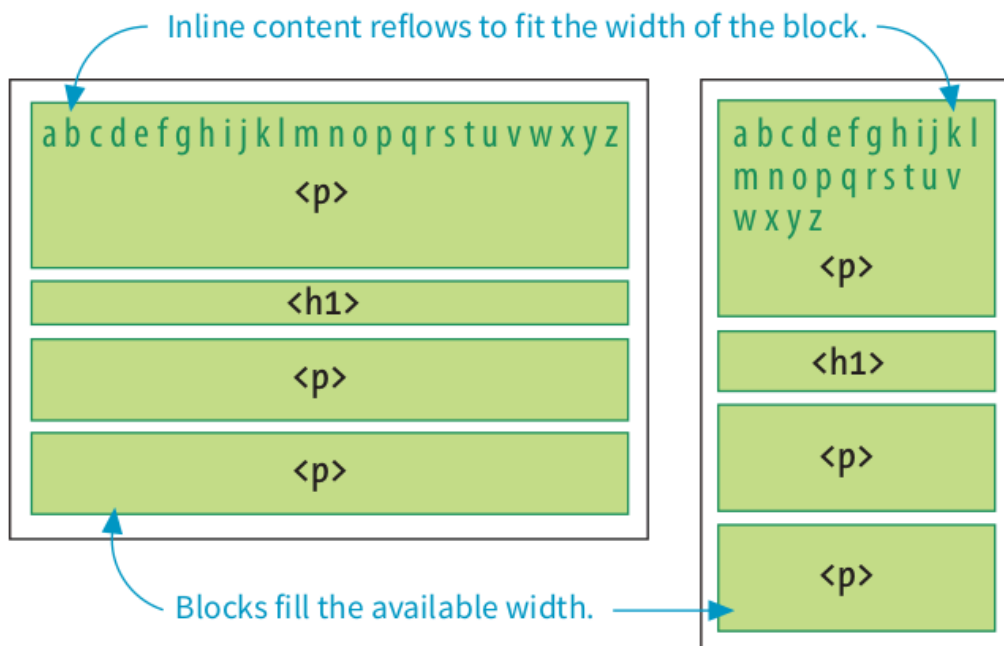
## Step 8

W modelu układu `CSS` elementy tekstowe są układane od góry do dołu w kolejności, w jakiej pojawiają się w źródle, oraz od lewej do prawej w językach czytanych od lewej do prawej. Elementy blokowe układają się jeden na drugim i wypełniają dostępną szerokość okna przeglądarki lub innego elementu zawierającego. Elementy śródliniowe i znaki tekstowe ustawiają się obok siebie, aby wypełnić elementy blokowe.

Kiedy zmienia się rozmiar okna lub elementu zawierającego, elementy blokowe rozszerzają się lub kurczą do nowej szerokości, a treść wstawiana jest ponownie wlewana.

Obiekty w normalnym przepływie wpływają na układ obiektów wokół nich. Jest to zachowanie, którego można się spodziewać na stronach internetowych — elementy nie nakładają się ani nie gromadzą. Robią sobie miejsce.

Widzieliśmy to wszystko już wcześniej, ale teraz skupimy się na tym, czy elementy znajdują się w przepływie, czy też są z niego usuwane. **Pozycjonowanie elementów** (ang. `floating`) zmienia stosunek elementów do normalnego przepływu na różne sposoby. Przyjrzyjmy się najpierw specjalnemu zachowaniu elementów `float` (lub w skrócie `float`).



Mówiąc najprościej, właściwość `float` przesuwa element jak najdalej w lewo lub w prawo, umożliwiając zawinięcie następującej treści wokół niego. Jest to unikalna funkcja wbudowana w `CSS` z kilkoma interesującymi zachowaniami.

`float`

Wartości: `left` | `right` | `none`

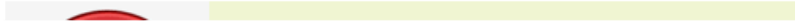
Domyślna wartość: `none`

Dotyczy: wszystkich elementów

Dziedziczy: nie

Najlepszym sposobem wyjaśnienia `float` jest zademonstrowanie go. W tym przykładzie właściwość `float` jest stosowana do elementu `img` w celu przesunięcia go w prawo. Domyślnie renderowany jest akapit i zawarty w nim obraz (na górze) oraz jak wygląda po zastosowaniu właściwości `float` (na dole).

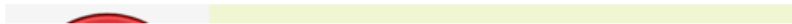
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float example</title>
<style>
  p {
    background-color: #f2f5d5;
    width: 500px;
  }
</style>
</head>
<body>
  <p>
    
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse
    interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare
    aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed
    lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
    rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus
    risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis
    blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit
    ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit
    egestas dui id ornare arcu odio.
  </p>
  <p>
    
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse
    interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare
    aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed
    lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
    rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus
    risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis
    blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit
    ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit
    egestas dui id ornare arcu odio.
  </p>
</body>
</html>
```



To niezły efekt. Pozbyliśmy się dużej ilości zmarnowanego miejsca na stronie, ale teraz tekst nachodzi na obraz. Dodajmy `1em` do marginesu ze wszystkich stron obrazu za pomocą właściwości `margin`. Możemy zacząć widzieć, jak właściwości pudełka współpracują ze sobą, aby poprawić układ strony.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float example</title>
<style>
  p {
    background-color: #f2f5d5;
    width: 500px;
  }
</style>
</head>
<body>
  <p>
    
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse
    interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare
    aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed
    lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
    rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus
    risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis
    blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit
    ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit
    egestas dui id ornare arcu odio.
  </p>
  <p>
    
    Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
    dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse
    interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare
    aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed
    lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
    rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus
    risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis
    blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit
    ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit
    egestas dui id ornare arcu odio.
  </p>
</body>
</html>
```



Poprzednie dwa rysunki przedstawiają niektóre kluczowe zachowania pływających elementów:

- **Element pozycjonowany jest jak wyspa na strumieniu.** Przede wszystkim widać, że obraz jest usuwany ze swojej pozycji w normalnym przepływie, ale nadal wpływa na otaczającą treść. Tekst kolejnego akapitu zostanie ponownie wlany, aby zrobić miejsce dla pozycjonowanego elementu `img`. Jedną z popularnych analogii porównuje konary drzew w strumieniu – nie są one w nurcie, ale strumień musi je opływać. To zachowanie jest unikalne dla elementów pozycjonowanych.
- **Element pozycjonowany pozostaje w obszarze zawartości elementu zawierającego.** Należy również zauważyć, że pozycjonowany element obrazu jest umieszczony w obszarze treści (wewnętrzne krawędzie) akapitu, który go zawiera. Nie obejmuje obszaru wypełnienia akapitu.
- **Marginesy są zachowane.** Ponadto marginesy są utrzymywane ze wszystkich stron pozycjonowanego obrazu, jak pokazano na powyższym przykładzie. Innymi słowy, całe pole elementu, od zewnętrznej krawędzi do zewnętrznej krawędzi, jest unoszone.

