Flexbox

Layout elastyczny

Jak MUSIELIŚMY tworzyć layouty stron (elementów)?

position (absolute)

float

inline-block

Jak MOŻEMY TERAZ tworzyć layouty stron?

Tak samo ale możemy też użyć Flexbox

Flexbox czyli?

Elastyczny układ pudełkowy. Zapewnia płynność (skalowalność). Elementy dopasowują się do miejsca, które mają. Ograniczeniem dla nich jest kontener w którym się znajdują.

Teoretycznie upraszcza.

Flexbox czyli?

Teoretycznie upraszcza.

W praktyce też, ale wymaga pewnej biegłości ;)

KONTENER

```
<section>
        <div>1</div>
        <div>2</div>
        <div>3</div>
<section>

section {display: flex;}  /* KONTENER */
```

ELEMENTY ELASTYCZNE

```
<section>
        <div>1</div> // elementy elastyczne* - bezpośredni potomkowie kontenera.
        <div>2</div>
        <div>3</div>
<section>
```

```
section {display: flex;}
```

TYLKO DZIECI

```
<section>
        <div>1</div>
        <div>2</div>
        <div>3
                <p>4</p>
        </div>
<section>
```

```
section {display: flex;}
```

* ten element nie jest już elastyczny! Tylko dzieci.
<p> w tym przykładzie zachowuje się juz "normalnie".

USTAWIENIA DOMYŚLNE

```
<section>
        <div>1</div>
        <div>2</div>
        <div>3</div>
<section>

section {
        display: flex;
}
div {
        }
```

DOMYŚLNIE

- **elementy elastyczne ustawiają się obok siebie w poziomie (flex-direction: row) od lewej strony (justify-content: flex-start)**
- **zajmują tyle miejsca ile potrzebują na szerokość (flex-basis: auto) i na 100% wysokości rodzica (align-items: stretch).**
- **Wszystkie mieszczą się w jednej linii (flex-wrap: nowrap) i zmniejszą się w tej samej proporcji (flex-shrink: 1)**

FLEXBOX - WIELE NA STRONIE

```
<nav></nav>
<section style="display:flex">
        <div>1</div>
        <div>1</div>
        <div style="display:flex">
                <p>coś</p>
                <p>coś</p>
        </div>
<section>
```

Granice układu elastycznego - znacznik, któremu nadaliśmy display:flex

Elementy układu elastycznego - dzieci (bezpośredni potomkowie) elementu, któremu nadaliśmy właściwość display:flex;

WŁAŚCIWOŚCI NADAWANE RODZICOWI

- display (niezbędny by "włączyć" flex)
- flex-direction /domyślnie "row"/
- flex-wrap /domyślnie "nowrap"/
- align-items /domyślnie "stretch"/
- justify-content /domyślnie "flex-start"/
- align-content /domyślnie "stretch"/

display (flex)

Przyjmuje dwie wartości związane z flexem

display: flex; // kontener ma szerokość na osi głównej 100% rodzica.

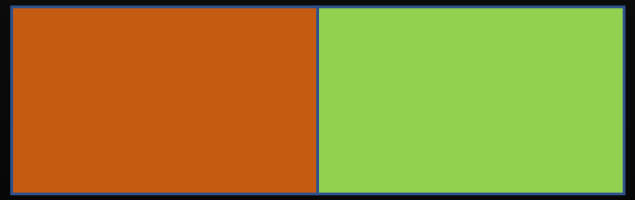
display: inline-flex; //kontener ma szerokość na osi głównej tyle ile zajmują dzieci.

display

display: flex;

Kontener zajmuje 100% szerokości rodzica (przy osi głównej poziomej)

display: inline-flex;



Kontener zajmuje NA SZEROKOŚĆ tyle miejsca ile potrzebują dzieci

flex-direction

Przyjmuje cztery wartości.

flex-direction: row; //(domyślna) elementy obok siebie
flex-direction: column; //elementy pod sobą

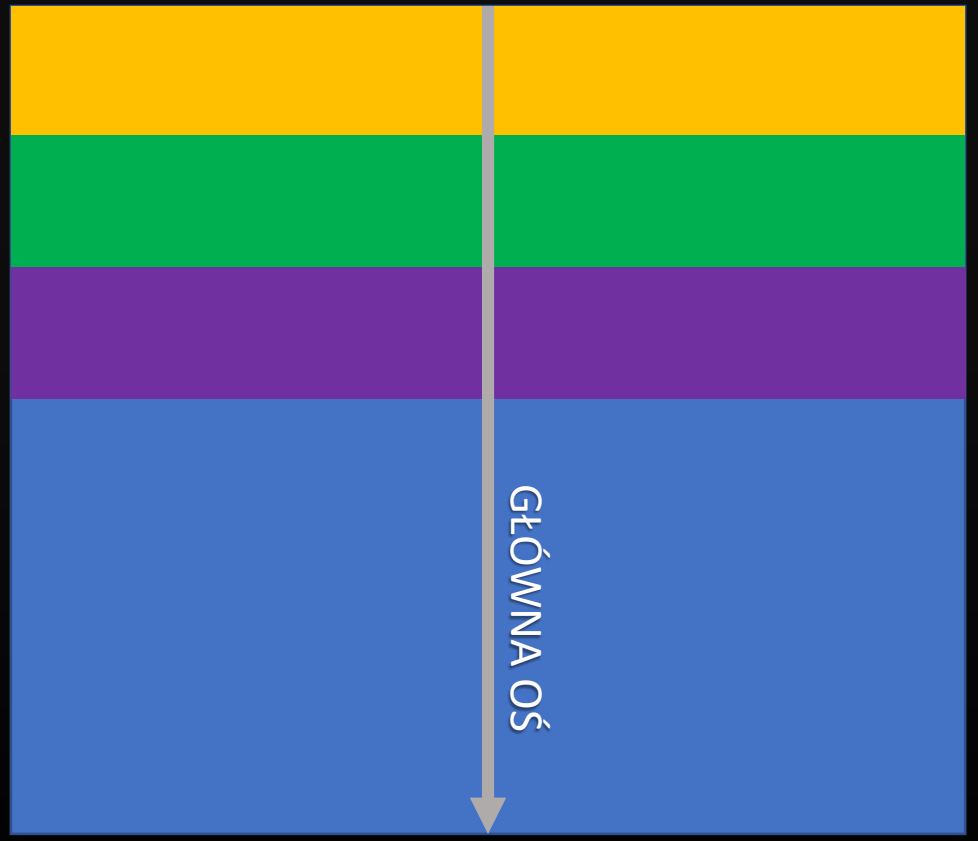
flex-direction: row-reverse;
flex-direction: column-reverse;

flex-direction

Kluczem do zrozumienia flexboxa jest zrozumienie czym jest oś główna

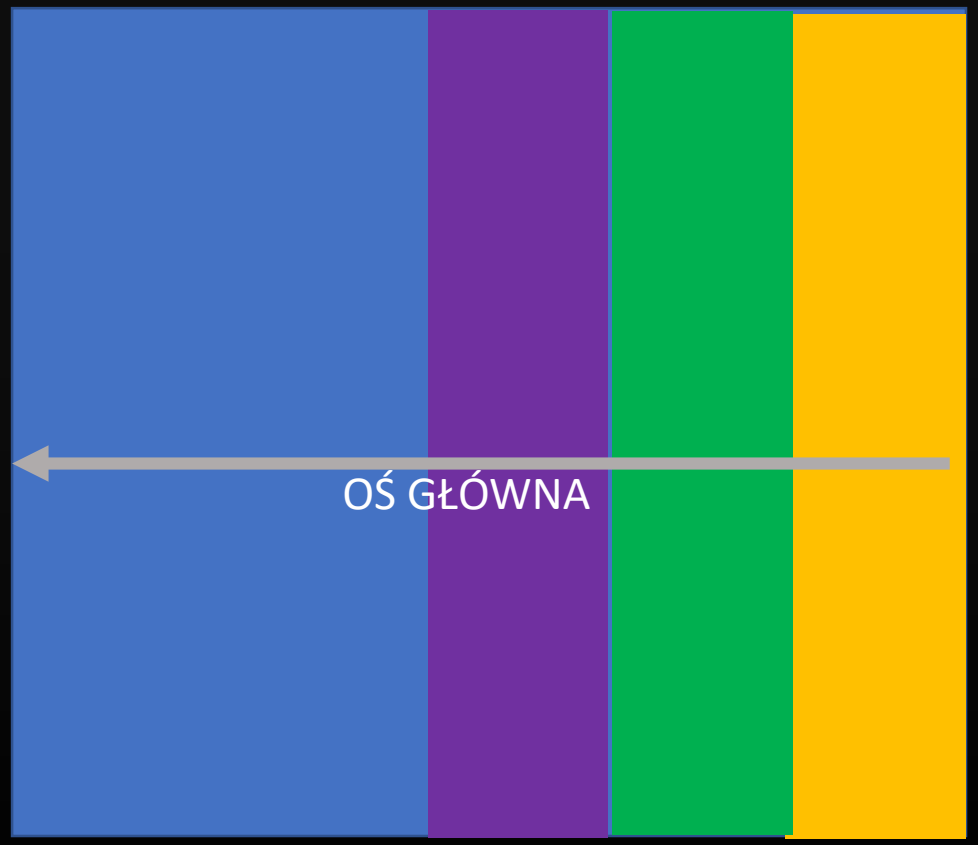
row

column

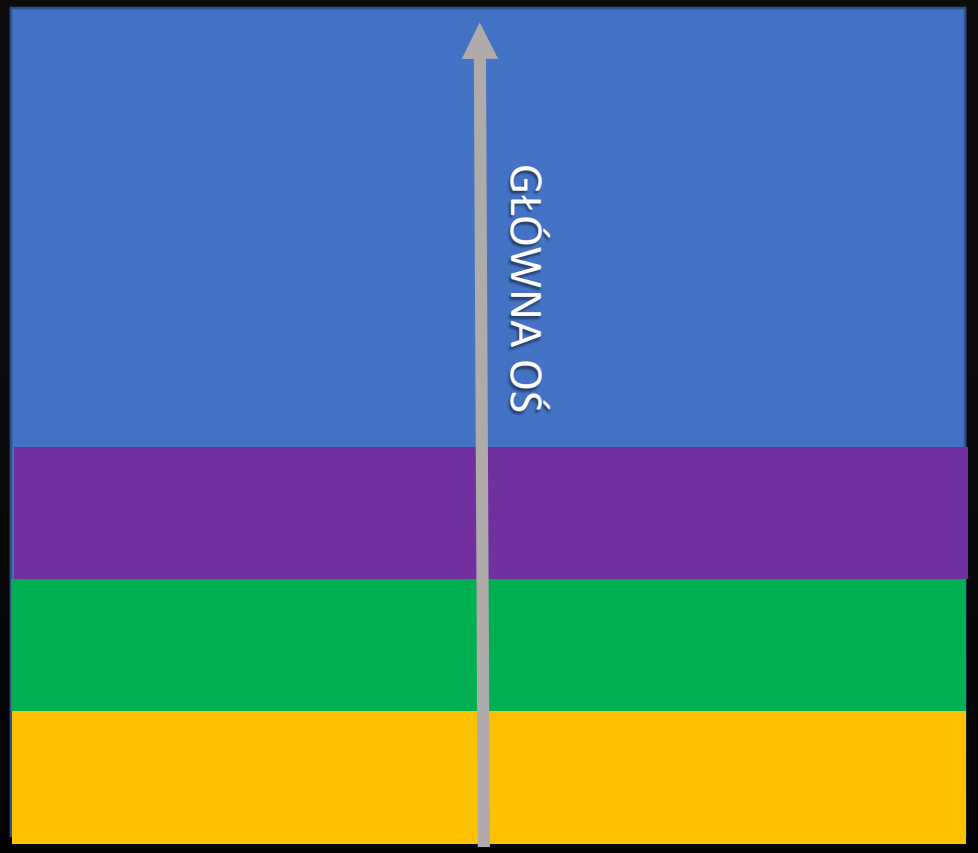


flex-direction

row-reverse



column-reverse

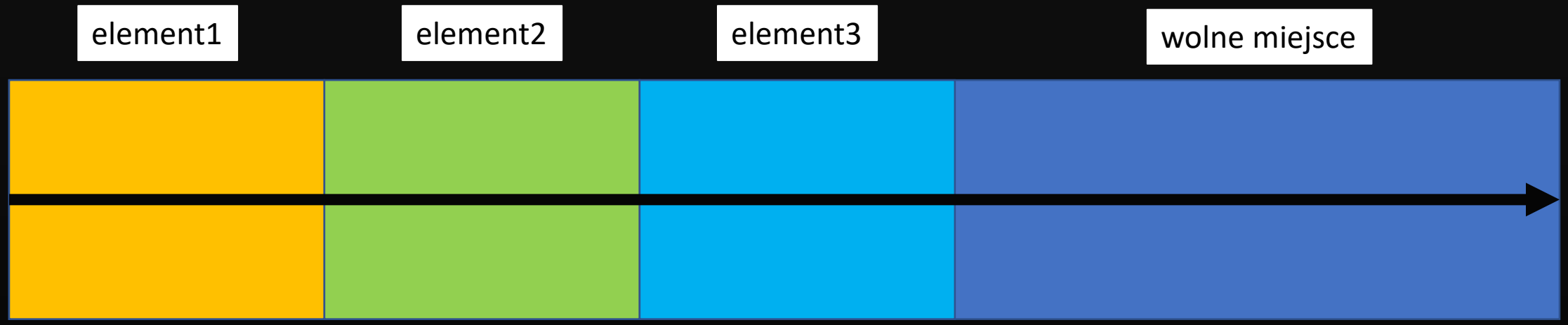


flex-direction

domyślna wartość to row
oś jest wtedy pozioma a kierunek z lewej do prawej.

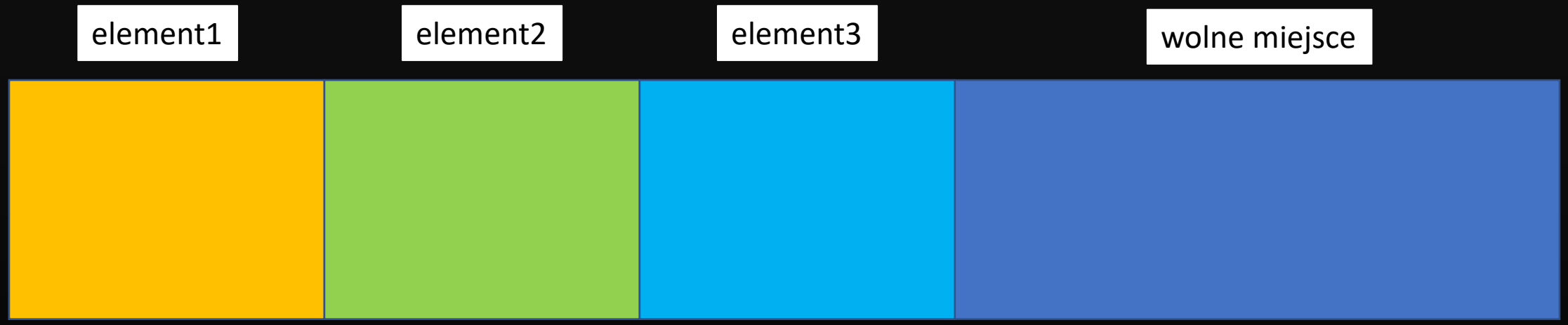
justify-content (układ row)

Tylko wtedy gdy rodzic zajmuje więcej miejsca niż dzieci.



justify-content (row)

Ma znaczenie yylko wtedy gdy rodzic zajmuje więcej miejsca niż dzieci.



