2주차: CSS 기초, Figma 실습

계명대학교 멋사 13기 프론트엔드 2팀

CSS

1. CSS란?

- CSS란?
- CSS장점
- CSS사용방법

2. CSS 적용

- 텍스트 서식 지정 및 색상
- BOX관련 CSS
- border, padding, margin, position

3. Flex, Grid

1. CSS란?

CSS란?

Cascading Style Sheets (CSS)

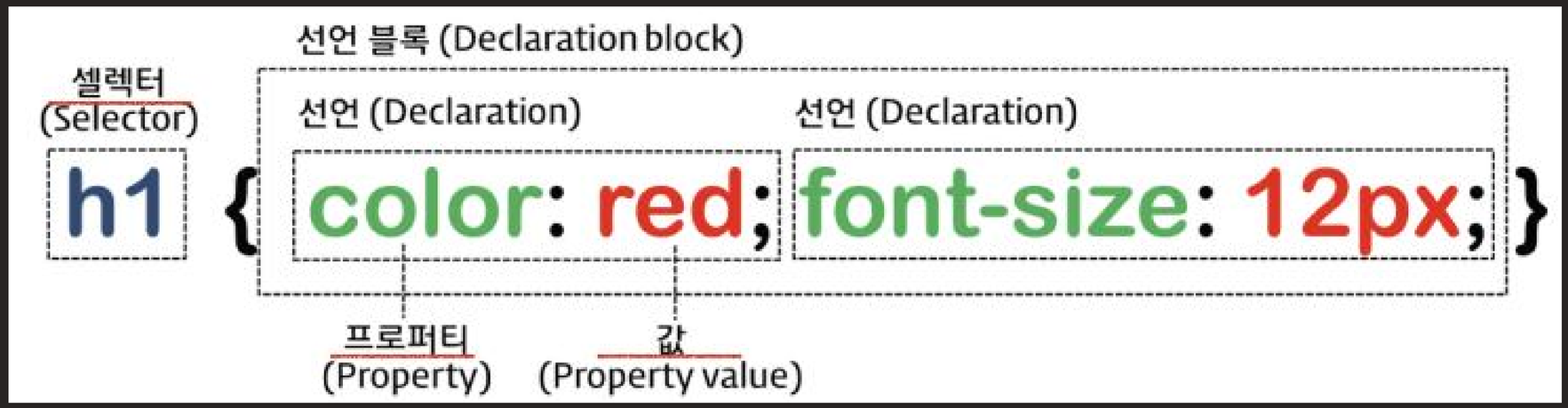
모양, 글꼴, 색상, 배경 이미지, 줄 간격, 페이지 레이아웃 등을 설명하는데 사용되는 스타일 시트 언어 (구조물)

CSS장점

- 정확한 유형 및 레이아웃 컨트롤을 제공한다. 하나의 스타일을 편집하여 전체 사이트의 모양을 변경할 수 있다.
- 접근성이 높아진다. CSS에 의해 표현이 처리되면 콘텐츠를 의미있게 만들 수 있고 웹 뿐 아니라 모바일 기기 등 에서도 시각적으로 마크업 하는 것이 가능해진다

사용방법

selector
요소를 선택해주는 것 - 특정 요소들을 선택해 스타일을 적용하도록 해줍니다.



사용방법

html문서에 적용하는 방법

1. css파일를 따로 만든다
2. html파일 안에 <style>을 사용한다
3. <label style= “”> 사용한다


```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=<device-width>, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <style>
        h1 {
            color: blue;
        }
    </style>
    <title>Document</title>
</head>
```

```
<main>
    <label style="color : blue">ID : </label>
     <input type="text"/>

     <br/>
```

사용방법

class, id

class
같은 이름의 클래스를 여러개 사용할 수 있습니다.

id
같은 이름의 아이디는 사용할 수 없습니다.

```
<body>
  <!-- id 사용 -->
  <div id="header">This is the header</div>

  <!-- class 사용 -->
  <p class="content">This is a paragraph with the "content" class.</p>
  <p class="content">Another paragraph with the same class.</p>
</body>
```

2.CSS적용

텍스트 서식 지정

font-family: 폰트 지정
font-size: 폰트의 사이즈 지정
font-weight: 폰트 굵기 지정
font-style: 폰트 스타일 지정(주로 이탈릭체)
font-variant:변형 스타일 지정(주로 스몰 캡스(small-caps))

```
font-family: Georgia, serif;
```

```
font-size: 12px;
```

```
font-weight: 900;
```

```
font-style: italic;
```

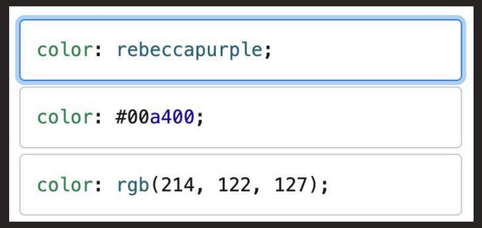
```
font-variant: common-ligatures small-caps;
```

폰트 적용 방법

1. 폰트를 다운받는다.
2. 다운 받은 폰트 폴더를 해당 작업 파일로 이동한다.
3. font-family: 다운 받은 폰트 이름 저장

색상 지정

color: 글자 색 지정
background-color: 배경 색 지정
RGBa로 불투명도를 설정할 수 있습니다.
색 지정은 #FFFFFF white 두가지 둘 다 가능



BOX관련 CSS

-Width : 너비
-Height : 세로, 높이
-Padding : 컨텐츠 영역과 선택적 테두리 사이에 있는 영역.
-Border : padding과 border를 둘러싸는 선. 테두리는 선택사항이다.
-Margin : 테두리 바깥쪽에 추가되는 공간

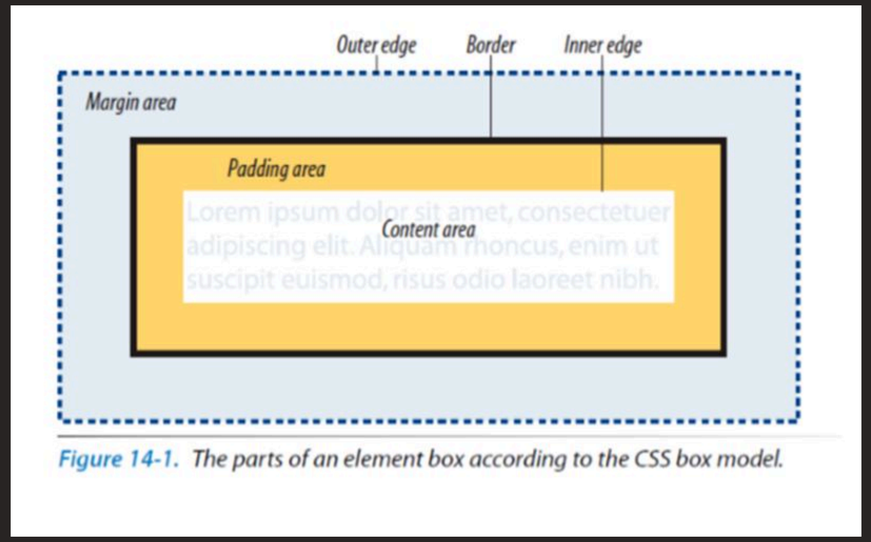
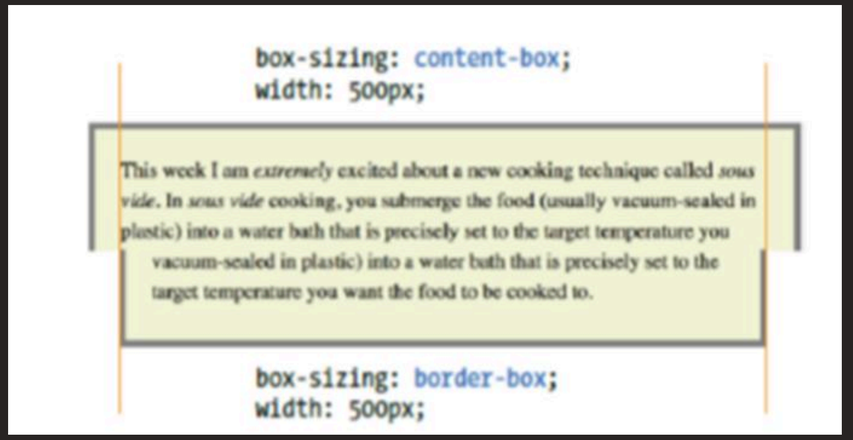