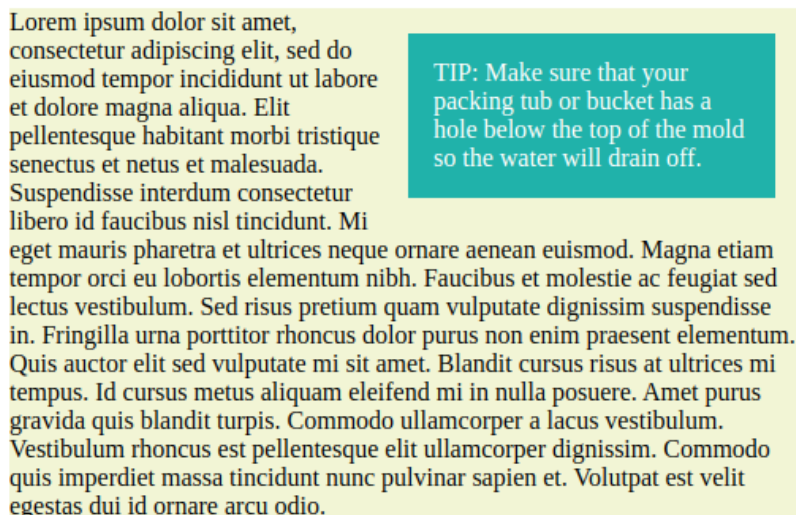
To są podstawy, więc teraz spójrzmy na więcej przykładów i zbadajmy dodatkowe zachowania elementów pozycjonowanych. Jak zobaczymy w poniższych przykładach, możliwe jest umieszczenie dowolnego elementu `HTML`, zarówno w wierszu, jak i na poziomie bloku.

W poprzednim przykładzie umieściliśmy osadzony element obrazu. Tym razem przyjrzyjmy się, co się stanie, gdy umieścimy element tekstu w wierszu – w tym przypadku fragment tekstu.

```

<!--float_example2.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float example 2</title>
<style>
  p {
    background-color: #f2f5d5;
    width: 500px;
  }
  span.tip {
    float: right;
    margin: 1em;
    width: 200px;
    color: #fff;
    background-color: lightseagreen;
    padding: 1em;
  }
</style>
</head>
<body>
<p>
  <span class="tip">TIP: Make sure that your packing tub or bucket
has a hole below the top of the mold so the water will drain
off.</span>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse
interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare
aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed
lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus
risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis
blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit
ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit
egestas dui id ornare arcu odio.
</p>
</body>
</html>

```



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.

TIP: Make sure that your packing tub or bucket has a hole below the top of the mold so the water will drain off.

Na pierwszy rzut oka zachowuje się tak samo jak pozycjonowany wcześniej obraz, czego można się spodziewać. Ale działają tu pewne subtelne rzeczy, na które warto zwrócić uwagę:

- Zawsze podawaj szerokość pozycjonowanych elementów tekstowych.** Po pierwsze, zauważmy, że reguła stylu określająca `span` obejmuje właściwość `width`. Konieczne jest określenie szerokości pozycjonowanego elementu tekstowego, ponieważ bez niego jego pole jest wystarczająco szerokie, aby zmieściło się w nim zawartość ( `auto` ). W przypadku krótkich fraz, które są węższe niż kontener, może to nie stanowić problemu. Jednak w przypadku dłuższego, zawiniętego tekstu pole rozszerza się do szerokości kontenera, czyniąc go tak szerokim, że nie byłoby miejsca na zawinięcie czegokolwiek wokół niego. Obrazy mają nieodłączną szerokość, więc nie musieliśmy określać szerokości w poprzednim przykładzie (choć z pewnością moglibyśmy to zrobić).

- **Pozycjonowane elementy śródliniowe zachowują się jak elementy blokowe.** Zauważmy, że margines jest utrzymywany ze wszystkich czterech stron pływającego tekstu `span`, mimo że górny i dolny margines zwykle nie są renderowane w elementach śródliniowych. Dzieje się tak, ponieważ wszystkie elementy pozycjonowane zachowują się jak elementy blokowe. Po przestawieniu elementu w wierszu jest on zgodny z regułami wyświetlania elementów na poziomie bloku, a marginesy są renderowane ze wszystkich czterech stron.
- **Marginesy pozycjonowanych elementów nie zawijają się.** W normalnym przepływie stykające się marginesy górny i dolny zapadają się (nachodzą na siebie), ale marginesy elementów pozycjonowanych są zachowywane ze wszystkich stron zgodnie z ustawieniami.

Przyjrzyjmy się, co się dzieje, gdy pozycjonujemy blok w normalnym przepływie. W tym przykładzie cały drugi element akapitu jest przesuwany w lewo.

```
<!--float_example_3.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float example 3</title>
<style>
  p {
    border: 2px red solid;
  }
  #float {
    float: left;
    width: 300px;
    margin: 1em;
    background: white;
  }
</style>
</head>
<body>
<p>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.
</p>
<p id="float">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.
</p>
<p>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.
</p>
<p>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.
</p>
<p>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
```

```
rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus  
risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis  
blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit  
ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit  
egestas dui id ornare arcu odio.  
</p>  
</body>  
</html>
```

- Dodano czerwoną obwódkę wokół wszystkich elementów `p`, aby odsłonić ich granice. Ponadto zmieniono tło pozycjonowanego akapitu na `papaywhit`, aby się wyróżniało, i dodano margines `1em` ze wszystkich stron.
- Tak jak widzieliśmy na obrazku, akapit przesuwają się na bok (tym razem w lewo), a następna treść zawija się wokół niego, mimo że bloki normalnie układają się jeden na drugim. W tym przykładzie chcemy zwrócić uwagę na kilka rzeczy:
- **Musimy podać szerokość pozycjonowanych elementów blokowych.** Jeśli nie podamy wartości szerokości, szerokość pozycjonowanego bloku zostanie ustawiona na `auto`, co wypełni dostępną szerokość okna przeglądarki lub innego elementu zawierającego. Nie ma większego sensu posiadanie pozycjonowanego pola o pełnej szerokości, ponieważ chodzi o to, aby zawijać tekst obok pozycjonowanego elementu, a nie zaczynać pod nim.
  - **Elementy nie pozycjonują się wyżej niż ich odniesienie w źródle.** Pozycjonowany blok będzie unosił się w lewo lub w prawo względem miejsca, w którym występuje w źródle, umożliwiając owijanie się wokół niego następujących elementów przepływu. Pozostaje poniżej elementów blokowych, które go poprzedzają w przepływie (w efekcie jest przez nie „blokowany”). Oznacza to, że nie możemy przenieść elementu do górnego rogu strony, nawet jeśli jego najbliższym przodkiem jest element `body`. Jeśli chcesz, aby element pozycjonowany zaczynał się u góry strony, musi pojawić się jako pierwszy w źródle dokumentu.
  - **Elementy niepozycjonowane utrzymują normalny przepływ.** Czerwone obramowania na górnym obrazie pokazują, że pola elementów dla otaczających akapitów nadal rozciągają się na całą szerokość normalnego przepływu. Tylko zawartość tych elementów owija się wokół elementu pozycjonowanego. To dobry model, o którym warto pamiętać. Na przykład dodanie lewego marginesu do otaczających akapitów spowodowałoby zwiększenie przestrzeni na lewej krawędzi strony, a nie między tekstem a elementem pozycjonowanym. Jeśli chcemy zachować odstęp między elementem pozycjonowanym a zawiniętym tekstem, musimy zastosować margines do samego elementu pozycjonowanego.

