15

Flex & Grid

김근형 강사

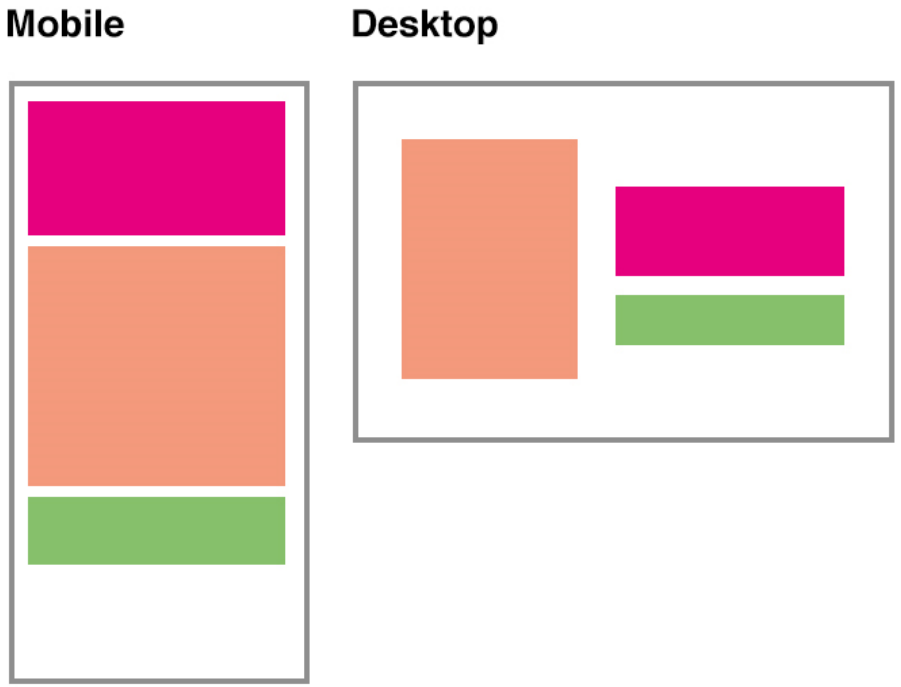
Flex

Flex 란?

레이아웃을 좀 더 편리하게 배치시키기 위한 기능

기존의 수직 형태의 블록 배치 방식보다 좀 더 유연한 배치를 지원하기 위해 고안된 배치 방식

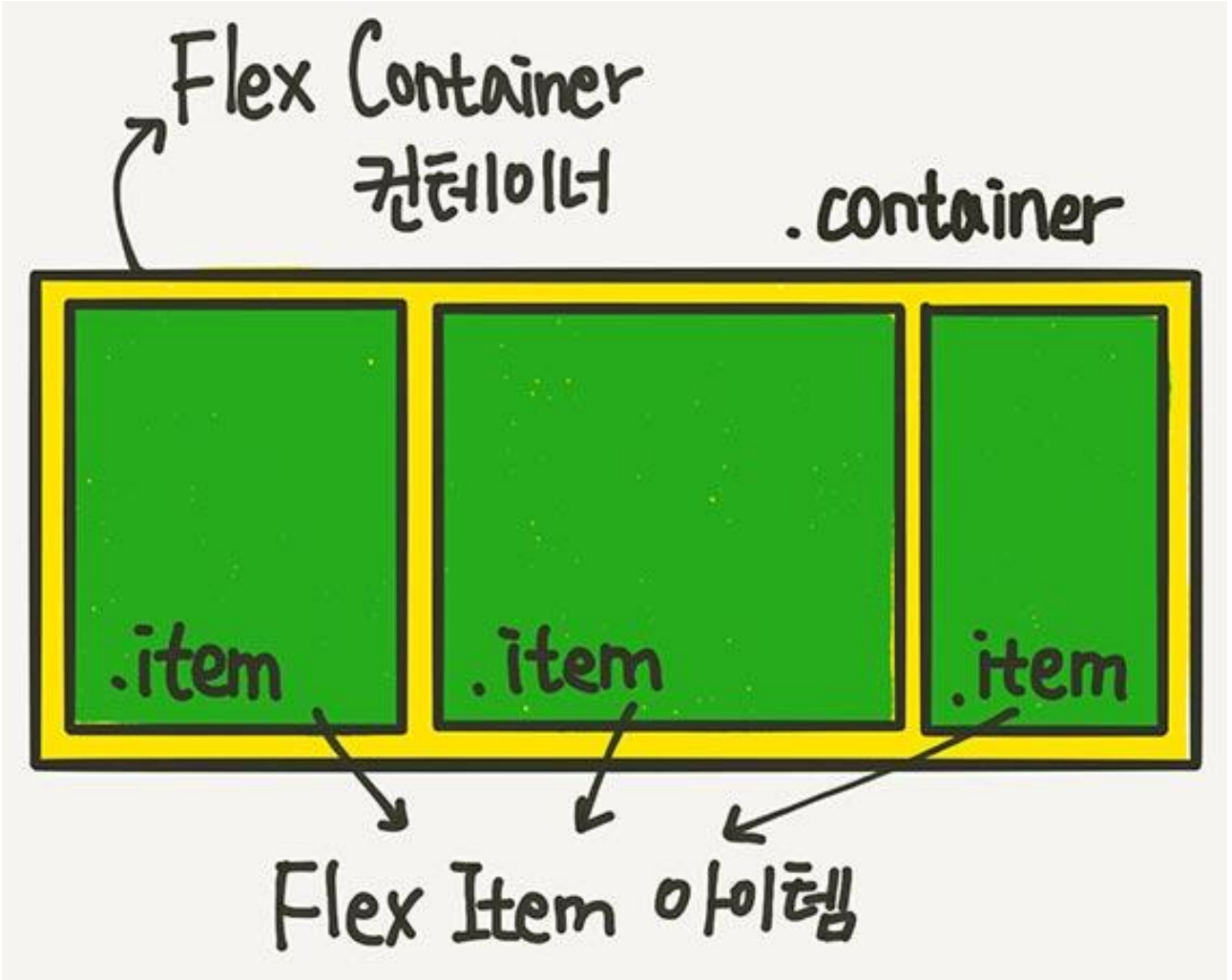
반응형 웹 개발 시 거의 준필수로 사용하게 되는 기능 중 하나



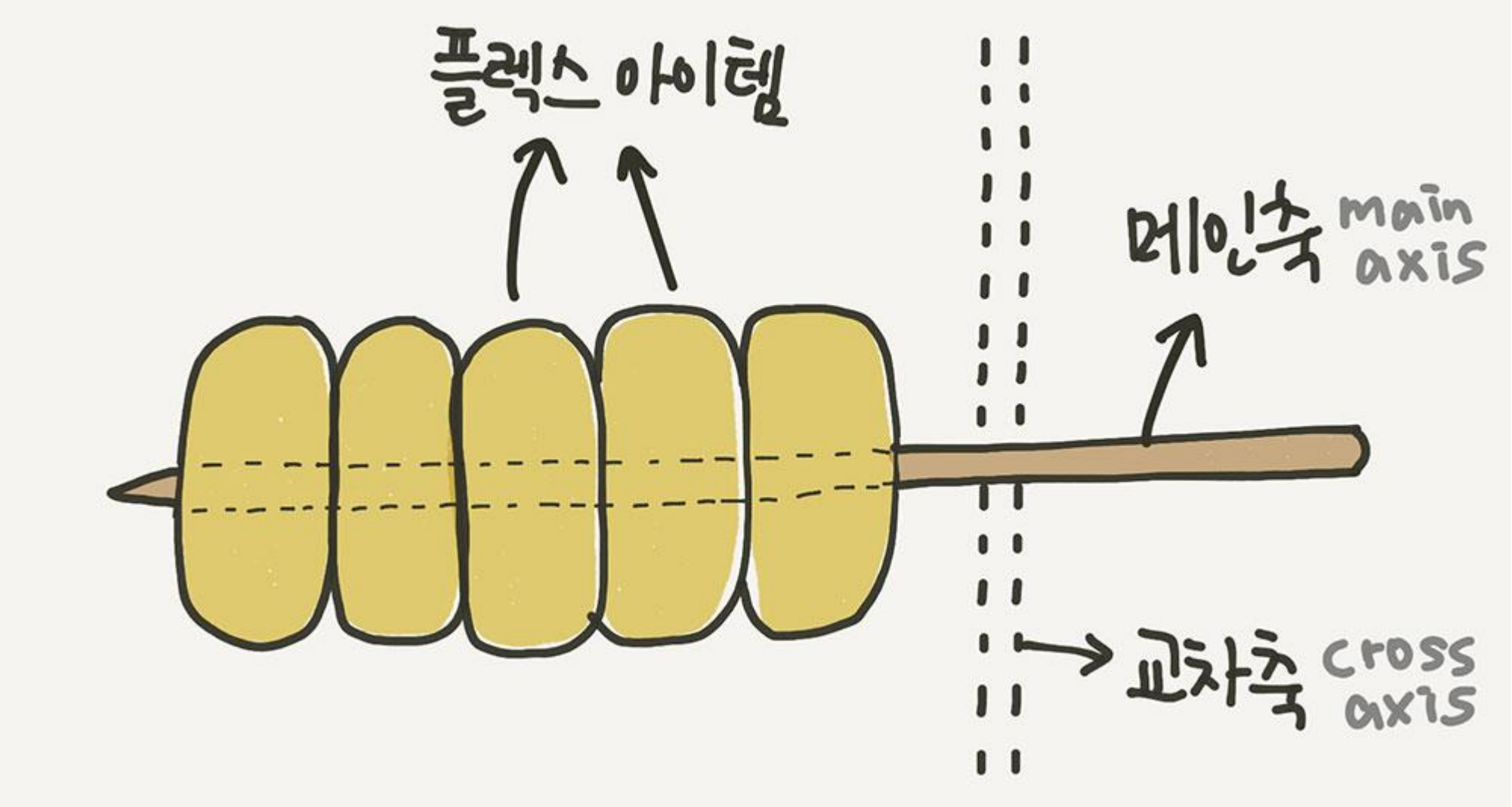
Flex 구성요소

크게 Container와 Item으로 구분된다.

각 위치에 따라 적용되는 css 옵션이 다르다.