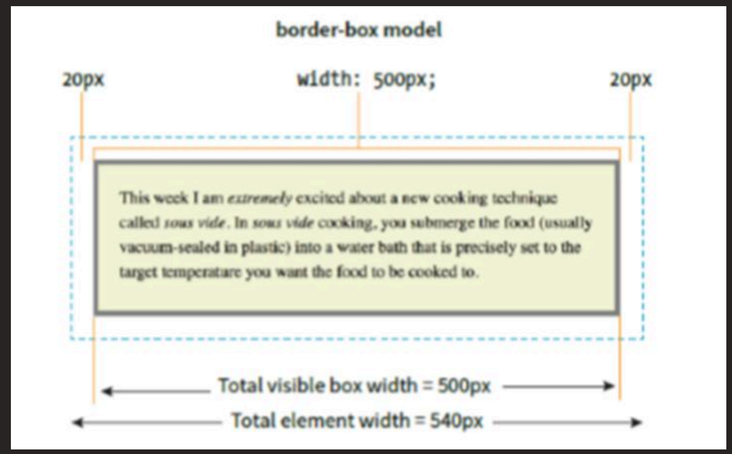
Figure 14-1. The parts of an element box according to the CSS box model.

The border-box model

- box-sizing : border-box;
-Width 그리고 height 치수는 padding 및 border를 포함하여 전체 보이는 상자에 적용된다. Margin까지 포함 x
- box-sizing : content-box;
-Content-box의 경우 주로 content의 크기를 변경한다.




```
p {
  …
  box-sizing: border-box;
  width: 500px;
  height: 150px;
}
```



Border

border는 단순히 내용 영역과 그 주위에 그려진 선
따로 지정된 border style은 따로 없다.
색상이 지정되지 않은 경우 테두리는 요소의 전경색을 사용한다.

border-style
Value : none | solid | hidden | dotted | dashed | double | groove | ridge | inset
Default : none

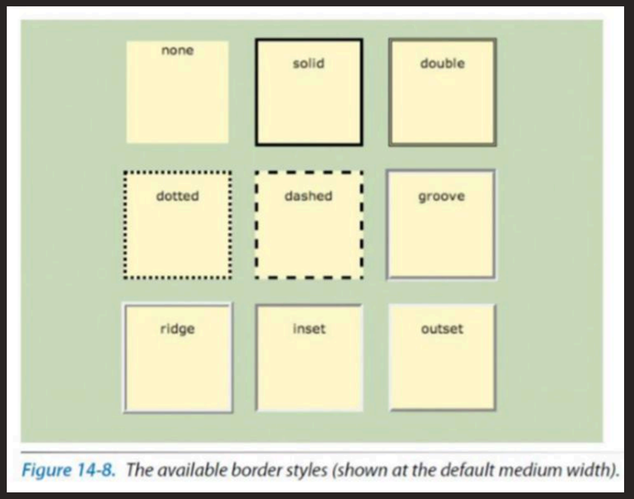


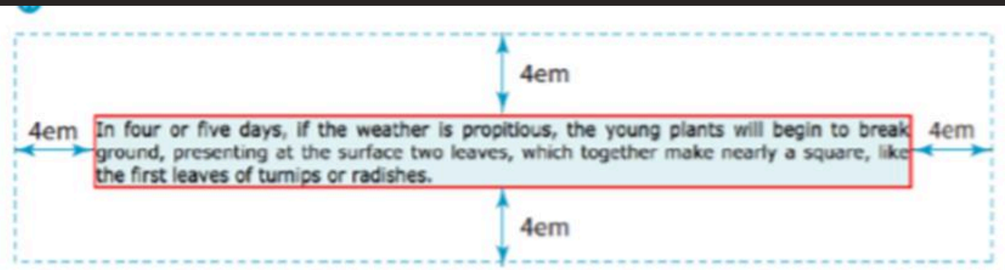
Figure 14-8. The available border styles (shown at the default medium width).

-CSS에서 border-radius 속성을 이용하여 요소에 둥근 모서리를 넣을 수 있다.
-경계 반경에 하나의 값을 제공하면 네 모서리에 모두 적용된다. 왼쪽 위 모서리에서 시작하여 4개의 값이(왼쪽 위, 오른쪽 위, 오른쪽 아래, 왼쪽 아래) 시계 방향으로 적용된다.
-두개의 값을 쓰면 첫 번째 값은 왼쪽 위와 오른쪽 아래에 사용되고 두번째 값은 다른 두 모서리에 사용된다.

Margin

-Margin은 border 바깥쪽에 추가할 수 있는 선택적 공간으로 elements가 서로 부딪치는 것을 방지한다.
-Margin을 사용하여 다른 컨텐츠 열을 위한 공간을 확보 할 수 있다.
-이런 식으로 Margin은 CSS 기반 페이지 레이아웃에서 중요한 도구이다.
-측면 및 짧은 margin 속성은 이미 살펴본 padding 속성과 매우 유사하지만, margin에는 특별한 동작이 있다.
-대부분의 웹의 규격을 측정할 때 상위 요소의 em, pixel, 백분율이 margin을 지정하는 가장 일반적인 방법이다.
-Auto 키워드를 사용하면 브라우저가 사용 가능한 공간을 채우거나 필요한 여백을 자동으로 메워준다.

```
A  p#A {
     margin: 4em;
     border: 2px solid red;
     background: #e2f3f5;
   }
B  p#B {
     margin-top: 2em;
     margin-right: 250px;
     margin-bottom: 1em;
     margin-left: 4em;
     border: 2px solid red;
     background: #e2f3f5;
   }
```



Padding

-패딩은 내용 영역과 테두리 사이의 공간이다.
-내용에 약간의 공간이 생기고 배경 테두리나 가장자리가 텍스트에 부딪히지 않도록 한다.
-요소의 개별측면(블록 수준 또는 인라인)에 패딩을 추가 할 수 있다.
-한 번에 한쪽에 패딩을 설정하는 대신 단축 패딩 속성을 사용하여 요소 주위에 패딩을 추가 할 수 있다.
-순서는 항상 top부터 시계 방향으로 적용되며 이를 고려하여 작성하면 된다.

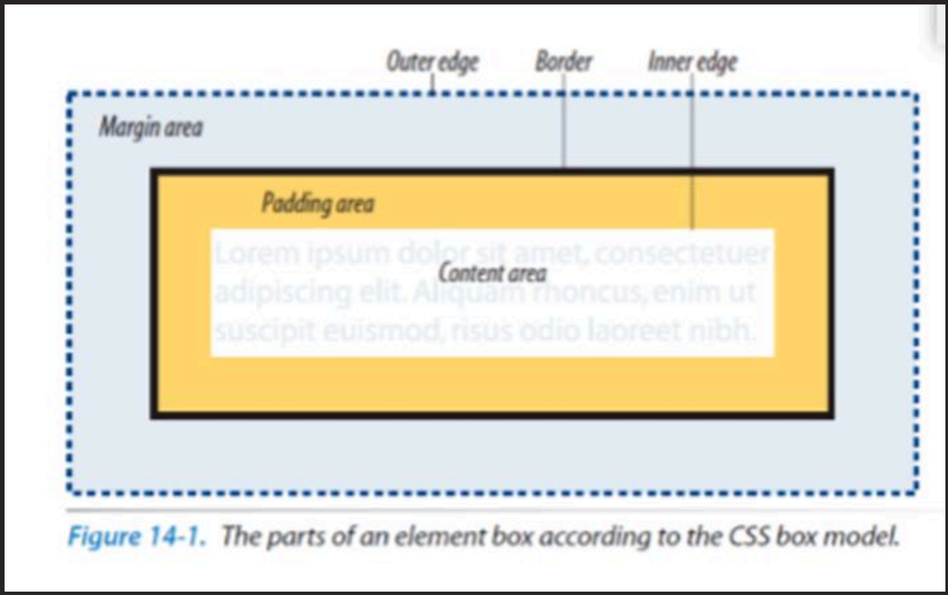


Figure 14-1. The parts of an element box according to the CSS box model.

