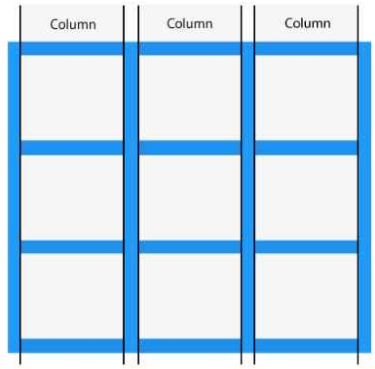
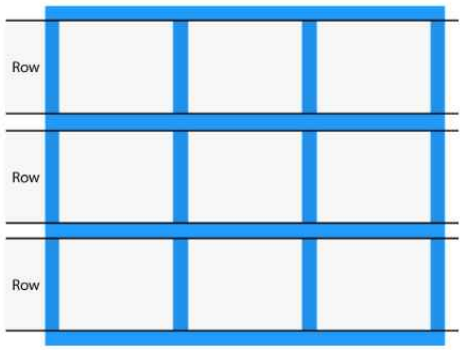
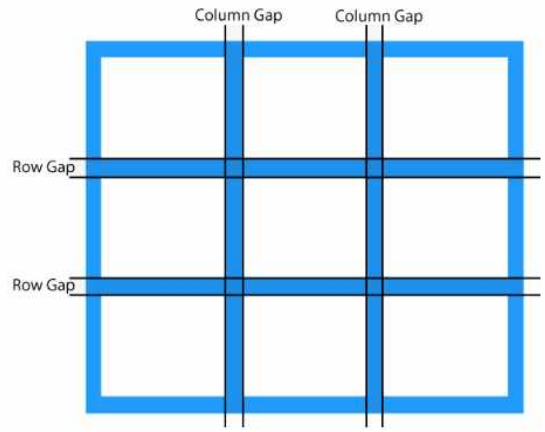
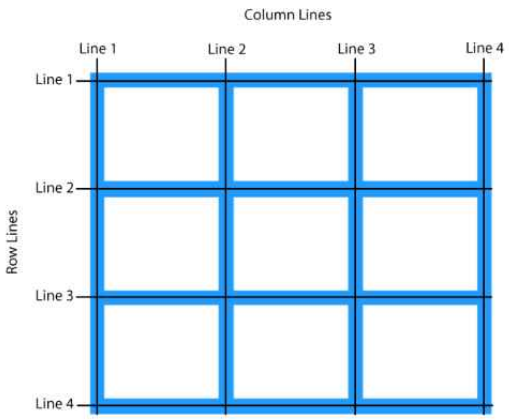
CSS grid

CSS Grid(그리드)는 2차원(행과 열)의 레이아웃 시스템을 제공합니다.
Flexible Box도 훌륭하지만 비교적 단순한 1차원 레이아웃을 위하며, 좀 더 복잡한 레이아웃을 위해 우리는 CSS Grid를 사용할 수 있습니다.
CSS 그리드 레이아웃(Grid Layout)은 페이지를 여러 주요 영역으로 나누거나, 크기와 위치 및 문서 계층 구조의 관점에서 HTML 기본 요소로 작성된 콘트롤 간의 관계를 정의하는 데 아주 탁월합니다.
테이블과 마찬가지로 그리드 레이아웃은 세로 열과 가로 행을 기준으로 요소를 정렬할 수 있습니다. 하지만, 테이블과 달리 CSS 그리드는 다양한 레이아웃을 훨씬 더 쉽게 구현할 수 있습니다. 예를 들어, 그리드 컨테이너 속 자식 요소를, 마치 CSS로 일일이 위치를 지정해 준 것처럼, 실제로 겹치게 층을 지면서 자리를 잡도록 각 요소의 위치를 지정해 줄 수도 있습니다.









