18일 (일)

1. Codecademy - Google Chrome Devtool - Simulate mobile devices

https://developer.chrome.com/docs/devtools/device-mode/

2. Understanding the Fundamentals of Responsive Design

https://www.taniarascia.com/you-dont-need-a-framework/

3. Codecademy Making a Website Responsive - FlexBox

- **What is Flexbox?**

플렉스박스는 태그들을 자유자재로 이동시킬 수 있는 강력한 css 툴이다.

플렉스박스 레이아웃에는 두가지의 중요한 컴포넌트들이 있다. (flex containers, flex items) flex container 은 flex item 을 포함하고 있다. 즉, 자식요소 인 것이다.

하나의 요소를 플렉스 컨테이너로 지정하기 위해, display : flex or inline-flex

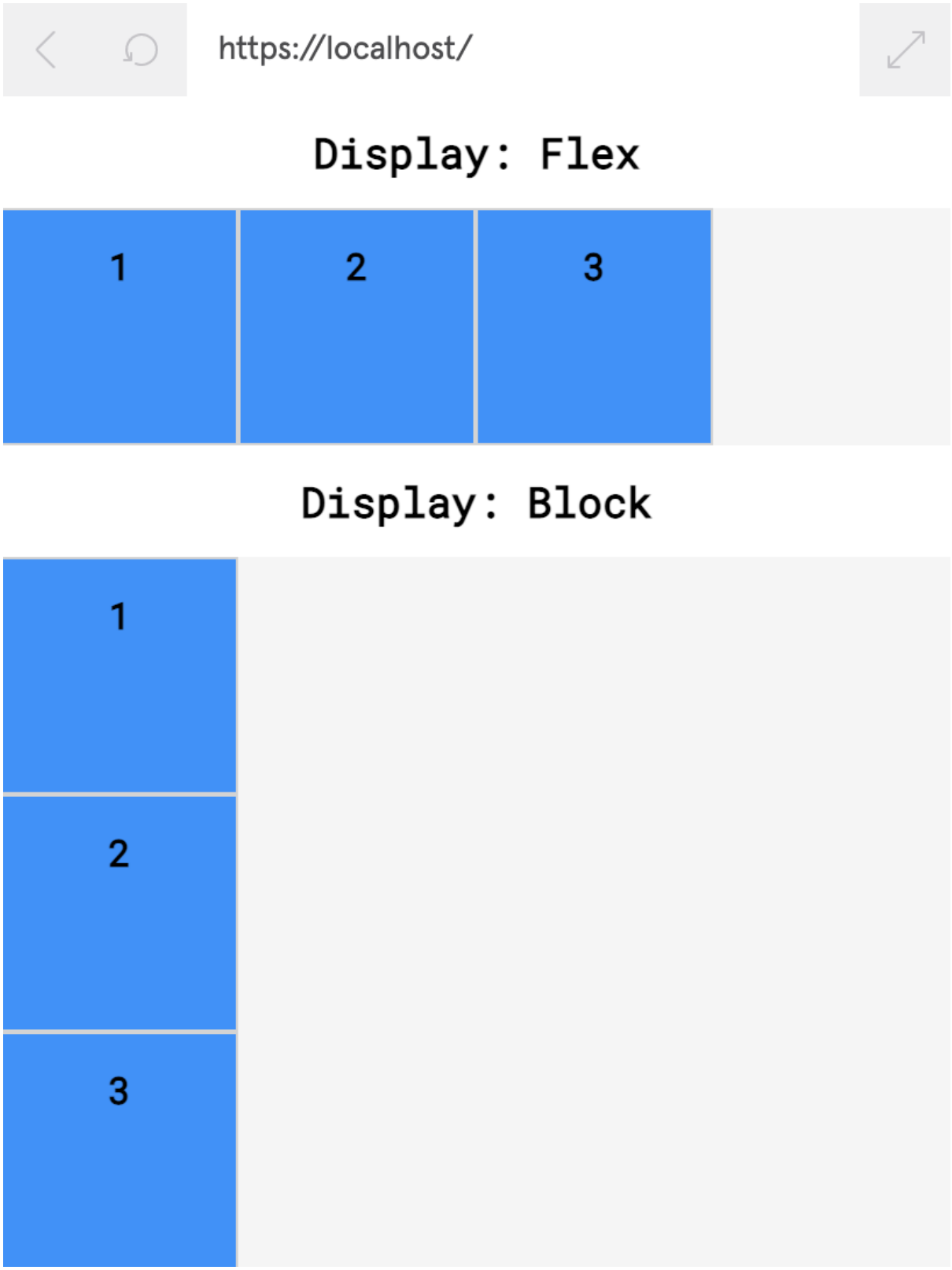
1. `justify-content`
2. `align-items`
3. `flex-grow`
4. `flex-shrink`
5. `flex-basis`
6. `flex`
7. `flex-wrap`
8. `align-content`
9. `flex-direction`
10. `flex-flow`

- **display : flex (flex container 룰셋에 적용)**

어떤 요소든 간에 flex container 가 될 수 있다. flex container 은 반응형 웹페이지를 쉽게 구성할 수 있게 한다.

flex 컨테이너의 자식 태그들(flex items)은 컨테이너에 따라서 크기가 바뀌고, 위치가 바뀐다.

이렇게 flex 가 지정되면 그 요소는 자동으로 block level 이 된다.


https://localhost/
Display: Flex
1
2
3
Display: Block
1
2
3

- **inline-flex (flex container 룰셋에 적용)**

만약 flex 로 변환 뒤, 컨테이너를 block level 로 남기고 싶지 않다면, inline-flex 를 사용하자.

```
<div class='container'>
  <p>I’m inside of a flex container!
</p>
  <p>A flex container’s children are
flex items!</p>
</div>
<div class='container'>
  <p>I’m also a flex item!</p>
  <p>Me too!</p>
</div>
```

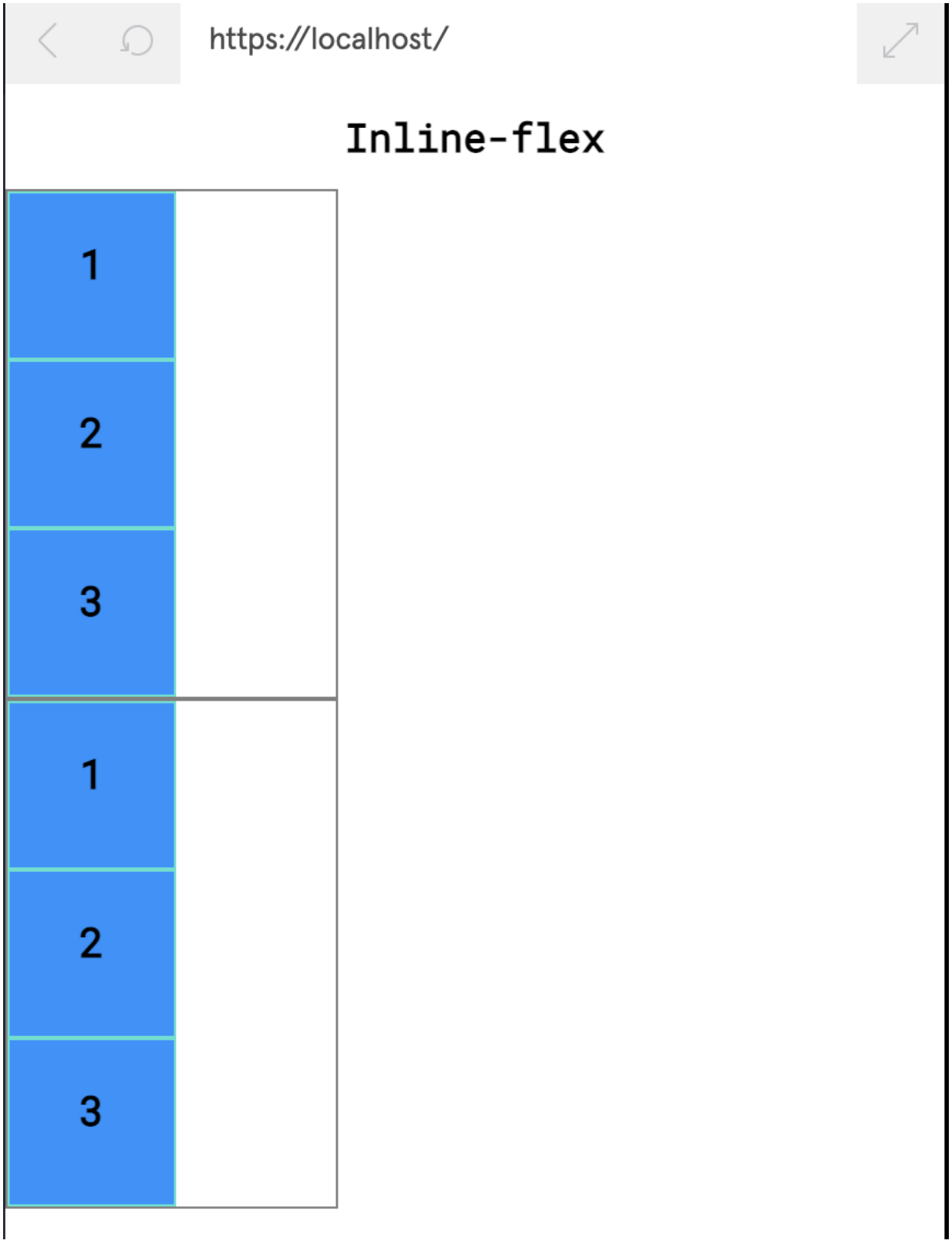
```
.container {
  width: 200px;
  height: 200px;
  display: inline-flex;
}
```

