

Kurs Front-End Developer
CSS3

SELEKTORY PODSTAWOWE (I-***)

Selektor - służy do wyszukania elementu bądź grupy elementów w dokumencie HTML
Wyszukanym elementom można nadać właściwości CSS (postylować je :)

Poniższe selektory zostały omówione w materiałach przygotowawczych do kursu:

- selektor znacznika,
- selektor identyfikatora i klasy
- selektor potomka i potomka bezpośredniego
- selektor rodzeństwa i rodzeństwa bezpośredniego

Powtórka:

- `p` - selektor znacznika
- `#identyfikator` - selektor identyfikatora
- `.klasa` - selektor klasy
- `li > a` - selektor bezpośredniego potomka
- `li a` - selektor potomka
- `p + a` - selektor bezpośredniego rodzeństwa
- `p ~ a` - selektor rodzeństwa

SELEKTORY PODSTAWOWE (I-***)

Selektor uniwersalny

Tworzymy go za pomocą gwiazdki. Selektor ten odwołuje się do każdego typu elementu w dokumencie HTML.

```
*{  
  color:#313131;  
}
```

wszystkie elementy będą miały
kolor czcionki #313131

Selektor łączony

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

selektor `div.background`
odszuka wszystkie elementy `div` z
klasą `background` i zmieni kolor
tła na czerwony

SELEKTORY PSEUDO-ELEMENTÓW (2-***)

Selektor pseudo-elementów

Pseudo-elementy to elementy, które nie istnieją w drzewie dokumentu.

Selektor pseudo-elementu składa się z dwóch dwukropków (: :), za którymi znajduje się jego nazwa.

`::before` – służy do dodania dodatkowej treści na początku zawartości HTML

```
div::before {  
  content: "*";  
  color: #444;  
}
```

Na początku zawartości
każdego elementu `div` dodaj *
w kolorze #444.

`::after` – służy do dodania dodatkowej treści na końcu zawartości HTML

```
div::after {  
  content: "*";  
  color: #444;  
}
```

Na końcu zawartości każdego
elementu `div` dodaj * w kolorze
#444.

SELEKTORY PSEUDO-KLAS (3-***)

Pseudo-klasy dotyczące działań użytkownika:

:hover – określa wygląd elementu, na który najjeżdżamy kursorem myszy.

WAŻNE! Urządzenia mobilne mogą różnie interpretować ten selektor

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

W momencie najechania kursorem
myszy na element `div` kolor tekstu
stanie się czerwony

:focus – określa aktualnie aktywny element - np. aktualnie uzupełniane pole w formularzu

```
input:focus {  
    color: red;  
}
```

Aktualnie edytowane pole
formularza będzie mieć kolor
tekstu ustawiony na czerwony

SELEKTORY PSEUDO-KLAS (3-***)

Selektor pseudo-klas

Pseudo-klasy wykorzystywane są do nadawania specjalnych właściwości elementom języka HTML. Właściwości te są dodawane do elementów po wykonaniu określonej akcji **użytkownika** lub **gdy** element ma określone położenie w strukturze dokumentu. Selektor pseudo-klasy składa się z dwukropka (:) i nazwy.

Pseudo-klasy potomstwa:

:first-child – reprezentuje element będący pierwszym zagnieżdżonym elementem

```
ol li:first-child {  
    color: red;  
}
```

Pierwszy element li listy ol
będzie miał kolor tekstu
ustawiony na czerwony

:last-child – reprezentuje element będący ostatnim zagnieżdżonym elementem

```
ol li:last-child {  
    color: red;  
}
```

Ostatni element li listy ol
będzie miał kolor tekstu
ustawiony na czerwony

SELEKTORY PSEUDO-KLAS (3-***)

`:nth-of-type(pattern)` – wyszukuje elementy kolejne określonego typu wg. wzoru `pattern`

`:nth-child(pattern)` – rozpoczyna wyszukiwanie elementów potomnych od pierwszego do ostatniego wg. wzoru `pattern`

`:nth-last-child(pattern)` – rozpoczyna wyszukiwanie elementów potomnych od ostatniego do pierwszego wg. wzoru `pattern`

Wartości jakie możemy podać jako `pattern`:

- `even` - parzyste elementy potomne
- `odd` - nieparzyste elementy potomne
- `liczba` - np. 5, selektor wskaże piąty element potomny
- `a+b` - wartości `a` i `b` muszą być liczbami całkowitymi (dodatnimi, ujemnymi lub równe zero) - Liczba `a` oznacza co którą liczbę selektor ma wskazywać kolejne elementy potomne, a liczba `b` oznacza przesunięcie, czyli od którego elementu potomnegoa ma zacząć się odliczanie

SELEKTORY ATRYBUTU (4-***)

Selektor atrybutu - znajduje każdy element HTML, który posiada atrybut podanego przez nas typu.

[att] – znajduje wszystkie elementy HTML mające atrybut `att` o dowolnej wartości.

```
[title] {  
  color:#313131;  
}
```

Selektor znajdzie wszystkie linki, które posiadają atrybut `title` i nada im kolor tekstu #313131

[att="val"] – znajduje wszystkie elementy HTML, które posiadają atrybut `att` o wartości `val`

```
a[title="opis"] {  
  color:#313131;  
}
```

Selektor znajdzie wszystkie linki, które posiadają atrybut `title` o wartości `opis` i nada im kolor tekstu #313131

SELEKTORY ATRYBUTU (4-***)

`[att~="val"]` – znajduje każdy element HTML mający atrybut `att`, którego wartość zawiera w sobie ciąg znaków `val` (musi być on oddzielony spacją od pozostałych znaków)

```
a[title~="opis"] {  
    color:#313131;  
}
```

Selektor znajdzie wszystkie linki, które posiadają atrybut `title` i ich wartość zawiera w sobie ciąg znaków `opis`.
Następnie nada im kolor tekstu `#313131`

`[att*="val"]` - znajduje każdy element HTML mający atrybut `att`, którego wartość zawiera przynajmniej jedno wystąpienie ciągu znaków `val`.

```
a[title*="opis"] {  
    color:#313131;  
}
```

Selektor znajdzie wszystkie linki, które posiadają atrybut `title`, a ich wartość zawiera w sobie ciąg znaków `opis`.
Następnie nada im kolor tekstu `#313131`

SELEKTORY ATRYBUTU (4-***)

`[att^="val"]` – znajduje każdy element HTML mający atrybut `att`, którego wartość zaczyna się od ciągu znaków `val`.

```
a[title^="opis"] {  
    color:#313131;  
}
```

Znajduje wszystkie linki, które posiadają atrybut `title`, a ich wartość zaczyna się od ciągu znaków `opis`. Następnie nadaje im kolor tekstu `#313131`

`[att$="val"]` – znajduje każdy element HTMLR mający atrybut `att`, którego wartość kończy się ciągiem znaków `val`.

```
a[title$="opis"] {  
    color:#313131;  
}
```

Znajduje wszystkie linki, które posiadają atrybut `title`, a ich wartość kończy się ciągiem znaków `opis`. Następnie nadaje im kolor tekstu `#313131`

SELECTORY – HIERARCHIA (5-***)

Hierarchia selektorów CSS - to system zależności między selektorami określający, która reguła css (dotycząca tego samego elementu) jest ważniejsza i będzie brana pod uwagę przez przeglądarkę internetową.

Hierarchia selektorów ma wyższy priorytet niż kaskadowość!

<http://cssspecificity.com/>

Kalkulator hierarchii selektorów – czyli jak widzi to przeglądarka internetowa 😊

<https://specificity.keegan.st/>

WARSZTATY ☺ STYLOWANIE PÓŁ FORMULARZA

Stylowanie pól formularza

Pobierz kod formularza ze strony:

<http://akademia108.pl/kurs-front-end/formularz-html5.zip>

i nadaj mu stylowanie jak na obrazku poniżej:

<https://akademia108.pl/kurs-front-end/form-css.png>

Kody kolorów:

- tło: #213b50
- pole nieaktywne: #2b4a64
- pole aktywne: #4e708c
- ramka aktywnego pola i kolor tekstu: #ffffff

POZYCJA W CSS (6-****)

Pozycja - czyli właściwość `position` pozwala określać położenie elementu HTML względem innych.

Pozycja statyczna - domyślna pozycja elementów HTML na stronie. Nie pozwala na zmianę położenia elementu względem innych elementów

```
div#orange {  
    position: static;  
}
```



`position: static;`

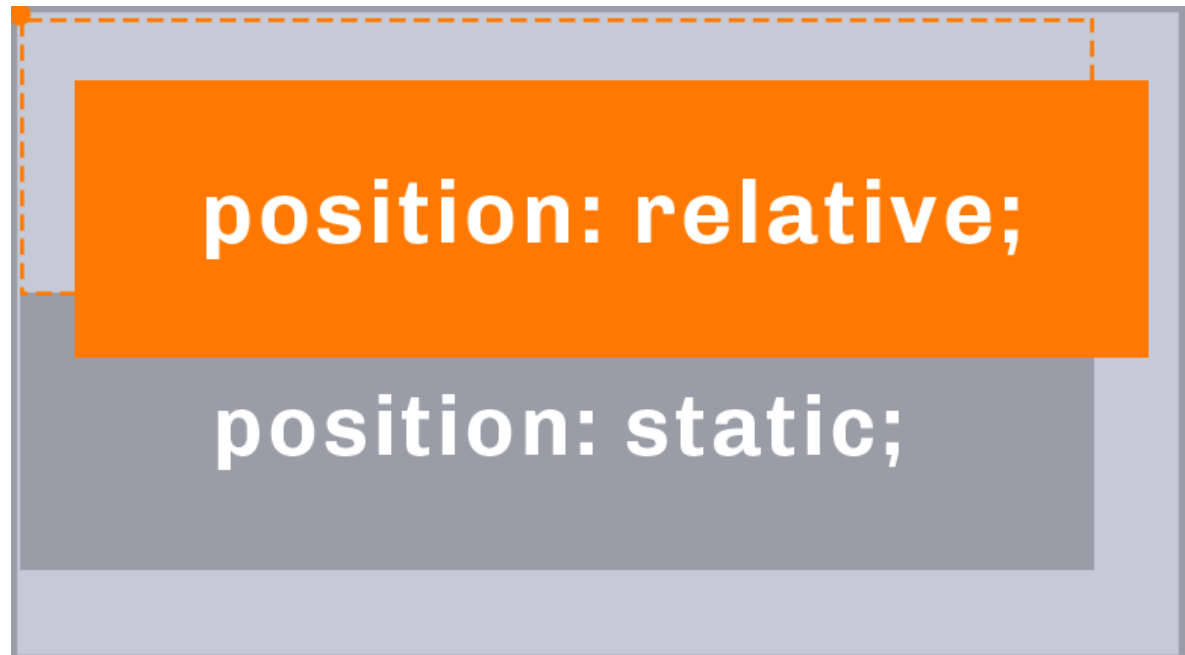
`position: static;`

POZYCJA W CSS (6-****)

Pozycja relatywna - pozwala na ustalenie pozycji elementu HTML za pomocą właściwości `top`, `bottom`, `left`, `right` względem elementu nadrzędnego.

Pozycja relatywna elementu nie zmienia położenia innych elementów.

```
div#orange {  
  position: relative;  
  top: 30px;  
  left: 20px;  
}
```



POZYCJA W CSS (6-****)

Pozycja absolutna - pozwala na ustalenie pozycji elementu HTML za pomocą właściwości `top`, `bottom`, `left`, `right` względem najbliższego elementu nadrzędnego, który znajduje się w pozycji relatywnej, ustalonej (`fixed`) lub absolutnej. Element w pozycji absolutnej zwalnia miejsce innym elementom w kontenerze.