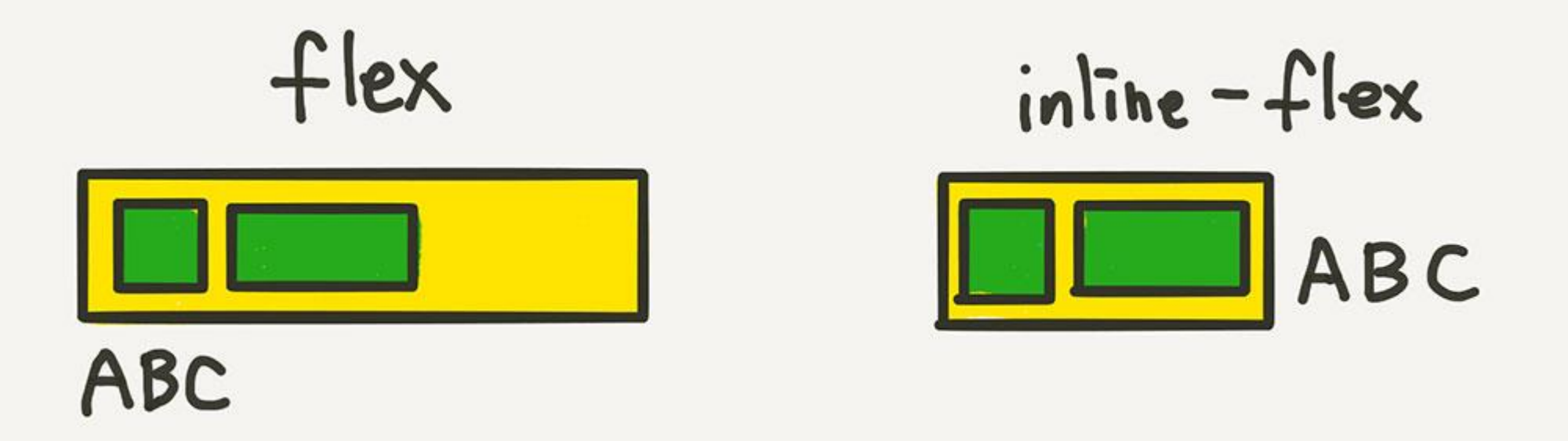
Flex의 기본 원리



Flex Container 옵션

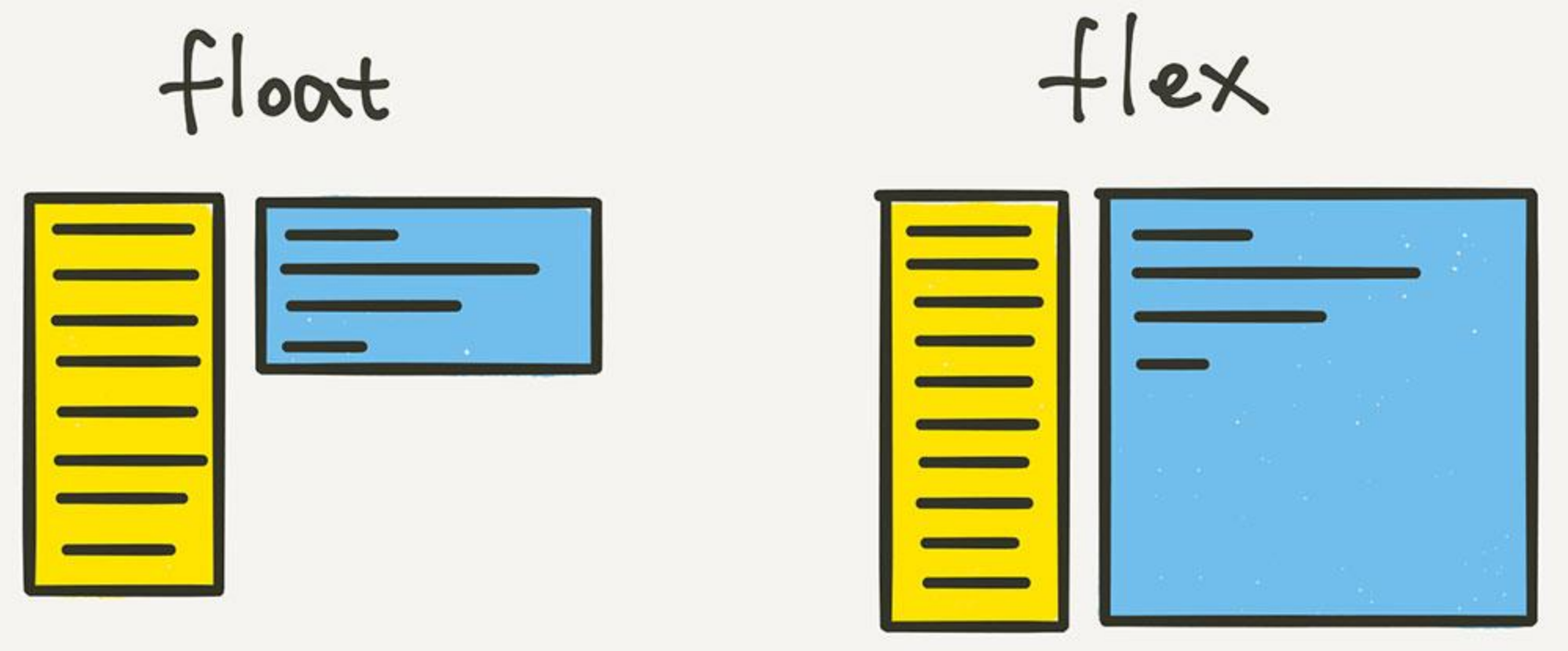
Container에 Flex 속성을 입히기 위한 필수 display 옵션

속성	설명
flex	지정한 크기만큼 x축과 y축으로 이동한다
inline-flex	지정한 크기만큼 x축으로 이동한다



Flex와 Float의 차이점

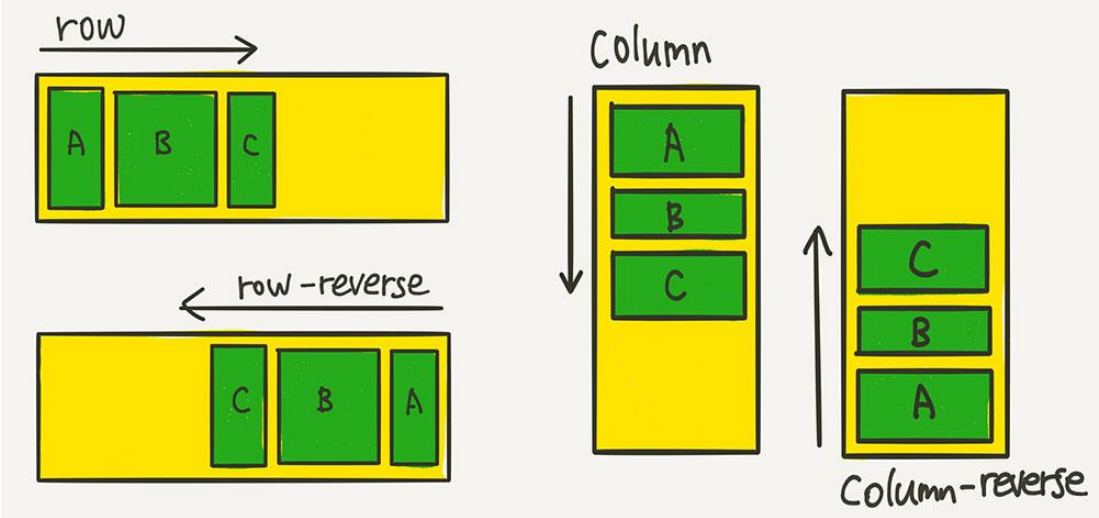
높이가 다른 요소와 맞춰진다는 차이점이 존재



Flex Container 옵션

아이템이 배치되는 축의 방향을 결정하는 옵션 – flex-direction

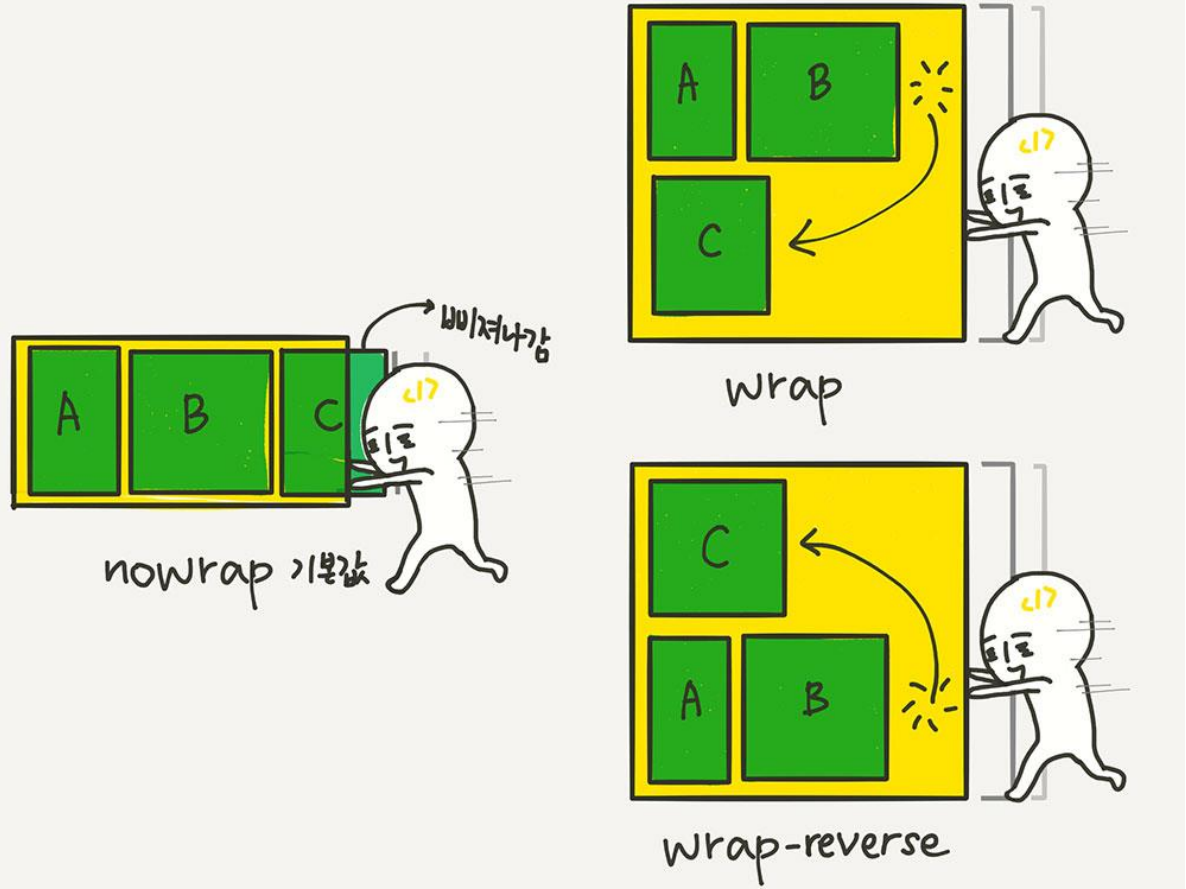
속성	설명
row	좌에서 우로 배치
row-reverse	우에서 좌로 배치
column	위에서 아래로 배치
column-reverse	아래에서 위로 배치



Flex Container 옵션

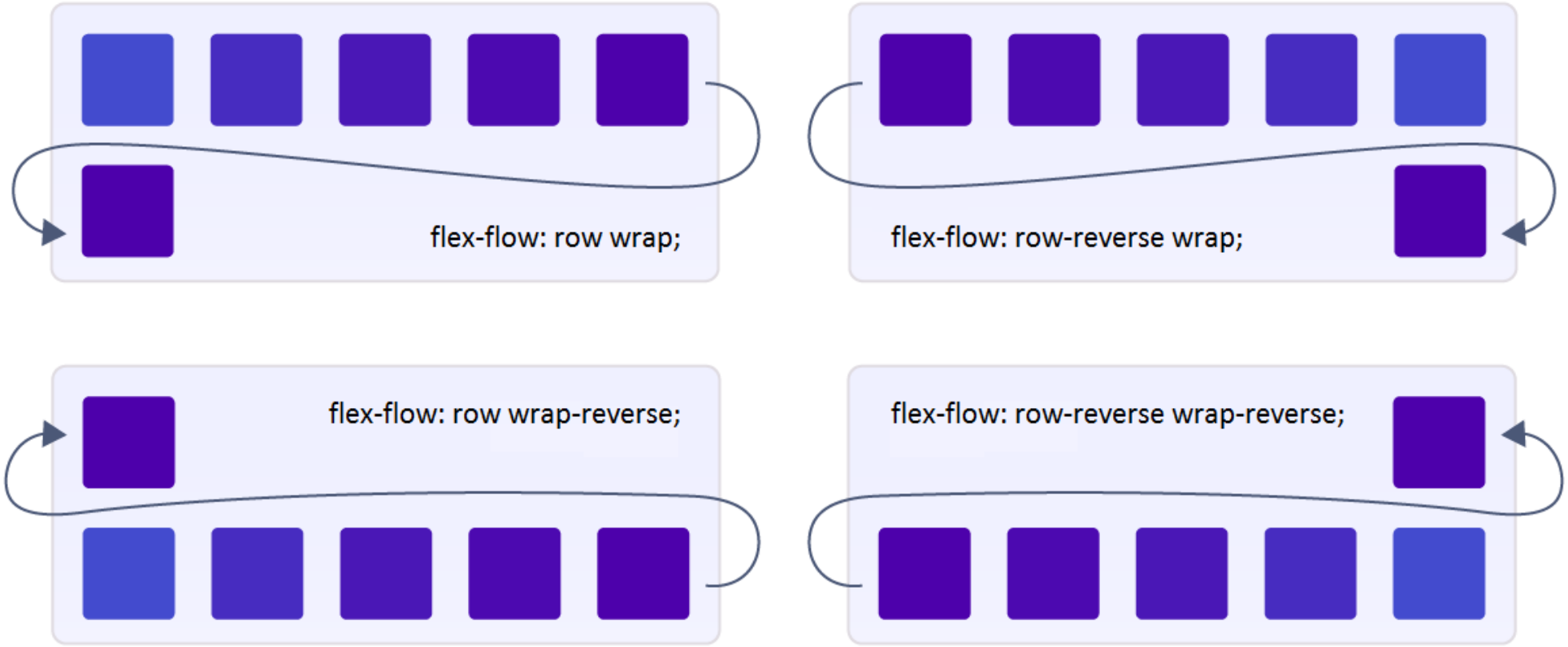
줄넘김 처리 설정 – flex-wrap

속성	설명
nowrap	좌에서 우로 배치
wrap	우에서 좌로 배치
wrap-reverse	위에서 아래로 배치



Flex Container 옵션

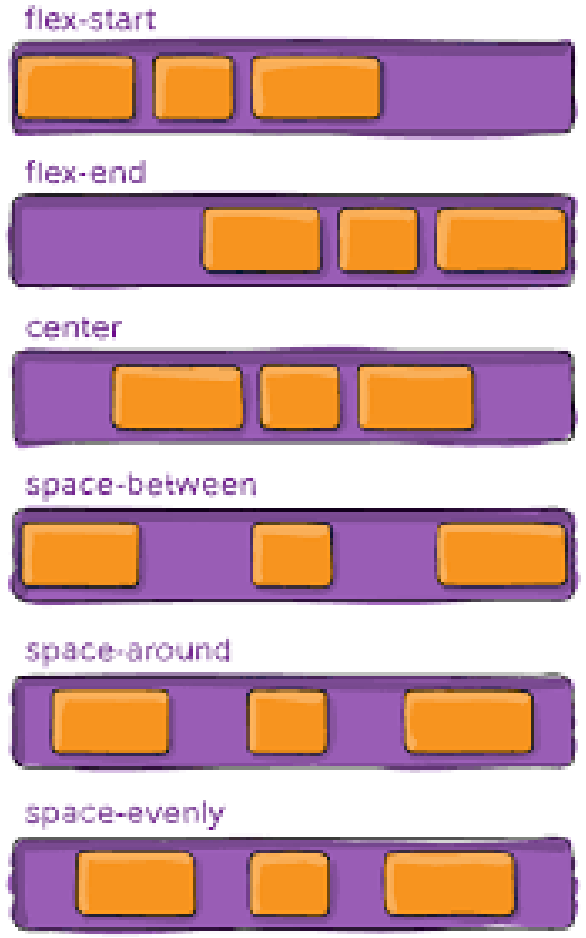
flex-direction 과 flex-wrap 을 같이 처리할 수 있는 옵션 – flex-flow



