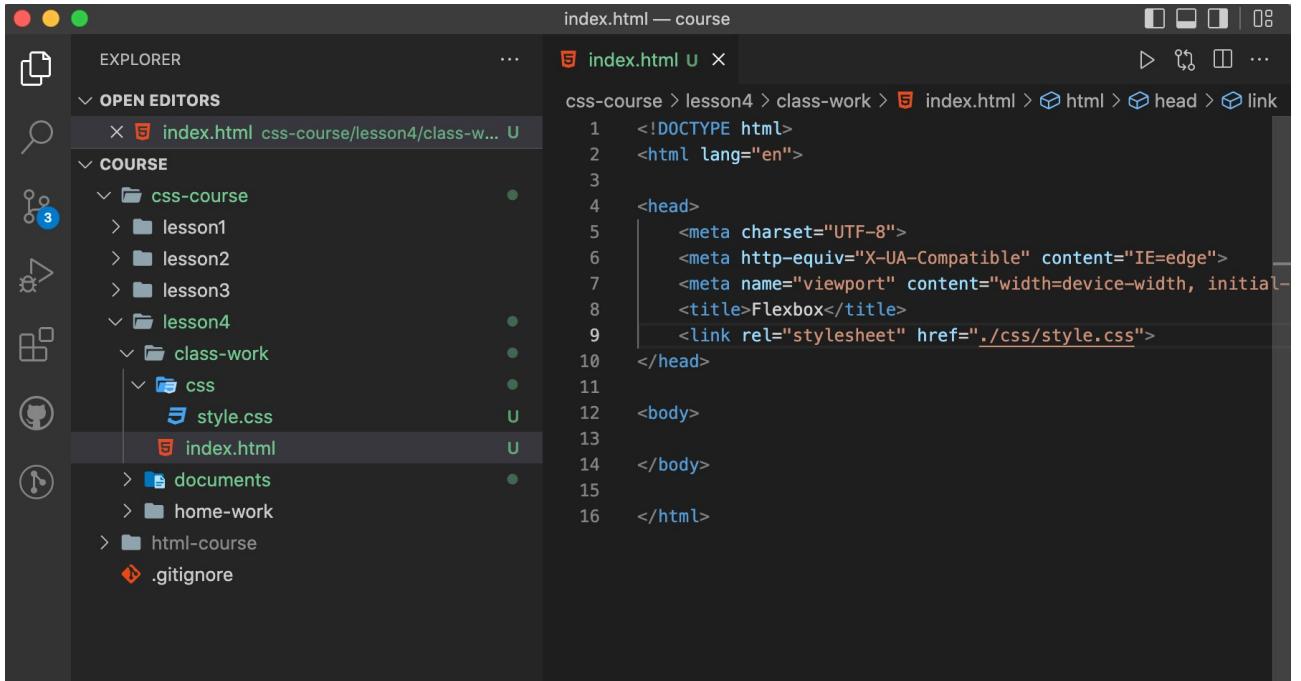


Вітаю всіх на 4-му уроці з вивчення CSS.

На цьому уроці ми розбираємося з такою важливою темою, як модель верстки — Flexbox(або флекси).

**CSS Flexible Box Layout Module** (CSS модуль для макетів з гнучкими блоками), коротко flexbox (флексбокс), створена, щоб прибрати недоліки при створенні різних HTML конструкцій, у тому числі адаптованих під різну ширину і висоту, і зробити верстку логічною і простою. Flexbox дозволяє елегантно контролювати різні параметри елементів всередині контейнера: напрям, порядок, ширину, висоту, вирівнювання вздовж і поперек, розподіл вільного місця, розтягування і стиснення елементів.

Почнемо наше заняття. Створюємо наступну структуру нашого уроку:



Створюємо контейнер і 4 поки що однакові елементи з рівними відступами з усіх боків.

```
1 .container {  
2     max-width: 600px;  
3     height: 600px;  
4     margin: 0 auto;  
5     border: 2px solid black;  
6     font-size: 40px;  
7     font-weight: bold;  
8 }  
9  
10 .item-1 {  
11     width: 100px;  
12     height: 100px;  
13     margin: 10px;  
14     background: #lightskyblue;  
15 }  
16  
17 .item-2 {  
18     width: 100px;  
19     height: 100px;  
20     margin: 10px;  
21     background: #lightskyblue;  
22 }  
23  
24 .item-3 {  
25     width: 100px;  
26     height: 100px;  
27     margin: 10px;  
28     background: #lightskyblue;  
29 }  
30  
31 .item-4 {  
32     width: 100px;  
33     height: 100px;  
34     margin: 10px 10px;  
35     background: #lightskyblue;  
36 }  
1 <!DOCTYPE html>  
2 <html lang="en">  
3  
4     <head>  
5         <meta charset="UTF-8">  
6         <meta http-equiv="X-UA-Compatible" content="IE=edge">  
7         <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">  
8         <title>Flexbox</title>  
9         <link rel="stylesheet" href="./css/style.css">  
10    </head>  
11  
12    <body>  
13  
14        <div class="container">  
15            <div class="item-1 ">1</div>  
16            <div class="item-2 ">2</div>  
17            <div class="item-3 ">3</div>  
18            <div class="item-4 ">4</div>  
19        </div>  
20    </body>  
21 </html>
```

Давайте змінимо властивість `display` у контейнері і пропишемо йому значення `flex` і також створимо окремий клас `.center` у якому ми вирівняємо по центру контент наших елементів та додамо цей клас кожному елементу, щоб усе виглядало рівно і красиво(ці властивості ми розберемо пізніше).