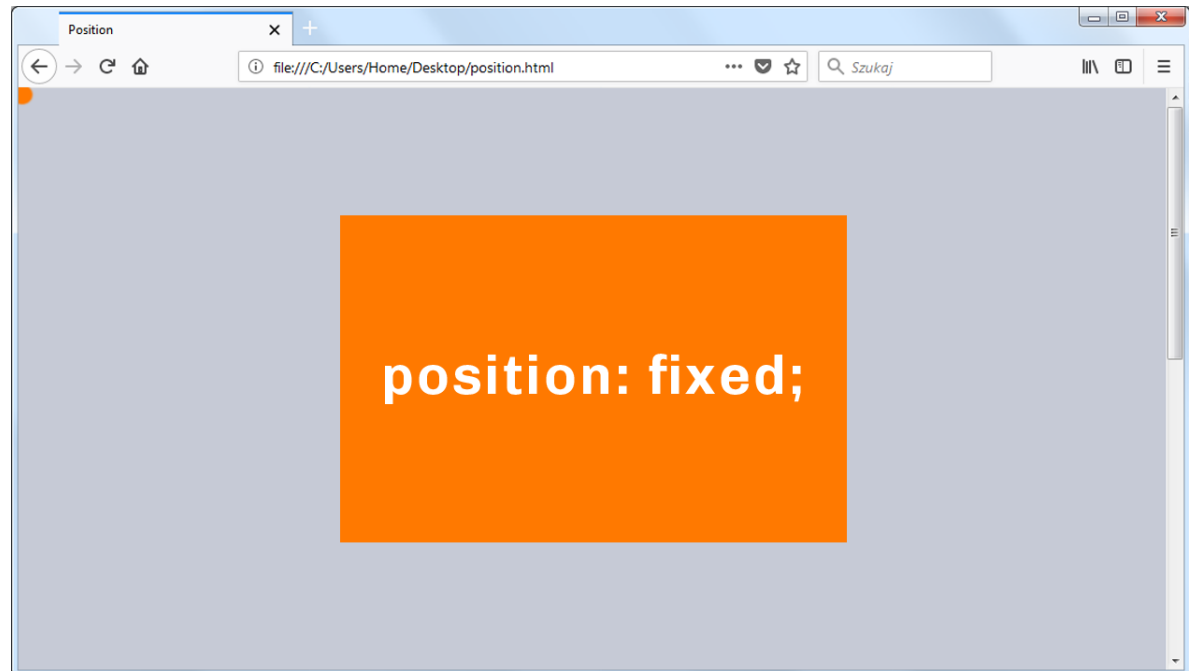
```
div#orange {  
    position: absolute;  
    top: 60px;  
    left: 30px;  
}
```



POZYCJA W CSS (6-****)

Pozycja ustalona (fixed) - pozwala na ustalenie pozycji elementu HTML za pomocą właściwości `top`, `bottom`, `left`, `right` względem okna przeglądarki. Element HTML w pozycji fixed ma stałą pozycję względem okna przeglądarki.

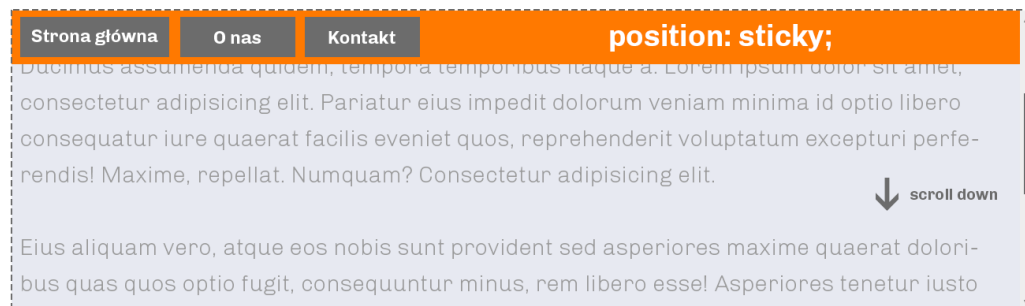
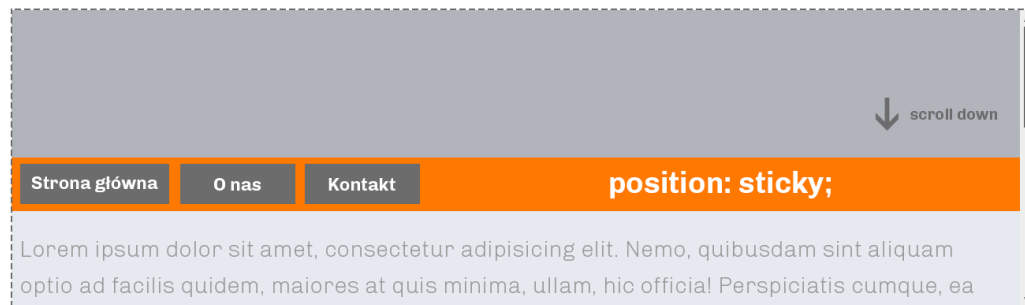
```
div#orange {  
    position: fixed;  
    top: 50px;  
    left: 150px;  
}
```



POZYCJA W CSS (6-****)

Pozycja sticky - element zachowuje się tak jakby był pozycjonowany relatywnie, aż do momentu, gdy znajdzie się w określonej odległości od krawędzi przeglądarki. Odległość tę definiujemy poprzez właściwości `top`, `bottom`, `left`, `right`. Po osiągnięciu ustawionej odległości element zaczyna być pozycjonowany w sposób ustalony, tak jakby miał ustawioną pozycję `fixed`.

```
nav#orange {  
  position: sticky;  
  top: 0;  
}
```



NARZĘDZIE – REPL.IT

Repl.it jest bardzo przydatnym narzędziem, które może wykorzystać Front-End Developer w swojej pracy:

<https://repl.it>

Jest środowiskiem online, które pozwala na stworzenie, edycję i uruchomienie w przeglądarce bloku kodu HTML, arkusza stylu CSS oraz dodanie skryptów JavaScript.

Dzięki temu możemy pokazać nasz kod innym programistom 😊

Serwis repl.it ma zastosowanie także w innych językach programowania!

WARSZTATY 😊 POZYCJONOWANIE W CSS

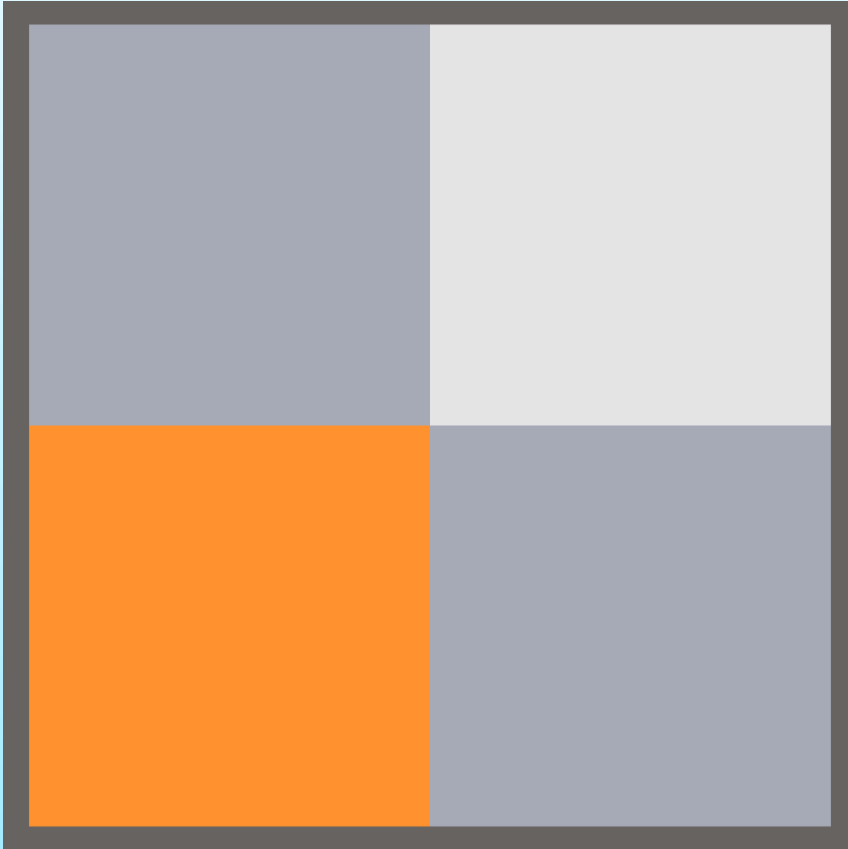
Stwórz zewnętrzny kontener. Nadaj mu wysokość i szerokość 400px oraz ustaw kolor tła jak na rysunku poniżej.

Stwórz wewnętrzne kontenery i nadaj im wysokość oraz szerokość 200px. Następnie ustaw im kolory tła jak na rysunku poniżej

Wykorzystując pozycjonowanie `absolute` spraw, żeby wewnętrzne kontenery były wypozocjonowane w sposób przedstawiony na poniższym obrazku

Kodujemy w serwisie <https://repl.it>

WARSZTATY ☺ POZYCJONOWANIE W CSS



Kolory w projekcie:

- ramka: #666360
- ciemnoszary kwadrat (tło): #a6a9b6
- pomarańczowy kwadrat: #ff912e
- jasnoszary kwadrat: #e4e4e4

JEDNOSTKI MIARY W CSS (7-***)

Jednostki miary w CSS mają zastosowanie przy określaniu wielkości elementów na stronie oraz miejsca, w którym te elementy się znajdują.

Jednostki miary dzielą się one na dwa rodzaje:

- jednostki absolutne (bezwzględne) – np. px (piksele);
- jednostki relatywne (względne) – np. em, % (procenty) oraz viewport units (najczęściej używana to vh)

Opis innych jednostek znajdziesz na stronie:

<https://developer.mozilla.org/en-US/docs/Web/CSS/length>

Piksel (px) (zbitka angielskich słów `pics` i `element`) – najmniejszy jednolity element obrazu wyświetlanego na ekranie np. monitora komputerowego, telewizora czy telefonu (przedstawiający konkretny kolor).

JEDNOSTKI MIARY W CSS (7-***)

em (em) – wartość 1em jest obliczana przez przeglądarkę internetową jako wielokrotność wartości właściwości font-size elementu rodzica.

#HTML

```
<div>  
  <p>Tekst z czcionką 3em.</p>  
</div>
```

Wielkość czcionki elementu p wynosi 3em, czyli 45px, ponieważ font-size rodzica tego elementu (czyli div) - wynosi 15px.

Obliczając: $3em = 3 \times 15px$, jest równe 45px.

#CSS

```
div { font-size: 15px; }  
div p { font-size: 3em; }
```

Procenty (%) - niektóre wartości właściwości CSS możemy określić w wartościach procentowych np. 50%.

#HTML

```
<section>  
  <div>  
    <p>Paragraf w sekcji</p>  
  </div>  
</section>
```

Szerokość elementu div wynosi 50% szerokości elementu section, czyli 250px.

Wielkość czcionki dla elementu div wynosi 40% wielkości czcionki elementu section, czyli 8px.

#CSS

```
section { width: 500px; font-size: 20px; }  
div { width: 50%; font-size: 40%; }
```

JEDNOSTKI MIARY W CSS (7-***)

Viewport height (vh) - 1vh jest równy 1% wartości wysokości przestrzeni w jakiej została wyświetlona nasza strona internetowa (przeważnie przestrzeni przeglądarki internetowej).

