Jak używać Flexbox w CSS (przykłady)

Sprawdź, jak możesz korzystać z funkcji Flexbox w CSS, kierując się przy tym konkretnymi przykładami.

Flexbox to jedna z najlepszych funkcji CSS. Zaprojektowano ją jako jednowymiarowy model tworzenia układu elementów oraz metodę, która oferuje dystrybucję przestrzeni między elementami w interfejsie. **Flexbox ułatwia też wyrównanie elementów.** Dzięki Flexbox łatwiej jest projektować elastyczne i responsywne layouty bez korzystania z float, czy position. W artykule tym nauczymy się, jak korzystać z Flexbox w CSS, przyglądając się tym samym kilku praktycznym przykładom. Zaczynajmy!

Definiowanie kontenera

Aby zacząć korzystać z Flexbox, należy zedefiniować element, który stanie się kontenerem, który opakowuje wszystkie elementy potomne w następujący sposób:

```
<div class="container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

Kontener staje się elastyczny przez przestawienie właściwości display na flex:

```
.container {
  display: flex;
  background-color: red;
}
.container div{
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
```

Oto rezultat:

Czerwony kontener, który zawiera 3 elastyczne elementy div

Jak widać, ustawiając właściwość display na flex, elementy potomne automatycznie stały się elastyczne. Możesz teraz wykorzystać właściwości kontenera, takie jak **justify-content**, czy **align-items**, aby, na przykład, wyśrodkować elementy wewnątrz kontenera. Przyjrzymy się temu w poniższych przykładach.

Właściwość flex-direction

Właściwość flex-direction definiuje, w którym kierunku ułożyć elementy potomne. Dostępne wartości to column lub row. Poniższy przykład ustawia flex-direction na column (od góry do dołu). W rezultacie, elementy potomne wewnątrz kontenera div będą formowały linię pionową.

Spójrzmy na poniższy przykład:

```
.container {
  display: flex;
  flex-direction: column;
  background-color: red;
}
.container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

Oto rezultat:

Ustawienie elementów w kolumnie

Oto ten sam przykład, ale właściwość flex-property została ustawiona na row, co sprawi, że element potomny będzie ustawiony w kontenerze poziomo:

```
.container {
  display: flex;
  flex-direction: row;
  background-color: red;
}
.container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

Rezultat:

Ustawienie elementów w wierszu

Możesz także odwrócić kierunek elementu potomnego wewnątrz kontenera, ustawiając właściwość flex-direction na column-reverse, czy row-reverse.

Właściwość flex-wrap

Właściwość **flex-wrap** określa, czy elementy wewnątrz mają być zawijane, czy nie. Poniższy przykład ma 12 obiektów flex i ustawia właściwość flex-wrap na wrap.

Polecam wstawić poniższy kod do edytora albo do Codepen i dopasować rozmiar okna przeglądarki - pozwoli to w pełni pokazać, do czego flex-wrap jest zdolne.

HTML:

```
<div class="container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
  <div>11</div>
  <div>12</div>
</div>
```

CSS:

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

```
  flex-wrap: wrap;
  background-color: red;
}
.container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

Jeśli chcesz, aby elementy potomne nie były zawijane, ustaw właściwość flex-direction na nowrap (to domyślna wartość):

```
.container {
  display: flex;
  flex-wrap: nowrap;
  background-color: red;
}
```

Właściwość justify-content

Właściwości **justify-content** używa się do wyrównania elementów wewnątrz flex. Możesz przekazać tej właściwości takie wartości, jak center, flex-start, flex-end, space-between itd.

Wartość center zgrupuje wszystkie elementy na środku kontenera:

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

Wartość flex-start sprawi, że elementy zostaną wyrównane do początku kontenera:

```
.container {
  display: flex;
  justify-content: flex-start;
}
```

Wartość flex-end sprawi, że elementy zostaną wyrównane do końca kontenera:

```
.container {
  display: flex;
  justify-content: flex-end;
}
```

Wartość space-between sprawi, że obiekty zostaną rozmieszczone z równymi odstępami między sobą.

```
.container {
  display: flex;
  justify-content: space-between;
}
```