The screenshot shows a browser window with developer tools open. On the left, the CSS file 'style.css' contains the following code:

```

1 <div class="container" style="display: flex; justify-content: center; align-items: center;">
2   <div>1</div>
3   <div>2</div>
4   <div>3</div>
5   <div>4</div>
6 </div>
7 
```

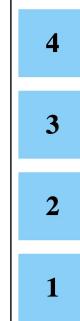
On the right, the browser preview shows four blue squares arranged horizontally, each containing a number from 1 to 4.

Ми бачимо, що наші елементи вишикувалися в горизонтальний рядок.

Далі для зручності я буду писати в який клас додавати які властивості і як при цьому змінюватиметься розташування наших елементів. Ви повинні додавати цей код і аналізувати зміни. Хочу зазначити, що за замовчуванням напрямок флексу — рядок (`flex-direction: row`)

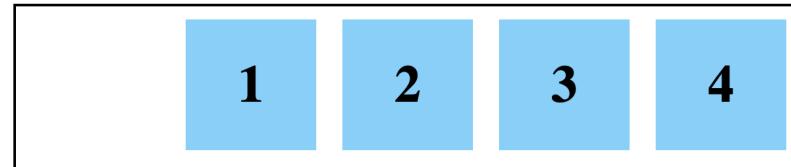
<pre> .container {   max-width: 600px;   height: 600px;   margin: 0 auto;   border: 2px solid black;   font-size: 40px;   font-weight: 800;   display: flex;   flex-direction: row-reverse; } </pre>	
<pre> .container {   max-width: 600px;   height: 600px;   margin: 0 auto;   border: 2px solid black;   font-size: 40px;   font-weight: 800;   display: flex;   flex-direction: column; } </pre>	

```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    flex-direction: column-reverse;
}
```



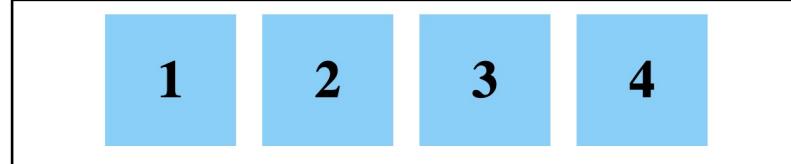
Властивість **justify-content** визначає вирівнювання вздовж головної осі. Воно допомагає розподілити додатковий залишок вільного простору, коли всі flex елементи в рядку негнучкі, або гнучкі, але досягли свого максимального розміру. Це також забезпечує певний контроль за вирівнюванням елементів, коли вони переповнюють лінію. За замовчуванням значення **justify-content: flex-start;**  
**justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right;**

```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    justify-content: flex-end;
}
```



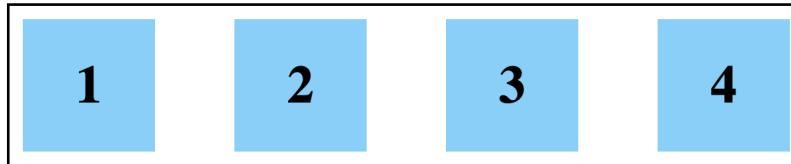
**flex-end:** елементи зсунуті близче до кінця flex-напрямку.

```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    justify-content: center;
}
```



**center:** елементи центровані вздовж лінії

```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    justify-content: space-between;
}
```



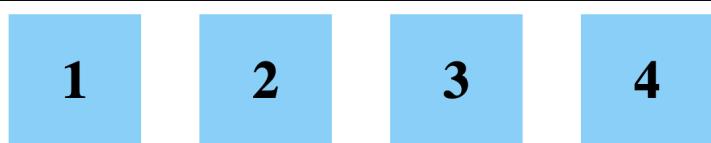
**space-between:** елементи рівномірно розподілені по лінії; перший елемент знаходиться на початку рядка, останній елемент в кінці рядка

```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    justify-content: space-around;  
}
```



**space-around**: елементи рівномірно розподілені по лінії з однаковим простором навколо них. Зверніть увагу, що візуально простири не рівні, тому що всі елементи мають одинаковий простір з обох боків.

```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    justify-content: space-evenly;  
}
```



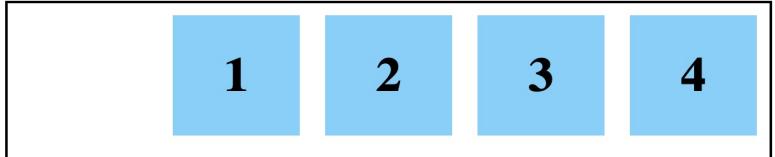
**space-evenly**: елементи розподіляються таким чином, щоб відстань між будь-якими двома елементами (і відстань до країв) була однаковою.

```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    justify-content: start;  
}
```



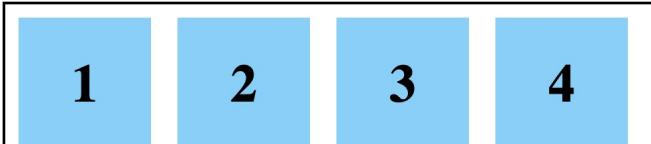
**start**: елементи зміщені до початку напряму.

```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    justify-content: end;  
}
```



**end**: елементи зсунуті наприкінці напряму.

```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    justify-content: left;  
}
```