## Step 9

---

Jeśli zamierzamy przesuwac elementy wokół, ważne jest, aby wiedzieć, jak wyłączyć zawijanie tekstu i jak zwykle wrócić do normalnego przepływu. Robimy to, czyszcząc element, który chcemy rozpocząć poniżej elementu pozycjonowanego. Zastosowanie właściwości `clear` do elementu zapobiega pojawianiu się obok pozycjonowanego elementu i zmusza go do rozpoczęcia od następnej dostępnej „czystej” przestrzeni poniżej elementu pozycjonowanego.

`clear`

Wartości: `left` | `right` | `both` | `none`

Domyślna wartość:

Dotyczy: elementów blokowych

Inherits: nie

Pamiętajmy, aby stosować właściwość `clear` do elementu, który chcemy rozpocząć pod elementem pozycjonowanym, a nie do samego elementu pozycjonowanego. Wartość `left` rozpoczyna element poniżej elementów, które zostały przesunięte w lewo. Podobnie, wartość `right` sprawia, że element usuwa wszystkie elementy pozycjonowane na prawej krawędzi bloku zawierającego. Jeśli istnieje wiele elementów pozycjonowanych i chcemy mieć pewność, że element zaczyna się pod nimi wszystkimi, należy użyć wartości obu, aby wyczyścić elementy pozycjonowane po obu stronach.

W tym przykładzie właściwość `clear` została użyta, aby elementy `h2` zaczynały się poniżej elementów pozycjonowanych w lewo.

```
<!--clear_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Clear example</title>
<style>
  img {
    float: left;
    margin-right: .5em;
  }
  h2 {
    clear: left;
    margin-top: 2em;
  }
</style>
</head>
<body>
<p>
  
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.
</p>
<h2>This is a heading</h2>
<p>
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.
</p>
</body>
</html>
```



Bez właściwości `clear: left` strona wyglądałaby następująco:



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## This is a heading

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Zwróćmy uwagę że chociaż do elementu `h2` zastosowano górny margines o szerokości `2em`, nie jest on renderowany między nagłówkiem a pozycjonowanym obrazem. Jest to wynikiem zapadania się pionowych marginesów w przepływie. Jeśli chcemy mieć pewność, że między elementem pozycjonowanym a następującym tekstem zachowana jest spacja, należy zastosować dolny margines do samego pozycjonowanego elementu.

Umieszczanie wielu elementów na stronie lub nawet w jednym elemencie jest całkowicie w porządku. W rzeczywistości od lat elementy pozycjonowane są podstawową metodą wyrównywania elementów, takich jak menu nawigacyjne, a nawet tworzenia układów całej strony.

Gdy pozycjonujemy wiele elementów, istnieje złożony system reguł renderowania za kulisami, który zapewnia, że elementy pozycjonowane się nie nakładają się. Możemy zapoznać się ze specyfikacją `CSS`, aby uzyskać szczegółowe informacje, ale wynikiem tego jest to, że elementy pozycjonowane zostaną umieszczone tak daleko w lewo lub w prawo (zgodnie z określeniem) i tak wysoko, jak pozwala na to miejsce.

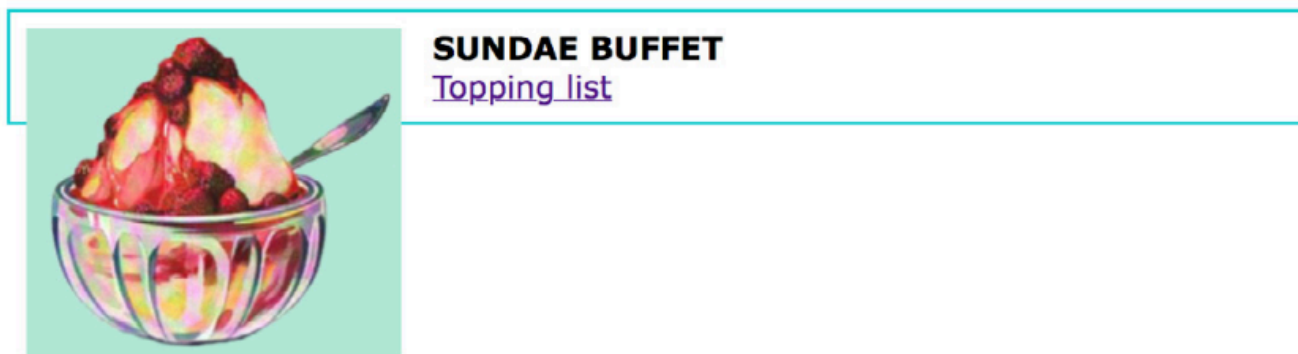
Przykład poniżej pokazuje, co się stanie, gdy seria kolejnych akapitów zostanie przesunięta na tę samą stronę. Pierwsze trzy elementy pozycjonowane zaczynają się układać od lewej krawędzi, ale kiedy nie ma wystarczająco dużo miejsca na czwarty, przesuwa się on w dół i w lewo, aż na coś wpadnie – w tym przypadku na krawędź okna przeglądarki. Gdyby jednak jeden z elementów pozycjonowanych, taki jak `P2`, był bardzo długi, zamiast tego uderzyłby o krawędź długiego elementu pozycjonowanego. Zauważmy, że następny akapit w normalnym przepływie (`P6`) zaczyna się zawiijać w najwyższym punkcie, jaki można znaleźć, tuż poniżej `P1`.

```
<!--multiple_float_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Multiple float example</title>
<style>
  p.float {
    float: left;
    width: 200px;
    margin: 0;
    background: #F2F5d5;
    color: #DAEAB1;
    border: 2px solid;
  }
  p {
    border: 2px solid;
  }
</style>
</head>
<body>
  <p>[PARAGRAPH 1] Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus
  et malesuada. </p>
  <p class="float">[P2]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus
  et malesuada. </p>
  <p class="float">[P3]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus
  et malesuada. </p>
  <p class="float">[P4]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus
  et malesuada. </p>
  <p class="float">[P5]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
  incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus
  et malesuada. </p>
  <p>[P6]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
  labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
  </p>
  <p>[P7]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
  labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
  </p>
  <p>[P8]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
  labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
  </p>
  <p>[P9]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
  labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
  </p>
  <p>[P10]Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
  labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
  </p>
</body>
</html>
```



To dobry moment, aby zająć się dziwnym zachowaniem elementów pozycjonowanych: powstrzymywanie pozycjonowania. Domyślnie elementy pozycjonowane są zaprojektowane tak, aby zwiisały z elementu, w którym się znajdują. To wystarczy, aby umożliwić przepływ tekstu wokół pozycjonowanego obrazu, ale czasami może powodować niepożądane zachowania.

Spójrmy na przykład poniżej. Byłoby ładniej, gdyby obramowanie rozszerzyło się wokół całej zawartości, ale pozycjonowany obraz zwisa z dołu.