Na przykład jeżeli wysokość przeglądarki jest równa 768px to:

$$1\text{vh} = 1\% * 768\text{px} = 7.68\text{px}$$

$$8\text{vh} = 8\% * 768\text{px} = 61.44\text{px}$$

#CSS

```
div {  
  height: 10vh;  
}
```

Wysokość elementu `div` jest uzależniona od wartości wysokości okna przeglądarki internetowej w jakiej została wyświetlona strona internetowa.

KOLORY W CSS (8-***)

Zapis kolorów w CSS - za pomocą języka CSS mamy do dyspozycji następujące zapisy kolorów:

- kolory predefiniowane (jest ich 140)
- zapis heksadecymalny – np. #56a216
- skrócony zapis heksadecymalny – np. #234
- za pomocą funkcji rgb – np. rgb(155, 122, 108)
- za pomocą funkcji rgba – np. rgba(155, 122, 108, 0.6)

Warto poznać narzędzie poświęcone kolorom i paletom barw:

<http://color.adobe.com>

Kolory predefiniowane (podstawowe) - do określenia kolorów możemy posłużyć się ich angielską nazwą. Kolorów predefiniowanych jest 140 np:

#CSS

background-color: red; // tło w kolorze czerwonym

KOLORY W CSS (8-***)

Zapis heksadecymalny (szesnastkowy) - składa się z 6 znaków poprzedzonych hashem (#).

KOLOR CZERWONY

KOLOR ZIELONY

KOLOR NIEBIESKI

#CSS

`background-color: #00ffff; // zapis pełny`

`color: #234 // zapis skrócony`



#00FFFF

Zapis skrócony heksadecymalny (szesnastkowy) - składa się z 3 znaków poprzedzonych hashem (#).

KOLOR CZERWONY

KOLOR ZIELONY

KOLOR NIEBIESKI



#234

KOLORY W CSS (8-***)

Za pomocą funkcji `rgb(czerwony, zielony, niebieski)` - po literach `rgb`, w nawiasie znajdują się trzy liczby określające intensywność barw - kolejno: czerwonej, zielonej, niebieskiej. Liczby te oddzielone są od siebie przecinkami i każda z nich znajduje się w zakresie od 0 do 255.

```
background-color: rgb(255, 0, 0); // tło w kolorze czerwonym
```

Za pomocą funkcji `rgba(czerwony, zielony, niebieski, transparent)` - zapis jest podobny jest do zapisu w funkcji `rgb`, tylko że wymaga on dodatkowego parametru określającego stopień przezroczystości.

Do określenia stopnia przezroczystości używa się liczb dziesiętnych z zakresu 0 - 1.0, przy czym wartość 1.0 oznacza brak przezroczystości, a 0 pełną przezroczystość.

```
background-color: rgba(173, 216, 230, 0.5); // tło transparentne w połowie
```

WYŚWIELANIE ELEMENTÓW (9-***)

Elementy blokowe - to takie elementy, które domyślnie wyświetlane są jeden pod drugim i zajmują domyślnie 100% szerokości swojego rodzica. Do tych elementów należą np. `div`, `p` czy nagłówki od `h1` do `h6`.

Elementy liniowe - to takie elementy, które domyślnie wyświetlane są w jednej linii (obok siebie). Takimi przykładowymi elementami są np: `span` lub `a`. Elementy te są traktowane przez przeglądarkę jak tekst. Wyśrodkowanie elementu liniowego jest możliwe, jeżeli do elementu rodzica, który jest elementem blokowym, zostanie dodana właściwość `text-align: center;`

Wyświetlanie elementów - do zmiany sposobu wyświetlania elementów na stronie służy właściwość CSS: `display`.

Zmieniając domyślną wartość właściwości `display` elementu HTML możemy określić czy dany element HTML ma być traktowany przez przeglądarkę internetową, np. jako element liniowy `inline`, element blokowy `block`.

WYŚWIETLANIE ELEMENTÓW (9-***)

`<p>To jest pierwszy paragraf, który jest
wyświetlany domyślnie jako element blokowy</p>`

`<p>To jest drugi paragraf, który jest
wyświetlany domyślnie jako element blokowy</p>`

display: block;

//element blokowy HTML będzie
// wyświetlany w formie bloku
// (jeden pod drugim)

display: inline;

// element liniowy HTML będzie
// wyświetlany w linii
// (jeden obok drugiego)

`To jest pierwszy element, który jest wyświetlany domyślnie jako`

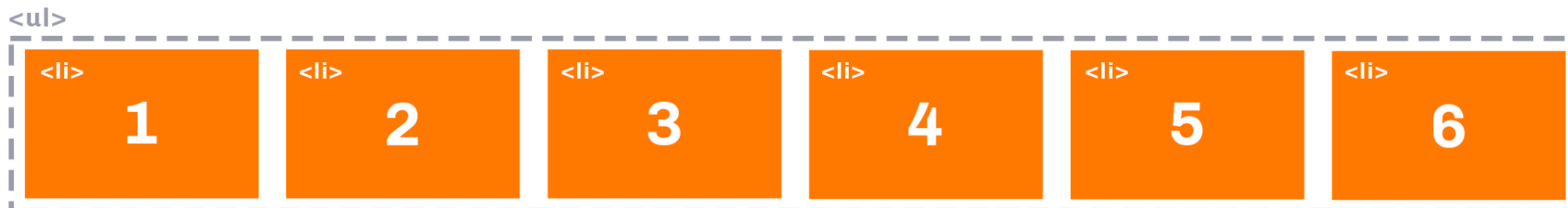
`element liniowy`

`To jest drugi element, który jest`

`wyświetlany domyślnie jako element liniowy`

WYŚWIETLANIE ELEMENTÓW (9-***)

`display: inline-block;`



Element z `display: inline-block` będzie miał pewne cechy elementu blokowego, dzięki czemu takie właściwości jak: `padding`, `margin`, `width` czy `height` będą na niego oddziaływały jak na normalny element blokowy. Równocześnie elementy liniowo-blokowe układają się w jednej linii, jak elementy liniowe.

W zaprezentowanym przykładzie elementy `li` są domyślnie elementami blokowymi i powinny być jeden pod drugim oraz zajmować całą szerokość kontenera `ul`. Po zmianie właściwości `display` elementów `li` na wartość `inline-block` wyświetlają się one w jednej linii jak elementy liniowe, równocześnie zachowując właściwości elementów blokowych.

#HTML

``

`1`

`2`

`3`

`4`

`5`

`6`

``

FLOATING - OPŁYWANIE ELEMENTÓW (10-***)

`float` – to właściwość CSS, która określa czy element powinien być opływany i z której strony

```
#CSS
img {
  float: left;
}
/* opływanie z lewej */
```

Bootcamp to coś więcej niż czysta teoria kodowania. To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić w sobie nawyki, które ułatwiają programowanie. Przybliżymy Ci narzędzia, które znacznie ułatwią pracę, a także wytłumaczymy, na jakiej zasadzie one działają. Kładziemy nacisk przede wszystkim na praktyczne umiejętności i przekładamy ponad suchą akademicką wiedzę. Pokażemy Ci również jak uczyć się bardzo szybko i mieć przy tym niezłą frajdę.

Bootcamp to coś więcej niż czysta teoria kodowania. To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić w sobie dobre nawyki, które w znacznym stopniu ułatwiają programowanie. Przybliżymy Ci narzędzia, które bardzo ułatwią pracę, a także wytłumaczymy, na jakiej zasadzie działają. Kładziemy nacisk przede wszystkim na praktyczne umiejętności i przekładamy je ponad suchą akademicką wiedzę. Pokażemy Ci jak uczyć się bardzo szybko i mieć przy tym niezłą frajdę.

```
#CSS
img {
  float: right;
}
/* opływanie z prawej */
```

OPŁYWANIE ELEMENTÓW – FLOATING (10-***)

```
#CSS
img {
  float: none;
}
/* brak opływania
(domyślna wartość
elementów HTML) */
```

Bootcamp to coś więcej niż czysta teoria kodowania.



To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić sobie nawyki, które ułatwiają programowanie. Kładziemy nacisk na praktykę.

```
#CSS
img {
  float: none;
}
p:last-child {
  clear: both;
}

/* czyszczenie opływania */
```



Bootcamp to coś więcej niż czysta teoria kodowania.

To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić sobie nawyki, które ułatwiają programowanie. Kładziemy nacisk na praktykę.

TRANSITION - EFEKTY PRZEJŚCIA W CSS (II-***)

`transition` - pozwala utworzyć efekt animacji tzw. efekt przejścia. Inaczej mówiąc na ekranie monitora zaobserwujemy płynne przejście jednych wartości właściwości CSS w inne.

Na przykład po najechaniu myszką na element `div` tło elementu zmieni się (przejdzie) z szarego na pomarańczowy. Wspomniany efekt możemy uzyskać za pomocą pseudoklasy `:hover`. Zmiana ta dokona się skokowo, ale jednak dzięki właściwościom pochodzącym z grupy `transition` możemy uzyskać efekt animacji.



TRANSITION - EFEKTY PRZEJŚCIA W CSS (II-***)

transition – właściwości efektu przejścia:

transition-property - określa, która własność stylu ma podlegać efektowi przejścia

- **dozwolone wartości:** (all – wszystkie, wybrana(e) właściwości)
- **przykład:** `transition-property: background-color;`

transition-duration – określa czas trwania efektu przejścia

- **dozwolone wartości:** (jednostki czasu - sekundy s oraz milisekundy ms)
- **przykład:** `transition-duration: 2s;`

transition-timing-function – określa sposób efektu przejścia

- **dozwolone wartości:** (np. linear (stałe tempo), ease-in (wolniej na początku), ease-out (wolniej na końcu), ease-in-out (wolniej na początku i na końcu))
- **przykład:** `transition-timing-function: linear;`

transition-delay – określa czas opóźnienia startu efektu przejścia

- **dozwolone wartości:** (jednostki czasu - sekundy s oraz milisekundy ms)
- **przykład:** `transition-duration: 500ms;`

TRANSITION - EFEKTY PRZEJSCIA W CSS (II-***)

Efekty `transition` można również zapisać w skróconej formie w jednej linii.

```
#CSS
a {
  color: gray;
  transition-property: color;
  transition-duration: 5s;
  transition-timing-function: ease-in;
  transition-delay: 2s;
}
a: hover {
  color: orange;
}
```

```
#CSS (zapis w jednej linii)
a {
  color: gray;
  transition: color 5s ease-in 2s;
}
a: hover {
  color: orange;
}
```

Właściwości `transition` mogą przyjmować wiele wartości równolegle.

```
#CSS (kilka właściwości na raz)
a {
  color: gray;
  transition: color 5s ease-in 2s, background-color 5s ease-in 2s;
}
a: hover {
  color: orange;
}
```

```
#CSS (wszystkie właściwości)
a {
  color: gray;
  background-color: white;
  transition: all 5s ease-in 2s;
}
a: hover {
  color: orange;
  background-color: black;
}
```

WARSZTATY 😊 TRANSITION

Wykorzystując dotychczasową wiedzę wykonaj następujące zadanie:

Zrób animację boxów z wykorzystaniem właściwości transition na HOVER

1. Stwórz trzy boxy w rozmiarze 150px na 150px
2. Pierwszy z nich ma się zmieniać szerokość do 300px z wykorzystaniem efektu transition
3. Drugi z nich ma się zmieniać wysokość do 300px z wykorzystaniem efektu transition
4. Trzeci z nich ma się zmieniać wysokość i szerokość do 300px z wykorzystaniem efektu transition

Podgląd:

<https://akademia108.pl/kurs-front-end/transition/>

WŁAŚCIWOŚCI TŁA (12-***)

Właściwości tła w CSS

`background-color` – określa kolor tła tła elementu HTML.

