

CSS

CSS 값 과 단위

절대 길이 단위

다음은 모두 **절대** 길이 단위이며 — 다른 것과 관련이 없으며 일반적으로 항상 동일한 크기로 간주됩니다.

단위	이름	다음과 동일함
<code>cm</code>	센티미터	1cm = 96px/2.54
<code>mm</code>	밀리미터	1mm = 1/10th of 1cm
<code>Q</code>	4분의 1 밀리미터	1Q = 1/40th of 1cm
<code>in</code>	인치	1in = 2.54cm = 96px
<code>pc</code>	Picas	1pc = 1/6th of 1in
<code>pt</code>	포인트	1pt = 1/72th of 1in
<code>px</code>	픽셀	1px = 1/96th of 1in

참고:

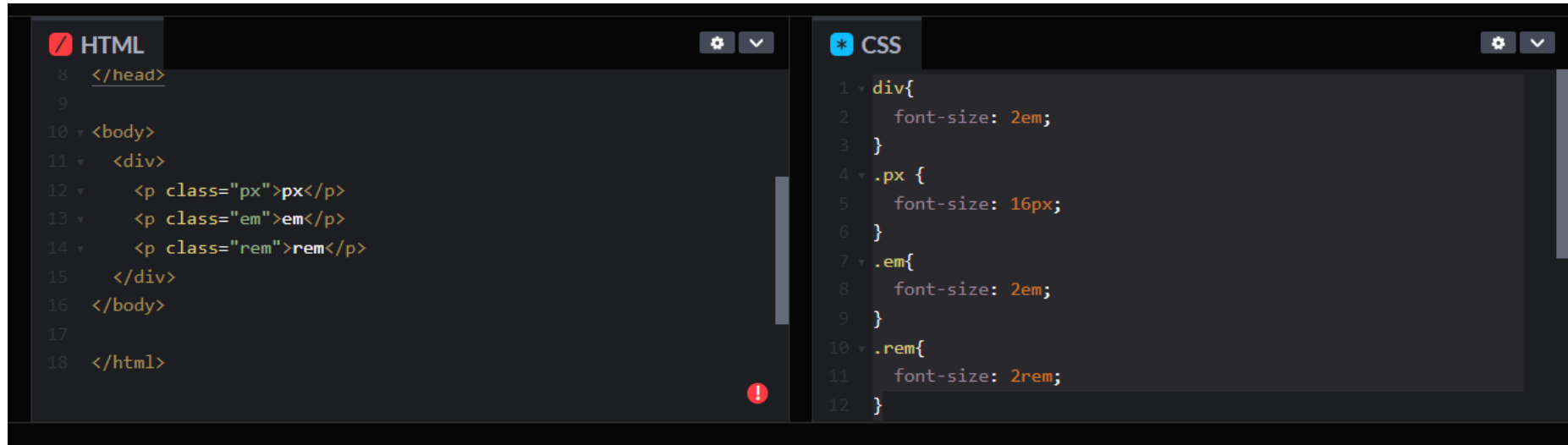
https://developer.mozilla.org/ko/docs/Learn/CSS/Building_blocks/Values_and_units

상대 길이 단위

상대 길이 단위는 다른 요소 (상위 요소의 글꼴 크기 또는 viewport 크기) 와 관련이 있습니다. 상대 단위를 사용하면 텍스트나 다른 요소의 크기가 페이지의 다른 모든 것에 비례하여 조정되도록 신중하게 계획할 수 있다는 이점이 있습니다. 웹 개발에 가장 유용한 단위가 아래 표에 나열되어 있습니다.

단위	관련 사항
<code>em</code>	요소의 글꼴 크기.
<code>ex</code>	요소 글꼴의 x-height.
<code>ch</code>	요소 글꼴의 glyph "0" 의 사전 길이 (너비) 입니다.
<code>rem</code>	루트 요소의 글꼴 크기.
<code>lh</code>	요소의 라인 높이.
<code>vw</code>	viewport 너비의 1%.
<code>vh</code>	viewport 높이의 1%.
<code>vmin</code>	viewport 의 작은 치수의 1%.
<code>vmax</code>	viewport 의 큰 치수의 1%.

CSS 값 과 단위



```
HTML
8 </head>
9
10 <body>
11 <div>
12 <p class="px">px</p>
13 <p class="em">em</p>
14 <p class="rem">rem</p>
15 </div>
16 </body>
17
18 </html>

CSS
1 div{
2   font-size: 2em;
3 }
4 .px {
5   font-size: 16px;
6 }
7 .em{
8   font-size: 2em;
9 }
10 .rem{
11   font-size: 2rem;
12 }
```

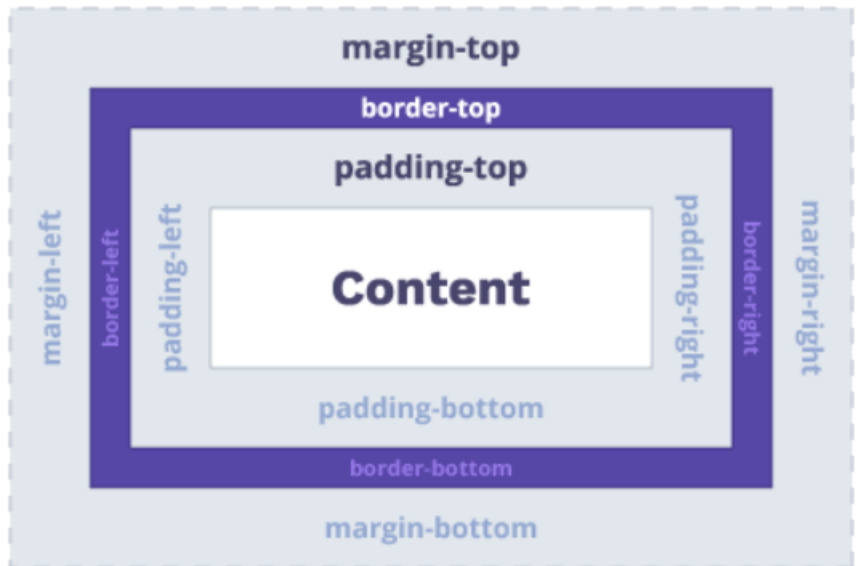
px

em

rem

- px은 고정크기여서 다른 것과 상관없이 나타남
- em은 상대크기로 부모요소가 기준
- rem은 최상위(root 즉 html) 기준의 상대크기

Box_sizing



- content-box는 기본 CSS 박스 크기 결정법을 사용합니다. 요소의 너비를 100 픽셀로 설정하면 콘텐츠 영역이 100 픽셀 너비를 가지고, 테두리와 안쪽 여백은 이에 더해집니다.
- border-box는 테두리와 안쪽 여백의 크기도 요소의 크기로 고려합니다. 너비를 100 픽셀로 설정하고 테두리와 안쪽 여백을 추가하면, 콘텐츠 영역이 줄어들어 총 너비 100 픽셀을 유지합니다. 대부분의 경우 이 편이 크기를 조절할 때 쉽습니다.

Border size 테두리까지 포함하려면

box-sizing: border-box;

Box_sizing

기본

box-sizing: content-box;

```
HTML
1 <div id="small">Hello</div>
2 <div id="large">Hello</div>

CSS
1 /*      *{
2     box-sizing: border-box;
3     } */
4 div{
5     margin: 10px;
6     width: 150px;
7 }
8 #small{
9     border: 10px solid red;
10 }
11 #large{
12     border: 30px solid red;
13 }
```

Hello

Hello

box-sizing: border-box;

