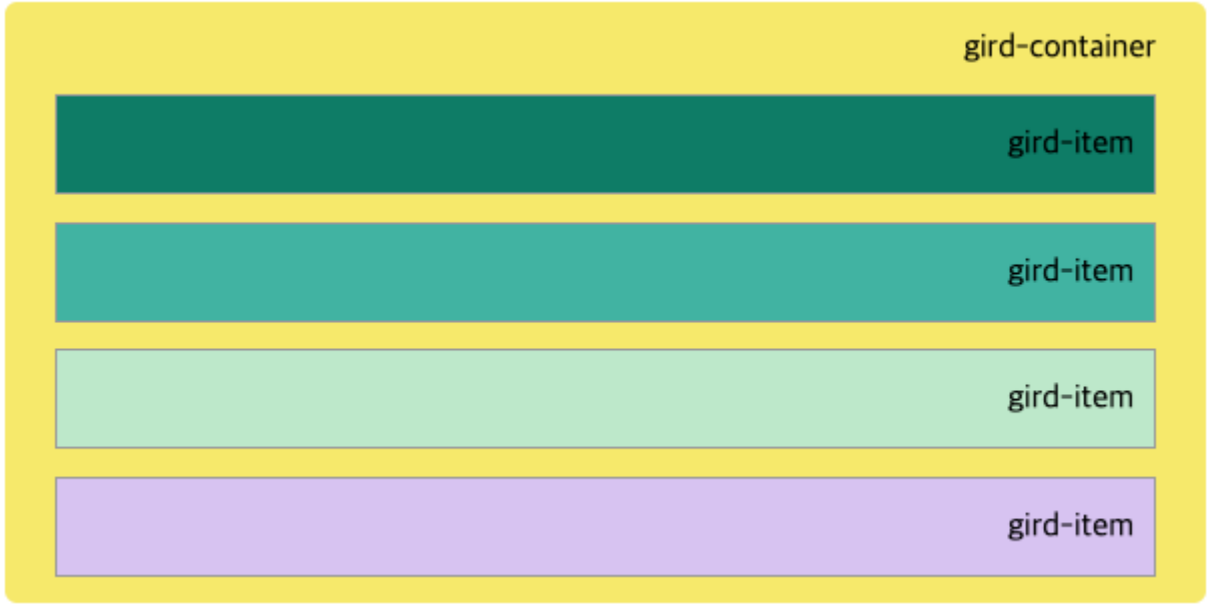
CSS Grid

CSS Grid란?

CSS 그리드 레이아웃 모듈은 브라우저가 선택한 HTML 요소를 'Grid 박스 모델'로 표시하도록 합니다.
그리드를 사용하면 그리드 컨테이너와 그 항목의 크기를 쉽게 조정하고 2차원으로 재배치할 수 있습니다.

"2차원"이란 Grid 모듈을 사용하면 상자 모델을 행과 열로 동시에 배치할 수 있습니다.
요소의 크기를 1차원으로 조정하고 재배치 하려면 Flexbox를 사용합니다.

