Zaawansowany HTML i CSS
Dzień 2

v3.3

Plan

- Box sizing
- Pseudoklasy i pseudoelementy
- Pozycjonowanie elementów
- Flexbox

Box sizing

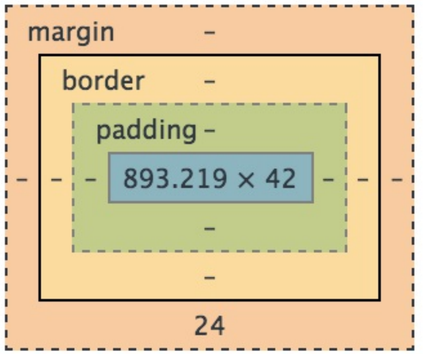
box-sizing

Ważne

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

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

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



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

border-box

border-box

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

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

Szerokość i wysokość zawartości możemy wyliczyć następująco:

`width – padding – border, height – padding – border`.

Czas na zadania

Pseudoklasy i pseudoelementy

Pseudoklasy i pseudoelementy

Dodatkowe elementy

Chcesz ostylować kliknięty link czy np. zaznaczony przycisk wyboru?

Chcesz nadać kolor np. ostatniemu wierszowi w liście?

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

Selektor i **pseudoklasę** oddzielamy dwukropkiem.

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

Dynamiczne pseudoklasy – hover

Efekt po najechaniu myszką

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

:hover

Dzięki tej pseudoklasie znajdziesz element, nad którym znajduje się kursor myszy.
Jest to bardzo przydatne m.in. podczas tworzenia animacji na stronie.

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

Dynamiczne pseudoklasy – active

Efekt po naciśnięciu

:active

Dzięki tej pseudoklasie wybierzesz element, który jest właśnie naciskany.
Używane jest to np. do stworzenia efektu wciskanego guzika (przez zmianę jego cienia).

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

Pseudoklasy związane z linkami

Efekt po odwiedzeniu linka

:visited

Jeśli już weszliśmy na dany link, domyślnie podświetla się on na fioletowo.

Za pomocą pseudoklasy visited możemy zmienić styl wyświetlania odnośnika do strony, na której już byliśmy.

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

Pseudoklasy związane z linkami

Efekt wyróżniający

:focus

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

Ta pseudoklasa jest często używana do wyróżnienia wybranego elementu formularza.

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

LoVe and HAte

Zasada kolejności

Przy stosowaniu pseudoklas dla linków należy trzymać się określonego porządku.

Prawidłowa kolejność użycia pseudoklas jest podana po prawej.

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

Pseudoklasy dla formularzy

:enabled

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

:disabled

Dzięki tej pseudoklasie znajdziemy element, który jest wyłączony.

:checked

Dzięki tej pseudoklasie znajdziemy checkboksy, które są wybrane.

```
input:enabled { }
input:disabled { }
input:checked { }
```

Pseudoklasy związane z położeniem

:first-child

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

:last-child

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

```html
<body>
    <p>Paragraf 1</p>
    <p>Paragraf 2</p>
    <p>Paragraf 3</p>
</body>
```

```css
p:first-child {
  font-size: 1.5em;
}
p:last-child {
  font-size: 1.5em;
}
```

Pseudoklasy związane z położeniem

:nth-child()

Pseudoklasa, która wybiera n-ty element potomny swojego rodzica.

Przykład 1 - ustawi czcionkę na **1.5em** elementom **p**, które są nieparzystymi dziećmi elementu **body**.

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

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

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

```
p:nth-child(2n+1) {
  font-size: 1.5em;
  /* 1., 3., 5. … dziecko */
}
p:nth-child(4n+4) {
  font-size: 1.5em;
  /* 4., 8., 12. … dziecko */
}
```

:nth-child

n	2n+1	4n+1	4n+4	4n	5n+2
0	1	1	4	0	2
1	3	5	8	4	7
2	5	9	12	8	12
3	7	13	16	12	17
4	9	17	20	16	22

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

Pseudoklasy związane z położeniem

:nth-last-child()

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

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

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

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

Pseudoklasy – element danego typu

:first-of-type

Pseudoklasa, która wybiera pierwszy element danego typu.

:last-of-type

Pseudoklasa, która wybiera ostatni element danego typu.

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

Pseudoklasy – element danego typu

:nth-of-type

Pseudoklasa, która wybiera **`n-ty`** element danego typu.

:nth-last-of-type

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

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

Pseudoelementy

Czym są pseudoelementy?

Poza pseudoklasami w CSS mamy też pseudoelementy.

Są one używane do stylowania części danego elementu albo treści wokół elementu. Możemy też stworzyć element HTML przy pomocy CSS. Pseudoelementy są nieklikalne.

:before

Pseudoelement używany do dodania danej treści jako pierwsze dziecko danego elementu.

:after

Pseudoelement używany do dodania danej treści jako ostatnie dziecko danego elementu.

```
p:before {
}

ul:after {
}
```

Atrybut content

Ważne!

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

Wynik na stronie:

- 1
- 2

  Koniec listy!