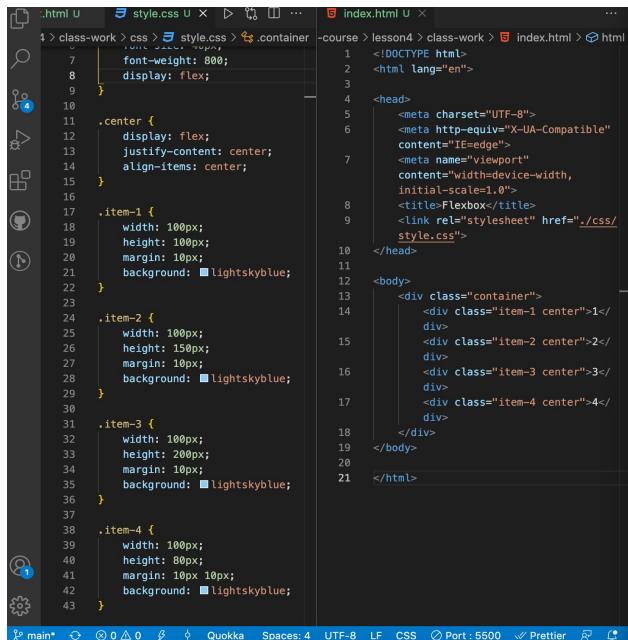
**left**: елементи зсунуті до лівого краю контейнера, якщо це не має сенсу flex-direction, тоді він веде себе як **start**.

```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    justify-content: right;
}
```

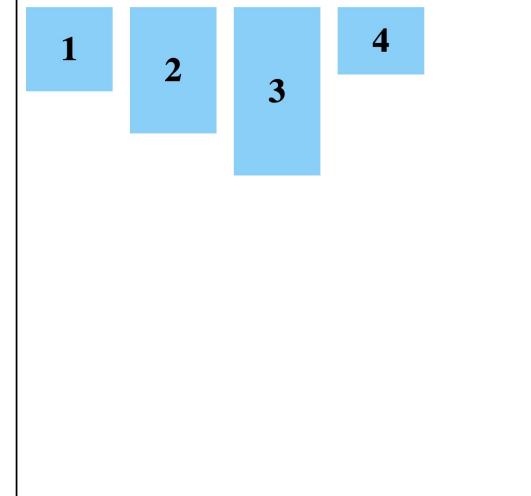
1 2 3 4

**right**: елементи зсунуті до правого краю контейнера, якщо це не має сенсу flex-direction, тоді він веде себе як start.

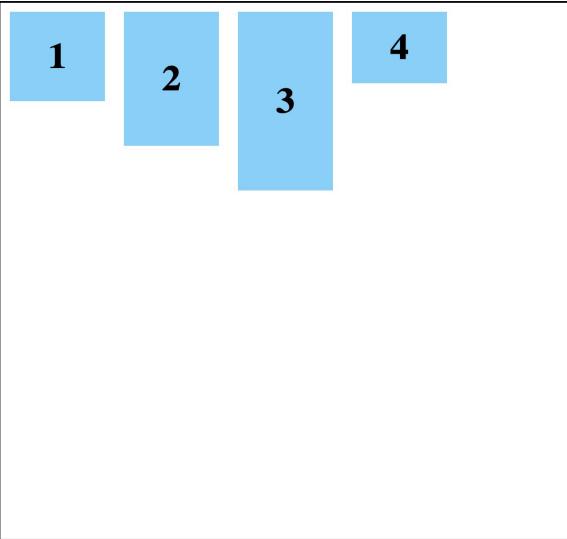
Властивість **align-items** визначає поведінку за умовчанням того, як flex елементи розташовуються вздовж поперечної осі поточної лінії. Давайте видалимо властивість **justify-content** і змінимо висоту наших блоків.



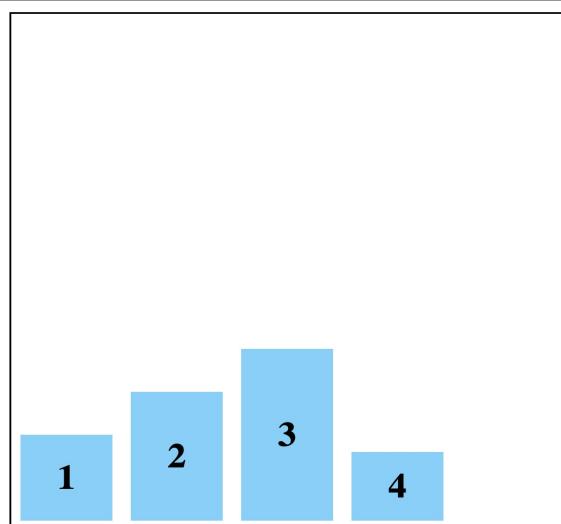
```
class-work > css > style.css > .container
course > lesson4 > class-work > index.html > html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Flexbox</title>
<link rel="stylesheet" href="./css/style.css">
</head>
<body>
<div class="container">
<div class="item-1 center">1</div>
<div class="item-2 center">2</div>
<div class="item-3 center">3</div>
<div class="item-4 center">4</div>
</div>
</body>
</html>
```



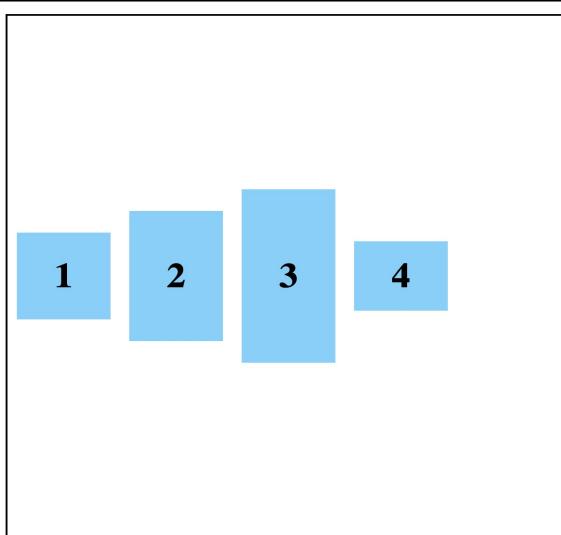
```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    align-items: flex-start;
}
```