Flex Container 옵션

메인축을 기준으로 정렬하는 옵션 – justify-content

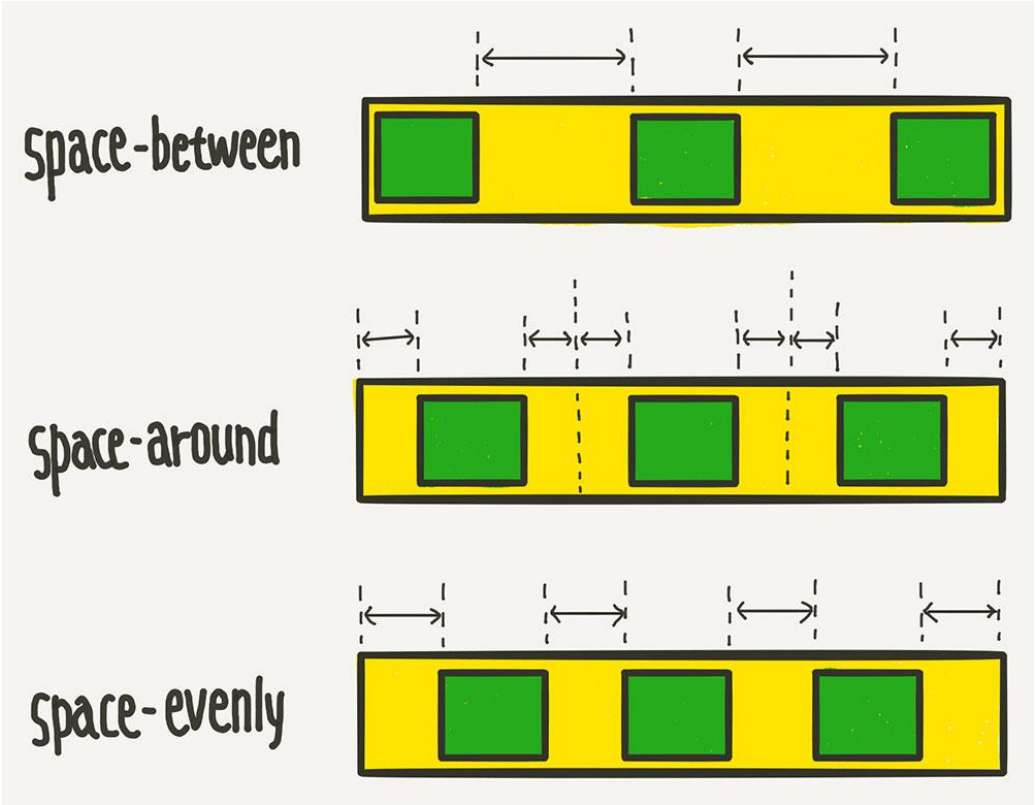
속성	설명
flex-start	좌에서 우로 배치
flex-end	우에서 좌로 배치
center	중앙에 배치
space-between	아이템들 사이에 균일한 간격을 만들어 배치
space-around	아이템들 둘레에 균일한 간격을 만들어 배치
space-evenly	아이템들 사이 및 양 끝에 균일한 간격을 만들어 배치



Flex Container 옵션

메인축을 기준으로 정렬하는 옵션 – justify-content

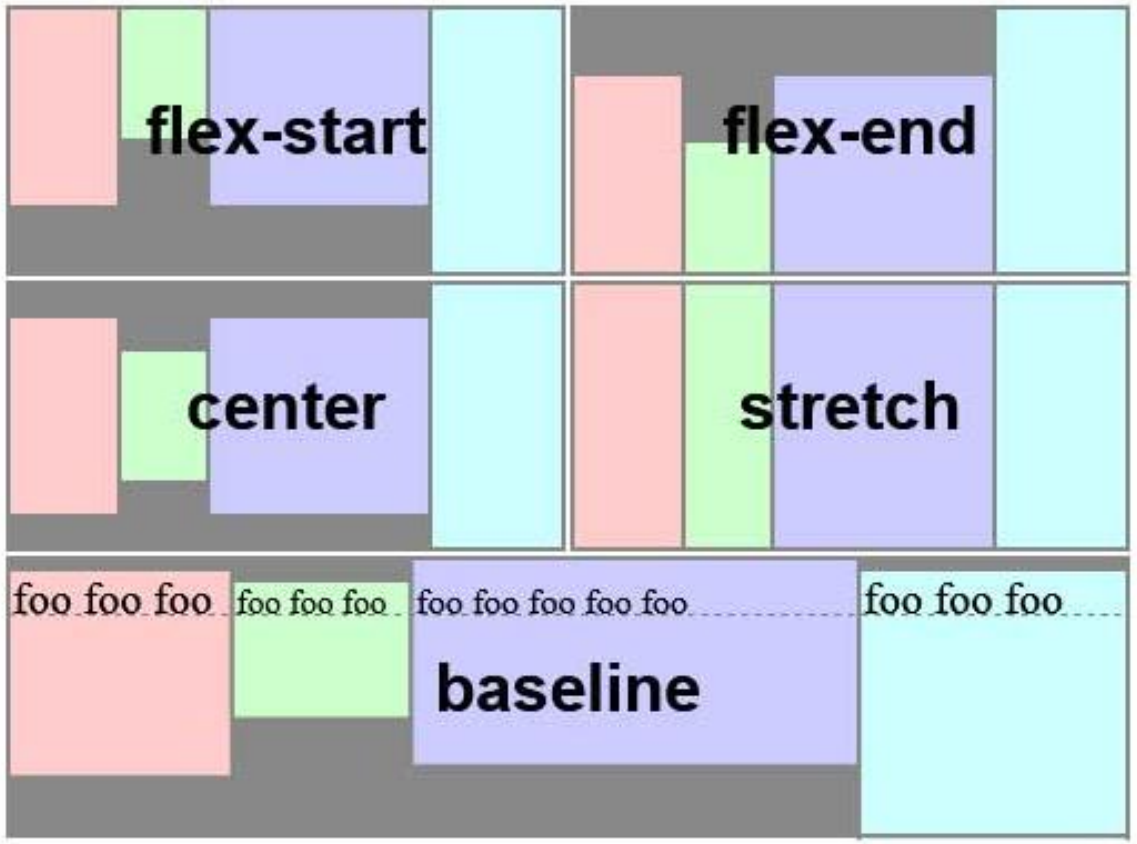
=> space-between, space-around, space-evenly의 차이점



Flex Container 옵션

교차축을 기준으로 정렬하는 옵션 – align-items

속성	설명
stretch	기본 배치
flex-start	위를 기준으로 배치
flex-end	아래를 기준으로 배치
center	중앙을 기준으로 배치
baseline	컨텐츠(텍스트)를 기준선을 잡아 배치

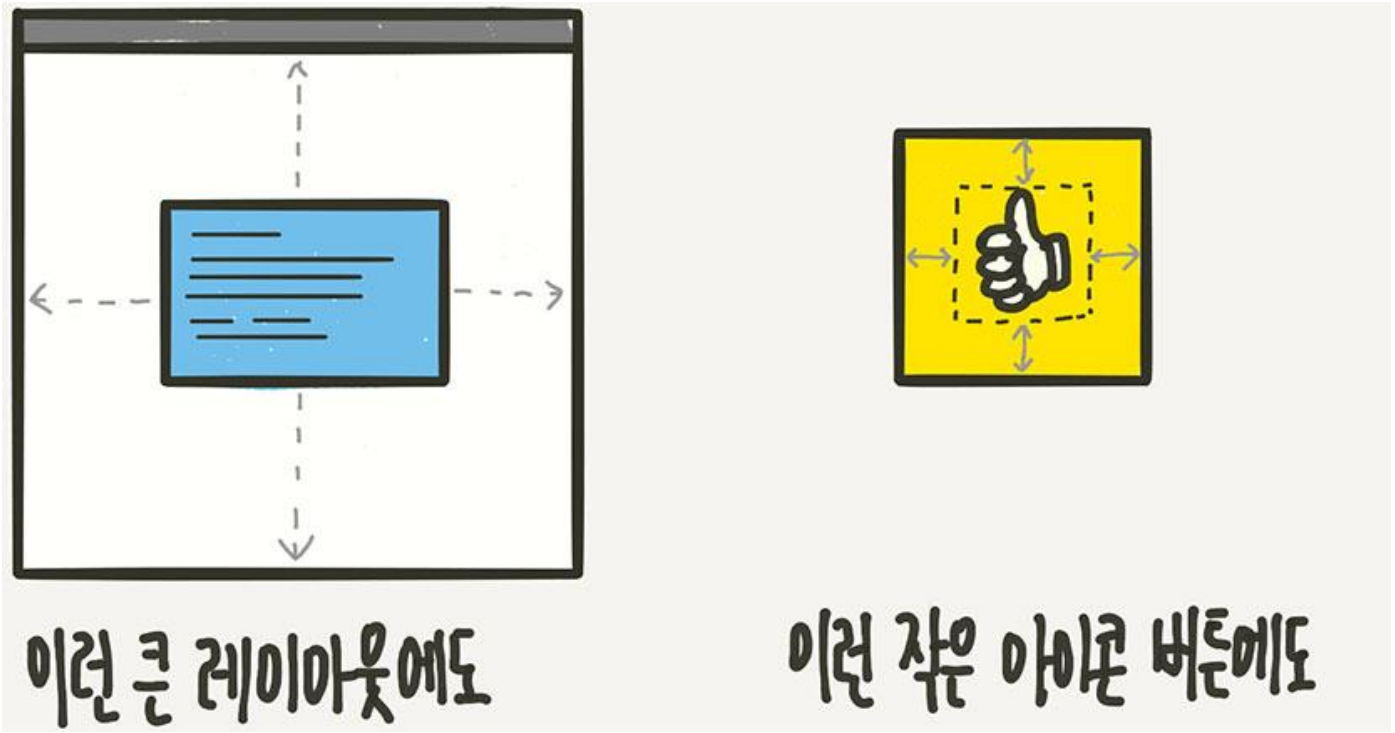


Flex Container 옵션

아이템을 한 가운데 놓는 방법

justify-content: center;

align-item : center

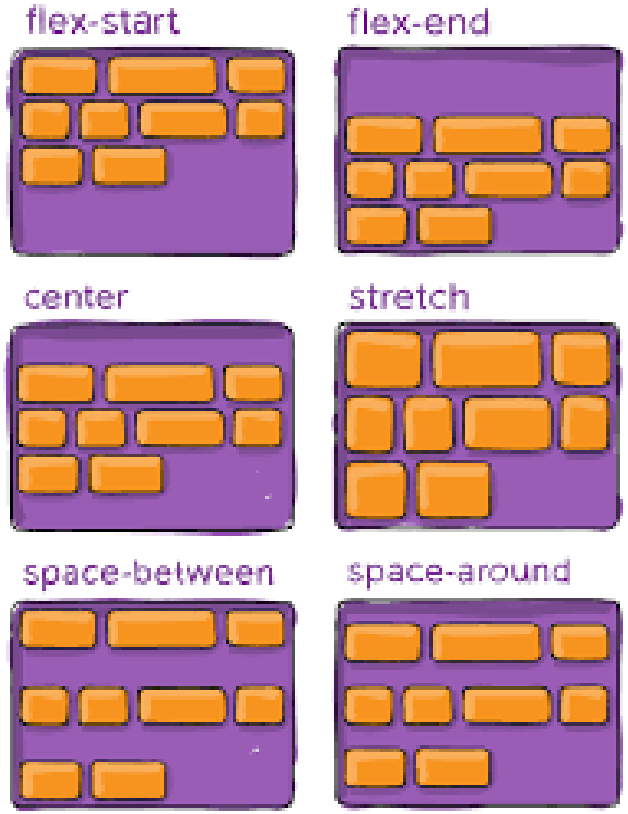


Flex Container 옵션

여러 행 정렬 옵션 – align-content

flex-wrap : wrap; 이 설정된 상태에서, 아이템들의 행이 2줄 이상 되었을 시 수직축 방향 정렬을 결정하는 속성

속성	설명
stretch	기본 배치
flex-start	위를 기준으로 배치
flex-end	아래를 기준으로 배치
center	중앙을 기준으로 배치
space-between	위 아래로 아이템들 사이에 균일한 간격을 만들어 배치
space-around	아이템들의 둘레에 균일한 간격을 만들어 배치
space-evenly	아이템들의 사이와 양 끝에 균일한 간격을 만들어 배치



Flex Item 옵션

flex-basis : Flex 아이템의 기본 크기를 설정
- flex-direction이 row일 때는 너비, column일 때는 높이
- 컨텐츠가 flex-basis를 넘어갈 경우 컨텐츠 크기에 맞춰 적용
- flex-basis의 값으로는 우리가 사용하는 각종 단위의 수가 들어갈 수 있다.
- 기본값은 auto

