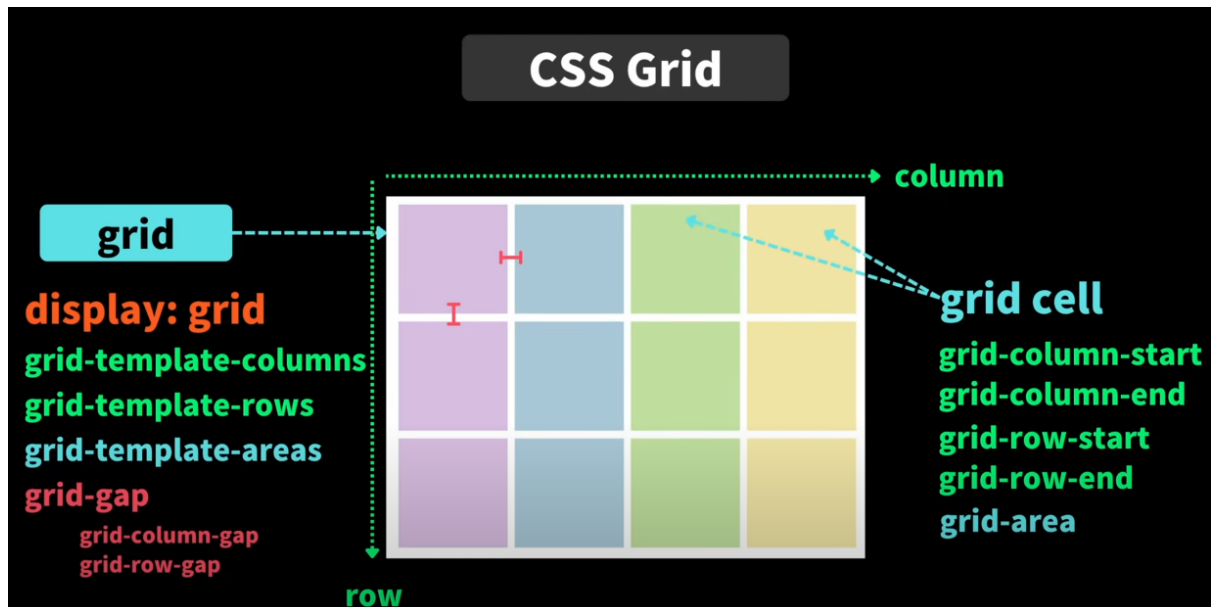


Day4

📎 자료	Web
☰ 구분	Web

그리드 시스템에 대해서 살펴보자



[출처: 유튜브 에 있는것 캡처함]

flex로 1차원 적인 공간을 나누고 배치를 했다면

grid를 이용하면 2차원 적으로 공간을 나누고 배치할 수 있다.

위 그림에서 보면 가로축으로 몇 개의 col 이 들어갈지 결정하는 grid-template-columns

세로축으로 몇 개의 row 가 들어갈지 결정하는 grid template rows

조금더 파워풀하고 플렉시블 하게 가로 세로를 나눌 수 있는 grid-template-areas 이 있다

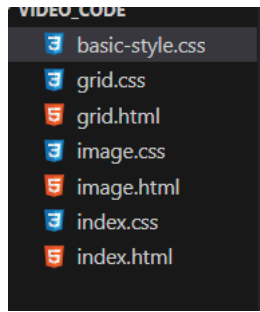
그리고 그리드 아이템 (그리드 셀) 을 조작을 할텐데

grid-column-start와 grid-column-end 로 가로 몇 번째 셀부터 몇 번째 셀까지 조작을 할 것인지

grid-row-start 와 end로 세로 몇 번째부터 몇 번째 까지 조작을 할 것인지를 지정할 것이다.

grid-area를 통해서 어떤 구역을 묶어서 조작을 할 것인지 지정해 주는 것 들을 실습 해 볼 것이다.

먼저 박스 부터 만들고 컨테이너에 grid를 부여해서 스타일링 하는것을 연습해 볼 것이다.



다음과 같이 실습할 파일들을 미리 만들어 놓자

그리고 grid.html 파일에서

먼저 박스 부터 만들고 컨테이너에 grid를 부여해서 스타일링 하는것을 연습해 볼 것이다.

[grid.html]

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="basic-style.css" />
    <link rel="stylesheet" href="grid.css" />
    <title>그리드 예제</title>
  </head>
  <body>
    <div class="container">
      .item.color{Item$}*10 빙금 라이브에서 배운 emit 이용해보자.
      <div class="item color1">Item1</div>
      <div class="item color2">Item2</div>
      <div class="item color3">Item3</div>
      <div class="item color4">Item4</div>
      <div class="item color5">Item5</div>
      <div class="item color1">Item6</div>
      <div class="item color2">Item7</div>
      <div class="item color3">Item8</div>
      <div class="item color4">Item9</div>
      <div class="item color5">Item10</div>
    </div>
  </body>
</html>
```

그다음 basic-style.css 파일에 위에서 만든 박스에 이미지를 입힐 것이다.

[basic-style.css]

```
* {
  /* 태두리를 기준으로 박스 사이즈를 하고 */
  box-sizing: border-box;
}

body {
  /* 실행중인 스크린 크기에 맞춰 전체화면을 body로 잡았다
  그리고 마진 0을 넣어서 html문서와 body사이의 간격을 없앴다 */
  width: 100vw;
  height: 100vh;
  margin: 0;
}

.item {
  /* 컨테이너 안에 items 들의 style을 부여하고 */
  display: flex;
  justify-content: center;
  align-items: center;
  border: 1px solid #181818;
  font-size: 1.2rem;
  font-weight: bold;
}
```