Jeśli umieścimy wszystkie elementy w elemencie kontenera, w przepływie nie pozostaną żadne elementy, które utrzymają otwarty element zawierający. Zjawisko to zilustrowano poniżej. `#container div` zawiera dwa akapity. Widok normalnego przepływu (u góry) pokazuje, że `#container` ma kolor tła i obramowanie otaczające zawartość.

```
<!--float_containment_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float containment example</title>
<style>
  #container {
    background: #f2f5d5;
    border: 2px dashed green;
  }
</style>
</head>
<body>
<div id="container">
  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices
neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac
feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna
porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet.
Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet
purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est
pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et.
Volutpat est velit egestas dui id ornare arcu odio.</p>
  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices
neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac
feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna
porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet.
Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet
purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est
pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et.
Volutpat est velit egestas dui id ornare arcu odio.</p>
</div>

</body>
</html>
```

---

Jednak gdy oba akapity (tj. całą zawartość elementu `div` ) są pozycjonowane, jak pokazano na rysunku na dole), pole elementu `#container` zamyka się do wysokości zera, pozostawiając elementy pozycjonowane zwisające poniżej (nadal możemy zobaczyć puste obramowanie u góry). W normalnym przepływie nie ma treści, która mogłaby podać wysokość elementu `div` . Zdecydowanie nie o taki efekt nam chodzi.

```
<!--float_containment_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float containment example</title>
<style>
  #container {
    background: #f2f5d5;
    border: 2px dashed green;
  }
  p {
    float: left;
    width: 44%;
    padding: 2%;
  }
</style>
</head>
<body>
<div id="container">
  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices
neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac
feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna
porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet.
Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet
purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est
pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et.
Volutpat est velit egestas dui id ornare arcu odio.</p>
  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices
neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac
feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna
porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet.
Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet
purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est
pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et.
Volutpat est velit egestas dui id ornare arcu odio.</p>
</div>

</body>
</html>
```

Na szczęście istnieje rozwiązanie tego problemu i są one dość proste. Nadajmy elementowi `container` szerokość `100%` oraz `float: left` .

```
<!--float_containment_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Float containment example</title>
<style>
  #container {
    float: left;
    width: 100%;
    background-color: #f2f5d5; /*light green*/
    border: 2px dashed green;
    padding: 1em;
  }
  p {
    float: left;
    width: 44%;
    padding: 2%;
  }
</style>
</head>
<body>
<div id="container">
  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices
neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac
feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna
porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet.
Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet
purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est
pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et.
Volutpat est velit egestas dui id ornare arcu odio.</p>
  <p> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore
et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada.
Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices
neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac
feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna
porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet.
Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet
purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est
pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et.
Volutpat est velit egestas dui id ornare arcu odio.</p>
</div>

</body>
</html>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.

## Step 10

Spójrzmy na poprzednie przykłady elementów pozycjonowanych, a zobaczymy, że tekst zawsze zawija się w prostokątny kształt wokół pozycjonowanego obrazu lub pola elementu. Możemy jednak zmienić kształt zawiniętego tekstu na okrąg, elipsę, wielokąt lub dowolny kształt obrazu, używając właściwości `shape-outside`.

`shape-outside`

Wartości: `none` | `circle()` | `ellipse()` | `polygon()` | `url()` | `[margin-box | padding-box | content-box]`  
Domyślna wartość: `none`  
Dotyczy: pozycjonowane elementy  
Dziedziczy: nie

Warto zauważyć, że można zmienić kształt "ramki" tekstu wokół dowolnego pozycjonowanego elementu, ale w tym omówieniu skupimy się na obrazach, ponieważ elementy tekstowe to na ogół ramki, które dobrze pasują do domyślnego prostokątnego pozycjonowania.

Istnieją dwa podejścia do zawijania tekstu wokół kształtu. Jednym ze sposobów jest podanie współrzędnych ścieżki kształtu zawijania za pomocą funkcji `circle()`, `ellipse()` lub `polygon()`. Innym sposobem jest użycie `url()` do określenia obrazu, który ma przezroczyste obszary (takie jak `GIF` lub `PNG`). W przypadku metody obrazkowej tekst przepływa do przezroczystych obszarów obrazu i zatrzymuje się w obszarach nieprzezroczystych.

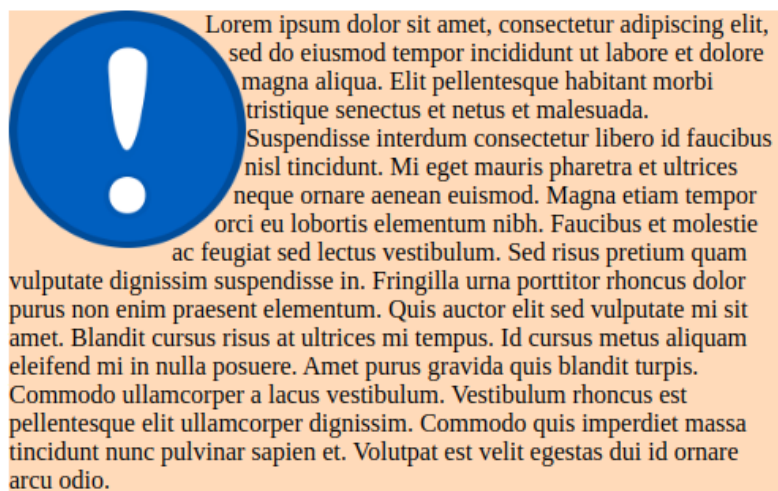
W przykładzie poniżej umieszczono obraz `sundae.png` w dokumencie `HTML`, który ma być wyświetlany na stronie, i określono ten sam obraz w regule stylu za pomocą `url()` tak, aby jego przezroczyste obszary definiowały zawijanie kształtu. Sensowne jest użycie tego samego obrazu w dokumencie i dla kształtu `CSS`, ale nie jest to wymagane. Możemy zastosować kształt zawijania pochodzący z jednego obrazu do innego obrazu na stronie.

```

<!--transparent_image_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Transparent image example</title>
<style>
  img.wrap {
    float: left;
    width: 150px;
    height: 150px;
    shape-outside: url(images/mark.png);
  }
  p {
    background-color: peachpuff;
  }
</style>
</head>
<body>
<p>  Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit
pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur
libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna
etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed
risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non
enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi
tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis.
Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim.
Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare
arcu odio.</p>