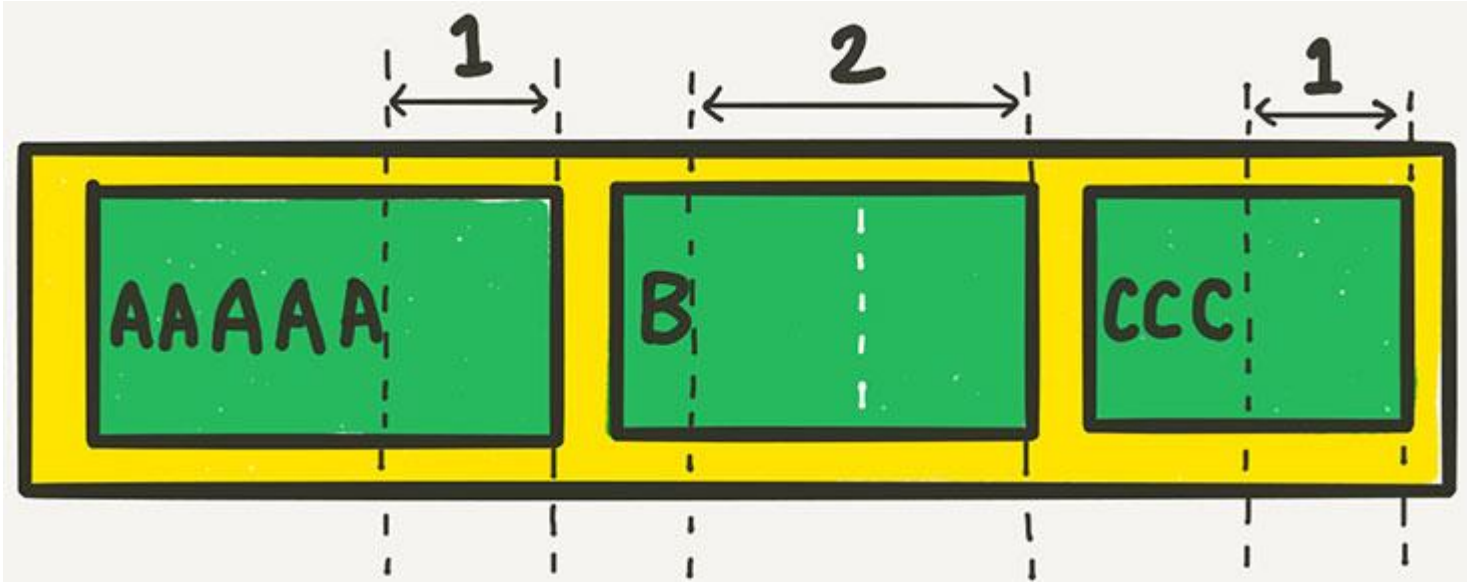
사용되는 수치	설명
px, pt, em, rem, %	일반적으로 사용되는 수치
content	컨텐츠의 크기

Flex Item 옵션

flex-grow : flex아이템을 컨테이너 크기에 맞춰서 확장

- 아이템들의 flex-basis를 제외한 여백 부분을 flex-grow에 지정된 숫자의 비율로 나누어 가진다

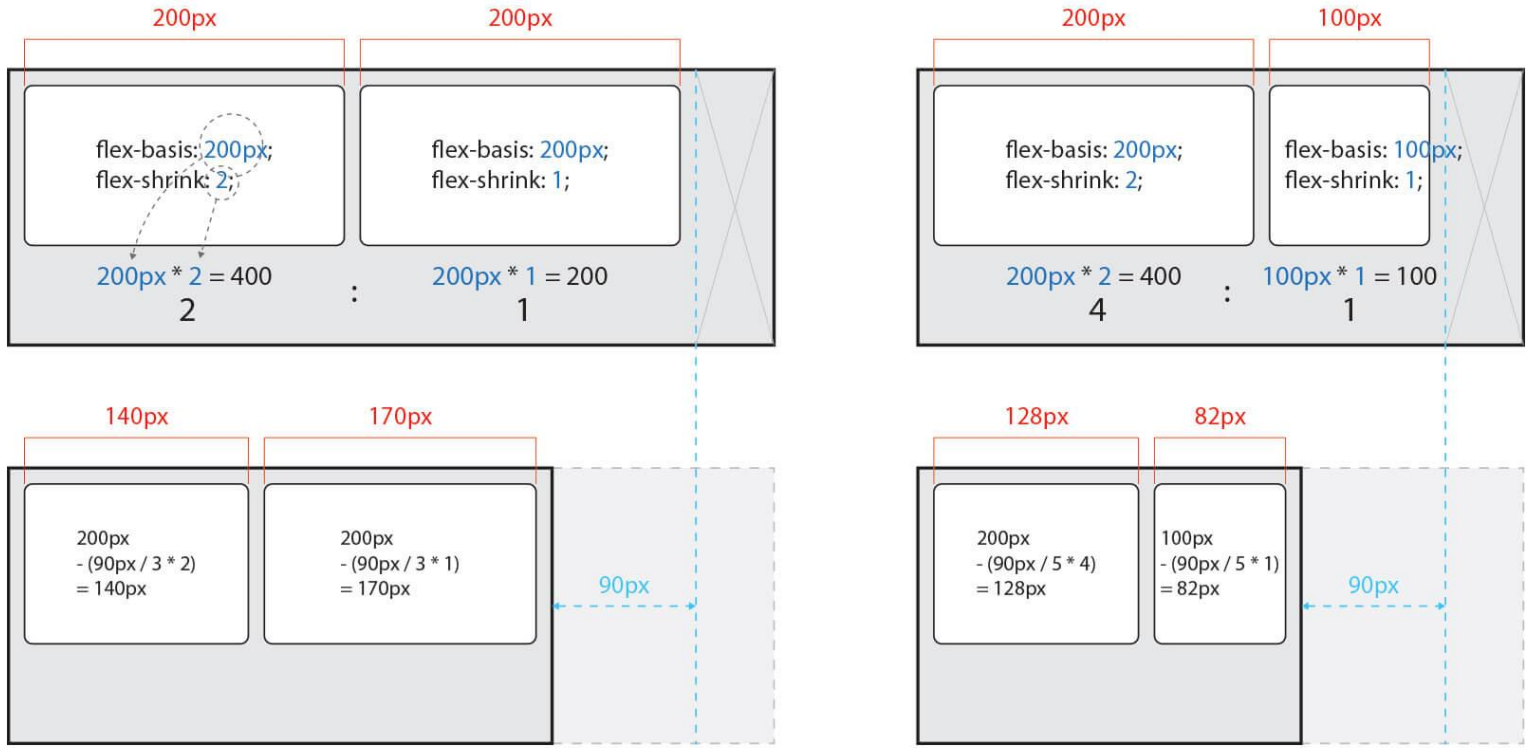
```
/* 1:2:1의 비율로 세팅할 경우 */
.item:nth-child(1) { flex-grow: 1; }
.item:nth-child(2) { flex-grow: 2; }
.item:nth-child(3) { flex-grow: 1; }
```



Flex Item 옵션

flex-shirnk : 아이템이 flex-basis의 값보다 작아질 수 있는지를 결정

- flex-shirnk 는 아이템의 감소 너비 비율을 설정한다(계산이 까다로움)



Flex Item 옵션

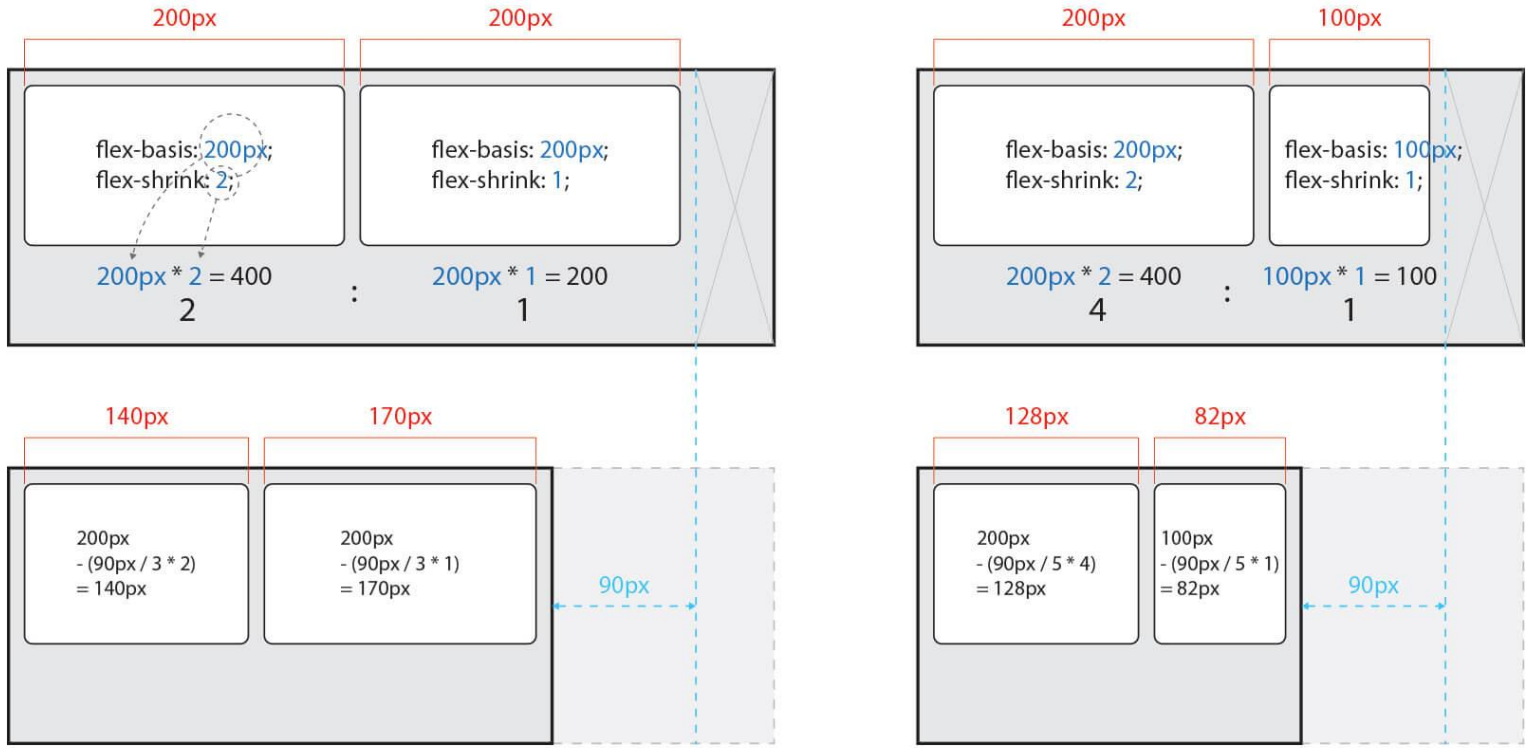
flex : flex-grow, flex-shrink, flex-basis를 한 번에 쓸 수 있는 축약형 속성

```
.item {
        flex: 1;
        /* flex-grow: 1; flex-shrink: 1; flex-basis: 0%; */
        flex: 1 1 auto;
        /* flex-grow: 1; flex-shrink: 1; flex-basis: auto; */
        flex: 1 500px;
        /* flex-grow: 1; flex-shrink: 1; flex-basis: 500px; */
}
```

Flex Item 옵션

flex-shirnk : 아이템이 flex-basis의 값보다 작아질 수 있는지를 결정

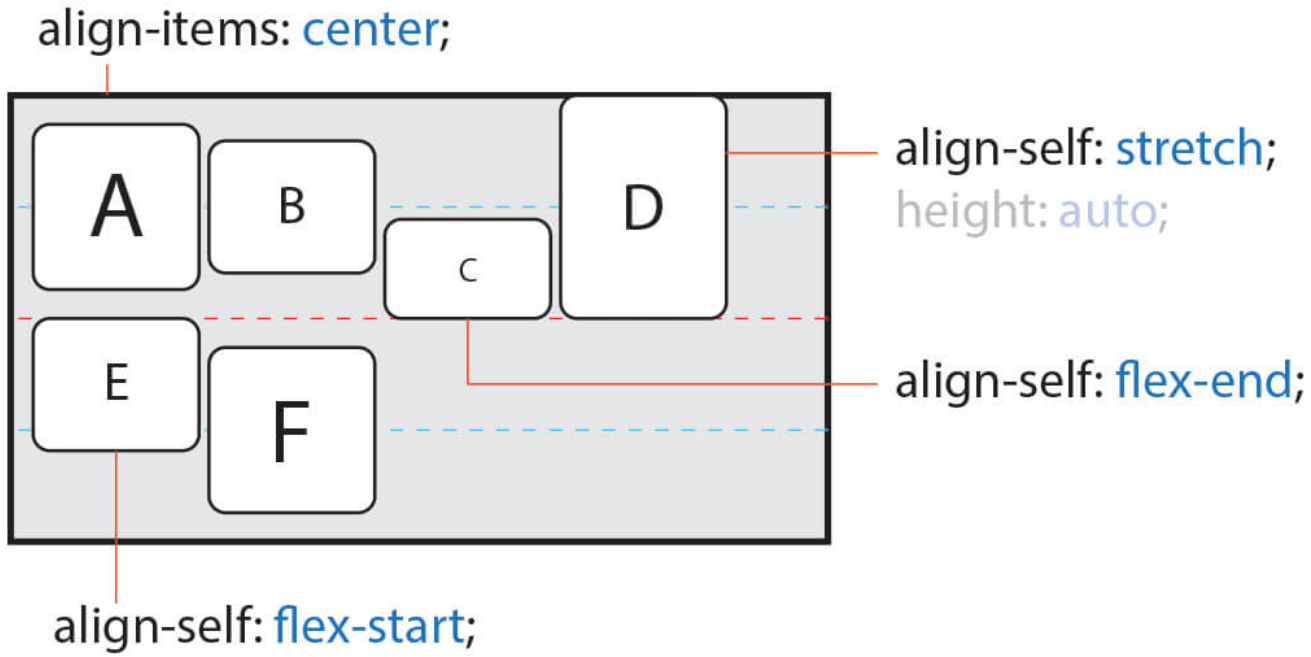
- flex-shirnk 는 아이템의 감소 너비 비율을 설정한다(계산이 까다로움)



Flex Item 옵션

align-self : 아이템을 수직축으로 정렬한다.