위 예시에서 컨테이너의 사이즈가 지정되었다. ㅇ현재 부모 컨테이너의 사이즈는 그것의 자식 요소들의 크기를 덮어씌울 것이다. 만약 부모 컨테이너의 크기가 너무 작으면, 자식요소인 플렉스 아이템들은 부모컨테이너 사이즈에 맞게 줄어들 것이다.

```html
<div class='container'>
  <div class='child'>
    <h1>1</h1>
  </div>
  <div class='child'>
    <h1>2</h1>
  </div>
</div>
```

```css
.container {
  width: 200px;
}

.child {
  display: inline-flex;
  width: 150px;
  height: auto;
}
```

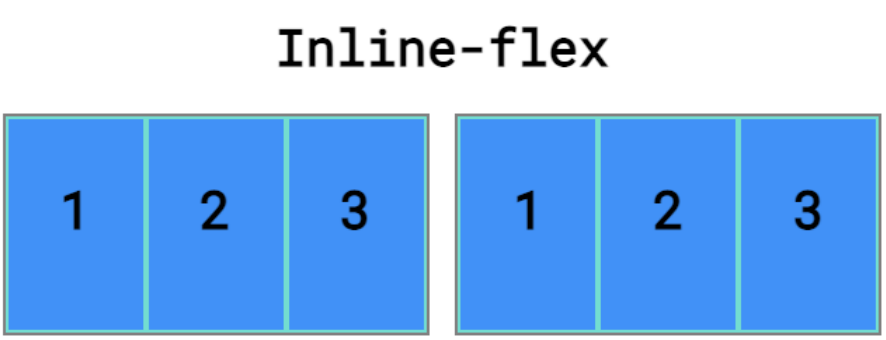
예시



inline-flex 가 적용이 안되어 현재 각 컨테이너가 block-level 상태이다.

```
12
13  .container {
14      width: 150px;
15      border: 1px solid grey;
16
17    }
18
19  .box {
20      background-color: dodgerblue;
21      height: 75px;
22      width: 75px;
23      border: 1px solid turquoise;
24    }
25
```

https://localhost/



```
12
13  .container {
14      width: 150px;
15      border: 1px solid grey;
16      display: inline-flex;
17    }
18
19  .box {
20      background-color: dodgerblue;
21      height: 75px;
22      width: 75px;
23      border: 1px solid turquoise;
24    }
25
```

display : inline-flex 추가

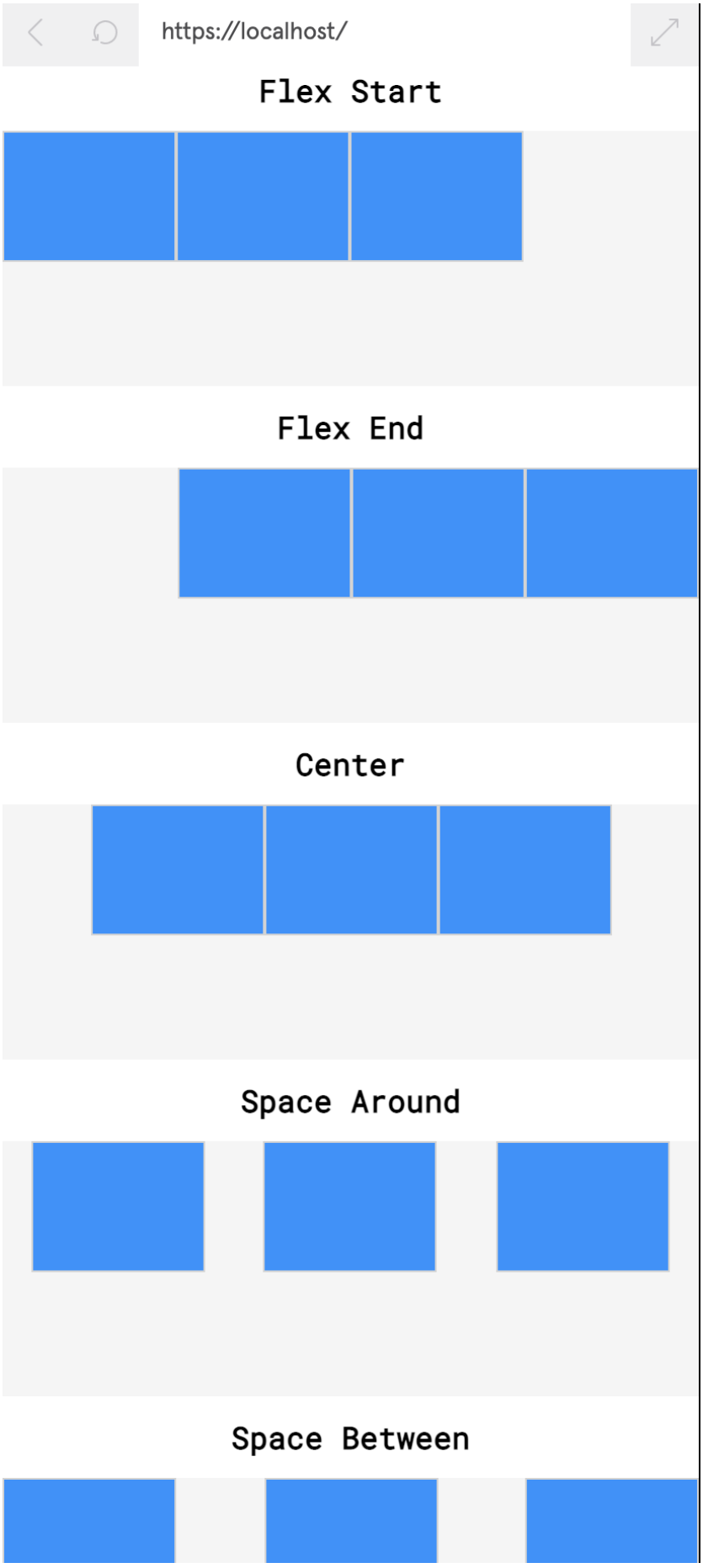
- **justify-content (아이템들의 가로축 이동) (flex container 룰셋에 적용)**

display 프로퍼티가 flex 나 inline-flex로 변하게 될 때, 모든 자식 item들은 컨테이너의 왼쪽 구석 위로 이동하게 된다. 이것이 flex 의 디폴트 행위이다. 우리는 flex item들이 어떻게 펼쳐지게 할 지 메인 축을 정할 수 있다. 아이템들을 왼쪽에서부터 오른쪽으로 위치시키기 위해 justify-content 프로퍼티를 사용할 것이다.

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

```
28  #flexstart {
29      justify-content: flex-start;
30    }
31
32  #flexend {
33      justify-content: flex-end;
34    }
35
36  #center {
37      justify-content: center;
38    }
39
40  #spacearound {
41      justify-content: space-around;
42    }
43
44  #spacebetween {
45      justify-content: space-between;
46    }
47
```

5가지가 있다.

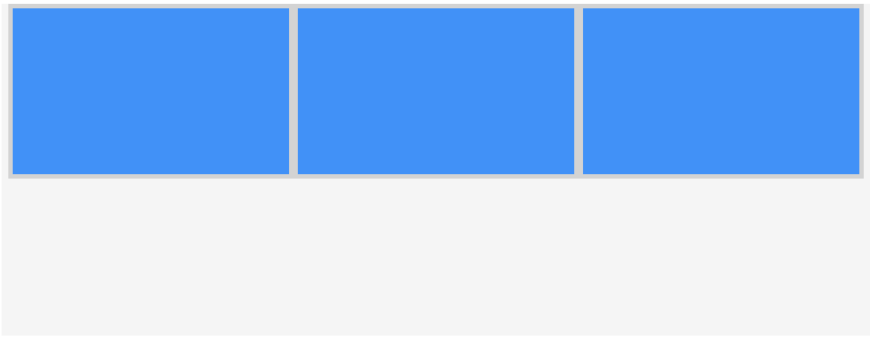

https://localhost/
Flex Start
Flex End
Center
Space Around
Space Between

- **align-items (아이템들의 세로축 이동) (flex container 룰셋에 적용)**

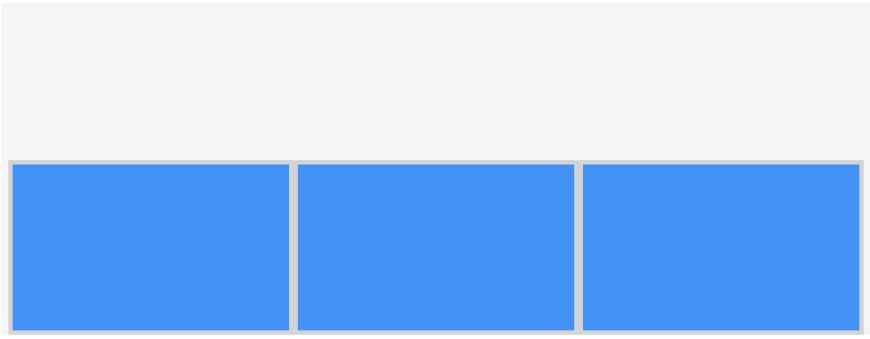
align-items 프로퍼티는 flex items 을 수직적으로 위치시킬 수 있다.

```
  .left,
  .center,
  .right {
    min-height: 75px;
    width: 125px;
    background-color: dodgerblue;
    border: 2px solid lightgrey;
  }

#baseline .center {
    height: 100px;
    width: 100px;
    border: 5px solid turquoise;
  }

#flexstart {
    align-items: flex-start;
  }

#flexend {
    align-items: flex-end;
  }

#center {
    align-items: center;
  }

#baseline {
    align-items: baseline;
  }