Position

- Position : 속성은 태그를 어떻게 위치시킬지를 정의하며, 아래의 5가지 값을 갖는다

- Position 속성 값
  - static: 문서 흐름에 따라(기본값)
  - absolute: 가장 가까운 위치 지정된 조상 요소를 기준으로 배치
  - relative: 원래 위치를 기준으로 상대적으로 위치 조정 가능
  - fixed: 브라우저 창을 기준으로 고정되어 스크롤해도 위치 유지
  - sticky: 스크롤 위치에 따라 relative와 fixed 사이를 전환하는 혼합형 배치
- 보통 left, right, top, bottom, z-index 속성과 함께 사용함.

```
em {
  position: relative;
  top: 30px;
  left: 60px;
  background-color: fuchsia;
}
```

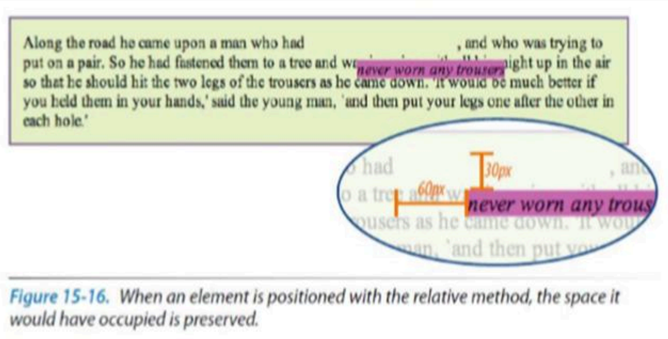


Figure 15-16. When an element is positioned with the relative method, the space it would have occupied is preserved.

```
em {
  position: absolute;
  top: 30px;
  left: 60px;
  background-color: fuchsia;
}
```

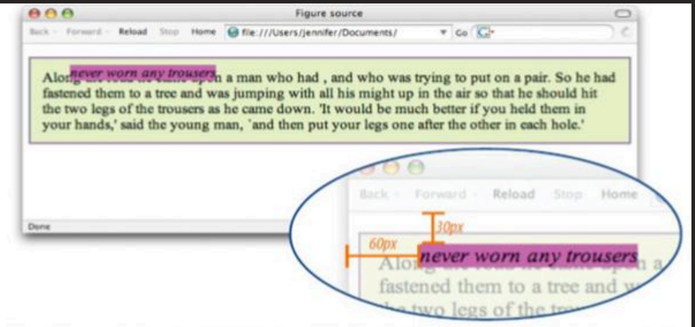


Figure 15-17. When an element is absolutely positioned, it is removed from the flow and the space is closed up.

3.Flex,Grid

Flex

flex

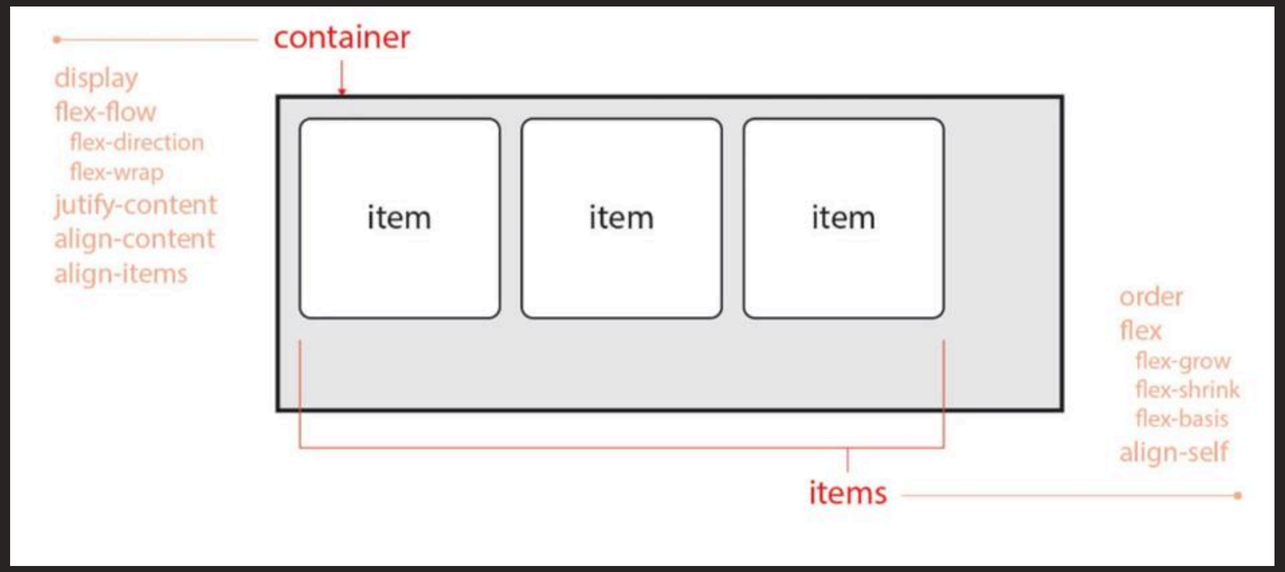
-CSS Flex는 요소의 크기가 불분명하거나 동적인 경우에도 각 요소를 정렬할 수 있는 효율적인 방법을 제공한다.
-컨테이너 내부의 항목이 늘어나거나 줄어들어 공간 낭비와 Overflow를 방지한다.
-Flex 는 2개의 개념으로 나뉜다.
-Container display, flex-flow, justify-content 등의 속성을 사용할 수 있다.
-Item에는 order, flex, align-self 등의 속성을 사용할 수 있다.

html

```
<ul class = “parent”>
  <li class = “child”></li>
  <li class = “child”></li>
  <li class = “child”></li>
</ul>
```

CSS

```
.parent{
  display : flex;
}
```



display

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

flex 와 inline-flex 의 차이

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

flex-flow

-단축 속성으로 Flex items 의 주 축(main-axis)과 교차 축(cross-axis)을 설정하고 Item의 여러 줄 묶음(줄 바꿈)도 설정한다.

row	Itmes를 수평축(왼쪽에서 오른쪽으로)으로 표시	row
row-reverse		
column	Items를 수직축(위에서 아래로)으로 표시	
column-reverse	Items를 column의 반대 축으로 표시	

flex-direction

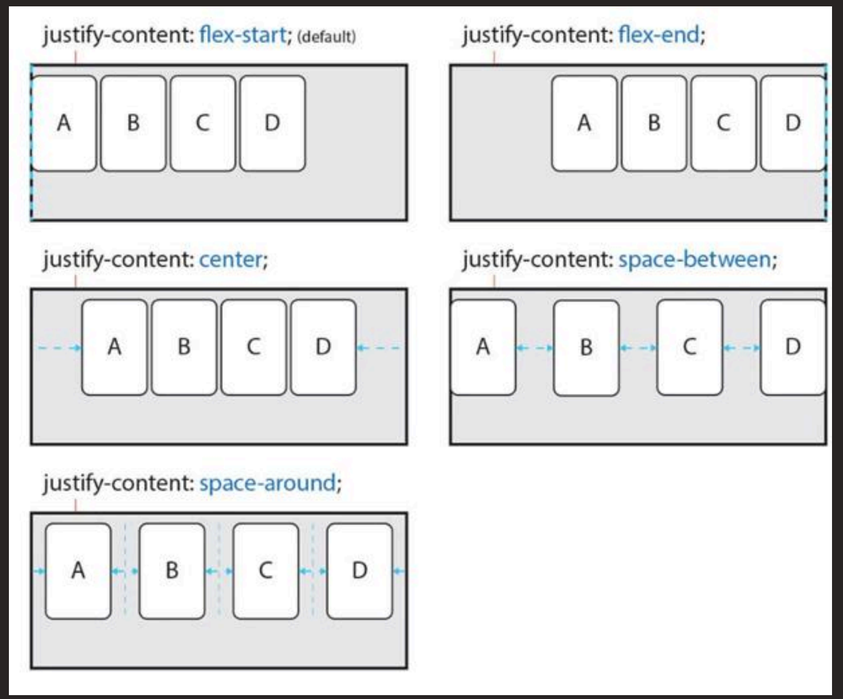
nowrap	모든 Itmes를 여러 줄로 묶지 않음(한 줄에 표시)	nowrap
wrap	Items를 여러 줄로 묶음	
wrap-reverse	Items를 wrap의 역 방향으로 여러 줄로 묶음	

flex-wrap

Justify-content

-주 축(main-axis)의 정렬 방법을 설정한다..