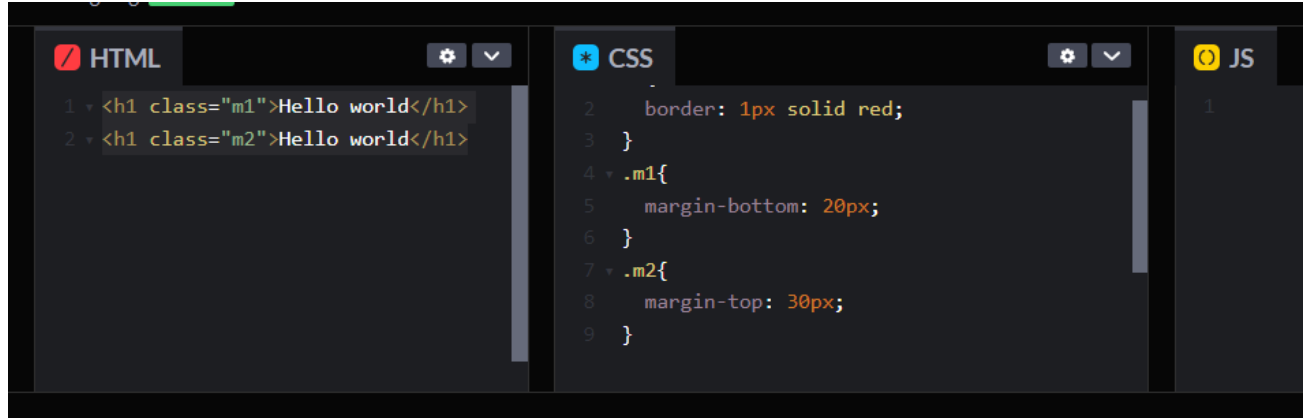
```
HTML
1 <div id="small">Hello</div>
2 <div id="large">Hello</div>

CSS
1 *{
2     box-sizing: border-box;
3 }
4 div{
5     margin: 10px;
6     width: 150px;
7 }
8 #small{
9     border: 10px solid red;
10 }
11 #large{
12     border: 30px solid red;
13 }
```

Hello

Hello

Margin 겹침(1)



```
HTML
1 <h1 class="m1">Hello world</h1>
2 <h1 class="m2">Hello world</h1>

CSS
2 border: 1px solid red;
3 }
4 .m1{
5   margin-bottom: 20px;
6 }
7 .m2{
8   margin-top: 30px;
9 }
```

마진이 겹치는 부분은
값이 큰 마진으로 대체됨

Hello world

Hello world

마진이 50px 아니라
큰 값인 30px로 나타남

Margin 겹침(2)



```
HTML
1 <div id="parent">
2   <div id="child">
3     abc
4   </div>
5 </div>

CSS
1 #parent{
2   margin-top: 100px;
3 }
4 #child{
5   background-color: powderblue;
6   margin-top: 50px;
7 }
```

← 마진인 큰 #parent의
100px이 적용됨

abc



```
HTML
1 <div id="parent">
2   <div id="child">
3     abc
4   </div>
5 </div>

CSS
1 #parent{
2   border: 1px solid red;
3   margin-top: 100px;
4 }
5 #child{
6   background-color: powderblue;
7   margin-top: 50px;
8 }
```

#parent 에 border가 있으면
margin은 각각 적용됨

100px

50px

abc

레이아웃

position

CSS position 속성은 문서 상에 요소를 배치하는 방법을 지정합니다.

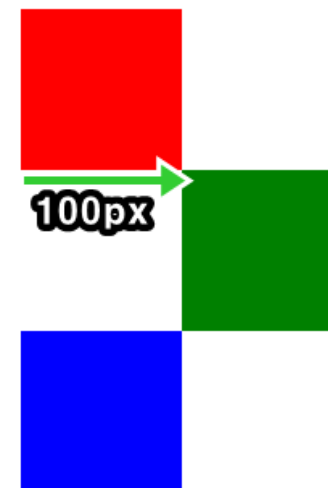
static	요소를 문서 흐름에 맞추어 배치.
relative	이전 요소(주로 부모 요소)에 자연스럽게 연결하여 위치를 지정.
absolute	원하는 위치를 지정해 배치.
fixed	지정한 위치에 고정하여 배치.
sticky	위치에 따라서 동작방식이 달라짐. 요소가 임계점(scroll 박스 기준) 이전에는 relative와 같이 동작. 그 이후에는 fixed와 같이 동작.

Position -relative

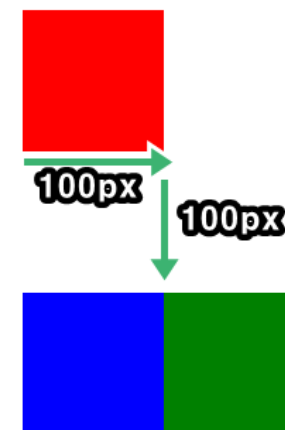
```
<style>
  div{
    width: 100px;
    height: 100px;
  }
  .red{
    background-color: red;
  }
  .green{
    background-color: green;
  }
  .blue{
    background-color: blue;
  }
</style>
```



```
<style>
  div{
    width: 100px;
    height: 100px;
  }
  .red{
    background-color: red;
  }
  .green{
    background-color: green;
    position: relative;
    left: 100px;
  }
  .blue{
    background-color: blue;
  }
</style>
```



```
.green{
  background-color: green;
  position: relative;
  left: 100px;
  top: 100px;
}
.blue{
  background-color: blue;
}
</style>
```



Position -absolute (1)

```
<style>
  div{
    width: 100px;
    height: 100px;
  }
  .red{
    background-color: red;
  }
  .green{
    background-color: green;
  }
  .blue{
    background-color: blue;
  }
</style>
</head>
<body>
  <div class="box red"></div>
  <div class="box green"></div>
  <div class="box blue"></div>
</body>
```



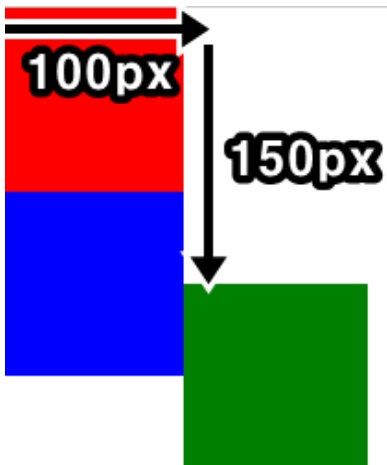
```
body{
  margin: 0;
}
.green{
  background-color: green;
  position: absolute;
  left: 100px;
}
.blue{
  background-color: blue;
}
```



- 좌표속성 중 한쪽 축만 사용하면 다른 축은 원래 본인위치 임
 - > x 좌표 즉 left, right 만 사용 -> y 좌표 세로 축은 원래 위치임
 - > y 좌표 즉 top, bottom만 사용하면 -> x, 가로 축은 원래 위치임
- 이동 후 원래 있던 공간은 빈공간으로 이식되어 파란색이 올라옴

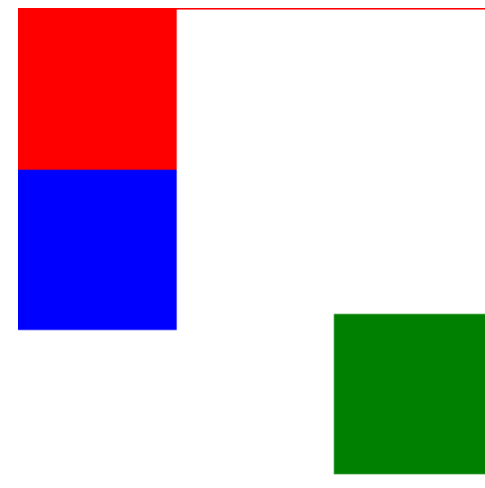
Position - absolute (2)

```
body{
  margin: 0;
}
.green{
  background-color: green;
  position: absolute;
  left: 100px;
  top: 150px;
}
.blue{
  background-color: blue;
}
```



```
.above{
  border: 1px solid red;
  width: 300px;
  height: 300px;
  position: relative;
}
.green{
  background-color: green;
  position: absolute;
  right: 2px;
  bottom: 10px;
}
.blue{
  background-color: blue;
}
```