```

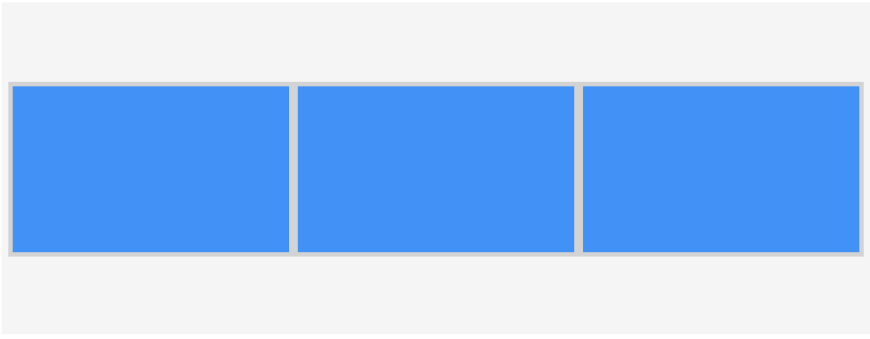
Flex Start



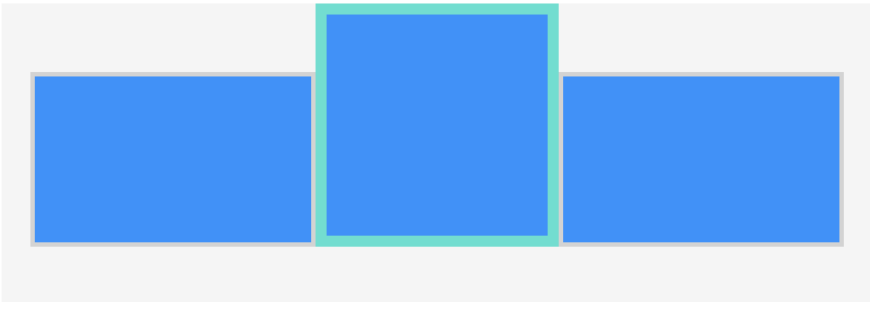
Flex End



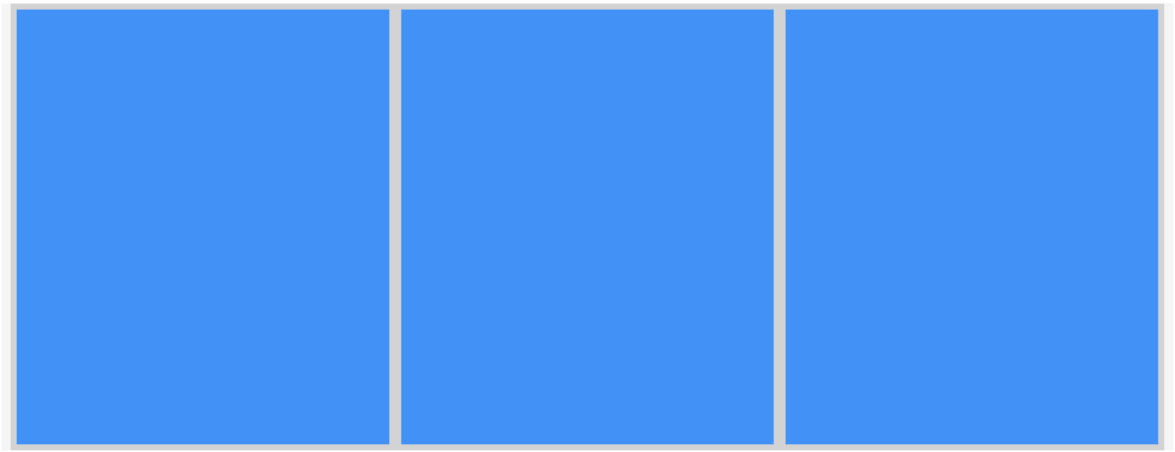
Center



Baseline



Stretch

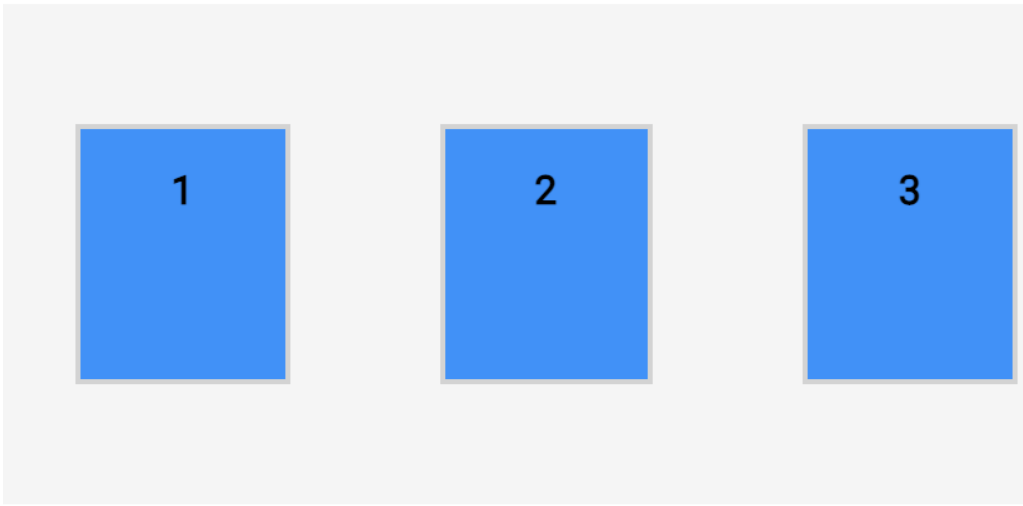


주의할 점 !!

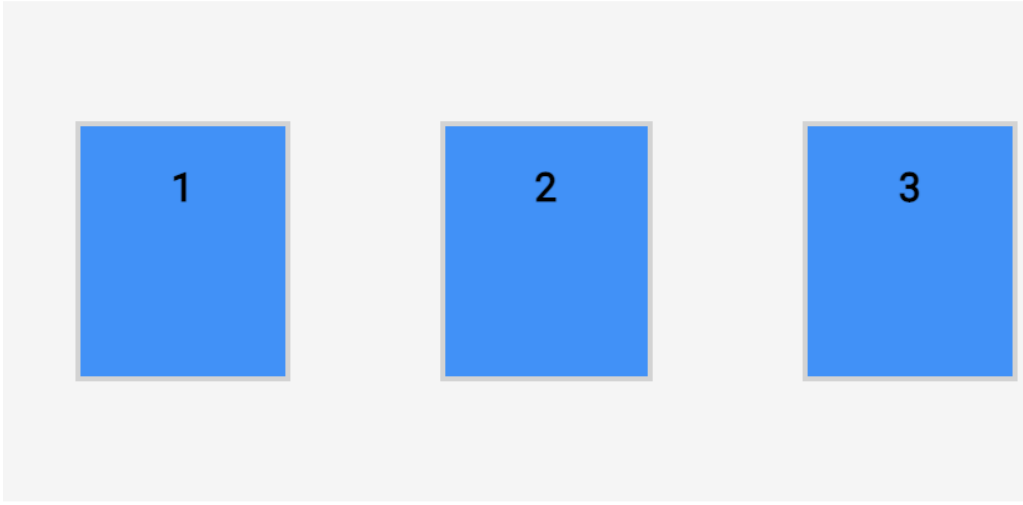
stretch 는 박스의 min-height을 지정해야 적용이 된다 !!