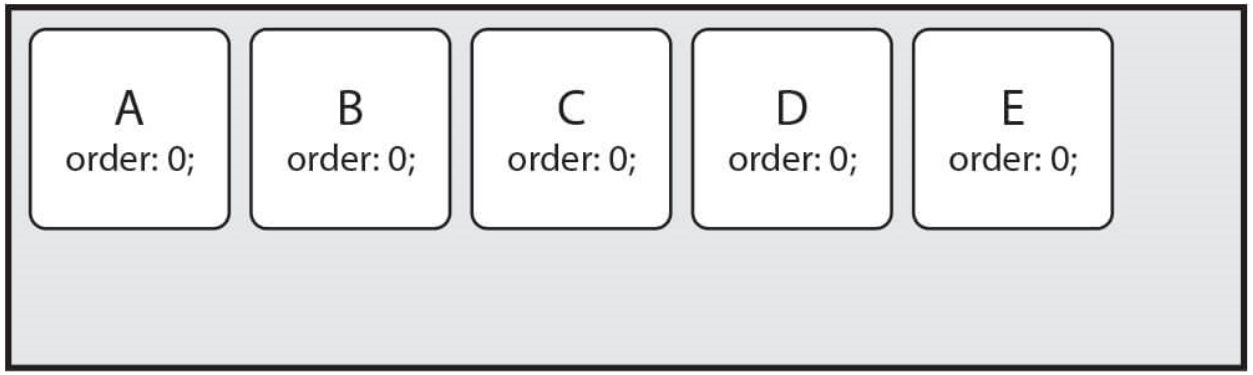
- align-self는 align-items보다 우선권이 있다.
- 속성 값은 align-items와 동일

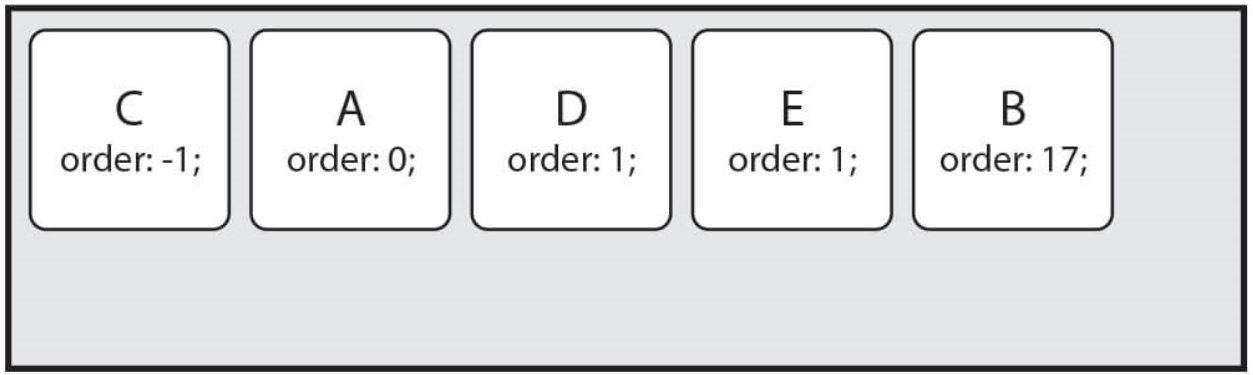


Flex Item 옵션

order : 아이템의 배치 순서를 바꾼다

- Item에 숫자를 지정하고 숫자가 클수록 순서가 뒤로 간다.





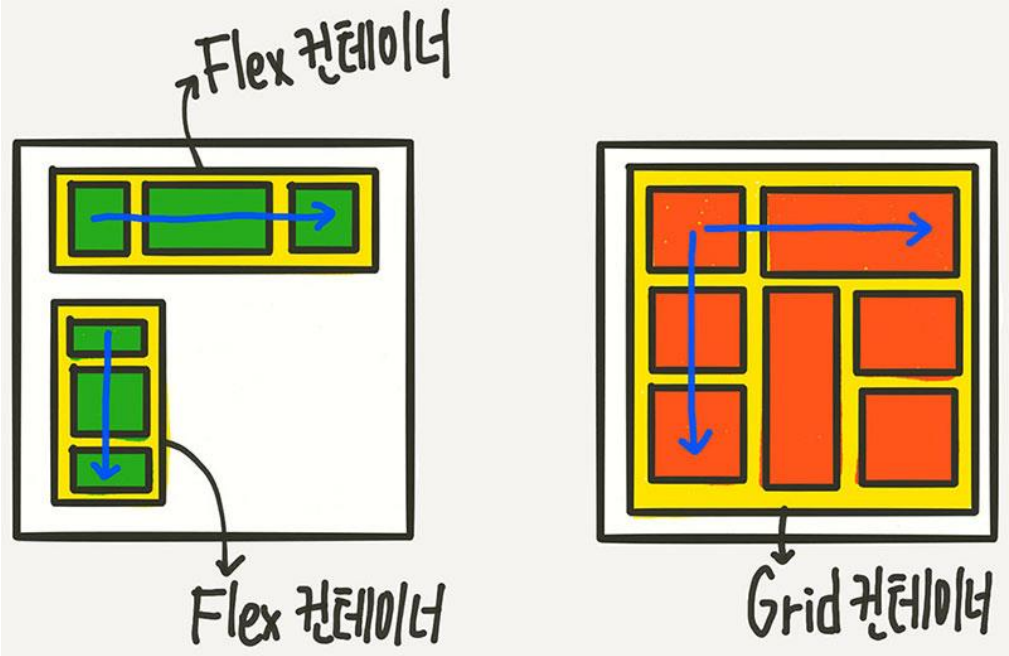
Flex Item 옵션

z-index : position에서의 z-index랑 동일

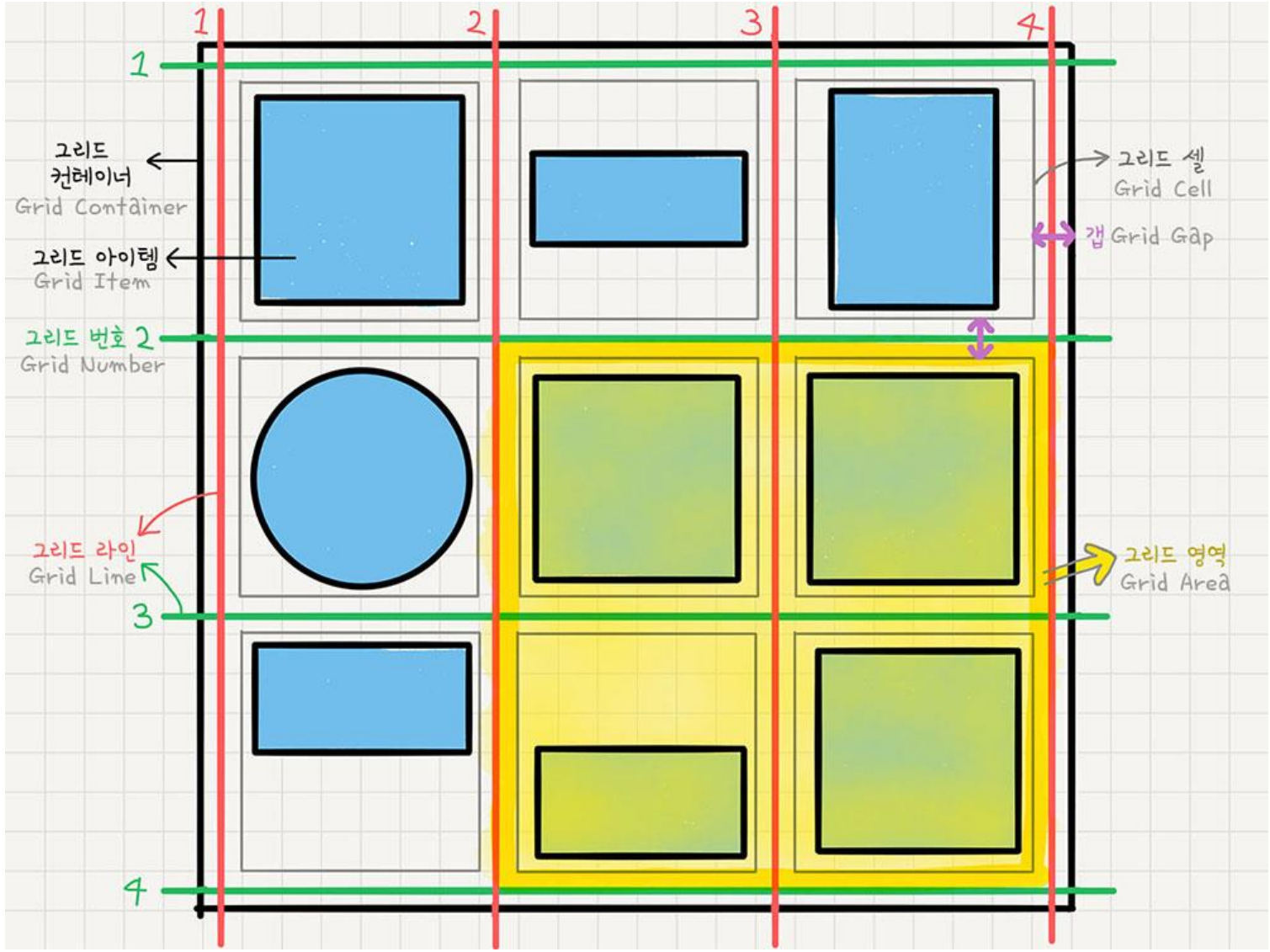
Grid

Grid 란?

- 격자 모양의 레이아웃을 표현하기 위한 기능
- Flex는 한방향의 레이아웃을 지원하지만 Grid는 가로,세로 양방향의 레이아웃을 지원한다.
- 페이지의 전체적인 구조를 잡는데 상당히 편리함



Grid 기본 용어 정리



용어	설명
그리드 컨테이너	display: grid를 적용하는, Grid의 전체 영역
그리드 아이템	Grid 컨테이너의 자식 요소들
그리드 트랙	Grid의 행(Row) 또는 열(Column)
그리드 셀	Grid의 한 칸
그리드 라인	Grid 셀을 구분하는 선
그리드 번호	Grid 라인의 각 번호
그리드 갭	Grid 셀 사이의 간격
그리드 영역	Grid 라인으로 둘러싸인 사각형 영역으로, 그리드 셀의 집합

Grid Container 옵션

Container에 Grid 속성을 입히기 위한 필수 display 옵션

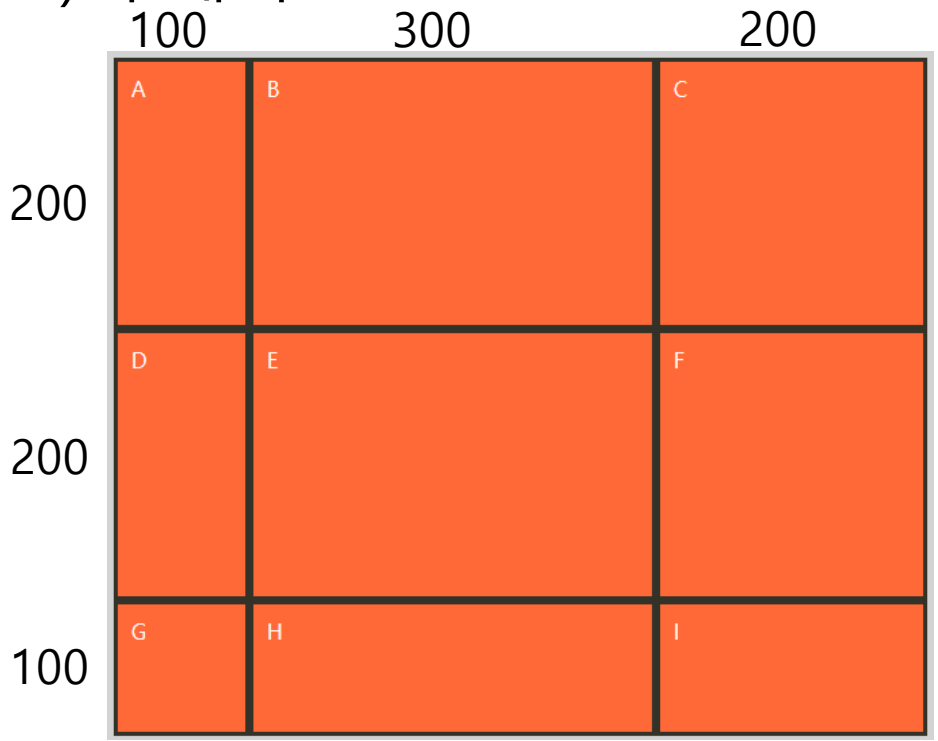
속성	설명
grid	Block 특성의 Grid Container를 정의
inline-grid	Inline 특성의 Grid Container를 정의

Grid Container 옵션

grid-template-rows, grid-template-columns

- 가로 세로 아이템들의 크기를 정의하는 옵션
- grid-template-rows는 행(row)의 배치
- grid-template-columns는 열(column)의 배치

```
grid-template-columns: 100px 300px 200px;
grid-template-rows: 200px 200px 100px;
```



Grid Container 옵션

fr

- 그리드에서는 fr이라는 단위를 사용
- fraction의 약자로 숫자 비율대로 트랙의 크기를 나눈다.
- 사이즈의 변화를 원하지 않는 트랙이 있을 경우 해당 트랙은 fr을 사용하지 않고 고정값을 사용하여 변화를 막을 수 있다.