Gird Container vs Grid Item

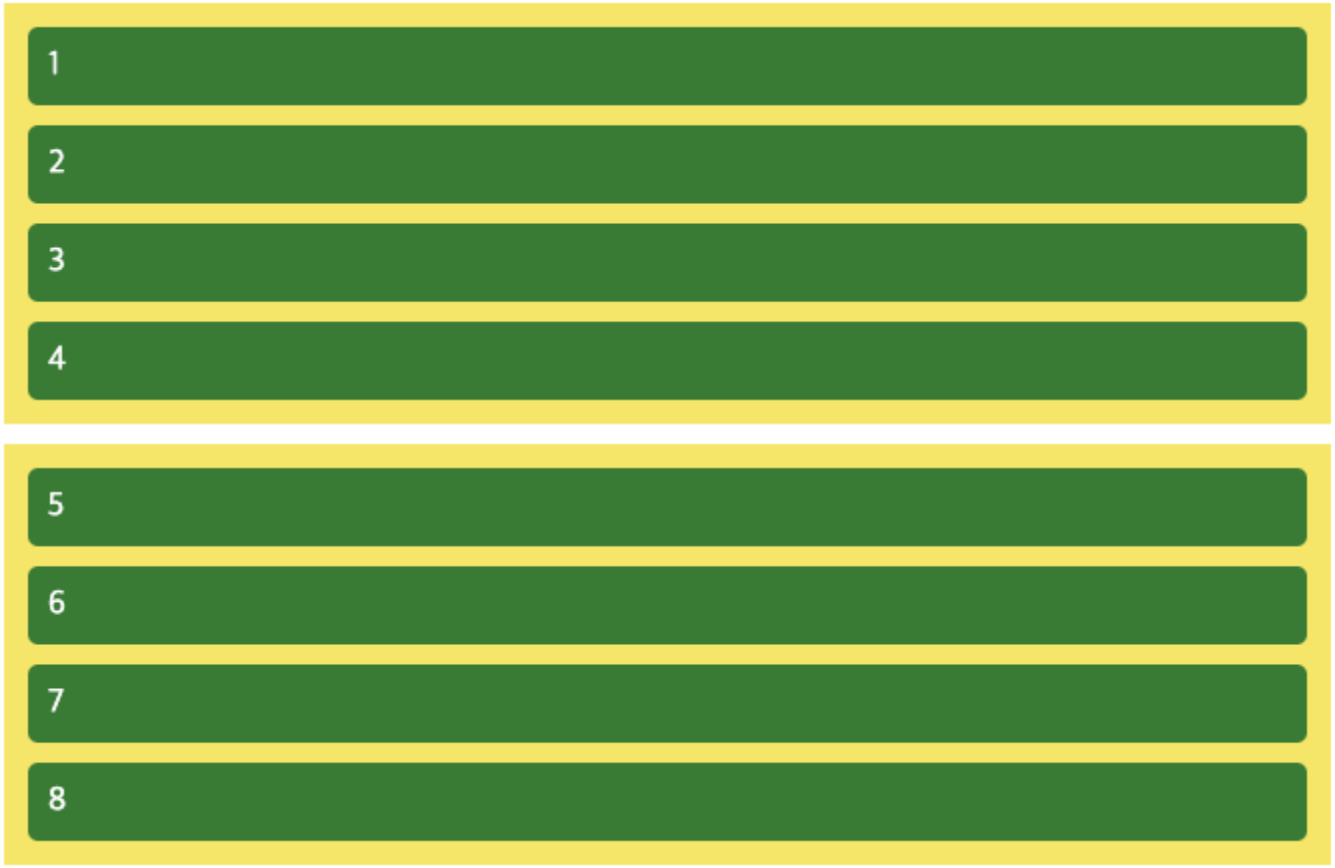


grid container는 display 속성 값이 grid 또는 inline-grid인 HTML 요소입니다. grid-item은 grid container의 하위 자식입니다.

display: grid

grid는 선택한 HTML 요소를 block-level의 grid box model로 표시하도록 브라우저에 지시합니다.
즉, 요소의 표시 속성 값을 grid로 설정하면 box model이 block-level grid 레이아웃 모듈로 전환됩니다.

```
section {
  display: grid;
  background-color: #F6E96B;
  margin: 10px;
  padding: 7px;
}
```

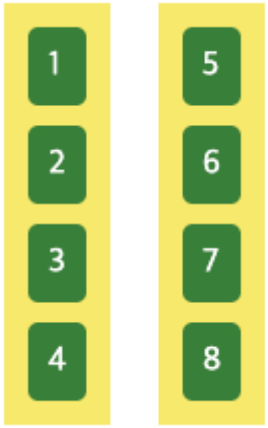


- `display: grid`는 하나의 열(column)의 grid-container를 생성합니다. 따라서 grid-item은 일반적 레이아웃 흐름대로 하나의 아이템이 다른 아이템 아래에 계속 표시됩니다.
- 노드를 grid box model로 변환하면 요소 바로 밑의 자식은 grid-item이 됩니다.
- `display: grid`는 box model과 바로 하위의 자식에만 영향을 미칩니다. 손자 노드에는 영향을 미치지 않습니다.

display: inline-grid

inline-grid는 선택한 HTML 요소를 inline-level의 grid box model로 표시하도록 브라우저에 지시합니다.

```
section {
  display: inline-grid;
  background-color: #F6E96B;
  margin: 10px;
  padding: 7px;
}
```



- 노드를 grid box model로 변환하면 요소 바로 밑의 자식은 grid-item이 됩니다.
- `display: inline-grid`는 box model과 바로 하위의 자식에만 영향을 미칩니다. 손자 노드에는 영향을 미치지 않습니다.