- **flex-grow (아이템 가로로 늘리기) (flex item 룰셋에 적용)**

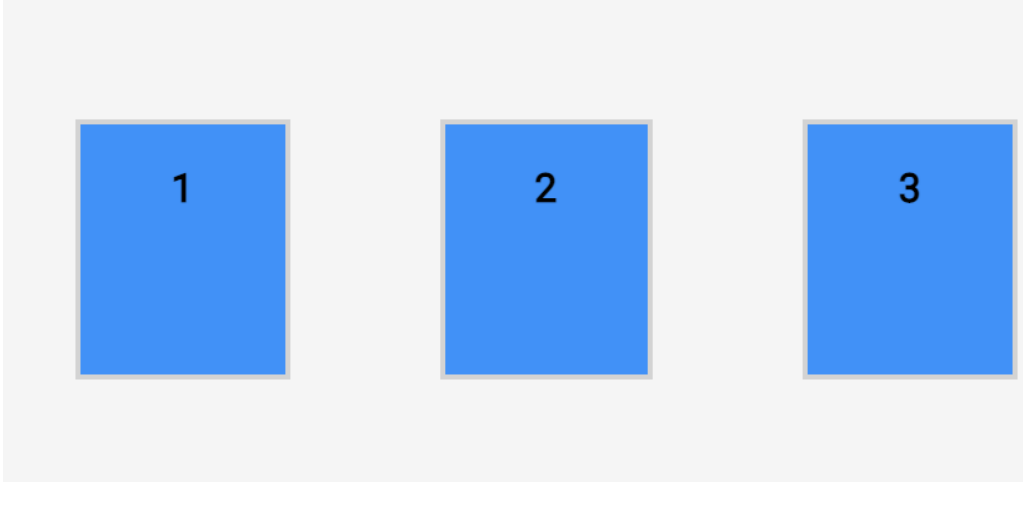
Step 1



Step 2



Step 3



```
29
30  .top.side {
31      flex-grow: 1;
32    }
33
34  .top.center {
35      flex-grow: 1;
36    }
37
38    .middle.side {
39    }
40
41  .middle.center {
42      flex-grow: 1;
43    }
44
45  .bottom.side {
46      flex-grow: 1;
47    }
48
49  .bottom.center {
50      flex-grow: 2;
51    }
52
```

만약 60px 의 추가 공간이 있고, side 가 1, center가 2 라면 center 는 30px , side 는 각각 15px 의 크기로 늘어나게 된다. 즉, 숫자는 비율과 같다.

만약 max-width 가 요소에 정해진다면, 공간이 늘어날지라도 박스는 더이상 늘어나지 않는다.

- **flex-shrink (아이템 가로로 줄이기) (flex items 에 적용)**

item 의 flex-shrink 의 디폴트 값은 1 이다. 즉 지정을 안해도 자동으로 1의 값을 갖는다. 반면에 flex-grow 는 디폴트값을 0을 갖으며, 지정하지 않으면 늘어나지 않는다.

flex-grow와 flex-shrink 는 min-width, max-width 의 설정이 선행되어애ㅑ 한다는 것을 명심해라.

- **flex-basis (아이템의 너비 정하기) (flex-items 에 적용)**

flex item 의 width 을 정하는 또 다른 방법은 프로퍼티 flex-basis 를 사용하는 것이다. 넓이 지정은 px 를 쓴다.

```
.container {
  display: flex;
}

.side {
  flex-grow: 1;
  flex-basis: 100px;
}

.center {
  flex-grow: 2;
  flex-basis: 150px;
}
```

- **flex (flex item 에 적용)**

flex 프로퍼티는 flex-grow, flex-shrink, flex-basis 를 한 줄에 쓰게 해주는 숄핸드 프로퍼티이다.

```css
.big {
  flex-grow: 2;
  flex-shrink: 1;
  flex-basis: 150px;
}

.small {
  flex-grow: 1;
  flex-shrink: 2;
  flex-basis: 100px;
}
```

```css
.big {
  flex: 2 1 150px;
}

.small {
  flex: 1 2 100px;
}
```

위 커맨드를 이와같이 짧게 바꿀 수 있다.

```css
.big {
 flex: 2 1;
}
```

grow 와 shrink 를 적용

```css
.small {
  flex: 1 20px;
}
```

grow 와 basis 를 적용

flex 프로퍼티로 shrink 와 basis 만 정하는 방법은 없으므로 따로 프로퍼티를 적어줘야한다.

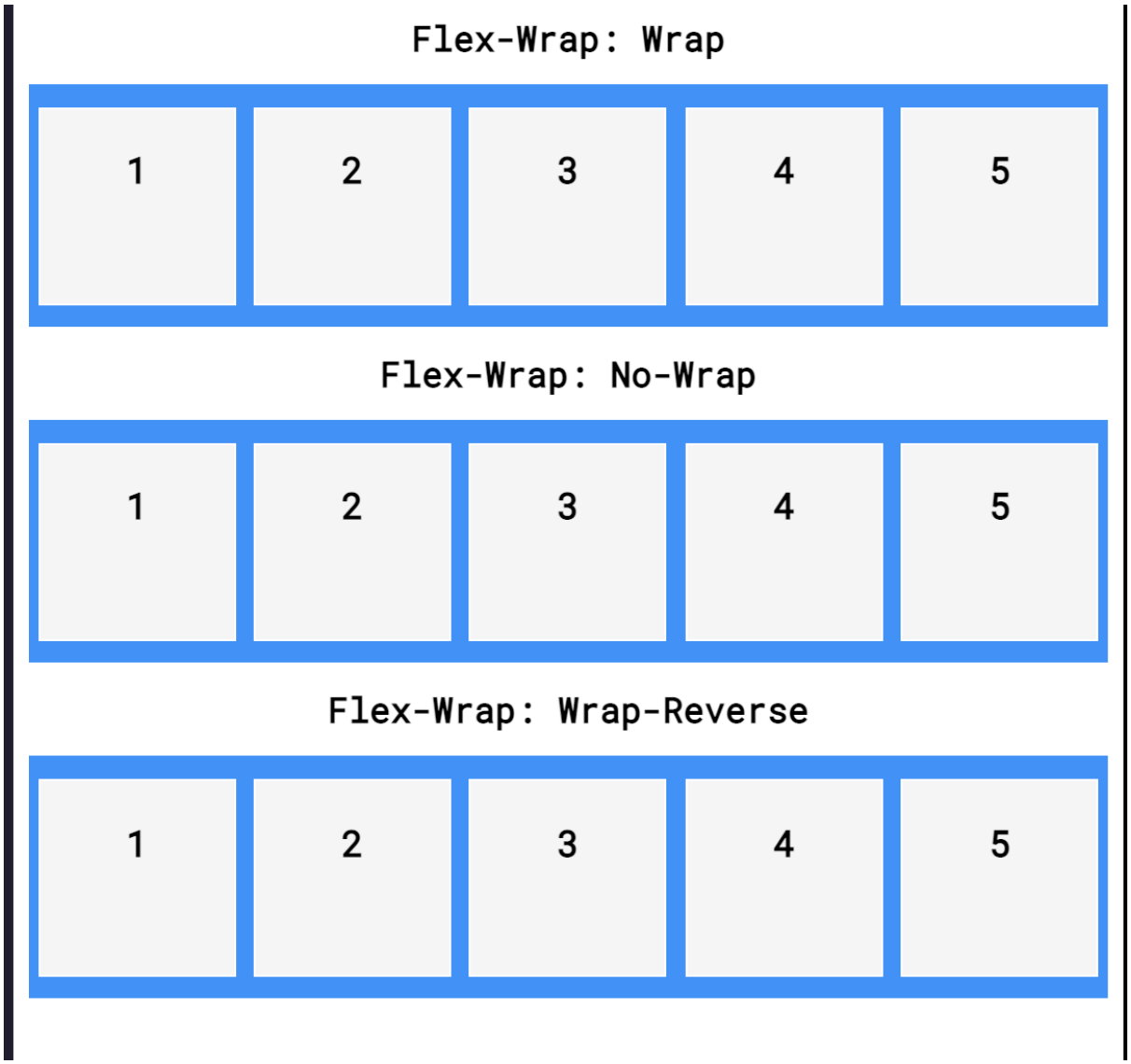
- **flex-wrap (flex container 에 적용) (shrink 와 관련)**

때로, 컨텐츠가 줄어들지 않길 원하면서 줄어들 경우 새로운 라인으로 이동하는 것을 원할 때가 있다. 이럴 경우 flex-wrap 프로퍼티를 사용하면 된다.

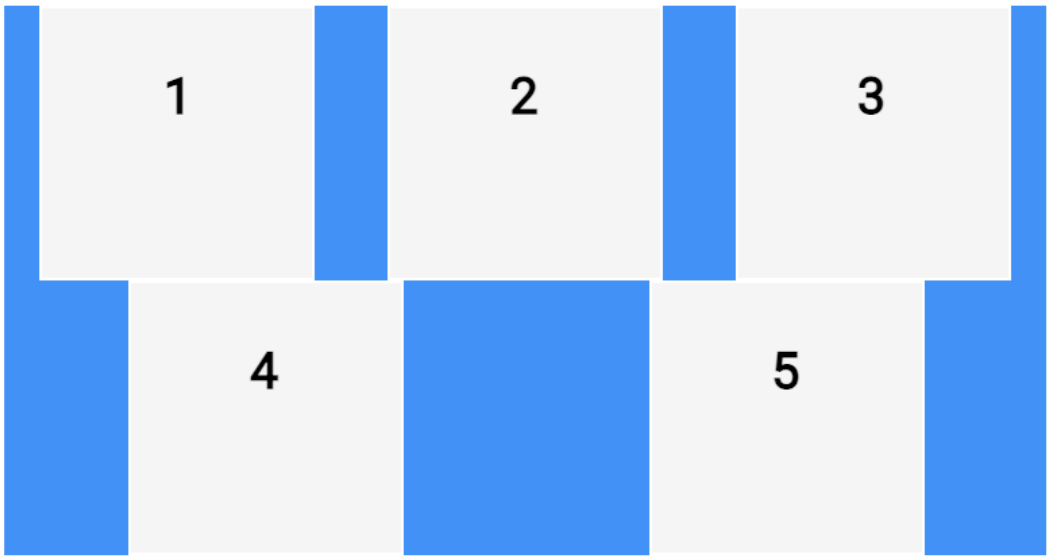
flex-wrap 은 3가지의 밸류를 갖고 있다.

