

17강

# CSS Flex(Flexible Box) & Google Web Font





## Flex

## ✓ CSS Flex(Flexible Box)

- 대부분 사이트는 전체 레이아웃이 수직 구성이며 '위-아래'로 스크롤 하여 사용한다.  
레이아웃을 구성할 때 가장 많이 사용하는 요소(Elements)들이 기본적으로 **블록(Block)** 개념으로 **표시(Display)**되며 이는 **뷰(View)**에 수직(위에서 아래로)으로 쌓이기 때문에 수직 구성은 상대적으로 쉽게 만들 수 있다.  
하지만 수평(왼쪽에서 오른쪽으로) 구성의 경우는 상황이 조금 다르다.
- 문제는 수평 구조를 만드는 속성이 명확하지 않았기 때문인데, 그래서 많은 경우 **<table>**나 **float** 혹은 **inline-block** 등의 도움을 받아 사용하였다.  
하지만 이러한 방법들은 다양한 문제(**Clear**, **White space** 등, 해결은 가능하지만..)를 가지고 있기 때문에 결국 수평 레이아웃 구성의 차선책일 뿐이며, 이제 우리는 **Flex(Flexible Box)**라는 명확한 개념(속성들)으로 레이아웃을 쉽게 구성할 수 있다.

## ✓ 간단한 Flex(Flexible Box) 예제

## HTML

```
<body>
  <div class="box-container">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
  </div>
</body>
```

## CSS

```
.box-container {
  display: flex;
}
```



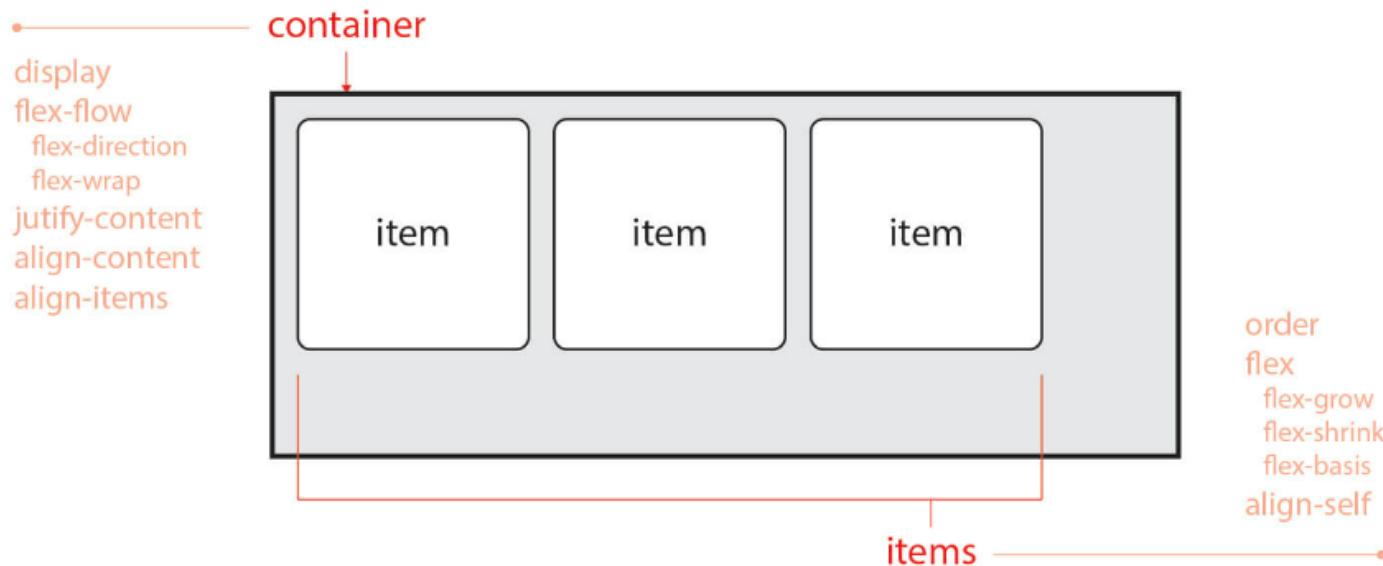
box1box2box3



## Flex

## ✓ CSS Flex(Flexible Box)

- Flex는 요소의 크기가 불분명하거나 동적인 경우에도, 각 요소를 정렬할 수 있는 효율적인 방법을 제공한다.
- 우선 Flex는 2개의 개념으로 나뉜다.  
첫 번째는 Container, 두 번째는 Items 이다.  
위에서 살펴본 바와 같이 **Container**는 **Items**를 감싸는 부모 요소이며, 각 **Item**을 정렬하기 위해선 **Container**가 필수이다.
- 주의할 부분은 **Container**와 **Items**에 적용하는 속성이 구분되어 있다는 것이다.  
**Container**에는 **display, flex-flow, justify-content** 등의 속성을 사용할 수 있으며,  
**Items**에는 **order, flex, align-self** 등의 속성을 사용할 수 있다.





## Flex

### ✓ Flex Container

#### 속성

#### 의미

display      Flex Container를 정의

flex-flow      flex-direction와 flex-wrap의 단축 속성

flex-direction      Flex items의 주 축(main-axis)을 설정

flex-wrap      Flex items의 여러 줄 뮤음(줄 바꿈) 설정

justify-content      주 축(main-axis)의 정렬 방법을 설정

align-content      교차 축(cross-axis)의 정렬 방법을 설정(2줄 이상)

align-items      교차 축(cross-axis)에서 items의 정렬 방법을 설정(1줄)



## Flex

## ✓ display

- display 속성으로 Flex Container를 정의한다.