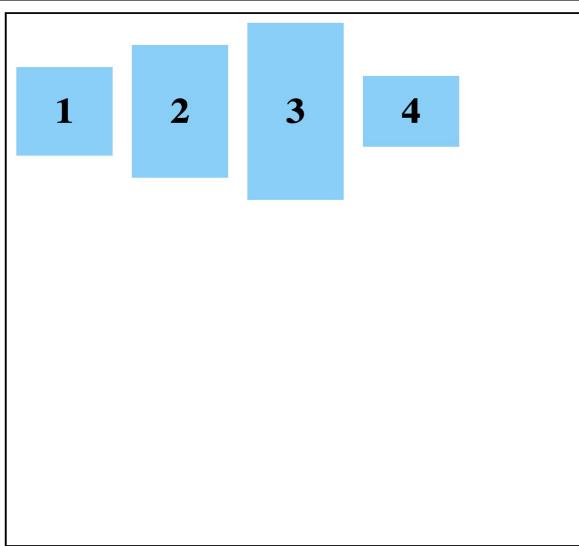
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    align-items: flex-end;  
}
```



```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    align-items: center;  
}
```

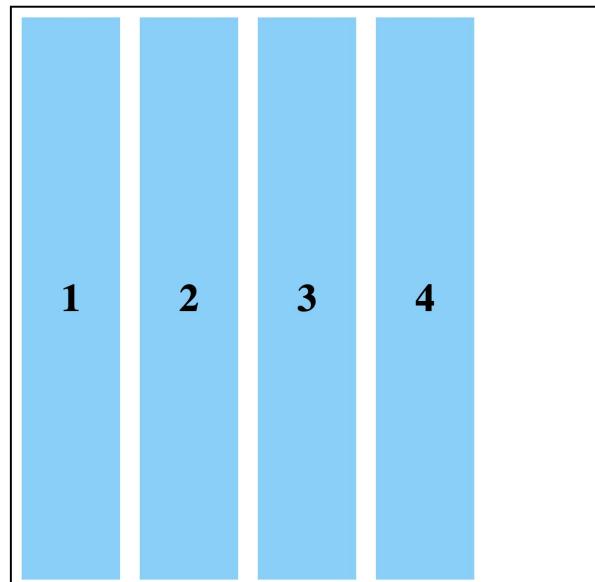


```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    align-items: baseline;  
}
```



```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    align-items: stretch;
}
```

**\*\*** Закоментуйте в усіх елементах item властивість height

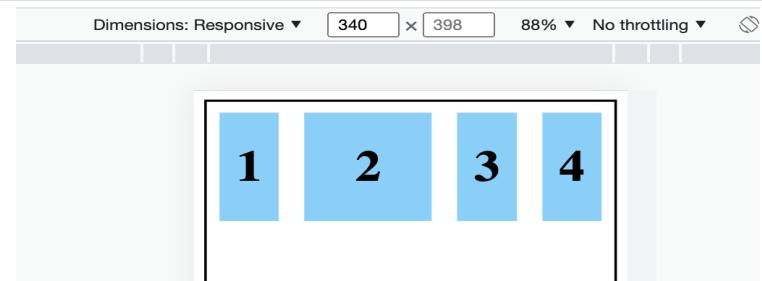


Видалимо align-items: stretch; і розкоментуємо в усіх елементах item властивість height та зробимо їх однаковими 100px.

Давайте розберемо наступну властивість — **flex-wrap**

Спочатку давайте включимо інструменти розробника і переключимося на мобільну верстку

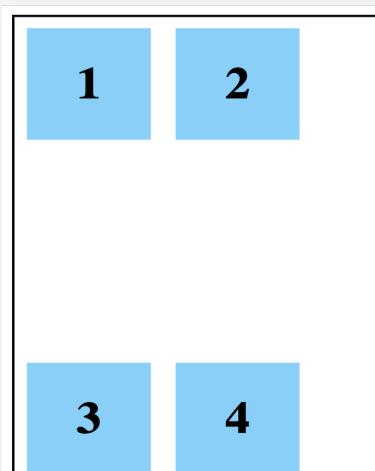
```
.container {
    max-width: 600px;
    height: 600px;
    margin: 0 auto;
    border: 2px solid black;
    font-size: 40px;
    font-weight: 800;
    display: flex;
    flex-wrap: nowrap;
```



**nowrap** (за замовчуванням): всі flex елементи будуть в одному рядку коли ми звузимо наш контейнер

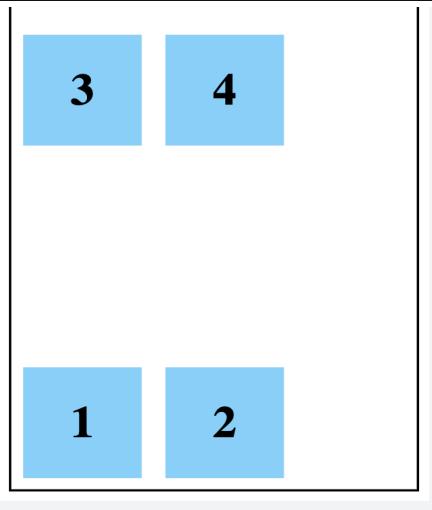
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
}
```

Dimensions: Responsive ▾ 313 × 560 88% ▾ No throttling ▾



**wrap**: flex-елементи буде перенесено на кілька рядків зверху вниз коли ми звузимо наш контейнер

```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap-reverse;  
}
```



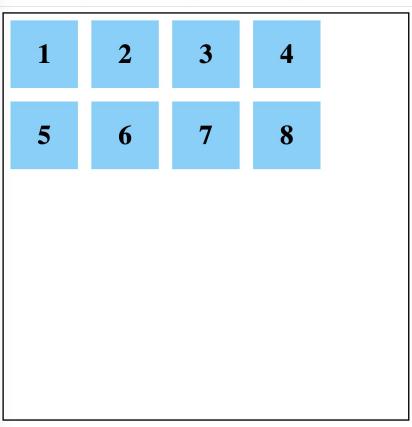
**wrap-reverse**: flex-елементи буде перенесено на кілька рядків знизу вгору коли ми звузимо наш контейнер

### Наступна властивість - align-content .

Властивість **align-content** вирівнює лінії в межах flex контейнера, коли є додатковий простір на поперечній осі, подібно до того, як **justify-content** вирівнює окремі елементи в межах головної осі.