```

/* 각각의 박스에 색을 달리 적용했다 */
.color1 {
  background-color: #d7bee2;
}
.color2 {
  background-color: #a9c7d8;
}
.color3 {
  background-color: #c0df9f;
}
.color4 {
  background-color: #f2e5a6;
}
.color5 {
  background-color: #e89d9d;
}

```

Item1
Item2
Item3
Item4
Item5
Item6
Item7
Item8
Item9
Item10

이렇게 기본 셋팅이 끝났으면

[grid.css]

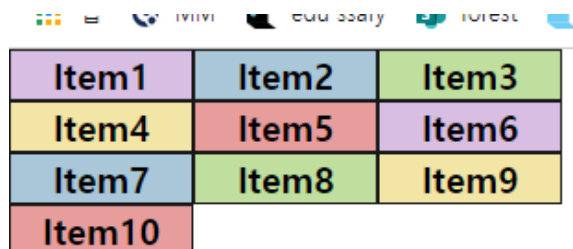
```

.container {
  display: grid;

  /* row 를 통해서 각각 3개의 컬럼을 만들것인데 각 100px씩 주겠다. */
  grid-template-columns: 100px 100px 100px;
}

```

[캡처2]



Item1	Item2	Item3
Item4	Item5	Item6
Item7	Item8	Item9
Item10		

그 다음에

```

.container {
  display: grid;

  grid-template-columns: 100px 100px 100px;
  grid-template-rows: 100px 100px 100px;
}

```

```
}
```

도 추가해 보자.

[캡처2-1]



그러면 끝에 한 줄이 남는 칸이 생기므로 마지막 template rows에 100px를 하나 더 부여해 줄 수 있다.

아래에 또다른 예시 이다. 간단한 구역 나누기를 할 수 있다.

(grid-template-rows: 100px 200px 100px 100px;)

```
.container {  
  display: grid;  
  
  grid-template-columns: 100px 100px 100px;  
  grid-template-rows: 100px 200px 100px 100px;  
}
```

[캡처3]

Item1	Item2	Item3
Item4	Item5	Item6
Item7	Item8	Item9
Item10		

그런데 컬럼을 5개로 나눈다고 가정 한다면

grid-template-columns: 100px 100px 100px 100px 100px;

100px를 5번이나 반복해서 쓰는것이 귀찮다. 이렇게 반복이 있을 경우

repeat 함수를 써서 아래와 같이 수정이 가능하다.

```
.container {
  display: grid;

  grid-template-columns: repeat(5,100px);
  grid-template-rows: 100px 200px repeat(2,100px);
}
```

[캡쳐4]

Item1	Item2	Item3	Item4	Item5
Item6	Item7	Item8	Item9	Item10

그런데 반응형으로 만들어야 하니까 px를 사용하는것 보다는 %비율로 나누거나 fr (fraction)을 통해서 나눠 주도록 하겠다.

```
.container {
  display: grid;
```

```

/* 내가 사용가능한 전체 너비(100%)중 20% 씩 5개로 나누겠다. */
grid-template-columns: repeat(5,20%);

grid-template-rows: 100px 200px repeat(2,100px);
}

```

Item1	Item2	Item3	Item4	Item5
Item6	Item7	Item8	Item9	Item10

이번에는

grid-template-columns: repeat(5,10%); 으로 바꿔보자.

그러면 전체 너비 100% 중 10씩 5 col을 만들기 때문에

전체 너비 중 50% 를 10%씩 5개의 칸으로 나눌 것이다.

[캡처6]

Item1	Item2	Item3	Item4	Item5
Item6	Item7	Item8	Item9	Item10

이번에는 % 대신 fr 을 이용해 보자

fr을 사용해서 내가 사용가능한 컨테이너 크기의 1fr 1fr 1fr 로 나누면 총 3개를 1:1:1 비율로 나눌 수 있다.

```

.container {
  display: grid;

  /* 내가 사용가능한 전체 너비를 1:1:1:1:1 비율로 나누겠다 */
  grid-template-columns: repeat(5,1fr);

  grid-template-rows: 100px 200px repeat(2,100px);
}

```

7번 캡처

Item1	Item2	Item3	Item4	Item5
Item6	Item7	Item8	Item9	Item10

실제로 row col을 이용해서 구역을 나눌 때

column을 나눈 후 정확하게 row가 몇줄이 나올지 계산하는 것은 상당히 귀찮다. 그때, 전체적으로 자동으로 사이즈를 지정해 주고 싶다면 grid-auto-rows 를 사용할 수 있다.

```
.container {
  display: grid;

  grid-template-columns: repeat(5,1fr);
  /*col을 나눈 후 모든 row를 150px로 나눠줘*/
  grid-auto-rows: 150px
}
```

8번 캡처

Item1	Item2	Item3	Item4	Item5
Item6	Item7	Item8	Item9	Item10

여기서 하나 더 추가 해 보자.

item 안에 lorem으로 랜덤한 글자를 넣어 볼 것인데

item 박스 크기에 비하여 글자가 너무 많이 들어가 있다면 텍스트가 잘릴 것이다.

그때에는 minmax() 를 사용해서 박스 크기를 콘텐츠의 크기만큼 늘려 줄 수 있다.

먼저 html items에 lorem 을 사용해서 문자열을 넣을 후

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="basic-style.css" />
    <link rel="stylesheet" href="grid.css" />
    <title>그리드 예제</title>
  </head>
```

```

<body>
  <div class="container">
    <div class="item color1">Item1</div>
    <div class="item color2">Item2</div>
    <div class="item color3">Item3</div>
    <div class="item color4">Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolorem libero facilis at non atque. Consectetur
    <div class="item color5">Item5</div>
    <div class="item color1">Item6</div>
    <div class="item color2">Item7</div>
    <div class="item color3">Item8</div>
    <div class="item color4">Item9</div>
    <div class="item color5">Item10</div>
  </div>
</body>
</html>

```

```

.container {
  display: grid;
  grid-template-columns: repeat(5, 1fr);

  /*컨텐츠를 박스 안에 넣고 나머지는 150px 으로 알아서 만들어줘 */
  grid-auto-rows: minmax(150px, auto);
}

```

9번

Item1	Item2	Item3	Lorem ipsum, dolor sit amet consectetur adipisicing elit. Dolorem libero facilis at non atque. Consectetur cupiditate, accusamus molestias ea voluptatem vero illum eaque ullam voluptate. Vitae cumque excepturi eveniet nam?	Item5
Item6	Item7	Item8	Item9	Item10

위 사진에 서 볼 수 있듯이 콘텐츠 크기만큼 박스가 늘어났다. 그리고

`grid-auto-rows: minmax(150px, auto);` 코드를 추가 했는데..