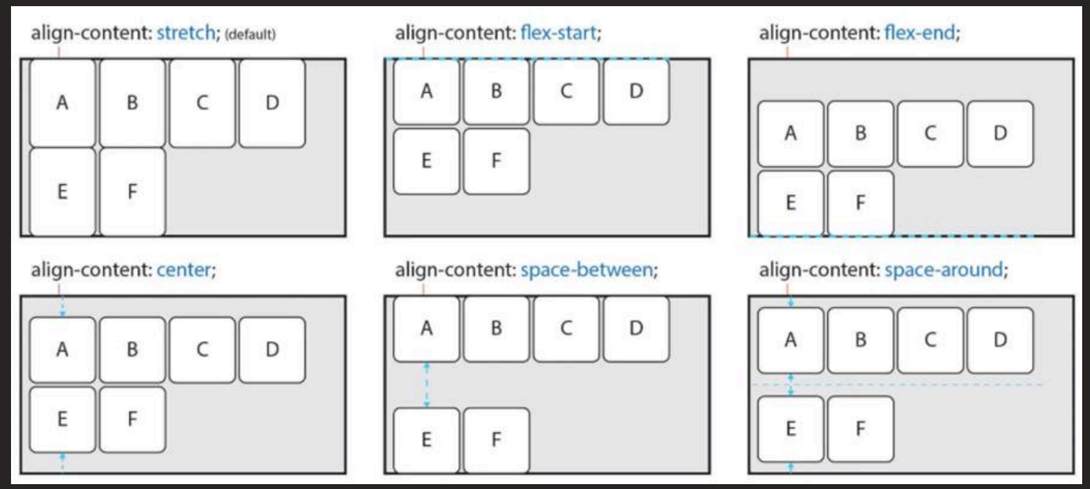
flex-start	Items를 시작점(flex-start)으로 정렬	flex-start
flex-end	Items를 끝점(flex-end)으로 정렬	
center	Items를 가운데 정렬	
space-between	시작 Item은 시작점에, 마지막 Item은 끝점에 정렬되고 나머지 Items는 사이에 고르게 정렬됨	
space-around	Items를 균등한 여백을 포함하여 정렬	



align-content

-교차축(cross-axis)의 정렬 방법을 설정한다.
-주의할 점은 align-content 속성은 Items가 여러 줄(2줄 이상)이고 여백이 있을 경우에만 사용할 수 있다.
Items가 한 줄일 경우 align-item 속성을 사용하면 된다..

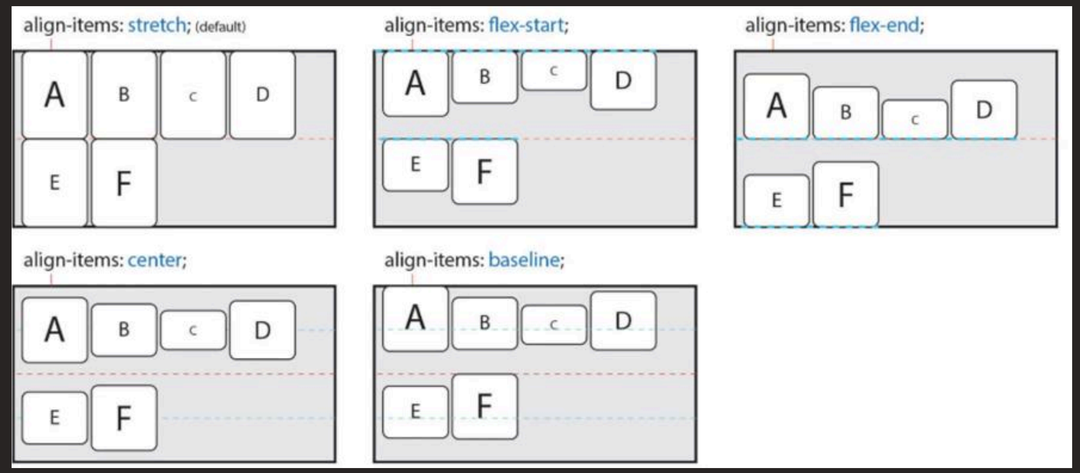
stretch	Container의 교차 축을 채우기 위해 Items를 늘림
flex-start	Items를 시작점(flex-start)으로 정렬
flex-end	Items를 끝점(flex-end)으로 정렬
center	Items를 가운데 정렬
space-between	시작 Item은 시작점에, 마지막 Item은 끝점에 정렬되고 나머지 Items는 사이에 고르게 정렬됨
space-around	Items를 균등한 여백을 포함하여 정렬



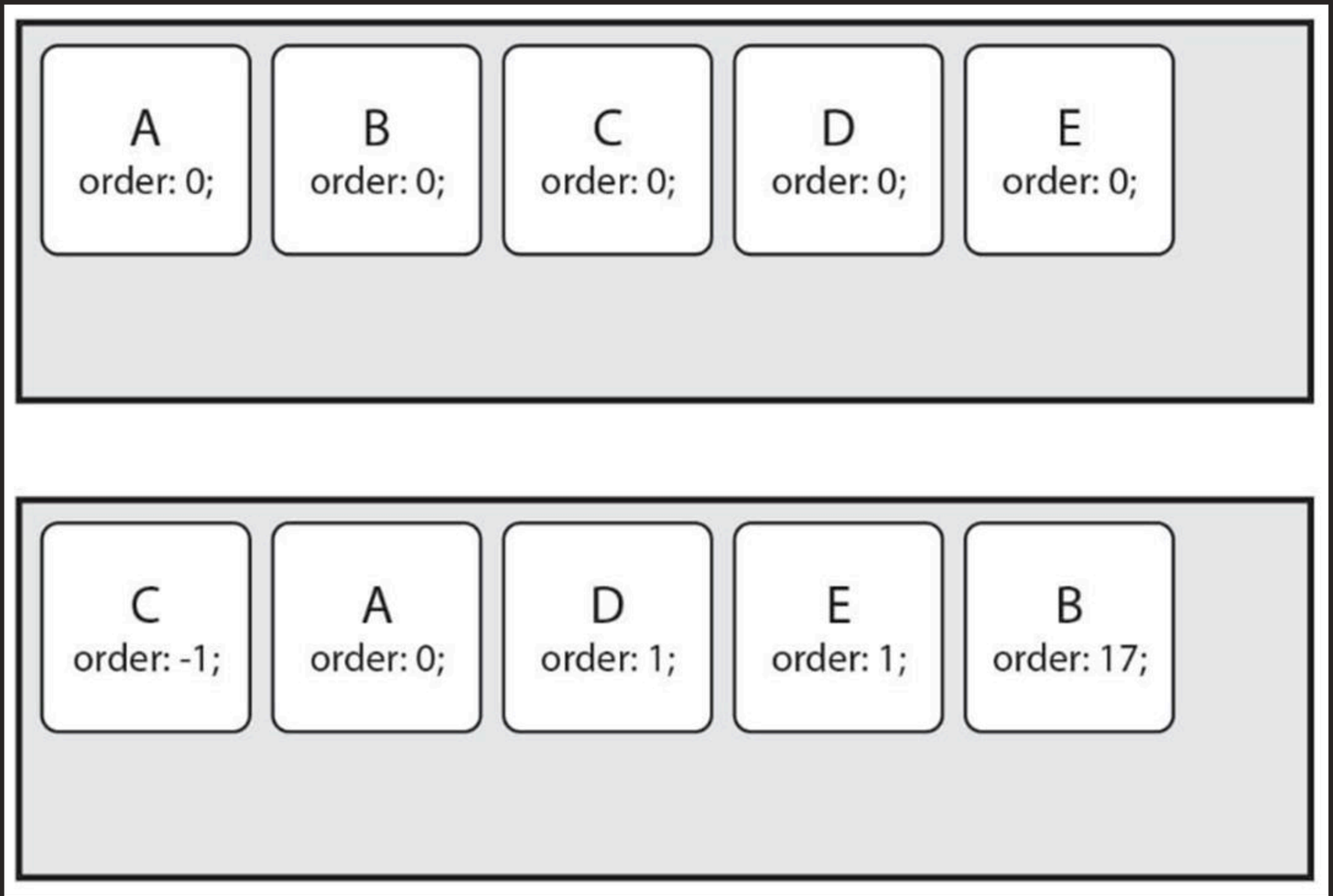
align-items

-교차축(cross-axis)의 정렬 방법을 설정한다.
-Items가 한 줄일 경우 많이 사용한다.
-주의할 점은 Items가 flex-wrap을 통해 여러 줄(2줄 이상)일 경우에는 align-content 속성이 우선한다.
-따라서 align-items를 사용하려면 align-content 속성을 기본값(stretch)으로 설정해야 한다.

