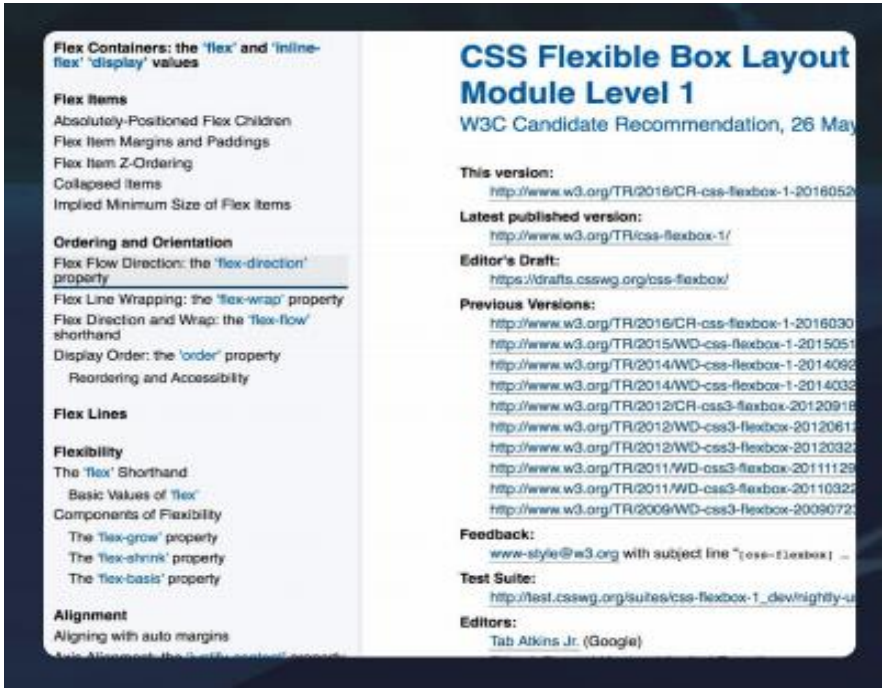
CSS Flexible Box Layout
elastyczny układ pudełkowy

Dr inż. Grzegorz Rogus

Czym jest FlexBox?

Flexbox – nowy model tworzenia szkieletów stron WWW.

Cel -> stworzenie prostszego i bardziej elastycznego sposobu tworzenia nowoczesnych witryn -> zwłaszcza w kontekście aplikacji RIA.



2 nowe wartości dla display

11 nowych własności

Flexbox jako nowy Display Type

- Block layout
- Table layout
- Inline layout
- Positioning layout

- Flex layout
    - Flex containers
    - Flex items
    - Flex lines

Model tabelaryczny

Zalety:
- nie ma problemów z ustawianiem wysokości
- łatwe wyśrodkowywanie elementów
- zachowują się w przewidywalny sposób

Wady:
- wolne renderowanie dla dużych tabel
- łatwo doprowadzić do bałaganu w kodzie
- szablon na tabelach jest nie semantyczny

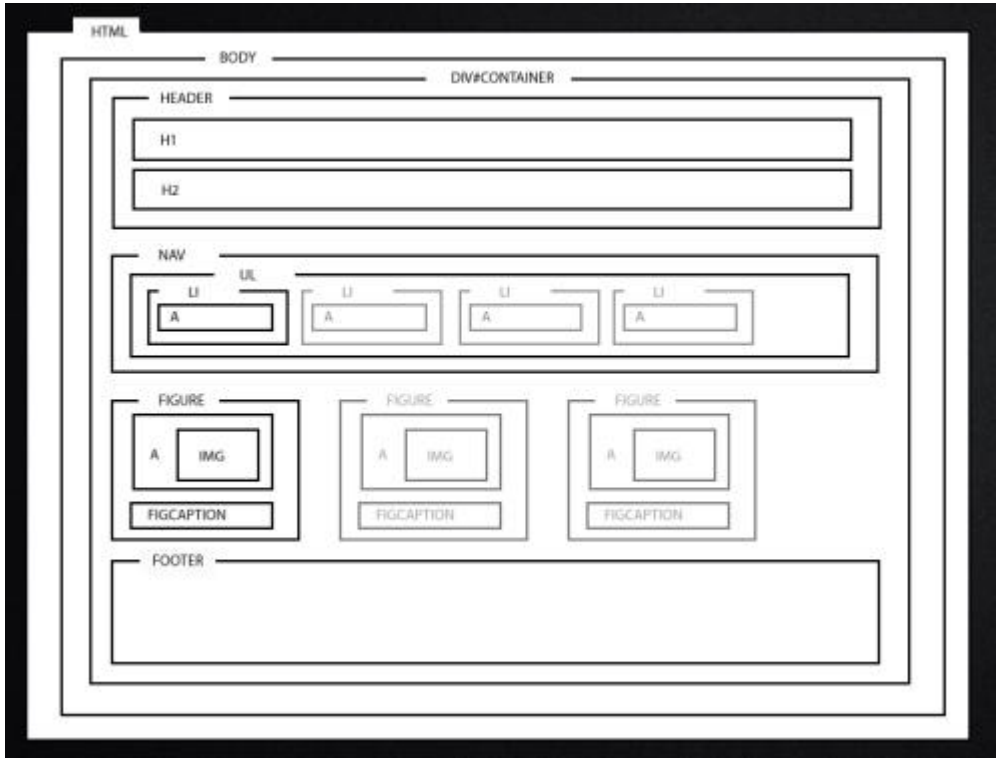
Day	Seminar		
	Schedule		Topic
	Begin	End	
Monday	8:00 a.m.	5:00 p.m.	Introduction to XML
			Validity: DTD and Relax NG
Tuesday	8:00 a.m.	11:00 a.m.	XPath
	11:00 a.m.	2:00 p.m.	XSL Transformations
	2:00 p.m.	5:00 p.m.	
Wednesday	8:00 a.m.	12:00 p.m.	XSL Formatting Objects

```
<table>
   <tbody>
      <tr>
         <td colspan="3">Header</td>
      </tr>
      <tr>
         <td style="width: 20%;">Sidebar</td>
         <td style="width: 60%;">Content</td>
         <td style="width: 20%;">Ad</td>
      </tr>
   </tbody>
</table>
```

Model pudełkowy

Zalety:
- przejrzysty i łatwy w utrzymaniu
- elementy szybko się wyświetlają
- pozwala nadać elementom semantyczne znaczenie

Wady:
- utrudnione wyśrodkowywanie
- gorsza elastyczność elementów
- wymaga znajomości sztuczek



Era pływających pudełek

```
<div>Header</div>
<div style="width: 20%; float: left;">
   Menu
</div>
<div style="width: 60%; float: left;">
   Content
</div>
<div style="width: 20%; float: left;">
   Ad
</div>
```

Flexbox

- Stworzony z myślą o aplikacjach z zaawansowanym interfejsem (RIA)
- Usprawnia układanie elementów w szablonach aplikacji
- Ułatwia wyrównywanie elementów względem siebie
- Pozwala lepiej wykorzystywać pustą przestrzeń

Zastosowanie FlexBox

Moduł ten dba, by znaczniki na stronie były płynnie skalowane i w efekcie, by dopasowywały się do dostępnego miejsca bez wprowadzania dodatkowych definicji opływania, pozycjonowania czy wykonywania złożonych obliczeń.