Dozwolone wartości:

- kolor – wartość koloru np. `#fff` lub funkcja `rgb(255, 255, 255)`
- `transparent` – przezroczyste tło

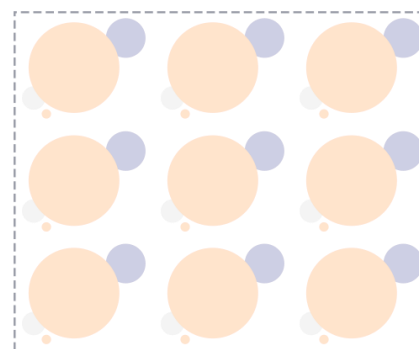


`background-color: red;`

`background-image` – wypełnienie tła obrazkiem w formacie np. `jpg, png, gif`

Dozwolone wartości:

- `url('obrazek.png')` – ścieżka do pliku z grafiką
- funkcja gradientowa – określa gradient kolorów np. `linear-gradient(gray, orange)`



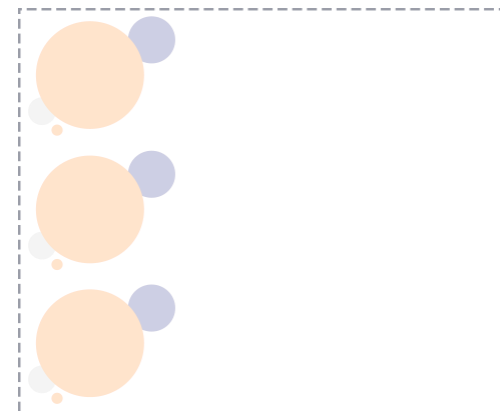
`background-image: url('obrazek.png');`

WŁAŚCIWOŚCI TŁA (12-***)

`background-repeat` – określa sposób powtarzania się obrazka jako tło elementu HTML

Dozwolone wartości:

- `repeat` – grafika powtarzana w pionie i poziomie
- `repeat-x` – grafika powtarzana w poziomie
- `repeat-y` – grafika powtarzana w pionie
- `no-repeat` – grafika nie jest powtarzana

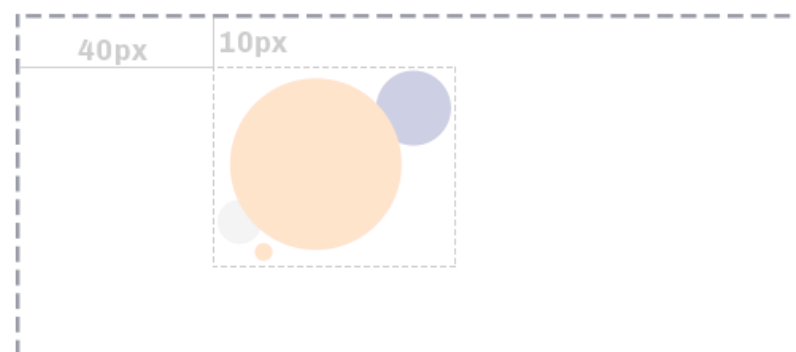


`background-repeat: repeat-y;`

`background-position` – pozycję obrazka w tle elementu HTML

Dozwolone wartości:

- `left/right/top/bottom` – wyrównanie według w/w wartości – jeśli podana jest tylko jedna wartość, to druga ustawiana jest domyślnie na `center`
- `x y` – pozycja tła w jednostkach miar np. `px`



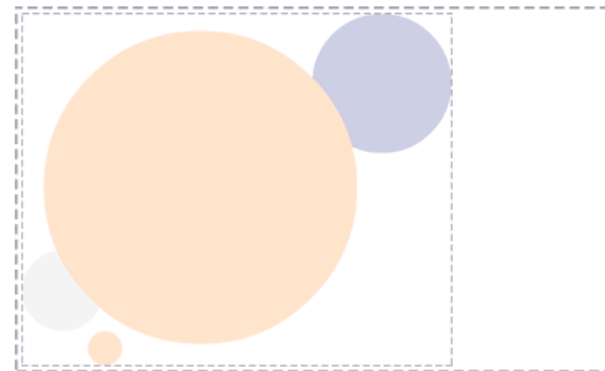
`background-position: 40px 10px;`

WŁAŚCIWOŚCI TŁA (12-***)

`background-size` – określa rozmiar obrazka w tle elementu HTML

Dozwolone wartości:

- `auto` – rzeczywiste rozmiary obrazka (wartość domyślna)
- `szerokość wysokość` – szerokość i wysokość obrazka w jednostkach miary np. `px`. Jeśli ustawiona jest tylko jedna wartość, to druga ma wartość `auto`
- `szerokość[%] wysokość[%]` – szerokość i wysokość obrazka w procentach rozmiaru rodzica `[%]`. Jeśli ustawiona jest tylko jedna wartość, to druga ma wartość `auto`
- `cover` – skalowanie grafiki tła elementu aby cały element został wypełniony grafiką. Zostają zachowane proporcje grafiki. Część grafiki nie będzie widoczna, jeśli szerokość i wysokość elementu HTML nie są proporcjonalne do rozmiarów grafiki
- `contain` – skalowanie grafiki tła elementu aby cała grafika była widoczna. Zostają zachowane proporcje grafiki. Część elementu nie będzie wypełniona grafiką jeśli rozmiar elementu HTML nie są proporcjonalne do rozmiarów grafiki



`background-size: contain;`

`background` – służy do zapisania wszystkich zaprezentowanych właściwości tła za pomocą jednej reguły

```
section#about {  
    background: white url('img/obrazek.png') center top no-repeat scroll;  
}
```

Reguły CSS: powyżej i po prawej stronie są równoważne.

```
section#about {  
    background-color: white;  
    background-image: url('img/obrazek.png');  
    background-position: center top;  
    background-size: 50% 75%;  
    background-repeat: no-repeat;  
    background-attachment: scroll;  
}
```

PRZEPEŁNIENIE ELEMENTU – OVERFLOW (13-***)

overflow (przepełnienie elementu) – ta właściwość CSS służy do tego, że jeżeli zawartość elementu nie mieści się w jego rozmiarach możliwe jest:

- ukrycie nadmiarowej zawartości,
- pokazanie wszystkiego poprzez powiększenie rozmiarów elementu (bez względu na parametry `width` i `height`)
- utworzenie suwaków (`scroll`) do przewijania przepełnionej treści.

Właściwości te działają tylko dla elementów blokowych z określoną wysokością.

Dozwolone wartości:

- `visible` – niemieszcząca się zawartość elementu pozostanie widoczna (wartość domyślna)
- `auto` – dodanie suwaków do elementu HTML gdy zawartość nie będzie się w nim mieścić
- `scroll` – dodanie suwaków do elementu, nawet jeśli zawartość mieści się w elemencie HTML
- `hidden` – ukrycie niemieszczącej się zawartości elementu

PRZEPEŁNIENIE ELEMENTU – OVERFLOW (13-***)

overflow: visible;

Bootcamp to coś więcej niż czysta teoria kodowania. To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić w sobie nawyki, które ułatwiają programowanie. Przybliżymy Ci narzędzia, które ułatwią pracę.

overflow: scroll;

Bootcamp to coś więcej niż czysta teoria kodowania. To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności

overflow: hidden;

Bootcamp to coś więcej niż czysta teoria kodowania. To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić w sobie nawyki, które ułatwiają

overflow: auto;

Bootcamp to coś więcej niż czysta teoria kodowania. To kurs, na którym pokażemy Ci jak szybko przyswoić niezbędne umiejętności oraz wyrobić w sobie nawyki, które

BOX-SIZING - ROZMIAR ELEMENTU (14-****)

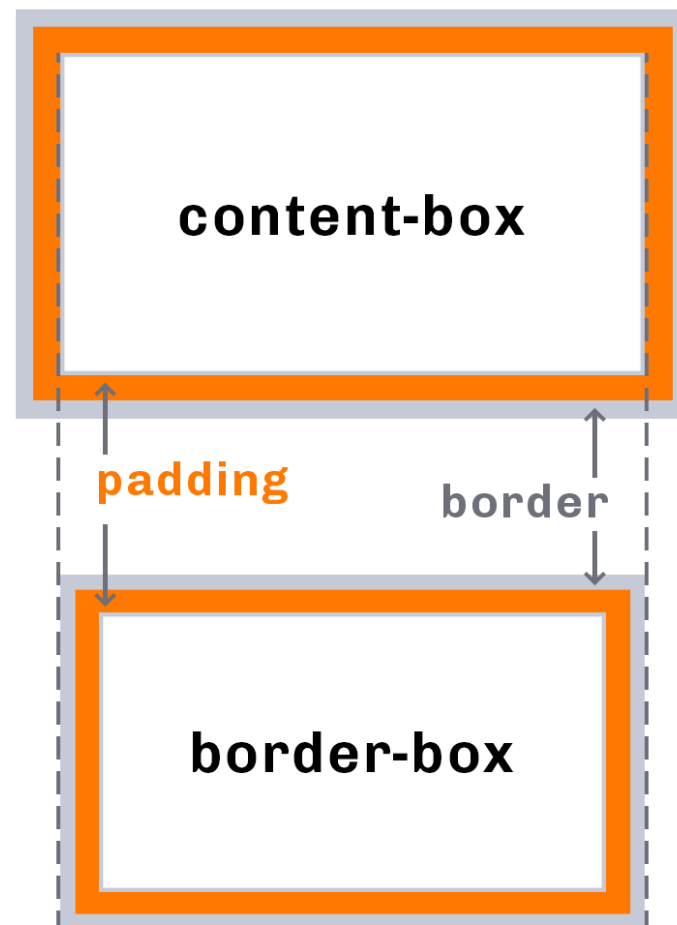
`box-sizing` - za pomocą tej właściwości możemy ustalić czy właściwości `border` oraz `padding` mają wpływać na całkowitą szerokość i wysokość elementu, które są obliczane przez przeglądarkę internetową

Dostępne wartości:

- `border-box` – właściwości `padding` oraz `border` nie mają wpływu na ustalony rozmiar elementu
- `content-box` – właściwości `padding` oraz `border` zmieniają ustalony rozmiar elementu – wartość domyślna

Przykład:

```
section {  
  box-sizing: content-box;  
}
```



RESPONSYWNOŚĆ ZA POMOCĄ CSS (15-***)

Responsywność RWD –
(Responsive Web Design)
– pozwala na szybkie i sprawne
dostosowywanie się strony
internetowej do rozdzielczości i
rozmiaru ekranu np. tabletu lub
smartfonu oraz ułatwia obsługę
strony za pomocą mobilnych
urządzeń.



RESPONSYWNOŚĆ ZA POMOCĄ CSS (15-***)

W celu uzyskania efektu automatycznego skalowania i odpowiedniej szerokości obrazu na urządzeniach mobilnych używamy metatagu `viewport`, który umieszczamy w nagłówku strony `<head>`

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" >
```

Wartość atrybutu `content` definiuje:

- wyjściową szerokość strony `width=device-width` - szerokość wyświetlacza urządzenia
- wyjściowe powiększenie `initial-scale=1.0` - domyślne powiększenie

Mobile first – podejście do tworzenia responsywności (RWD), uwzględniające najpierw projekt na urządzenia mobilne, a dopiero potem wersje dla większych ekranów.



RESPONSYWNOŚĆ ZA POMOCĄ CSS (15-***)

`@media queries` - służą do definiowania różnych właściwości CSS dla różnych typów mediów / urządzeń / rozdzielczości.

Za pomocą tych reguł możemy sprawdzić np.:

- typ urządzenia
- rozdzielczość urządzenia
- orientację pionową lub poziomą urządzenia

`min-width/min-height` oraz `max-width/max-height` - określa reguły CSS, które zostaną aktywowane tylko wtedy, gdy szerokość/wysokość okna przeglądarki internetowej przekroczy zadaną minimalną lub maksymalną wartość.