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

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

Pseudoelementy związane z tekstem

Pseudoelementy związane z tekstem

:first-letter

Dzięki temu pseudoelementowi wystylujemy pierwszą literę tekstu.

:first-line

Dzięki temu pseudoelementowi wystylujemy pierwszą linię tekstu.

```
p:first-letter {
  font-size: 2.5em;
}
p:first-line {
  text-decoration: underline;
}
```

Zaprzeczenie w CSS

Jeżeli nie chcemy czegoś wybrać?

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

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

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

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

Spis selektorów, które warto pamiętać.

Czas na zadania

Pozycjonowanie elementów

Opływanie elementów – przypomnienie

Ustawiamy elementy obok siebie

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

Możemy ustawiać je z lewej strony lub z prawej.

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

Czyszczenie floatów

Musimy natomiast pamiętać, że aby nasz układ strony działał prawidłowo, musimy "wyczyścić" działanie floatów. To bardzo ważne!.

```
.section-after {
  clear: both;
  /* reszta styli*/
}
```

Clearfix

Aby nie dodawać niepotrzebnego elementu do HTML możemy skorzystać z pseudoelementu.

Dodajemy dodatkową klasę do rodzica elementów, które korzystają z float.

Kod HTML

```
<div class="main clearfix">
  <div class="box-left"></div>
  <div class="box-right"></div>
</div>
```

Kod CSS

```
.box-left {
  float: left;
}
.box-right {
  float: right;
}
.clearfix:after{
  content:'';
  display:block;
  clear:both;
}
```

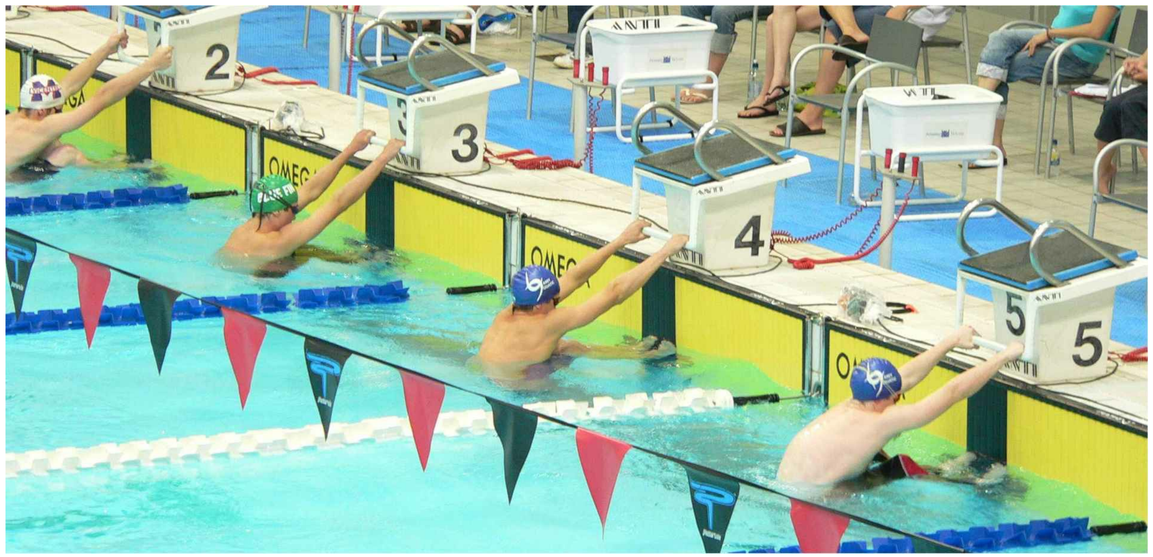
position

Pozycja

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

Inne wartości position to:

- **`relative,`**
- **`absolute,`**
- **`fixed.`**



position: relative

Pozycja i przesuwanie

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

- **`top`**
- **`right`**
- **`bottom`**
- **`left`**

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

position: absolute

Pozycja i przesuwanie

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

- **`top`**
- **`right`**
- **`bottom`**
- **`left`**

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

position: fixed

Pozycja i przesuwanie

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

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

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

position: absolute a punkt zaczepienia

Pozycja i przesuwanie

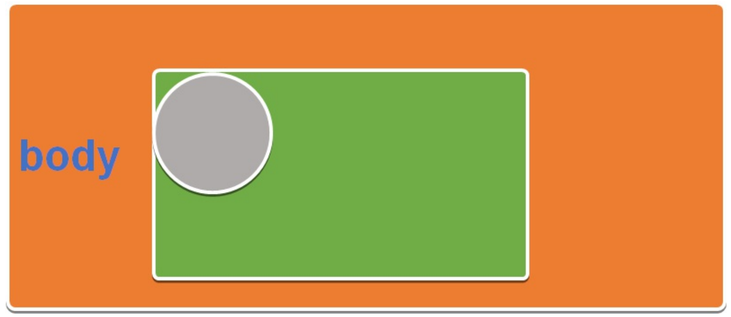
Element, któremu nadamy inną właściwość niż **`position:static`** staje się kontenerem zawierającym. Dzięki temu elementy w nim zawarte, którym nadaliśmy **`position absolute`**, będą ustawiane względem jego krawędzi, a nie krawędzi elementów nadrzędnych (np. body).
Najczęściej używa się **`position: absolute`** wraz z **`position: relative`** w elemencie nadrzędnym.