1. wrap → row 넓이에 맞지 않는 flex 컨테이너 안의 아이템들을 다음 라인으로 이동시킨다.
2. wrap-reverse → wrap 과 같은 기능이지만 그 순서가 반대로 되는 것이다.
3. nowrap → 디폴트 값이며, 다른 css 룰에 의해 지정된 하나의 wrap 값을 override 하기 위해 필요하다.

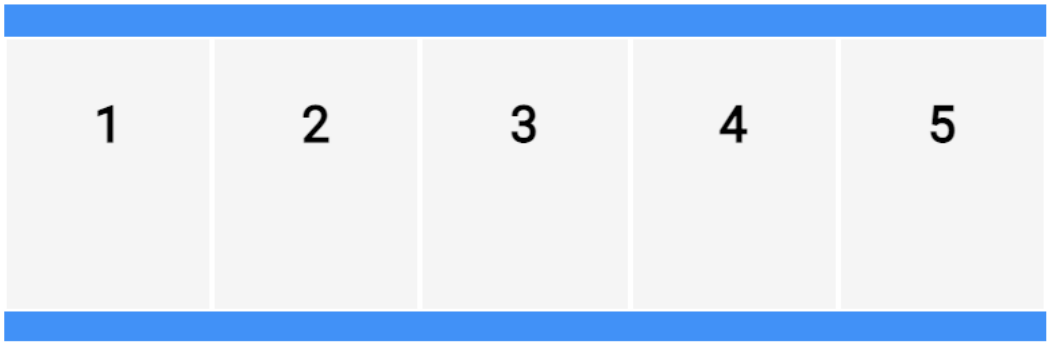
```css
.container {
    background-color: dodgerblue;
    display: flex;
    align-items: center;
    min-height: 125px;
    justify-content: space-around;
}

.box {
    background-color: whitesmoke;
    border: 1px solid white;
    width: 100px;
    height: 100px;
}

#wrap {
    flex-wrap: wrap;
}

#nowrap {
    flex-wrap: nowrap;
}

#reverse {
    flex-wrap: wrap-reverse;
}
```


Flex-Wrap: Wrap
1
2
3
4
5
Flex-Wrap: No-Wrap
1
2
3
4
5
Flex-Wrap: Wrap-Reverse
1
2
3
4
5

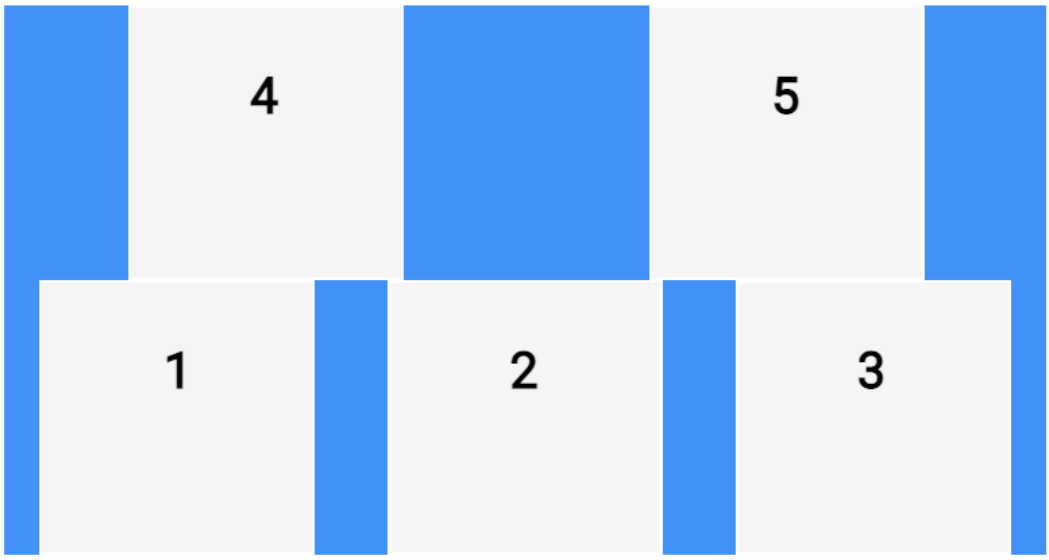
Flex-Wrap: Wrap

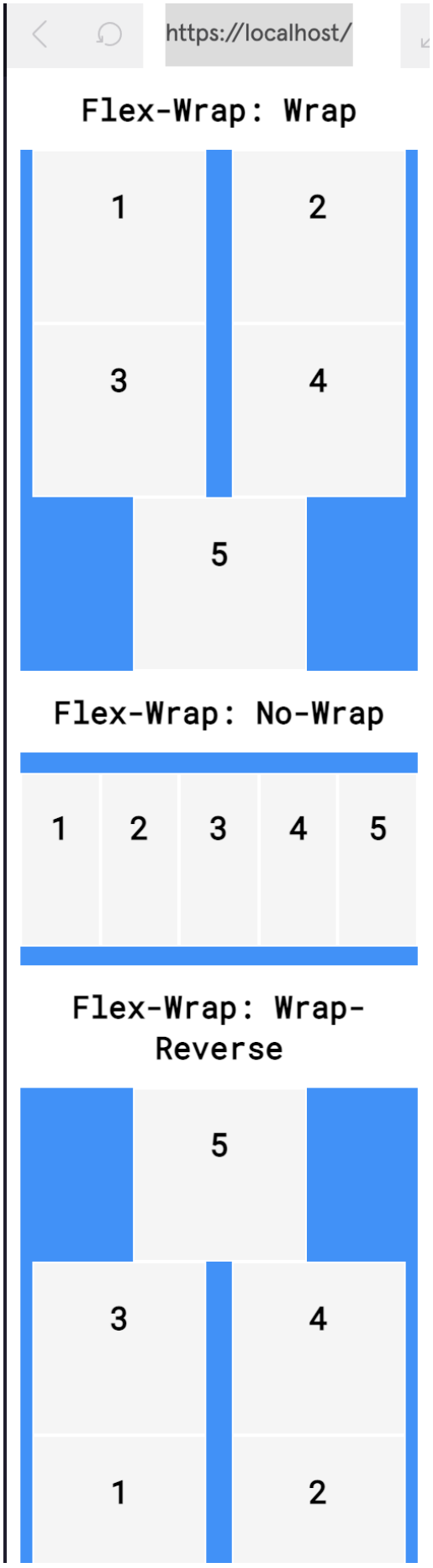


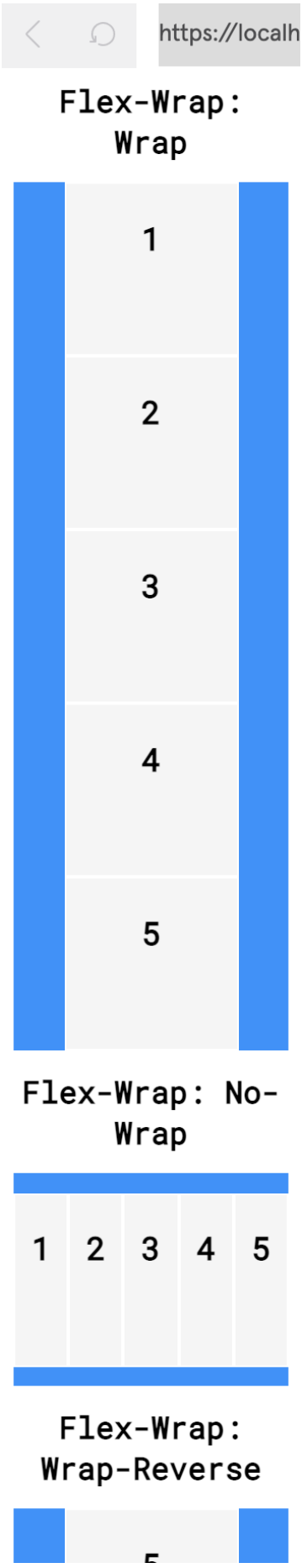
Flex-Wrap: No-Wrap



Flex-Wrap: Wrap-Reverse




https://localhost/
Flex-Wrap: Wrap
1
2
3
4
5
Flex-Wrap: No-Wrap
1
2
3
4
5
Flex-Wrap: Wrap-Reverse
5
3
4
1
2


https://localh
Flex-Wrap: Wrap
1
2
3
4
5
Flex-Wrap: No-Wrap
1
2
3
4
5
Flex-Wrap: Wrap-Reverse


Flex-Wrap:
Wrap-
Reverse
5
4
3
2
1

- **align-content (flex-container 에 적용)**

align-items 는 컨테이너 안의 하나의 row 를 위에서부터 아래로 움직이는 것이라면 align-content 는 다수의 row를 위에서부터 아래로 움직이는 것이다.