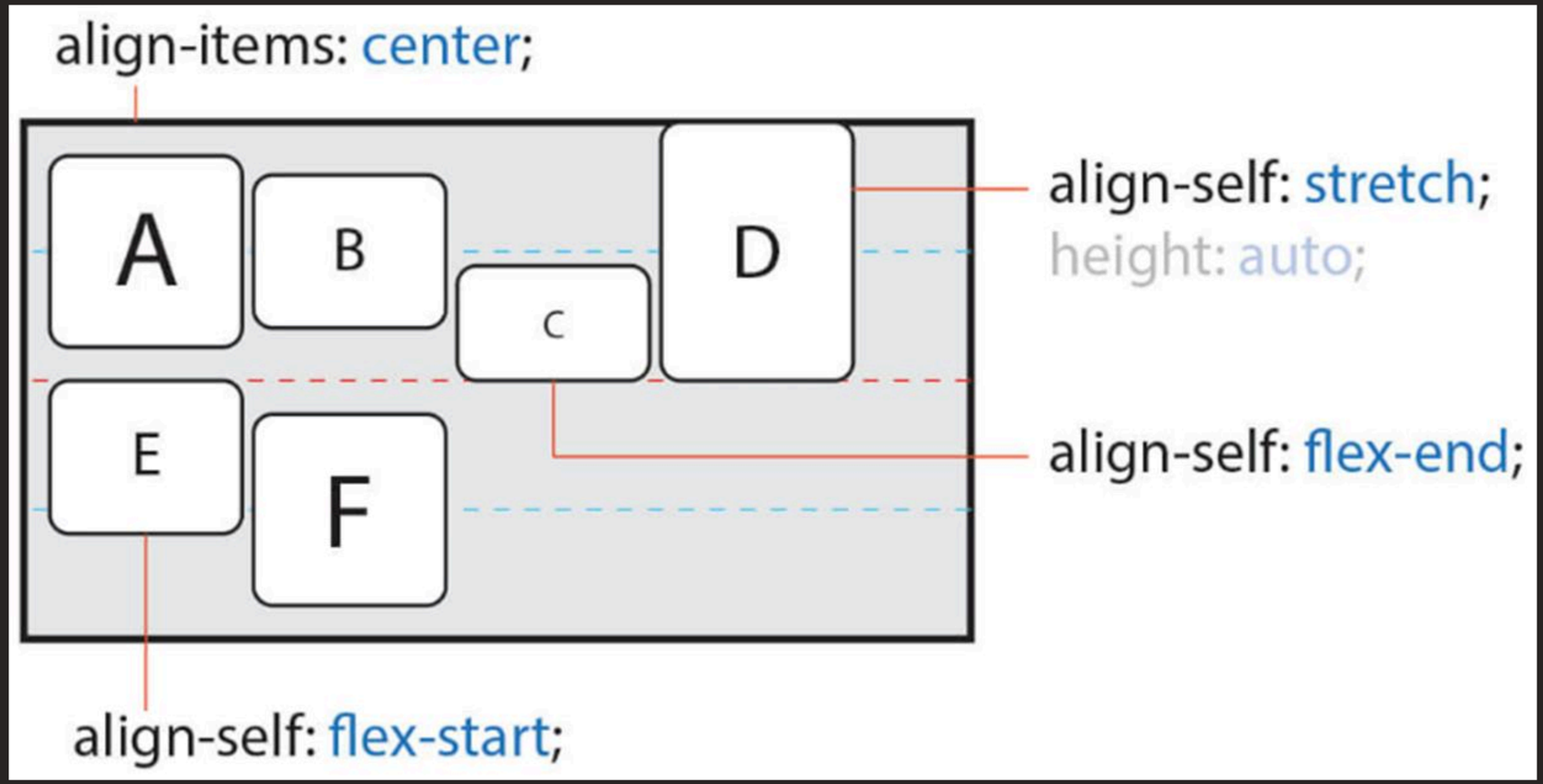
stretch	Container의 교차 축을 채우기 위해 Items를 늘림
flex-start	Items를 각 줄의 시작점(flex-start)으로 정렬
flex-end	Items를 각 줄의 끝점(flex-end)으로 정렬
center	Items를 가운데 정렬
baseline	Items를 문자 기준선에 정렬



items-order


A
order: 0;
B
order: 0;
C
order: 0;
D
order: 0;
E
order: 0;
C
order: -1;
A
order: 0;
D
order: 1;
E
order: 1;
B
order: 17;

Items – align-self



Grid

Grid란?