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

position: absolute a punkt zaczepienia

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

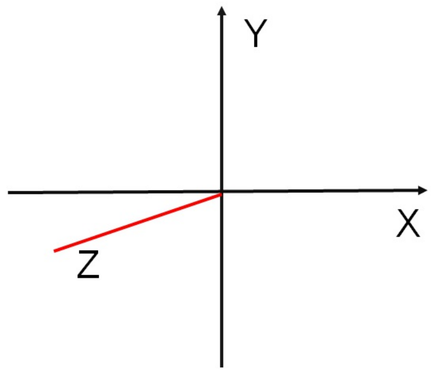
```css
.box-green {
  position: relative;
  bottom: 10px;
  left: 80px;
}
.circle-grey {
  position: absolute;
  top: 0;
  left: 0;
}
```



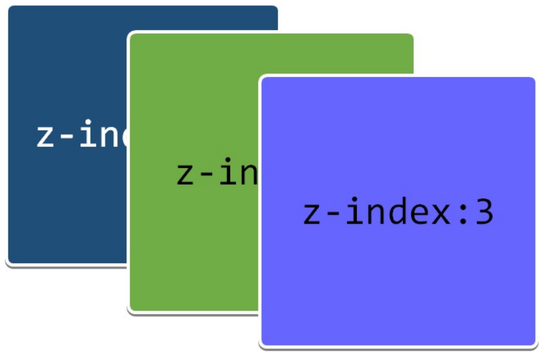
z-index

Trzeci wymiar

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



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



- https://www.jsfiddle.net/CodersLab/c7f75511

Czas na zadania

Flexbox

Wprowadzenie

Wraz z rozwojem internetu wzrósł również poziom skomplikowania układów stron internetowych.

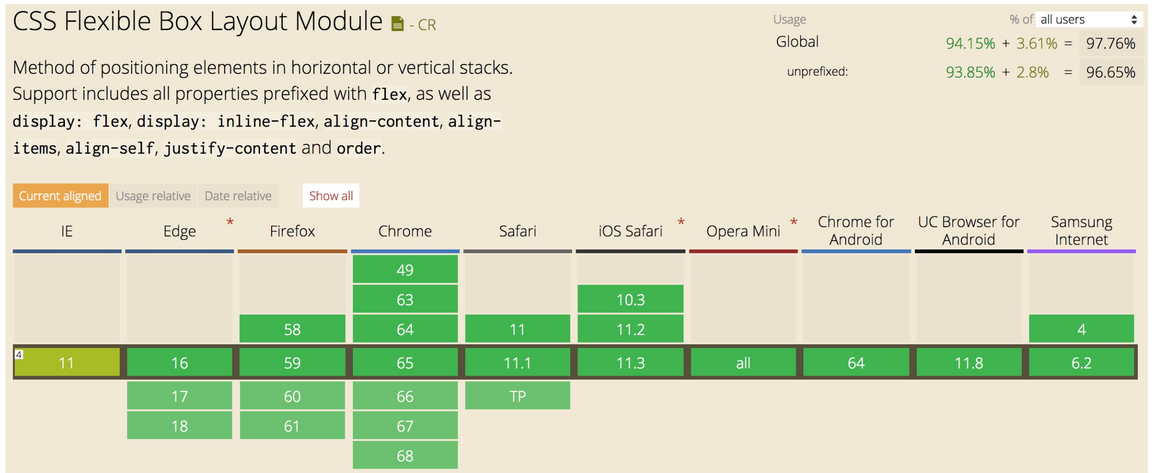
Organizacja W3C, której zadaniem jest dbanie o standardy języków HTML i CSS chciała sprostać oczekiwaniom nowoczesnych layoutów stron internetowych i aplikacji, więc stworzyła nowy sposób układania elementów na stronie - **flexbox**.

Głównym założeniem flexboxa jest jego **elastyczność**. Inne techniki tworzenia układów (float lub elementy inline-block) mają problem z dopasowaniem zawartości do kontenerów.

Wsparcie w przeglądarkach

Początkowo specyfikacja flexboxa była często zmieniania. Niektóre przeglądarki implementowały flexboxa jako eksperyment przy użyciu dodatkowych prefixów. Po pewnym czasie, specyfikacja była już doprecyzowana, dzięki czemu doczekała się ogólnego wsparcia.

Obecnie możemy używać flexboxa we wszystkich popularnych przeglądarkach (od IE10+). Niektóre wersje przeglądarek do poprawnego działania flexboxa potrzebują natomiast prefixów. Dobrym rozwiązaniem jest używanie np. autoprefixera - https://autoprefixer.github.io/



Jak działa flexbox

Zapis

Aby korzystać z możliwości flexboxa potrzebujemy zaaplikować do naszego kodu CSS następującą linijkę:

```
.container {
  display: flex;
}
```

Powoduje ona, że wewnątrz tego kontenera ma zostać użyty flexbox. Użycie jej spowoduje, że dzieci tego elementu zmienią swój **styl wyświetlania**. Domyślnie dzieci kontenera flexowego ustawiają się obok siebie, zajmując całą możliwą przestrzeń.