이또한 flex-start, flex-end, center, space-around, space-between, stretch 를 밸류로 갖는다.

```html
<div class='container'>
  <div class='child'>
    <h1>1</h1>
  </div>
  <div class='child'>
    <h1>2</h1>
  </div>
  <div class='child'>
    <h1>3</h1>
  </div>
  <div class='child'>
    <h1>4</h1>
  </div>
</div>
```

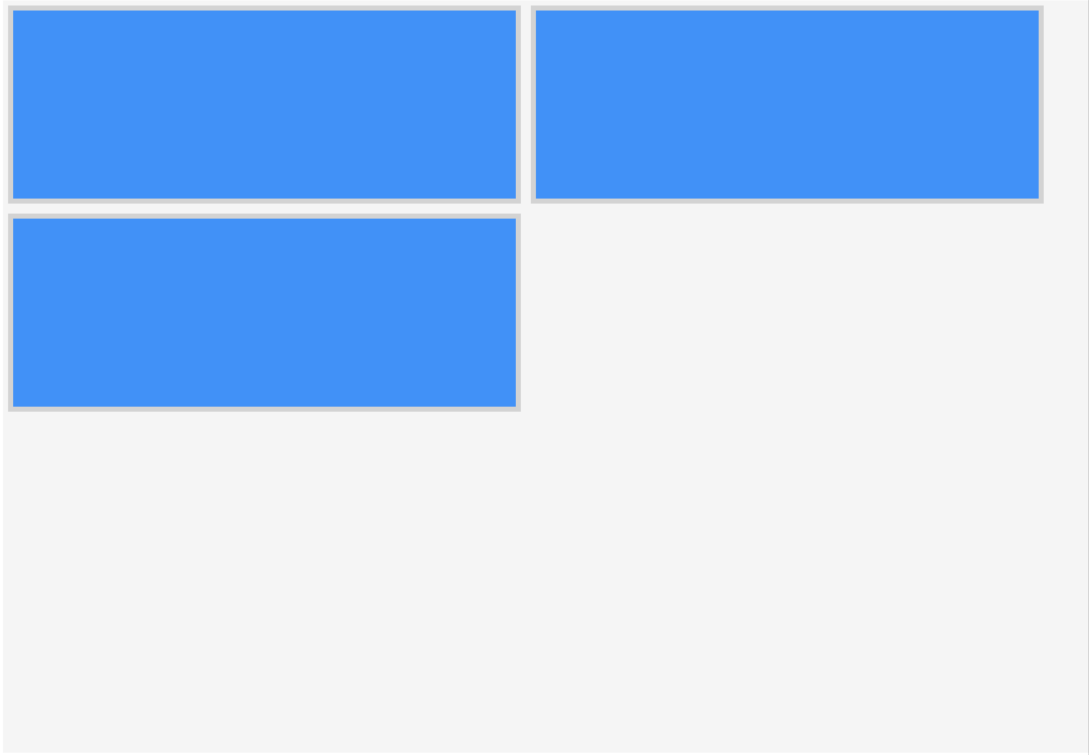
```css
.container {
  display: flex;
  width: 400px;
  height: 400px;
  flex-wrap: wrap;
  align-content: space-around;
}

.child {
  width: 150px;
  height: 150px;
}
```

실습

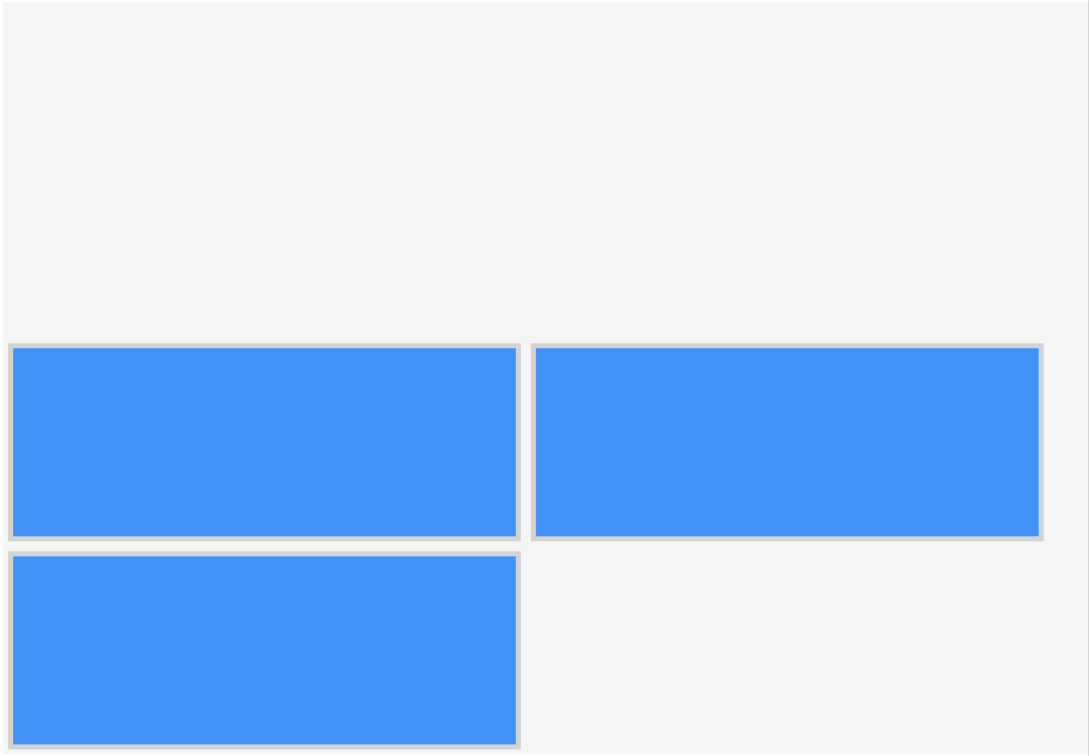
```
.container {
    height: 300px;
    width: 600px;
    background-color: whitesmoke;
    display: flex;
    flex-wrap: wrap;
    margin: auto;
}

.left,
.center,
.right {
    min-height: 75px;
    width: 200px;
    margin: 2px;
    background-color: dodgerblue;
    border: 2px solid lightgrey;
}
```

```
28    }
29
30  ▼ #flexstart {
31      align-content: flex-start;
32    }
33
34  ▼ #flexend {
35      align-content: flex-end;
36    }
37
38  ▼ #center {
39      align-content: center;
40    }
41
42  ▼ #between {
43      align-content: space-between;
44    }
45
46  ▼ #around {
47      align-content: space-around;
48    }
49
```