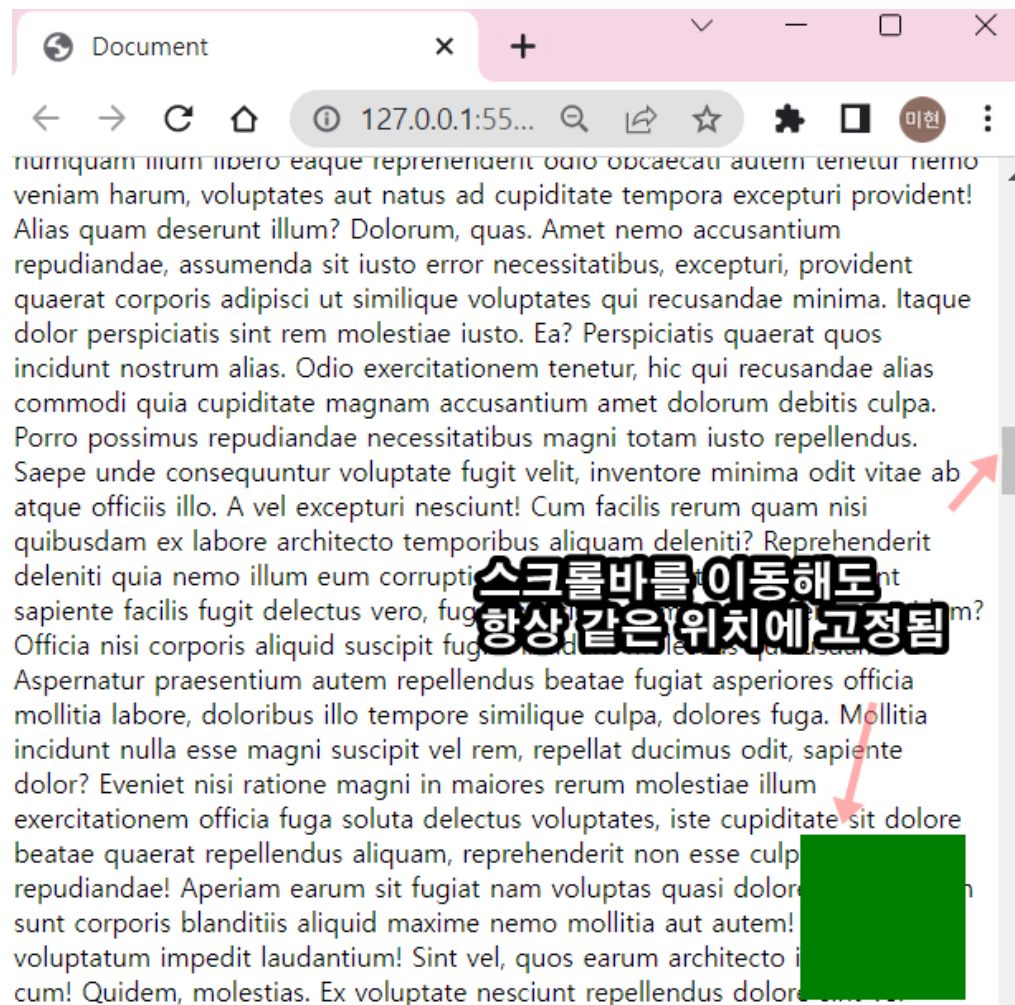
```
</style>
</head>
<body>
  <div class="above">
    <div class="box red"></div>
    <div class="box green"></div>
    <div class="box blue"></div>
  </div>
```



Absolute 속성에서 좌표 기준을 바꿀 수 있음
부모 혹은 조상 요소 중에서 Position: relative를 지정함

Position – fixed

```
<style>
  div{
    background-color: green;
    width: 100px;
    height: 100px;
    position: fixed;
    right: 20px;
    bottom: 20px;
  }
</style>
```



Position – z-index

```
<style>
  div{
    width: 100px;
    height: 100px;
  }
  .red{
    background-color: red;
    position: relative;
  }
  .green{
    background-color: green;
    position: absolute;
    top: 50px;
  }
</style>
</head>
<body>
  <div class="box red"></div>
  <div class="box green"></div>
```



```
}
.red{
  background-color: red;
  position: relative;
  z-index: 10;
}
.green{
  background-color: green;
  position: absolute;
  top: 50px;
```



float

이미지 요소와 텍스트 요소 배치



Lorem, ipsum dolor sit amet consectetur adipisicing elit. Dolor autem maxime, hic nihil obcaecati cumque? Eum, adipisci consete harum voluptatem dolorem debitis facilis quo at, mollitia nostrum ex ullam iure autem voluptas. Dolor aspernatur doloremque sin debitis, esse sequi accusantium non quos est aspernatur ut molestiae totam ab sint deserunt, tenetur doloremque! Sunt, minus. Q libero eum repudiandae commodi adipisci, quo culpa ipsum quis, tenetur necessitatibus quae sequi voluptate? Vitae, deserunt sin aperiam voluptatibus laboriosam velit porro tenetur unde reiciendis, sint nemo sit placeat voluptatem praesentium similique. Quis consectetur minima voluptate repudiandae, vero quam ducimus? Harum sequi ex voluptatem mollitia, impedit eveniet amet delen Saepe voluptatem dolore eum laudantium nihil amet recusandae perspiciatis veniam deleniti ullam modi esse sunt officia ea offici nisi? Tempora sequi aliquam, adipisci numquam ipsum inventore pariatur ratione harum ipsa! Mollitia, porro unde voluptas aliquid



```
<title>Document</title>
<style>
  .float{
    float: left;
    margin-right: 10px;
  }
</style>
</head>
<body>
  <div class="float"><img src="https://www.artinsight
  Lorem, ipsum dolor sit amet consectetur adipisicing
  adipisci consectetur voluptatem harum error dolorem
```



Lorem, ipsum dolor sit amet conseq
Rem, veniam laborum consequatur
autem voluptas. Dolor aspernatur di
minus debitis, esse sequi accusantiu
corrupti quas. Debitis animi ratione
necessitatibus quae sequi voluptate
veniam. Quidem aperiam voluptatib
perspiciatis explicabo. Consequatur,
ducimus? Harum sequi ex voluptate
Saepe voluptatem dolore eum laud
Debitis, sint beatae! Optio dolor sus
harum ipsa! Mollitia, porro unde vol
libero praesentium ratione quos itac
numquam, modi iusto rerum eos mi
doloremque. Officia sequi quasi dol
numquam ratione voluptatum volup
veniam quia quisquam, impedit cor
temporibus officiis quisquam? Iure c
sit animi laborum veniam consequu
similique, doloremque repellendus e

quibusdam aut distinctio quas vel, rem officia, amet, architecto error asperiores voluptatem ipsa dolore debitis? Odit neque pra
assumenda tempora earum provident perspiciatis voluptates architecto voluptate, quis ipsum quaerat iure, voluptas minus. Verc
vel aut perferendis, nisi impedit incidunt mollitia fugit praesentium nihil eveniet? Quam, ducimus. Recusandae laudantium sapie
porro inventore consequatur quam ut temporibus repellat? Numquam quos laboriosam cupiditate sit perferendis officiis. Tenetu
cumque dolore eligendi quis enim labore voluptatibus, quibusdam quos minus mollitia commodi dignissimos. Porro neque a pr
harum deleniti quidem? Quod nemo necessitatibus aspernatur, ea a culpa. Quos neque, iure repellat fugit culpa ea sint unde at
quod blanditiis? Nam, enim. Modi, sed fuga saepe dignissimos quod pariatur. Rem, consectetur? Praesentium illum maiores qua
autem. Facere fuaiat tempora maxime suscipit modi quia saepe provident tenetur cumque? Suscipit consequuntur laboriosam c

clear

Float 속성을 해제할 때 사용

← → ↺ 🏠 ⓘ 127.0.0.1:5500/float.html



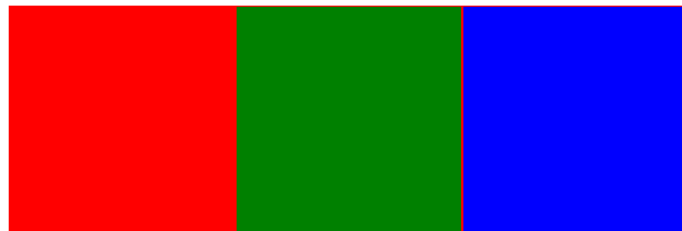
Lorem, ipsum dolor
Rem, veniam laboru
autem voluptas. Do
minus debitis, esse
corrupti quas. Debit
necessitatibus quae
veniam. Quidem ap
perspiciatis explicab
ducimus? Harum se

Saepe voluptatem dolore eum laudantium nihil amet recusandae perspiciatis veniam deleniti ullam modi esse s
Tempora sequi aliquam, adipisci numquam ipsum inventore pariatu ratione harum ipsa! Mollitia, porro unde vc
ratione quos itaque inventore harum debitis quam odit, facere at rerum corrupti sapiente eius dolores tempore
repudiandae quae! Sapiente molestias sunt cupiditate eum doloremque. Officia sequi quasi doloribus fugiat rep
voluptatibus eveniet, natus quidem doloribus alias, in non modi excepturi dolorem tenetur ut minima. Quasi au
eaeque deserunt libero voluptatum dolore temporibus officiis quisquam? Lure obcaecati illum error fuga, quidem
nobis sed nisi! Necessitatibus, sunt. Repellat consectetur, a laborum ipsa quo omnis! Ratione facere harum neq
Deleniti eveniet minus adipisci quibusdam aut distinctio quas vel, rem officia, amet, architecto error asperiores
similique alias omnis animi error qui assumenda tempora earum provident perspiciatis voluntates architecto ve

```
margin-right: 10px;
}
.clear{
  clear: both;
}
</style>
</head>
<body>
<div class="float">
  Saepe voluptatem dolore eum lauda
  sunt officia ea officiis mollitia
출력 디버그 콘솔 터미널
0" - 오후 11:12:35] Extension Name: esbenp.
0" - 오후 11:12:35] Extension Version: 9.10
```