Uwaga

Zmieniony styl wyświetlania obowiązuje tylko względem **bezpośrednich dzieci**. Oznacza to, że jeśli kontener ustawiony na flexbox ma wnuki (czyli bezpośrednie elementy HTML mają w sobie inne elementy), to one nie zmieniają swojego domyślnego stylu wyświetlania (block lub inline).

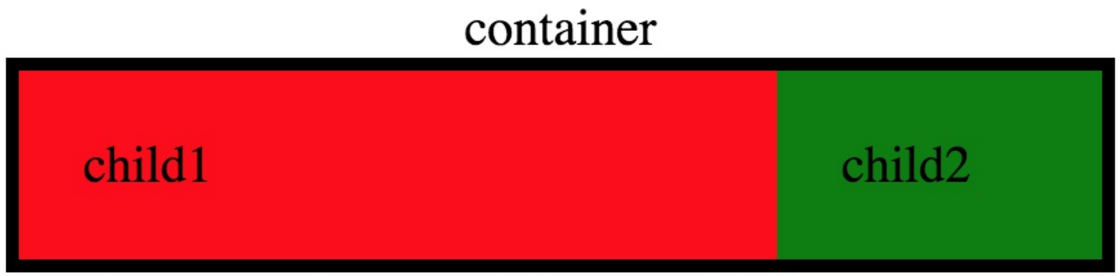
Pamiętaj o tym!

Ustawianie wymiarów

Zapis

Z racji tego, że dzieci zmieniają swój styl wyświetlania, również zmienia się ich szerokość. Żeby elementy wyglądały tak jak oczekujemy, musimy pamiętać, żeby ustawić dla nich wymiary np. width.

```
.container {
  display: flex;
}
.child1 {
  width: 70%;
}
.child2 {
  width: 30%;
}
```



Układanie elementów na stronie

Elementy ustawione na flexbox posiadają bardzo dużo możliwości zmiany swojego położenia.

Wyróżniamy dwa miejsca, gdzie aplikujemy style:

- w kontenerze głównym (ustawionym na flex)
- bezpośrednio do elementów, na które wpływa flexbox

Najważniejsze funkcje flexboxa stosuje się na kontenerach głównych (czyli na tych elementach, które mają wpisane display: flex). Poznamy teraz kilka głównych reguł CSS do układania elementów na stronie tą metodą.

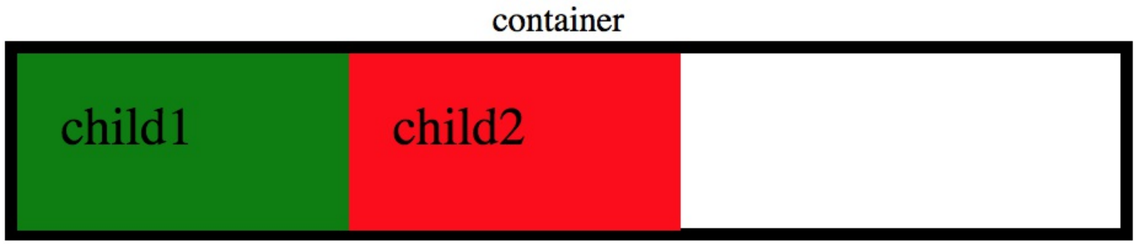
Spis większości właściwości flexboxa można znaleźć pod linkiem: Guide to flexbox

justify-content

Z racji tego, że ta technika tworzenia układów stron jest bardzo elastyczna, wyróżniamy możliwość przesuwania elementów względem osi X i osi Y (początkowo X jest poziomo, a Y pionowo).

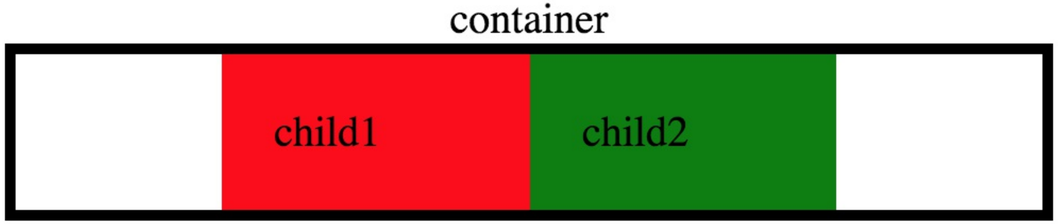
Żeby przesuwać elementy wewnątrz kontenera względem osi X, używamy właściwości justify-content.

```
.container {
  display: flex;
  justify-content: flex-start;
  /* wartosc domyslna */
}
```