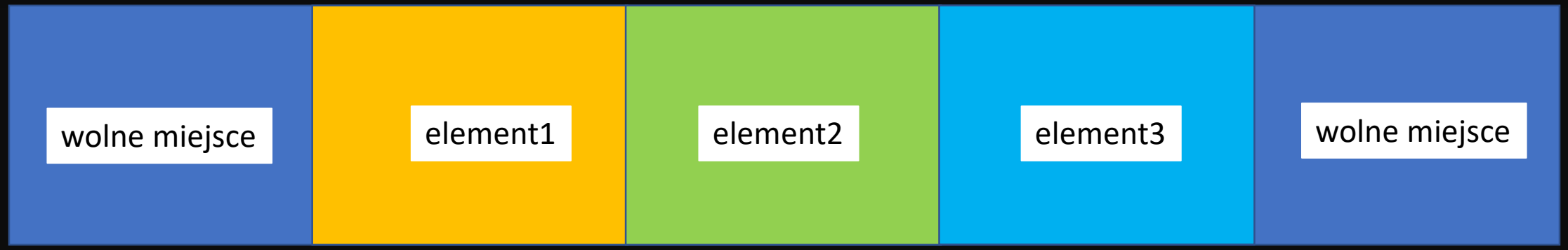
I to jest domyślne ustawienie dla właściwości justify-content

justify-content: flex-start

justify-content (row)

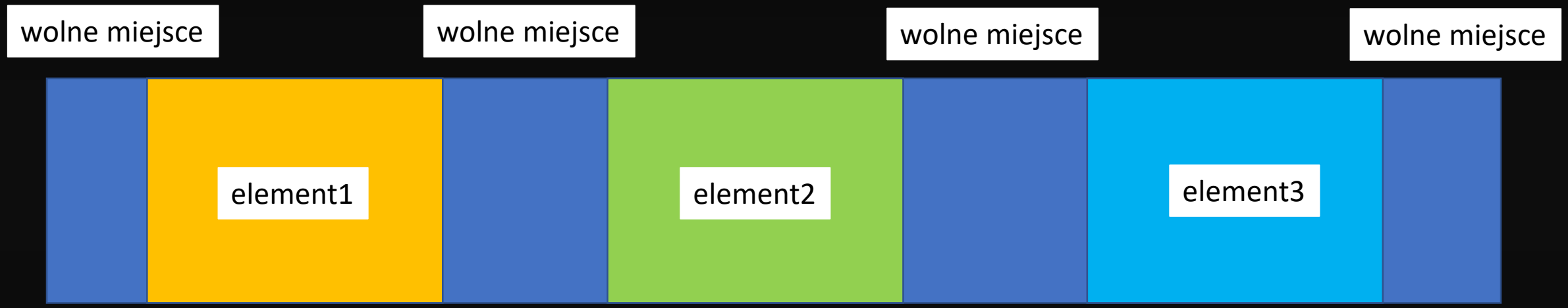
justify-content: flex-end

justify-content (row)



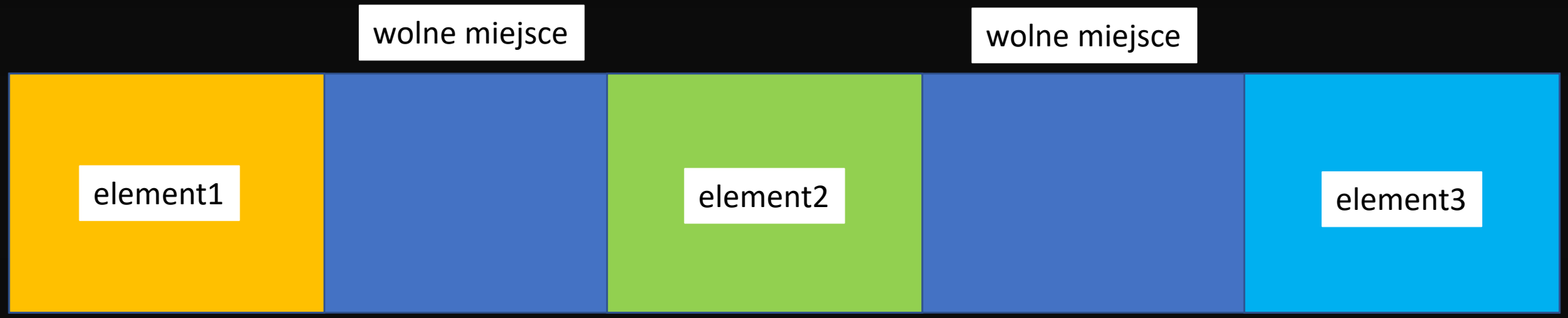
justify-content: center

justify-content (row)



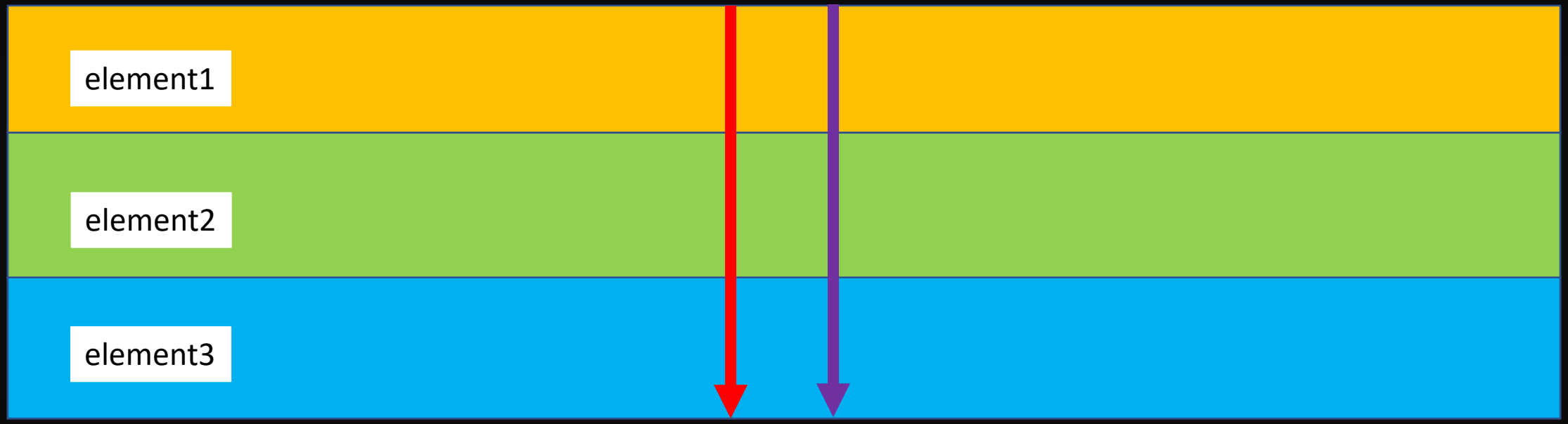
justify-content: space-around

justify-content (row)



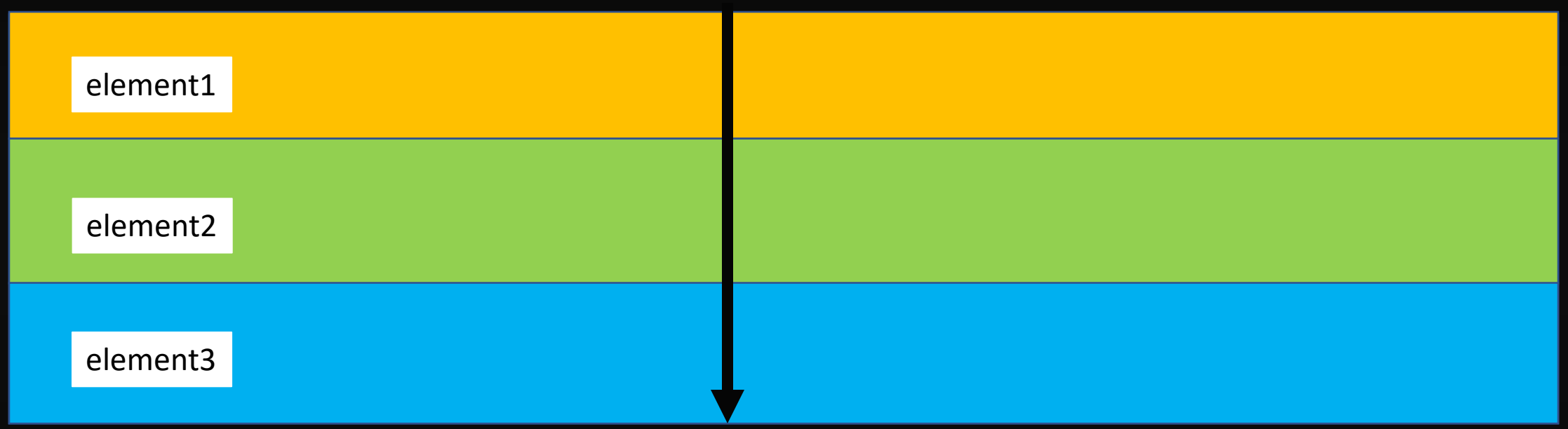
justify-content: space-between

justify-content (układ column)



Pamiętajmy, że justify-content jest właściwością **równoległą do głównej osi**. Czyli w układzie row jest to szerokość ale w układzie row jest to wysokość.

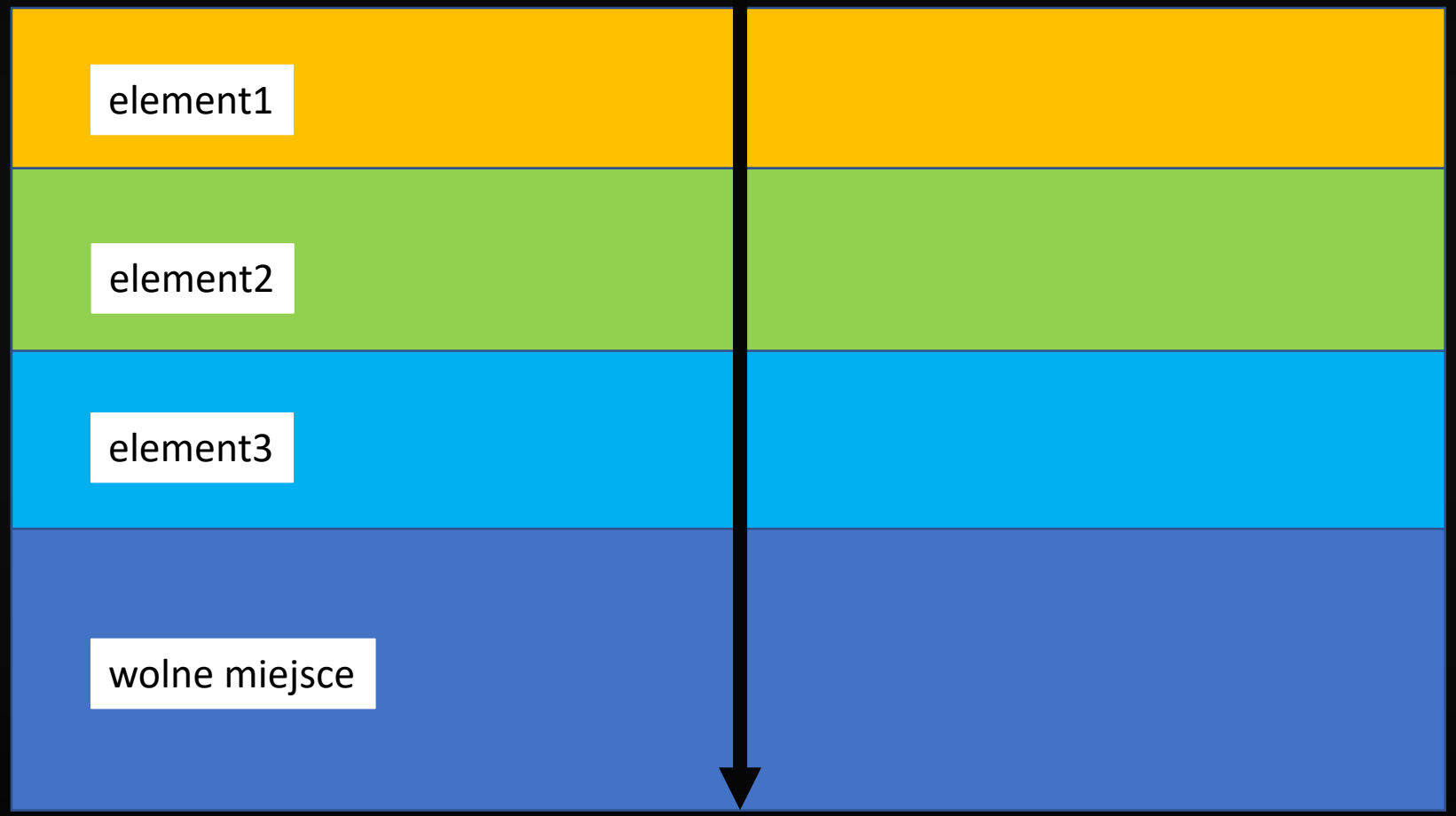
justify-content (układ column)



wolne miejsce? BRAK

Druga sprawa, pamiętaj, że wartość justify-content **dotyczy jedynie wolnej przestrzenie**. Domyślnie kontener (element posiadający display: flex) ma 100% szerokości, ale na wysokość jest to domyślnie "tyle ile potrzebują dzieci"