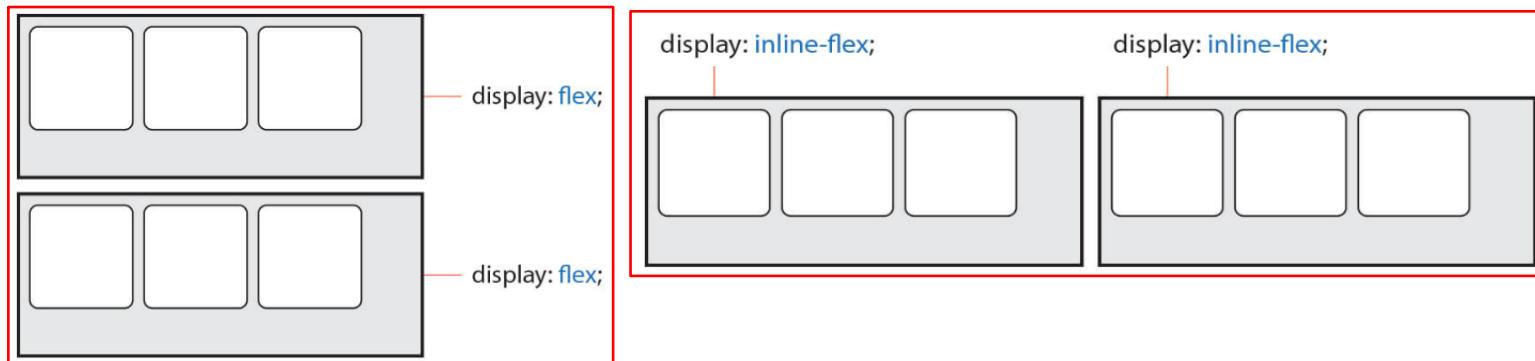
보통 요소의 표시 방법을 **display: block;**, **display: inline-block** 혹은 **display: none;** 같이 사용하는 경우가 많은데, 같은 요소의 표시 방법으로 Block이나 Inline이 아닌 Flex(**display: flex**, **display: inline-flex**)로 정의한다.

값	의미
flex	Block 특성의 flex Container를 정의
inline-flex	Inline 특성의 flex Container를 정의

- 여기서 말하는 수직과 수평 쌓임은 **Items**이 아니라 **Container**라는 것에 주의하자.
- 두 값의 차이는 내부에 **Items**에는 영향을 주지 않는다.

- flex와 inline-flex의 차이는 단순하다.

**display: flex;**로 지정된 Flex Container는 Block 요소와 같은 성향(수직 쌓임)을 가지며,  
**display: inline-flex;**로 지정된 Flex Container는 Inline(Inline Block) 요소와 같은 성향(수평 쌓임)을 가진다.





## Flex

### ✓ flex-flow

- 이 속성은 단축 속성으로 Flex Items의 주 축(main-axis)을 설정하고 Items의 여러 줄 묶음(줄 바꿈)도 설정한다.

값	의미	기본값
flex-direction	Items의 주 축(main-axis)을 설정	row
flex-wrap	Items의 여러 줄 묶음(줄 바꿈)설정	nowrap

#### • flex-direction

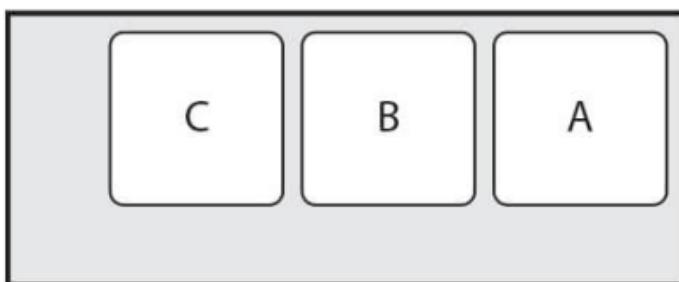
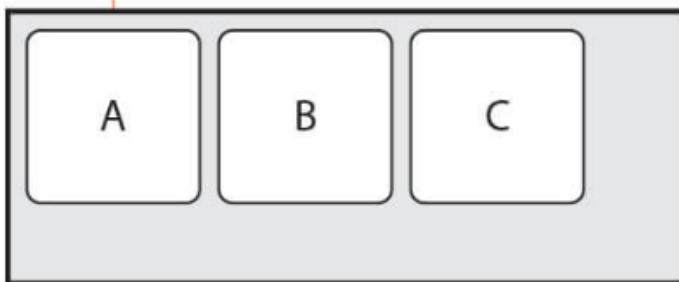
값	의미	기본값
row	Items를 수평축(왼쪽에서 오른쪽으로)으로 표시	row
row-reverse	Items를 row의 반대 축으로 표시	
column	Items를 수직축(위에서 아래로)으로 표시	
column-reverse	Items를 column의 반대 축으로 표시	



## Flex

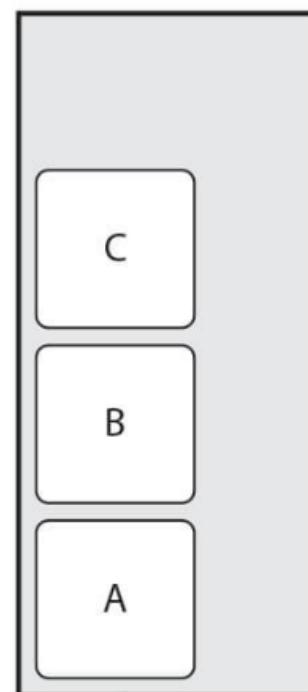
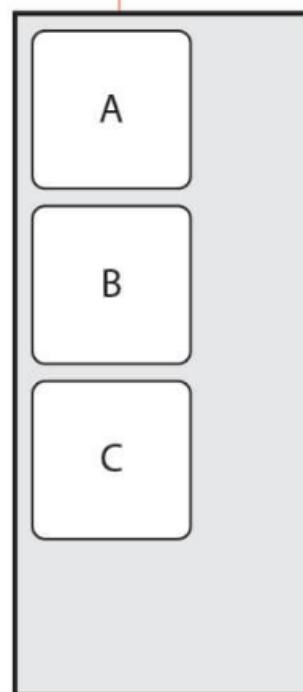
### ✓ flex-flow

flex-direction: **row**; (default)



flex-direction: **row-reverse**;

flex-direction: **column**;