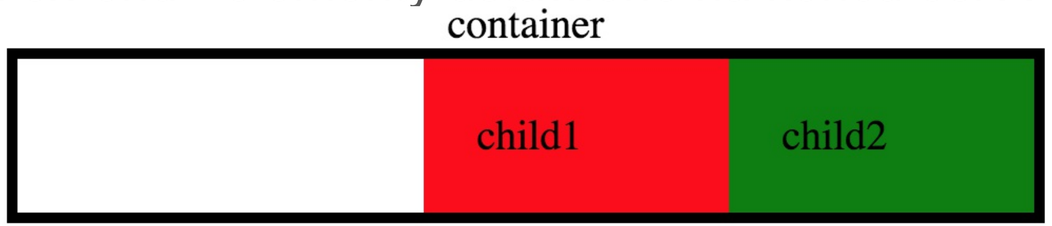


justify-content - popularne opcje

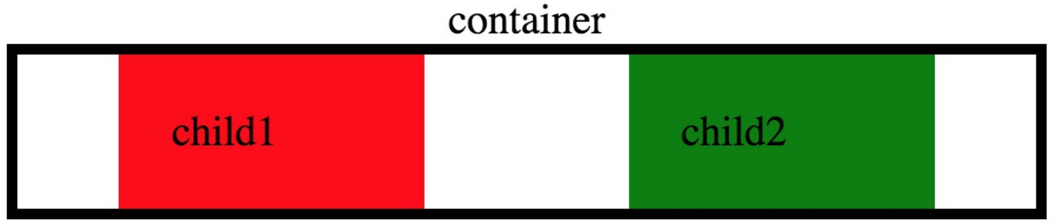
- flex-start - początkowa, elementy ustawione do początku osi X
- center - elementy ustawione na środku osi X



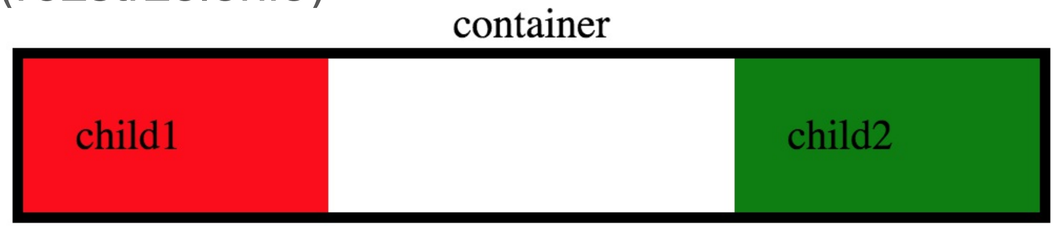
- flex-end - elementy ustawione na końcu osi X



- space-around - elementy rozsuwają się na boki, zachowując równy odstęp między sobą



- space-between - elementy rozsuwają się na boki, zachowując jak maksymalny odstęp (rozstrzelenie)

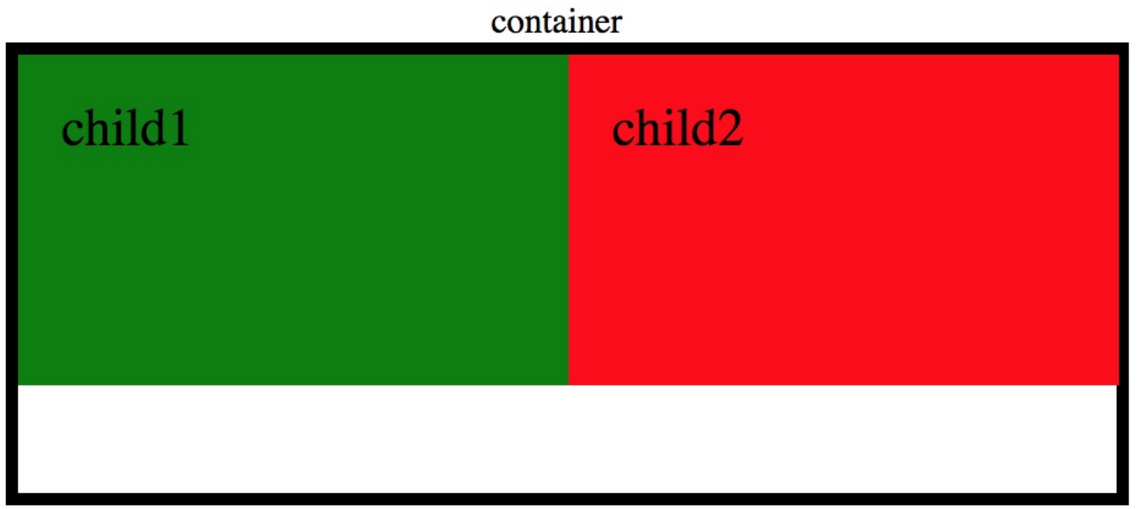


align-items

Druga właściwość służy natomiast do ustawiania elementów względem osi Y (początkowo w pionie).