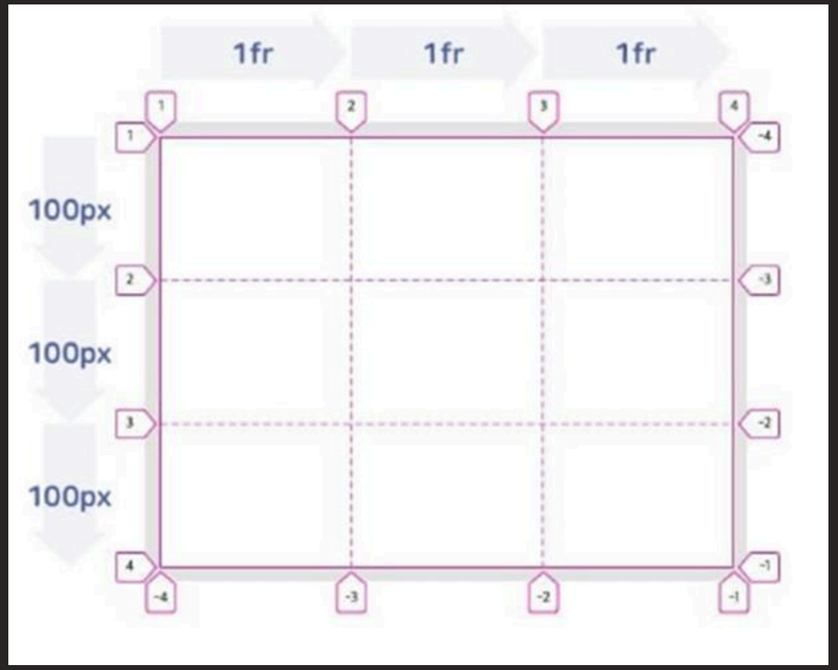
행과 열로 레이아웃을 구성 하는 것

값	의미
grid	Block 특성의 Grid Container를 정의
inline-grid	Inline 특성의 Grid Container를 정의

Container

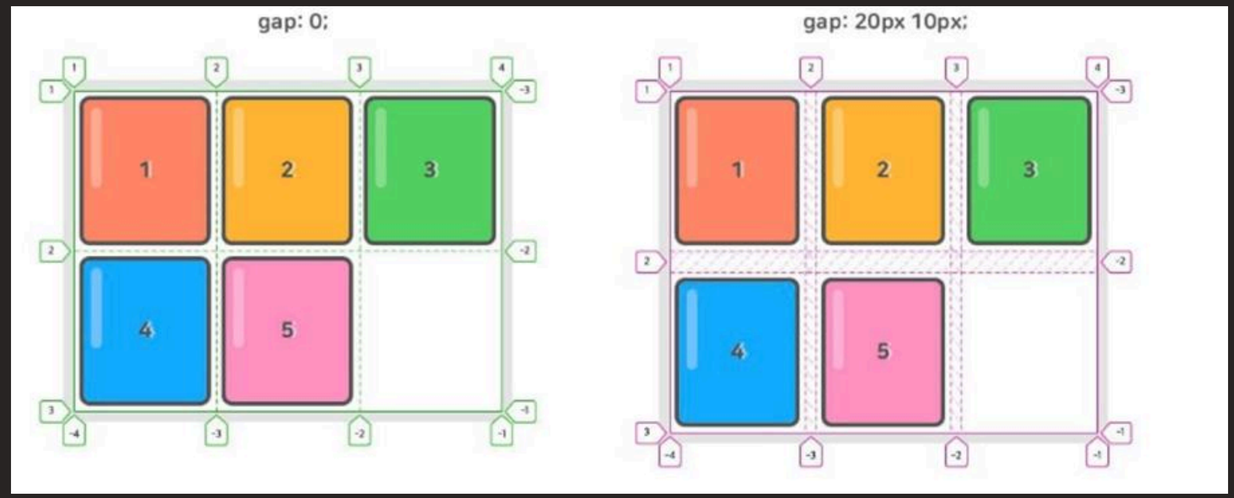
```
.container {
  display: grid;
  grid-template-rows: 1행크기 2행크기 ...;
  grid-template-rows: [선이름] 1행크기 [선이름] 2행크기 [선이름] ...;
}
```

```
.container {
  width: 400px;
  display: grid;
  grid-template-rows: repeat(3, 100px);
  grid-template-columns: repeat(3, 1fr);
}
```

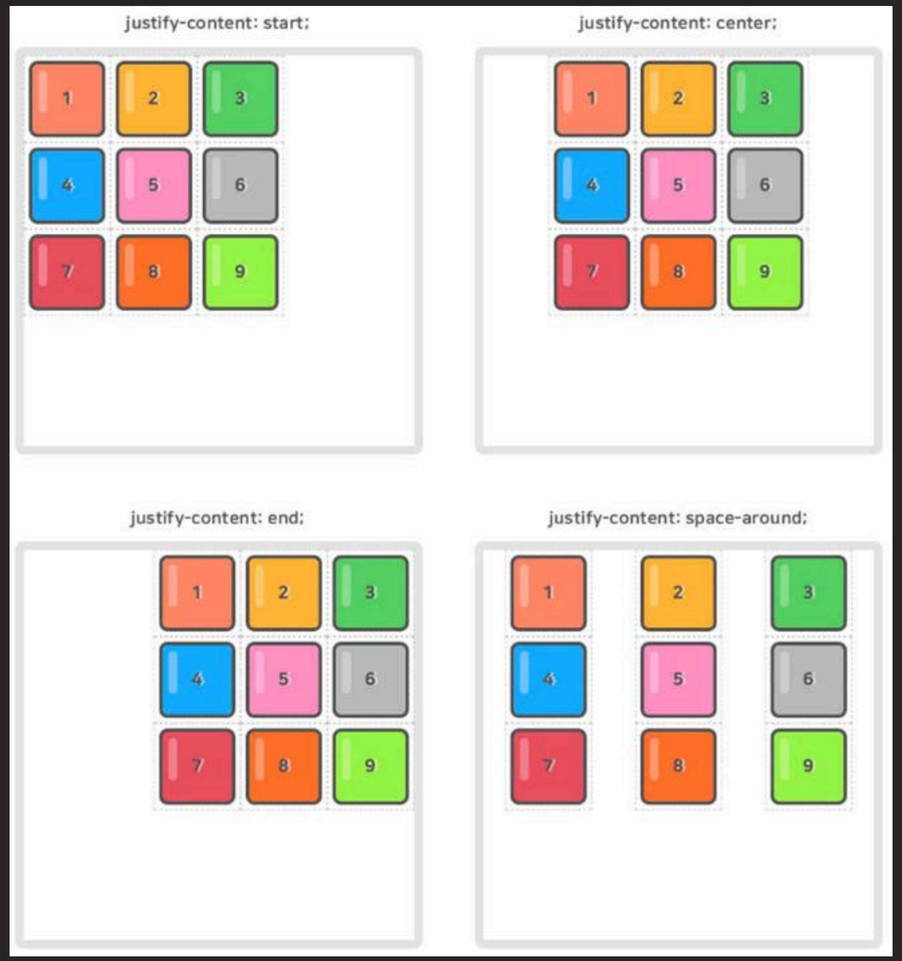


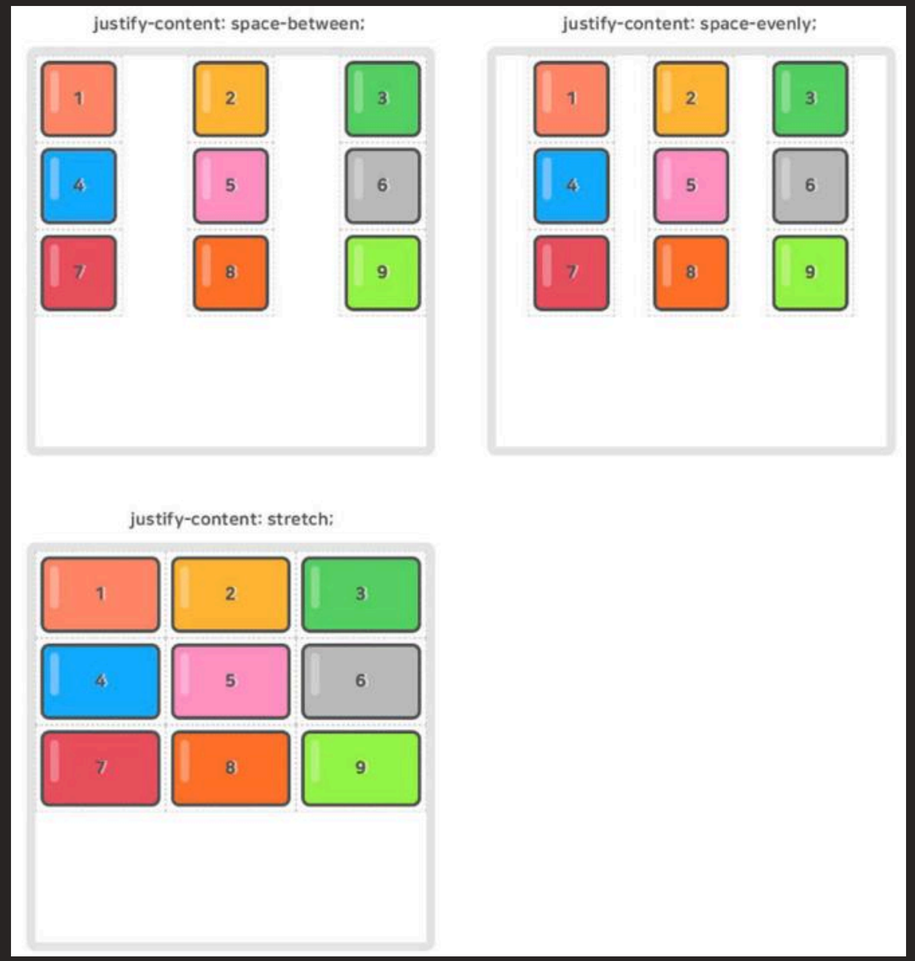
Container

```css
.container {
  display: grid;
  grid-template-rows: repeat(2, 150px);
  grid-template-columns: repeat(3, 1fr);
  gap: 20px 10px;
}
/* 하나의 값으로 통일할 수 있습니다. */
.container {
  gap: 10px;  /* row-gap: 10px; + column-gap: 10px; */
}
/* 하나의 값만 적용하고자 한다면 다음과 같이 사용할 수 있습니다. */
.container {
  gap: 10px 0; /* row-gap */
  gap: 0 10px; /* column-gap */
}
```