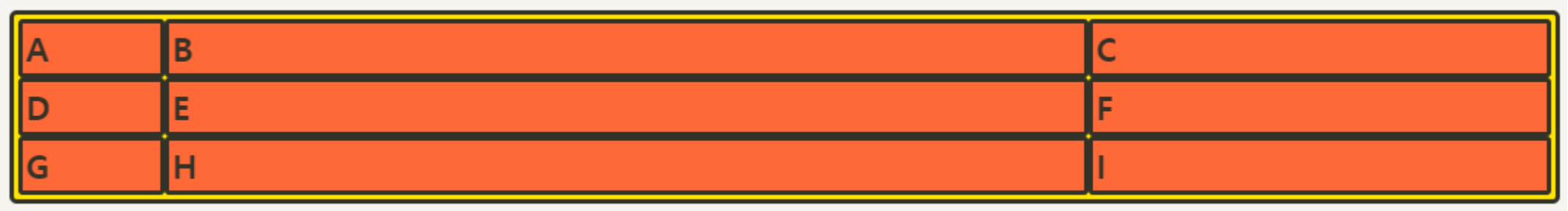
```
grid-template-columns: 1fr 1fr 1fr;
```

Grid Container 옵션

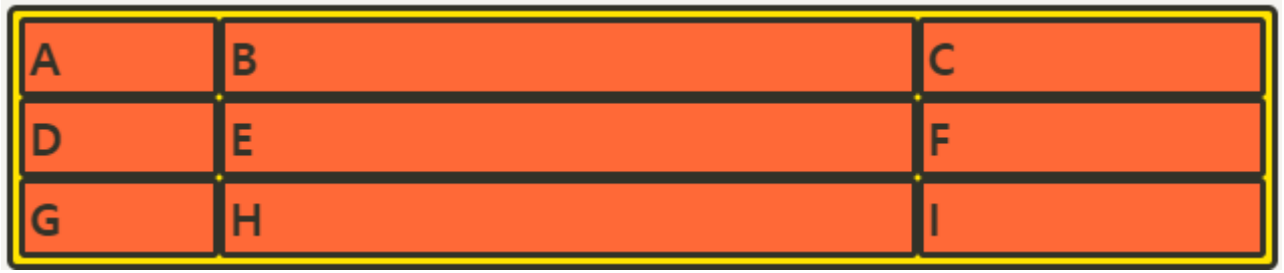
fr

```
grid-template-columns: 100px 2fr 1fr;
```

100%



60%



Grid Container 옵션

Auto

- grid-template에서 auto를 사용할 경우 해당 셀은 다른 셀이 차지한 공간을 제외한 나머지 공간을 전부 차지하게 된다.

70px auto

Grid Container 옵션

repeat

- 반복되는 값을 자동으로 처리할 수 있는 함수
- 사용방법 : repeat(반복횟수, 반복값)

```
grid-template-columns: repeat(5, 1fr);

/* grid-template-columns: 1fr 1fr 1fr 1fr 1fr */
```

Grid Container 옵션

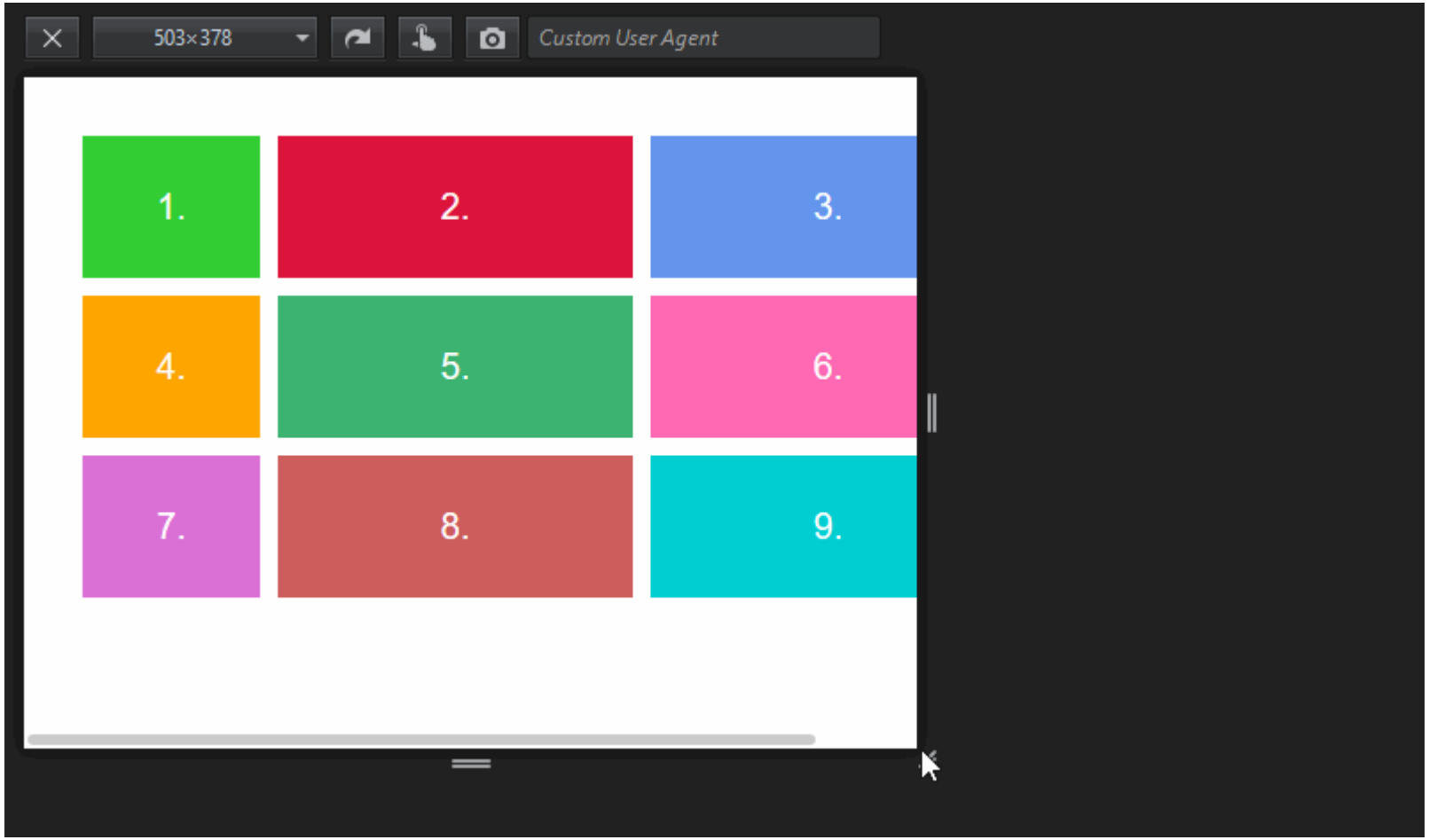
minmax

- 최소값과 최댓값을 지정할 수 있는 함수
- 사용방법 : minmax(최소값, 최대값)
- 단독으로 사용도 되지만 repeat 함수와 같이 사용도 가능하다.

Grid Container 옵션

minmax

grid-template-columns: minmax(100px, 200px) 200px 200px;



Grid Container 옵션

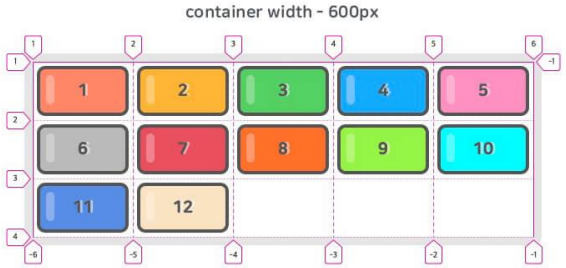
auto-fill, auto-fit

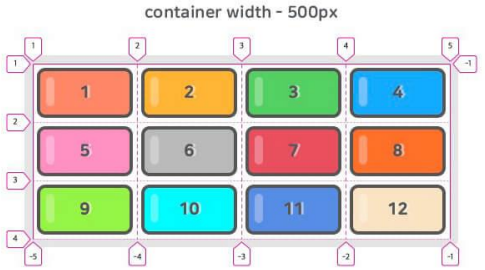
- repeat 함수와 같이 사용되는 옵션 함수.
- 반복 횟수에 이 함수가 사용되며 컨테이너의 크기가 아이템들을 수용하기 충분하지 않을 경우 아이템을 자동으로 줄 바꿈 처리하며, 그에 맞게 암시적 행/열도 자동으로 수정한다.
- auto-fill과 auto-fit의 차이점은 그리드 컨테이너가 하나의 행/열(Track)에 모든 아이템을 수용하고 남는 공간이 있을 때 발생한다.
- auto-fill은 남는 공간(빈 트랙)을 그대로 유지하고, auto-fit은 남는 공간을 축소한다.

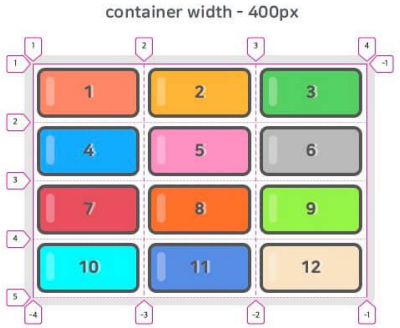
Grid Container 옵션

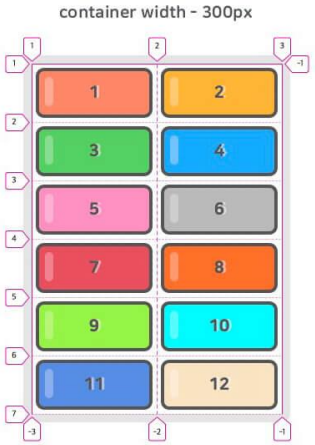
auto-fill, auto-fit

repeat(auto-fill, minmax(120px, 1fr)





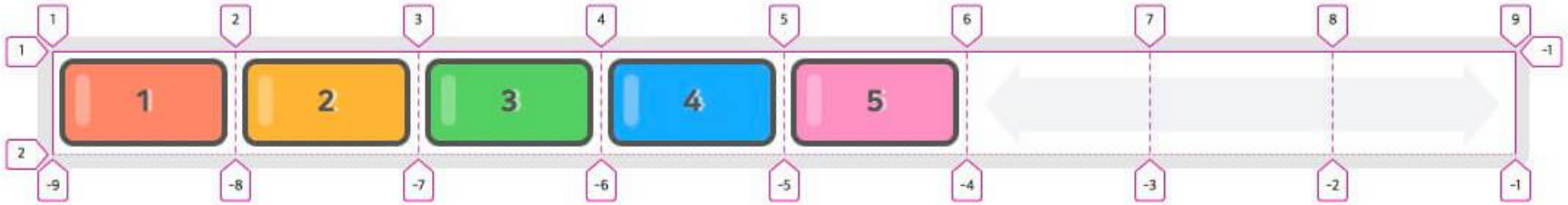




Grid Container 옵션

auto-fill, auto-fit

grid-template-columns: repeat(**auto-fill**, minmax(120px, 1fr));



grid-template-columns: repeat(**auto-fit**, minmax(120px, 1fr));

Grid Container 옵션

간격설정 : row-gap, column-gap, gap

- row-gap : 가로줄 간격을 설정
- column-gap : 세로줄 간격을 설정
- gap : 둘 다 설정 (첫번째가 row, 두번째가 column)

gap : 10px;

영역 지정 옵션

각 셀의 영역을 지정하는 속성

- 하나의 셀이 하나 이상의 그리드를 차지할 수 있으며 이때 사용하는 옵션이 아래와 같은 옵션이다.