Grid 레이아웃 지정을 위한 속성

일반 HTML 요소를 grid 또는 inline-grid box model로 변환할 때 grid 레이아웃 모듈은 grid box와 그 바로 하위의 자식을 배치하기 위한 두 가지 범주의 속성을 제공합니다.

- Grid Container의 속성
- Grid Item의 속성

Grid Container의 속성

Grid Container의 속성은 브라우저가 grid box model 내에서 항목을 배치하는 방법을 지정합니다.
참고: Grid Container의 속성을 grid-item이 아닌 container에 정의합니다.

Grid Container 속성 8가지

- `grid-template-columns`
- `grid-template-rows`
- `grid-auto-columns`
- `grid-auto-rows`
- `justify-content`
- `justify-items`
- `align-content`
- `align-items`

grid-template-columns

`grid-template-columns` 는 브라우저가 선택한 grid container의 열(columns)의 수와 너비(widths)를 지정합니다.

Ex1) 2열 Grid Container 만들기

```
section {
  display: grid;
  grid-template-columns: 95px 1fr;
  background-color: #F6E96B;
  margin: 10px;
```

```
  padding: 7px;
}
```

grid-template-columns 속성을 사용하여 선택한 `<section>` grid container에 너비가 다른 두 개의 열을 표시했습니다.

참고: grid container에서 사용 가능한 공간의 비율을 기준으로 두 번째 열을 확장하기 위해 fr(fraction) 단위를 사용했습니다.

Fraction(fr)
`fr` 은 grid container에서 사용 가능한 공간의 비율에 상대적으로 요소를 확장합니다.

Ex2) 3열 Grid Container 만들기

```
section {
  display: grid;
  grid-template-columns: 15% 60% 25%;
  background-color: #F6E96B;
  margin: 10px;
  padding: 7px;
}
```

grid-template-columns 속성을 사용하여 선택한 `<section>` grid container에 너비가 다른 세 개의 열을 표시했습니다.

- `grid-auto-columns` 속성을 사용하여 grid container의 모든 열에 대한 기본 열 너비를 지정할 수 있습니다. 예를 들어 `grid-auto-columns: 150px` 는 모든 열에 대해 기본 너비를 150px로 설정합니다. 그러나 `grid-template-columns` 속성을 선언하면 이는 재정의 됩니다.
- 명시적 grid 열은 `grid-template-columns` 속성으로 명시적으로 정의한 열입니다.
- 암시적 grid 열은 브라우저가 자동으로 만드는 열입니다. 우리는 `grid-auto-columns` 속성을 사용하여 암시적 열에 대한 트랙 크기를 지정합니다.
- 반복되는 패턴의 `grid-template-columns` 값을 지정하려면 CSS `repeat()` 함수를 사용합니다.
- CSS `column-gap` 속성을 사용하여 grid 열 사이에 간격을 만듭니다.

grid-template-rows

`grid-template-rows` 는 브라우저가 선택한 grid container의 행(rows)의 수와 높이(heights)를 지정합니다.