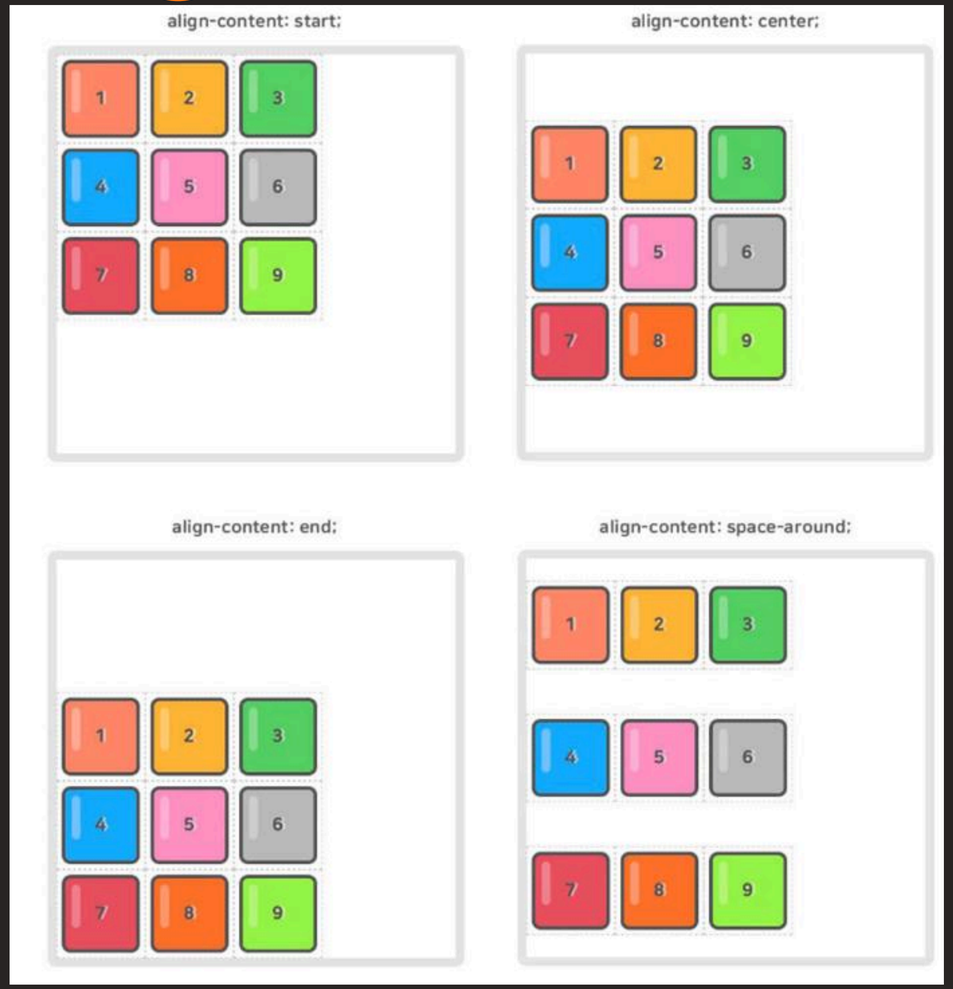


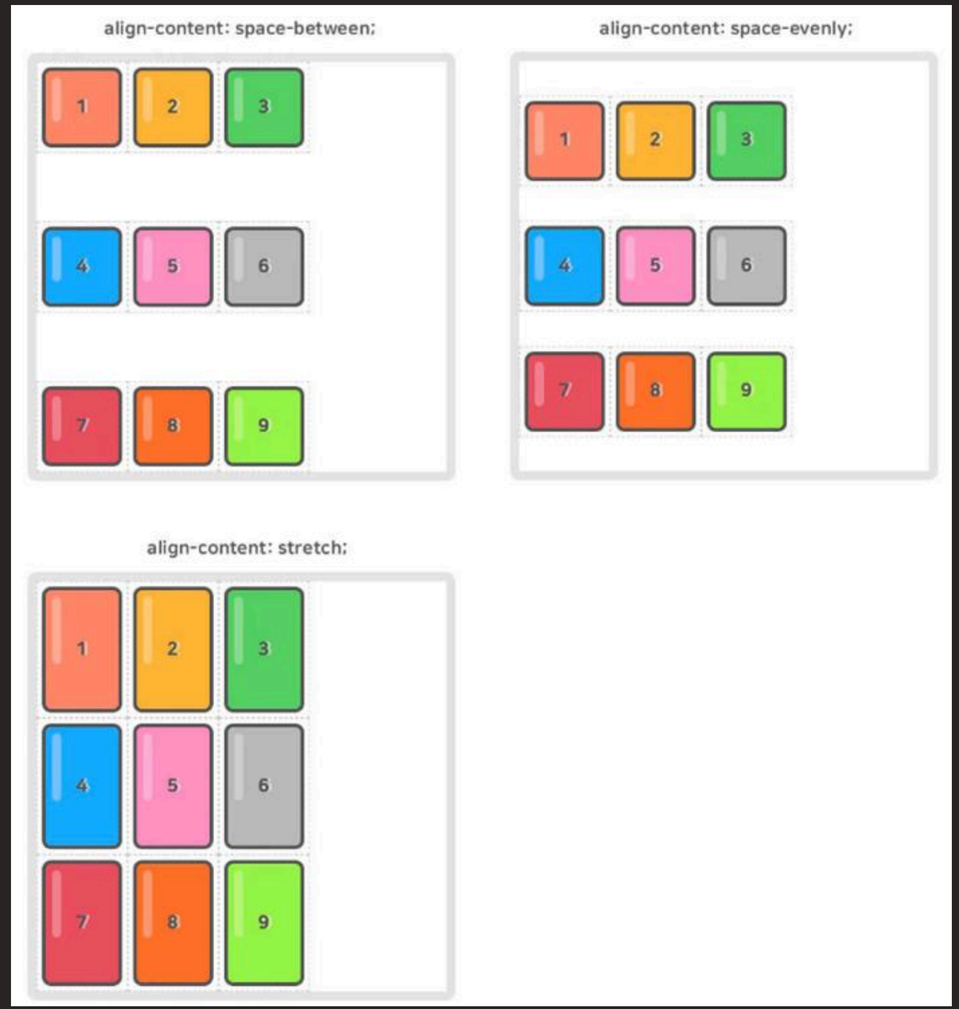
Justify-content



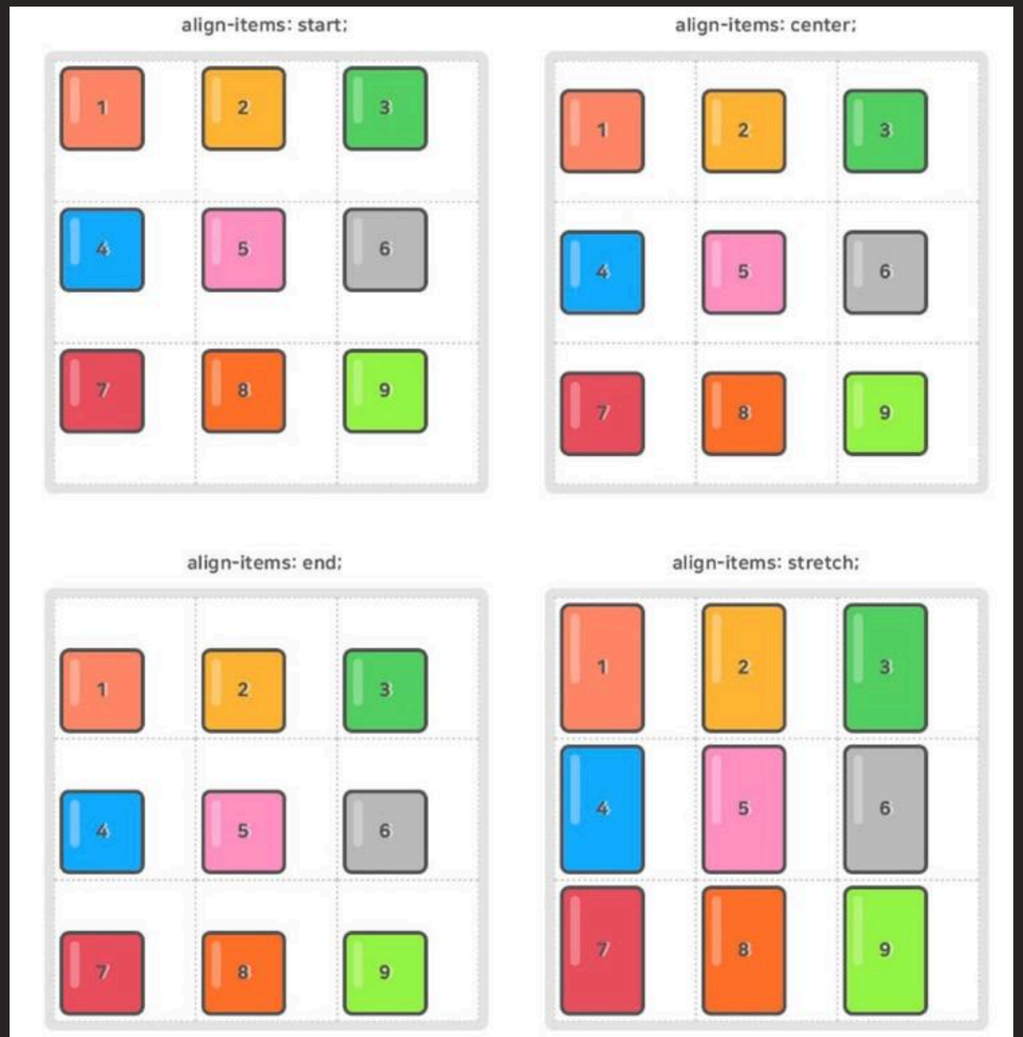


align-content

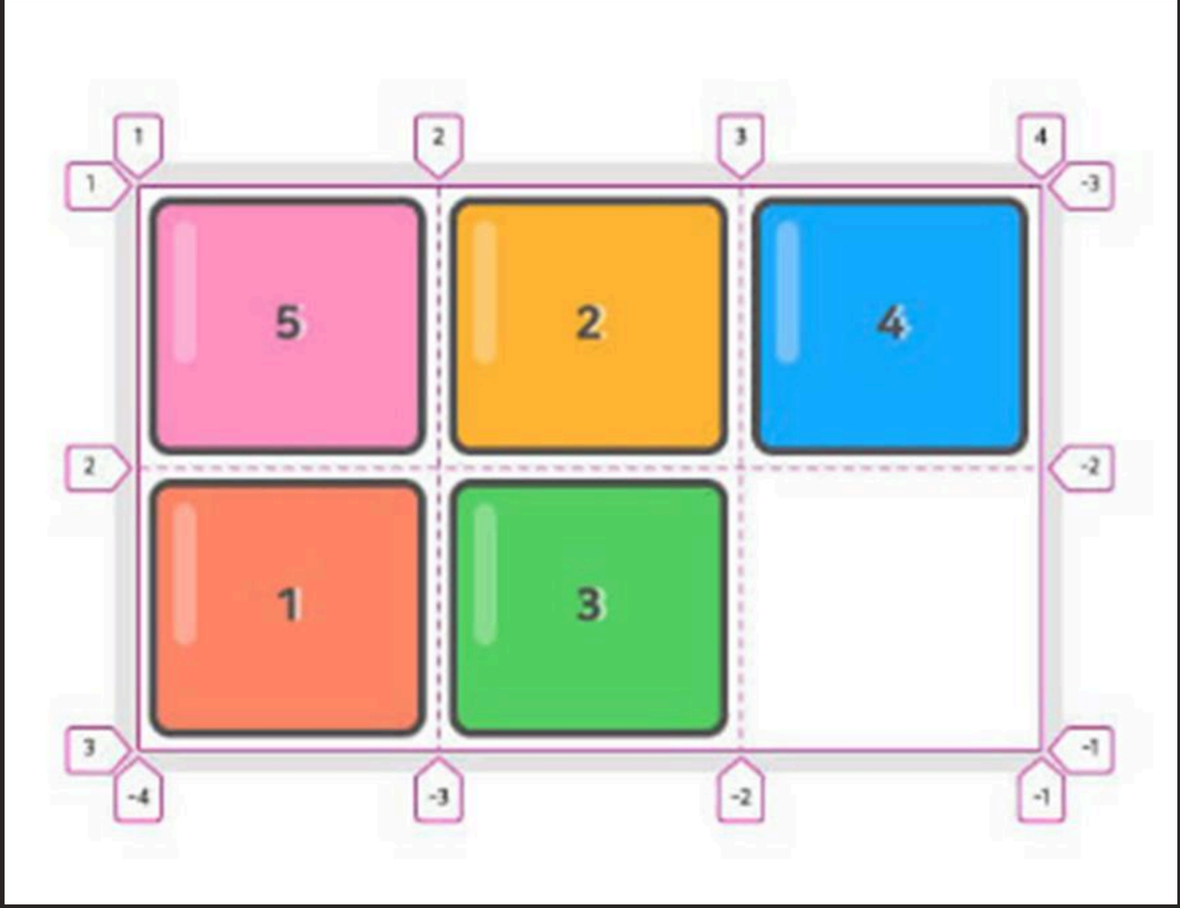




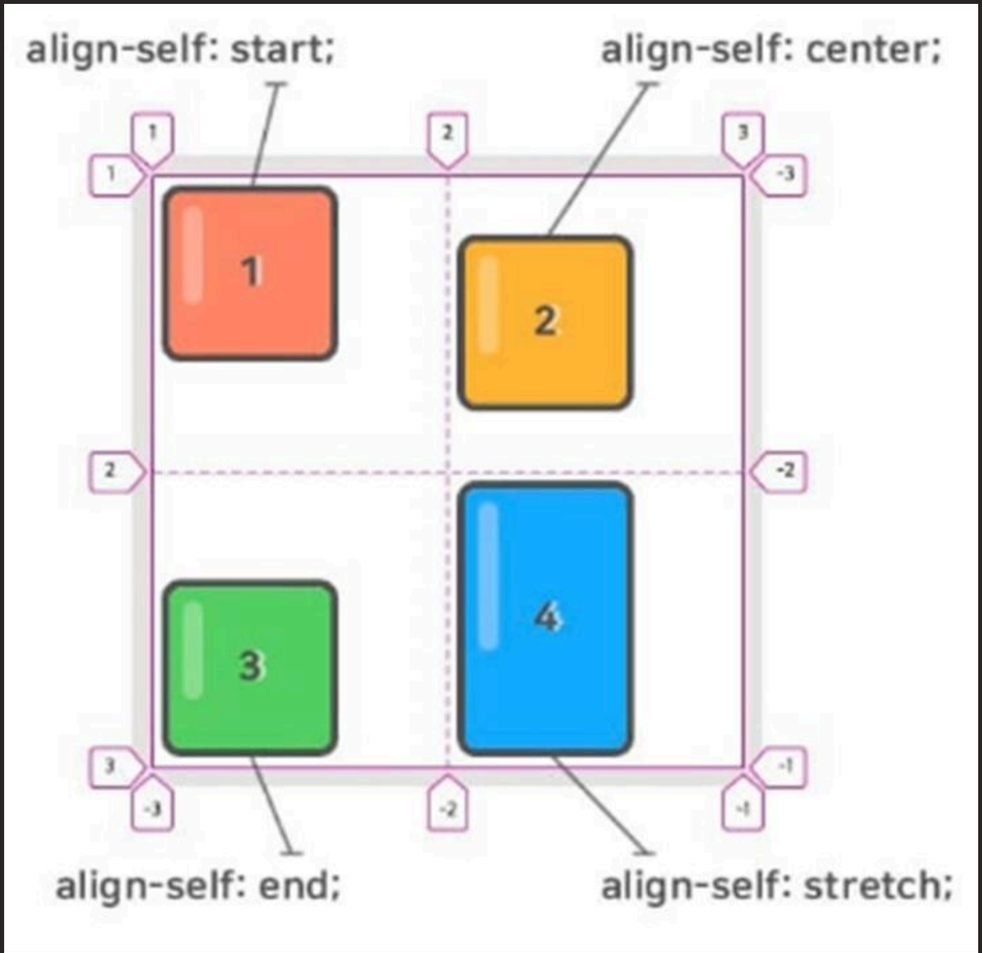
align-items


align-items: start;
1
2
3
4
5
6
7
8
9
align-items: center;
1
2
3
4
5
6
7
8
9
align-items: end;
1
2
3
4
5
6
7
8
9
align-items: stretch;
1
2
3
4
5
6
7
8
9

items-order



Items – align-self



피그마 실습

수고하셨습니다 :)