Grid Container Properties

속성	의미
display	그리드 컨테이너(Container)를 정의
grid-template-rows	명시적 행(Track)의 크기를 정의
grid-template-columns	명시적 열(Track)의 크기를 정의
grid-template-areas	영역(Area) 이름을 참조해 템플릿 생성
grid-template	grid-template-xxx의 단축 속성
row-gap(grid-row-gap)	행과 행 사이의 간격(Line)을 정의
column-gap(grid-column-gap)	열과 열 사이의 간격(Line)을 정의
gap(grid-gap)	xxx-gap의 단축 속성
grid-auto-rows	암시적인 행(Track)의 크기를 정의
grid-auto-columns	암시적인 열(Track)의 크기를 정의
grid-auto-flow	자동 배치 알고리즘 방식을 정의
grid	grid-template-xxx과 grid-auto-xxx의 단축 속성
align-content	그리드 콘텐츠(Grid Contents)를 수직(열 축) 정렬
justify-content	그리드 콘텐츠를 수평(행 축) 정렬
place-content	align-content와 justify-content의 단축 속성
align-items	그리드 아이템(Items)들을 수직(열 축) 정렬
justify-items	그리드 아이템들을 수평(행 축) 정렬
place-items	align-items와 justify-items의 단축 속성

Grid Item Properties

속성	의미
grid-row-start	그리드 아이템(Item)의 행 시작 위치 지정
grid-row-end	그리드 아이템의 행 끝 위치 지정
grid-row	grid-row-xxx의 단축 속성(행 시작/끝 위치)
grid-column-start	그리드 아이템의 열 시작 위치 지정
grid-column-end	그리드 아이템의 열 끝 위치 지정
grid-column	grid-column-xxx의 단축 속성(열 시작/끝 위치)
grid-area	영역(Area) 이름을 설정하거나, grid-row와 grid-column의 단축 속성
align-self	단일 그리드 아이템을 수직(열 축) 정렬
justify-self	단일 그리드 아이템을 수평(행 축) 정렬
place-self	align-self와 justify-self의 단축 속성
order	그리드 아이템의 배치 순서를 지정
z-index	그리드 아이템의 쌓이는 순서를 지정

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
```

1. 그리드 형태 정의

grid-template-rows

grid-template-columns

```
.grid-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-template-rows는 행(row)의 배치, grid-template-columns는 열(column)의 배치를 결정합니다.

grid-template-columns: 200px 200px 500px;-->column을 200px, 200px, 500px

grid-template-columns: 1fr 1fr 1fr;
fr은 fraction(뜻은 여기로[새 창])인데요, 숫자 비율대로 트랙의 크기를 나눕니다.
즉 위의 1fr 1fr 1fr은 균일하게 1:1:1 비율인 3개의 column을 만들겠다는 의미에요.
고정 크기와 가변 크기를 섞어 쓸 수도 있습니다.
첫번째 column은 100px로 고정되고, 나머지 두번째 세번째 column은 2:1의 비율로 유연하게 움직이게 됩니다.
grid-template-columns: 100px 2fr 1fr;

repeat 함수

```
.container {
    grid-template-columns: repeat(5, 1fr);
    /* grid-template-columns: 1fr 1fr 1fr 1fr 1fr */
}
```

repeat는 반복되는 값을 자동으로 처리할 수 있는 함수입니다.
repeat(반복횟수, 반복값)
즉, 위 코드의 repeat(5, 1fr)은 1fr 1fr 1fr 1fr 1fr과 같습니다.
repeat(3, 1fr 4fr 2fr); 이런 식의 패턴도 가능합니다.

minmax 함수
최소값과 최대값을 지정할 수 있는 함수입니다.
minmax(100px, auto)의 의미는 최소한 100px, 최대는 자동으로(auto) 늘어나게 입니다. 즉 아무리 내용의 양이 적더라도 최소한 높이 100px은 확보하고, 내용이 많아 100px이 넘어가면 알아서 늘어나도록 처리해 준 예시입니다.

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

auto-fill과 auto-fit
auto-fill과 auto-fit은 column의 개수를 미리 정하지 않고 설정된 너비가 허용하는 한 최대한 셀을 채웁니다.

```
.grid-container" {
    grid-template-columns: repeat(auto-fill, minmax(20%, auto));
}
```

2. 간격 만들기
row-gap, column-gap
gap
그리드 셀 사이의 간격을 설정합니다.

```
.grid-container {
    row-gap: 10px;
    /* row의 간격을 10px로 */
    column-gap: 20px;
    /* column의 간격을 20px로 */
}
.grid-container {
    gap: 10px 20px;
    /* row-gap: 10px; column-gap: 20px; */
}
```

3. 그리드 형태를 자동으로 정의
grid-auto-columns
grid-auto-rows
grid-template-columns(또는 grid-template-rows)의 통제를 벗어난 위치에 있는 트랙의 크기를 지정하는 속성입

니다. 속성 이름이 헷갈린다면 –template- 자리에 – auto-가 들어간다고 생각하세요.