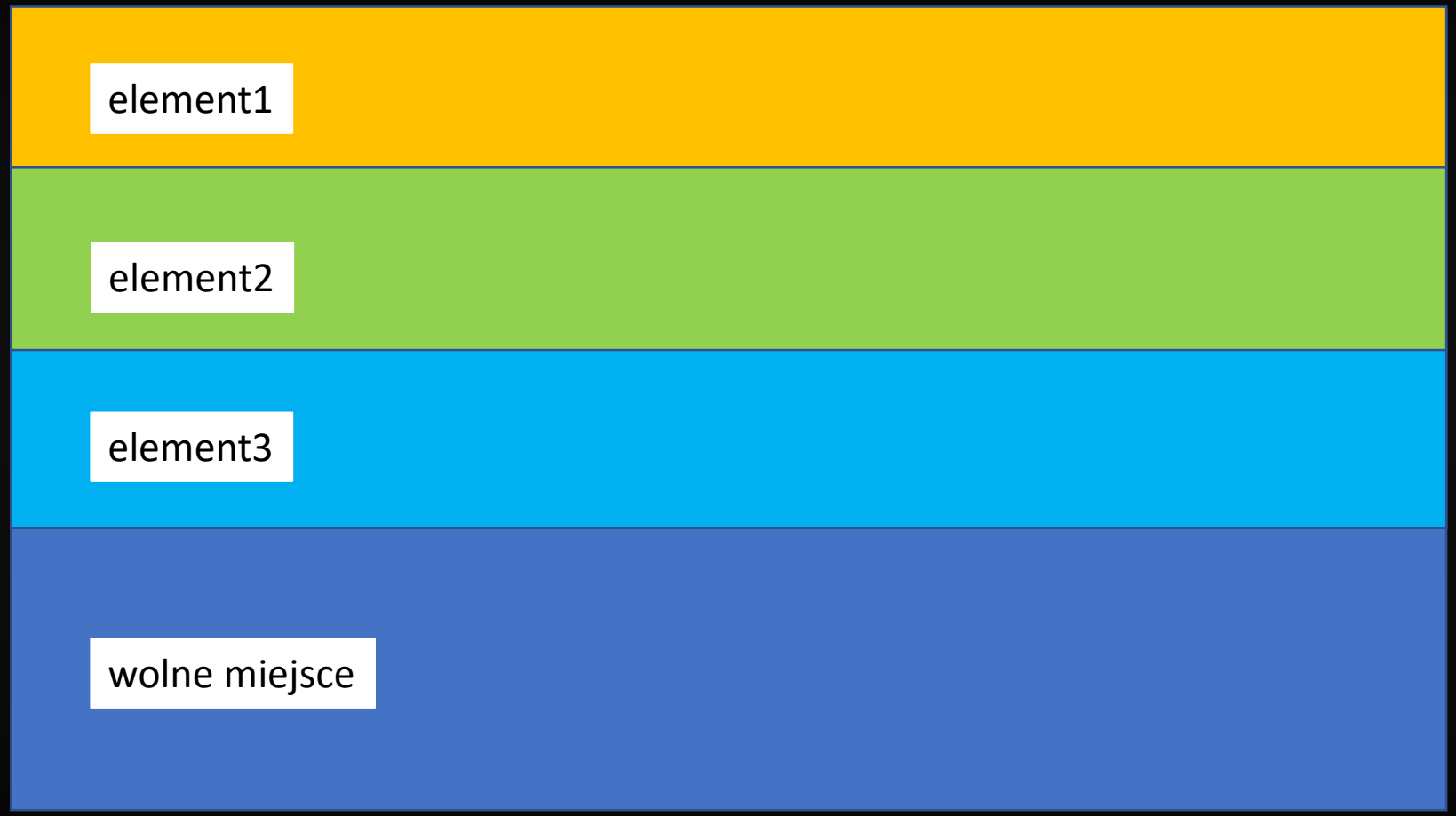
justify-content (układ column)



Dlatego możemy zwiększyć ten obszar

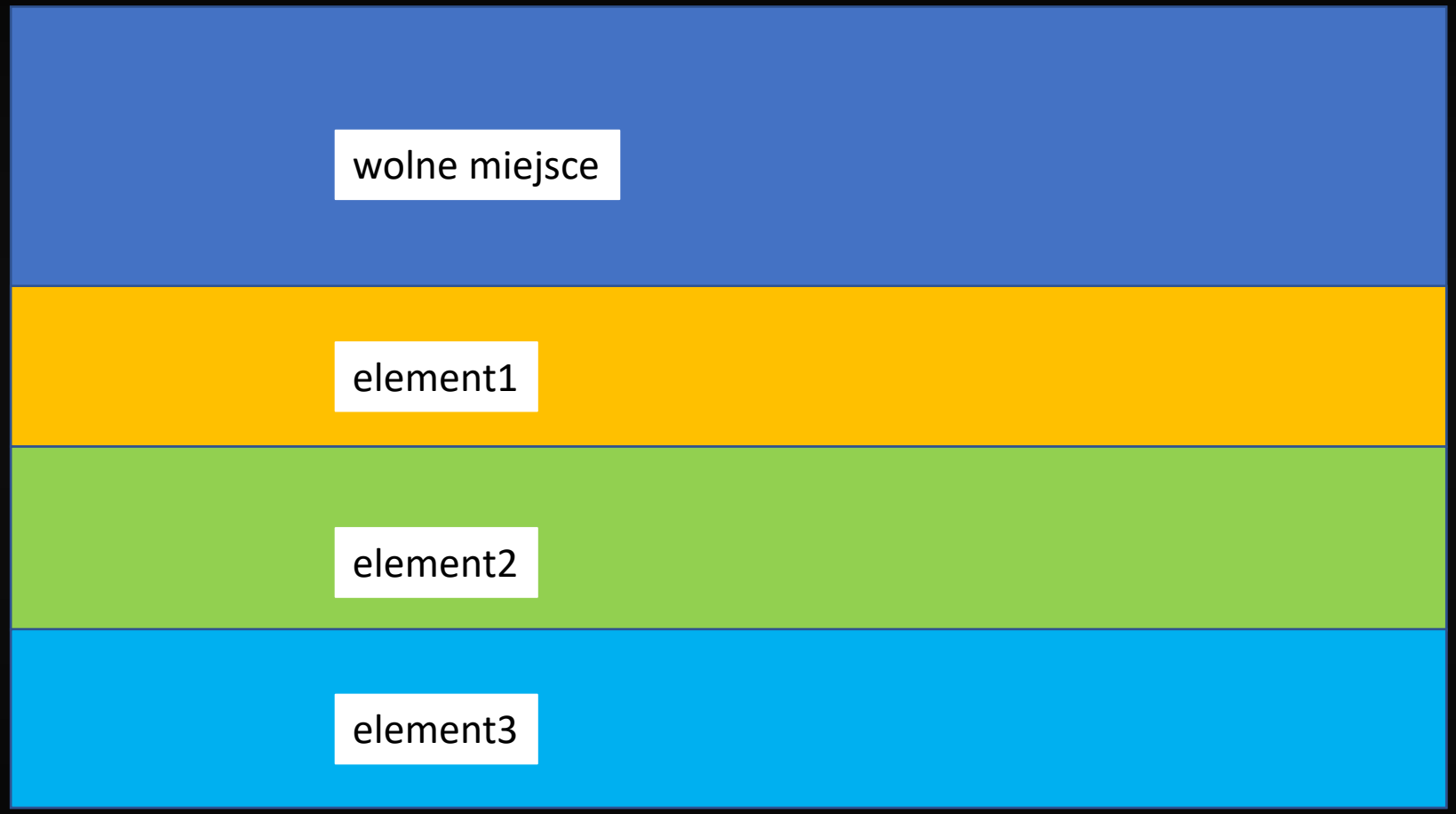
.container {height:50vh}
.element {height:10vh}

justify-content (układ column)



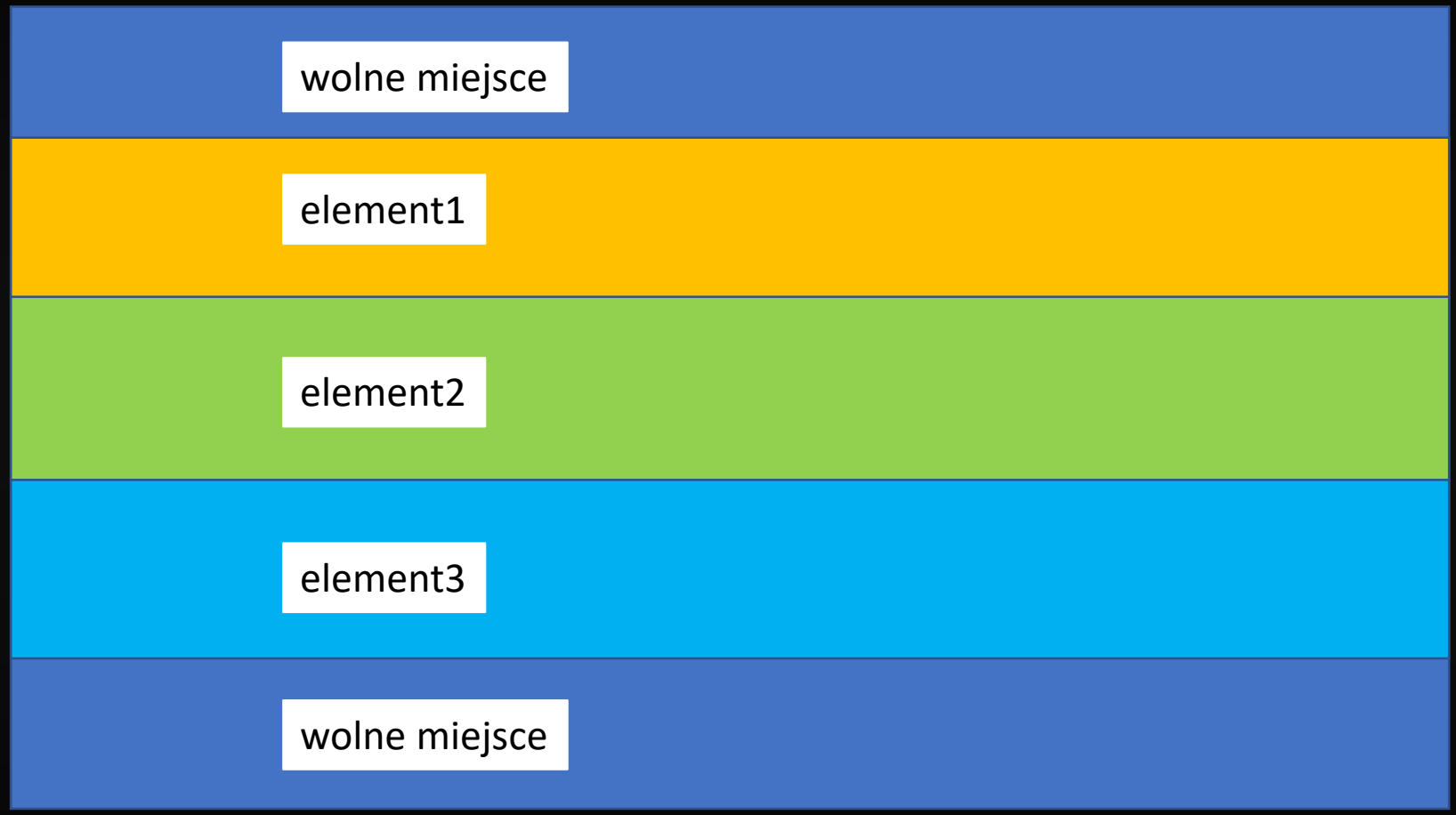
justify-content: flex-start

justify-content (układ column)



justify-content: flex-end

justify-content (układ column)

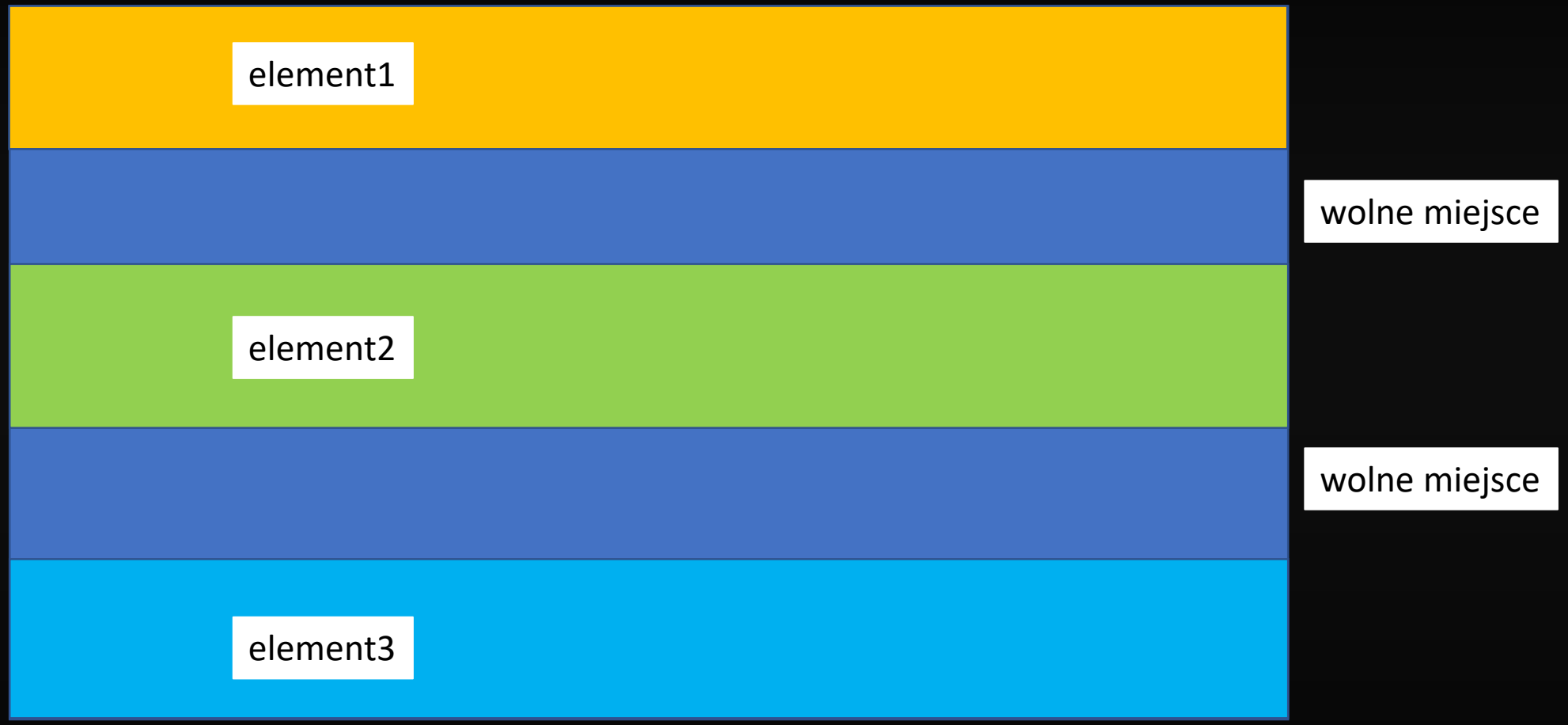


justify-content: center

justify-content (układ column)

justify-content: space-around

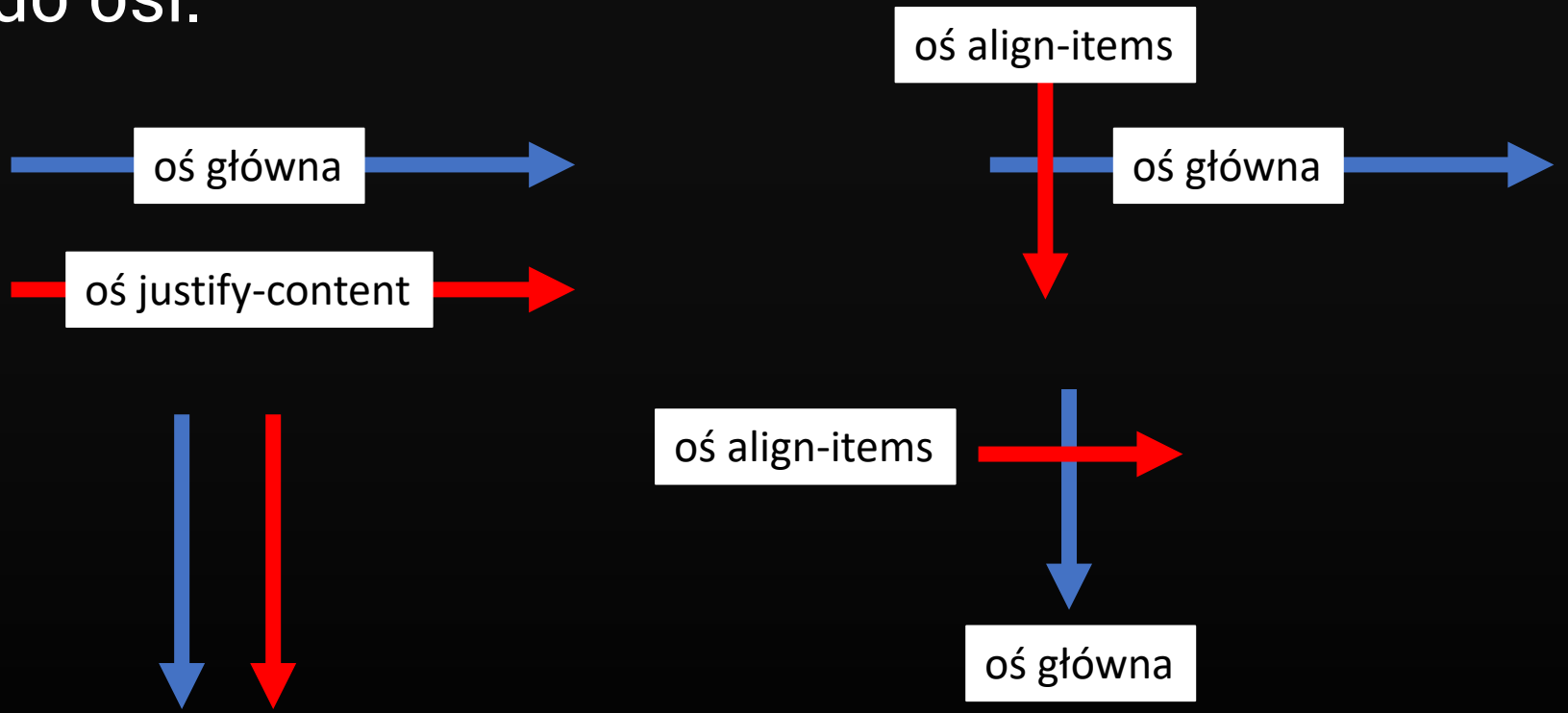
justify-content (column)