**align-content**: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline

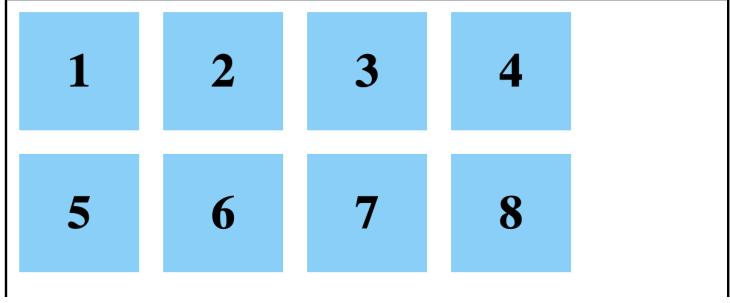
**Примітка**: ця властивість не діє, коли є лише один рядок flex елементів, тому додамо ще один рядок і трохи спростимо наш код.



A screenshot of a browser window showing a 2x4 grid of numbered items (1-8) in light blue boxes. The browser tabs at the top include 'Первое задание...', 'Книги на англий...', 'The 3588 Most C...', and 'Учебательный А...'. The address bar shows the URL. The code editor on the left displays the CSS and HTML files used to create the grid.

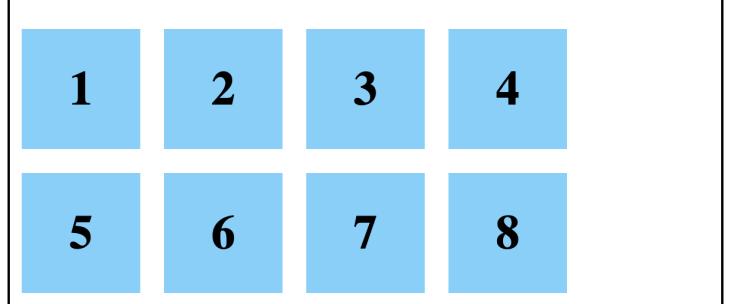
```
on4 class-work > style.css > container > lesson4 > class-work > index.html > body
1 .container {
2   max-width: 600px;
3   height: 600px;
4   margin: 0 auto;
5   border: 2px solid black;
6   font-size: 40px;
7   font-weight: 800;
8   display: flex;
9   flex-wrap: wrap;
10  align-content: flex-start;
11 }
12 .center {
13   display: flex;
14   justify-content: center;
15   align-items: center;
16 }
17 .row {
18   display: flex;
19 }
20 .item {
21   width: 100px;
22   height: 100px;
23   margin: 10px;
24   background: #lightskyblue;
25 }
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>flexbox</title>
    <link rel="stylesheet" href=".css/style.css"/>
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="item center">1</div>
        <div class="item center">2</div>
        <div class="item center">3</div>
        <div class="item center">4</div>
      </div>
      <div class="row">
        <div class="item center">5</div>
        <div class="item center">6</div>
        <div class="item center">7</div>
        <div class="item center">8</div>
      </div>
    </div>
  </body>
</html>
```



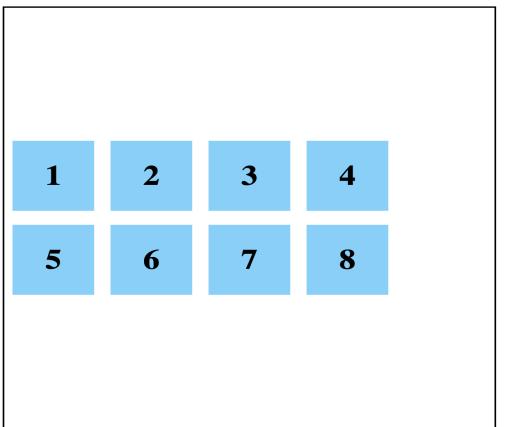
A screenshot of a browser window showing a 2x4 grid of numbered items (1-8) in light blue boxes. The browser tabs at the top include 'Первое задание...', 'Книги на англий...', 'The 3588 Most C...', and 'Учебательный А...'. The address bar shows the URL. The code editor on the left displays the CSS and HTML files used to create the grid.

```
.container {
  max-width: 600px;
  height: 600px;
  margin: 0 auto;
  border: 2px solid black;
  font-size: 40px;
  font-weight: 800;
  display: flex;
  flex-wrap: wrap;
  align-content: flex-start;
}
```



A screenshot of a browser window showing a 2x4 grid of numbered items (1-8) in light blue boxes. The browser tabs at the top include 'Первое задание...', 'Книги на англий...', 'The 3588 Most C...', and 'Учебательный А...'. The address bar shows the URL. The code editor on the left displays the CSS and HTML files used to create the grid.