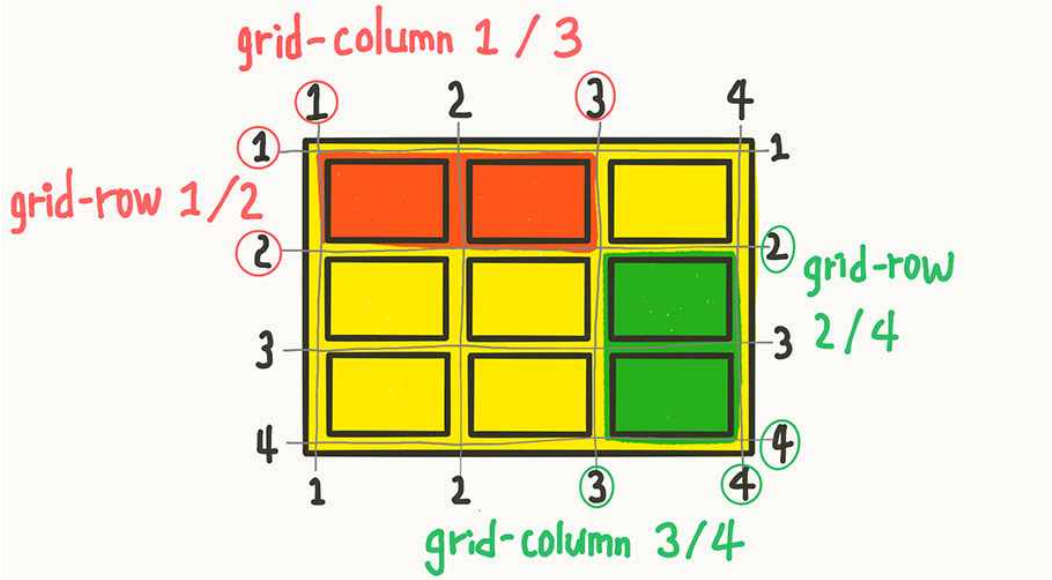
```
.grid-container {
        grid-template-rows: repeat(3, minmax(100px, auto));
}
```

각 셀마다 최소 100px의 높이를 확보하고, 컨텐츠가 높이 100px을 넘어가면 알아서 자동으로 늘어나도록(auto)
-->row 개수를 미리 알 수 없는 경우에 grid-auto-rows

4. 각 셀의 영역 지정

grid-column-start
grid-column-end
grid-column
grid-row-start
grid-row-end
grid-row

이 속성들은 Grid 아이템에 적용하는 속성으로, 각 셀의 영역을 지정합니다.



1부터 4까지의 Grid 라인 번호가 매겨져 있는데요, 바로 그 번호를 이용해서 column과 row의 범위를 결정하는 겁니다.

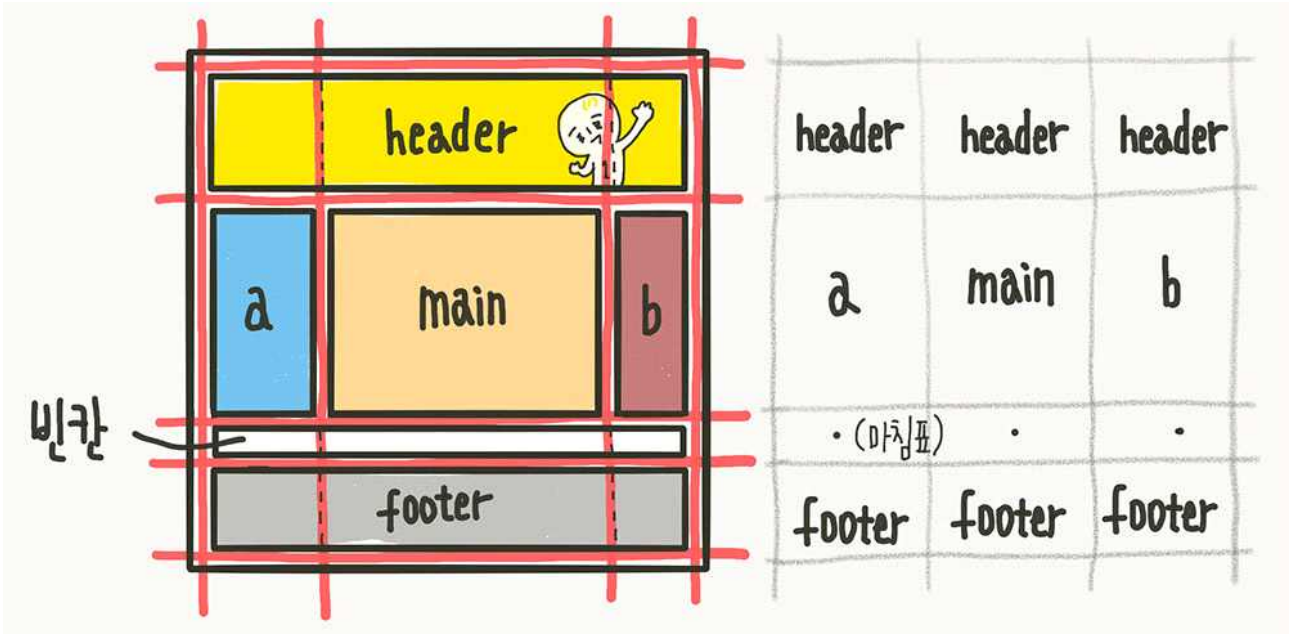
column으로 살펴보면, grid-column-start가 시작 번호, grid-column-end가 끝 번호입니다. grid-column은 start와 end 속성을 한번에 쓰는 축약형입니다.