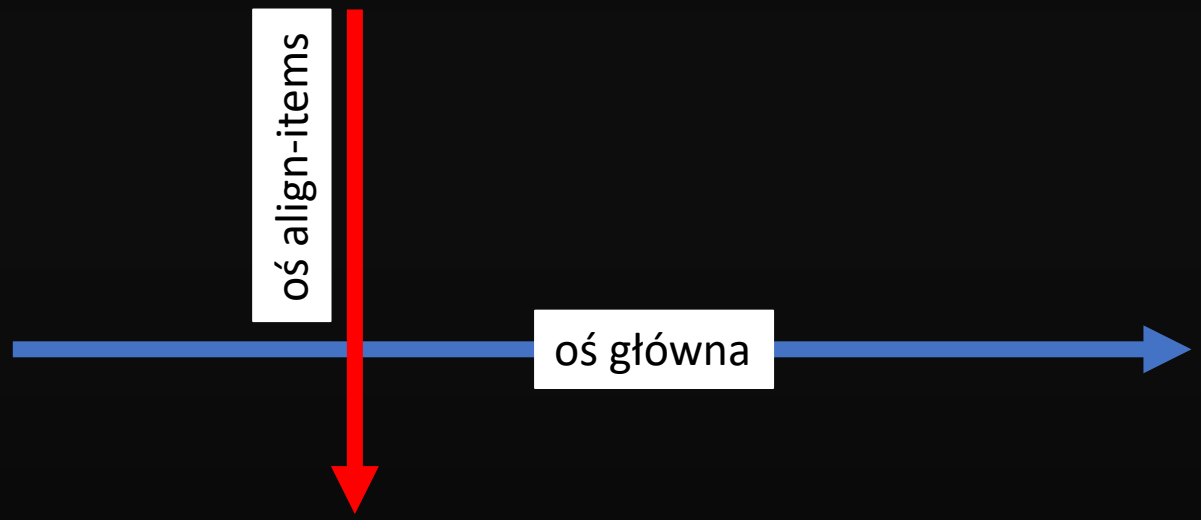
justify-content: space-between

align-items

W przeciwieństwie do justify-content który tworzy zachowanie równoległe do osi, align-items określa zachowanie prostopadłe do osi.

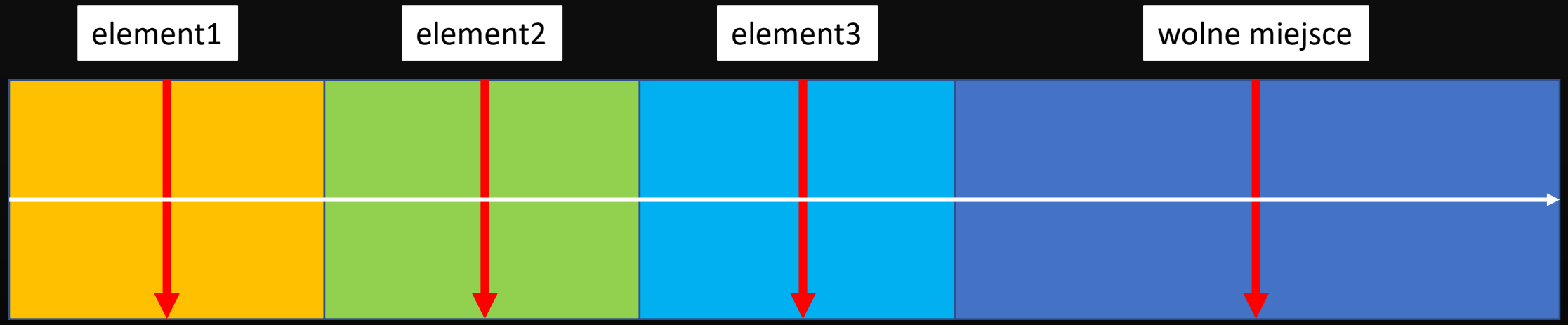


align-items (układ row)



align-items (row)

align-items przyjmuje 5 wartości: stretch (domyślna) oraz center, flex-end, flex-start, baseline;

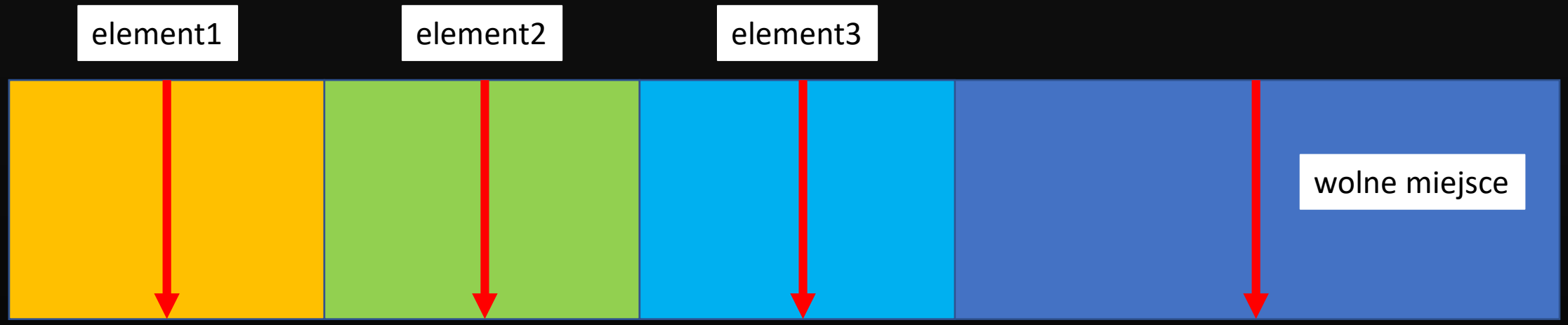


align-items **wpływa na każdy element elastyczny** (dziecko flex) i określa zachowanie dziecka.

align-items (row)

```
container {
        height: 200px;
}

.element {
        }
```

align-items: stretch

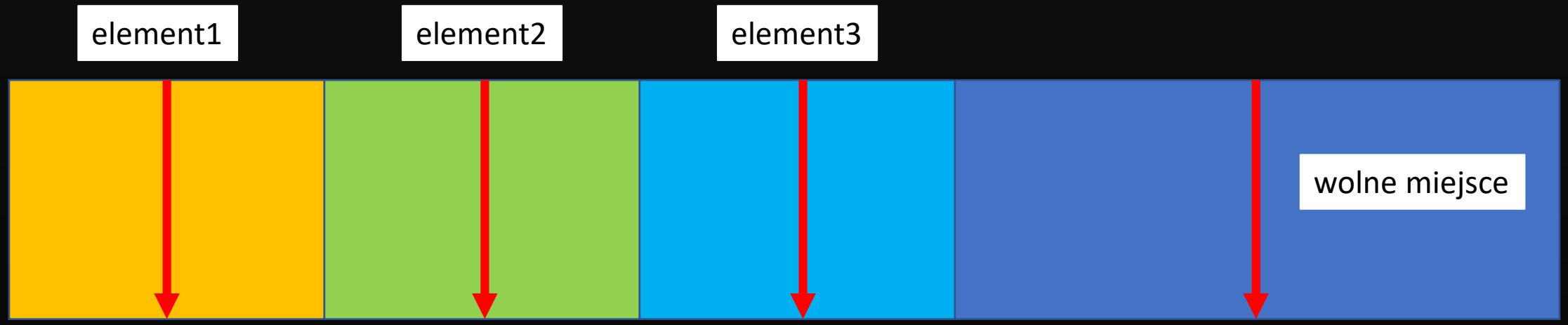


Przyjmują domyślnie wielkość rodzica.

align-items (row)

```
container {
}

.element2 {
        height: 200px
        }
```

align-items: stretch

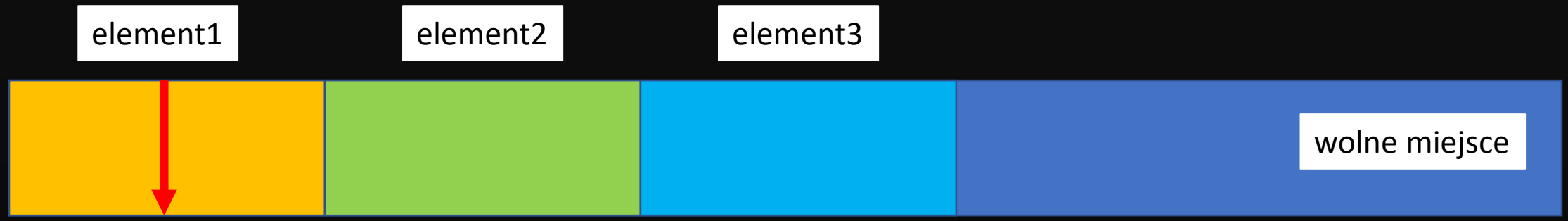


Przyjmują domyślnie wielkość rodzica. Wielkosc rodzica może być też wyznaczana przez największy element.

align-items (row)

```
container {
}

.element1 {
        height: 100px;
}
```

align-items: stretch

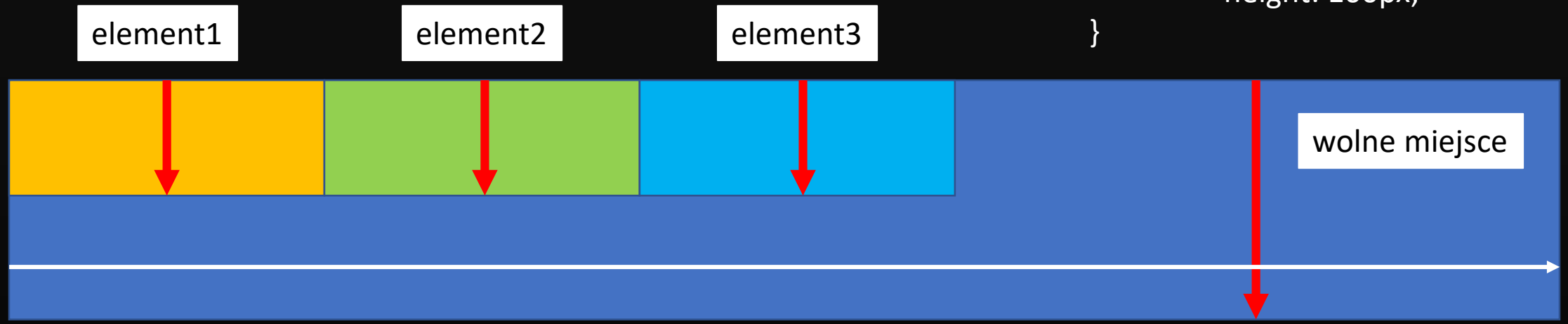


Wszystkie elementy przyjmują wartość (w tym wypadku wysokość) największego elementu.

align-items (row)

align-items: flex-start

```
container {
        height: 200px;
        align-items: flex-start
}

.element {
        height: 100px;
}
```

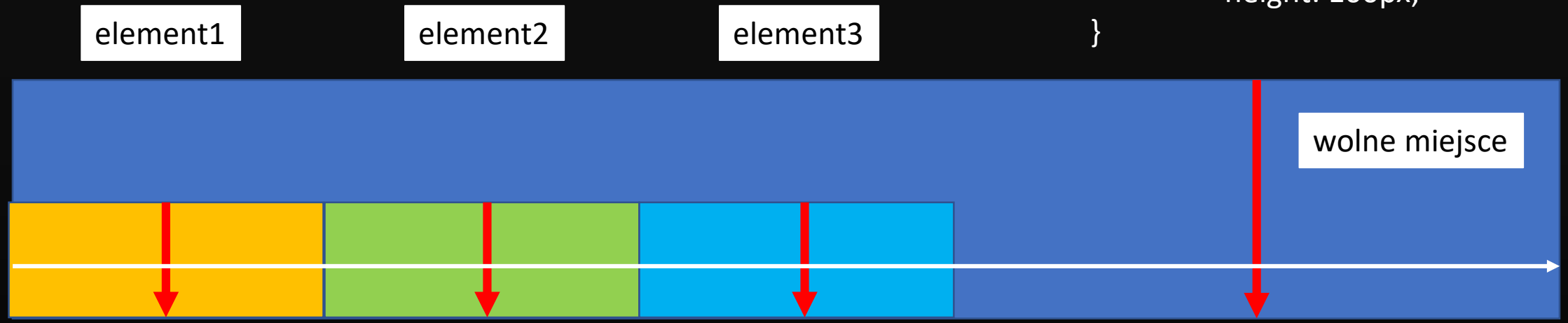


Układa się na początku linii prostopadłej względem osi głównej.

align-items (row)

align-items: flex-end

```
container {
        height: 200px;
        align-items: flex-end;
}

.element {
        height: 100px;
}
```