속성	설명
grid-column-start	셀이 시작하는 열 시작 라인 번호
grid-column-end	셀이 끝나는 열 끝 라인 번호
grid-column	grid-column-start + grid-column-end
grid-row-start	셀이 시작하는 행 시작 라인 번호
grid-row-end	셀이 끝나는 행 끝 라인 번호
grid-row	grid-row-start + grid-row-end
grid-area	grid-row-start + grid-column-start + grid-row-end + grid-column-end

영역 지정 옵션

각 셀의 영역을 지정하는 속성

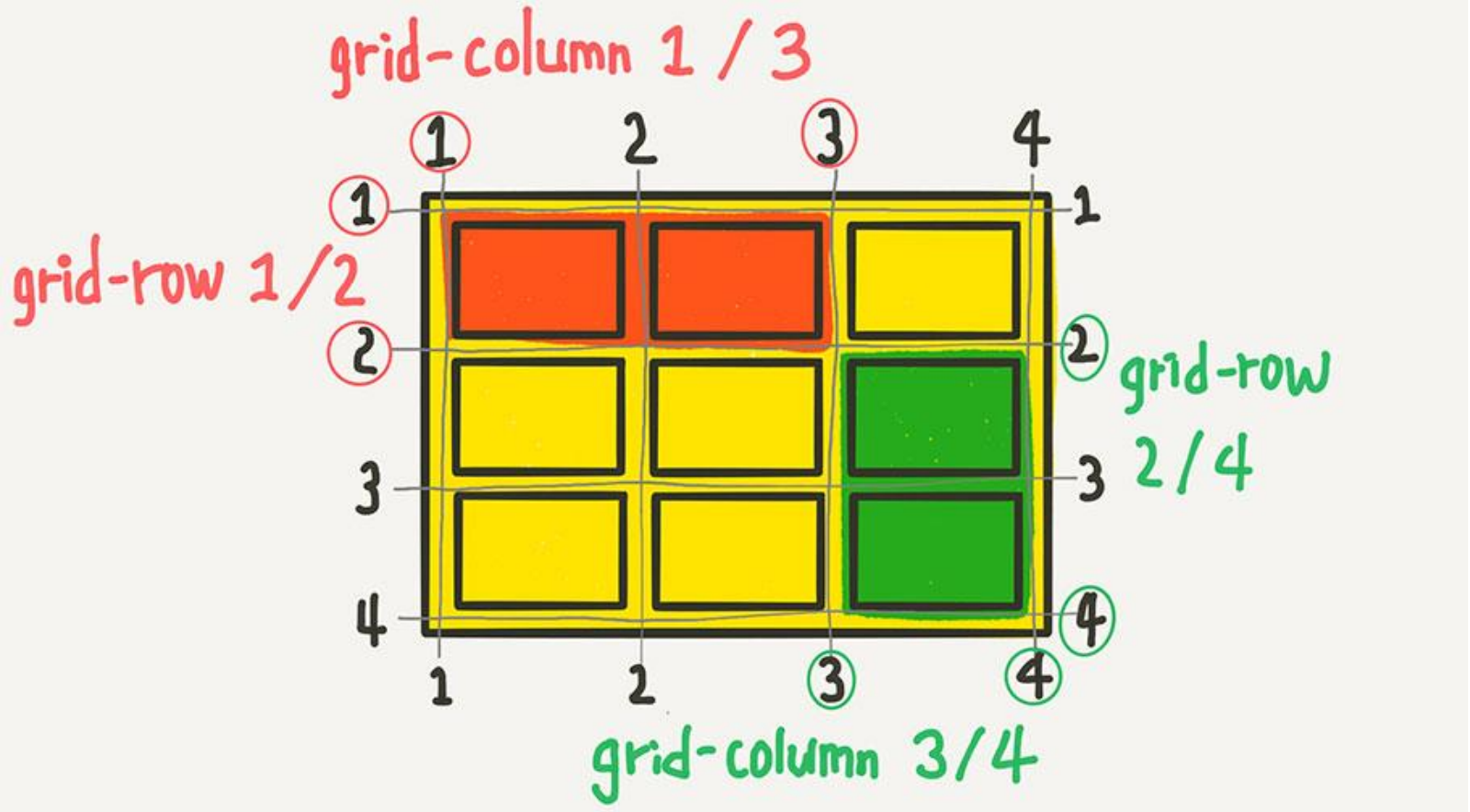
- 여기서 설정한 라인 만큼 셀이 공간을 차지한다.
- 주의할 점은 셀 기준이 아닌 라인 기준이다.

```
.item:nth-child(1) {
        grid-column-start: 1;
        grid-column-end: 3;
        grid-row-start: 1;
        grid-row-end: 2;
}
```

```
.item:nth-child(1) {
        grid-column: 1 / 3;
        grid-row: 1 / 2;
}
```

영역 지정 옵션

각 셀의 영역을 지정하는 속성



영역 지정 옵션

각 셀의 영역을 지정하는 속성

- 몇 개의 셀을 차지할지 설정하기 위해서는 span 이라는 값을 이용한다.

```
.item:nth-child(1) {
        /* 1번 라인에서 2칸 */
        grid-column: 1 / span 2;
        /* 1번 라인에서 3칸 */
        grid-row: 1 / span 3;
}
```

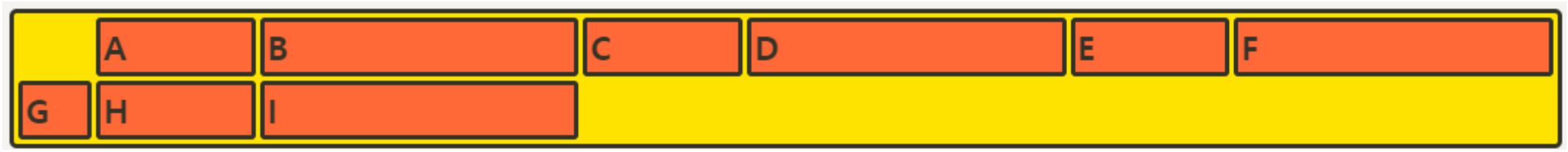
영역 지정 옵션

각 셀의 영역을 지정하는 속성

- grid-column을 활용해 grid-auto-columns과 grid-template-columns

의 통제를 받지 않는 column들의 배치를 할 수 있다.

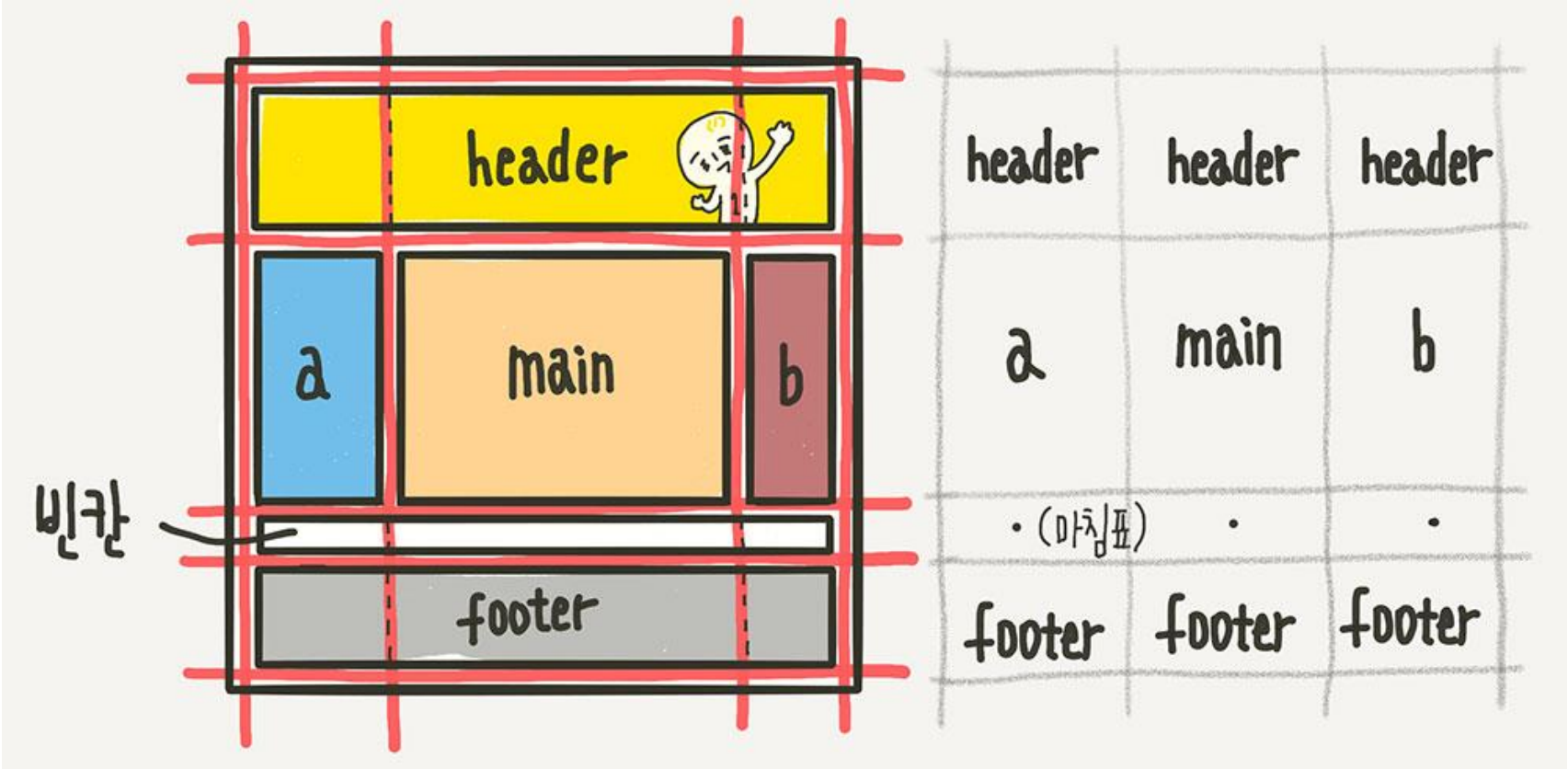
```
.container {
        grid-template-columns: 50px;
        grid-auto-columns: 1fr 2fr;
}
.item:nth-child(1) { grid-column: 2; }
.item:nth-child(2) { grid-column: 3; }
.item:nth-child(3) { grid-column: 4; }
.item:nth-child(4) { grid-column: 5; }
.item:nth-child(5) { grid-column: 6; }
.item:nth-child(6) { grid-column: 7; }
/* end를 생략하면 그냥 한 칸임 */
```



영역 지정 옵션

영역 이름으로 그리드 정의 : grid-template-areas

- 각 영역(Grid Area)에 이름을 붙이고, 그 이름을 이용해서 배치하는 방법



영역 지정 옵션

자동 배치 : grid-auto-flow

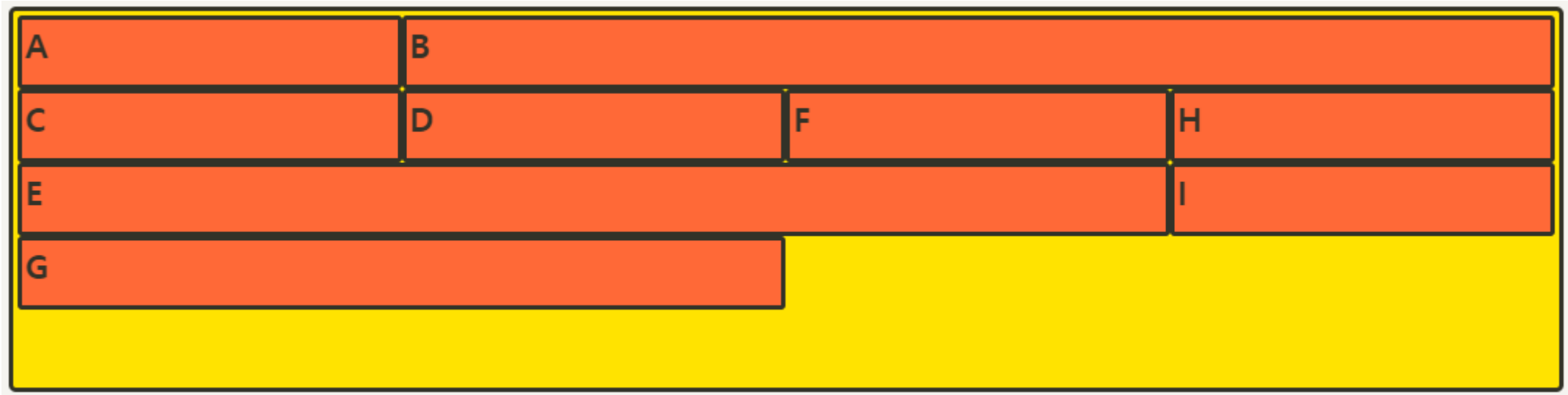
- 배치하지 않은 아이템(Item)을 어떤 방식의 '자동 배치 알고리즘'으로 처리할지 정의하는 속성

속성	설명
row	각 행 축을 따라 차례로 배치
column	각 열 축을 따라 차례로 배치
row dense(dense)	각 행 축을 따라 차례로 배치, 빈 영역 메움
column dense	각 열 축을 따라 차례로 배치, 빈 영역 메움

영역 지정 옵션

자동 배치 : grid-auto-flow

```
.container {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(25%, auto));
        grid-template-rows: repeat(5, minmax(50px,auto));
        grid-auto-flow: dense;
}
item:nth-child(2) { grid-column: auto / span 3; }
item:nth-child(5) { grid-column: auto / span 3; }
item:nth-child(7) { grid-column: auto / span 2; }
```

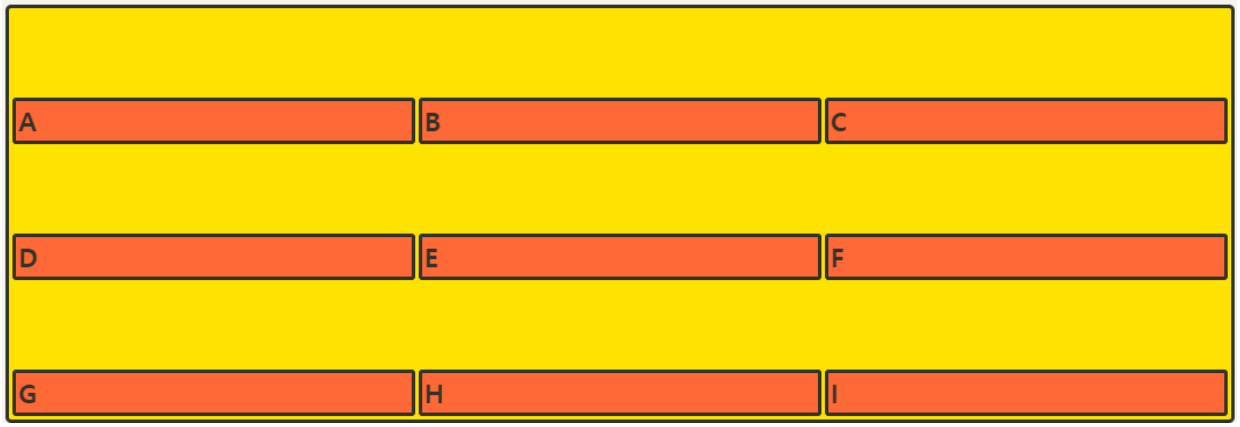


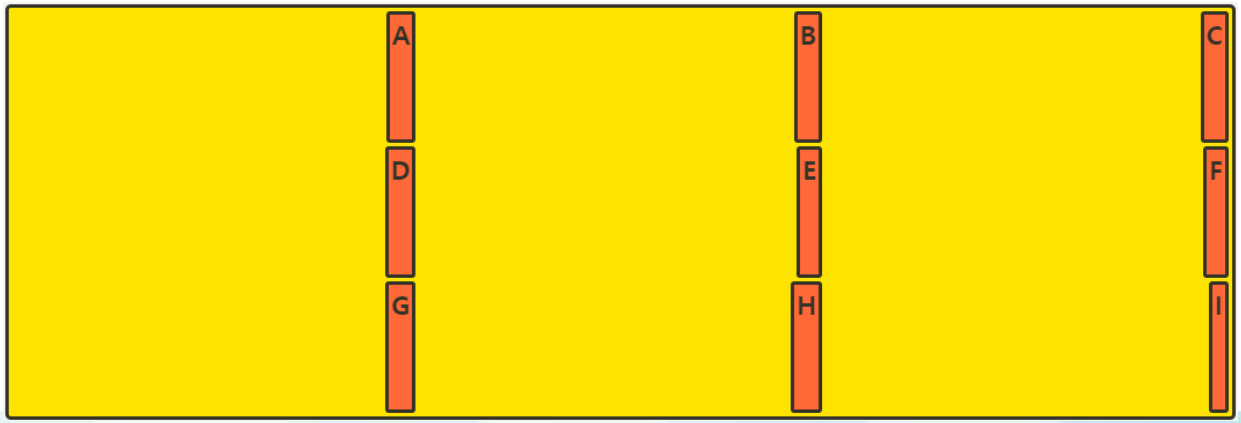
정렬 옵션

세로 방향 정렬 : align-items　　**가로 방향 정렬 : justify-items**

- 컨테이너 옵션이며 셀들을 열 방향 기준, 행 방향 기준 으로 정렬한다.