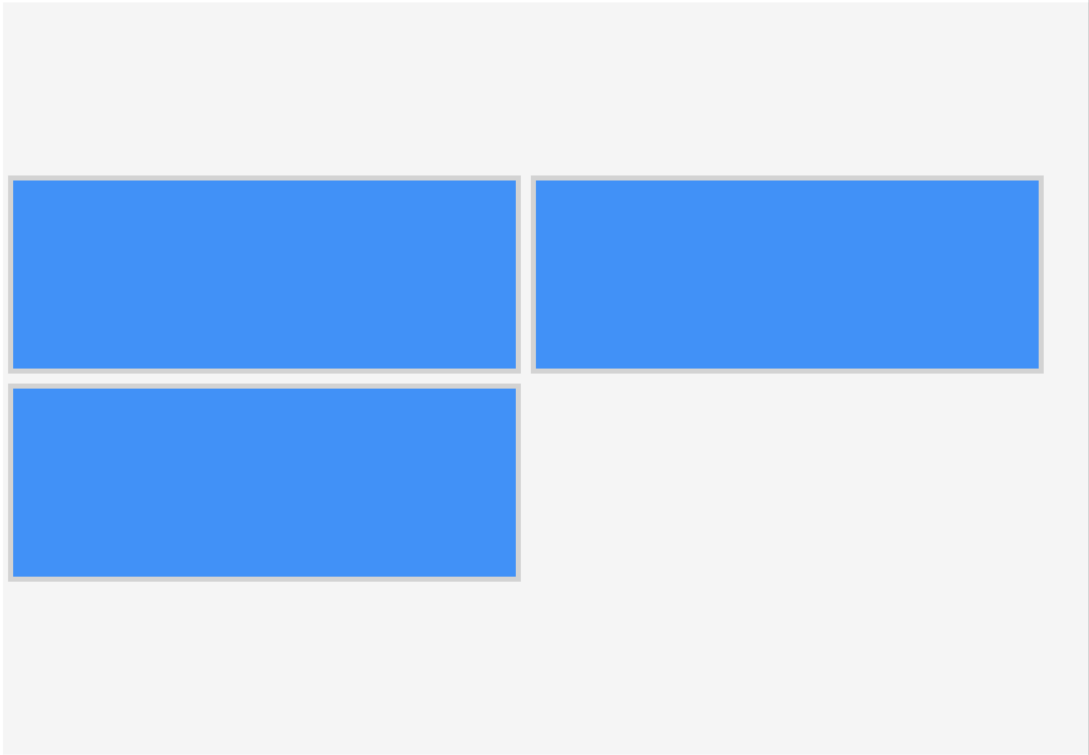
Flex Start



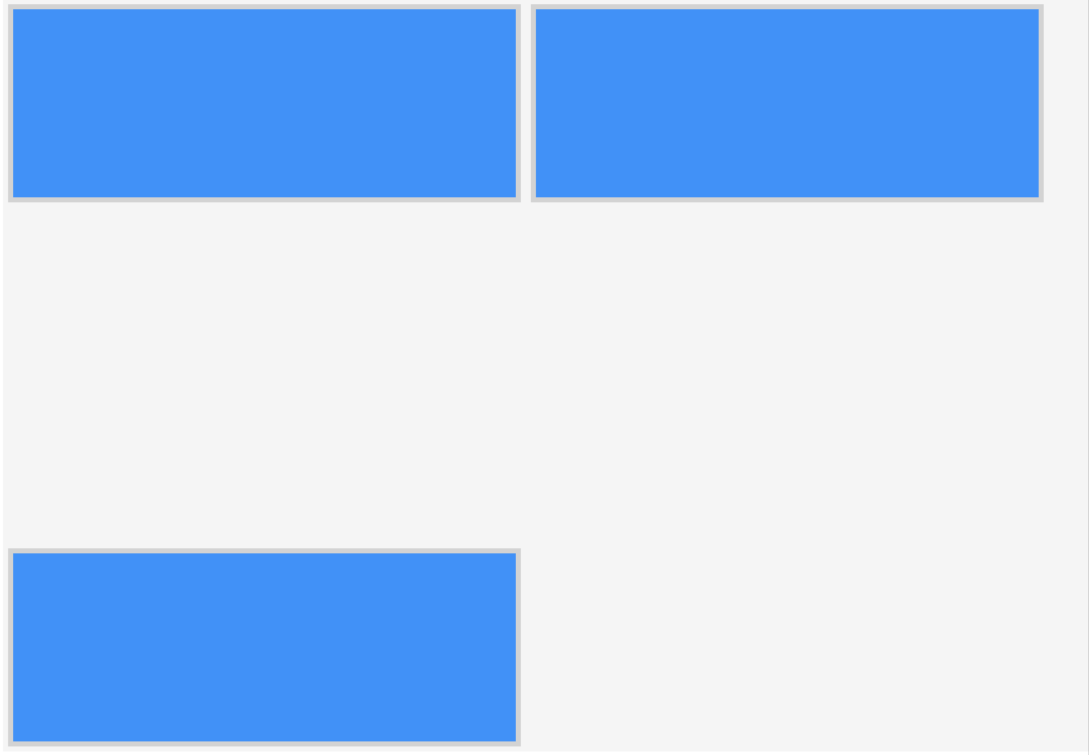
Flex End



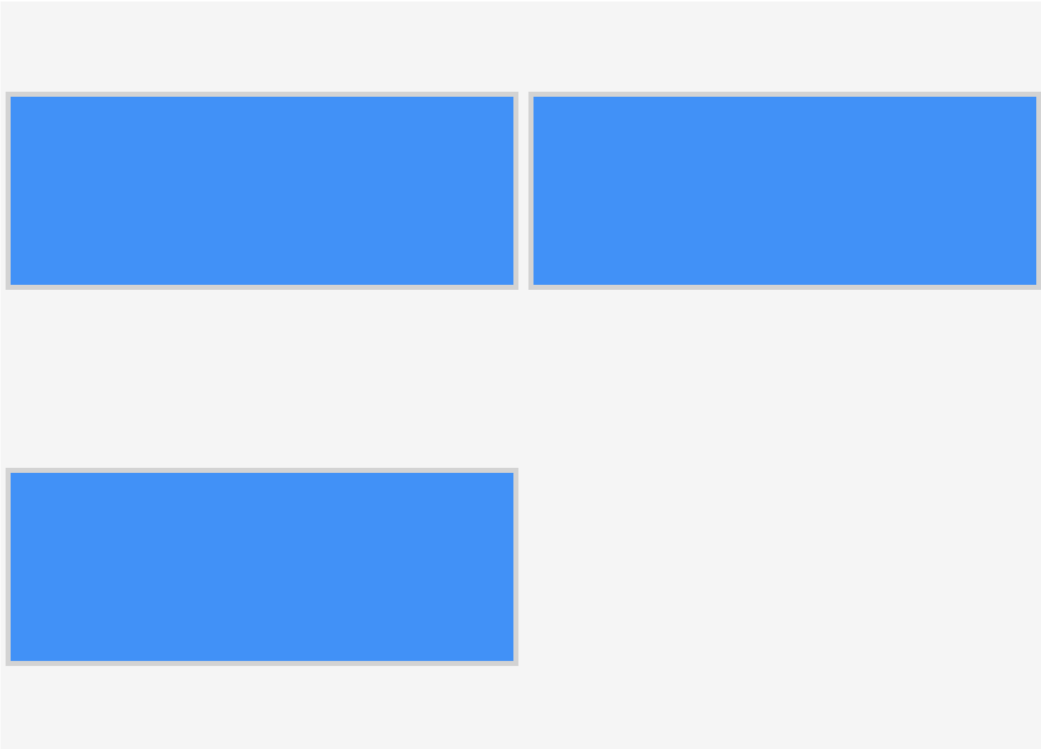
Center



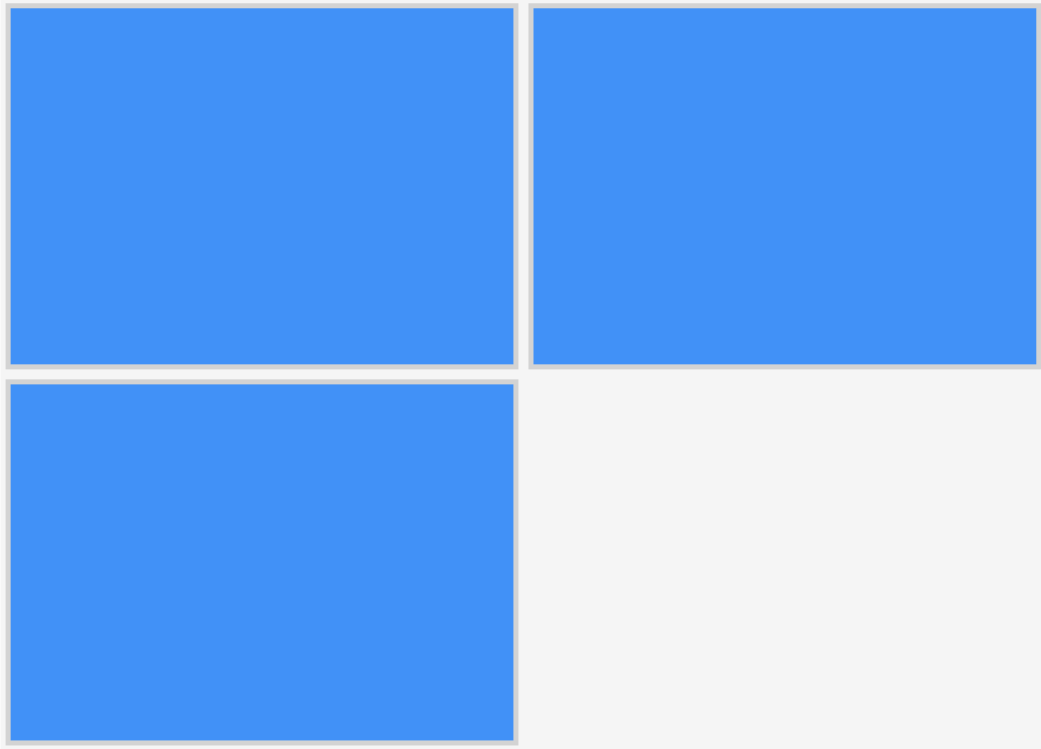
Space Between



Space Around



Stretch



stretch 는 아이템 css rule set 에 min-height 값을 지정해야만 늘어나게 된다.

- **flex-direction (flex-container 에 적용)**

지금까지 flex-item 에 대해서만 다뤘다. 수평적으로 늘리고 줄이고, 수직적으로 wrap 하는 것들을. flex container 은 두 개의 축을 가지고 있다.(main axis, cross axis) 메인 축은 수평축이고, 크로스축은 수직축이다.

메인 축은 아래 프로퍼티와 함께 flex item 을 위치시키기 위해 사용된다.

1. justify-content
2. flex-wrap
3. flex-grow
4. flex-shrink

크로스 축은 아래와 같다.

1. align-items
2. align-content

우리는 flex-direction 프로퍼티를 사용하여 메인축과 크로스축을 서로 변경할 수 있다.
flex-direction : column 을 적용하게 되면, flex item 들은 모두 수직으로 나열하게 된다.

그렇게 되면 메인축에서 작동하는 justify-content 는 작동하지 않게 된다. !!

```html
<div class='container'>
  <div class='item'>
    <h1>1</h1>
  </div>
  <div class='item'>
    <h1>2</h1>
  </div>
  <div class='item'>
    <h1>3</h1>
  </div>
  <div class='item'>
    <h1>4</h1>
  </div>
  <div class="item">
    <h1>5</h1>
  </div>
</div>
```

```css
.container {
  display: flex;
  flex-direction: column;
  width: 1000px;
}
.item {
  height: 100px;
  width: 100px;
}
```

flex-direction 은 다음 4가지 벨류를 포함한다.