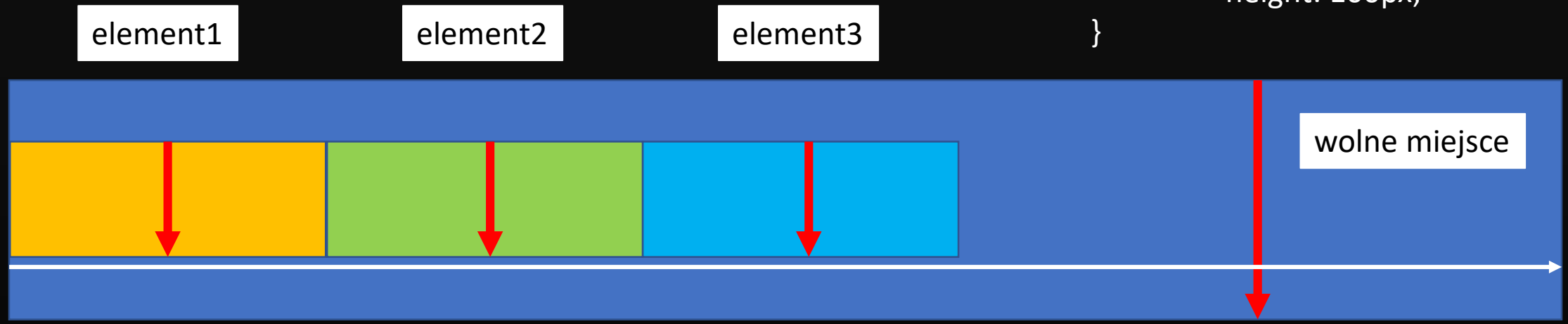


Układa się na końcu linii prostopadłej względem osi głównej.

align-items (row)

align-items: center

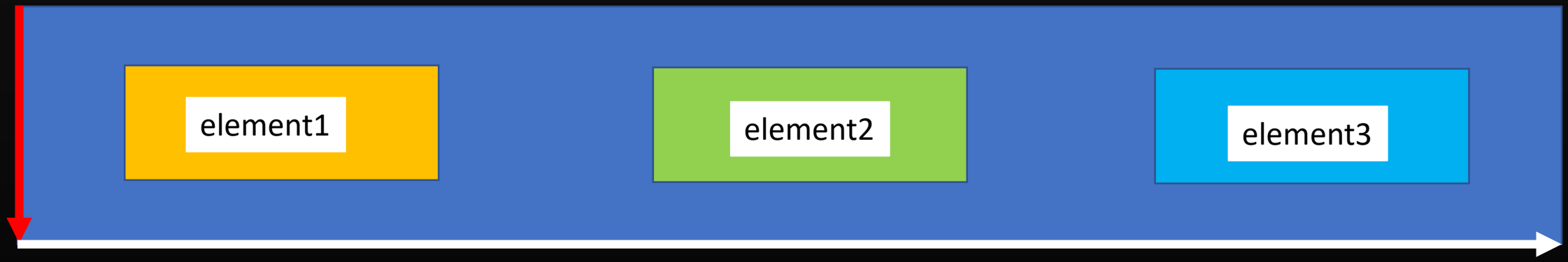
```
container {
        height: 200px;
        align-items: center;
}

.element {
        height: 100px;
}
```



Układa się pośrodku linii prostopadłej względem osi głównej.

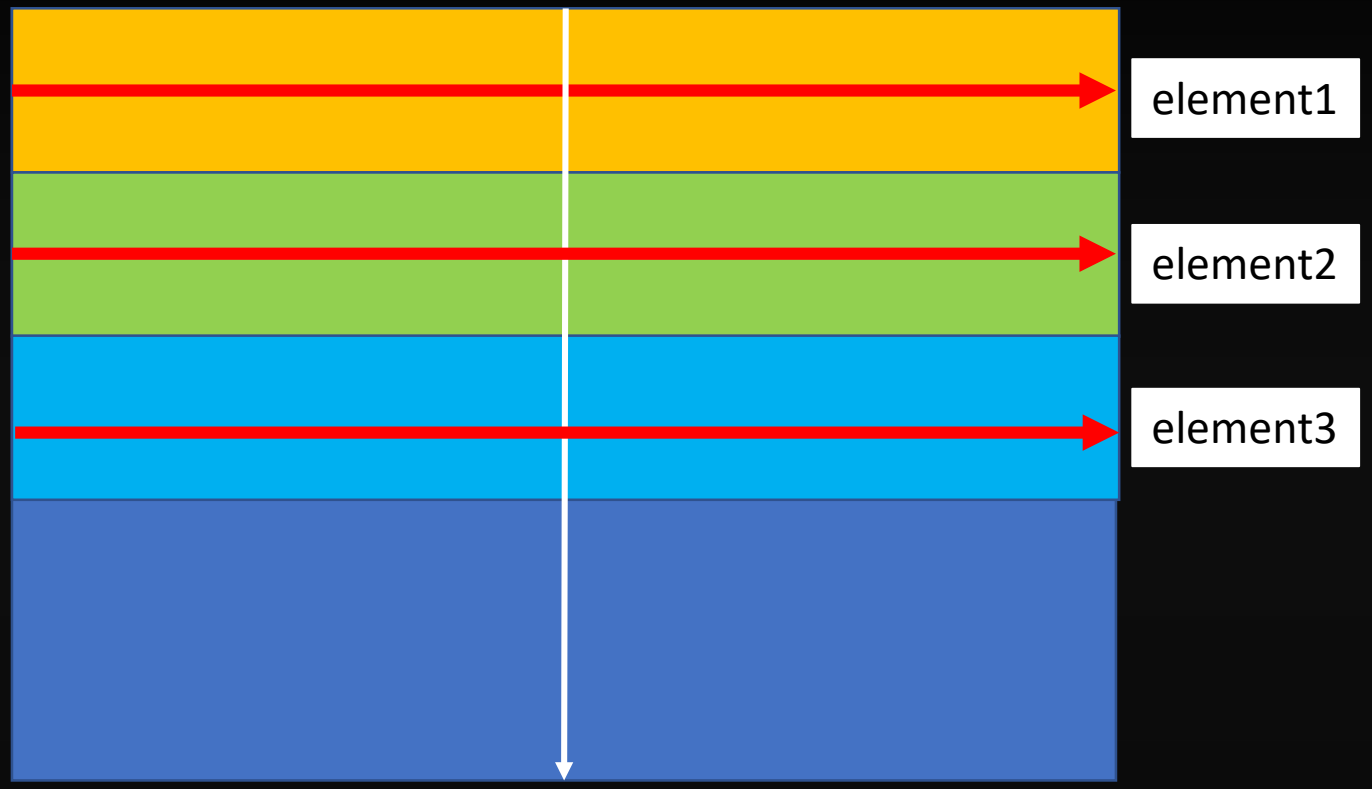
align-items (row) i justify-content (row)

```
container {
        height: 200px;
        align-items: center;
        justify-content: space-around
}
.element {
        height: 100px;
}
```



Układa się pośrodku linii prostopadłej względem osi głównej (align-items: center).
Zostawia miejsce dookoła (równe) pomiędzy elementami na osi główne (justify-content: space-around)

align-items (column)

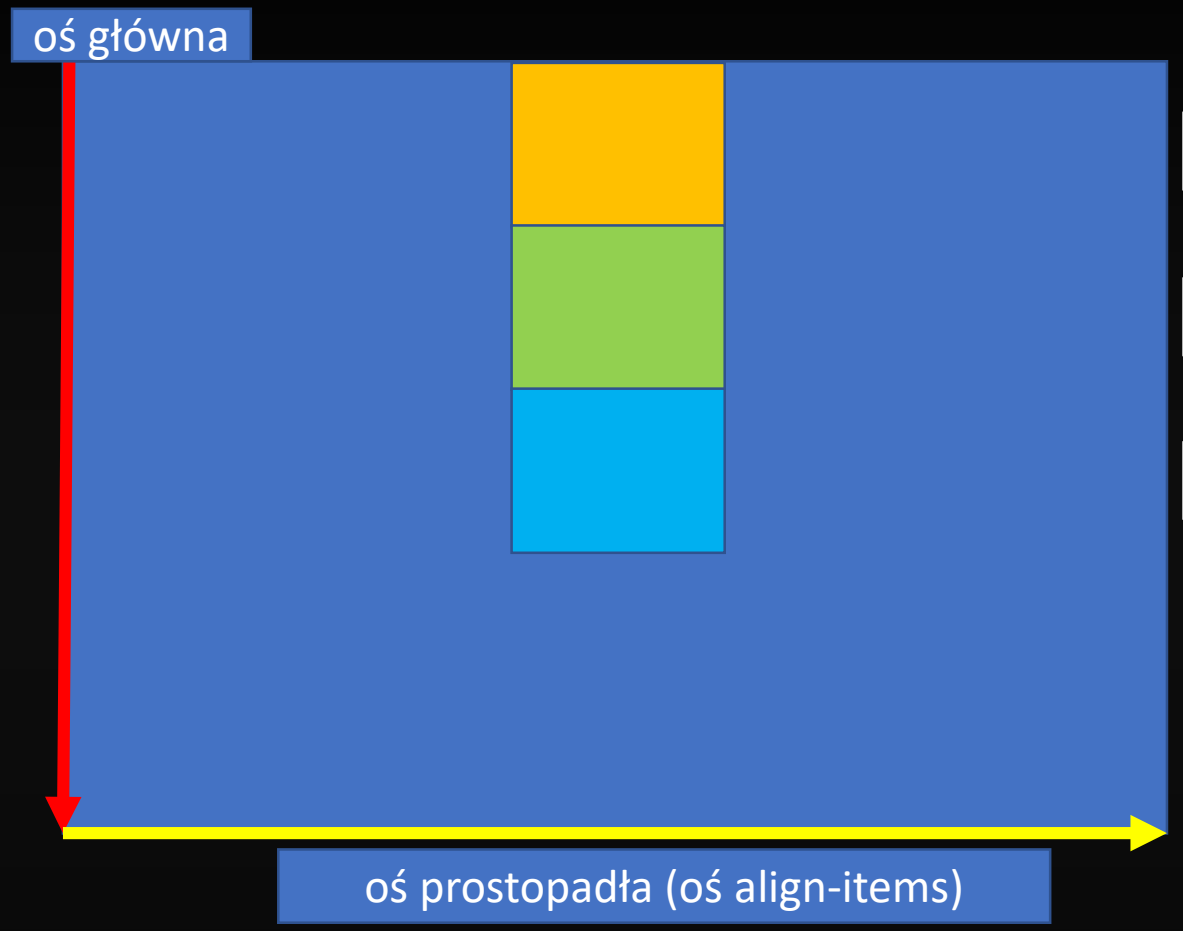



```
container {
        display: flex;
        flex-direction: column;

        align-items: stretch;
        height: 50vh;
}

.element {
        /* wysokość */
        flex-basis:100px;
        }
```

align-items: stretch

Stretch jest domyślną wartością, dlatego wszystkie elementy mają (domyślnie) 100% na osi prostopadłej do głownej przy układzie column

align-items (column)

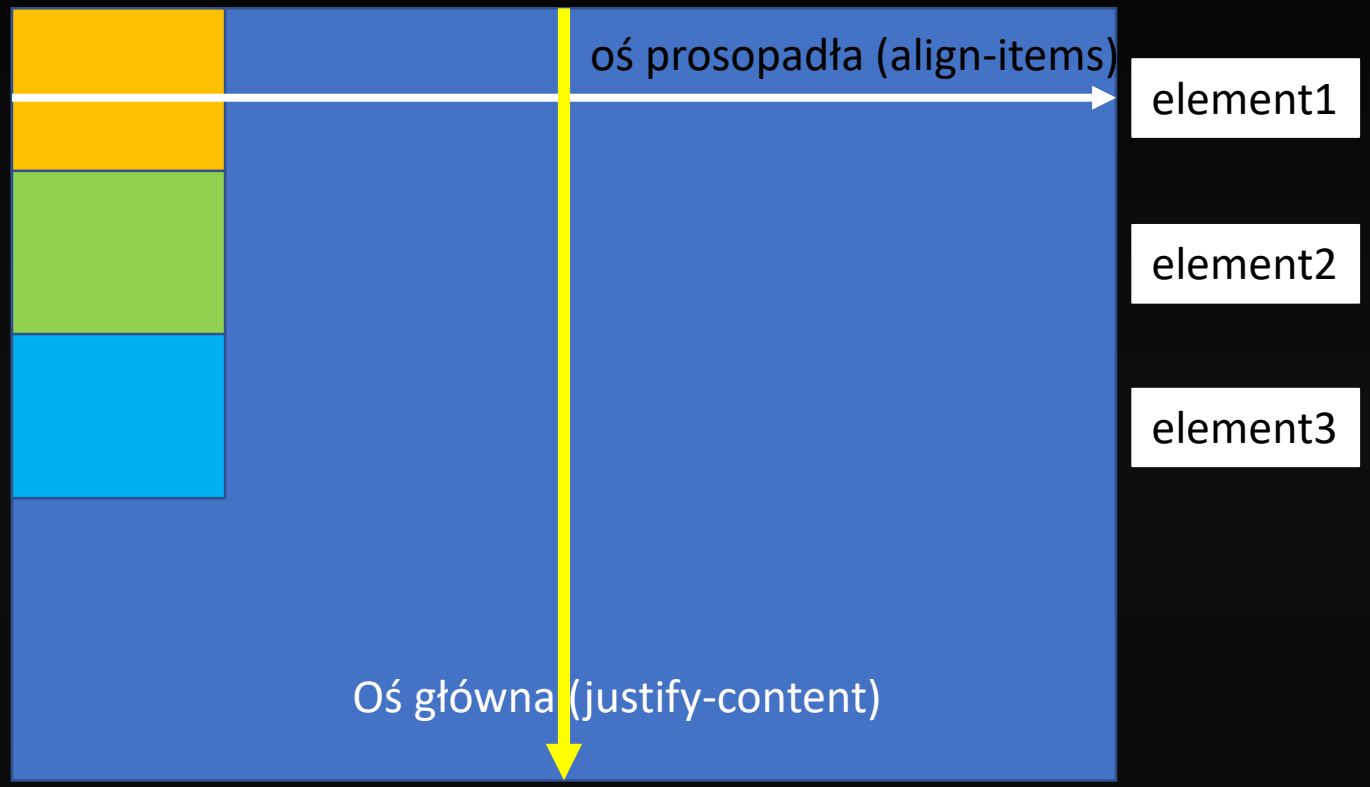



```
container {
        display: flex;
        flex-direction: column;
        align-items: center;
        height: 50vh;
}

.element {
        flex-basis:100px;
        width:100px
        }
```

align-items: center

Center - Przede wszystkim każda inna wartość (center, flex-start, flex-end, baseline) nie rozciąga już elementów na całą długość (w osi prostopadłej do głównej). Zajmują one tyle miejsca ile potrzebują (lub ile określimy, w tym przypadku 100px).

align-items (column)