Zastosowanie:

Choć można oczywiście zbudować całą stronę za pomocą narzędzi modułu *Flexbox*, to jednak najlepiej sprawdzają się one w czasie przygotowywania fragmentów interfejsu dla mniejszych składowych strony (menu, elementy formularza, wiele różnych przycisków itp

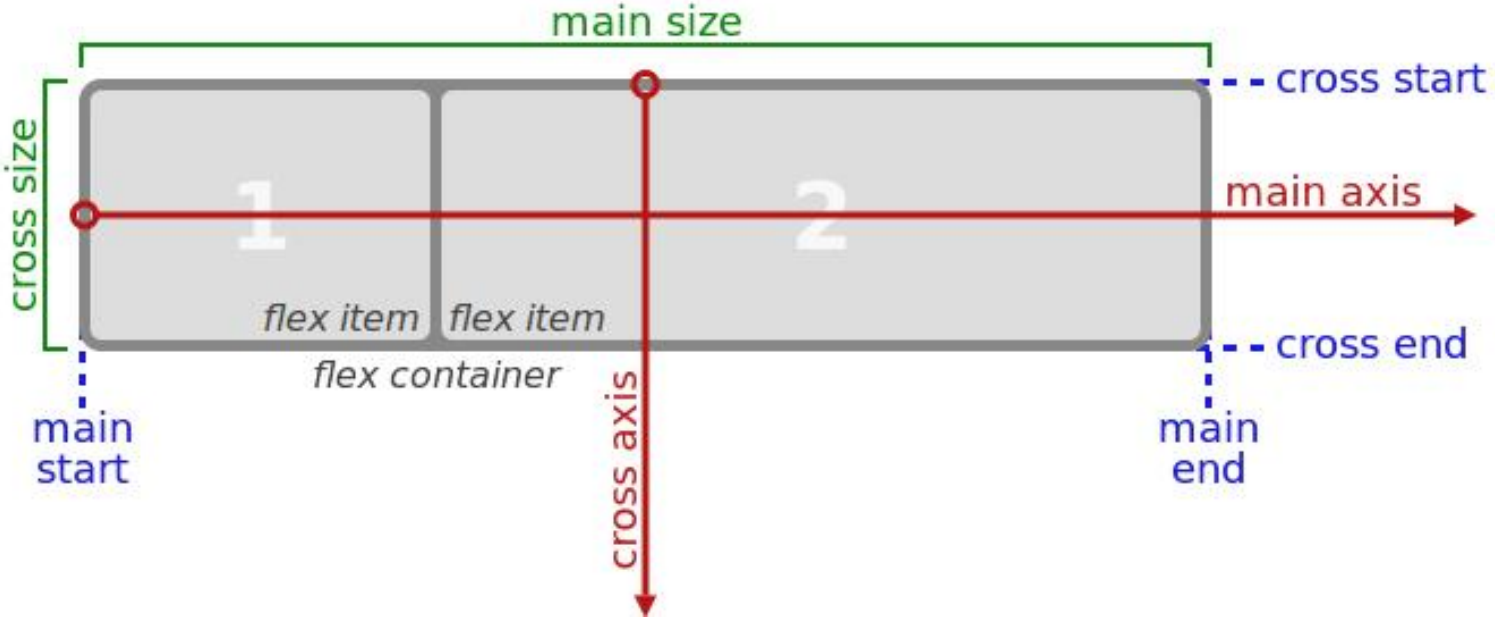
Wstawiamy flexboxa

```
.container{
    display: flex;
    /* display: inline-flex */
}
.item{
    /* automatycznie: */
    /* display: flex-item; */
}
```

Domyślny wygląd:

- układ horyzontalny
- szerokość dopasowana do treści
- wysokość wyrównana
- elementy dosunięte do lewej

Jak to działa



Domyślny wygląd:

- układ horyzontalny
- szerokość dopasowana do treści
- wysokość wyrównana
- elementy dosunięte do lewej