</body>
</html>

```



Zauważmy, że zawinięty tekst wpada teraz bezpośrednio na obraz. Możemy dać mu trochę więcej miejsca z `shape-margin`.

`shape-margin`

Wartości: length | percentage

Domyślna wartość: 0

Dotyczy: pozycjonowanych elementów

Dziedziczy: nie

Właściwość `shape-margin` określa ilość miejsca między kształtem a zawiniętym tekstem. W poniższym przykładzie widać efekt dodania `1em` odstępu między nieprzezroczystymi obszarami obrazu a zawiniętymi wierszami tekstu. Daje mu to trochę miejsca do oddychania, tak jak powinien to zrobić każdy dobry margines.



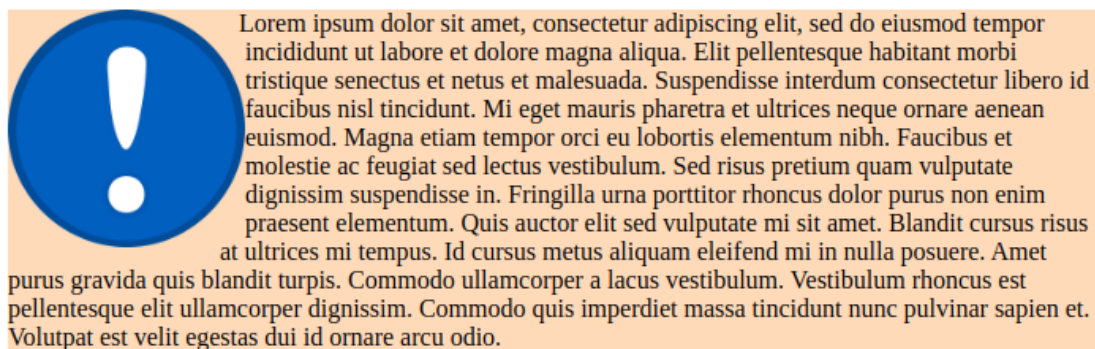
```

<!--transparent_image_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Transparent image example</title>
<style>
  img.wrap {
    float: left;
    width: 150px;
    height: 150px;
    shape-margin: 1em;
    shape-outside: url(images/mark.png);

  }
  p {
    background-color: peachpuff;
  }
</style>
</head>
<body>
<p> 
  Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et
  dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse
  interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare
  aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed
  lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor
  rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus
  risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis
  blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit
  ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit
  egestas dui id ornare arcu odio.</p>

</body>
</html>

```



Inną metodą tworzenia kształtu otaczającego tekst jest zdefiniowanie go za pomocą jednego ze słów kluczowych: `circle()`, `ellipse()` i `polygon()`. Notacja `circle()` tworzy kształt okręgu, wokół którego można zawijać tekst. Wartość podana w nawiasach oznacza długość promienia okręgu. W tym przykładzie promień wynosi 150 pikseli, czyli połowę szerokości obrazu wynoszącej 300 pikseli. Domyślnie okrąg jest wyśrodkowany w pionie i poziomie w elemencie pozycjonowanym:

```

img.round {
  float: left;
  shape-outside: circle(150px);
}

```

Rysunek poniżej przedstawia tę regułę stylu zastosowaną do różnych obrazów. Zauważmy, że przezroczystość obrazu nie ma tutaj znaczenia. To tylko ścieżka nałożona na obraz, która wyznacza granice zawijania tekstu. Dowolną ścieżkę można zastosować do dowolnego obrazu lub innego pływającego elementu.

---

Jest to dobry moment, aby zademonstrować krytyczne zachowanie `shape-outside`. Pozwalają na wpłynięcie tekstu do pływającego obrazu lub elementu, ale nie mogą utrzymać wolnego miejsca poza nim.

W kolejnym przykładzie zwiększono średnicę ścieżki okręgu ze 150 pikseli do 200 pikseli. Zwróćmy uwagę, że tekst jest wyrównany wzdłuż prawej krawędzi obrazu, mimo że okrąg znajduje się 50 pikseli poza krawędzią. Ścieżka nie wypycha tekstu z elementu pozycjonowanego. Jeśli chcemy, aby zawinięty tekst znajdował się z dala od zewnętrznej krawędzi pozycjonowanego obrazu lub elementu, należy zastosować margines do samego elementu (oczywiście będzie to standardowy prostokątny kształt).

```
img.round {  
  float: left;  
  shape-outside: circle(200px);  
}
```

---

Kształty eliptyczne są tworzone za pomocą notacji `ellipse()`, która podaje długość promienia poziomego i pionowego, po których następuje słowo `at`, a następnie współrzędne `x`, `y` środka kształtu. Współrzędne pozycji mogą być podane jako konkretny pomiar lub procent. Tutaj utworzono elipsę o promieniu poziomym 100 pikseli i promieniu pionowym 150 pikseli, wyśrodkowaną w pozycjonowanym elemencie, do którego jest stosowana.

```
img.round {  
  float: left;  
  shape-outside: ellipse(200px 100px at 50% 50%);  
}
```

Na koniec dochodzimy do funkcji `polygon()`, która pozwala utworzyć niestandardową ścieżkę przy użyciu serii współrzędnych `x, y` oddzielonych przecinkami wzdłuż ścieżki.

```
img.wrap {  
  float: left;  
  width: 300px;  
  height: 300px;  
  shape-outside: polygon(0px 0px, 186px 0px, 225px 34px, 300px 34px,  
    300px 66px, 255px 88px, 267px 127px, 246px 178px, 192px 211px, 226px  
    236px, 226px 273px, 209px 300px, 0px 300px);  
}
```

## Step 11

CSS udostępnia kilka metod pozycjonowania elementów na stronie. Można je umieścić względem miejsca, w którym zwykle pojawiają się w przepływie, lub całkowicie usunąć z przepływu i umieścić w określonym miejscu na stronie. Możemy także ustawić element względem widocznego obszaru, który pozostanie na swoim miejscu podczas przewijania reszty strony.

`position`

Wartości: `static` | `relative` | `absolute` | `fixed`  
Domyślna wartość: `static`  
Dotyczy: wszystkich elementów  
Dziedziczy: nie

Właściwość `position` wskazuje, że element ma być pozycjonowany i określa, której metody pozycjonowania należy użyć. W tym miejscu pokrótce przedstawimy każdą wartość słowa kluczowego, a następnie przyjrzymy się każdej metodzie bardziej szczegółowo w dalszej części.

- `static` - Jest to normalny schemat pozycjonowania, w którym elementy są umieszczane tak, jak występują w normalnym obiegu dokumentów.

- `relative` - Pozycjonowanie względne przesuwa pole elementu względem jego pierwotnej pozycji w przepływie. Charakterystyczne zachowanie względnego pozycjonowania polega na tym, że przestrzeń, którą element zajmowałby w normalnym przepływie, jest zachowywana jako pusta przestrzeń.
- `absolute` - Elementy pozycjonowane bezwzględnie są całkowicie usuwane z obiegu dokumentów i ustawiane względem **rzutni** (ang. `viewport` ) lub elementu zawierającego. W przeciwieństwie do elementów pozycjonowanych relatywnie, przestrzeń, którą by zajmowały, jest zamknięta. W rzeczywistości nie mają one żadnego wpływu na układ otaczających elementów.
- `fixed` - Cechą charakterystyczną stałego pozycjonowania jest to, że element pozostaje w jednym miejscu w oknie podglądu, nawet gdy dokument jest przewijany. Stałe elementy są usuwane z obiegu dokumentów i umieszczane względem widocznego obszaru, a nie innego elementu w dokumencie.
- `sticky` - Pozycjonowanie sticky jest kombinacją względnego i stałego, ponieważ zachowuje się tak, jakby było ustawione względnie, dopóki nie zostanie przewinięte do określonej pozycji względem rzutni, w którym to momencie pozostaje nieruchome.

Każda metoda pozycjonowania ma swój cel, ale pozycjonowanie bezwzględne jest najbardziej uniwersalne. Dzięki pozycjonowaniu bezwzględnemu możemy umieścić obiekt w dowolnym miejscu rzutni lub w innym elemencie. Pozycjonowanie bezwzględne było używane do tworzenia układów wielokolumnowych, ale jest częściej używane do mniejszych zadań, takich jak umieszczanie pola wyszukiwania w górnym rogu nagłówka. Jest to przydatne narzędzie, gdy jest używane ostrożnie i oszczędnie.

Po ustaleniu metody pozycjonowania rzeczywista pozycja jest określana za pomocą kombinacji do czterech właściwości przesunięcia.

`top, right, bottom, left`

Wartości: `length` | `percentage` | `auto`

Domyślna wartość: `auto`

Dotyczy: elementów z ustawioną własnością `position`

Dziedziczy: nie

Wartość podana dla każdej właściwości odsunięcia określa odległość, na jaką element powinien zostać odsunięty od odpowiedniej krawędzi. Na przykład wartość `top` określa odległość, na jaką górna zewnętrzna krawędź pozycjonowanego elementu powinna być przesunięta od górnej krawędzi przeglądarki lub innego elementu zawierającego. Dodatnia wartość góry powoduje przesunięcie pola elementu w dół o tę wartość. Podobnie dodatnia wartość dla `left` przesunie pozycjonowany element w prawo (w kierunku środka bloku zawierającego) o tę wartość.

Dalsze wyjaśnienia i przykłady właściwości przesunięcia zostaną podane w omówieniu każdego sposobu pozycjonowania.

Rozpoczniemy naszą eksplorację pozycjonowania od dość prostej metody względnej (ang. `relative` ).

Jak wspomniano wcześniej, pozycjonowanie względne przesuwa element względem jego pierwotnego miejsca w przepływie.

Przestrzeń, którą by zajmowała, zostaje zachowana i nadal wpływa na układ otaczających treści. Łatwiej to zrozumieć na prostym przykładzie.

Tutaj umieszczono wbudowany element `em` . Jasny kolor tła na `em` i obramowanie na zawierającym je akapicie sprawiają, że ich granice są widoczne. Najpierw użyto właściwości `position` , aby ustawić metodę na `relative` , a następnie użyto właściwości `top` , aby przesunąć element o `2em` w dół od jego początkowej pozycji, oraz właściwości `left` , aby przesunąć go o `3em` w prawo.

Pamiętajmy, że wartości właściwości przesunięcia odsuwają element od określonej krawędzi, więc jeśli chcemy, aby coś przesunęło się w prawo, tak jak ja tutaj, należy użyć właściwości przesunięcia w lewo.