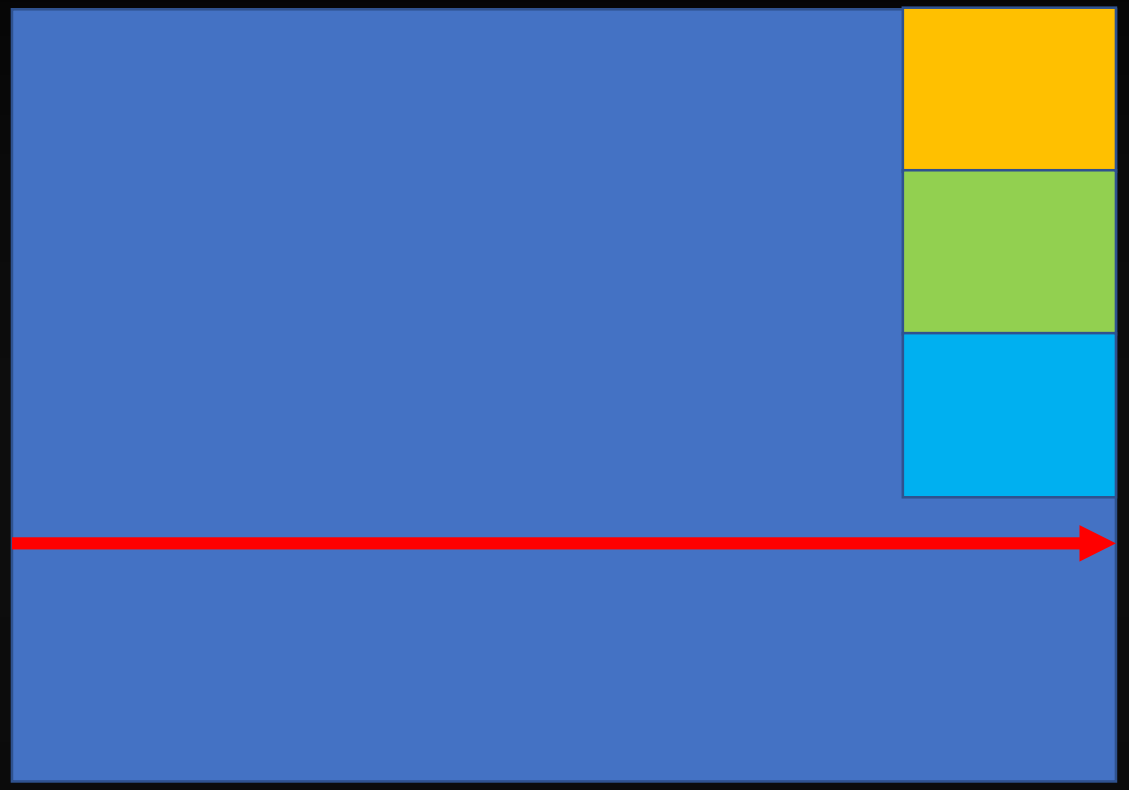



```
container {
        display: flex;
        flex-direction: column;
        align-items: flex-start;
        height: 50vh;
}

.element {
        flex-basis:100px;
        width:100px
        }
```

align-items: flex-start

align-items: flex-start. Ustawia WSZYSTKIE elementy do początku osi prostopadłej.

align-items (column)

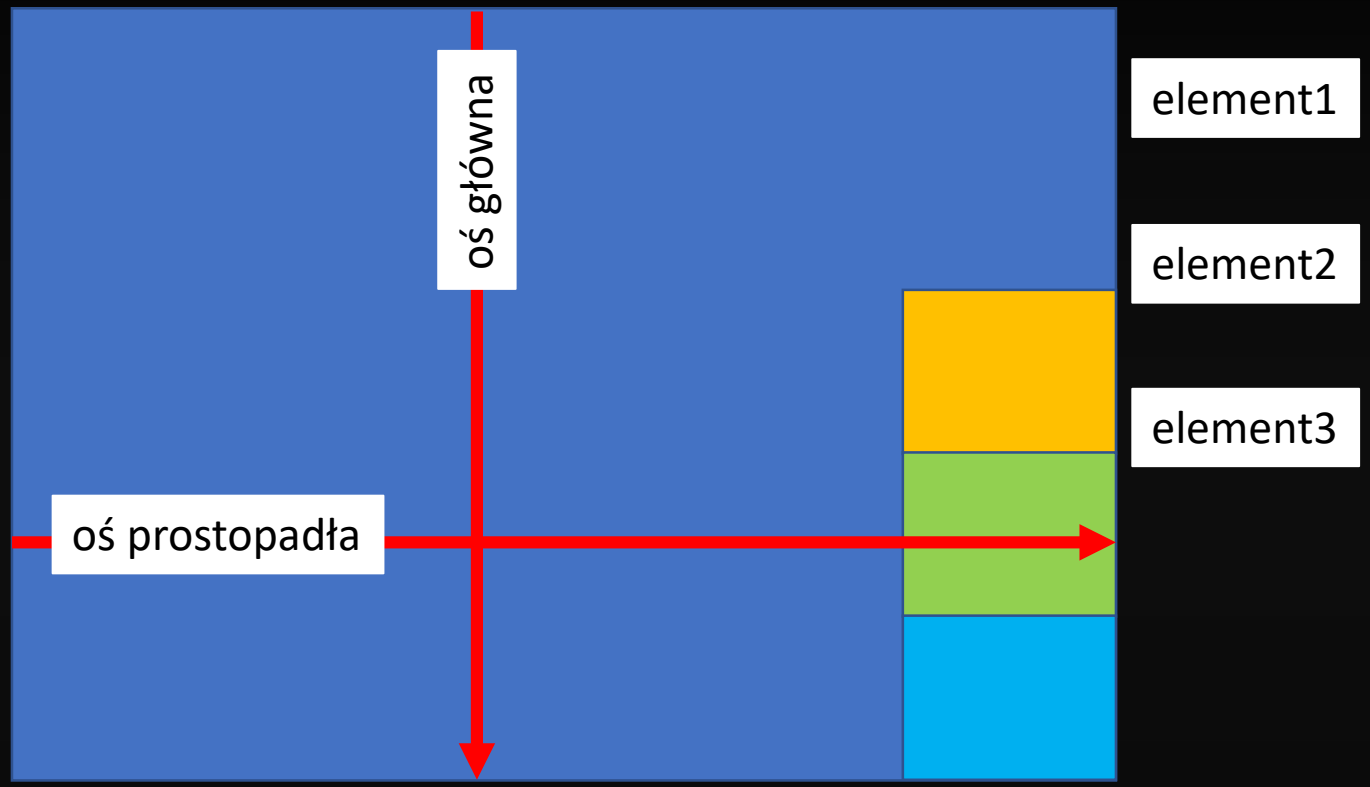



```
container {
        align-items: flex-end;
        height: 50vh;
}

.element {
        flex-basis:100px;
        width:100px
        }
```

align-items: flex-end

Align-items: flex-end. Ustawia WSZYSTKIE elementy na końcu osi prostopadłej.

align-items (column)



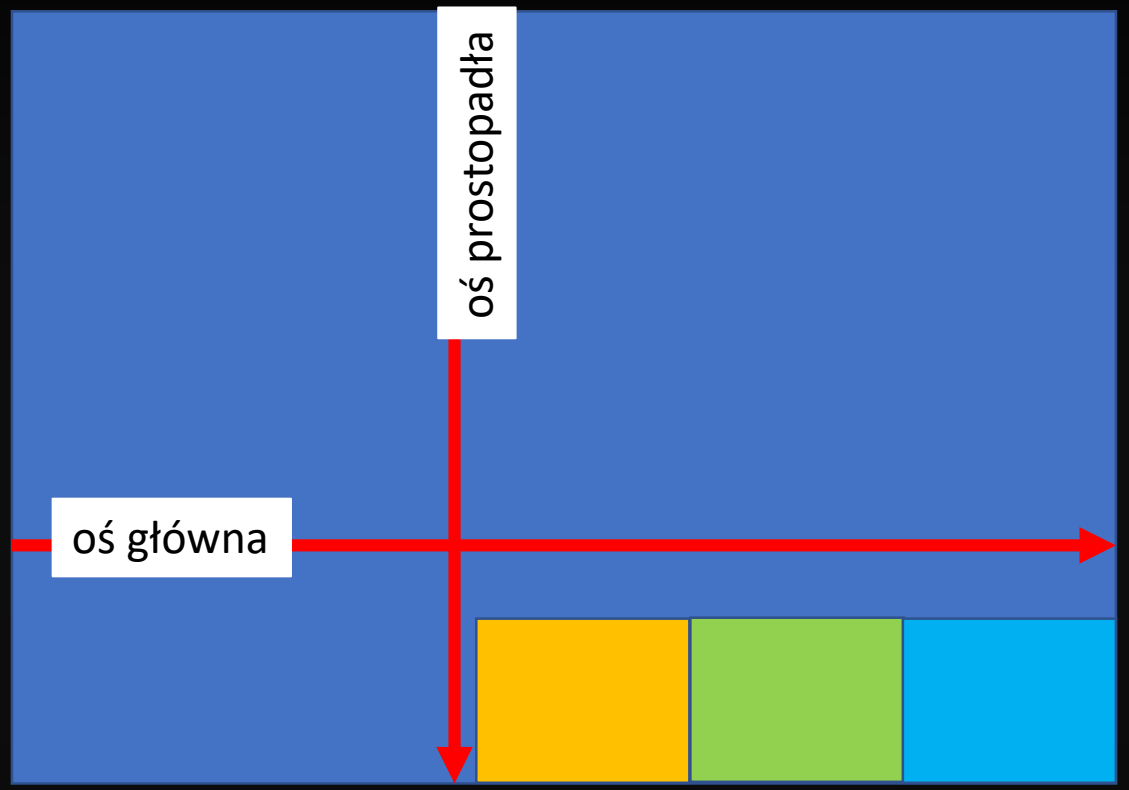

```
container {
        display: flex;
        flex-direction: column;

        align-items: flex-end;
        justify-content: flex-end;
        height: 50vh;
}
.element {
        flex-basis:100px;
        width:100px
        }
```

Align-items: flex-end. Ustawia WSZYSTKIE elementy na końcu osi prostopadłej.
justify-content: flex-end. Ustawia WSZYSTKIE elementy na końcu osi głównej

align-items: flex-end

justify-content: flex-end

align-items (row)




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

        align-items: flex-end;
        justify-content: flex-end;
        height: 50vh;
}
.element {
        flex-basis:100px;
        width:100px
        }
```

Align-items: flex-end. Ustawia WSZYSTKIE elementy na końcu osi prostopadłej.
Justify-content: flex-end. Ustawia WSZYSTKIE elementy na końcu osi głównej

align-items: flex-end

justify-content: flex-end

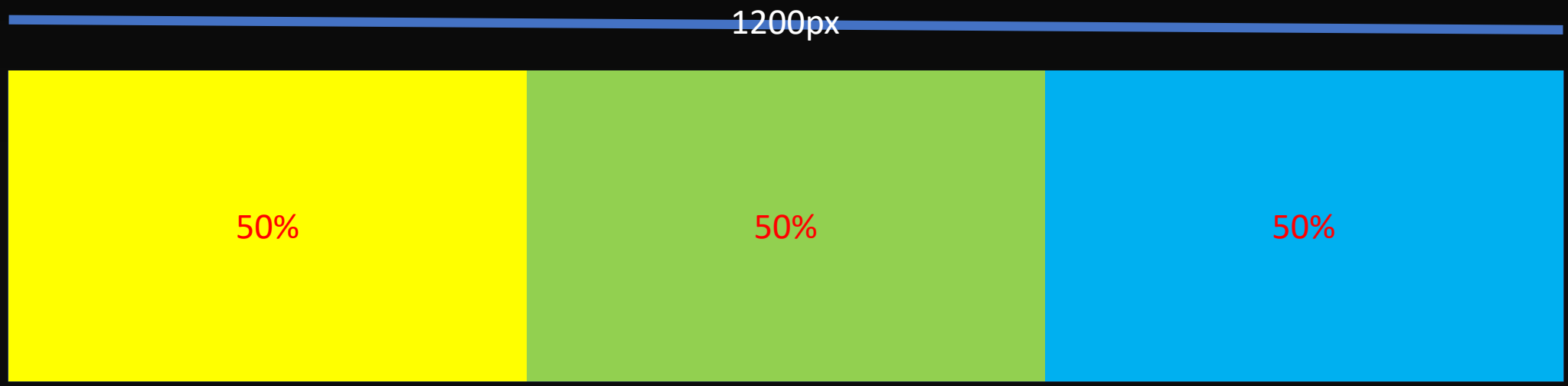
flex-wrap

Czyli jeśli się nie mieścisz w linii, zrób sobie kolejną albo upchaj wszystko.

Domyślnie elementy są zmienjszane co wynika z właściwości flex-shrink, która będzie jeszcze omówiona (flex-shrink:1 - domyślna)

flex-wrap ma swoją domyślną wartość, która nie pozwala przejść do kolejnej linii. flex-wrap: nowrap;

flex-wrap (nowrap)

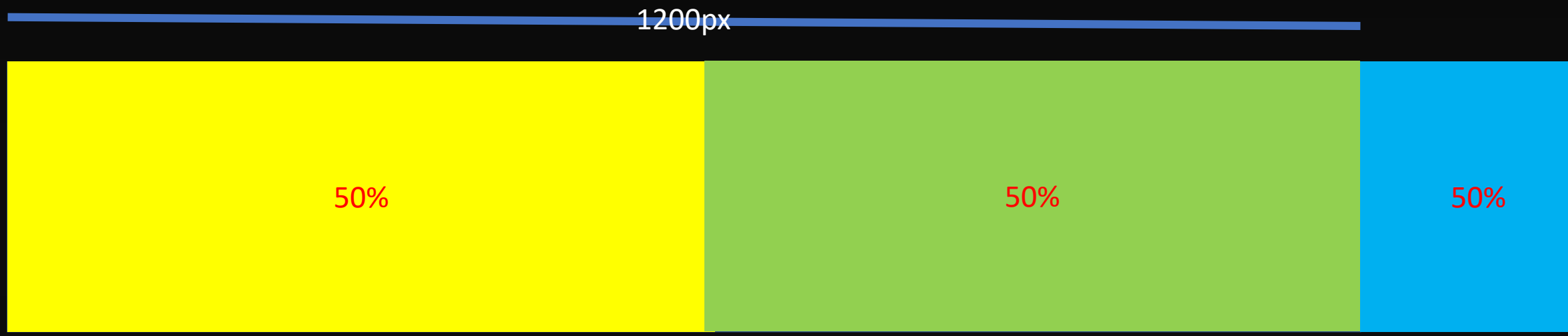


Każdy element ma mieć 600px (50% rodzica), ale w praktyce w tym przykładzie każdy element ma 400px, ponieważ domyślnie jeśli elementy się nie mieszczą w kontenerze, to są zmiejszane.

```
container {
        display: flex;
        width:1200px;
        flex-wrap: nowrap //domyślne
}
```

```
.element {
        width: 50%
        flex-shrink: 1 //domyślne
}
```

flex-wrap (nowrap)

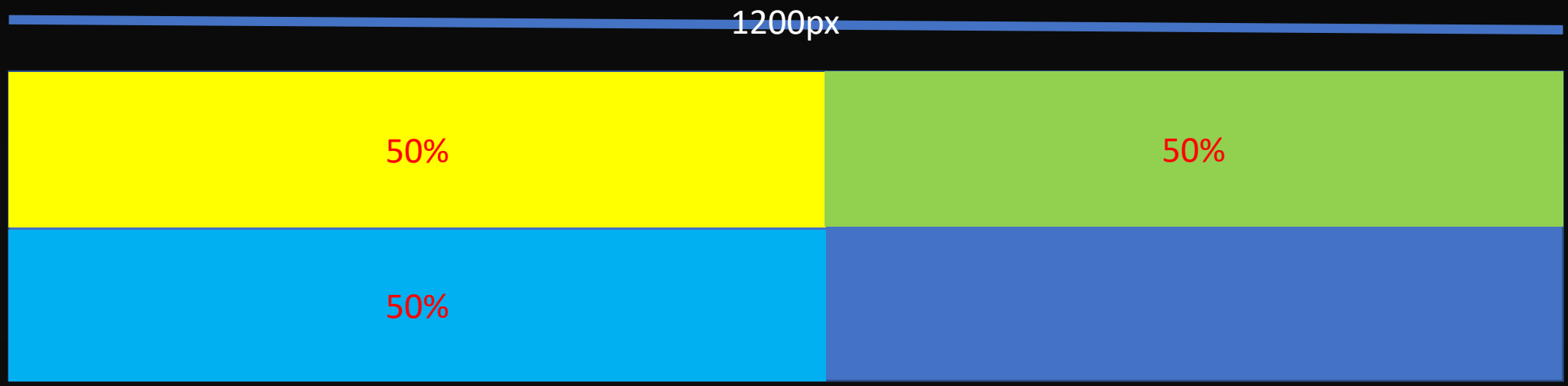


Elementy wyjdą jeśli nie pozwolimy im się zmniejszać.