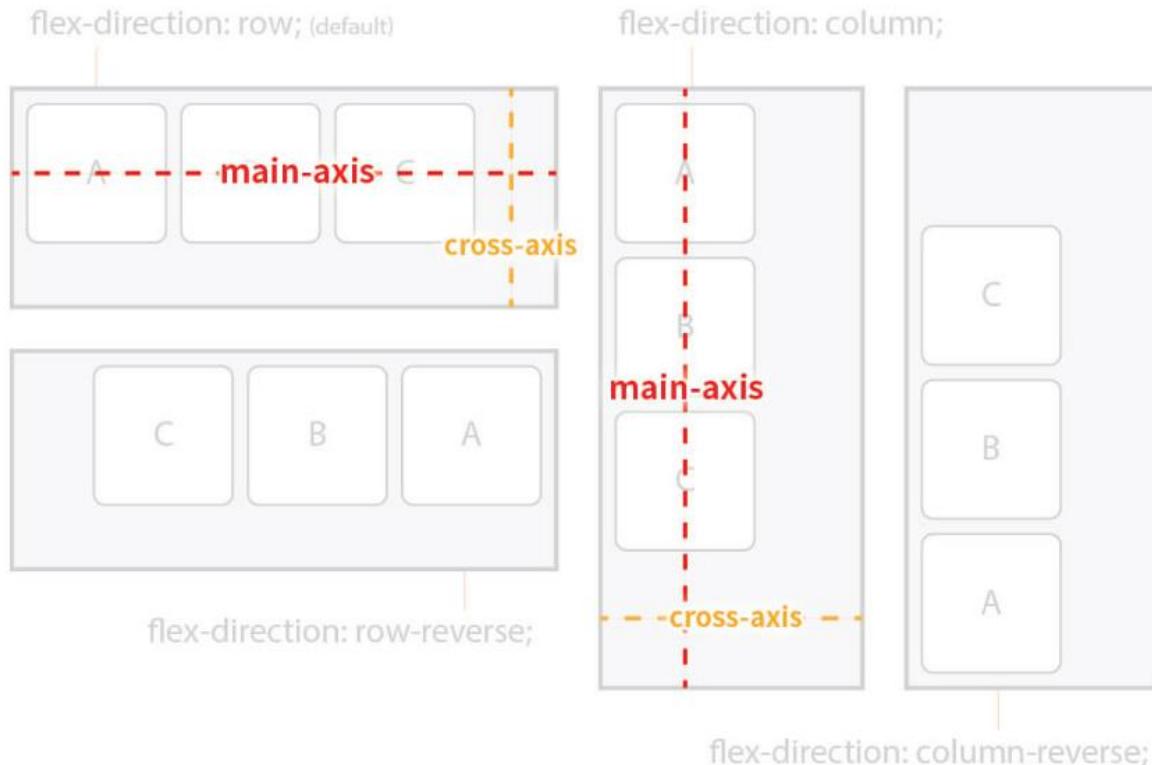
flex-direction: **column-reverse**;



## Flex

## ✓ 주 축(main-axis)과 교차 축(cross-axis)

- 값 **row**은 Items를 수평축으로 표시하므로 이때는 주 축이 수평이며 교차 축은 수직이 된다.
- 반대로 값 **column**은 Items를 수직축으로 표시하므로 주 축은 수직이며 교차 축은 수평이 된다.  
즉, 방향(수평, 수직)에 따라 주 축과 교차 축이 달라진다.