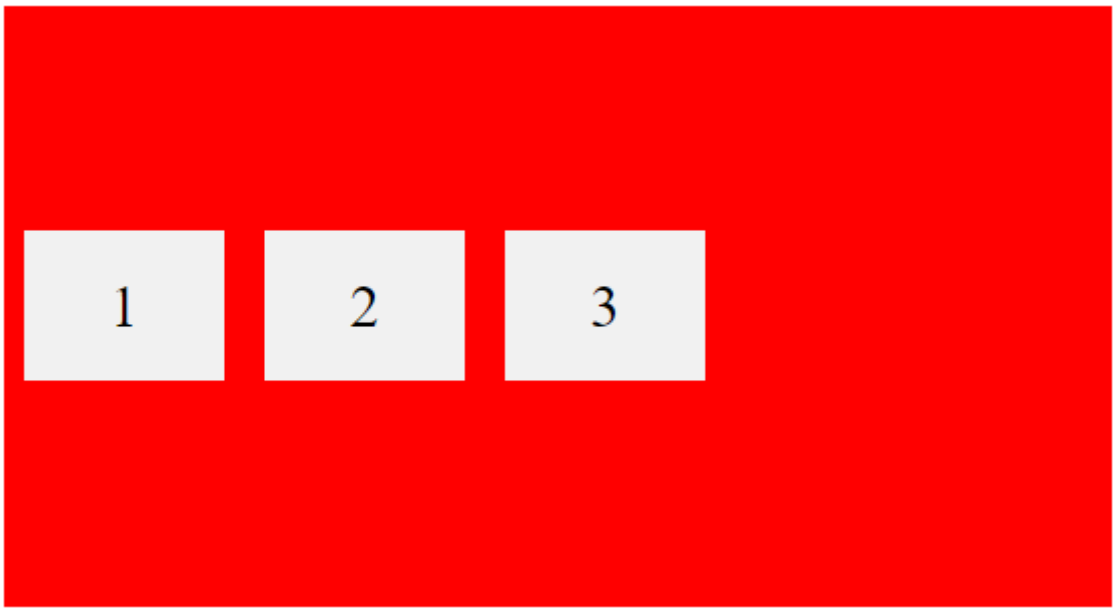
Właściwość align-items

Właściwość align-items służy wyrównaniu obiektów flex. To prawie to samo, co justify-content, ale pracujemy pionowo, a nie poziomo. To dlatego pokażę tylko jeden przykład, zamiast powtarzać ciągle te same:

Oto przykład, który wyśrodkowuje elementy potomne w pionie wewnątrz kontenera:

```
.container {
  display: flex;
  height: 300px;
  align-items: center;
  background-color: red;
}
.container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

Rezultat:



Elementy wyrównane w pionie

Właściwość align-items posiada te same wartości, co justify-content. Jedyną różnicą jest to, że pracujemy pionowo, a nie poziomo.

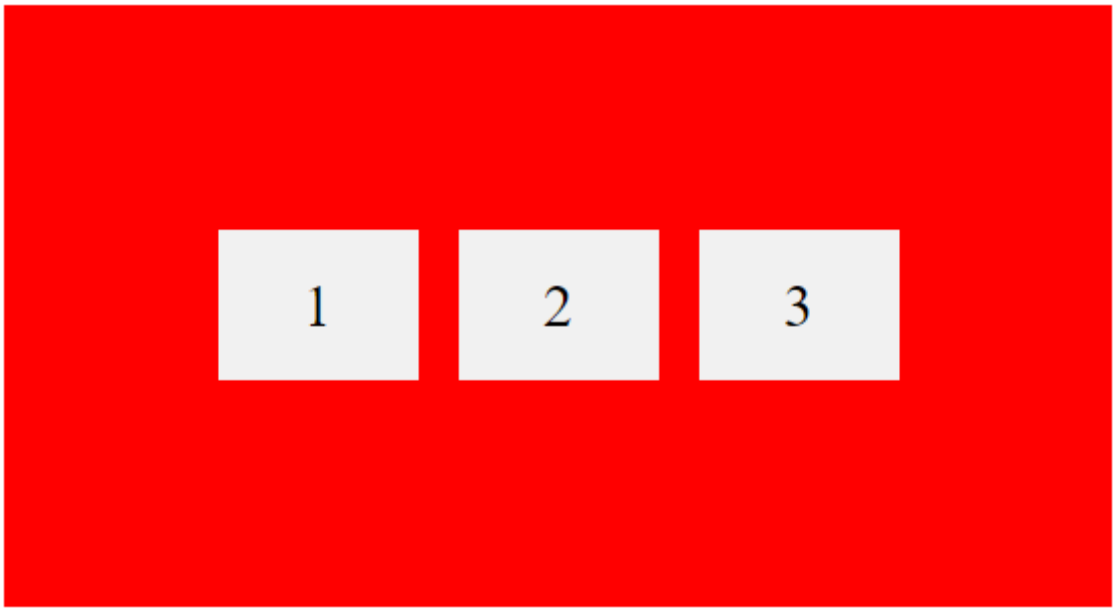
Pionowe i poziome wyśrodkowanie

Tym razem użyjemy zarówno justify-content, jak i align-items, aby łatwiej wyśrodkować elementy potomne w sposób pionowy oraz poziomy.

Oto przykład:

```
.container {
  display: flex;
  height: 300px;
  align-items: center;
  justify-content: center;
  background-color: red;
}
.container > div {
  background-color: #f1f1f1;
  width: 100px;
  margin: 10px;
  text-align: center;
```

```
  line-height: 75px;
  font-size: 30px;
}
```



Pionowe i poziome wyśrodkowanie

Elementy potomne

Elementy potomne również mają kilka przydatnych właściwości:

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