Float -레이아웃

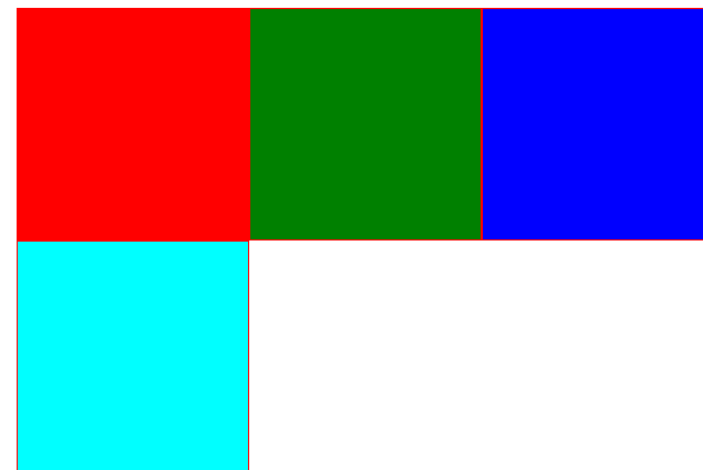
```
<style>
div{
  border: 1px solid red;
  width: 200px;
  height: 200px;
}
.red{
  background-color: red;
  float: left;
}
.green{
  background-color: green;
  float: left;
}
.blue{
  background-color: blue;
  float: left;
}
.cyan{
  background-color: cyan;
}
</style>
</head>
<body>
<div class="red"></div>
<div class="green"></div>
<div class="blue"></div>
<div class="cyan"></div>
```



```
float: left;
}
.cyan{
  background-color: cyan;
  clear: both;
}
```

Cyan요소가 red요소 밑에 있어 안보임

⇒ Cyan 요소에 clear:both 를 css에 추가함



flex

플렉서블 박스 레이아웃은 1차원 방식 레이아웃 설계임

display:flex; CSS속성이 적용된 div.container가 부모 요소인 flex container가 되고, 하위 요소인 div.item은 자식 요소인 flex item이라 부릅니다.
쉽게 말해서 flex container, flex item 부모 요소와 자식 요소는 ul과 li의 관계처럼 떼려야 뗄 수 없는 관계라고 생각하시면 이해하기 쉽습니다.

Flex 부모 요소와 자식 요소 CSS 속성 구분

위에서도 언급했지만 flex 속성은 크게 **컨테이너 속성**과 **아이템 속성**으로 두 가지로 나누어집니다.

✓ **flex container 부모 속성** : flex-direction, flex-wrap, justify-content, align-items, align-content

✓ **flex item 자식 속성** : flex, flex-grow, flex-shrink, flex-basis, order

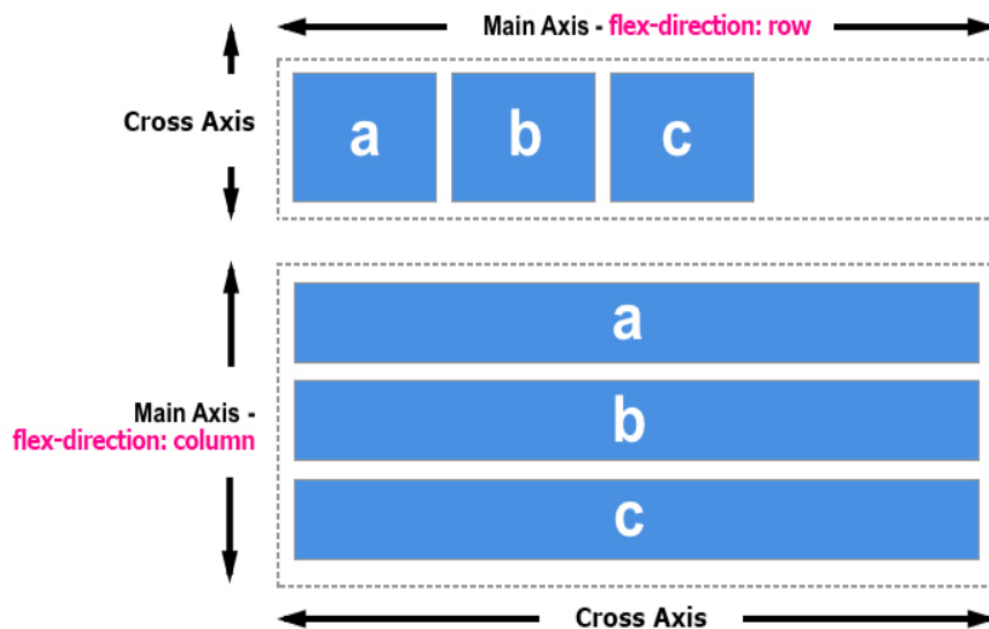
✓ **row(기본값)** : Item이 행(수평축: 왼쪽에서 오른쪽) 방향으로 배치됩니다.

✓ **row-reverse** : Item이 **row** 반대축 방향으로, 요소들이 역순으로 배치됩니다.

✓ **column** : Item이 열(수직축: 위에서 아래) 방향으로 배치됩니다.

✓ **column-reverse** : Item이 **column** 반대축 방향으로, 요소들이 역순으로 배치됩니다.

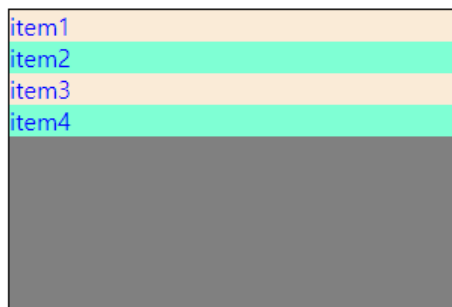
메인축(Main Axis)과 교차축(Cross Axis) 구분 방법



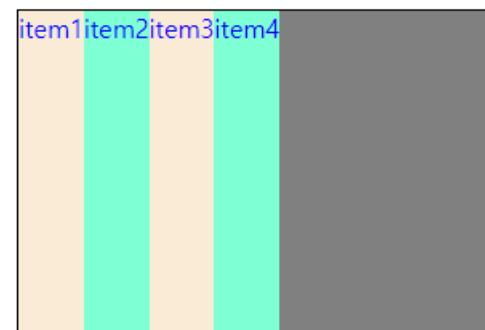
flex

display:flex

```
<style>
  .flex-container{
    width: 300px;
    height: 200px;
    background-color: grey;
    border: 1px solid black;
  }
  .flex-item{
    color: blue;
    background-color: antiquewhite;
  }
  .flex-item:nth-child(2n){
    background-color: aquamarine;
  }
</style>
</head>
<body>
  <div class="flex-container">
    <div class="flex-item">item1</div>
    <div class="flex-item">item2</div>
    <div class="flex-item">item3</div>
    <div class="flex-item">item4</div>
  </div>
</body>
</html>
```



```
.flex-container{
  width: 300px;
  height: 200px;
  background-color: grey;
  border: 1px solid black;
  display: flex;
}
```



Display 속성이 지정된 요소는 플렉스 컨테이너가 되고,
자식 요소는 플렉스 아이템이 됨

Flex : flex-direction 속성

flex-direction

flex item 요소들의 배치되는 **메인축의 방향을 결정**하는 속성입니다.

item의 배치된 방향의 축을 **메인축(main axis)**, 메인축과 수직인 축을 **교차축(Cross Axis)** 이라고 합니다.

메인축의 방향은 부모 요소인 container의 flex-direction 속성으로 결정합니다.

flex-direction 속성을 따로 지정하지 않으면 기본값인 row가 적용됩니다.