Ex1) 3행 Grid Container 만들기

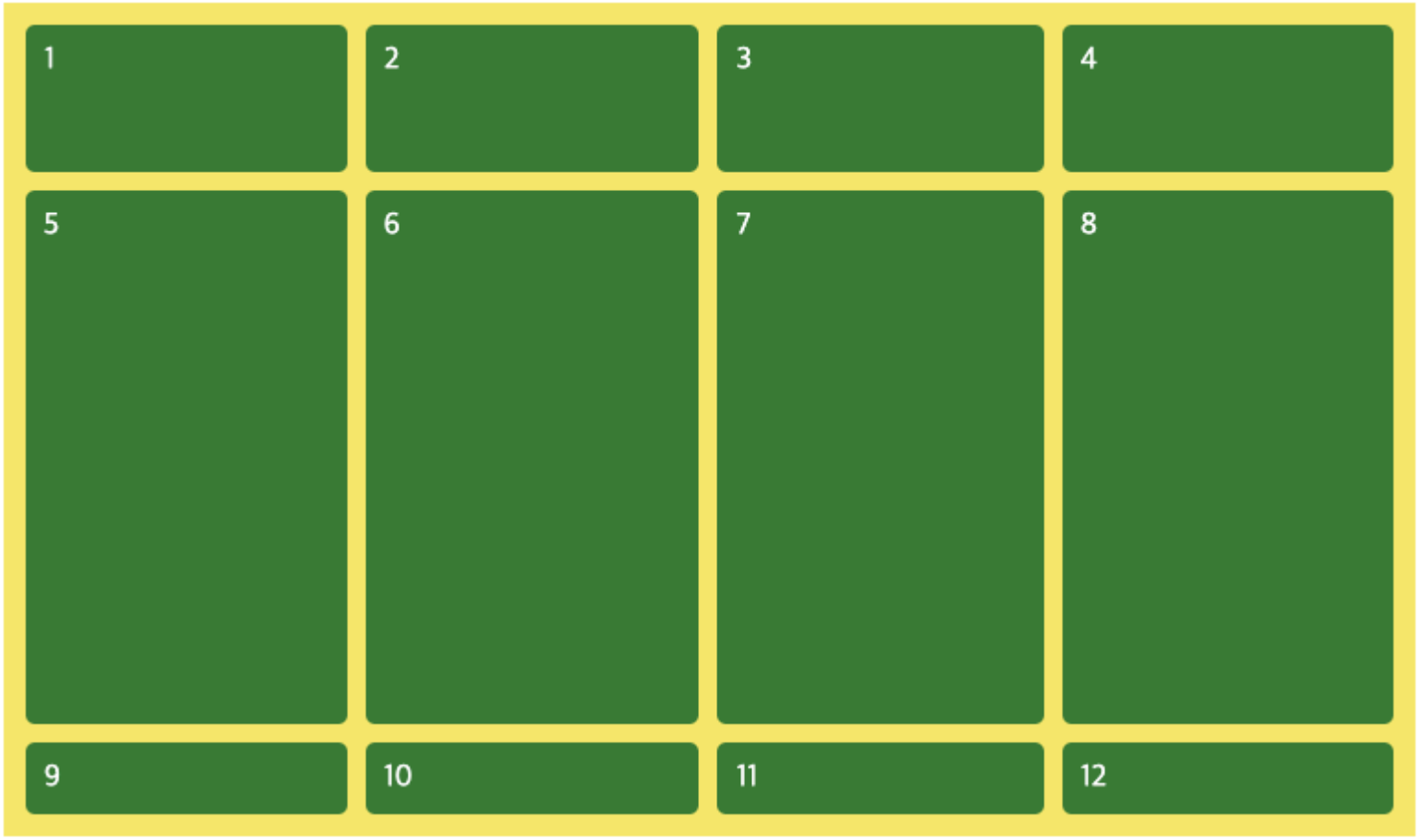
```
section {
  display: grid;
  grid-template-rows: 95px 1fr 70px;
  background-color: #F6E96B;
```

```
  margin: 10px;
  padding: 7px;
}
```

`grid-template-rows` 속성을 사용하여 선택한 `<section>` grid container에 서로 다른 높이의 세 행을 표시했습니다.

Ex2) 3행 4열의 Grid Container 만들기

```
section {
  display: grid;
  grid-template-rows: 90px 300px 1fr;
  grid-template-columns: auto auto auto auto;
  background-color: #F6E96B;
  margin: 10px;
  padding: 7px;
}
```



`<section>` grid container에 `grid-template-rows` 속성을 사용하여 높이가 다른 세 개의 행과 `grid-template-columns` 속성을 사용하여 너비가 같은 4개의 열을 표시했습니다.