컨텐츠를 해당 박스 안에 넣고 나머지는 auto를 사용해서 150px 으로 알아서 만들어줘 라는뜻이다.

그리고 이제 item 별로 gap도 넣어 줄 수 있는데 한번 해보겠다.

```

.container {
  display: grid;
  grid-template-columns: repeat(5, 1fr);
  grid-auto-rows: minmax(150px, auto);

  /* col 갭을 10px 줘 */
}

```



```
grid-column-gap:10px;
}
```

브라우저 확인해 보라. 실수로 캡처를 못했다. 미안하다 ㅠ

물론 `grid-row-gap:10px;` 추가로 넣어주면 row col 모두 갭이 생기겠다. 그런데 row col 모두 갭을 줄 것이라면 아래와 같이 한 줄로 통 칠 수 있다.

```
.container {
  display: grid;
  grid-template-columns: repeat(5,1fr);
  grid-auto-rows: minmax(150px, auto);

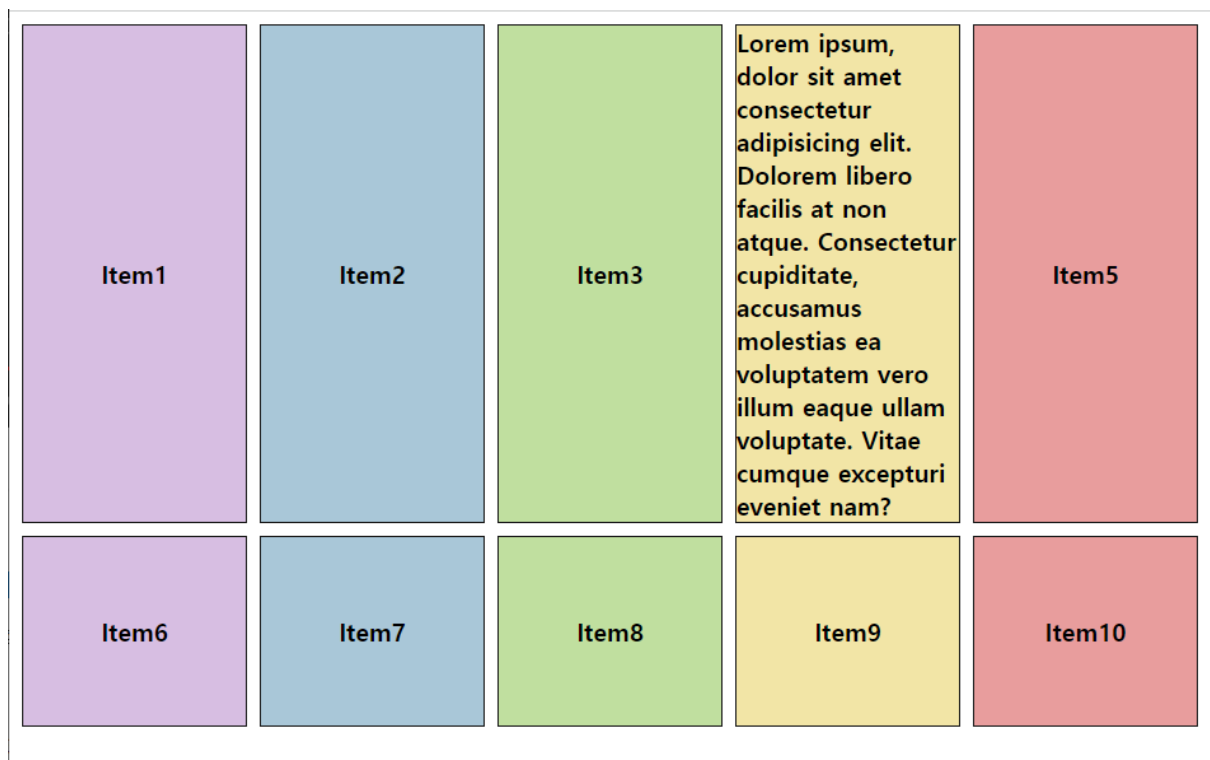
  /* row col 모두 갭을 10px 줘 */
  grid-gap: 10px;
}
```

위의 코드로 아이템들 사이사이 row col 에 갭을 줬다. 추가로 container 전체에 갭을 주고 싶다면 `padding`을 사용 할 수도 있다.

```
.container {
  display: grid;
  grid-template-columns: repeat(5,1fr);
  grid-auto-rows: minmax(150px, auto);
  grid-gap: 10px;

  /* 컨테이너 전체에 10씩 띄어줘 */
  padding: 10px;
}
```