```
@media (min-width:800px) and (max-width:1200px) {  
  div {  
    color: red;  
  }  
}
```

/ W tym przykładzie czcionka wszystkich elementów div będzie miała kolor czerwony w zakresie rozdzielczości od 800px do 1200px */*

RESPONSYWNOŚĆ ZA POMOCĄ CSS (15-***)

Typy mediów - nazwy mediów są odbiciem urządzeń docelowych, dla których stosowne własności mają sens np:

- all – dla wszystkich mediów
- screen – dla komputerów, tabletów, smartfonów itp.
- print – dla drukarek
- speech – dla czytników ekranu

Przykład:

```
@media screen, print {  
  div {  
    color: red;  
  }  
}
```

*/*W tym przykładzie czcionka wszystkich elementów div będzie miała kolor czerwony dla wersji do druku oraz dla wszystkich urządzeń ekranowych typu monitor, tablet, smartfon */*

RESPONSYWNOŚĆ ZA POMOCĄ CSS (15-***)

Operatory logiczne - dla reguł @media queries

- and - łączy różne właściwości w regule @media
- not - operator ten umożliwia zaprzeczenia w regule @media
- przecinek(, lub OR) – operator ten umożliwia stworzenie jednego zapisu reguły @media dla różnych typów urządzeń

Przykład:

```
@media (min-width:800px) and (max-width:1200px) {  
  div {  
    color:red;  
  }  
}
```

/* Po lewej stronie przykład zastosowania operatora logicznego and */

```
@media not print {  
  div {  
    color:green;  
  }  
}
```

/* Po lewej stronie przykład zastosowania operatora logicznego not */

```
@media screen, print {  
  div {  
    color:red;  
  }  
}
```

/* Po lewej stronie przykład zastosowania operatora przecinek (,) */

WARSZTATY 😊 RWD MEDIA QUERIES

Wykorzystując dotychczasową wiedzę wykonaj następujące zadanie:

Dostosuj galerię do urządzeń mobilnych

1. Pobierz paczkę z plikami <https://akademia108.pl/kurs-front-end/rwd.zip>
2. Galeria jest nieresponsywna i ma bloki nierelatywnej wielkości
3. Stwórz plik `responsive.css` gdzie nadpisujemy wielkości wyrażone w pikselach na jednostki relatywne
4. Dodaj odpowiednie warunki w media queries, które sprawią, że obrazki w galerii będą w trzech kolumnach dla urządzeń z dużym wyświetlaczem (powyżej 992px), w dwóch dla urządzeń ze średnim wyświetlaczem (powyżej 768px do 992px) i w jednej kolumnie dla urządzeń z małym wyświetlaczem (dla 768px i poniżej)
5. Dla urządzeń z bardzo małym wyświetlaczem (poniżej 480px) główna sekcja oraz obrazki w galerii mają mieć całe 100% szerokości ekranu

Podgląd: <https://akademia108.pl/kurs-front-end/rwd/>

TRANSFORM – ROTACJA (16-***)

`transform` – pozwala na wizualną zmianę elementu np. jego rotację, skalowanie.

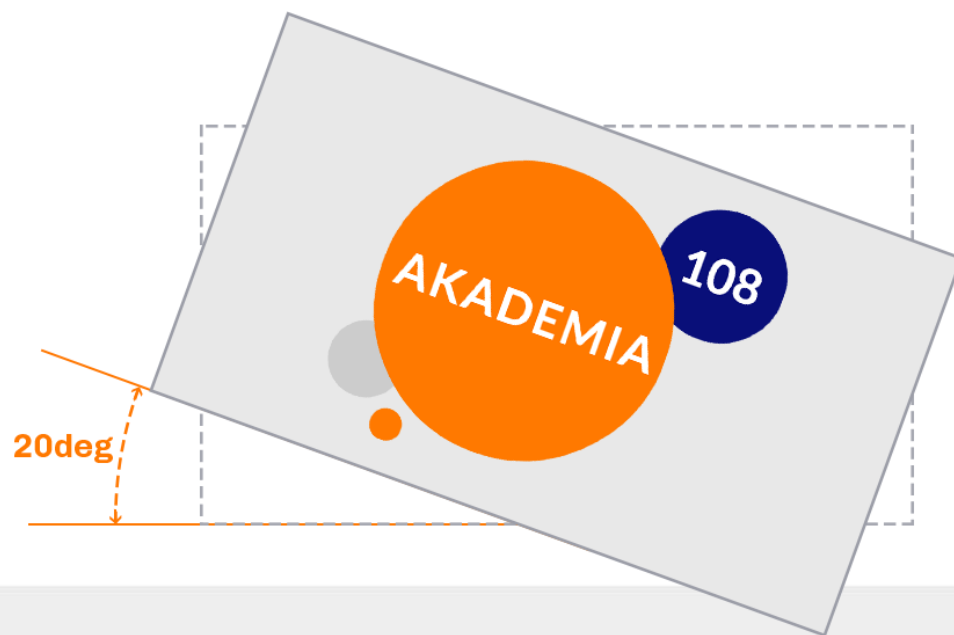
Rotacja - obraz jest obracany względem jego środka, który jest domyślnym punktem odniesienia dla wszystkich transformacji.

Dozwolone wartości:

- `rotate(deg)` – rotacja względem środka obiektu
- `rotateX(deg)` – funkcja rotuje element w osi poziomej
- `rotateY(deg)` – funkcja rotuje element w osi pionowej

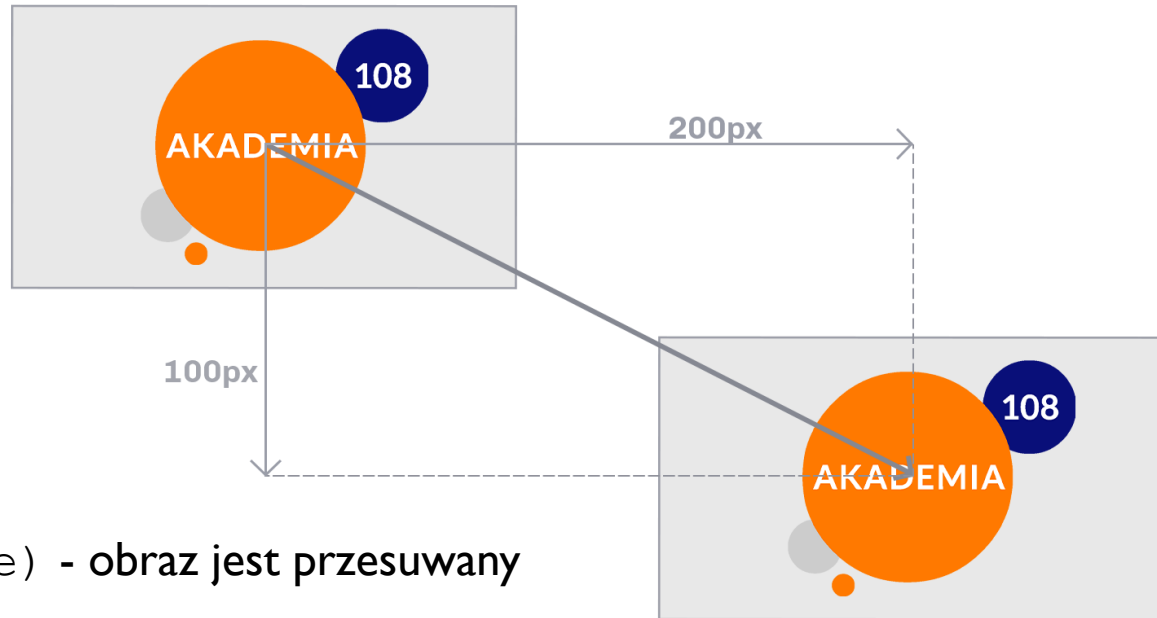
#CSS

```
div#logo {  
    transform: rotate(20deg);  
}
```



TRANSFORM – PRZESUNIĘCIE (16-***)

```
#CSS
div#logo {
  transform: translate(200px, 100px);
}
```



Przesunięcie (translate) - obraz jest przesuwany względem jego środka.

Dozwolone wartości:

- `translate(x, y)` – funkcja przesuwa element w osi poziomej i pionowej
- `translateX(x)` – funkcja przesuwa element w osi poziomej
- `translateY(z)` – funkcja przesuwa element w osi pionowej



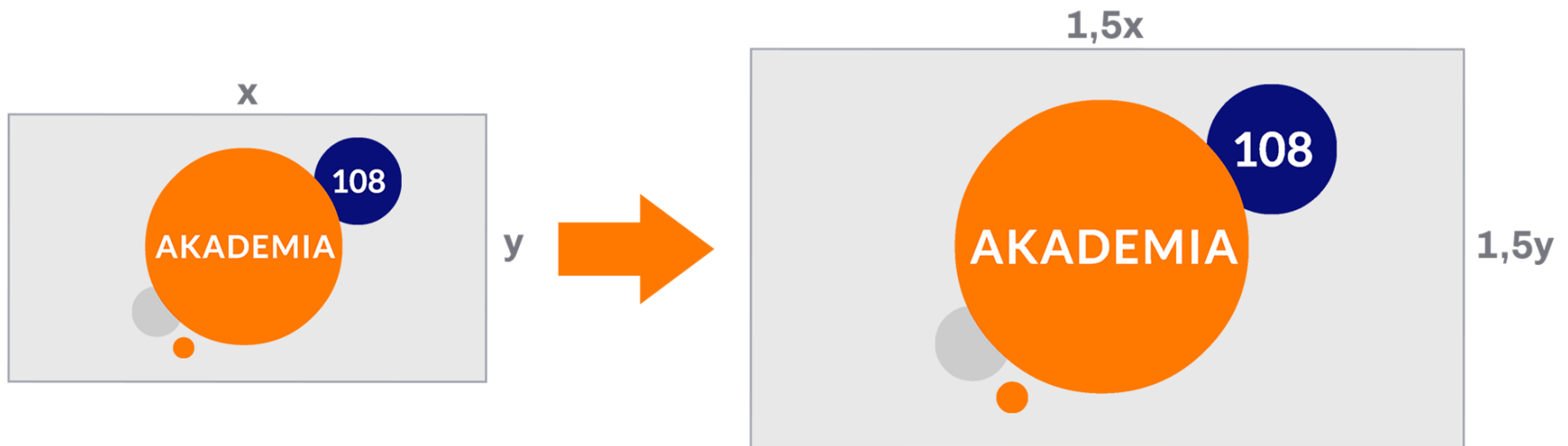
TRANSFORM – SKALOWANIE (16-***)

Skalowanie – powiększanie lub pomniejszanie obiektu na stronie

Dozwolone wartości:

- `scale(x, y)` – skalowanie w pionie i poziomie
- `scaleX(x)` – skalowanie w poziomie
- `scaleY(y)` – skalowanie w pionie