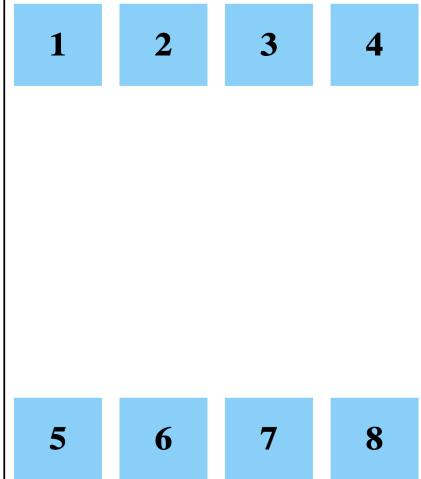
```
.container {
  max-width: 600px;
  height: 600px;
  margin: 0 auto;
  border: 2px solid black;
  font-size: 40px;
  font-weight: 800;
  display: flex;
  flex-wrap: wrap;
  align-content: flex-end;
}
```



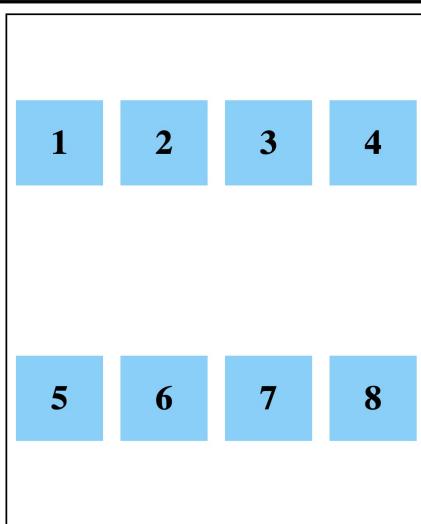
A screenshot of a browser window showing a 2x4 grid of numbered items (1-8) in light blue boxes. The browser tabs at the top include 'Первое задание...', 'Книги на англий...', 'The 3588 Most C...', and 'Учебательный А...'. The address bar shows the URL. The code editor on the left displays the CSS and HTML files used to create the grid.

```
.container {
  max-width: 600px;
  height: 600px;
  margin: 0 auto;
  border: 2px solid black;
  font-size: 40px;
  font-weight: 800;
  display: flex;
  flex-wrap: wrap;
  align-content: center;
}
```

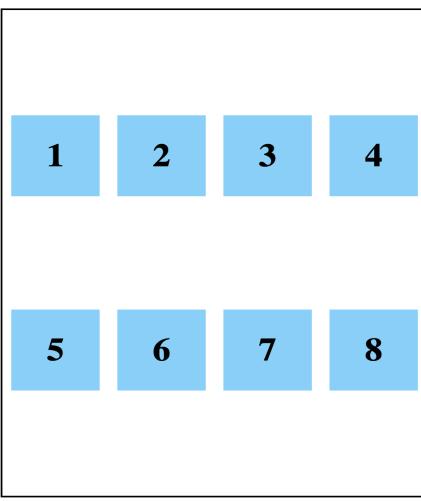
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: space-between;  
}
```



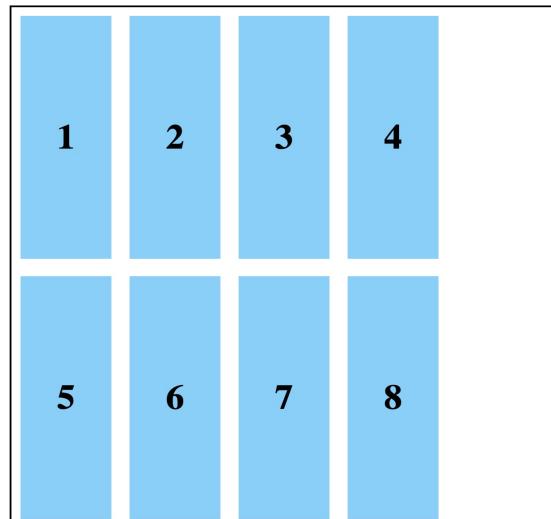
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: space-around;  
}
```



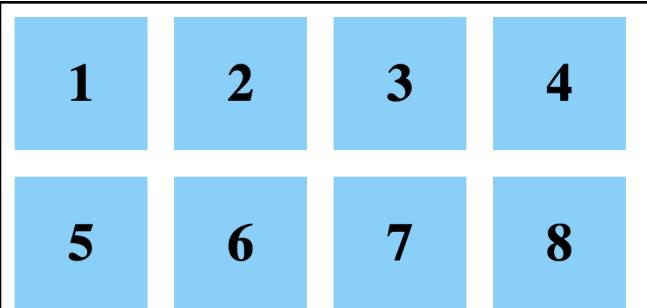
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: space-evenly;  
}
```



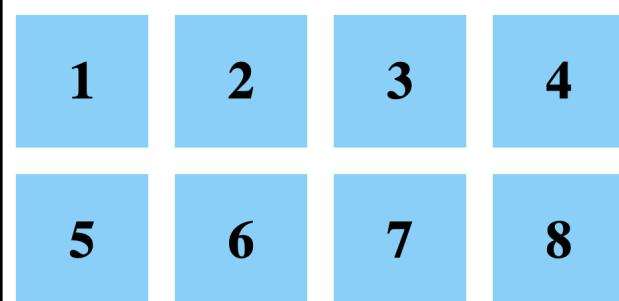
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: stretch;  
}  
  
.item {  
    width: 100px;  
    margin: 10px;  
    background: lightskyblue;  
}
```