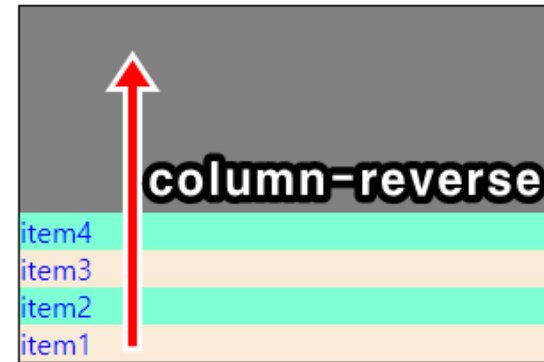
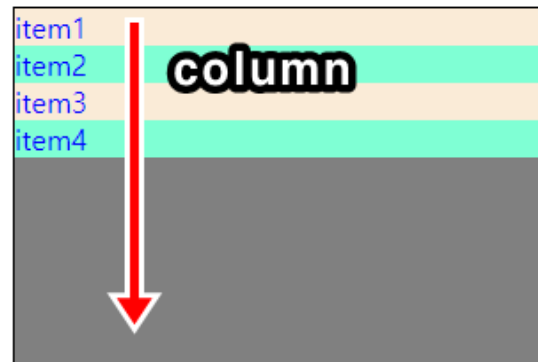
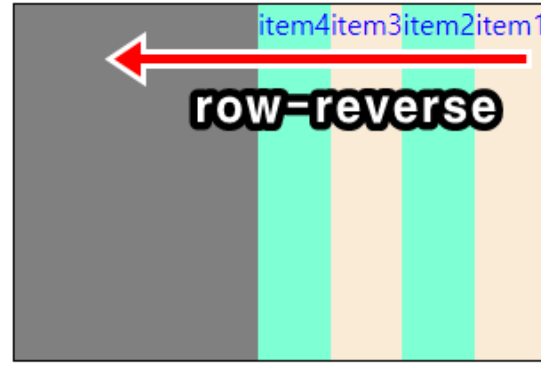
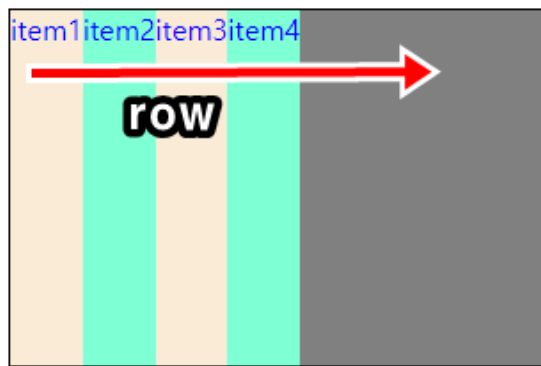
```
.flex-container{
  width: 300px;
  height: 200px;
  background-color: grey;
  border: 1px solid black;
  display: flex;
  flex-direction: row; /* row-reverse, column, column-reverse */
}
```

✓ **row(기본값)** : Item이 행(수평축: 왼쪽에서 오른쪽) 방향으로 배치됩니다.

✓ **row-reverse** : Item이 **row** 반대축 방향으로, 요소들이 역순으로 배치됩니다.

✓ **column** : Item이 열(수직축: 위에서 아래) 방향으로 배치됩니다.

✓ **column-reverse** : Item이 **column** 반대축 방향으로, 요소들이 역순으로 배치됩니다.



flex: flex-wrap 속성

flex-wrap

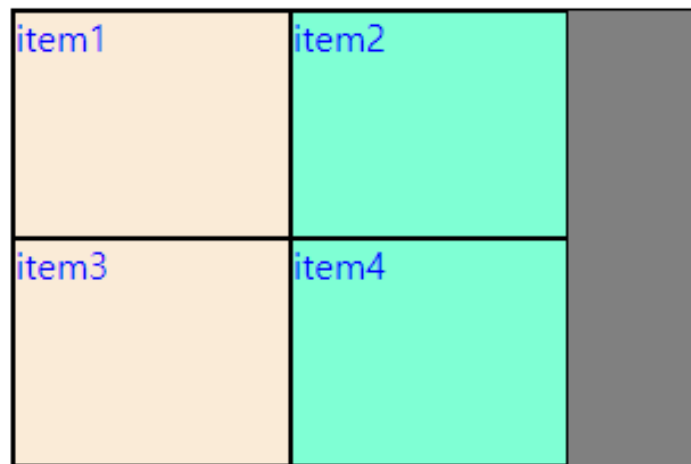
flex item이 flex container 영역을 넘어갈 경우 줄 바꿈을 할지를 결정하는 속성입니다.

flex-wrap 속성을 따로 지정하지 않으면 기본값인 nowrap이 적용됩니다.

✓ nowrap(기본값) : Item의 전체 값이 container의 영역 값을 넘치더라도 줄바꿈하지 않고 Item은 한줄로 배치됩니다.

✓ wrap : Item이 container의 영역값을 넘치면 줄 바꿈 되어 Item이 배치됩니다.

✓ wrap-reverse : Item이 wrap의 역순으로 배치됩니다.



```
.flex-container{
  width: 300px;
  height: 200px;
  background-color: grey;
  border: 1px solid black;
  display: flex;
  flex-wrap: wrap;
}
.flex-item{
  color: blue;
  background-color: antiquewhite;
  width: 120px;
  border: 1px solid black;
}
.flex-item:nth-child(2n){
  background-color: aquamarine;
}
```

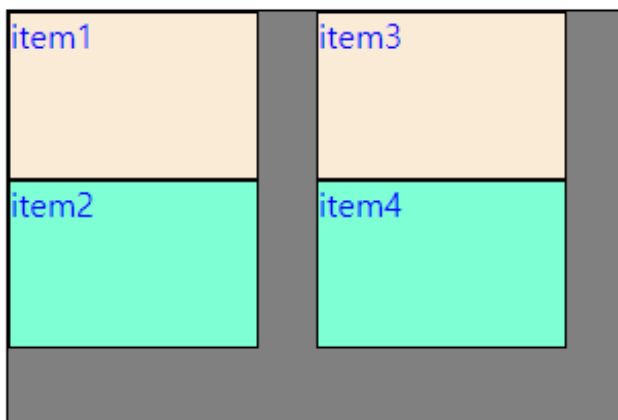
Flex: flex-flow속성

flex-flow

flex-direction과 flex-wrap을 한 번에 같이 지정할 수 있는 **축약 속성**으로 단축해서 사용할 수입니다.

flex-direction, flex-wrap의 순으로 한 칸 떼고 써주면 됩니다.

```
.flex-container{
  width: 300px;
  height: 200px;
  background-color: grey;
  border: 1px solid black;
  display: flex;
  flex-flow: column wrap;
}
.flex-item{
  color: blue;
  background-color: antiquewhite;
  width: 120px;
  height: 80px;
  border: 1px solid black;
}
```



flex-direction: column;

flex-wrap: wrap;



flex-flow: column wrap;

Flex: justify-content

justify-content

Item의 **메인축을 정렬**하는 속성입니다.

메인축은 flex-direction에서 설정합니다.

justify-content 속성을 따로 지정하지 않으면 기본값인 flex-start가 적용됩니다.

- ✓ **flex-start(기본값)** : 메인축의 시작 지점을 기준으로 Item을 정렬합니다.
- ✓ **flex-end** : 메인축의 마지막 지점을 기준으로 Item을 정렬합니다.
- ✓ **center** : 메인축의 Item을 가운데 정렬합니다.
- ✓ **space-between** : 메인축의 첫 번째 Item은 시작점에 마지막 Item은 끝 지점 정렬하고, 나머지 item은 사이에 동일한 간격으로 정렬합니다.
- ✓ **space-around** : 메인축을 Item둘레에 동일한 간격으로 정렬합니다.

```
display: flex;  
  
justify-content: flex-start;
```

Flex: justify-content

☒ flex-start: ☐ flex-end: ☐ center: ☐ space-between: ☐ space-around:



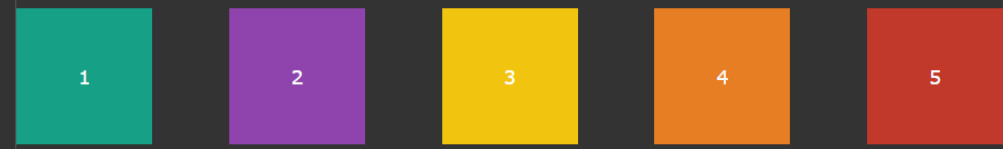
☐ flex-start: ☐ flex-end: ☒ center: ☐ space-between: ☐ space-around:



☐ flex-start: ☒ flex-end: ☐ center: ☐ space-between: ☐ space-around:



☐ flex-start: ☐ flex-end: ☐ center: ☒ space-between: ☐ space-around:



☐ flex-start: ☐ flex-end: ☐ center: ☐ space-between: ☒ space-around:



Flex: align-items

align-items

Item의 **교차축을 정렬**하는 속성입니다.