```
<!--relative_position.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Relative position example</title>
<style>
  p {
    background-color: peachpuff;
  }
  em {
    position: relative;
    top: 2em;
    left: 3em;
    background-color: fuchsia;
  }
</style>
</head>
<body>
<p> Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>

</body>
</html>
```

Lorem ipsum dolor sit amet, , sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum **consectetur adipiscing elit** faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.

Tutaj dzieje się parę rzeczy:

- **Oryginalna przestrzeń w obiegu dokumentów zostaje zachowana.** Możemy zobaczyć, że w miejscu, w którym znalazłby się wyróżniony tekst, gdyby element nie został umieszczony, jest puste miejsce. Otaczająca treść jest ułożona tak, jakby element nadal tam był, dlatego mówimy, że element nadal „wpływa” na otaczającą treść.
- **Nakładanie na inny element może się wydarzyć.** Ponieważ jest to element pozycjonowany, może potencjalnie nachodzić na inne elementy, jak to ma miejsce w przykładzie.

Pusta przestrzeń pozostawiona przez względnie ustawione obiekty może być trochę niezręczna, więc ta metoda nie jest używana tak często jak pozycjonowanie bezwzględne. Jednak pozycjonowanie względne jest powszechnie używane do tworzenia „kontekstu pozycjonowania” dla elementu pozycjonowanego bezwzględnie.

**Pozycjonowanie bezwzględne** (ang. `absolute` ) działa nieco inaczej i jest bardziej elastyczną metodą dokładnego umieszczania elementów na stronie niż pozycjonowanie względne. Teraz, gdy już wiemy, jak działa pozycjonowanie względne, weźmy ten sam przykład, tylko tym razem zmienimy wartość właściwości `position` na `absolute` .

```

<!--absolute_position_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Absolute position example</title>
  <style>
    p {
      background-color: peachpuff;
    }
    em {
      position: absolute;
      top: 2em;
      left: 3em;
      background-color: fuchsia;
    }
  </style>
</head>
<body>
<p> Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>

</body>
</html>

```

Lorem ipsum dolor sit amet, , sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellen **consectetur adipiscing elit** que senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.

Jak widać, przestrzeń zajmowana niegdyś przez element `em` jest teraz zamknięta, podobnie jak w przypadku wszystkich elementów pozycjonowanych bezwzględnie. W nowej pozycji pole elementu nachodzi na otaczającą zawartość. Ostatecznie elementy pozycjonowane bezwzględnie nie mają żadnego wpływu na układ elementów otaczających.

Najważniejszą różnicą jest tu jednak położenie pozycjonowanego elementu. Tym razem wartości przesunięcia ustawiają element `em` `2em` w dół i `3em` na prawo od lewego górnego rogu rzutni (okna przeglądarki).

To, co faktycznie dzieje się w pozycjonowaniu bezwzględnym, polega na tym, że element jest pozycjonowany względem najbliższego bloku zawierającego. Tak się składa, że najbliższym blokiem zawierającym jest element główny ( `html` ), znany również jako początkowy blok zawierający, więc wartości przesunięcia ustawiają element `em` względem całego dokumentu. Zapoznanie się z koncepcją bloku zawierającego jest pierwszym krokiem do rozwiązania problemu pozycjonowania bezwzględnego.

Moduł układu pozycjonowanego `CSS`, poziom 3, stwierdza: „Położenie i rozmiar pola elementu są czasami obliczane względem określonego prostokąta, zwanego blokiem zawierającym element”. Bardzo ważne jest, aby być świadomym bloku zawierającego element, który chcemy umieścić. Czasami nazywamy to **kontekstem pozycjonowania**.

Specyfikacja określa szereg skomplikowanych reguł określania zawierającego bloku elementu, ale zasadniczo sprowadza się to do tego:

- Jeśli element który chcemy pozycjonować nie jest zawarty w innym elemencie pozycjonowanym, to zostanie umieszczony względem początkowego bloku zawierającego (utworzonego przez element `html` ).

- Ale jeśli element ma przodka (tj. jest zawarty w elemencie), którego pozycja jest ustawiona na względną, bezwzględną lub stałą, zamiast tego element zostanie umieszczony względem krawędzi tego elementu.

Poprzedni przykład ilustrował pierwszy przypadek: element `p`, który zawiera element `em` pozycjonowany bezwzględnie, sam nie jest pozycjonowany i nie ma innych elementów pozycjonowanych wyżej w hierarchii. Dlatego element `em` jest umieszczony względem początkowego bloku zawierającego, co odpowiada obszarowi rzutni.

Celowo zamienimy element `p` w blok zawierający i zobaczmy, co się stanie. Wszystko, co musimy zrobić, to zastosować do niego właściwość `position`; nie musimy go właściwie przenosić. Najczęstszym sposobem uczynienia elementu blokiem zawierającym jest ustawienie jego pozycji na `relative`, ale bez przesuwania go z wartościami przesunięcia. Jest to przykład, kiedy pozycjonowanie względne jest używane do tworzenia kontekstu pozycjonowania dla elementu pozycjonowanego bezwzględnie.

W tym przykładzie zachowamy tę samą regułę stylu dla elementu `em`, ale dodamy właściwość `position` do elementu `p`, czyniąc go w ten sposób blokiem zawierającym pozycjonowany element `em`. RYSUNEK 15-21 przedstawia wyniki.