속성	설명
stretch	세로확장 / 가로확장 (default)
start	위로 배치 / 왼쪽으로 배치
center	중앙에 배치
end	아래 배치 / 오른쪽으로 배치

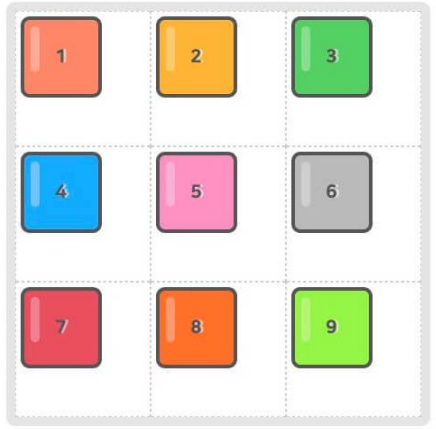




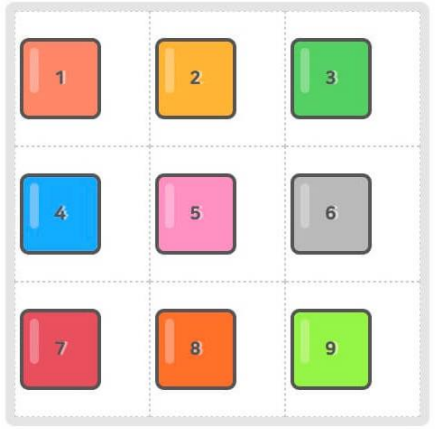
정렬 옵션

세로 방향 정렬 : align-items

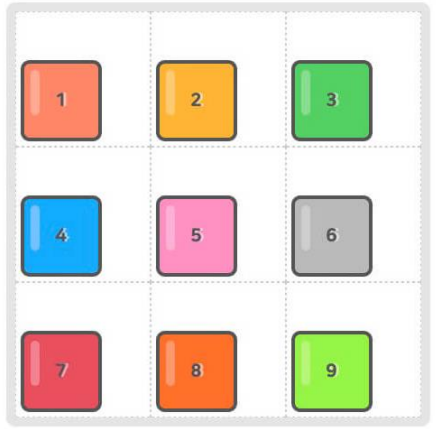
align-items: start;



align-items: center;



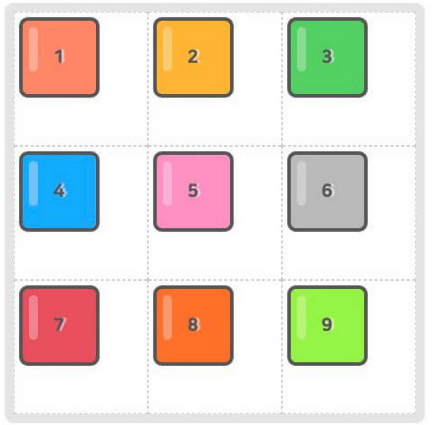
align-items: end;



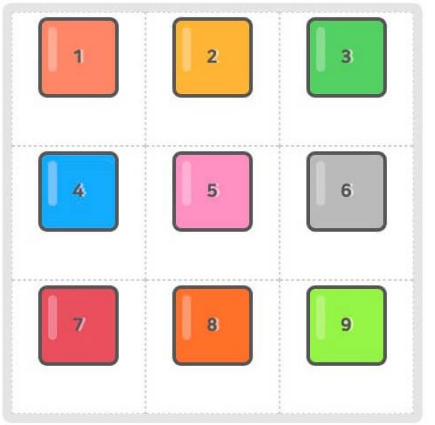
align-items: stretch;

가로 방향 정렬 : justify-items

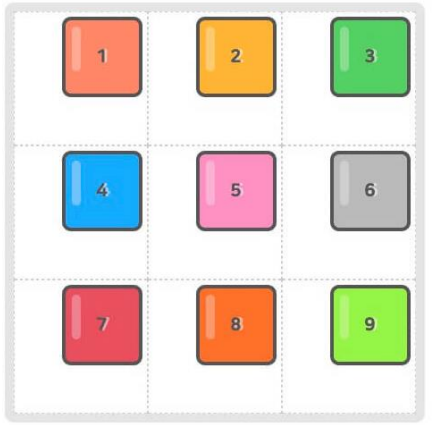
justify-items: start;



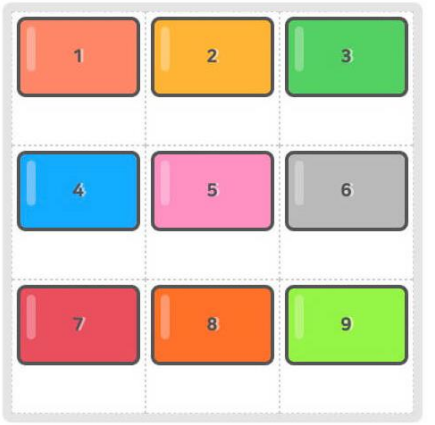
justify-items: center;



justify-items: end;



justify-items: stretch;



정렬 옵션

세로 방향 정렬 : align-items　　**가로 방향 정렬 : justify-items**

- 두개의 옵션을 같이 쓰고 싶다면 place-items를 쓸 수 있다.

```
.container {
  place-items: <align-items> <justify-items>;
}
```

정렬 옵션

아이템 그룹 세로 정렬 : align-content
아이템 그룹 가로 정렬 : justify-content
아이템 그룹 세로/가로 정렬 : space-content

- align-content 는 Grid 아이템들의 높이를 모두 합한 값이 Grid 컨테이너의 높이보다 작을 때 Grid 아이템들을 통째로 정렬한다.
- justify-content 는 Grid 아이템들의 너비를 모두 합한 값이 Grid 컨테이너의 너비보다 작을 때 Grid 아이템들을 통째로 정렬한다.
- 두개를 다 사용할 경우 space-content를 사용한다.
  => space-content : <align-content> <justify-content>

정렬 옵션

아이템 그룹 세로 정렬 : align-content

아이템 그룹 가로 정렬 : justify-content

아이템 그룹 세로/가로 정렬 : space-content

- 사용하는 값은 아래와 같다.

값	설명(align-content/justify-content)
stretch(normal)	열 축/행 축을 채우기 위해 그리드 콘텐츠를 늘림
start	시작점(위쪽/왼쪽) 정렬
center	수직/수평 가운데 정렬
end	끝점(오른쪽/아래쪽) 정렬
space-around	각 열 좌우/행 위아래에 여백을 고르게 정렬
space-between	첫 행은 시작점에, 끝 행은 끝점에 정렬되고 나머지 여백으로 고르게 정렬
space-evenly	모든 여백을 고르게 정렬

정렬 옵션

align-content

justify-content

예제

정렬 옵션

개별 아이템 세로 정렬 : align-self

개별 아이템 가로 정렬 : justify-self

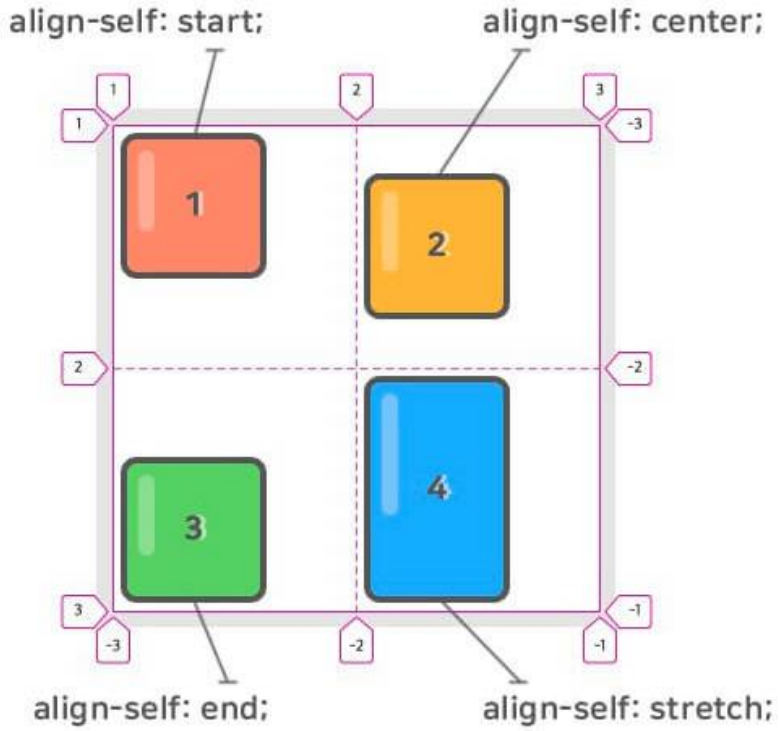
개별 아이템 세로/가로 정렬 : place-self

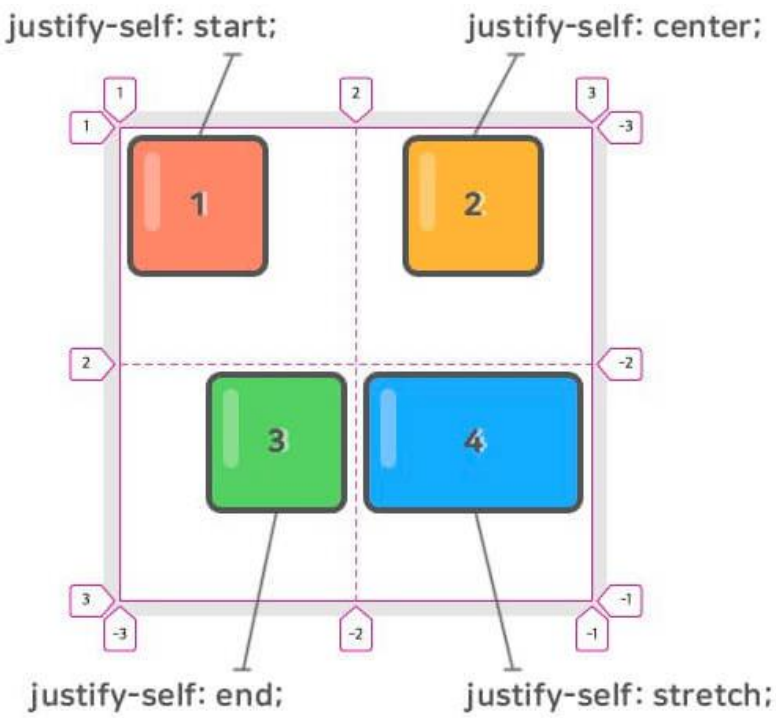
- 하는 역할과 사용법은 위의 속성과 동일하나 차이점은 각 개별 아이템에만 적용되며 아이템에 줘야 하는 속성이다.

값	설명(align-self/justify-self)
stretch(normal)	열 축/행 축을 채우기 위해 그리드 아이템을 늘림
start	시작점(위쪽/왼쪽) 정렬
center	수직/수평 가운데 정렬
end	끝점(오른쪽/아래쪽) 정렬

정렬 옵션

개별 아이템 세로 정렬 : align-self
개별 아이템 가로 정렬 : justify-self
개별 아이템 세로/가로 정렬 : place-self





그 외 옵션

배치 순서 : order

- 배치 순서를 정하는 옵션
- 숫자값이 들어가며, 작은 숫자일 수록 먼저 배치된다.
- "시각적" 순서일 뿐, HTML 자체의 구조를 바꾸는 것은 아니므로 접근성 측면에서 사용에 주의할 것.

```
.item:nth-child(1) { order: 3; } /* A */
.item:nth-child(2) { order: 1; } /* B */
.item:nth-child(3) { order: 2; } /* C */
```

그 외 옵션

Z축 정렬 : z-index

- 기존 z-index와 동일

```
.item:nth-child(5) {
        z-index: 1;
        transform: scale(2);
}
```