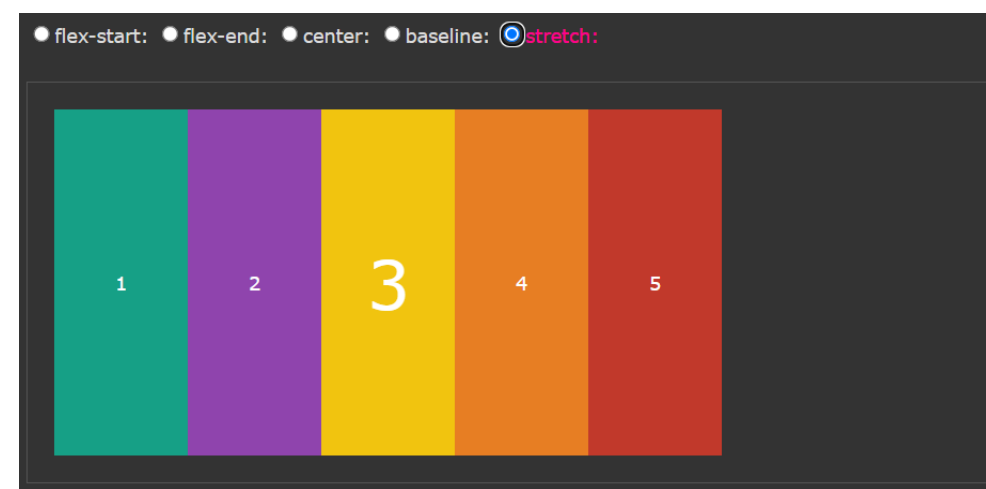
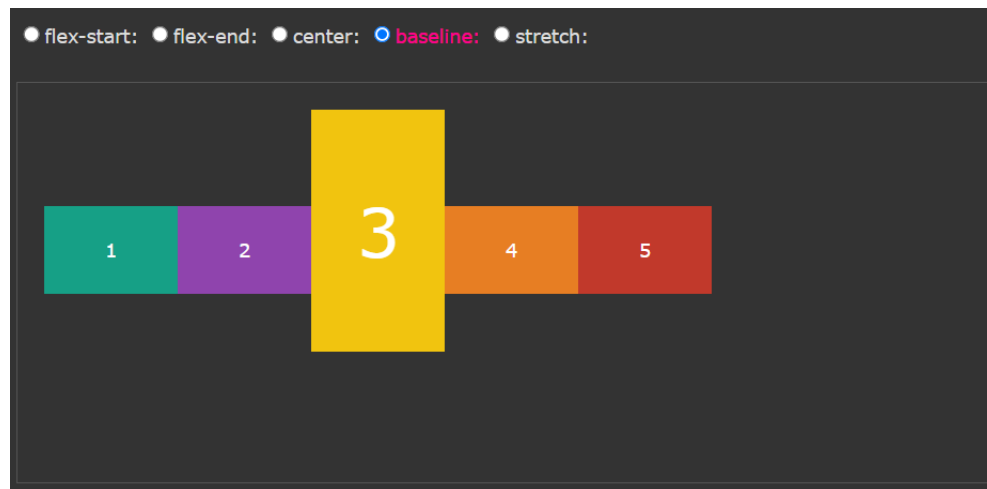
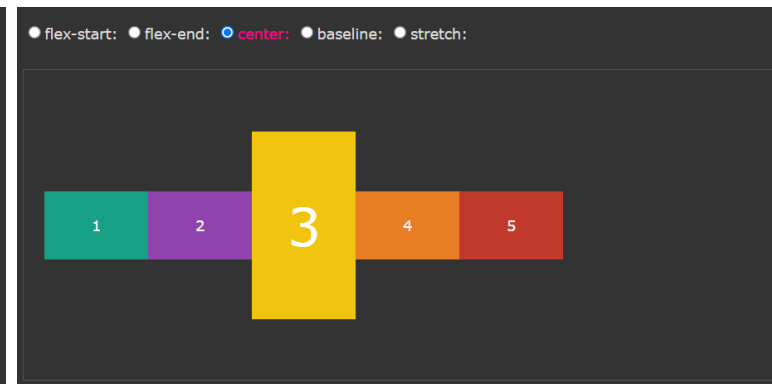
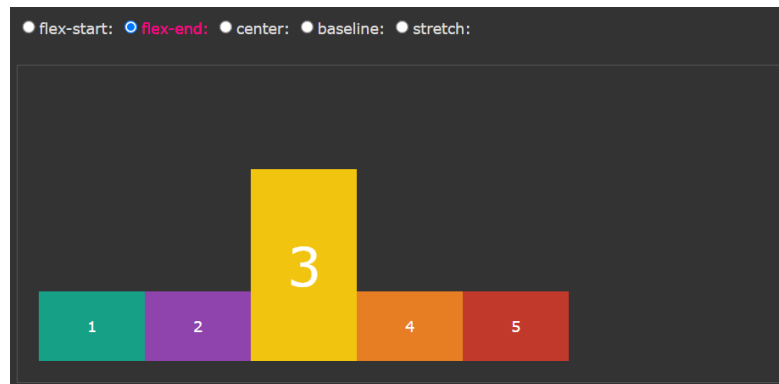
메인축에 수직인 축을 교차축이라고 합니다.

align-items 속성을 따로 지정하지 않으면 기본값인 stretch가 적용됩니다.

- ✓ **stretch(기본값)** : container의 높이만큼 교차축 방향으로 item을 늘려, 전체 높이를 채웁니다.
- ✓ **flex-start** : 교차축의 시작 지점을 기준으로 Item을 정렬합니다.
- ✓ **flex-end** : 교차축의 마지막 지점을 기준으로 Item을 정렬합니다.
- ✓ **center** : 교차축의 Item을 가운데 정렬합니다.
- ✓ **baseline** : 글꼴의 기준선인 baseline을 기준으로 Item을 정렬합니다.

```
display: flex;  
align-items: stretch;
```

Flex: align-items



Flex: align-content

align-content

Item의 **교차축을 정렬**하는 속성입니다.

주의해야 할 점은 align-items 속성과 비슷하다는 것입니다.

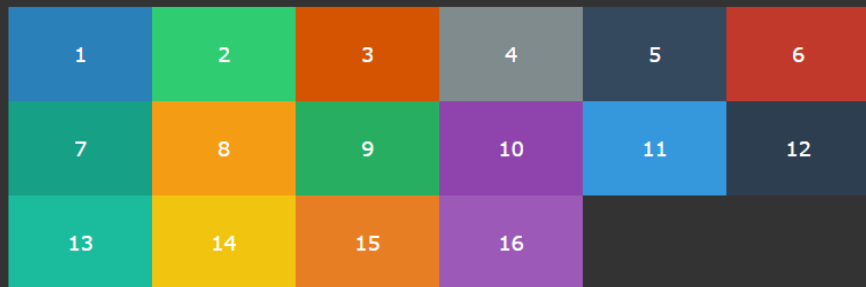
확실하게 다름을 구분해서 사용하는 방법은 align-items은 Item이 한 줄로 나열되었을 경우 사용할 수 있는 속성이고, align-content는 **Item이 여러 줄(2줄 이상) 나열**되어 있을 때 교차축 정렬을 위해 사용할 수 있는 속성입니다.

- ✓ **stretch(기본값)** : container의 높이만큼 교차축 방향으로 item을 늘려, 전체 높이를 채웁니다.
- ✓ **flex-start** : 교차축의 시작 지점을 기준으로 Item을 정렬합니다.
- ✓ **flex-end** : 교차축의 마지막 지점을 기준으로 Item을 정렬합니다.
- ✓ **center** : 교차축의 Item을 가운데 정렬합니다.
- ✓ **space-between** : 교차축의 첫 번째 Item은 시작점에 마지막 Item은 끝 지점 정렬하고, 나머지 item은 사이에 동일한 간격으로 정렬합니다.
- ✓ **space-around** : 교차축을 기준으로 Item둘레에 동일한 간격으로 정렬합니다.