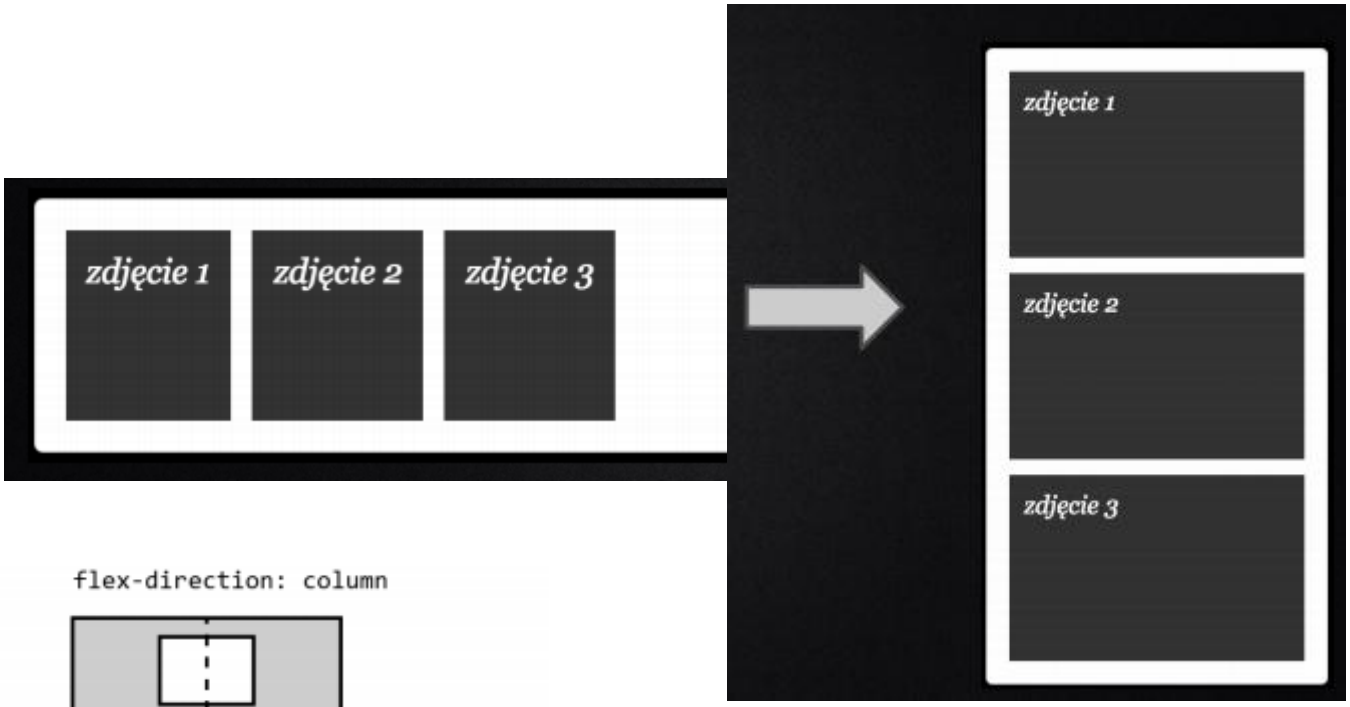
Właściwości flex container

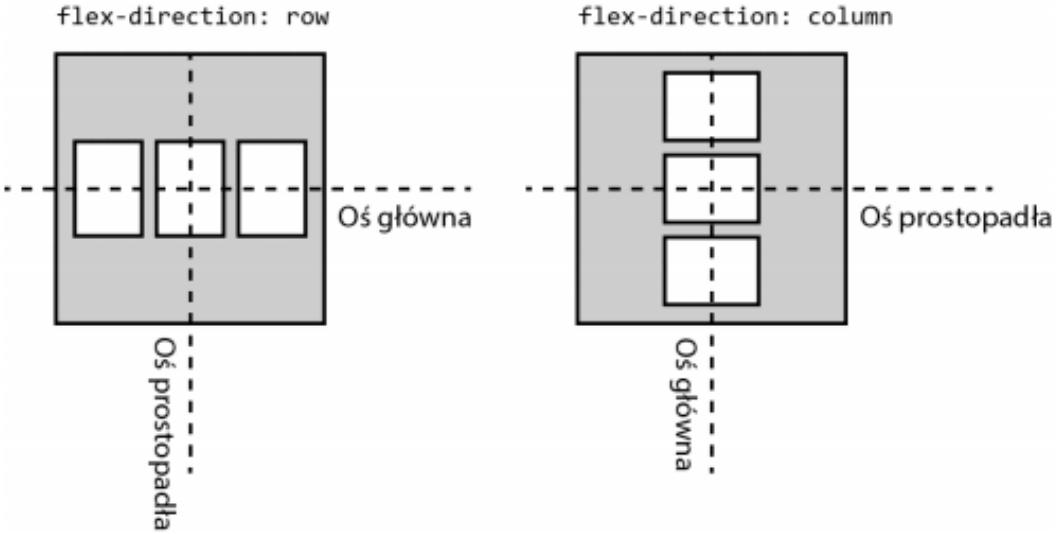
- **flex-wrap** określa ile ‚wierszy' może zawierać w sobie flex container oraz kierunek cross axis
- **flex-direction** ustala main axis, a więc kierunek, w którym układane są flex items
- **flex-flow** to tzw. ‚shorthand' dla flex-wrap i flex-direction
- **align-items** definiuje jak flex items rozłożone są wzdłuż cross axis – jeśli przyjąć za punkt wyjścia powyższą ilustrację to można powiedzieć, że odpowiada za pionowe ułożenie flex items
- **justify-content** określa ułożenie wzdłuż main axis i sposób dystrybucji wolnej przestrzeni – odnosząc się do ilustracji z początku artykułu można powiedzieć, że decyduje o horyzontalnym ułożeniu flex items
- **align-content** decyduje o rozmieszeniu ‚lini' flexbox'a gdy jest ich więcej niż jedna i gdy w ramach cross axis występuje dodatkowe, niezagospodarowane miejsce

Właściwości flex item:

- **flex-grow** zgodnie ze specyfikacją możemy określić możliwość zwiększenia rozmiarów flex items; zasada wzrostu oparta jest o proporcje
- **flex-shrink** tutaj analogicznie jak wyżej z tą różnica, że określamy możliwość zmniejszania rozmiarów flex items
- **flex-basis** domyślny rozmiar elementów (przed dystrybucją wolnej przestrzeni)
- **flex** tzw. ‚shorthand' dla *flew-grow*, *flex-shrink* i *flex-basis*
- **order** definiuje kolejność wyświetlania flex items
- **align-self** daje możliwość nadpisywania *align-items* lub domyślnego ułożenia dla jednego elementu