```
#CSS
div#logo {
  transform: scale(1.5, 1.5);
}
```



EFEKTY ANIMACJI W CSS (16-***)

`animation` – właściwość CSS pozwalająca utworzyć efekt pełnej animacji. Jej ideą animacji jest płynna zmiana stylu danego elementu.

Cztery podstawowe właściwości animacji to:

`animation-name` – nazwa reguły `@keyframes`, która ma być wykorzystana do animacji

`animation-duration` – określa czas trwania efektu animacji

- **dozwolone wartości:** (jednostki czasu - sekundy `s` oraz milisekundy `ms`)
- **przykład:** `animation-duration: 2s;`

`animation-iteration-count` – określa ilość powtórzeń efektu animacji

- **dozwolone wartości:** (liczby naturalne – określają ilość powtórzeń, `infinity` – animacja powtarza się w nieskończoność)
- **przykład:** `animation-iteration-count: 6;`

`animation-timing-function` – określa sposób efektu przejścia

- **dozwolone wartości:** (np. `linear` (stałe tempo), `ease-in` (wolniej na początku), `ease-out` (wolniej na końcu), `ease-in-out` (wolniej na początku i na końcu))
- **przykład:** `animation-timing-function: linear;`

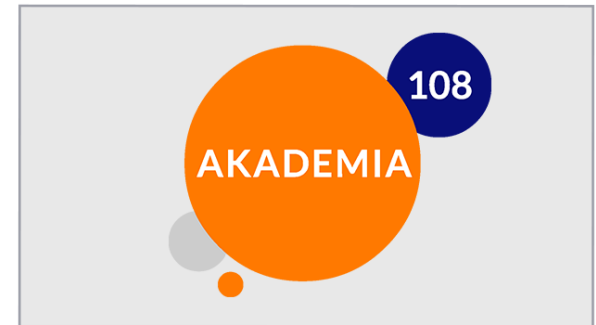
EFEKTY ANIMACJI W CSS (16-***)

@keyframes - określa stan początkowy i końcowy animacji lub procentowe rozmieszczenie klatek animacji.

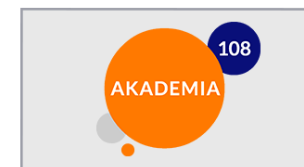
```
#CSS
@keyframes resize {
  0%, 100% {
    width: 100%;
  }
  50% {
    width: 50%;
  }
}

div#img {
  animation-name: resize;
  animation-duration: 2s;
  animation-iteration-count: infinite;
}
```

0, 100%



50%



FLEXBOX – display: flex (17-****)

Flexbox (elastyczne pudełko 😊) – to zbiór właściwości CSS pozwalających zarządzać elementami HTML w układzie rzędów i kolumn.

W flexbox wyróżniamy 2 najważniejsze pojęcia:

- kontener (flex container) – element HTML, który będzie kontenerem przechowującym elementy, z których będziemy budować rzędy i kolumny
- element (flex item) – elementy HTML, który jest dzieckiem kontenera i stanowi komórkę w rzędzie lub/i kolumnie

`display: flex;` - to podstawowa właściwość CSS, która sprawia, że dany element HTML zostaje wyświetlony jako kontener flexbox, a jego elementy potomne mogą być ustawiane pozostałymi właściwościami CSS jako flex item. Domyślnie wszystkie elementy potomne ustawia się w rzędzie.

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



FLEXBOX – flex-basis (17-***)

`flex-basis: <length> | auto;` – ta właściwość określa bazową szerokość elementu w rzędzie lub wysokość elementu w kolumnie. Wartością `flex-basis` jest zwykle jednostka `%`. Domyślną wartością jest `auto`. – czyli element(y) dopasuje swoją szerokość automatycznie.

#CSS

```
.flex-item {  
  flex-basis: 25%;  
}
```

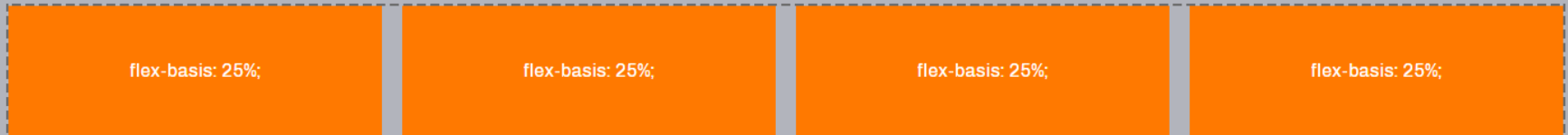
#CSS

```
.flex-item-first {  
  flex-basis: 50%;  
}
```

#CSS

```
.flex-item-last {  
  flex-basis: 10%;  
}
```

container



container



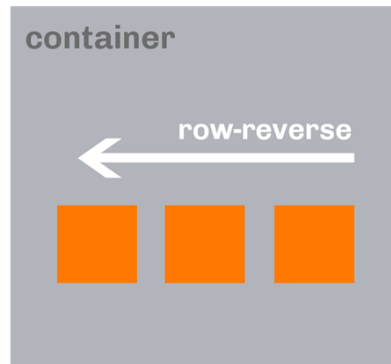
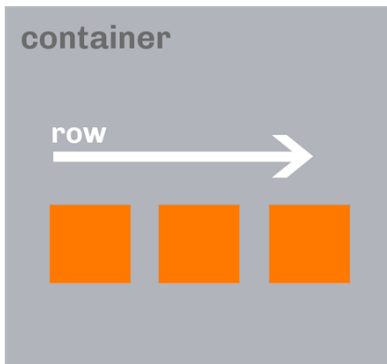
FLEXBOX – flex-direction (17-***)

`flex-direction: row | row-reverse | column | column-reverse;` - ta właściwość określa czy elementy mają się układać w rzędzie (domyślnie), odwróconym rzędzie, kolumnie czy odwróconej kolumnie.

#CSS

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

/ flex-direction określa sposób
ułożenia się item'ów w kontenerze flex */*



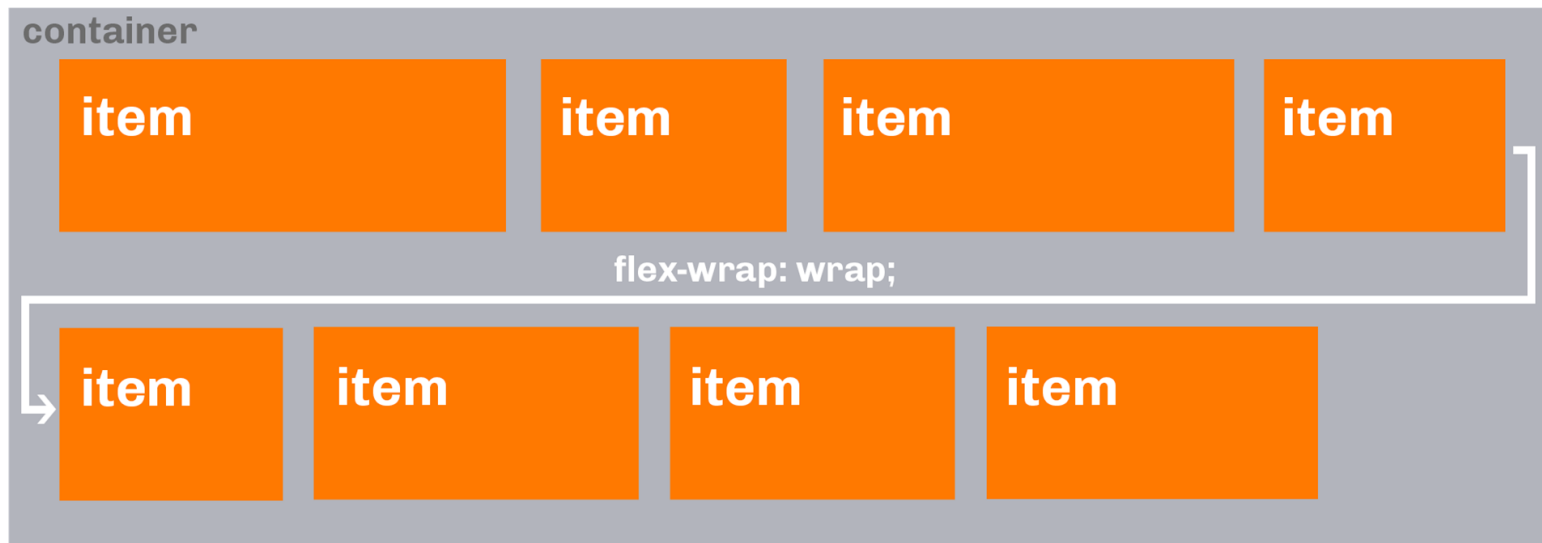
FLEXBOX – flex-wrap (17-***)

`flex-wrap: nowrap | wrap | wrap-reverse;` - ta właściwość określa jak elementy mają się zachować, gdy ich łączny rozmiar jest większy niż rozmiar kontenera

Wartości flex-wrap:

- `nowrap` - są ułożone w jednym rzędzie (domyślna wartość)
- `wrap` - zawijają się w normalnej kolejności
- `wrap-reverse` - lub zawijają się z odwróconą kolejnością

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



FLEXBOX – justify-content (17-***)

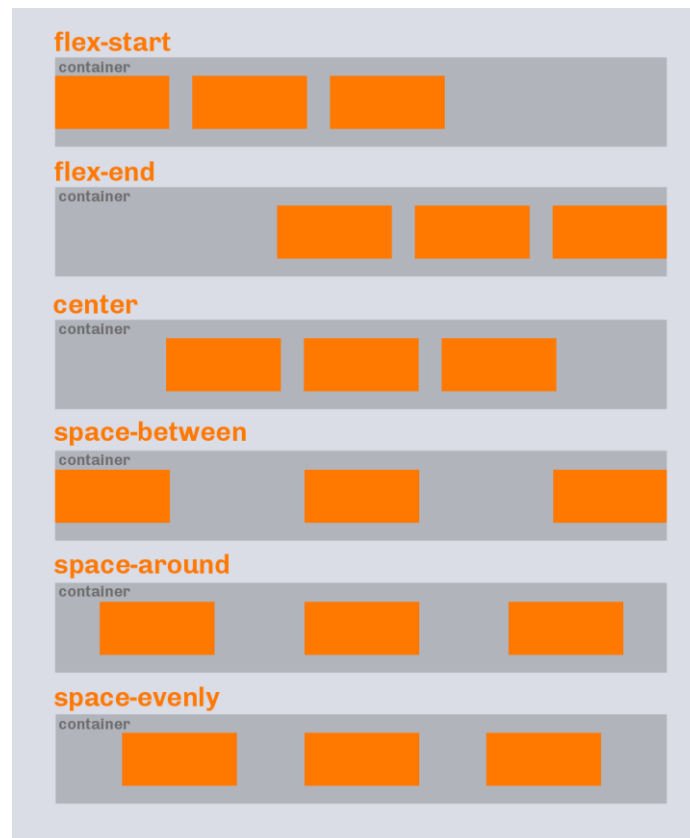
`justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;`
- ta właściwość określa jak mają się zachować elementy względem osi pionowej w swojej linii (wierszu) w kontenerze flex.

Wartości:

- `flex-start` – od początku kontenera
- `flex-end` – od końca kontenera
- `center` – do środka kontenera
- `space-between` – wyjustowane
- `space-around` – równa przestrzeń na około elementów
- `space-evenly` – równy odstęp między elementami

#CSS