```
<!--positioning_context_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Absolute position example</title>
  <style>
    p {
      position: relative;
      padding: 15px;
      background-color: #F2F5D5;
      border: 2px solid purple;
    }
    em {
      position: absolute;
      top: 2em;
      left: 3em;
      background-color: fuchsia;
    }
  </style>
</head>
<body>
<p>  Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>

</body>
</html>
```

>Lorem ipsum dolor sit amet, , sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pelle<strong>consectetur adipiscing elit</strong>tique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.

Widać, że element `em` jest teraz umieszczony `2em` w dół i `3em` od lewego górnego rogu akapitu, a nie okna przeglądarki. Zauważmy również, że jest on umieszczony względem krawędzi wypełnienia akapitu (tuż wewnątrz obramowania), a nie krawędzi obszaru zawartości. Jest to normalne zachowanie, gdy elementy blokowe są używane jako zawierające bloki.



Tym razem do pozycjonowanego elementu `em` dodano właściwości `width` i `margin` :

```
<!--positioning_context_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Positioning context example</title>
  <style>
    p {
      position: relative;
      padding: 15px;
      background-color: #F2F5D5;
      border: 2px solid purple;
    }
    em {
      width: 200px;
      margin: 25px;
      position: absolute;
      top: 2em;
      left: 3em;
      background-color: fuchsia;
    }
  </style>
</head>
<body>
<p>  Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>

</body>
</html>
```



Tutaj widzimy, że:

- Wartości przesunięcia odnoszą się do zewnętrznych krawędzi pudełka elementu (zewnętrznej krawędzi marginesu) dla elementów pozycjonowanych bezwzględnie.
- Elementy pozycjonowane bezwzględnie zawsze zachowują się jak elementy na poziomie bloku. Na przykład zachowane są marginesy ze wszystkich stron, mimo że jest to element liniowy. Pozwala również ustawić szerokość elementu.

Ważne jest, aby pamiętać, że po umieszczeniu elementu staje się on nowym blokiem zawierającym dla wszystkich zawartych w nim elementów. Załóżmy, że umieszczamy wąski element `div` w lewym górnym rogu strony, tworząc kolumnę. Jeśli mielibyśmy bezwzględnie ustawić obraz w tym `div` z wartościami przesunięcia, które umieszczają go w prawym górnym rogu, pojawi się on w prawym górnym rogu tego `div`, a nie na całej stronie. Po umieszczeniu elementu nadrzędnego działa on jako blok zawierający `img` i wszelkie inne zawarte elementy.



## Step 12

Teraz, gdy masz już lepsze wyczucie koncepcji bloku zawierającego, poświęćmy trochę czasu na lepsze zapoznanie się z właściwościami przesunięcia. Do tej pory widzieliśmy tylko element przesunięty o kilka `em` w dół i w prawo, ale to oczywiście nie wszystko, co możemy zrobić.

Jak wspomniano wcześniej, dodatnie wartości przesunięcia odpychają pozycjonowane pole elementu od określonej krawędzi w kierunku środka bloku zawierającego. Jeśli strona nie ma podanej wartości, jest ustawiana na `auto`, a przeglądarka dodaje wystarczająco dużo miejsca, aby układ działał. W tym przykładzie `div#B` jest zawarty w `div#A`, któremu nadano wymiary `600` pikseli szerokości i `300` pikseli wysokości. Użyto długości pikseli dla wszystkich czterech właściwości przesunięcia, aby umieścić pozycjonowany element (`#B`) w określonym miejscu w elemencie zawierającym (`#A`).

```
<!--pixel_measurements_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Pixel measurements example</title>
<style>
  div#A {
    position: relative;
    width: 600px;
    height: 300px;
    background-color: #C6DE89;
  }
  div#B {
    position: absolute;
    top: 25px;
    right: 50px;
    bottom: 75px;
    left: 100px;
    background-color: steelblue;
  }
</style>
</head>
<body>
  <div id="A">
    <div id="B">&nbsp;  </div>
  </div>
</body>
</html>
```



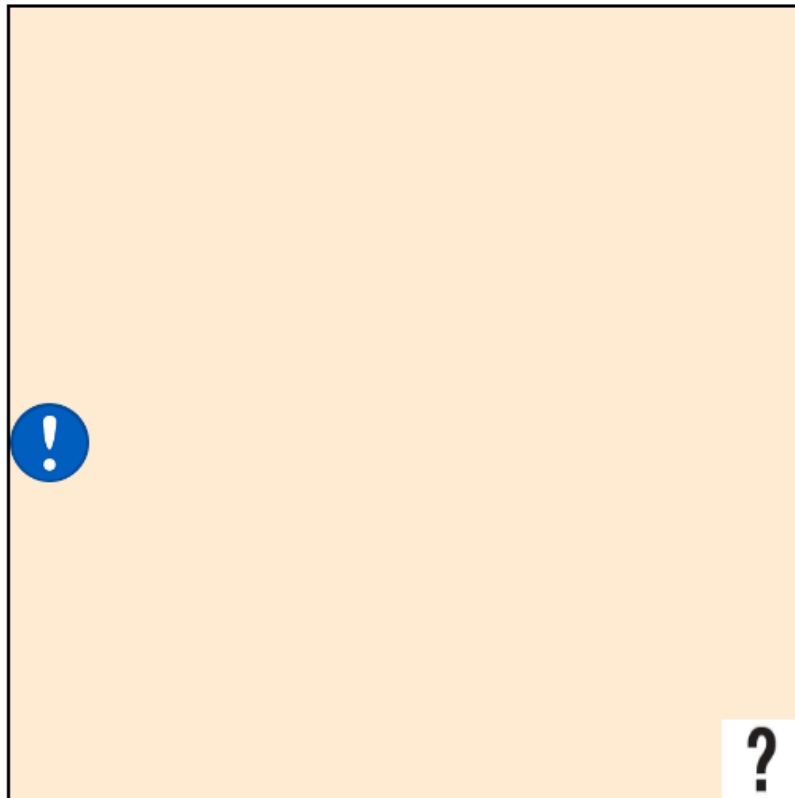
Zauważmy, że ustawiając przesunięcia ze wszystkich czterech stron, pośrednio ustaliłem wymiary pozycjonowanego `div#B`. Wypełnia przestrzeń `450 × 200` pikseli, która pozostała w bloku zawierającym po zastosowaniu wartości przesunięcia. Gdyby określono również szerokość i inne właściwości pola dla elementu `div#B`, istnieje możliwość wystąpienia konfliktów, jeśli suma wartości

ustawionego pola i jego przesunięcie nie pasuje do dostępnego miejsca w bloku zawierającym.

Specyfikacja `CSS` zapewnia zniechęcający zestaw reguł postępowania w przypadku konfliktów, ale w rezultacie należy po prostu uważać, aby nie przesadzić z właściwościami ramek i przesunięciami. Ogólnie rzecz biorąc, szerokość (uwzględniając marginesy, a także dopełnienie i obramowanie, jeśli używamy modelu rozmiaru pudełka z zawartością) i jedna lub dwie właściwości przesunięcia to wszystko, co jest potrzebne do uzyskania układu, którego szukamy. Pozwólmy przeglądarce zająć się pozostałymi obliczeniami.

Możemy także określić pozycje z wartościami procentowymi. W pierwszym przykładzie obraz jest umieszczony w połowie (50%) lewej krawędzi bloku zawierającego. W drugim przykładzie po prawej element `img` jest umieszczony tak, że zawsze pojawia się w prawym dolnym rogu bloku zawierającego.