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

5. 영역 이름으로 그리드 정의

grid-template-areas

각 영역(Grid Area)에 이름을 붙이고, 그 이름을 이용해서 배치하는 방법입니다.




```
.grid-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 */
```

6. 자동 배치

grid-auto-flow

아이템이 자동 배치되는 흐름을 결정하는 속성입니다.

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

7. 세로 방향 정렬

align-items

아이템들을 세로(column축) 방향으로 정렬합니다. 컨테이너에 적용합니다.

```
.grid-container {
        align-items: stretch;
        /* align-items: start; */
        /* align-items: center; */
        /* align-items: end; */
}
```

8. 가로 방향 정렬

justify-items

아이템들을 가로(row축) 방향으로 정렬합니다. 컨테이너에 적용합니다.

```
.grid-container {
        justify-items: stretch;
        /* justify-items: start; */
        /* justify-items: center; */
        /* justify-items: end; */
}
```

place-items

align-items와 justify-items를 같이 쓸 수 있는 단축 속성이에요.

align-items, justify-items의 순서로 작성하고, 하나의 값만 쓰면 두 속성 모두에 적용됩니다.

```
.grid-container {
        place-items: center start;
}
```

9. 아이템 그룹 세로 정렬

align-content

Grid 아이템들의 높이를 모두 합한 값이 Grid 컨테이너의 높이보다 작을 때 Grid 아이템들을 통째로 정렬합니다.

값	의미	기본값
normal	stretch와 같습니다.	normal
start	시작점(위쪽) 정렬	
center	수직 가운데 정렬	
end	끝점(아래쪽) 정렬	
space-around	각 행 위아래에 여백을 고르게 정렬	
space-between	첫 행은 시작점에, 끝 행은 끝점에 정렬되고 나머지 여백으로 고르게 정렬	
space-evenly	모든 여백을 고르게 정렬	
stretch	열 축을 채우기 위해 그리드 콘텐츠를 늘림	

```
.grid-container {
        align-content: stretch;
        /* align-content: start; */
        /* align-content: center; */
        /* align-content: end; */
        /* align-content: space-between; */
        /* align-content: space-around; */
        /* align-content: space-evenly; */
}
```

10. 아이템 그룹 가로 정렬

justify-content

Grid 아이템들의 너비를 모두 합한 값이 Grid 컨테이너의 너비보다 작을 때 Grid 아이템들을 통째로 정렬합니다.

```
.grid-container {
        justify-content: stretch;
        /* justify-content: start; */
        /* justify-content: center; */
        /* justify-content: end; */
        /* justify-content: space-between; */
        /* justify-content: space-around; */
        /* justify-content: space-evenly; */
}
```

place-content

align-content와 justify-content를 같이 쓸 수 있는 단축 속성이에요.

align-content, justify-content의 순서로 작성하고, 하나의 값만 쓰면 두 속성 모두에 적용됩니다.

```
.grid-container {
        place-content: space-between center;
}
```