- `grid-auto-rows` 속성을 사용하여 그리드 컨테이너의 모든 행에 대한 기본 행 높이를 지정할 수 있습니다. 예를 들어 `grid-auto-rows: 100px`는 모든 행에 대해 기본 높이를 100px로 설정합니다. 그러나 `grid-template-rows` 속성을 선언하면 이는 재정의됩니다.
- 반복되는 패턴의 `grid-template-rows` 값을 지정하려면 CSS `repeat()` 함수를 사용합니다.
- CSS `row-gap` 속성을 사용하여 grid 행 사이에 간격을 만듭니다.

justify-content

`justice-content`는 브라우저가 행 축을 따라 grid container의 열을 배치하는 방법을 지정합니다.

- 행 축은 인라인 축이라고 부르기도 합니다.
- 전체 열 너비가 grid container의 너비보다 작으면 `justify-content` 속성이 작동합니다. 즉, 열을 왼쪽이나 오른쪽으로 정렬하려면 컨테이너의 행 축을 따라 여유 공간이 필요합니다.

`justify-content` 속성 값 :

- `start`
- `center`
- `end`
- `stretch`
- `space-between`
- `space-around`
- `space-evenly`

justify-content: start

start는 행 시작 가장자리를 사용하여 grid container의 열을 배치합니다.


```
section {
  display: grid;
  justify-content: start;
  grid-template-columns: repeat(4, 40px);
  background-color: #F6E96B;
```

```
  margin: 10px;
}
```

justify-content: center

center는 grid container의 열을 grid의 행 축 중앙에 배치합니다.


```
section {
  display: grid;
  justify-content: center;
  grid-template-columns: repeat(4, 40px);
  background-color: #F6E96B;
  margin: 10px;
}
```

justify-content: end

end는 grid container의 열을 행 끝 가장자리에 배치합니다.

```
section {
  display: grid;
  justify-content: end;
  grid-template-columns: repeat(4, 40px);
  background-color: #F6E96B;
  margin: 10px;
}
```

justify-content: space-between

- 행 시작 가장자리를 사용하여 grid container의 첫 번째 열을 배치합니다.
- 행 끝 가장자리에 grid container의 마지막 열을 배치합니다.
- 첫 번째 열과 마지막 열 사이의 균일한 간격으로 나머지 열을 배치합니다.


```
section {
  display: grid;
  justify-content: space-between;
  grid-template-columns: repeat(4, 40px);
  background-color: #F6E96B;
  margin: 10px;
}
```

justify-content: space-around

space-around는 grid container 열의 양 옆에 동일한 간격을 할당합니다.
따라서 첫 번째 열의 앞과 마지막 열의 뒷 공간은 각 열들의 간격 너비의 절반입니다.


```
section {
  display: grid;
  justify-content: space-around;
  grid-template-columns: repeat(4, 40px);
  background-color: #F6E96B;
  margin: 10px;
}
```

justify-content: space-evenly

space-evenly는 grid container의 양쪽 끝과 해당 열 사이에 균일한 간격을 할당합니다.


```
section {
  display: grid;
  justify-content: space-evenly;
  grid-template-columns: repeat(4, 40px);
  background-color: #F6E96B;
```

```
  margin: 10px;
}
```

justify-items

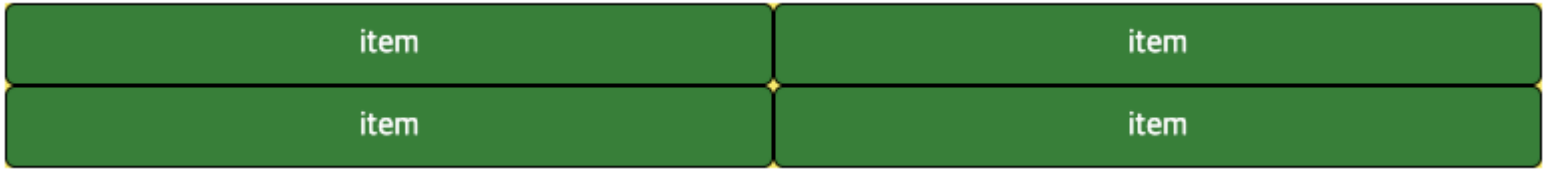
justify-items 속성은 grid item에 대해 기본 `justify-self`를 정의하여 해당 축을 따라 각 item을 배치시키는 기본 방식을 제공합니다.