```
.container {  
    display: flex;  
    flex-direction: row;  
    flex-wrap: wrap;  
    justify-content: center;  
}
```



FLEXBOX – align-content (17-***)

`align-content: flex-start | flex-end | center | space-between | space-around | stretch;` - ta właściwość określa jak mają się zachować linie (wiersze) elementów względem osi poziomej kontenera.

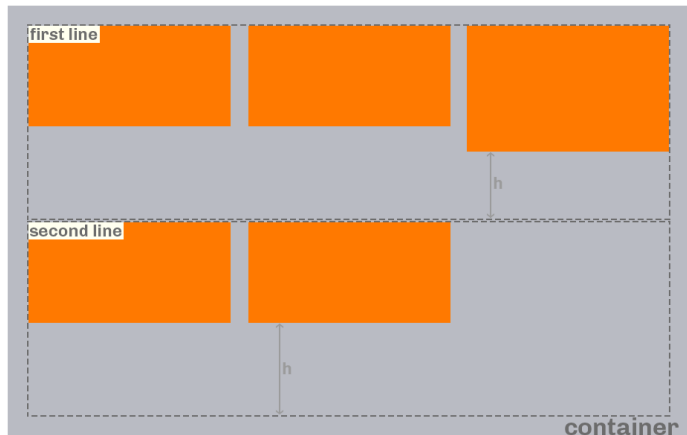
Wartości:

- `stretch` – rozciągnięte (domyślnie)
- `flex-start` – do góry kontenera
- `flex-end` – do dołu kontenera
- `center` – do środka kontenera
- `space-between` – wyjustowane w pionie
- `space-around` – równa przestrzeń na około rzędów

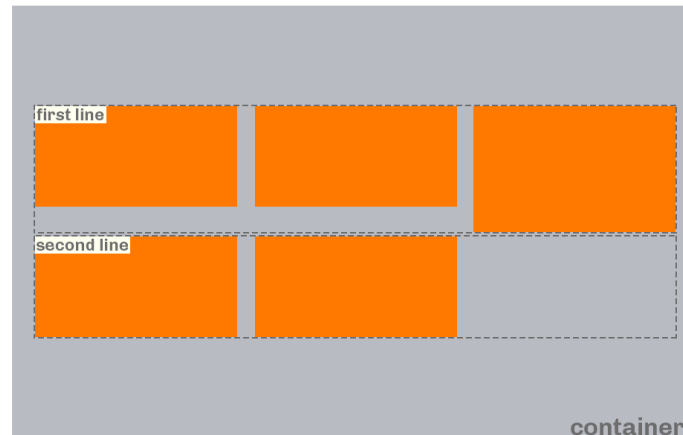
#CSS

```
.container {  
    display: flex;  
    flex-direction: row;  
    flex-wrap: wrap;  
    justify-content: center;  
    align-content: center;  
}
```

stretch



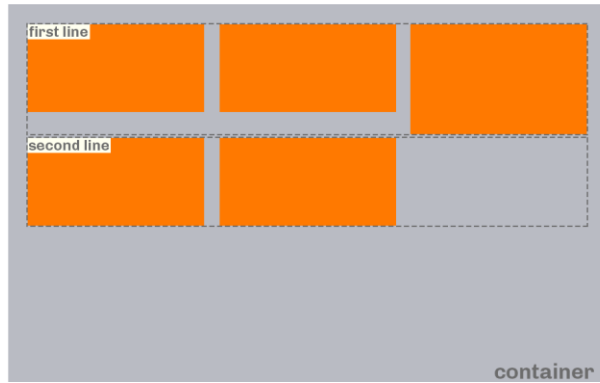
center



FLEXBOX – align-content (17-***)

align-content c.d.

flex-start



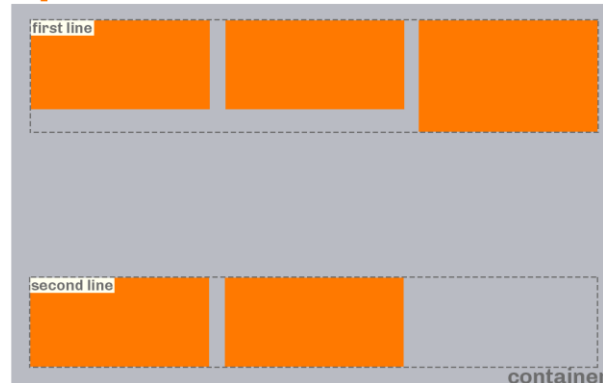
flex-end



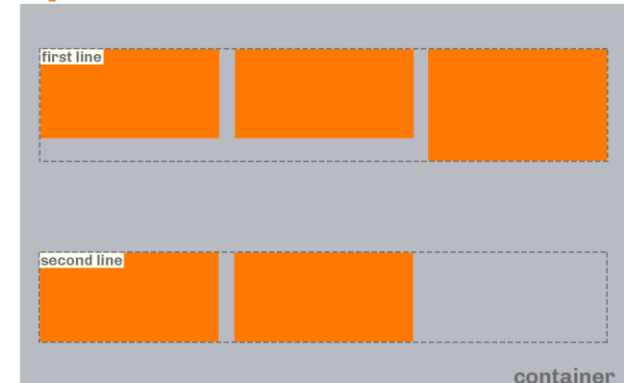
```
#CSS
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
  align-content: flex-end;
}
```

```
#CSS
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
  align-content: space-around;
}
```

space-between



space-around



FLEXBOX – align-items (17-***)

`align-items: flex-start | flex-end | center | baseline | stretch;` - ta właściwość określa jak elementy mają się zachować względem osi poziomej w swojej linii (swoim wierszu).

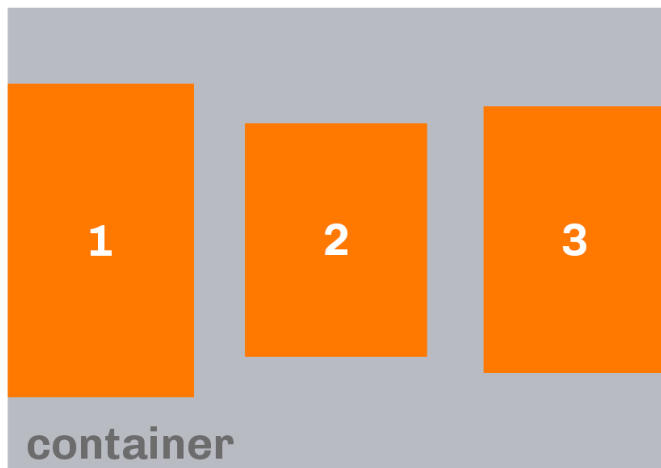
Wartości:

- `stretch` – rozciągnięte (domyślna)
- `flex-start` – do góry kontenera
- `flex-end` – do dołu kontenera
- `center` – do środka kontenera
- `baseline` – do bazowej linii tekstu

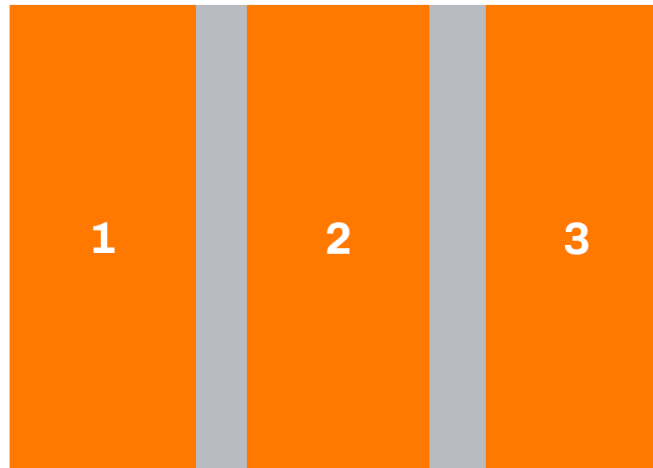
#CSS

```
.container {  
    display: flex;  
    flex-direction: row;  
    flex-wrap: wrap;  
    justify-content: center;  
    align-items: center;  
}
```

center



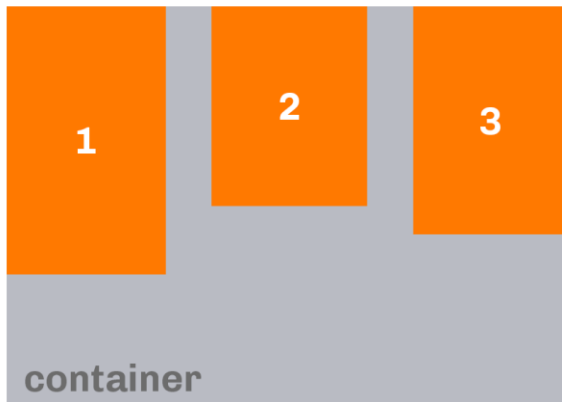
stretch



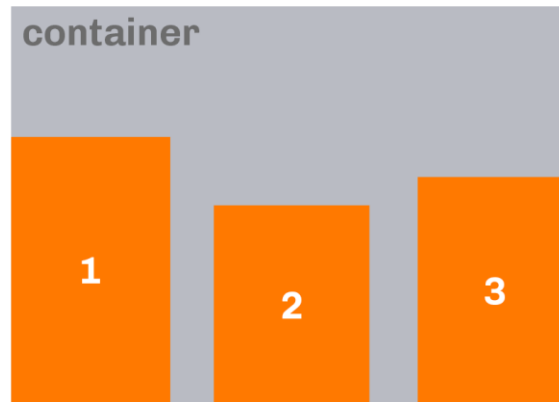
FLEXBOX – align-items (17-***)

align-items c.d.

flex-start



flex-end

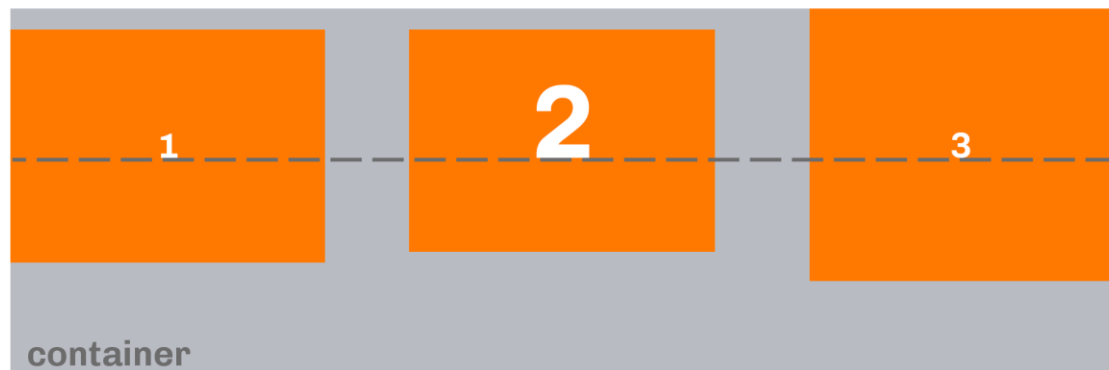


```
#CSS
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
  align-items: flex-start;
}
```

#CSS