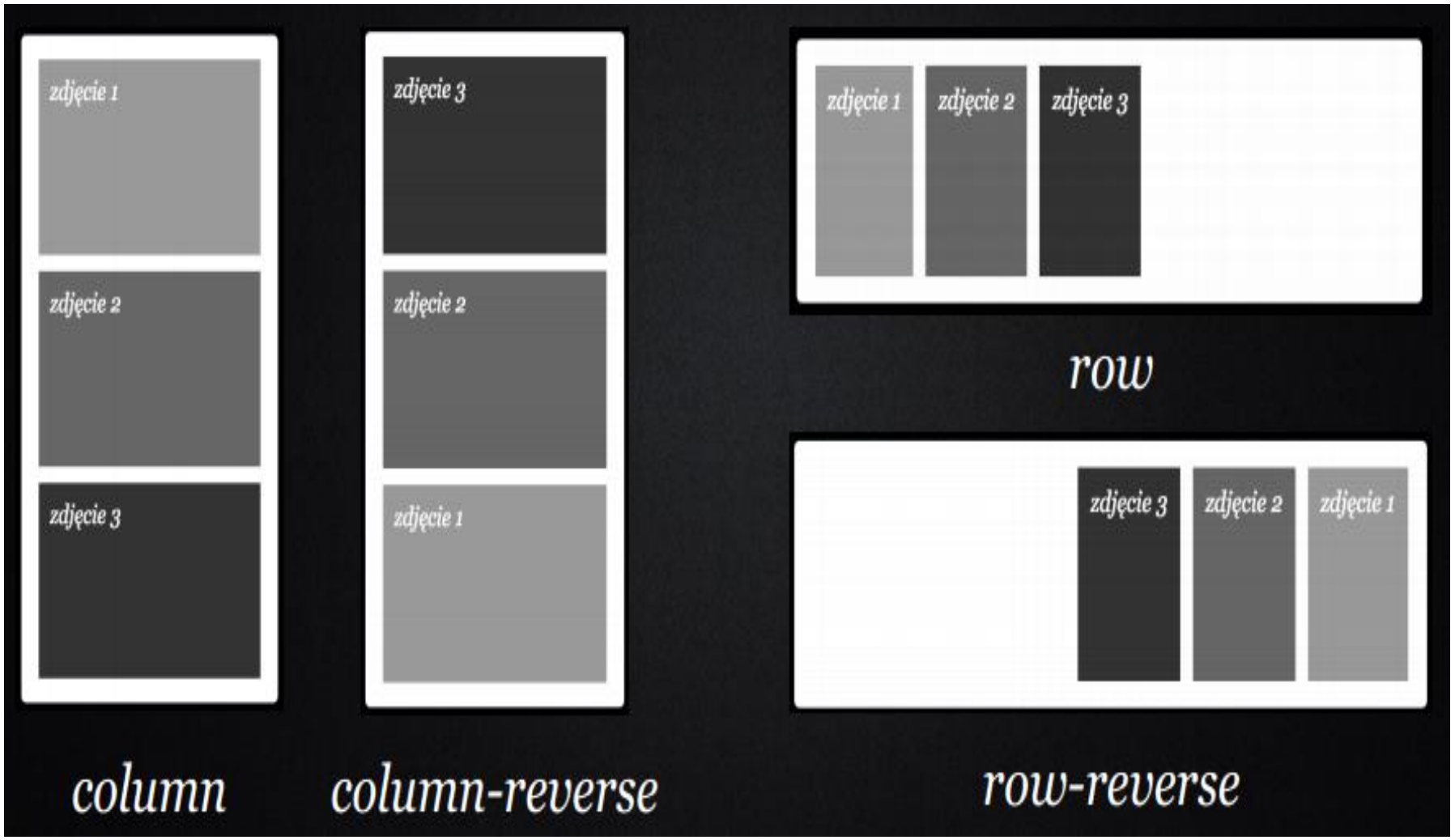
Kierunek układania



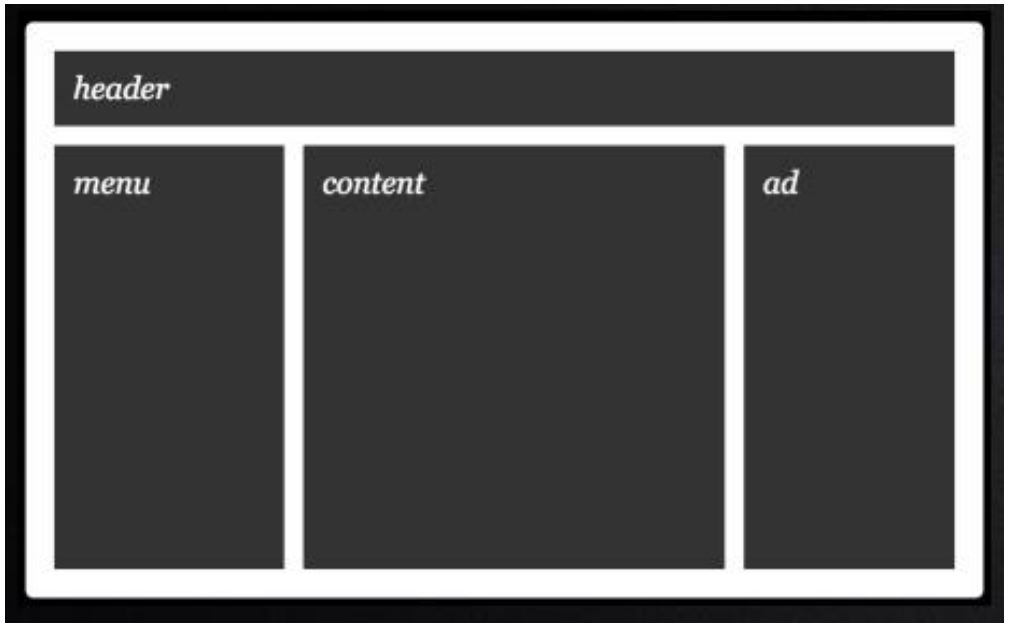


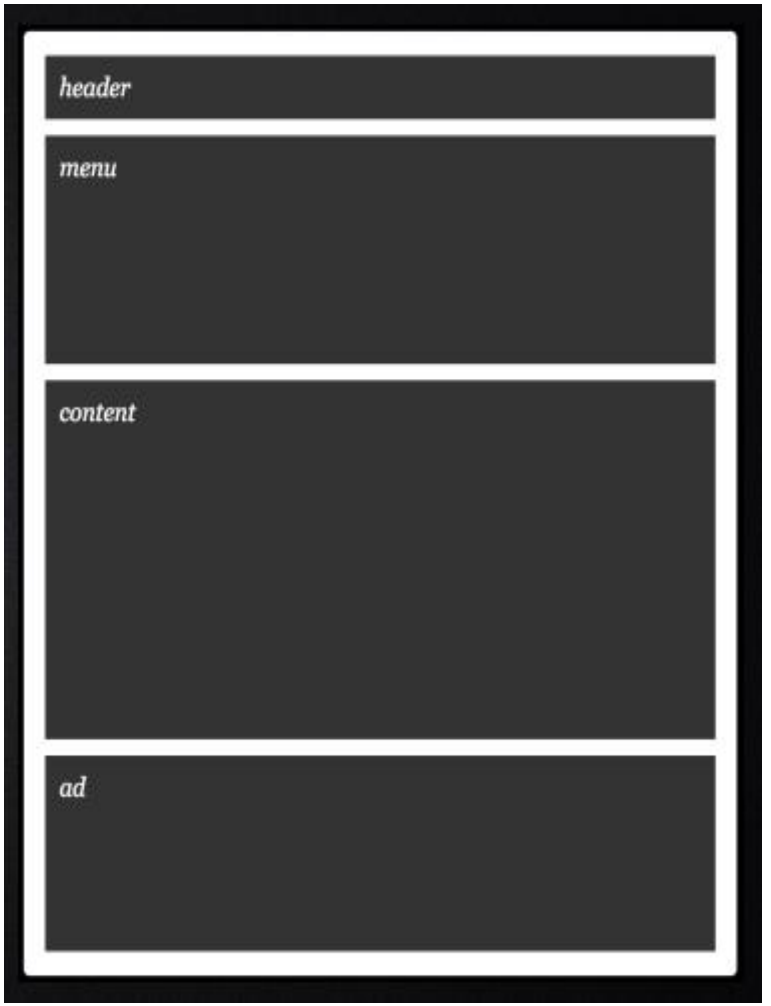

```
.container{
    /* flex-direction: row; */
    flex-direction: column;
}
```

Kierunek układania cd. (flex-direction)