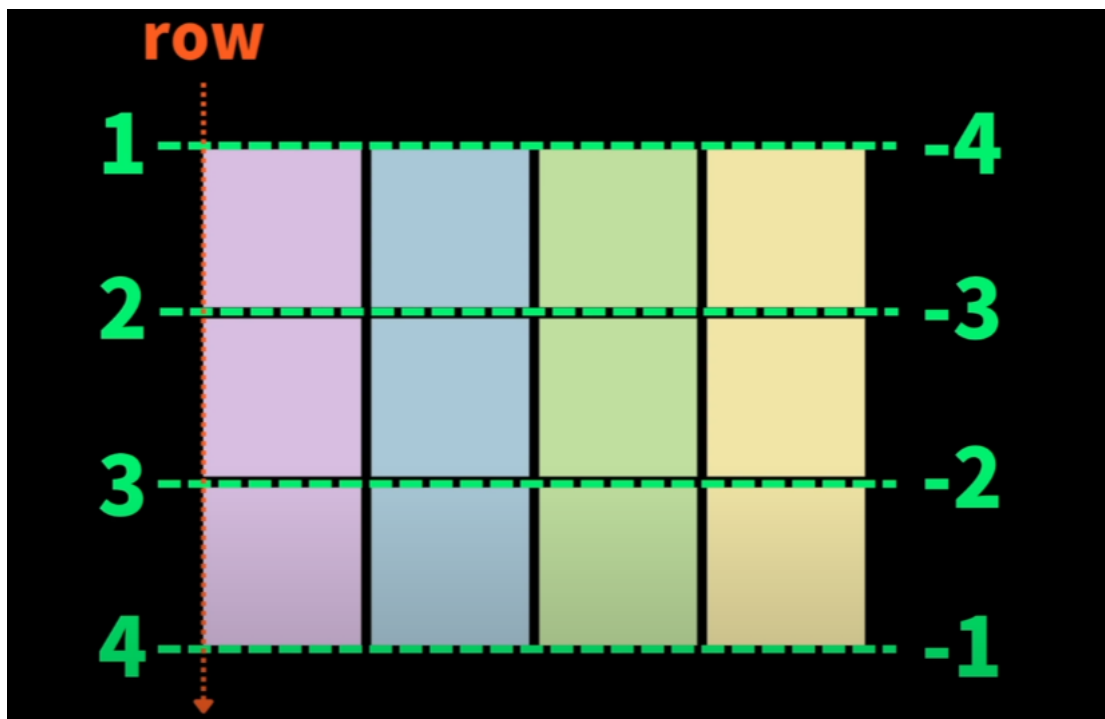
10번

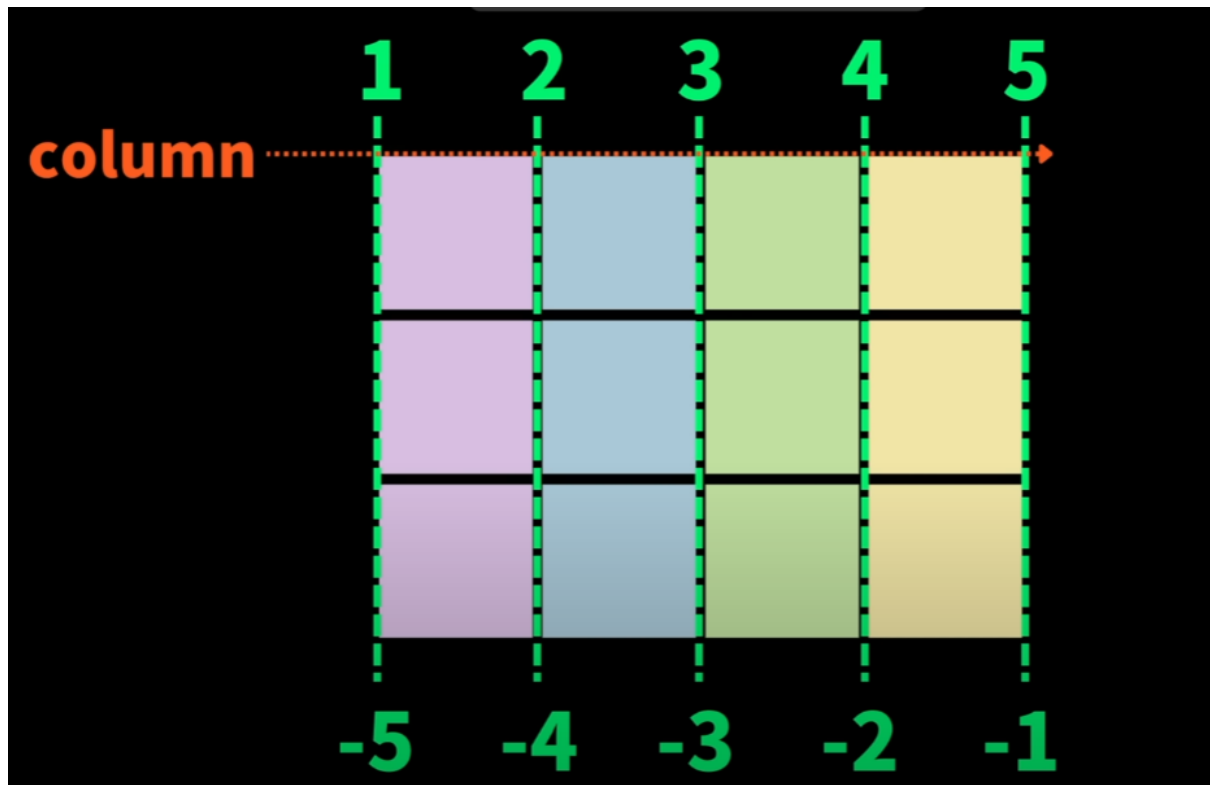


자 여기까지 grid를 적용한 부모 컨테이너에 컬럼과 로우를 지정해 보는것을 연습해 보았다.

지금 부터는 각각의 아이템 박스를 묶어서
어디서 부터 어디까지 범위를 지정해서
자리를 차지할 것 인지를 살펴 보겠다.

먼저 그리드 라인 (grid line) 에 대해서 공부해 보자.





11 // 12 캡처 [출처 유튜브]

col을 보면 좌측에서 우측으로 1,2,3,4,5 번 줄이며 뒤에서 부터는 카운트를 할 때에는

-1번 줄 부터 -5번 줄이 된다.

즉 첫번째 줄 부터 마지막 줄 까지 구역을 지정할 때에는 1 2 3 4 5 번으로 지정이 될 것이며 마지막 줄 부터 첫번째 줄 까지는 -1 -2 -3 -4 -5 로 구역을 지정할 수 있다.

row를 보면 위에서 아래로 아래로 내려 갈 때에는 1 2 3 4 번 줄이라고 보면 된다.

번호는 코드를 보면서 또 살펴보자.