Musimy tutaj ustawić wysokość kontenera, aby był większy niż elementy wewnątrz. Dzięki temu widać jak elementy są wyrównane.

```
.container {
  height: 400px;
  display: flex;
  align-items: flex-start;
  /* wartość domyślna */
}
```



align-items - popularne opcje

- flex-start - początkowa wartość, elementy są ustawione na początku osi Y
- center - elementy ustawione na środku osi Y

- flex-end - elementy ustawione na końcu osi Y

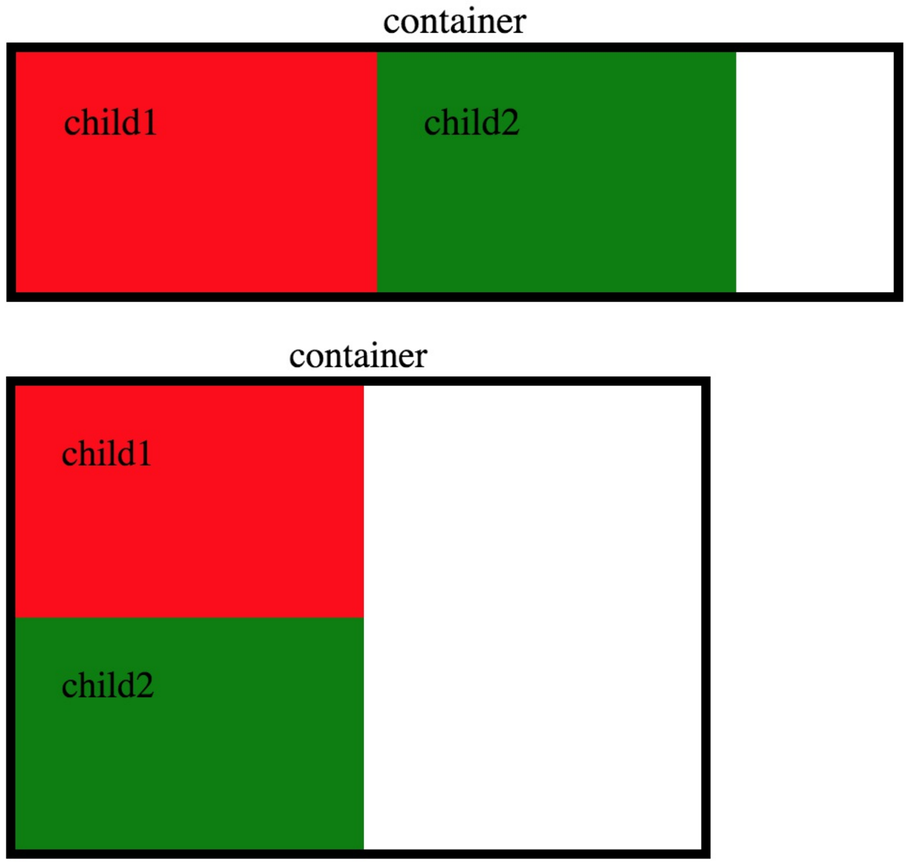
flex-wrap

W pewnych momentach, możemy chcieć powiedzieć naszemu kontenerowi (ustawionemu na flex), że, w przypadku gdy dwa elementy na stronie nie mieszczą się obok siebie (w jednej linii), to ma je ułożyć jeden pod drugim (ma to zastosowanie np.: przy zmniejszaniu okna przeglądarki).

Kod ten spowoduje, że elementy wewnątrz kontenera będą spadać pod siebie, jeśli nie zmieszczą się obok siebie na stronie.

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

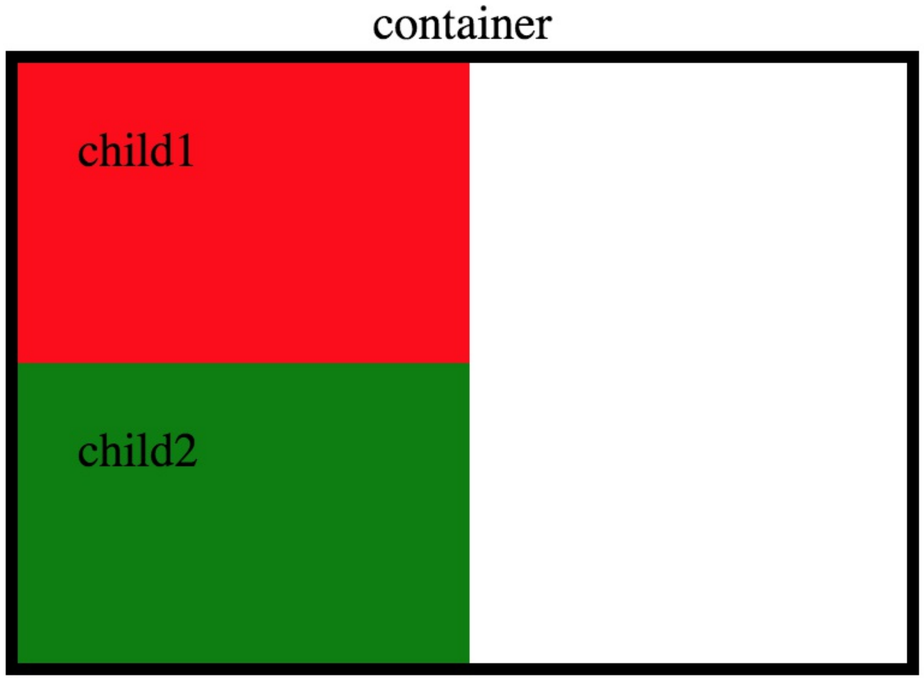
Na obrazku obok widać, że drugi kontener jest