1. row - 디폴트값 , 왼쪽에서 오른쪽으로 왼쪽 위 꼭대기에서 시작한다.
2. row-reverse - row 의 반대 순서
3. column - 아이템들을 수직으로 나열한다.
4. column-reverse - 반대 순서

```css
.container {
  background-color: dodgerblue;
  display: flex;
  max-height: 600px;
  align-items: center;
  justify-content: space-around;
}

.box {
  background-color: whitesmoke;
  border: 1px solid white;
  width: 100px;
  height: 100px;
  flex-grow: 1;
}

#row {
  flex-direction: row;
}

#row-reverse {
  flex-direction:row-reverse;
}

#column {
  flex-direction: column;
}

#column-reverse {
  flex-direction: column-reverse;
}

```

max-height 을 지정하면 아이템들이 컨테이너에 딱 맞추게 된다.

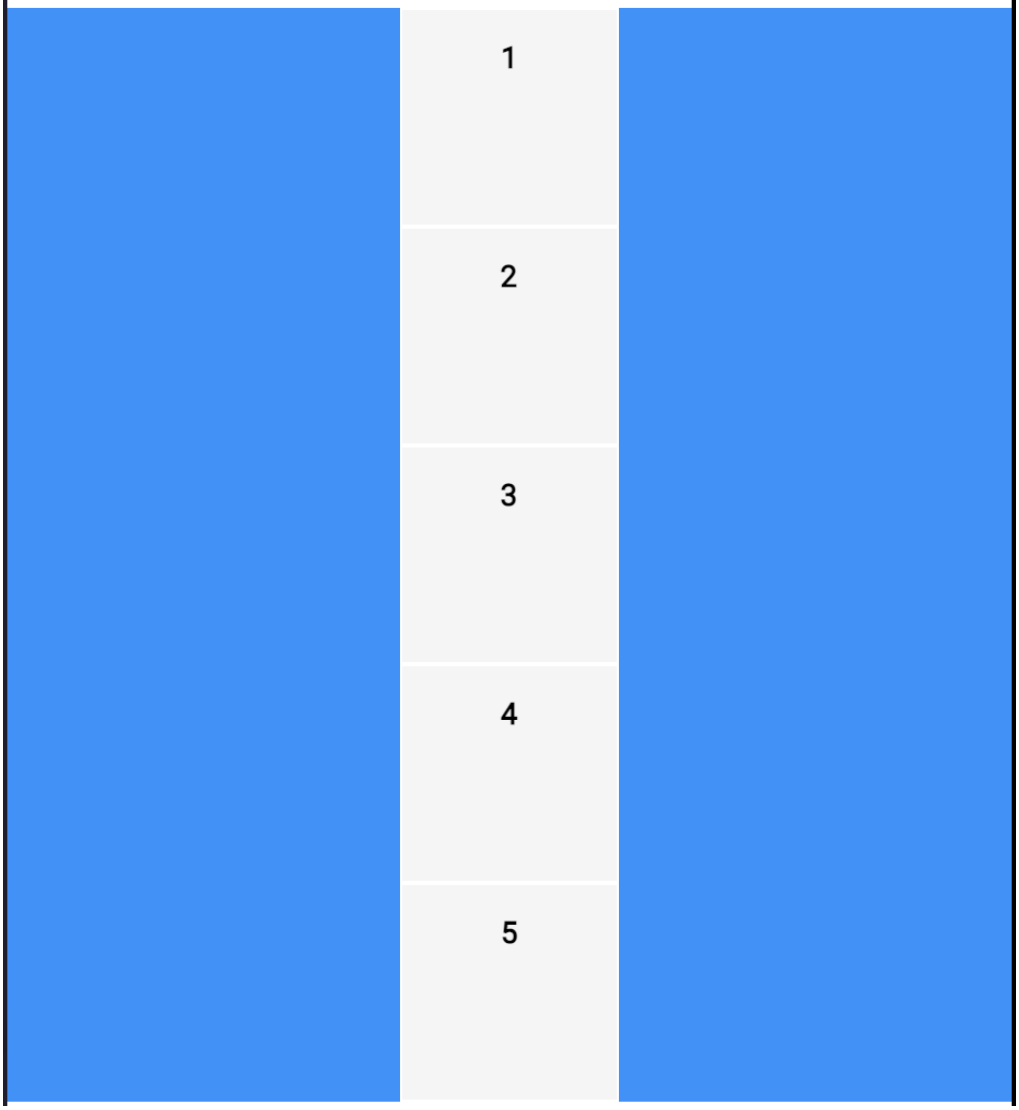
Flex-Direction: Row

1 2 3 4 5

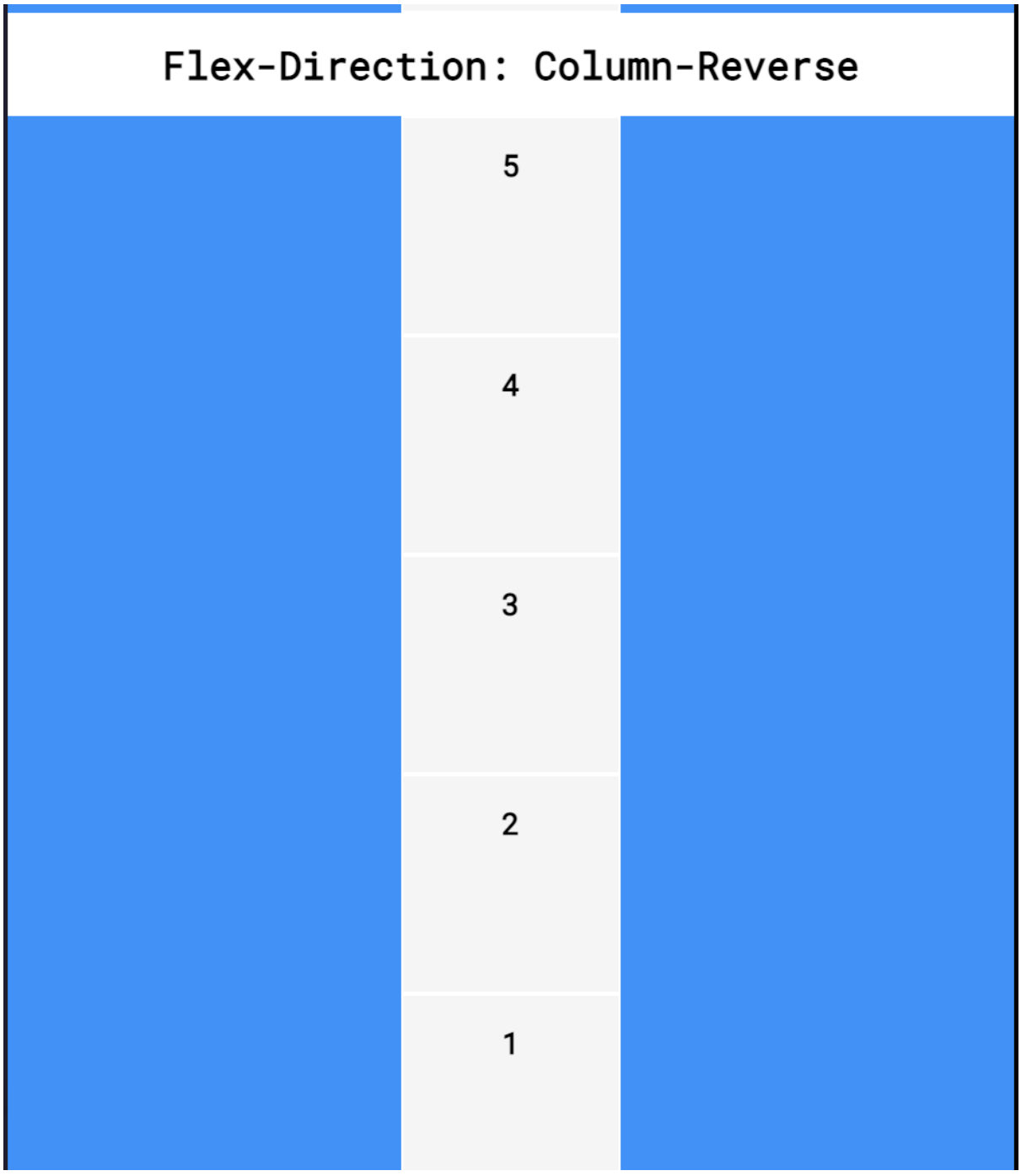
Flex-Direction: Row-Reverse

5 4 3 2 1

Flex-Direction: Column



container 에 지정한 align-items : center 은 row 형식에는 적용이 안되는 것을 볼 수 있다. 또한 column 에서는 justify-contents : space-around 와 flex-grow : 1 적용이 안되는 것을 볼 수 있다.



- **flex-flow (flex container 에 적용)**

flex 숄핸드 프로퍼티와 같이 flex-flow 숄핸드 프로퍼티가 존재한다. 이는 flex-wrap 과 flex-direction 을 한 줄에 쓰게 할 수 있다.

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

In the example above, we take two lines to accomplish what can be done with one.

```
.container {
  display: flex;
  flex-flow: column wrap;
}
```

- **Nested Flexboxes**

flex 박스 안에 flex 박스가 들어갈 수 있다.

오늘의 단어

- daunting : 벅찬, 주눅이 들게 하는
- put together : 조립하다. 모으다 , 만들다.
- fraction : 부분, 일부; 분수
- eradicate: 근절하다, 뿌리를 뽑다.
- philosophy: 철학
- axis : 축
- precedence : 선행