`justify-items` 속성 값 :

- `stretch`
- `start`
- `center`
- `end`

justify-items: stretch

stretch는 `justify-items` 속성의 기본값입니다. grid container의 item을 늘려 개별 셀의 행(인라인) 축을 채웁니다.




```
section {
  display: grid;
  justify-items: stretch;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
```

```
  margin: 10px;
}
```

justify-items: start

start는 개별 셀 행 축의 행 시작 가장자리에 grid item의 위치를 지정합니다.


```
section {
  display: grid;
  justify-items: start;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
}
```

justify-items: center

center는 grid item을 개별 셀의 행 축 중앙에 배치합니다.

```
section {
  display: grid;
  justify-items: center;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
}
```

justify-items: end

end는 grid item을 개별 셀 행 축의 행 끝 가장자리에 배치합니다.


```
section {
  display: grid;
  justify-items: end;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
}
```

align-content

align-content는 브라우저가 컨테이너의 열 축을 따라 grid container의 행을 정렬하는 방법을 지정합니다.

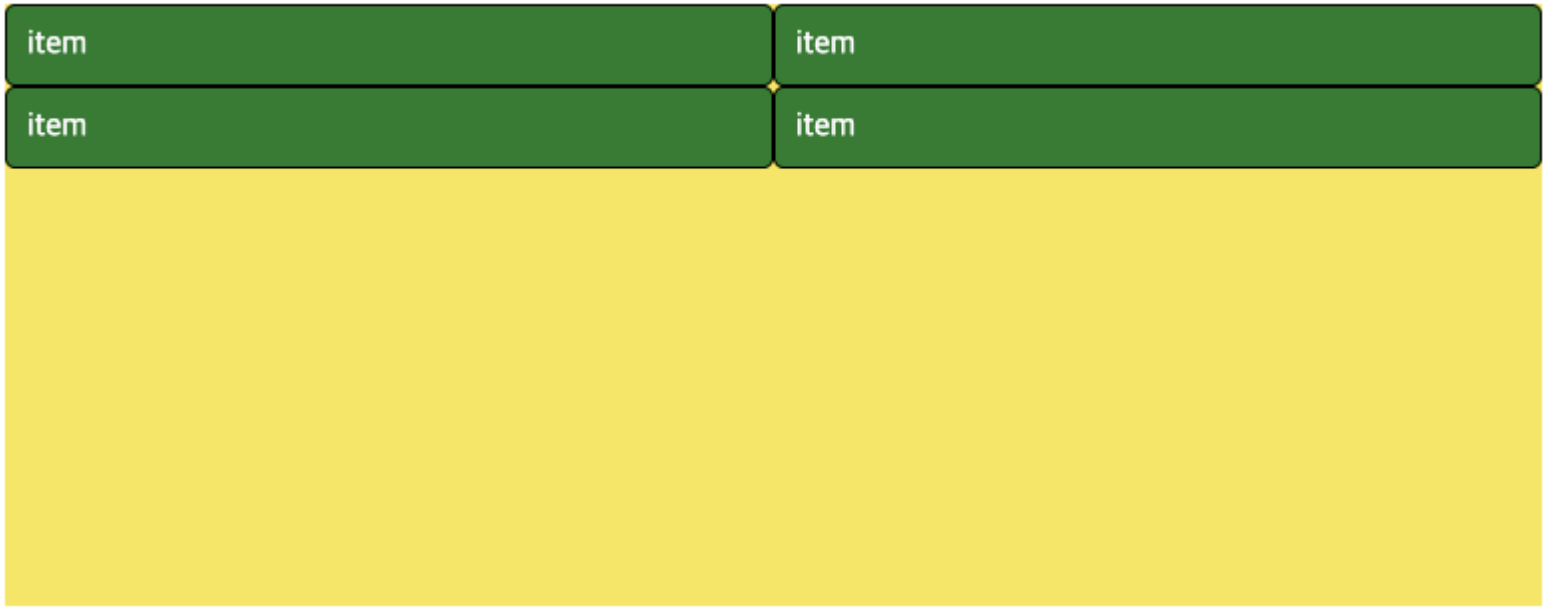
- 열 축은 block 축이라고도 합니다.
- align-content 속성은 총 행 높이가 grid container 높이보다 작은 경우에 작동합니다. 즉, 행을 위나 아래로 정렬하려면 컨테이너의 열 축을 따라 여유 공간이 필요합니다.