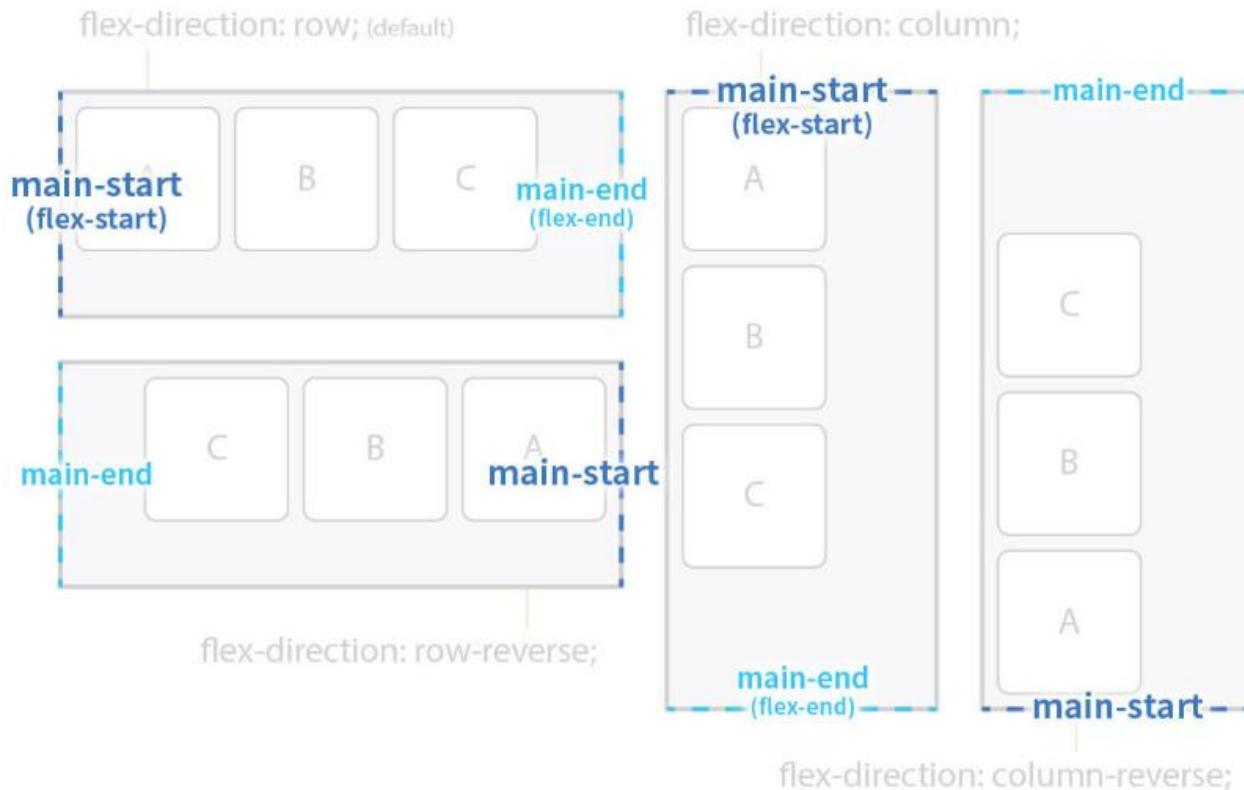




## Flex

## ✓ 시작점(flex-start)과 끝점(flex-end)

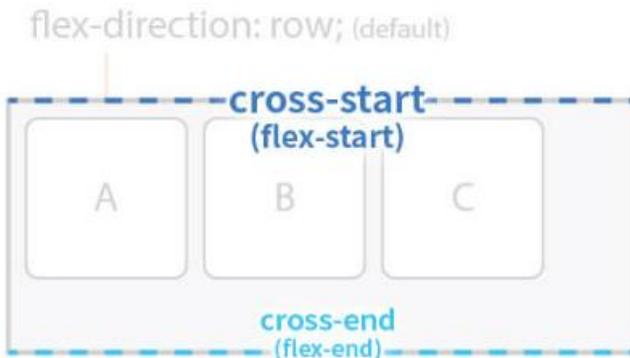
- 시작점(flex-start)과 끝점(flex-end)이라는 개념도 있다. 이는 주 축이나 교차 축의 시작하는 지점과 끝나는 지점을 지칭 한다. 역시 방향에 따라 시작점과 끝점이 달라진다.



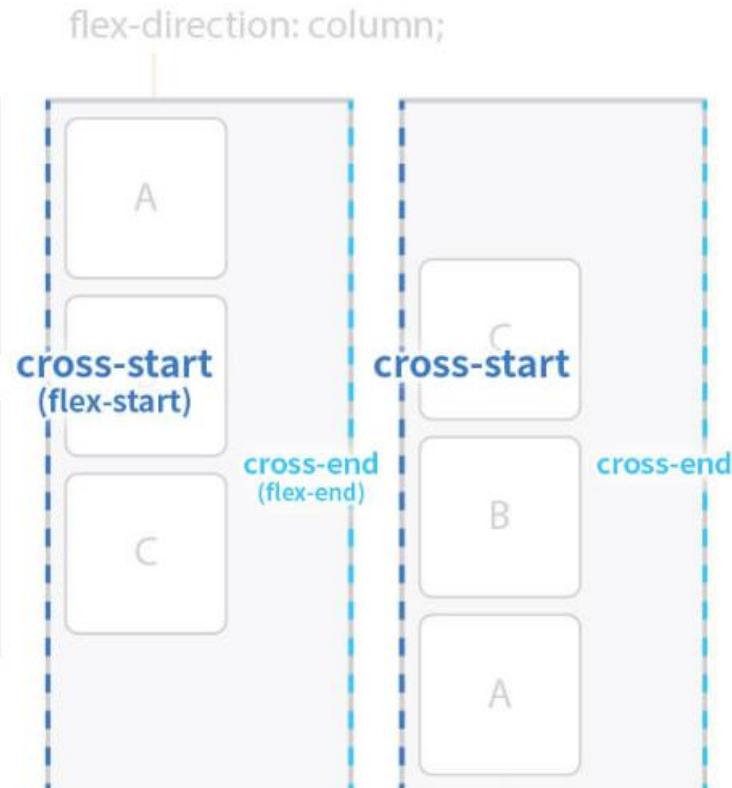


## Flex

## ✓ 시작점(flex-start)과 끝점(flex-end)



flex-direction: row-reverse;



flex-direction: column-reverse;



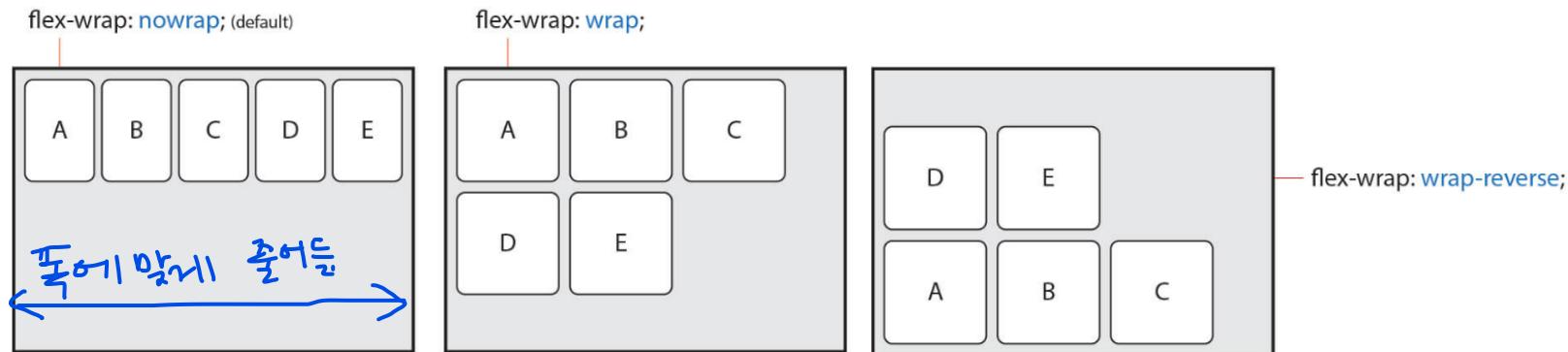
## Flex

## ✓ flex-wrap

- Items의 여러 줄 묶음(줄 바꿈)을 설정한다.