```
container {
        display: flex;
        width:1200px;
        flex-wrap: nowrap //domyślne
}
```

```
.element {
        width: 50%
        flex-shrink: 0 //zmiana
}
```

flex-wrap (wrap)

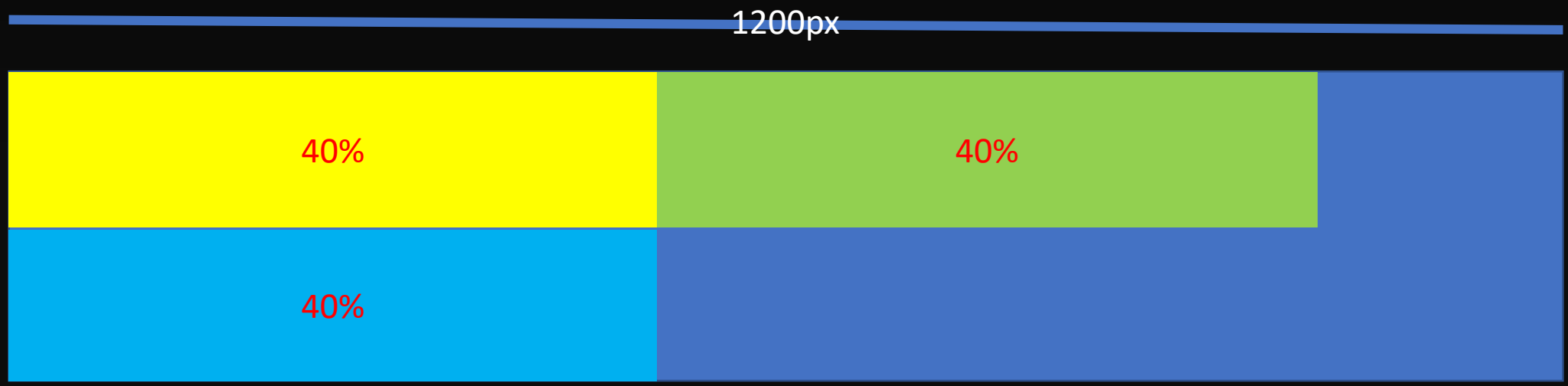


Każdy element ma mieć 50% rodzica (w praktyce 600px) Ponieważ się nie mieszczą, pierwszy, który się nie mieści, idzie do nowej linii

```
container {
        display: flex;
        width:1200px;
        flex-wrap: wrap //ZMIENIAMY
}
```

```
.element {
        width: 50%
        flex-shrink: 1 //domyślne
}
```

flex-wrap (wrap)

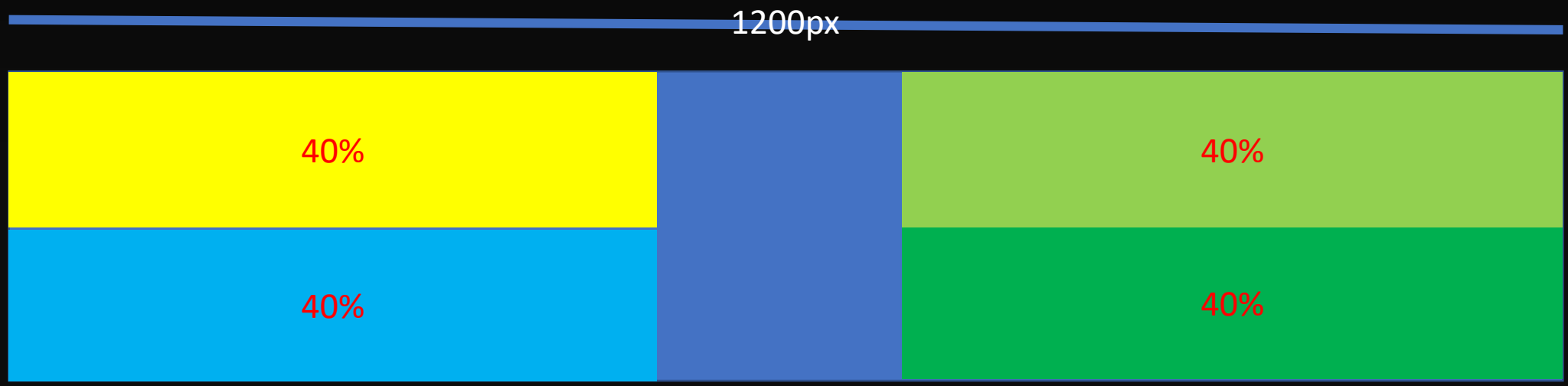


Każdy element ma mieć 40% rodzica. Ponieważ się nie mieszczą, pierwszy, który się nie mieści, idzie do nowej linii

```
container {
        display: flex;
        width:1200px;
        flex-wrap: wrap //ZMIENIAMY
}
```

```
.element {
        width: 40%
        flex-shrink: 1 //domyślne
}
```

flex-wrap (wrap)




```
container {
        display: flex;
        width:1200px;
        flex-wrap: wrap;
        justify-content: space-between;

}
```

```
.element {
        width: 40%
}
```

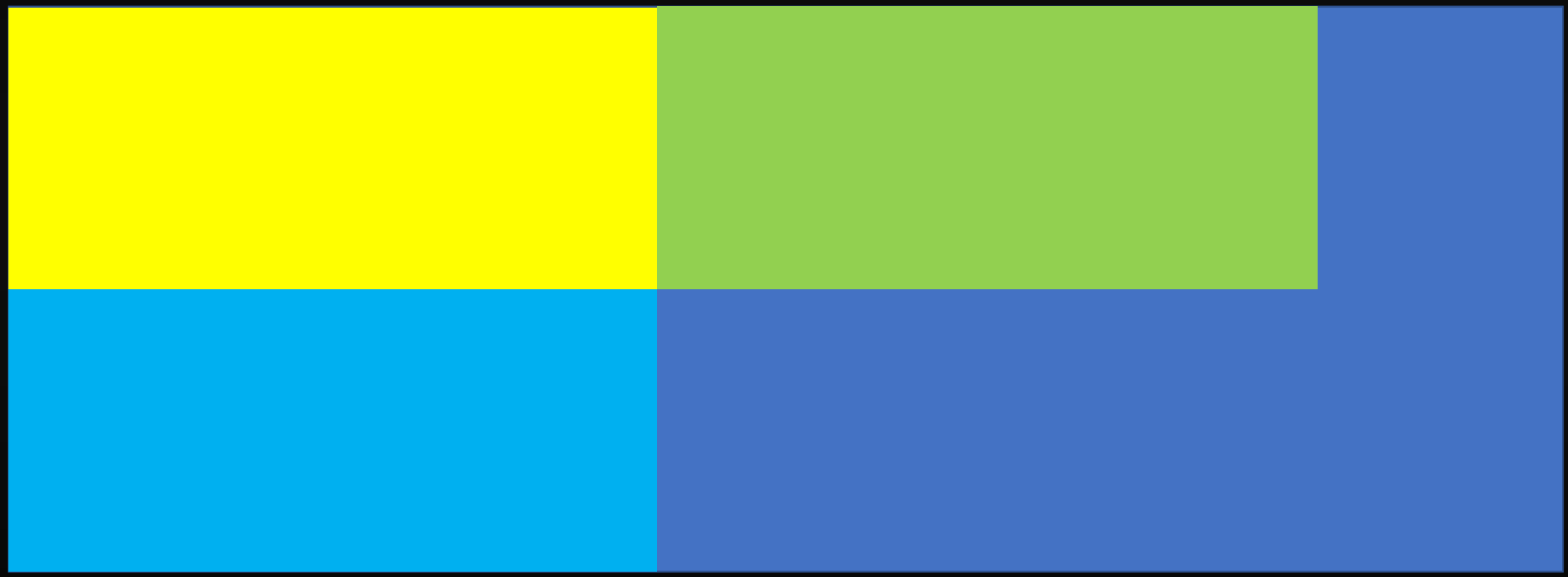
align-content

Gdy już mamy więcej niż jedną linie możemy tworzyć sposób zachowania między nimi.

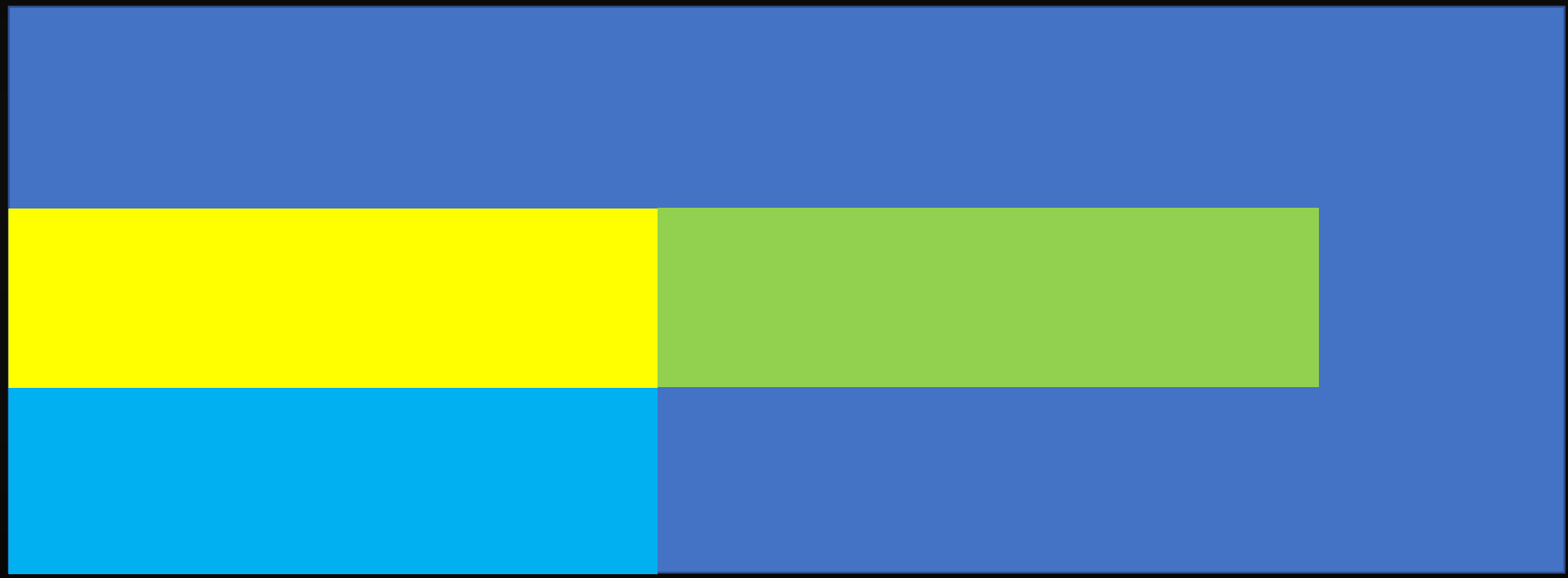
center; flex-end; flex-start; space-around; space-between; stretch.

Stretch jest ustawieniem domyślnym.

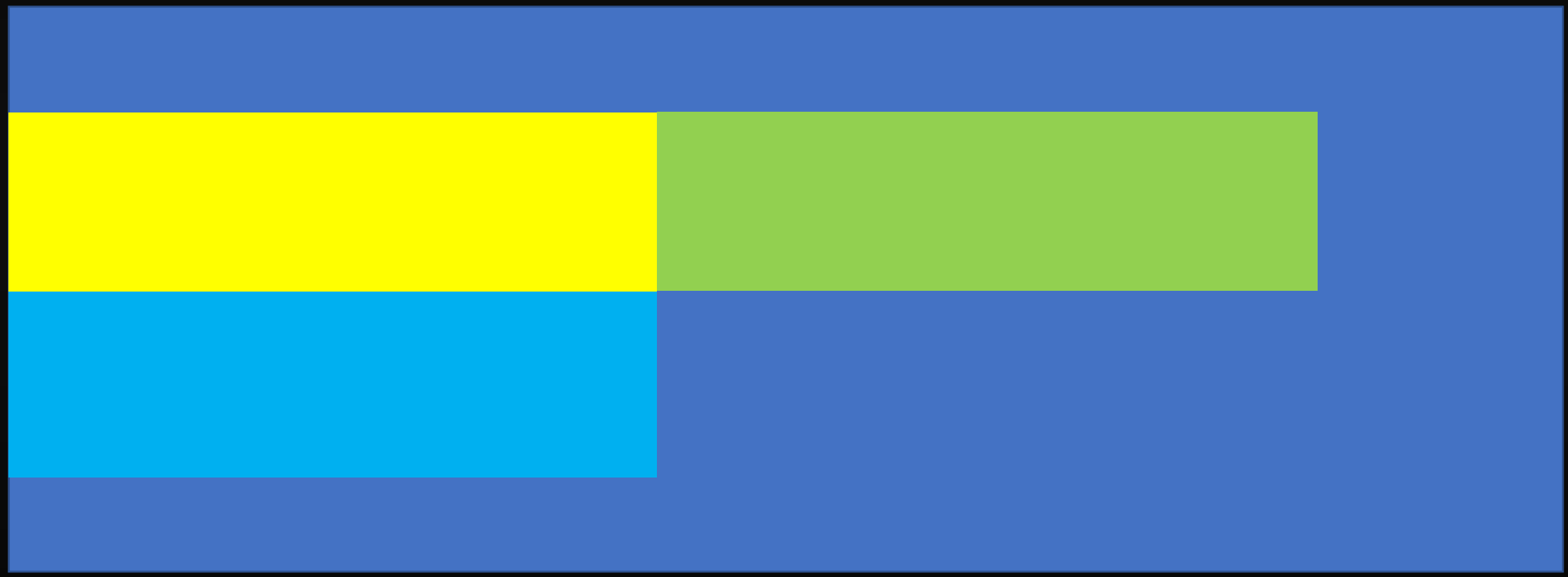
align-content: stretch (domyślnie)



align-content: flex-end



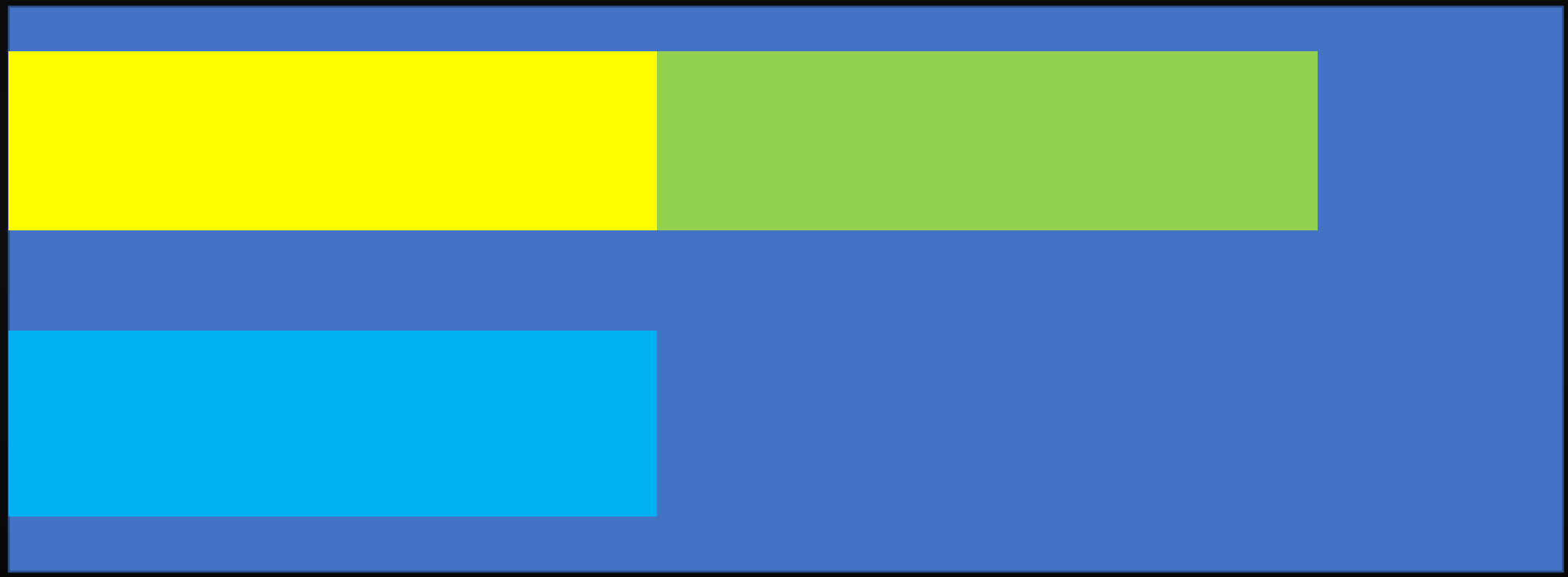
align-content: center



align-content: flex-start

align-content: space-between

align-content: space-around



flex-flow - skrót

Można podać dwie wartości:
flex-direction - 1. wartość (row, column ...)
flex-wrap - 2. wartość (nowrap, wrap)

flex-flow: row wrap;

WŁAŚCIWOŚCI NADAWANE ELEMENTOWI

Jeśli kontener ma display: flex, to możemy dla jego dzieci (elementów elastycznych) określić specjalne właściwości.

WŁAŚCIWOŚCI NADAWANE ELEMENTOM

- flex-grow
- flex-shrink
- flex-basis
- align-self
- order

order - liczba porządkowa

Zmiana domyślnego rozmieszczenie elementów w kontenerze.

Każdy element ma domyślnie właściwość order ustawioną na 0 czyli order: 0;

Jeśli elementy maja taką samą wartość order to decyduje kolejność w strukturze dokumentu (czyli tak jak są napisane w kodzie html)

order - liczba porządkowa

Jeśli jedna zmienimy wartość order to możemy zarządzać układem.