```
<!--percentage_position_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Percentage position example</title>
<style>
  img {
    width: 50px;
    height: 50px;
  }
  img#A {
    position: absolute;
    top: 50%;
    left: 0%;
  }
  img#B {
    position: absolute;
    bottom: 0%;
    right: 0%;
  }
  div {
    position: relative;
    width: 500px;
    height: 500px;
    background: papayawhip;
    border: 2px solid;
  }
</style>
</head>
<body>
  <div>
    
    
  </div>
</body>
</html>
```



CI

Zanim zamkniemy rozdział o pozycjonowaniu bezwzględnym, jest jeszcze jedna powiązana koncepcja, którą należy przedstawić. Jak widzieliśmy, elementy pozycjonowane bezwzględnie nakładają się na inne elementy, więc wynika z tego, że wiele elementów pozycjonowanych może się nakładać na siebie.

Domyślnie elementy układają się w kolejności, w jakiej pojawiają się w dokumencie, ale kolejność układania można zmienić za pomocą właściwości `z-index`.

`z-index`

Wartości: `number` | `auto`

Domyślna wartość: `auto`

Dotyczy: elementów z ustawioną właściwością `position`

Dziedziczy: nie

CI

Wartością właściwości `z-index` jest liczba (dodatnia lub ujemna). Im wyższa liczba, tym wyżej element pojawi się w stosie. Niższe liczby i wartości ujemne przesuwają element w dół stosu.

```

<!--z_index_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Z-Index example</title>
<style>
  img {
    width: 150px;
    height: 150px;
    border: 2px solid;
  }
  #A {
    z-index: 100;
    position: absolute;
    top: 175px;
    left: 200px;
  }
  #B {
    z-index: 5;
    position: absolute;
    top: 275px;
    left: 100px;
  }
  #C {
    z-index: 1;
    position: absolute;
    top: 325px;
    left: 250px;
  }
</style>
</head>
<body>
  <p id="A"></p>
  <p id="B"></p>
  <p id="C"></p>
</body>
</html>

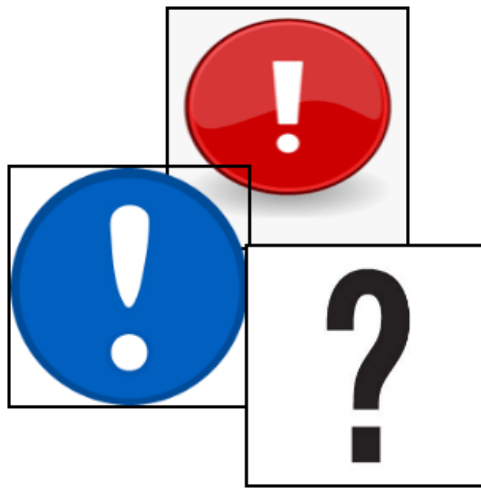
```

❏



❏

Domyślnie bez użycia `z-index` strona wyglądałaby następująco:



CI

Szczerze mówiąc, właściwość `z-index` nie jest często wymagana w przypadku większości układów stron, ale powinniśmy wiedzieć, że jest dostępna, jeśli jej potrzebujemy. Jeśli chcemy mieć pewność, że pozycjonowany element zawsze znajdzie się na górze, przypiszmy mu bardzo wysoką wartość `z-index`, np. `100` lub `1000`. Jeśli chcemy mieć pewność, że znajdzie się na dole, nadajmy mu wartość ujemną. Sama liczba tak naprawdę nie ma znaczenia.

## Step 13

---

Omówiliśmy pozycjonowanie względne i bezwzględne, więc teraz nadszedł czas, aby zająć się pozycjonowaniem stałym. W większości pozycjonowanie stałe działa tak samo jak pozycjonowanie bezwzględne. Znacząca różnica polega na tym, że wartości przesunięcia dla stałych elementów są zawsze względne względem widocznego obszaru, co oznacza, że pozycjonowany element pozostaje na swoim miejscu, nawet gdy reszta strony jest przewijana. Stałe elementy są często używane w menu, które pozostają w tym samym miejscu u góry, u dołu lub z boku ekranu, dzięki czemu są zawsze dostępne, nawet podczas przewijania zawartości. Pamiętajmy, że jeśli przymocujesz element na dole widocznego obszaru, będziemy musieli zostawić wystarczająco dużo miejsca na końcu dokumentu, aby zawartość nie była ukryta za stałym elementem. Stałe elementy są również problematyczne podczas drukowania dokumentu, ponieważ wydrukują się na każdej stronie, nie rezerwując dla siebie miejsca. Podczas drukowania dokumentu najlepiej wyłączyć stałe elementy.

```
<!--fixed_positioning_example.html-->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Fixed positioning example</title>
<style>
  img {
    width: 100px;
    height: 100px;
  }
  img.fixed {
    position: fixed;
    bottom: 0;
    right: 0;
    border: 3px solid #73AD21;
  }
</style>
</head>
<body>
<p> Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>
<p> Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>
<p> Lorem ipsum dolor sit amet, <em>consectetur adipiscing elit</em>, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Elit pellentesque habitant morbi tristique senectus et netus et malesuada. Suspendisse interdum consectetur libero id faucibus nisl tincidunt. Mi eget mauris pharetra et ultrices neque ornare aenean euismod. Magna etiam tempor orci eu lobortis elementum nibh. Faucibus et molestie ac feugiat sed lectus vestibulum. Sed risus pretium quam vulputate dignissim suspendisse in. Fringilla urna porttitor rhoncus dolor purus non enim praesent elementum. Quis auctor elit sed vulputate mi sit amet. Blandit cursus risus at ultrices mi tempus. Id cursus metus aliquam eleifend mi in nulla posuere. Amet purus gravida quis blandit turpis. Commodo ullamcorper a lacus vestibulum. Vestibulum rhoncus est pellentesque elit ullamcorper dignissim. Commodo quis imperdiet massa tincidunt nunc pulvinar sapien et. Volutpat est velit egestas dui id ornare arcu odio.</p>

</body>
</html>
```