값	의미	기본값
nowrap	모든 Items를 여러 줄로 묶지 않음(한 줄에 표시)	nowrap
wrap	Items를 여러 줄로 묶음	
wrap-reverse	Items를 wrap의 역 방향으로 여러 줄로 묶음	

- 기본적으로 Items는 한 줄에서만 표시되고 줄 바꿈 되지 않는다.  
이는 지정된 크기(주 축에 따라 **width**나 **height**)를 무시하고 한 줄 안에서만 가변 한다.  
Items를 줄 바꿈 하려면 값으로 **wrap**을 사용해야 한다.





## Flex

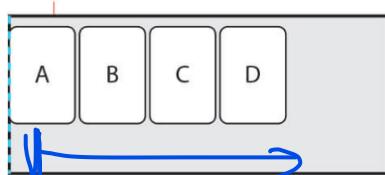
17

## ✓ Justify-content

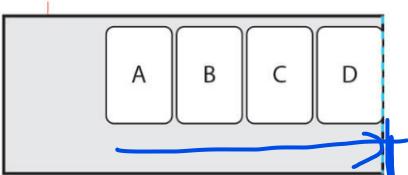
- 주 축(main-axis)의 정렬 방법을 설정합니다.

값	의미	기본값
flex-start	Items를 시작점(flex-start)으로 정렬	flex-start
flex-end	Items를 끝점(flex-end)으로 정렬	
center	Items를 가운데 정렬	
space-between	시작 Items은 시작점에, 마지막 Item은 끝점에 정렬되고, 나머지 Items는 사이에 고르게 정렬 됨	
space-around	Items를 균등한 여백을 포함하여 정렬	

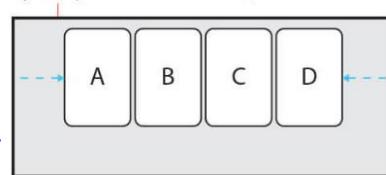
justify-content: flex-start; (default)



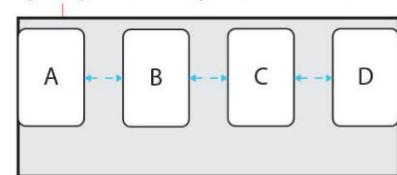
justify-content: flex-end;



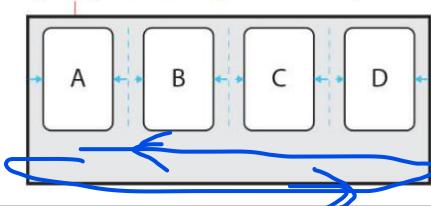
justify-content: center;



justify-content: space-between;



justify-content: space-around;





## Flex

### ✓ align-content

- 교차 축(cross-axis)의 정렬 방법을 설정한다.  
주의할 점은 **flex-wrap** 속성을 통해 Items가 여러 줄(2줄 이상)이고 여백이 있을 경우만 사용할 수 있다.
- Items이 한 줄일 경우 **align-items** 속성을 사용한다.

값	의미	기본값
stretch	Container의 교차 축을 채우기 위해 Items를 늘림	stretch
flex-start	Items를 시작점(flex-start)으로 정렬	
flex-end	Items를 끝점(flex-end)으로 정렬	
center	Items를 가운데 정렬	
space-between	시작 Items은 시작점에 마지막 Item은 끝점에 정렬되고, 나머지 Items는 사이에 고르게 정렬 됨	
space-around	Items를 균등한 여백을 포함하여 정렬	

- 값 **stretch**는 교차 축에 해당하는 너비(속성 **width** 혹은 **height**)가 값이 **auto**(기본값)일 경우 교차 축을 채우기 위해 자동으로 늘어난다.

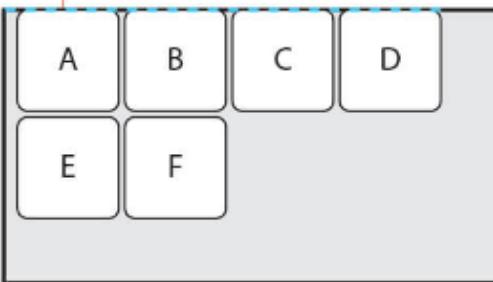


Flex  
✓ align-content

align-content: stretch; (default)



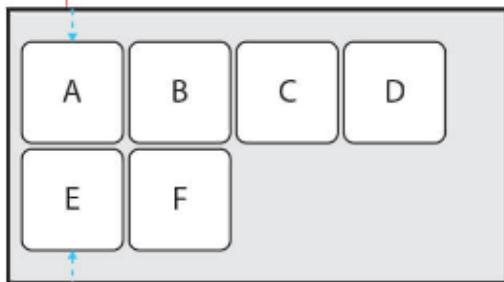
align-content: flex-start;



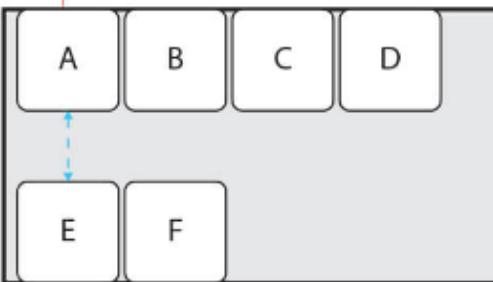
align-content: flex-end;