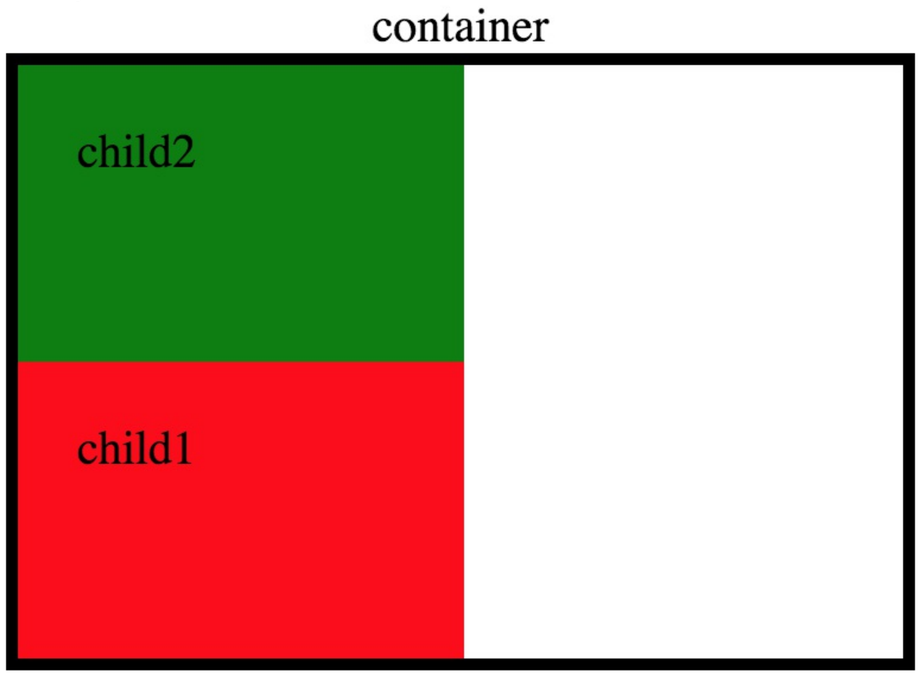
flex-wrap - popularne opcje

- nowrap - domyślna wartość, nie zawija elementów

- wrap - zawijanie elementów jeden pod drugim



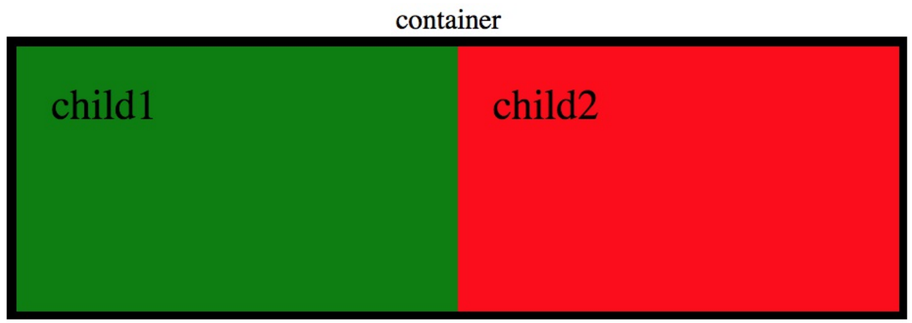
- wrap-reverse - zawijanie elementów w taki sposób, że jeśli elementy nie zmieszczą się obok siebie, zamieniają się kolejnością (element, który był po lewej stronie będzie na dole)



flex-direction

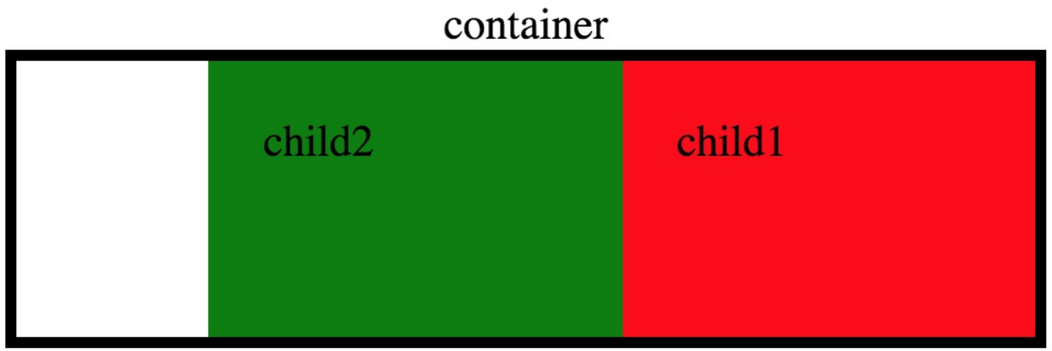
Reguła flex-direction zmienia kolejność występowania elementów na stronie (zmienia położenie osi X oraz Y).

```
.container {
  display: flex;
  flex-direction: row;
}
```