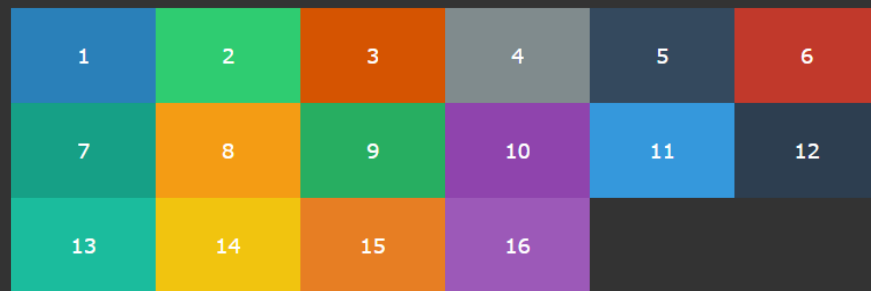
```
display: flex;  
flex-wrap: wrap;  
align-content: center;
```

Flex: align-content

☒ flex-start: ☐ flex-end: ☐ center: ☐ space-between: ☐ space-around: ☐ stretch:

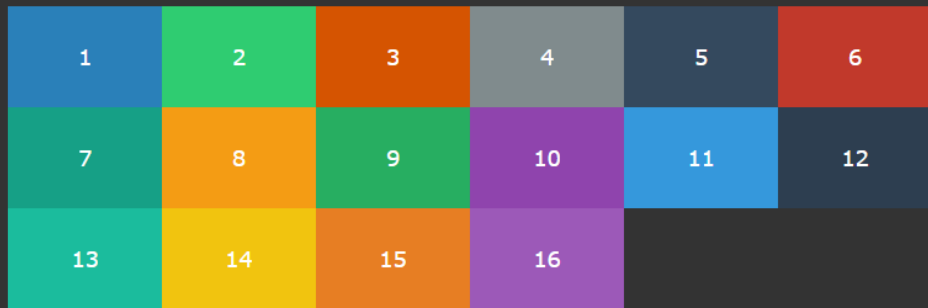


☐ flex-start: ☒ flex-end: ☐ center: ☐ space-between: ☐ space-around: ☐ stretch:



Flex: align-content

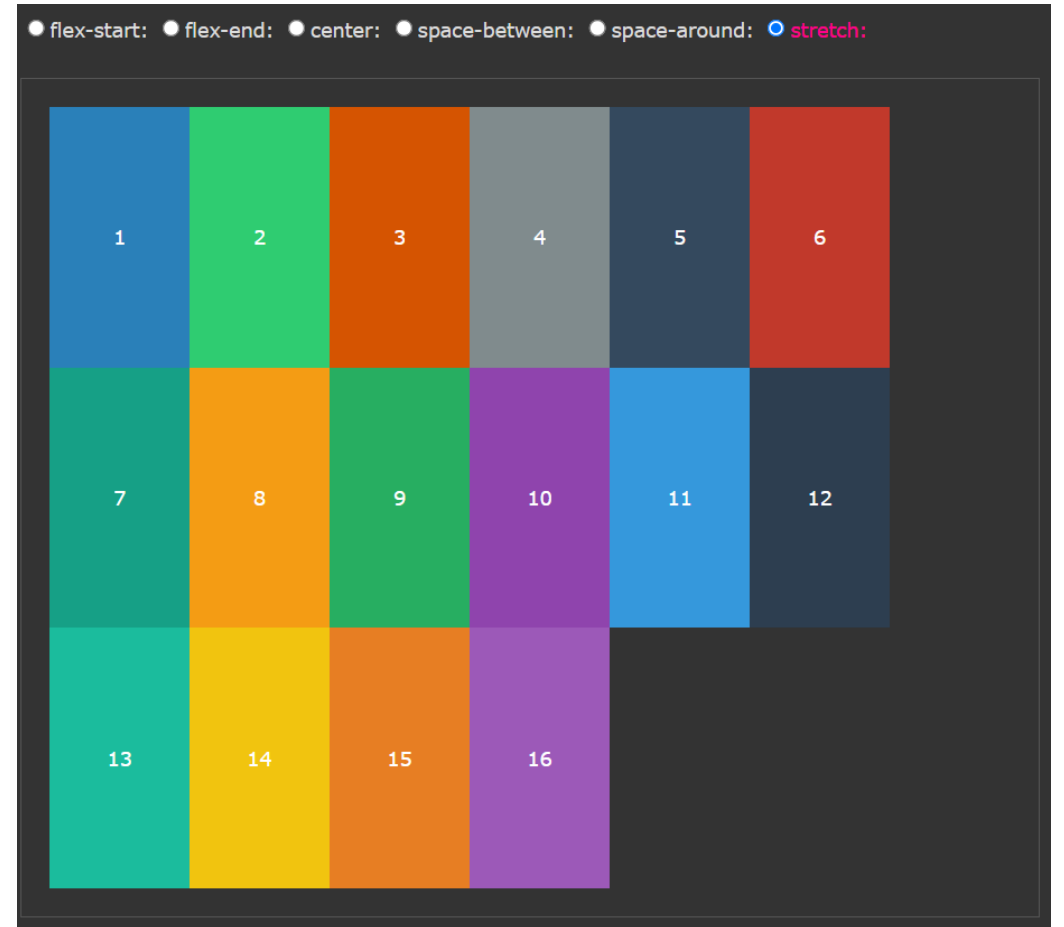
☐ flex-start: ☐ flex-end: ☒ center: ☐ space-between: ☐ space-around: ☐ stretch:



☐ flex-start: ☐ flex-end: ☐ center: ☒ space-between: ☐ space-around: ☐ stretch:

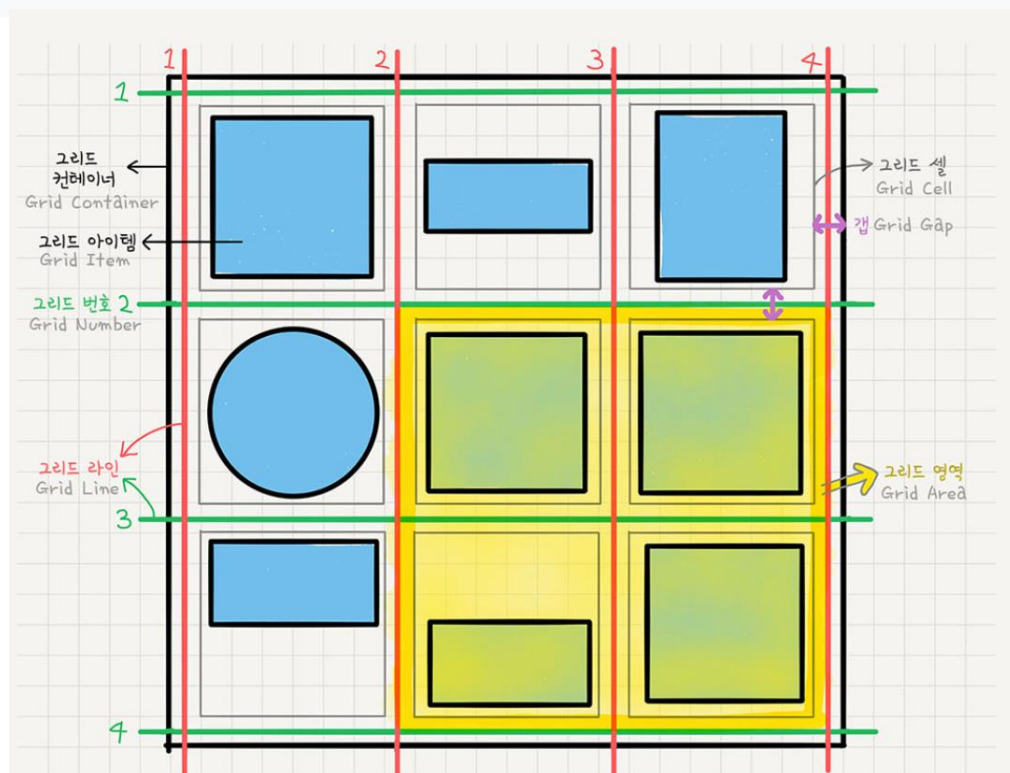


Flex: align-content



Grid

Flex가 한 방향 레이아웃 시스템이라면 Grid는 두 방향(가로-세로) 레이아웃 시스템이다.
Grid는 부모요소와 자식요소로 나뉜다 부모요소를 Container 자식요소를 item 이라 부른다