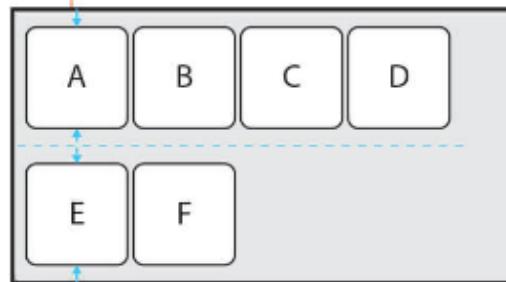
align-content: center;



align-content: space-between;



align-content: space-around;





## Flex

## ✓ align-items

- 교차 축(cross-axis)에서 Items의 정렬 방법을 설정한다.  
Items이 한 줄일 경우 많이 사용한다.
- 주의할 점은 Items이 **flex-wrap**을 통해 여러 줄(2줄 이상)일 경우에는 **align-content** 속성이 우선한다.  
따라서 **align-items**를 사용하려면 **align-content** 속성을 기본값(stretch)으로 설정해야 한다.

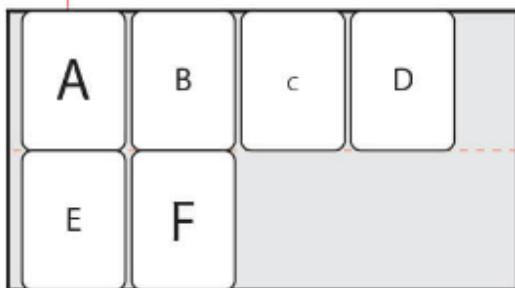
값	의미	기본값
stretch	Container의 교차 축을 채우기 위해 Items를 늘림	stretch
flex-start	Items를 각 줄의 시작점(flex-start)으로 정렬	
flex-end	Items를 각 줄의 끝점(flex-end)으로 정렬	
center	Items를 가운데 정렬	
baseline	Items를 문자 기준선에 정렬	



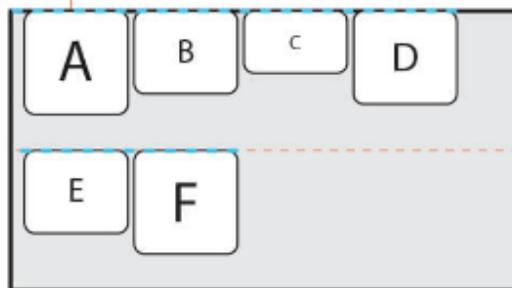
## Flex

## ✓ align-items

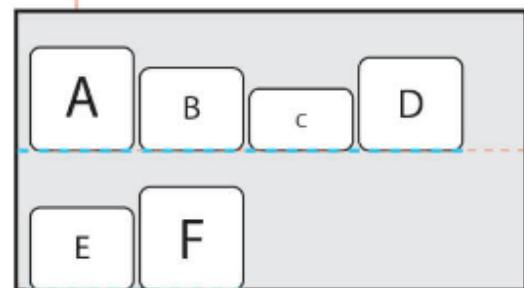
align-items: stretch; (default)



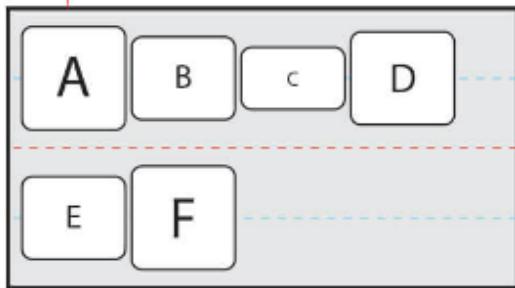
align-items: flex-start;



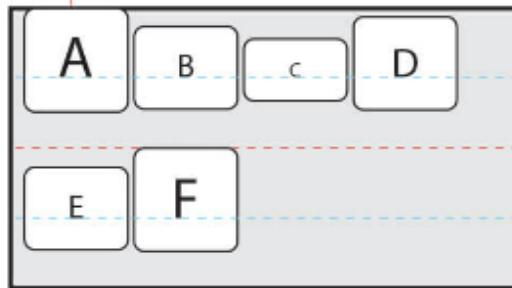
align-items: flex-end;



align-items: center;



align-items: baseline;





## Flex

### ✓ flex Items

값	의미
order	Flex Item의 순서를 설정
flex	Flex-grow, flex-shrink, flex-basis의 단축 속성
flex-grow	Flex Item의 증가 너비 비율 설정
flex-shrink	Flex Item의 감소 너비 비율 설정
flex-basis	Flex Item의 (공간 배분 전) 기본 너비 설정
align-self	교차 축(cross-axis)에서 Item의 정렬 방법 설정