```
.container {
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: center;
  align-items: baseline;
}
```

baseline

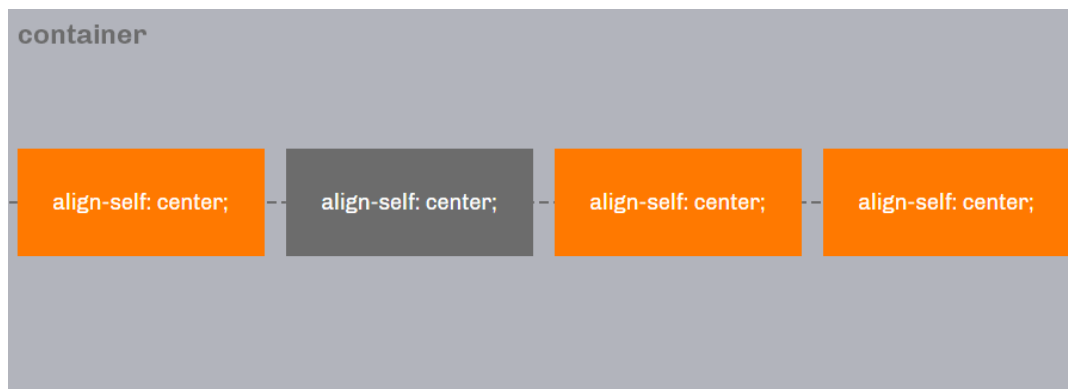


FLEXBOX – align-self (I 7-***)

`align-self: flex-start|flex-end|center|baseline|stretch;` – to odpowiednik `align-items`, tylko działający na pojedynczy element.

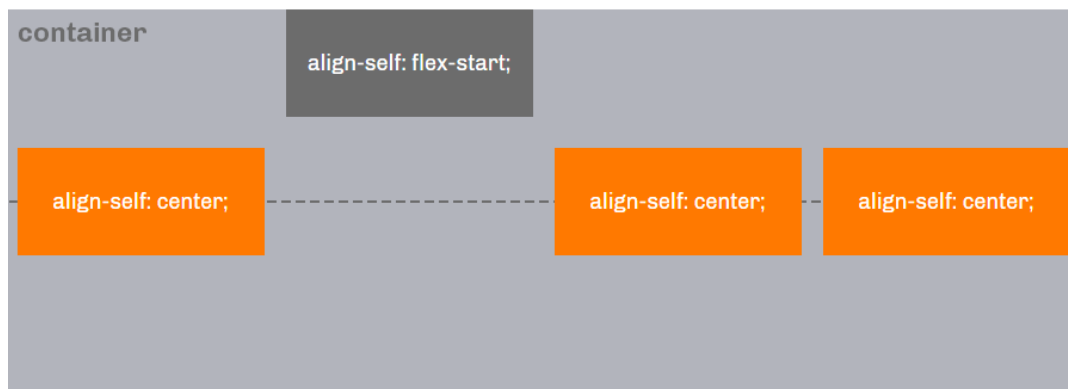
Wartości:

- `flex-start` – do góry kontenera
- `flex-end` – do dołu kontenera
- `center` – do środka kontenera
- `baseline` – do bazowej linii tekstu
- `stretch` – rozciągnięte (domyślna)



#CSS

```
.flex-item {  
    align-self: flex-start;  
}
```



FLEXBOX – order (17-***)

`order: <integer>;` - ta właściwość określa kolejność wyświetlania elementu w rzędzie lub w kolumnie. Wartością tej właściwości jest liczba całkowita – domyślnie 0.

#CSS

```
.flex-item {  
  order: 0;  
}
```

```
.first-ordered {  
  order: 1;  
}
```

```
.second-ordered {  
  order: 2;  
}
```

container

order: 0;

order: 1;

order: 1;

order: 2;

FLEXBOX – flex-grow (17-***)

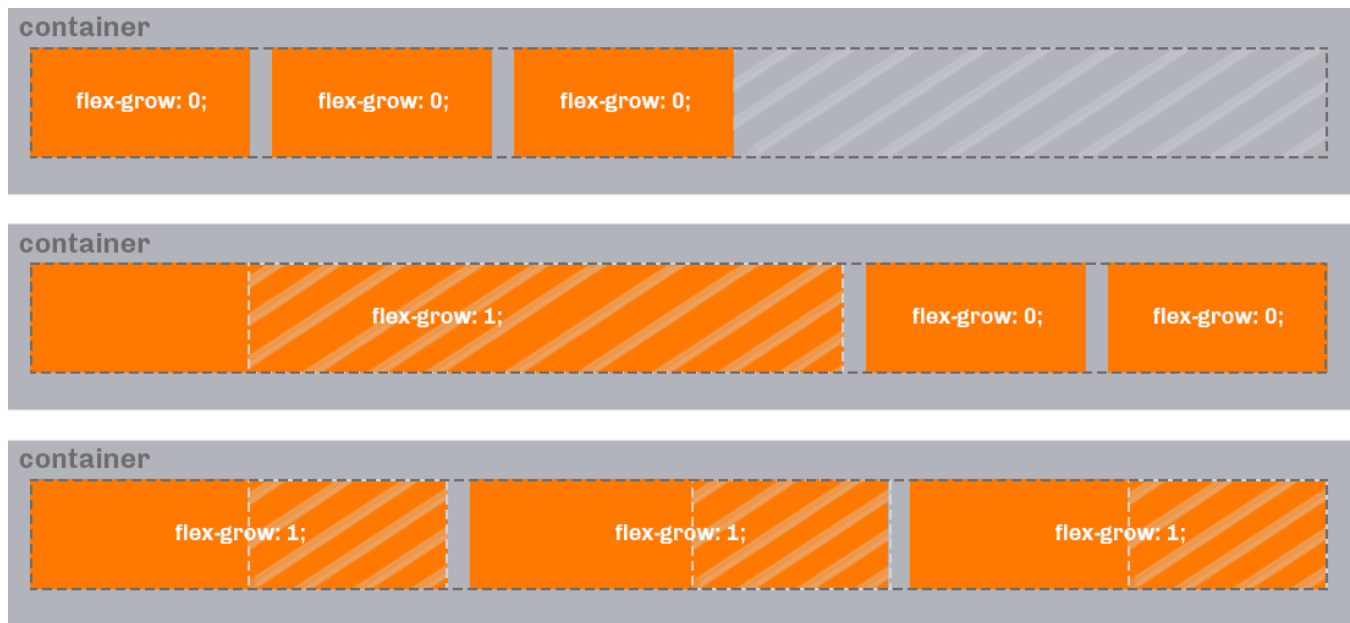
`flex-grow: <natural number>;` - ta właściwość działa, gdy w kontenerze zostaje wolna przestrzeń.

Wartości:

- Domyślną wartością jest 0, wtedy elementy ignorują wolną przestrzeń.
- Ustawienie właściwości na liczbę większą od 0 dla wszystkich elementów sprawia, że elementy wypełniają wolną przestrzeń.
- Ustawienie różnych wartości dla elementów w tym samym kontenerze określa proporcje w jakiej zajmą puste miejsce. Wartością tej właściwości CSS jest liczba naturalna.

#CSS

```
.flex-item {  
    flex-grow: 1;  
}
```

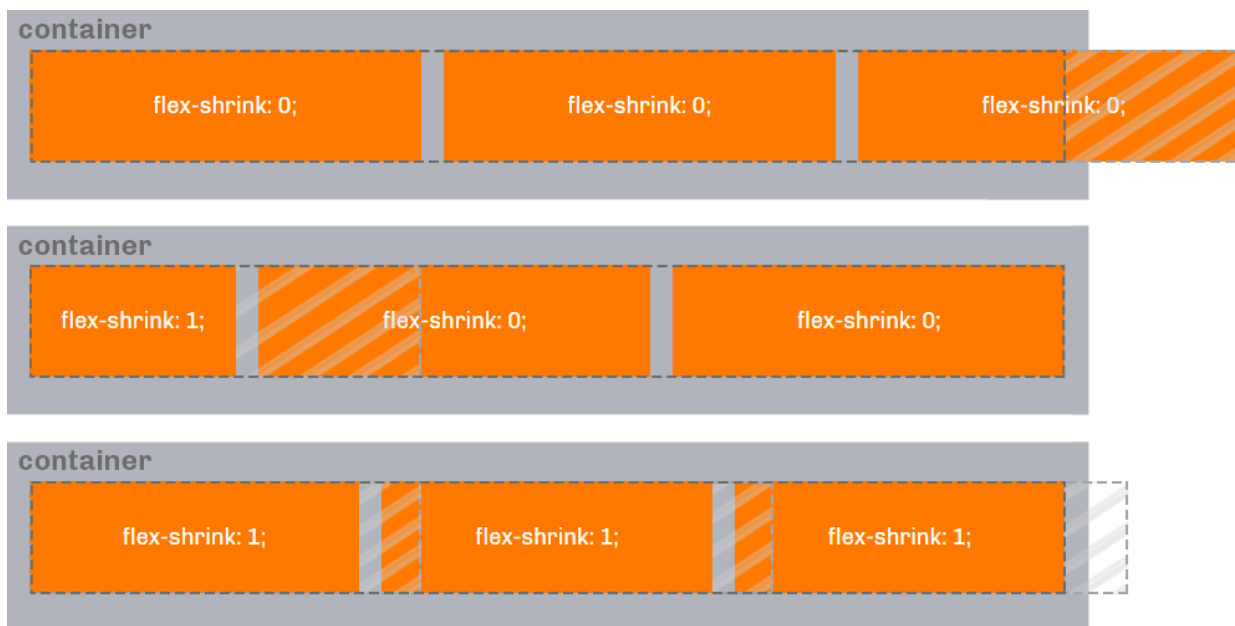


FLEXBOX – flex-shrink (17-***)

`flex-shrink: <natural number>;` - ta właściwość działa, gdy w kontenerze zostaje przepełniony (nie działa przy użyciu np. `flex-wrap: wrap`;). Domyślna wartość to 1.

Wartości:

- Domyślną wartością jest 0, wtedy elementy ignorują przepełnienie kontenera.
- Ustawienie właściwości na liczbę większą od 0 dla wszystkich elementów sprawia, że elementy oddają część swojej szerokości aby zmieścić się w kontenerze
- Ustawienie różnych wartości dla elementów w tym samym kontenerze określa proporcje w jakiej oddają część swojej szerokości. Wartością tej właściwości CSS jest liczba naturalna.



```
#CSS
.flex-item {
    flex-shrink: 1;
}
```

FLEXBOX – flex (17-***)

`flex: none | [<flex-grow> | <flex-shrink> | <flex-basis>]`; - to skrótowy zapis łączący właściwości `flex-grow`, `flex-shrink`, `flex-basis`.

Ustawienie wartości na `none` sprawi, że element przestanie być elementem flex.

```
#CSS
.flex-item {
  flex: 2 1 25%;
}
```

```
/* Basic values */
flex: auto;
flex: initial;
flex: none;
flex: 2;
flex: 25%;
```

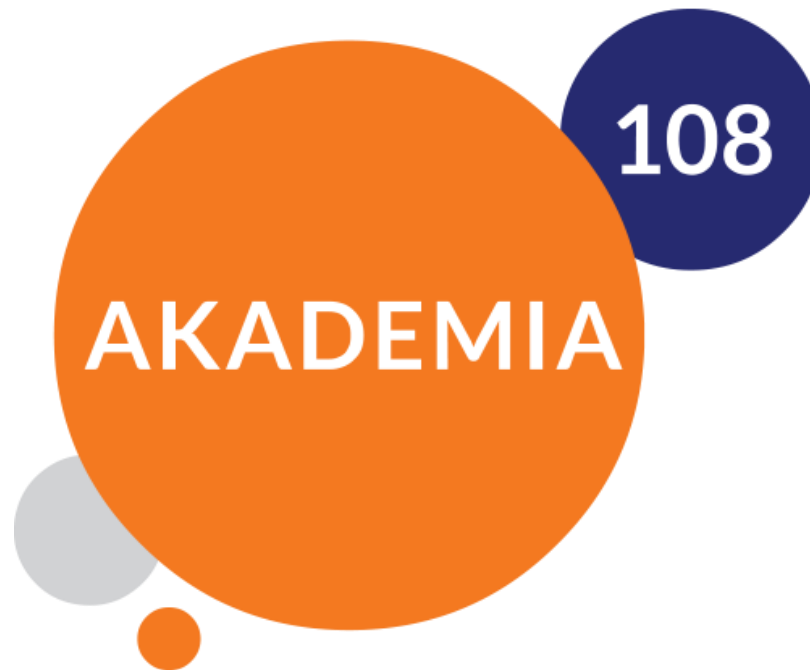
```
/* Jedna wartość, bez jednostki to: flex-grow */
flex: 2;
```

```
/* Jedna wartość z jednostką to: flex-basis */
flex: 10em;
flex: 30%;
```

```
/* Dwie wartości: flex-grow | flex-basis */
flex: 1 30%;
```

```
/* Dwie wartości: flex-grow | flex-shrink */
flex: 2 2;
```

```
/* Trzy wartości: flex-grow | flex-shrink | flex-basis */
flex: 2 2 10%;
```



Akademia 108

<https://akademia108.pl>