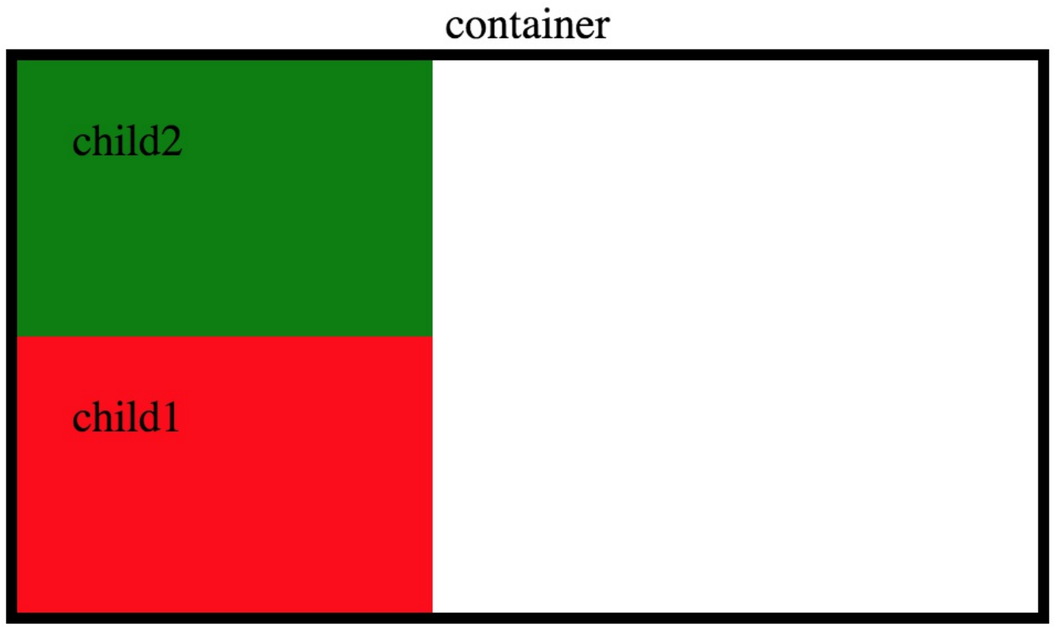


flex-direction - popularne opcje

- row - domyślna wartość, ustawia elementy od lewej do prawej
- row-reverse - ustawia elementy od prawej do lewej



- column - ustawia elementy od góry do dołu (tak jak elementy blokowe)
- column-reverse - ustawia elementy od dołu do góry



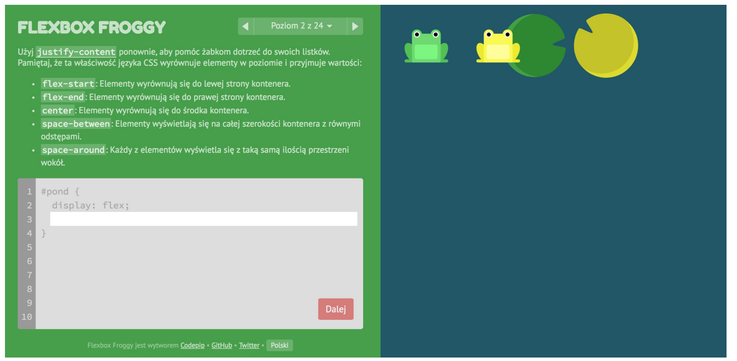
Flexbox Podsumowanie

Flexbox jest bardzo potężną techniką, dzięki której jesteśmy w stanie ułożyć najtrudniejsze układy stron internetowych i aplikacji. Można ją stosować do układania nawet najdrobniejszych elementów np. notyfikacji czy ikon wiadomości.

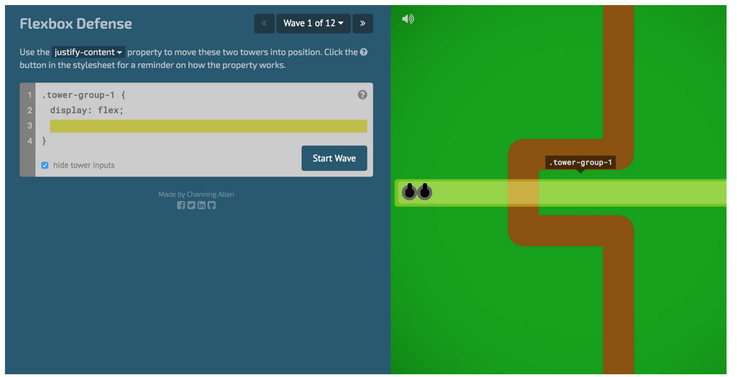
W internecie znajdują się dwie ciekawe gry, które uczą flexboxa. Zagraj w nie, aby zobaczyć w praktyce możliwości flexboxa.

Jeśli chcesz przećwiczyć flexboxa bardzo dokładnie, istnieje ciekawa gra przeglądarkowa Flexbox zombies. Ze względu na długość gry, rekomendujemy grać w nią po zajęciach lub w czasie wolnym :)

Flexbox Froggy



Flexbox Defense



Czas na zadania