### ✓ Order

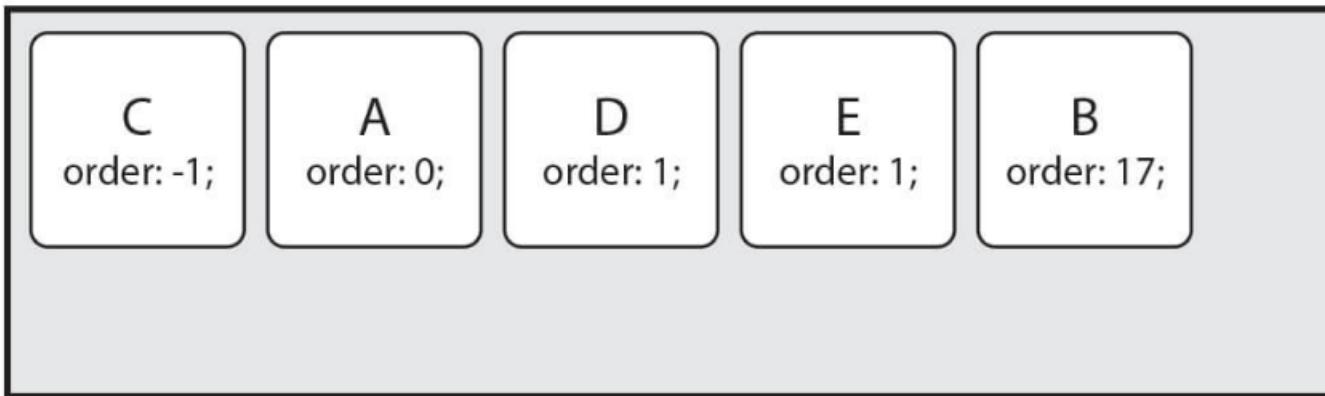
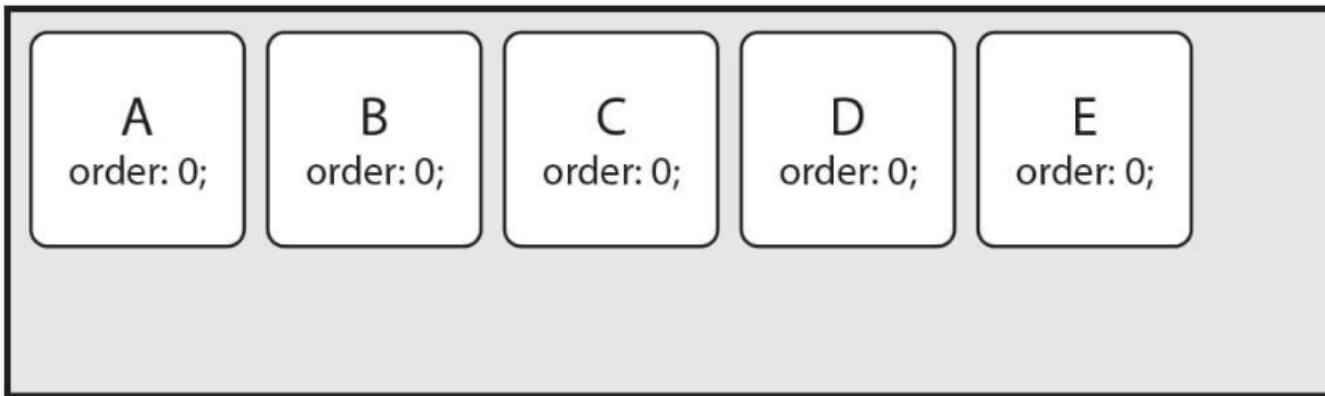
- Item의 순서를 설정한다.
- Item에 숫자를 지정하고 숫자가 클수록 순서가 밀린다.
- 음수가 허용된다.
- HTML 구조와 상관없이 순서를 변경할 수 있기 때문에 유용하다.

값	의미	기본값
숫자	Item의 순서 설정	0



## Flex

## ✓ Order





## Flex

### ✓ flex

- Item의 너비(증가, 감소, 기본)를 설정하는 단축 속성이다.

값	의미	기본값
flex-grow	Item의 증가 너비 비율을 설정	0
flex-shrink	Items의 감소 너비 비율을 설정	1
flex-basis	Items의 (공간 배분 전) 기본 너비 설정	auto

```
.item {  
    flex: 1 1 20px; /* 증가너비 감소너비 기본너비 */  
  
    flex: 1 1; /* 증가너비 감소너비 flex-basis는 생략하면 0이 된다.*/  
  
    flex: 1 20px; /* 증가너비 기본너비 (단위를 사용하면 flex-basis가 적용됩니다) */  
}
```



## Flex

## ✓ align-self

- 교차 축(cross-axis)에서 개별 Item의 정렬 방법을 설정한다..
- align-items**는 Container 내 모든 Items의 정렬 방법을 설정한다. 필요에 의해 일부 Item만 정렬 방법을 변경하려고 할 경우 **align-self**를 사용할 수 있다. 이 속성은 **align-items** 속성보다 우선한다.

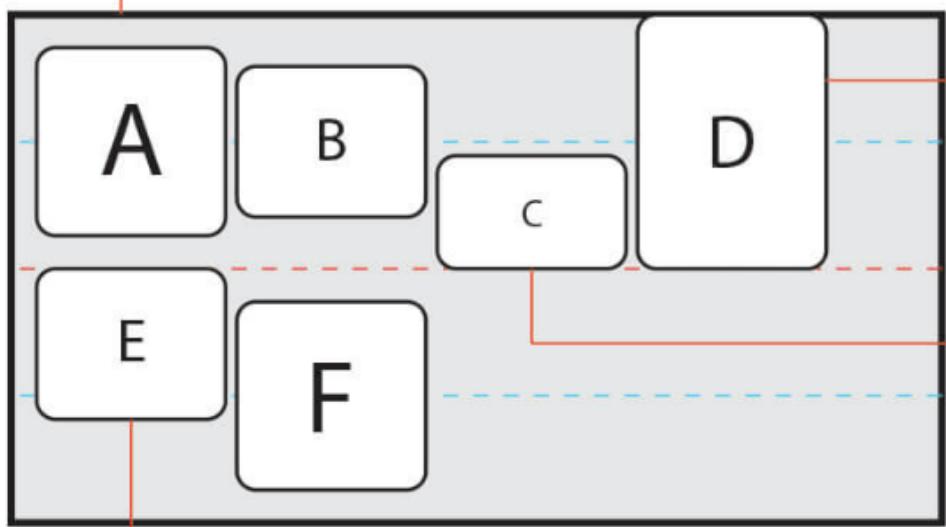
값	의미	기본값
auto	Container의 align-items 속성을 상속받음	auto
stretch	Container의 교차 축을 채우기 위해 Item을 늘림	
flex-start	Item을 각 줄의 시작점(flex-start)으로 정렬	
flex-end	Item을 각 줄의 끝점(flex-end)으로 정렬	
center	Item을 가운데 정렬	
baseline	Item을 문자 기준선에 정렬	