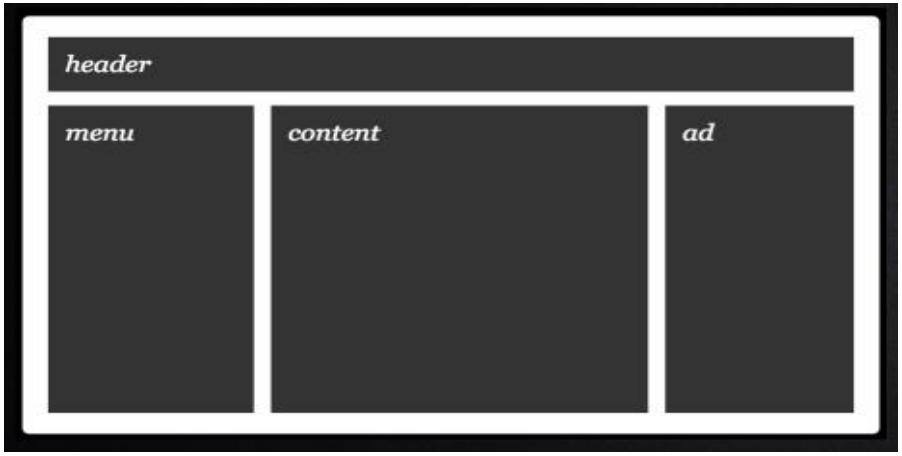


Zmiana kolejności

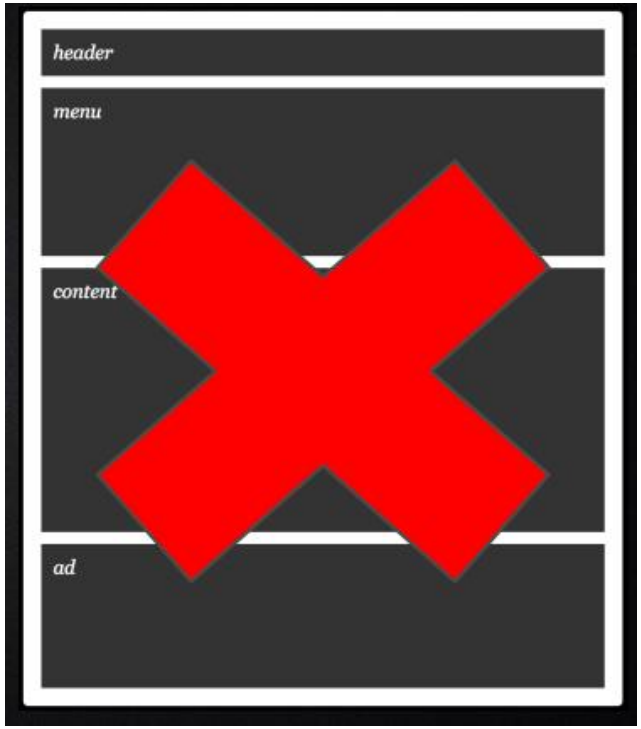


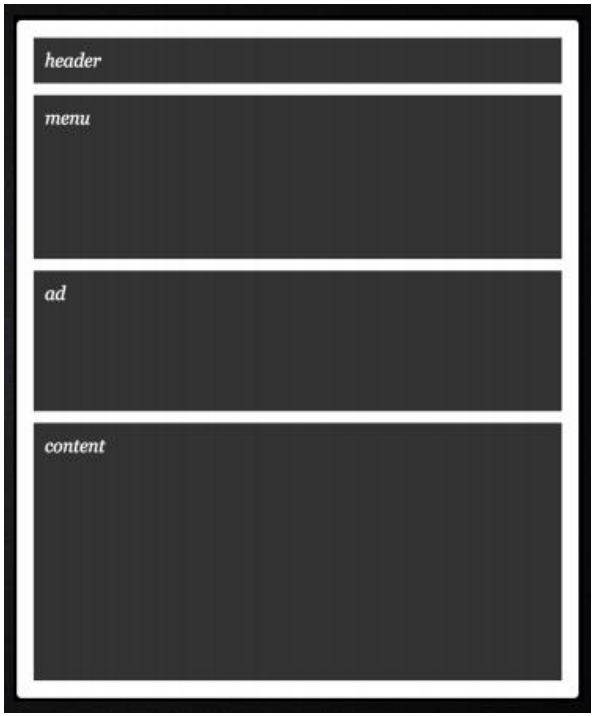


Zmiana kolejności




```
.menu    { order: 1 }
.content { order: 2 }
.ad      { order: 3 }

@media (max-width: 768px){
    .menu     { order: 1 }
    .content  { order: 3 }
    .ad       { order: 2 }
}
```

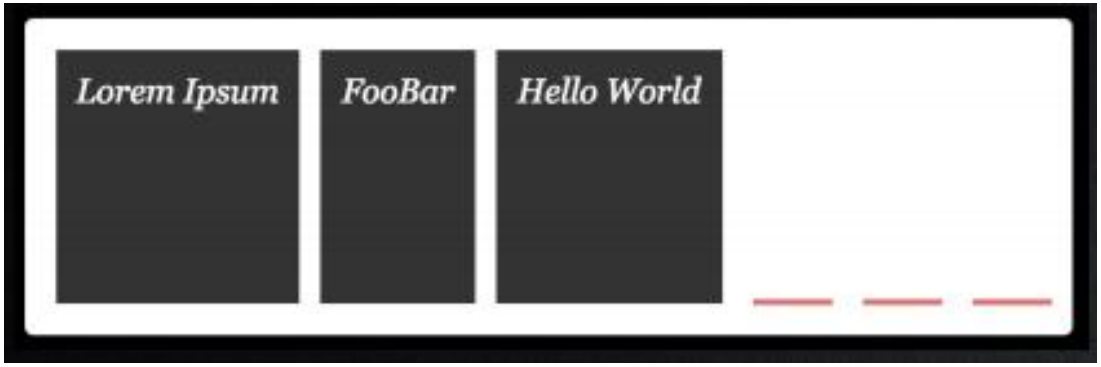




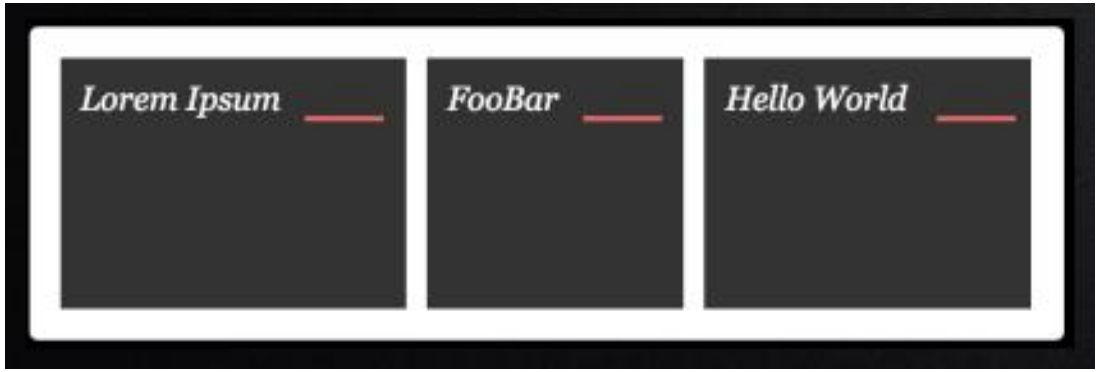
Kontrolowanie rozmiaru

flex-basis	podstawowy rozmiar elementu (auto | 0 | px | em | percent | …)
flex-grow	o jaką część element może się rozszerzyć względem innych elementów korzystając z wolnego miejsca (liczba całkowita bez jednostki)
flex-shrink	o jaką część element może się zmniejszyć względem innych elementów korzystając z dostępnego miejsca (liczba całkowita bez jednostki)

Kontrolowanie rozmiaru




```
.left   { flex-grow: 0 }
.middle { flex-grow: 0 }
.right  { flex-grow: 0 }
```




```
.left   { flex-grow: 1 }
.middle { flex-grow: 1 }
.right  { flex-grow: 1 }
```