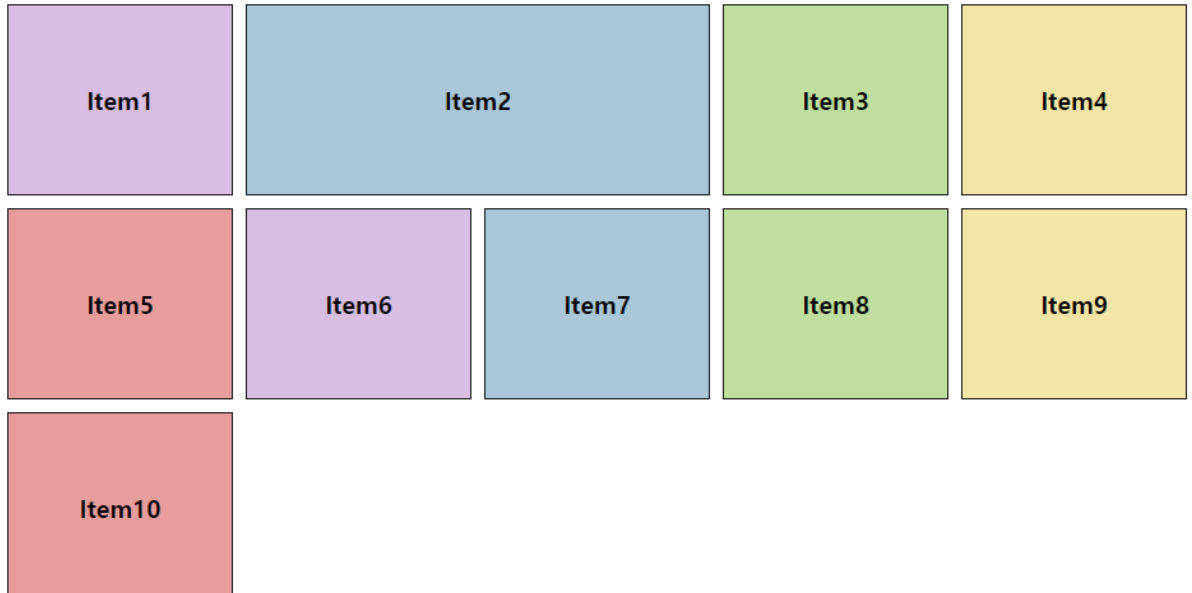
다음 실습을 위해서 grid.html 파일을 수정해 보자.

lorem 적어 둔 것을 지우자. 그리고 item2 라는 속성을 추가해 보자

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="basic-style.css" />
  <link rel="stylesheet" href="grid.css" />
  <title>그리드 예제</title>
</head>
<body>
  <div class="container">
    <div class="item color1">Item1</div>
    <div class="item item2 color2">Item2</div>
    <div class="item color3">Item3</div>
    <div class="item color4">Item4</div>
    <div class="item color5">Item5</div>
    <div class="item color1">Item6</div>
    <div class="item color2">Item7</div>
    <div class="item color3">Item8</div>
    <div class="item color4">Item9</div>
    <div class="item color5">Item10</div>
  </div>
```

```
</body>
</html>
```

아이템 두개를 겹쳐서 사진을 넣어 볼 것이다.



기존의 item 2와 item3 박스를 합쳐보자.

grid.css 파일에 . item2 라는 클래스를 추가해 주자

```
.container {
  display: grid;
  grid-template-columns: repeat(5,1fr);
  grid-auto-rows: minmax(150px, auto);
  grid-gap: 10px;
  padding: 10px;
}

.item2 {
  /* column이 2번줄 부터 4번줄 전까지 합칠꺼야 */
  grid-column-start: 2;
  grid-column-end: 4;
}
```

.container는 변경사항이 없다.

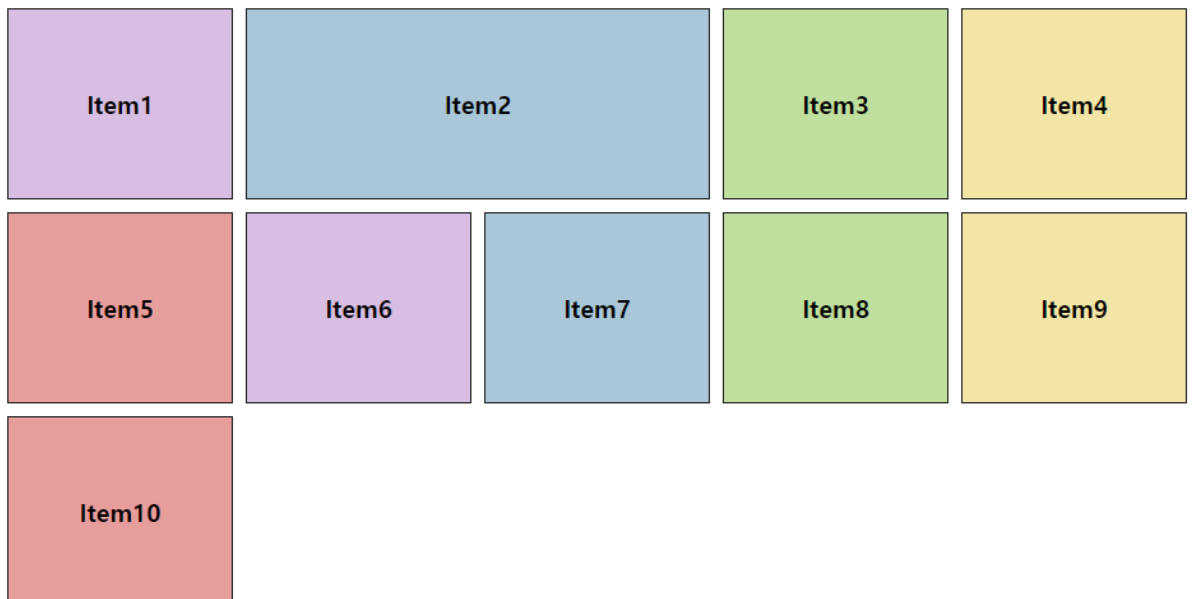
.item2 클래스에 css 적용시켜 보았다.

grid-column-start 와 end를 사용해서 어디까지 자리를 차지하고 싶은지 지정을 할 수 있는데

grid-column-start:2; end:4 를 통해서

column이 2번줄 부터 4번줄 전까지 즉 아이템 2번과 3번구역이 합쳤다.

13 캡처



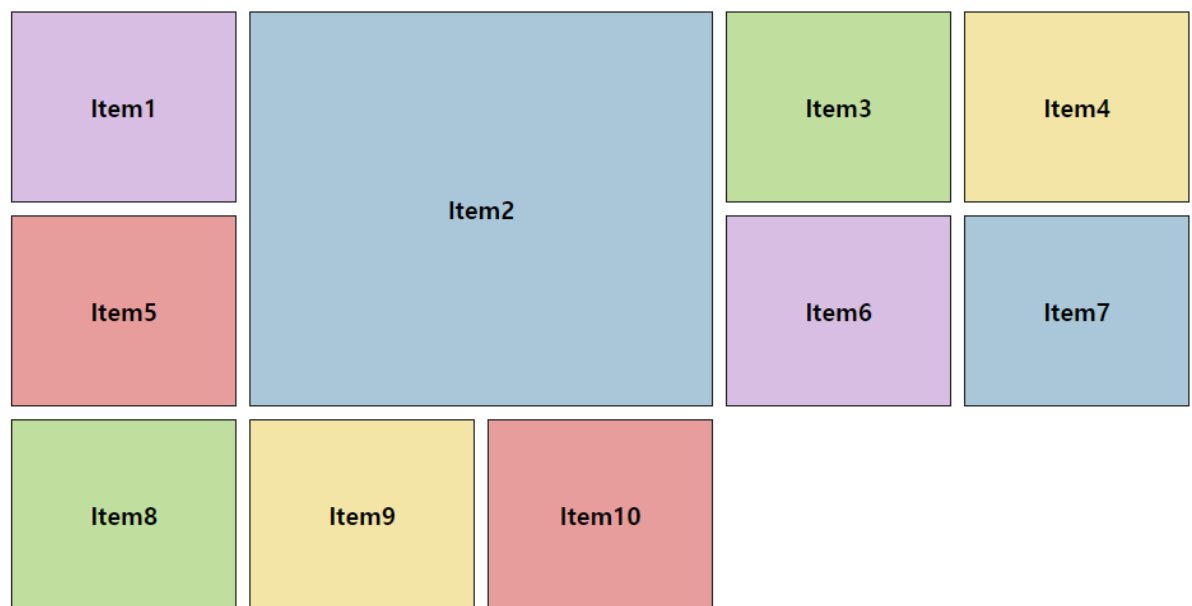
추가로