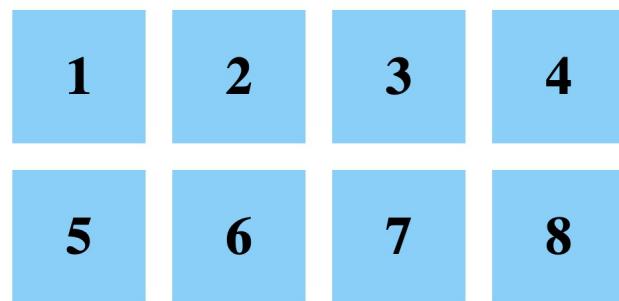
```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: start;  
}
```



```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: end;  
}
```



```
.container {  
    max-width: 600px;  
    height: 600px;  
    margin: 0 auto;  
    border: 2px solid black;  
    font-size: 40px;  
    font-weight: 800;  
    display: flex;  
    flex-wrap: wrap;  
    align-content: baseline;  
}
```



## Властивості для перших дочірніх елементів.

### - order

Властивість **order** контролює порядок, в якому елементи можуть розташовуватися. За замовчанням для кожного елемента **order: 0**; . Значення може бути як зі знаком **+**, так і зі знаком **-**. Давайте додамо класи для 1-го та 4-го елементів та змінимо їх порядок

```

index.html U style.css U ...
rse > lesson4 > class-work > css > style.css > item-1.css
1 .container {
2   max-width: 600px;
3   height: 100px;
4   margin: 0 auto;
5   border: 2px solid black;
6   font-size: 40px;
7   font-weight: 800;
8   display: flex;
9   flex-wrap: wrap;
10  align-content: baseline;
11 }
12
13 .center {
14   display: flex;
15   justify-content: center;
16   align-items: center;
17 }
18
19 .row {
20   display: flex;
21 }
22
23 .item {
24   width: 100px;
25   height: 100px;
26   margin: 10px;
27   background: lightskyblue;
28 }
29
30 .item-1 {
31   order: 1;
32 }
33
34 .item-4 {
35   order: 2;
36 }

```

```

css-course > lesson4 > class-work > index.html > style.css
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5     <title>Flexbox</title>
6     <link rel="stylesheet" href="./css/style.css">
7   </head>
8
9   <body>
10    <div class="container">
11      <div class="row">
12        <div class="item center item-1">1</div>
13        <div class="item center item-2">2</div>
14        <div class="item center item-3">3</div>
15        <div class="item center item-4">4</div>
16      </div>
17      <div class="row">
18        <div class="item center">5</div>
19        <div class="item center">6</div>
20        <div class="item center">7</div>
21        <div class="item center">8</div>
22      </div>
23    </div>
24  </body>
25
26
27
28
29 </html>

```

### - flex-grow

Ця властивість визначає здатність flex елементів за необхідності ставати більшими. Вона набуває безрозмірного значення, яке є пропорцією або часткою. Вона вказує скільки вільного місця всередині контейнера елемент повинен взяти. Якщо всі елементи в контейнері мають flex-grow зі значенням 1, це означає те, що місце в ньому розподілене в рівній мірі серед нащадків.

Щоб ця властивість почала працювати, потрібно задати ширину батьківського блоку **row**. Давайте додамо до 1-го та 4-го елементів **flex-grow: 2;** . Ми бачимо, що ці елементи стали ширшими за інші.

```

index.html U style.css U ...
rse > lesson4 > class-work > css > style.css > item-1.css
1 .container {
2   max-width: 600px;
3   height: 100px;
4   margin: 0 auto;
5   border: 2px solid black;
6   font-size: 40px;
7   font-weight: 800;
8   display: flex;
9   flex-wrap: wrap;
10  align-content: baseline;
11 }
12
13 .center {
14   display: flex;
15   justify-content: center;
16   align-items: center;
17 }
18
19 .row {
20   display: flex;
21   width: 600px;
22 }
23
24 .item {
25   width: 100px;
26   height: 100px;
27   margin: 10px;
28   background: lightskyblue;
29 }
30
31 .item-1 {
32   order: 1;
33   flex-grow: 2;
34 }
35
36 .item-4 {
37   order: 2;
38   flex-grow: 2;
39 }

```

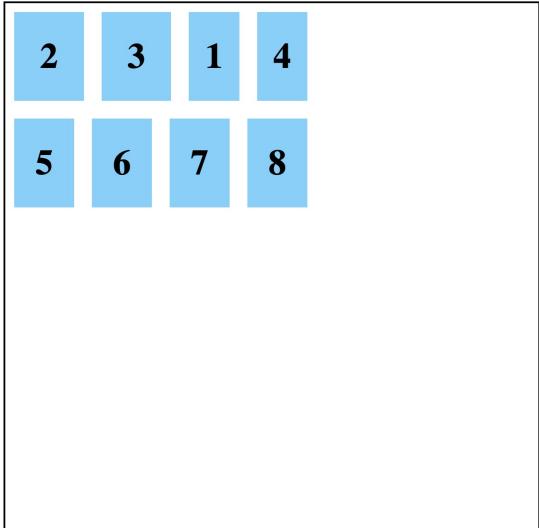
```

css-course > lesson4 > class-work > index.html > style.css
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta name="viewport" content="width=device-width, initial-scale=1.0">
5     <title>Flexbox</title>
6     <link rel="stylesheet" href="./css/style.css">
7   </head>
8
9   <body>
10    <div class="container">
11      <div class="row">
12        <div class="item center item-1">1</div>
13        <div class="item center item-2">2</div>
14        <div class="item center item-3">3</div>
15        <div class="item center item-4">4</div>
16      </div>
17      <div class="row">
18        <div class="item center">5</div>
19        <div class="item center">6</div>
20        <div class="item center">7</div>
21        <div class="item center">8</div>
22      </div>
23    </div>
24  </body>
25
26
27
28
29 </html>