Kontrolowanie rozmiaru

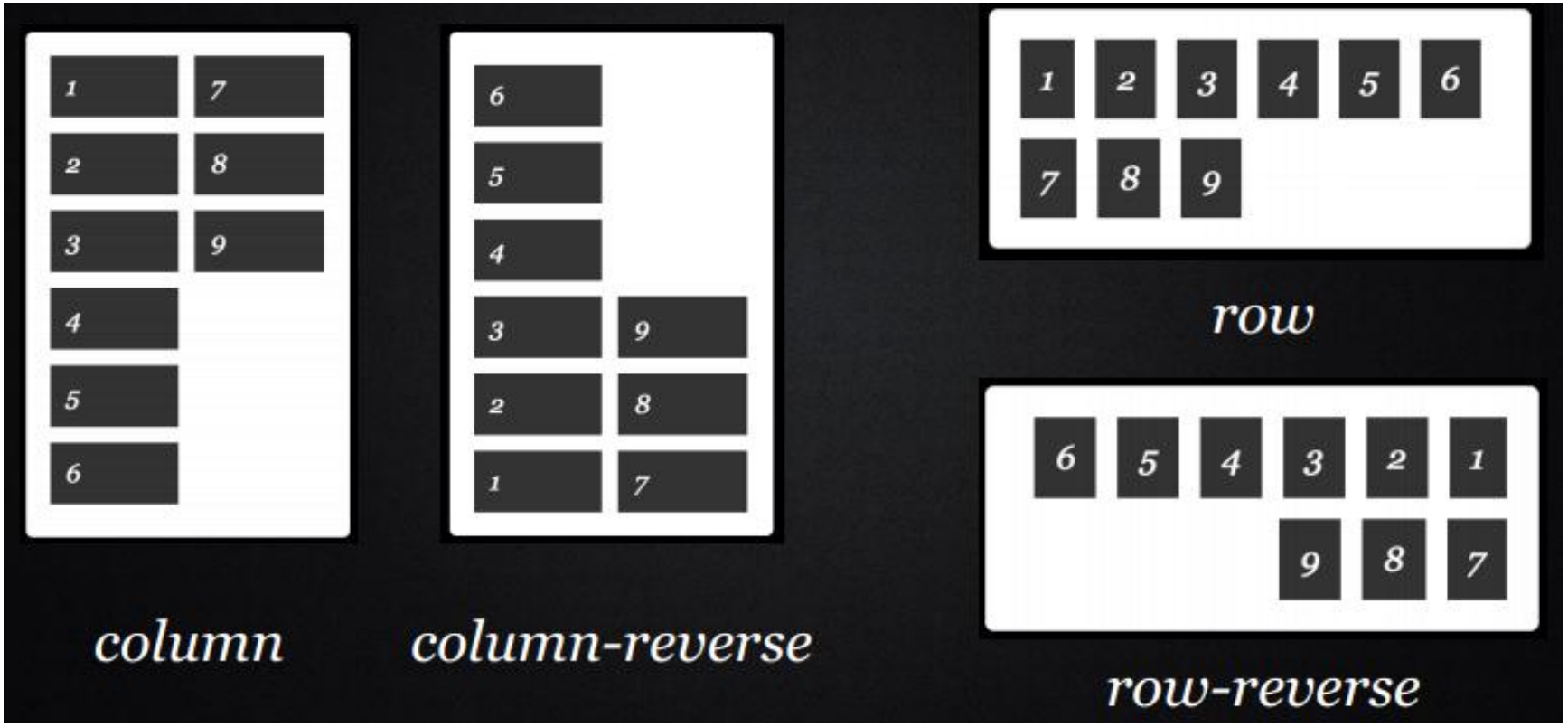
```
.left   { flex-grow: 1 }
.middle { flex-grow: 1 }
.right  { flex-grow: 1 }
```

```
.left   { flex-grow: 1 }
.middle { flex-grow: 2 }
.right  { flex-grow: 1 }
```

```
/* flex: [flex-grow] [flex-shrink] [flex-basis]; */
.avatar  { flex: 0 0 100px; }
.comment { flex: 1 0 400px; }
```

Zawijanie elementów (flex-wrap)

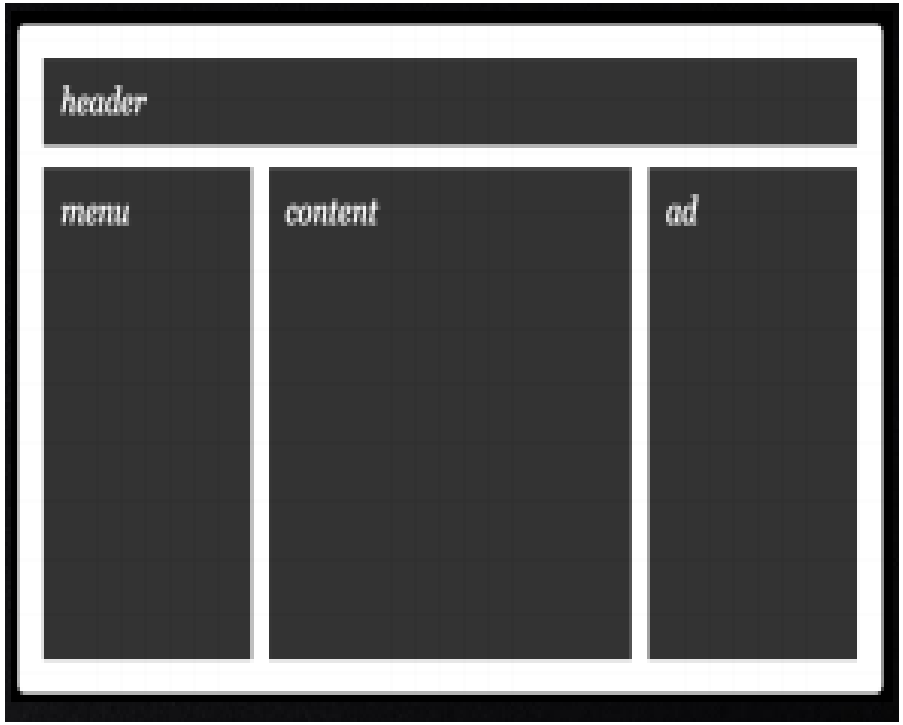
```
.container{
   /* flex-wrap: nowrap; */
   flex-wrap: wrap;

   /* flex-flow: [flex-direction] [flex-wrap] */
   flex-flow: row nowrap;
}
```



Przykład użycia

```
.container{
    display: flex;
    flex-flow: row wrap;
    max-width: 1024px;
}
```

```
.header{
    flex-basis: 100%;
}
```