```
.item2 {
  /* column이 2번줄 부터 4번줄 전까지 합칠꺼야 */
  grid-column-start: 2;
  grid-column-end: 4;

  /* row의 1번줄 부터 3번줄 까지 합칠꺼야 */
  grid-row-start: 1;
  grid-row-end: 3;
}
```

row도 합치면

14번 캡처



위와 같이 아이템을 조절할 수 있다. 그런데

```
.item2 {  
    grid-column-start: 2;  
    grid-column-end: 4;  
    grid-row-start: 1;  
    grid-row-end: 3;  
}
```

식으로 다 써줘도 되지만 축약을 해보자. 아래와 같이 축약도 가능하다.

```
.item2 {  
    grid-column: 2/4;  
    grid-row: 1/3;  
}
```

그리고 추가로 2번 col부터 4번 col까지 line을 세는 것이 귀찮을 수 있다.

그럴 때에는 아래와 같이 표현도 가능하다.

```
.item2 {  
    grid-column: 2 / span 2;  
    grid-row: 1/3;  
}
```

2번 줄 부터 아이템 2개의 영역을 차지 할 것이라고 할 수 있다. span이 의미 하는것은 아이템(그리드셀)의 개수이다. span3 으로 수정후 (grid-column: 2 / span 3;)

브라우저 확인해 또 span빼고 -1도 적어보면서 (grid-column: 2 / -1;)

직접 확인해 보자.

자 여기까지 봤다면 조금 더 자유롭게 구역을 지정하고 나눌 수 있는 grid area에 대해서 살펴보자.

이제는 grid.html 그리고 grid.css 파일이 아닌 image.html 그리고 image.css 파일로 실습을 해보자.

image.html 파일부터 보자

컨테이너 안에 아이템으로 img 태그 7개를 넣었다.

그리고 unsplash 라는 무료로 이미지를 이용할 수 있는 사이트에서 이미지 url주소를 넣었다.

그리고 class 값을 image image1 .. image2 .. image3.. 해서 7개의 이미지의 태그에 class 속성값을 부여 해 놓았다.

[image.html]

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8" />  
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
    <link rel="stylesheet" href="basic-style.css" />  
    <link rel="stylesheet" href="image.css" />  
    <title>Document</title>  
  </head>  
  <body>  
    <section class="container">  
        
      
    
    
    
    
    
</section>
</body>
</html>

```

그다음 image.css 파일을 보자.