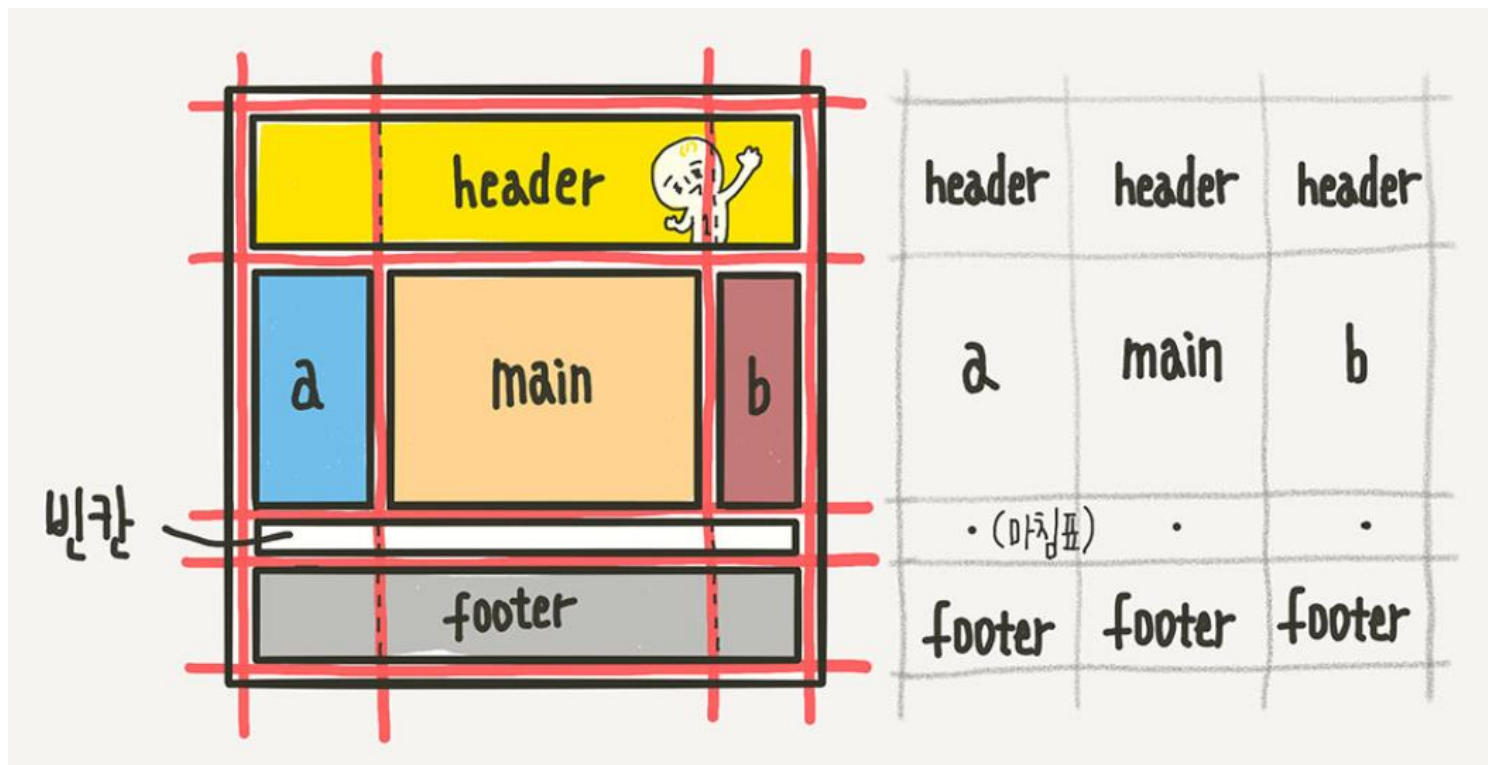
Grid: grid-template-columns와 grid-template-row

그리드 형태 정의
grid-template-rows
grid-template-columns

```
.container {  
    grid-template-columns: 200px 200px 500px;  
    /* grid-template-columns: 1fr 1fr 1fr */  
    /* grid-template-columns: repeat(3, 1fr) */  
    /* grid-template-columns: 200px 1fr */  
    /* grid-template-columns: 100px 200px auto */  
  
    grid-template-rows: 200px 200px 500px;  
    /* grid-template-rows: 1fr 1fr 1fr */  
    /* grid-template-rows: repeat(3, 1fr) */  
    /* grid-template-rows: 200px 1fr */  
    /* grid-template-rows: 100px 200px auto */  
}
```


Grid : grid-template-area

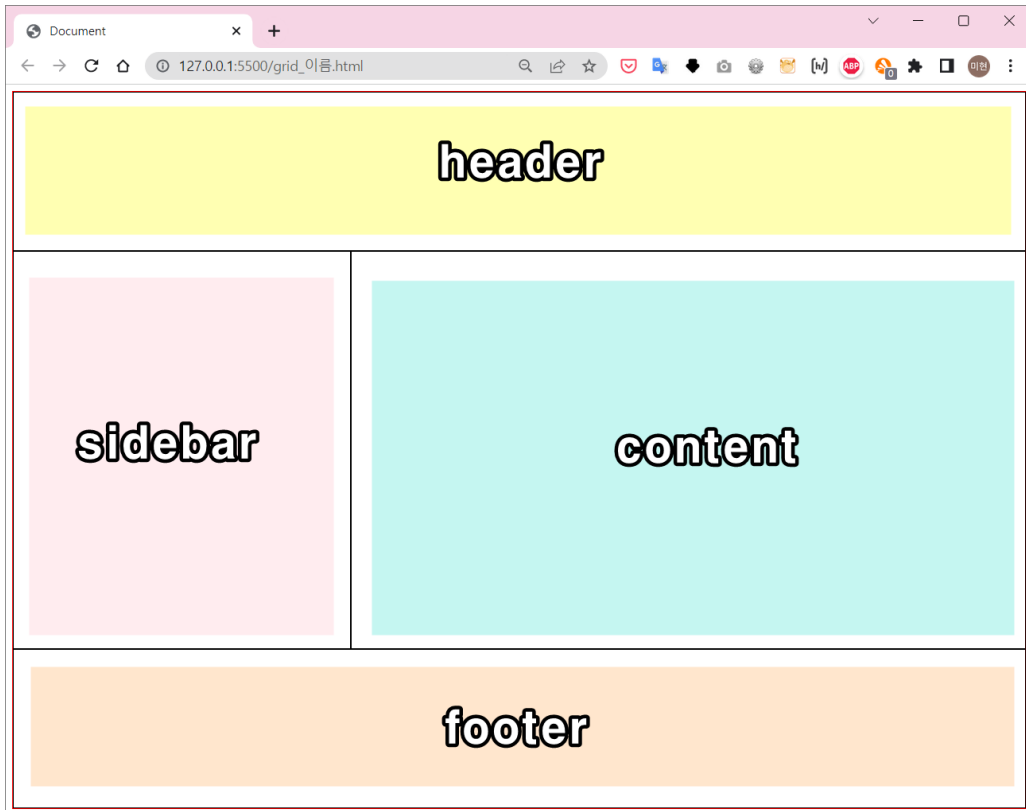
그리드 레이아웃 행과 열이름으로 지정



```
.container {
  grid-template-areas:
    "header header header"
    " a   main   b   "
    " .     .     .   "
    "footer footer footer";
}
```

```
.header { grid-area: header; }
.sidebar-a { grid-area: a; }
.main-content { grid-area: main; }
.sidebar-b { grid-area: b; }
.footer { grid-area: footer; }
/* 이름 값에 따옴표가 없는 것에 주의하세요 */
```

Grid : grid-template-area



```
<style>
.grid-container{
  display: grid;
  grid-template-areas:
    "header header header"
    "sidebar content content"
    "footer footer footer";
  border: 1px solid red;
  grid-template-rows: 200px 500px 200px;
}
.header{
  grid-area: header;
}
.sidebar{
  grid-area: sidebar;
}
.content{
  grid-area: content;
}
.footer{
  grid-area: footer;
}
div{
  border: 1px solid black;
}
</style>
```

```
<body>
  <div class="grid-container">
    <div class="header"></div>
    <div class="sidebar"></div>
    <div class="content"></div>
    <div class="footer"></div>
  </div>
</body>
```