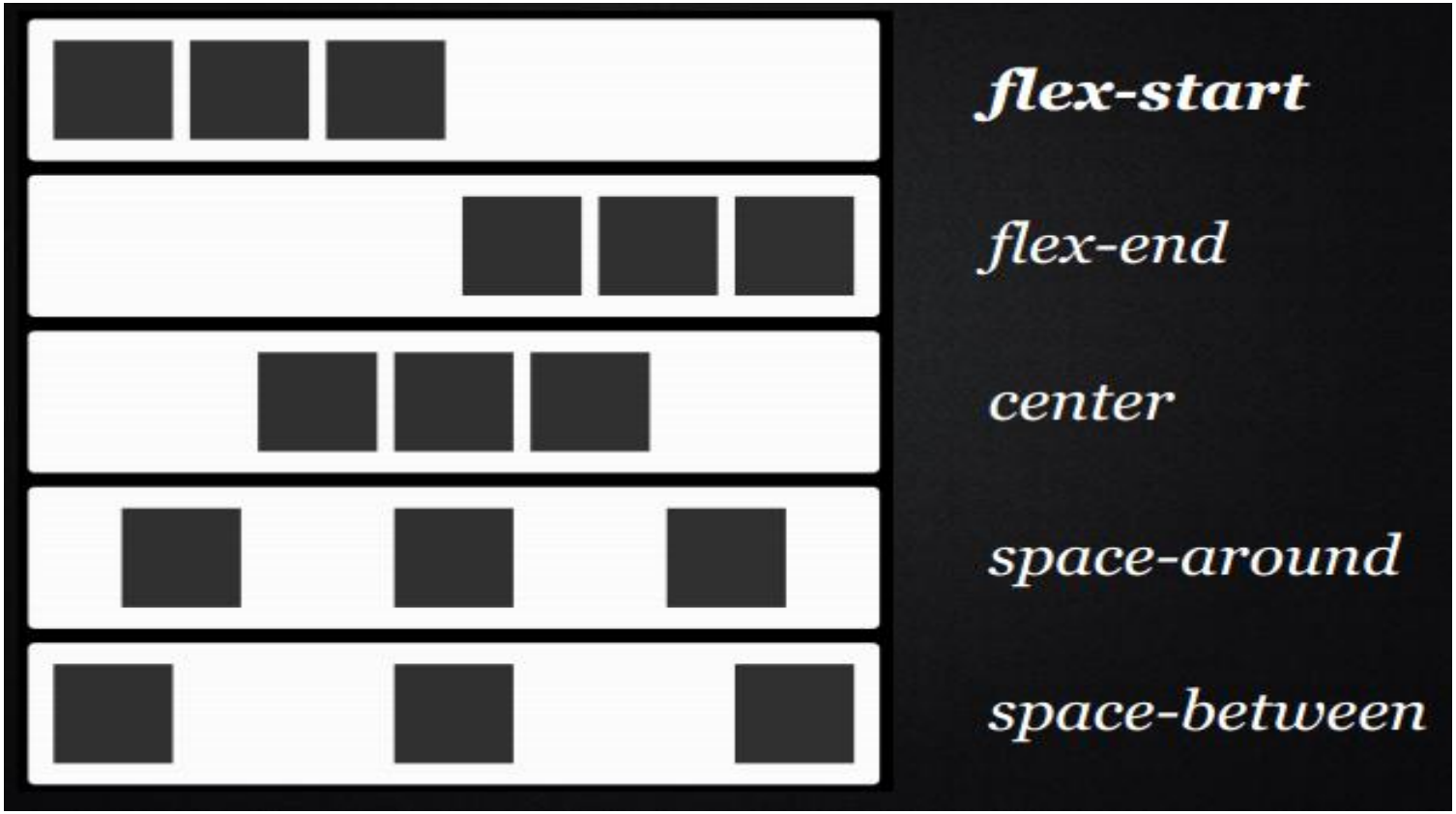


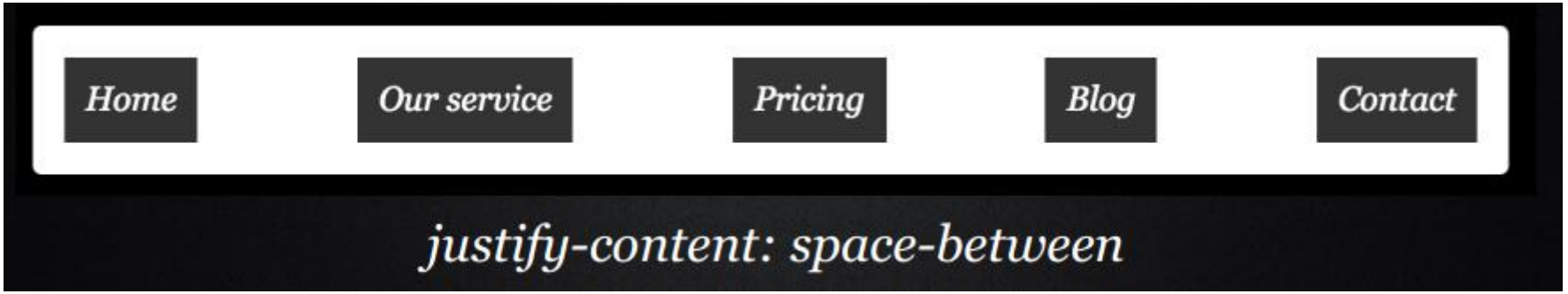

```
.menu    { flex: 0 1 100px; }
.content { flex: 1 1 auto; min-height: 200px; }
.ad      { flex: 0 1 100px; }
```

Wyrównywanie

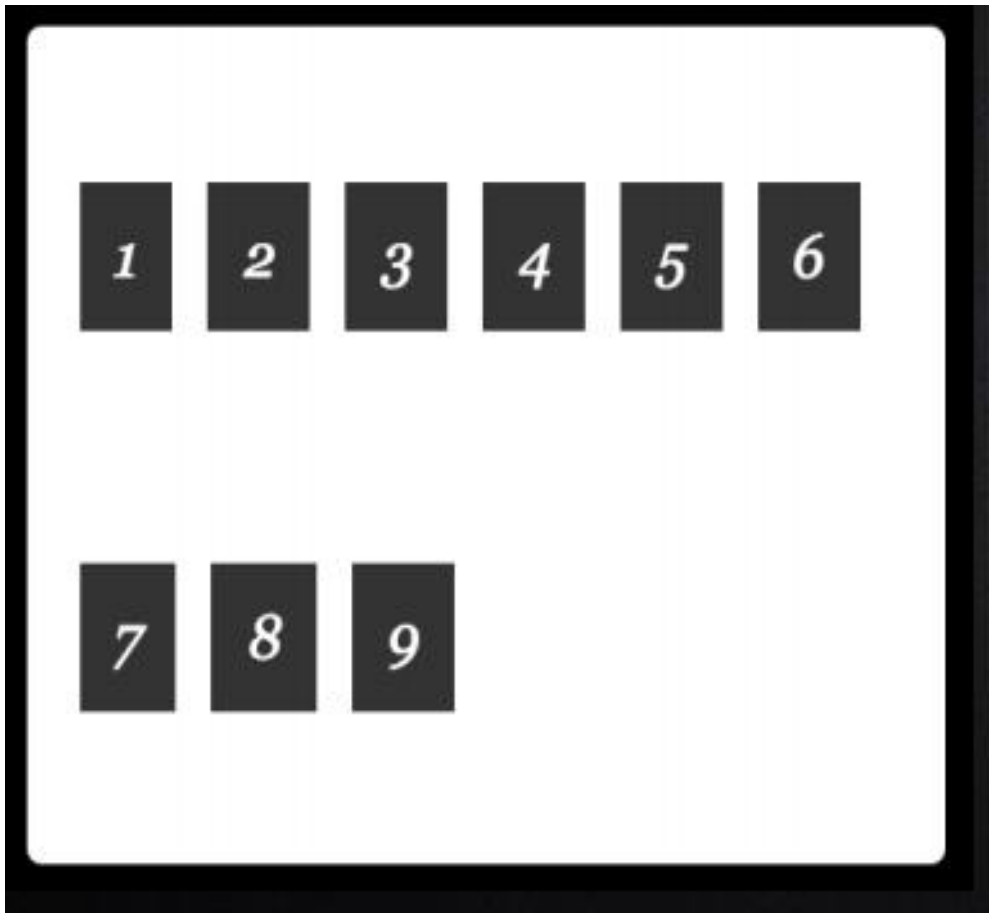
justify-content	wypełnienie elementami w osi głównej
align-content	wypełnienie rzędami w osi poprzecznej
align-items	wyrównanie wszystkich elementów w osi poprzecznej (przypisywane kontenerowi)
align-self	wyrównanie elementu w osi poprzecznej (przypisywane danemu elementowi)