[image.css]

```

body {
    padding: 1rem;
    background-color: black;
}

.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-auto-rows: 150px;
    grid-gap: 1rem;
}

.image {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

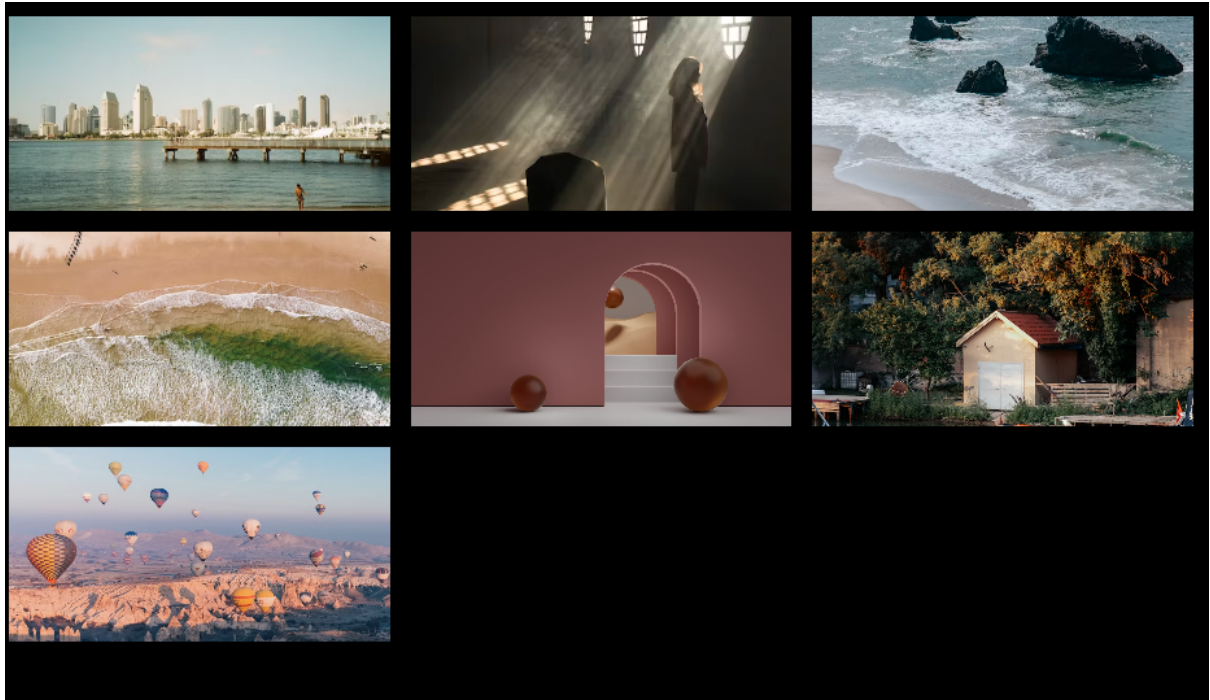
```

위 코드를 보면 object-fit: cover; 를 제외하고는 다 많이 보거나 사용해보았을 것이다.

object-fit: cover; 는 이미지 또는 비디오 태그의 크기를 박스안에 딱 맞도록 조정하는 속성이다. 추가적으로 구글링 검색해 보기를 권장한다.

다 완성이 되었다면 아래와 같은 모습을 확인할 수 있을 것이다.

캡처 15



셋팅이 끝났다면

캡처 16



위의 그림처럼 여러개의 영역을 묶어 볼 것이다.

여러개의 영역을 묶어서 표현해 주려고 할때
grid-template-areas를 사용할 수 있다.

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 150px;
  grid-gap: 1rem;

  /* 자유롭게 구역을 지정한다. */
  grid-template-areas:
    'a a a'
    'b c c'
    'b d g'
    'e f g';
}
```

이렇게 영역을 지정 한 후에는

7개의 class에 (처음 html 문서 만들 때 각 이미지 마다 클래스를 지정 했었음)

grid-area가 해당 이미지가 어떤 영역에 표기가 될 지정을 해주면 된다.

```
.image1 {
  grid-area: a;
}

.image2 {
  grid-area: b;
}
```

```

.image3 {
  grid-area: c;
}

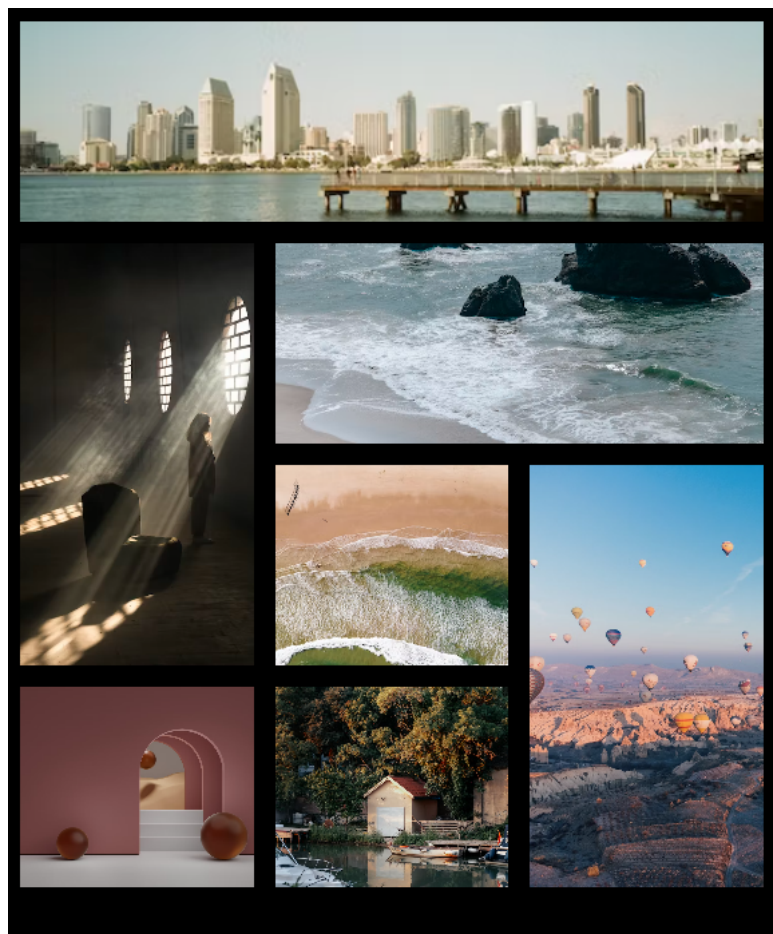
.image4 {
  grid-area: d;
}

.image5 {
  grid-area: e;
}

.image6 {
  grid-area: f;
}

.image7 {
  grid-area: g;
}

```



완성된 image.css 파일은 다음과 같다.