```

## - flex-shrink

Ця властивість визначає здатність flex елемента при необхідності скорочуватися. Давайте додамо до 1-го та 4-го елементів **flex-shrink: 2**; і зменшимо ширину батьківського блоку до 350px

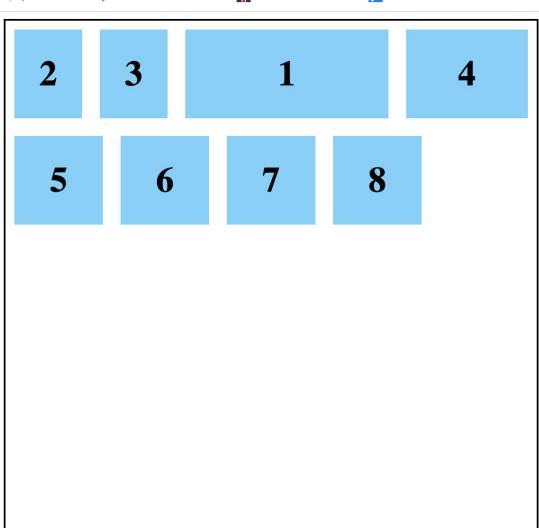


```
lesson4 > class-work > css > style.css > .row > index.html > html > body > div.container
  7   font-weight: 800;
  8   display: flex;
  9   flex-wrap: wrap;
 10  align-content: baseline;
 11 }
 12
 13 .center {
 14   display: flex;
 15   justify-content: center;
 16   align-items: center;
 17 }
 18
 19 .row {
 20   display: flex;
 21   width: 350px;
 22 }
 23
 24 .item {
 25   width: 100px;
 26   height: 100px;
 27   margin: 10px;
 28   background: lightskyblue;
 29 }
 30
 31 .item-1 {
 32   order: 1;
 33   flex-shrink: 2;
 34 }
 35
 36 .item-4 {
 37   order: 2;
 38   flex-shrink: 2;
 39 }
```

## - flex-basis

Ця властивість визначає розмір елемента за замовчанням перед розподілом простору, що залишився. Це може бути довжина (наприклад, 20%, 5rem і т.д.) або ключове слово auto означає "дивися на мою width або height властивість".

Давайте зробимо знов ширину батьківського блоку 600px а для 1-го та 4-го елементів додамо різні значення **flex-basis**



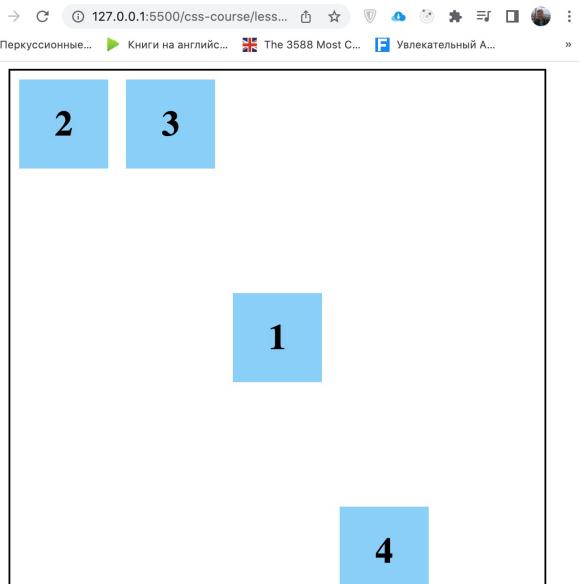
```
son4 > class-work > css > style.css > .item-4 > index.html > html > body > div.container
  7   font-weight: 800;
  8   display: flex;
  9   flex-wrap: wrap;
 10  align-content: baseline;
 11 }
 12
 13 .center {
 14   display: flex;
 15   justify-content: center;
 16   align-items: center;
 17 }
 18
 19 .row {
 20   display: flex;
 21   width: 600px;
 22 }
 23
 24 .item {
 25   width: 100px;
 26   height: 100px;
 27   margin: 10px;
 28   background: lightskyblue;
 29 }
 30
 31 .item-1 {
 32   order: 1;
 33   flex-basis: 50%;
 34 }
 35
 36 .item-4 {
 37   order: 2;
 38   flex-basis: 30%;
 39 }
```

## - align-self

Ця властивість дозволяє перевизначити вирівнювання за замовчуванням (або зазначене за допомогою align-items) окремих елементів flex.

**align-self:** auto | flex-start | flex-end | center | baseline | stretch;

Давайте закоментуємо 2-й рядок елементів, додамо висоту для блоку row 600px і додамо для 1-го та 4-го елементів різні значення align-self



```
lesson4 > class-work > css > style.css > .row
  7   font-weight: 800;
  8   display: flex;
  9   flex-wrap: wrap;
 10  align-content: baseline;
 11 }
 12
 13 .center {
 14   display: flex;
 15   justify-content: center;
 16   align-items: center;
 17 }
 18
 19 .row {
 20   display: flex;
 21   width: 600px;
 22   height: 600px;
 23 }
 24
 25 .item {
 26   width: 100px;
 27   height: 100px;
 28   margin: 10px;
 29   background: lightskyblue;
 30 }
 31
 32 .item-1 {
 33   order: 1;
 34   align-self: center;
 35 }
 36
 37 .item-4 {
 38   order: 2;
 39   align-self: flex-end;
 40 }
```

\*\* Додатково. Є ще одна скорочена властивість, яка має назву **flex-flow**. Ця властивість об'єднує такі дві властивості, як **flex-direction** та **flex-wrap**. Наприклад, ви можете застосувати **flex-flow: column wrap** для встановлення стовпців та перенесення їх.

Рекомендую для закрілення знань погратися на цьому ресурсі:

<https://flexboxfroggy.com>

До зустрічі на наступному уроці.