Wyrównywanie (justify-content)



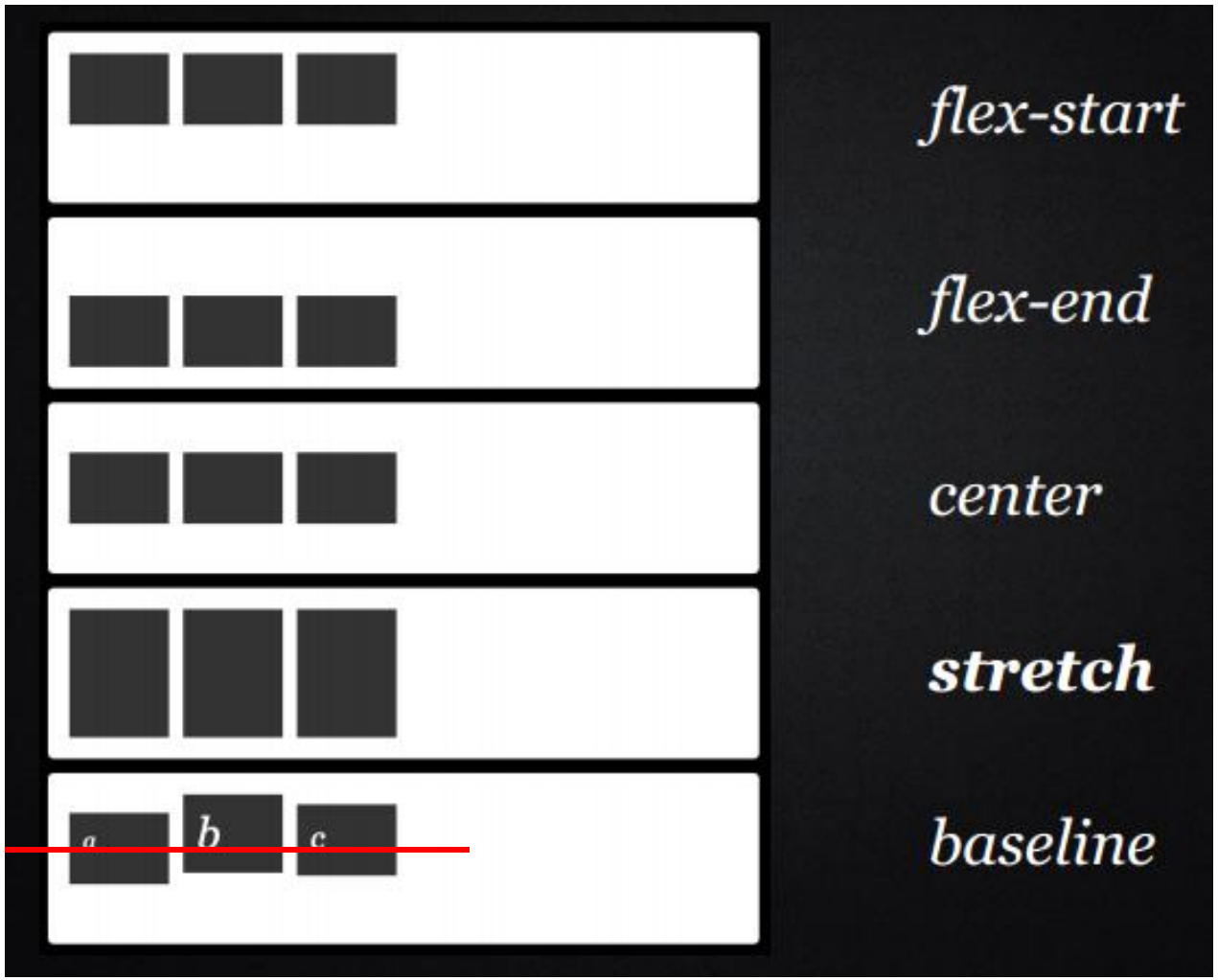


Wyrównywanie (align-content)

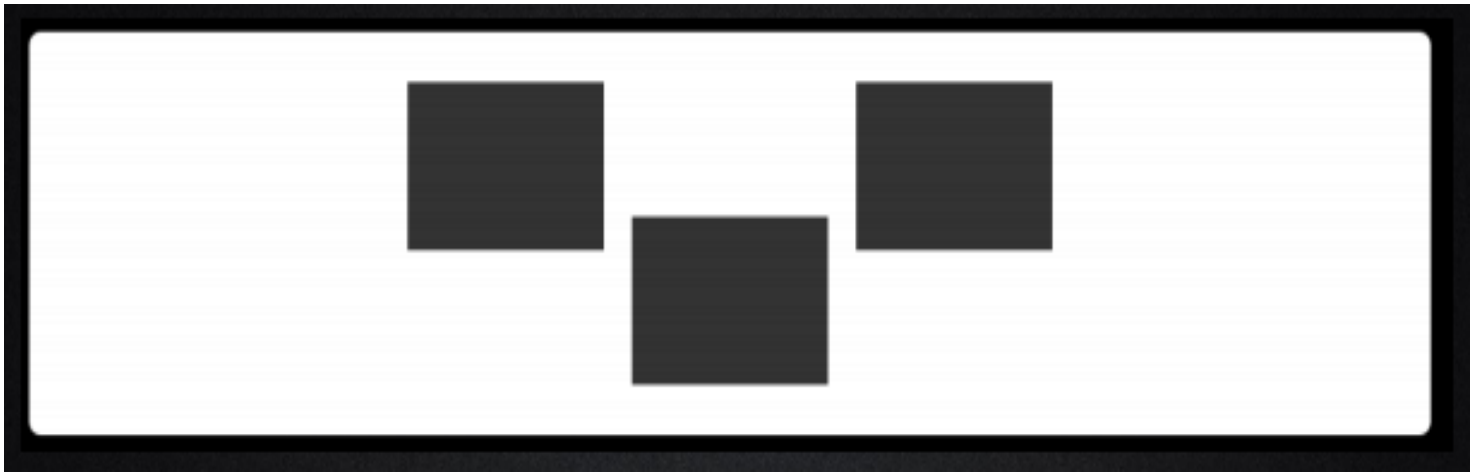



```
.container{
    display: flex;
    flex-flow: column wrap;
    align-content: space-around;
}
```

Wyrównywanie (justify-items)



Wyrównywanie (align-self)



```
.container{
    display: flex;
    align-items: flex-start;
}
.middle-item{
    align-self: flex-end;
}
```