```

body {
  padding: 1rem;
  background-color: black;
}

.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 150px;
  grid-gap: 1rem;
  grid-template-areas:
    'a a a'
    'b c c'
    'b d g'

```

```

    'e f g';
}

.image {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

.image1 {
  grid-area: a;
}

.image2 {
  grid-area: b;
}

.image3 {
  grid-area: c;
}

.image4 {
  grid-area: d;
}

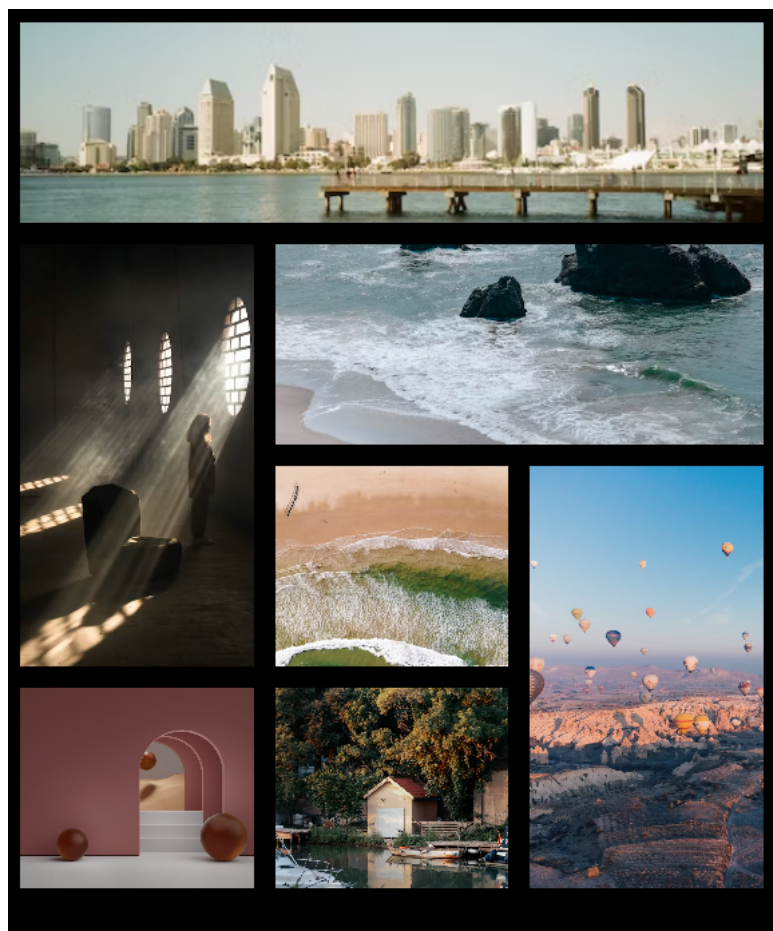
.image5 {
  grid-area: e;
}

.image6 {
  grid-area: f;
}

.image7 {
  grid-area: g;
}

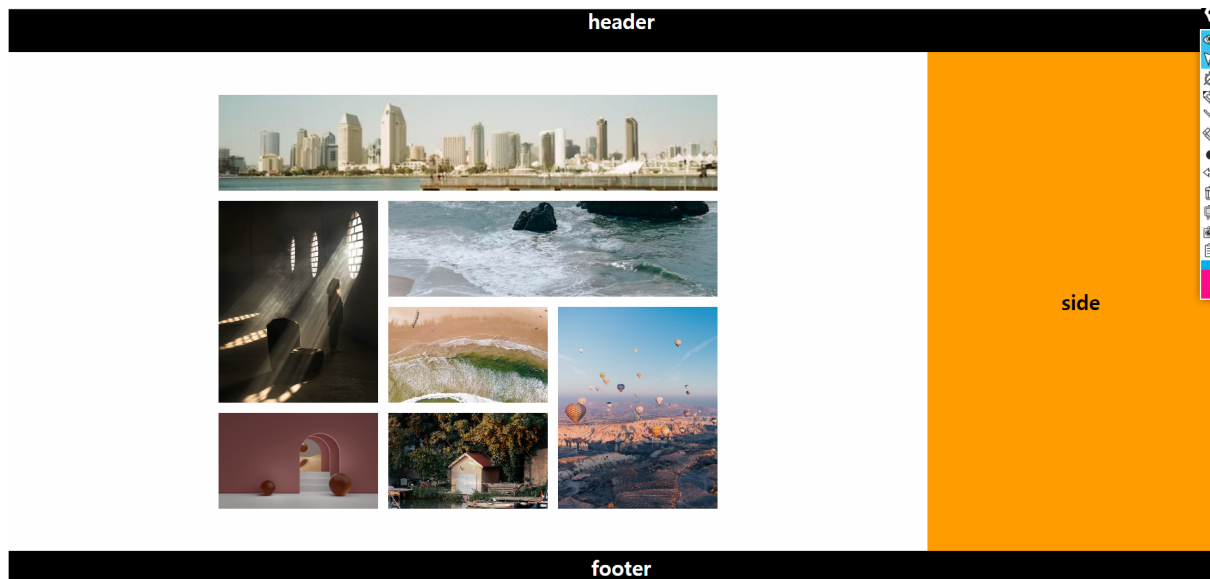
```

캡처 17



자 여기까지 했다면 마지막 미션이다. 정답을 보지 말고 직접 만들어 보자.

캡처 18



index.html 그리고 index.css 파일을 사용해서 위 그림과 같이 만들어 보자

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="image.css" />
    <link rel="stylesheet" href="index.css" />
    <title>Document</title>
  </head>
  <body>
    <header>header</header>
    <main>
      <section class="container">
        image.html 코드를 복사하면 됨
      </section>
    </main>
    <aside>side</aside>
    <footer>footer</footer>
  </body>
</html>
```

index.css

```
body {
  font-size: 2rem;
  font-weight: bold;
  background-color: white;
}

header,
main,
aside,
footer {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

```

header {
  background-color: black;
  color: white;
}

aside {
  background-color: #ff9d00;
}

footer {
  background-color: black;
  color: white;
}

body {
  width: 100vw;
  height: 100vh;
  margin: 0;
  display: grid;
  grid-template-columns: 3fr 1fr;
  grid-template-rows: 100px auto 50px;
  grid-template-areas:
    'header header'
    'main side'
    'footer footer';
}

header {
  grid-area: header;
}

main {
  grid-area: main;
}

aside {
  grid-area: side;
}

footer {
  grid-area: footer;
}

```

수고 많았습니다. 2반 화이팅