```
<div>1</div>
<div>2</div>
<div>3</div>
```


```
div:nth-child(1)
 { order: 1 }
```

Czym wyższy order tym niżej w hierarchii

order - liczba porządkowa

Jeśli jedna zmienimy wartość order to możemy zarządzać układem.

```
<div>1</div>
<div>2</div>
<div>3</div>
```


```
div:nth-child(3)
 { order: -1 }
```

Można też warość minusową.
Wtedy w hierarchii jest wyżej.

flex-grow

Gdy mamy wolne miejsca, tzn. kontener jest większy od dzieci.

Flex-grow pozwala rozdzielić miejsce, które zostało między dzieci rodzica (elementu z display:flex).

flex-grow ma domyślnie wartość 0 tzn. miejsce nie jest rozdzielane.

flex-grow

flex-grow możemy nadać elementom, które są dziecmi: wszystkim, części, albo tylko jednemu.

```
.child {
        flex-grow: 1
}
```

Wartość, którą przydzielamy określa proporcję wolnej przestrzeni jaką otrzyma element. Oczywiscie jeśli nie ma wolnej przestrzenie to nie zobaczymy efektu.

flex-grow

```
.rodzic {
        display: flex;
        width: 1200px;
}
```

```
.dzieci {
        width: 200px
}
```

flex-grow

```
.rodzic {
        display: flex;
        width: 1200px;
}
```

```
.dzieci {
        width: 200px
}
```

```
.dzieci.violet {
        flex-grow: 1
}
```

flex-grow

```
.rodzic {
        display: flex;
        width: 1200px;
}
```

```
.dzieci {
        width: 200px;
        flex-grow: 1
}
```

flex-grow - proporcja dzielnia

```
.rodzic {
        display: flex;
        width: 1200px;
}
```

```
.dzieci {
        width: 200px
        flex-grow: 1
}
```

```
.dzieci.violet {
        flex-grow: 3
}

.dzieci.orange {
        flex-grow: 2;
}
```

flex-grow

Dzielą się wolną przestrzenią!

```
.orange {
        flex-grow: 1
}
```

```
.violet {
        flex-grow: 5
}
```

```
.blue{
        flex-grow: 11
}
```

Jaki to da efekt w praktyce?

flex-grow

Dzielą się wolną przestrzenią!

```
.orange {
        flex-grow: 1
}
```

```
.violet {
        flex-grow: 5
}
```

```
.blue{
        flex-grow: 11
}
```

Wolna przestrzeń dzielona na 17 kawałków!

flex-grow

Dzielą się wolną przestrzenią!

```
.orange {
        flex-grow: 1
}
```

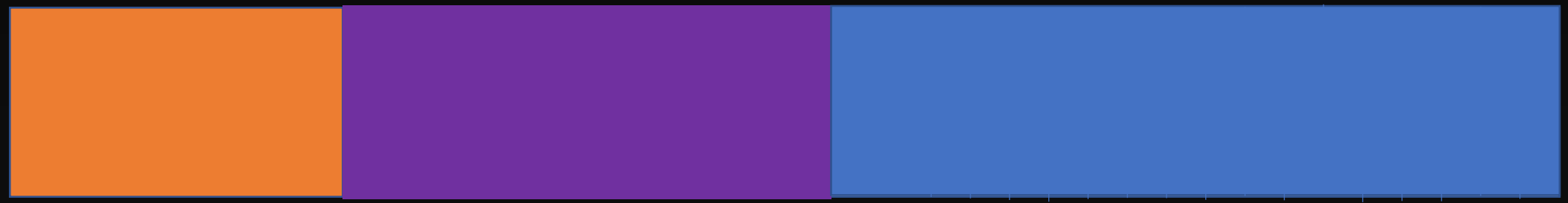
```
.violet {
        flex-grow: 5
}
```

```
.blue{
        flex-grow: 11
}
```

Wolna przestrzeń dzielona na 17 kawałków!

flex-grow

W praktyce będzie wyglądało to tak



```
.orange {
        flex-grow: 1
}
```

```
.violet {
        flex-grow: 5
}
```

```
.blue{
        flex-grow: 11
}
```

flex-grow

```
.children.one {
        flex-grow: 250
}
```

Jeśli tylko jeden element (dziecko) otrzyma flex-grow to zajmie on całe wolne miejsce, niezależnie od tego jak liczbę umieścimy.

Możemy przyjąć, że wolna przestrzeń zostanie "podzielona" na tyle części ile wpiszemy i przyporządkowa do elementów (a ponieważ będzie jeden to do jednego).

flex-grow i inne właściwości są zależne od szerokości i wysokości rodzica i wolnej przestrzeni w nim.

Wolna przestrzeń - to przestrzeń rodzica, która nie jest zagospodarowana przez elementy układu. Jeśli rodzic rozszerza się na szerokość (przy układzie row) lub wysokość (przy układzie column). Nalezy przy tym pamiętać, że kontener (element z własciwością display: flex) przyjmuje domyślnie 100% szerokości (width) swojego rodzica i tyle miejsca ile potrzebuje jeśli chodzi o wysokość.

flex-grow

Gdy mamy za dużo miejsca, tzn. kontener jest większy od dzieci.

Flex-grow pozwala rozdzielić miescje, które zostało między dzieci rodzica (elementu z display:flex).

flex-grow ma domyślnie dla każdego elementu układu wartość 0
tzn. miejsce nie jest rozdzielane i pozostaje niewykorzystane

flex-basis

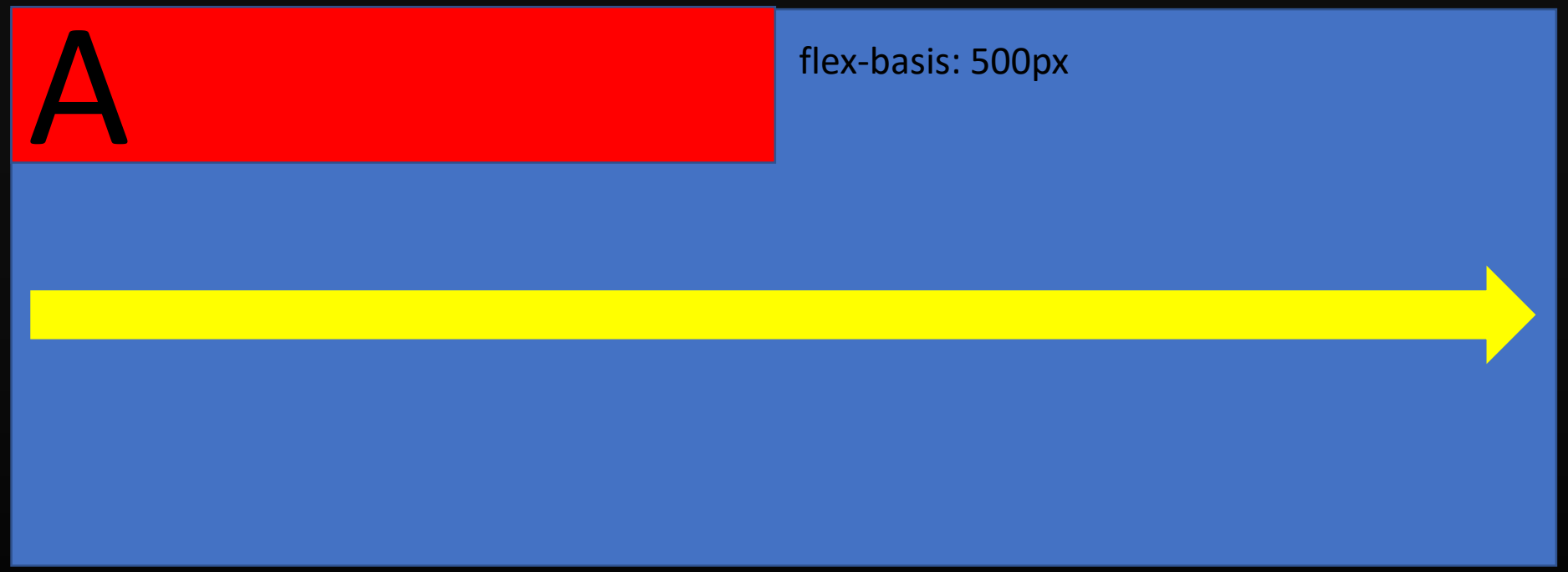
Rozmiar wejściowy (podstawa elementu)

Jak szeroki (przy ustawieniu row) lub jak wysoki (przy ustawieniu column) ma być dany element.

Flex-basis zachowuje się jak podobnie jak width (przy row) lub height (w column).

flex-basis - KIERUNEK STRZAŁKI

Flex-basis to przy row długość pozioma (kierunek strzałki)



flex-basis - DŁUGOŚĆ (zamiast wysokości czy szerokości)

Flex-basis to przy column długość pionowa (kierunek strzałki)

flex-basis

Domyślnie auto;

Auto - czyli zajmuj na szerokość (w układzie row) i na wysokość (w układzie column) tyle ile potrzebujesz.

flex-basis może przyjąć wartość w jednostkach bezwzględnych i względnych (px, %, rem, em, vh, vw)

flex-shrink

Gdy mamy za mało miejsca!

Domyślnie każdy element (element elastyczny) ma ustawiony flex-shrink na 1 tzn. zmniejszają się w takiej samej proporcji.

flex - skrót

Może przyjąć 3 wartości:

```
.elementUkładu {
        flex: 1 1 500px
}
```

Ale możemy też spotkać tak:
flex: 1 0; //flex-grow: 1; flex-shrink: auto
flex: 2; //flex-grow: 2
flex: 500px; //flex-basis: 500px;

Kolejne 3 właściowści to:
flex-grow
flex-shrink
flex-basis

Jeśli którejś nie podamy to przyjmuje wartość domyślną
czyli:
flex-grow: 0
flex-shrink: 1
flex-basis: auto

align-self

To samo co align-items ale **dla pojedynczego elementu**

stretch; center; flex-end; flex-start; baseline; auto.

Auto domyślne - przyjmuje wartość kontenera.

flex w wersji skróconej.

```
.children.one {
        flex: 1
}
```

flex-grow możemy zapisać też za pomocą wartości skrócenej flex: 1.

flex w wersji skróconej.

```
.children.one {
        flex: 0 1 auto /* domyślnie */
}
```

Normalnie ta wartość może przyjąć 3 elementy (flex-grow, flex-shrink i flex-basis), ale jeśli podamy tylko jedną, to przypisze ją do flex-grow (pozostałym przypisując wartości domyślne).

flex - skrót

Może przyjąć 3 wartości:

```
.elementUkładu {
        flex: 1 1 500px
}
```

Ale możemy też spotkać tak:
flex: 1 0;
flex: 2;
flex: 500px;

flex - skrót

Może przyjąć 3 wartości:

```
.elementUkładu {
        flex: 1 1 500px
}
```

Ale możemy też spotkać tak:
flex: 1 0;
flex: 2;
flex: 500px;

Kolejne 3 właściowści to:
flex-grow
flex-shrink
flex-basis

Jeśli którejś nie podamy to przyjmuje wartość domyślną czyli:
flex-grow: 0
flex-shrink: 1
flex-basis: auto

WŁAŚCIWOŚCI NADAWANE RODZICOWI

- flex-direction
- flex-wrap
- align-items
- justyfi-content
- align-content

WŁAŚCIWOŚCI NADAWANE ELEMENTOM

- flex-grow
- flex-shrink
- flex-basis
- align-self
- order

Kluczem jest umiejętności ich połączenia.

Biegłość oznacza

- świadomość istnienia właściwości i zdolności przewidywania efektu
- łączenia różnych właściwości flexa
- wiedza o tym jaki efekt można uzyskać
- umiejetnosć uzyskania zaplanowanego efektu za pomocą flexa.