## Flex

## ✓ align-self

align-items: center;



align-self: stretch;  
height: auto;

align-self: flex-end;

align-self: flex-start;



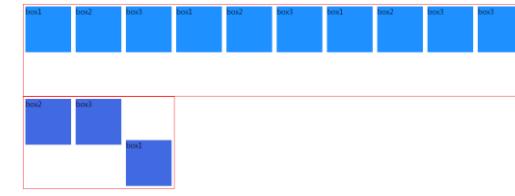
## Flex

## HTML

```
<body>
  <div class="box-container">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box3</div>
  </div>
  <div class="box-container">
    <div class="box3">box1</div>
    <div class="box2">box2</div>
    <div class="box2">box3</div>
  </div>
```

## CSS

```
.box-container {
  border: 1px solid red;
  height: 200px;
  display: inline-flex;
  flex-flow: row wrap;
  justify-content: center;
}
.box-container .box {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: dodgerblue;
}
.box-container .box2, .box3 {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: royalblue;
}
.box-container .box3{
  align-self: self-end;
  order: 17;
}
```





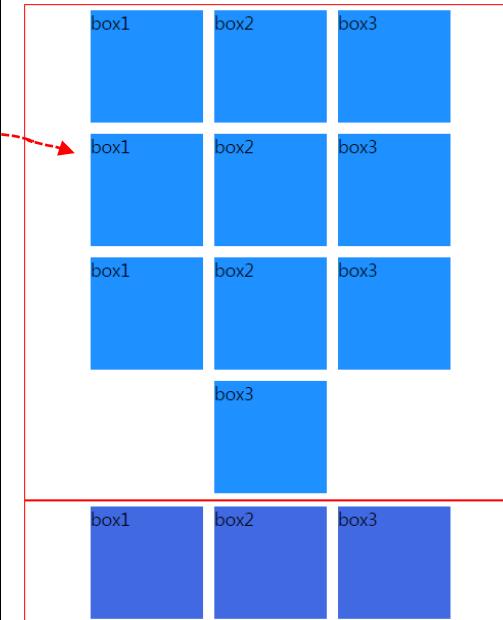
## Flex

## HTML

```
<body>
  <div class="box-container">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box3</div>
  </div>
  <div class="box-container">
    <div class="box3">box1</div>
    <div class="box2">box2</div>
    <div class="box2">box3</div>
  </div>
</body>
```

## CSS

```
.box-container {
  border:1px solid red;
  display:flex;
  flex-flow: row wrap;
  justify-content: center;
  /* align-items: center; */
}
.box-container .box {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: dodgerblue;
}
.box-container .box2, .box3 {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: royalblue;
}
.box-container .box3{
  align-self: self-end;
}
```





## Flex

## HTML

```
<body>
  <div class="box-container">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box3</div>
  </div>
  <div class="box-container">
    <div class="box3">box1</div>
    <div class="box2">box2</div>
    <div class="box2">box3</div>
  </div>
</body>
```

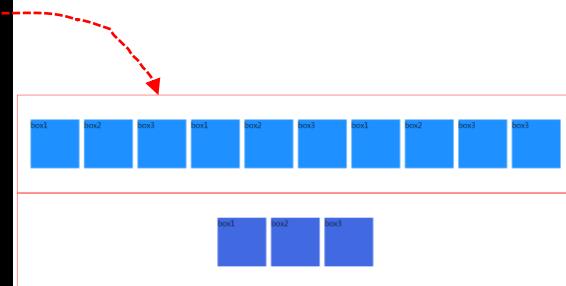
## CSS

```
.box-container {
  border:1px solid red;
  height: 200px;
  display:flex;
  flex-flow: row wrap;
  justify-content: center;
  align-content: center;
  /* align-items: center; */
}

.box-container .box {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: dodgerblue;
}

.box-container .box2, .box3 {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: royalblue;
}

.box-container .box3{
  align-self: self-end;
}
```





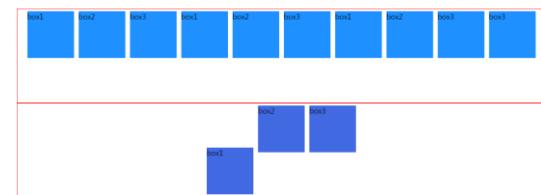
## Flex

## HTML

```
<body>
  <div class="box-container">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
  </div>
  <div class="box-container">
    <div class="box3">box1</div>
    <div class="box2">box2</div>
    <div class="box2">box3</div>
  </div>
</body>
```

## CSS

```
.box-container {
  border:1px solid red;
  height: 200px;
  display:flex;
  flex-flow: row wrap;
  justify-content: center;
}
.box-container .box {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: dodgerblue;
}
.box-container .box2, .box3 {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: royalblue;
}
.box-container .box3{
  align-self: self-end;
}
```





## Flex

## HTML

```
<body>
  <div class="box-container">
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
    <div class="box">box2</div>
    <div class="box">box3</div>
    <div class="box">box1</div>
  </div>
  <div class="box-container">
    <div class="box3">box1</div>
    <div class="box2">box2</div>
    <div class="box2">box3</div>
  </div>
</body>
```

## CSS

```
.box-container {
  border:1px solid red;
  height: 200px;
  display:flex;
  flex-flow: row wrap;
  justify-content: center;
}
.box-container .box {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: dodgerblue;
}
.box-container .box2, .box3 {
  width: 100px;
  height: 100px;
  margin: 5px;
  background-color: royalblue;
}
.box-container .box3{
  align-self: self-end;
  order: 17;
}
```