`align-content` 속성 값:

- `start`
- `center`
- `end`
- `stretch`
- `space-between`
- `space-around`
- `space-evenly`

align-content: start

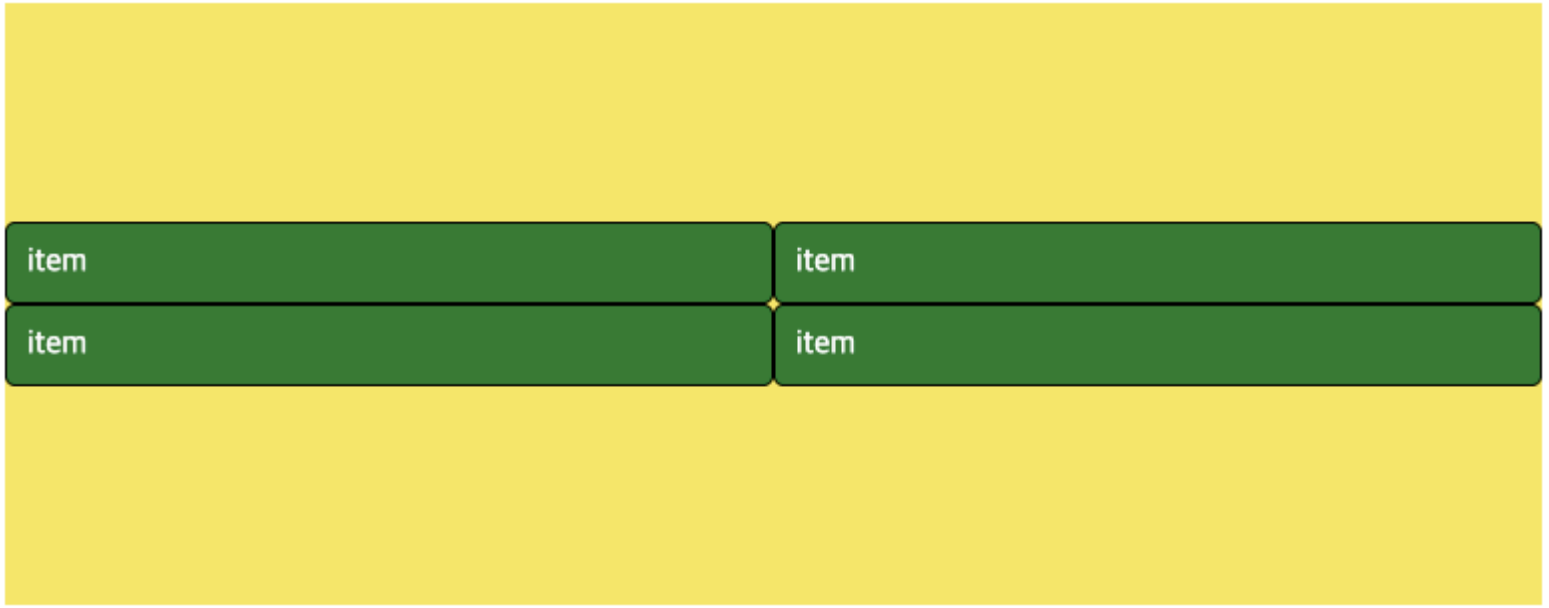
start는 grid 열 축의 열 시작 가장자리에 grid container의 행을 정렬합니다.




```
section {
  display: grid;
  align-content: start;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 300px;
}
```

align-content: center

center는 grid container의 행을 grid의 열 축 중심에 정렬합니다.




```
section {
  display: grid;
  align-content: center;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 300px;
}
```

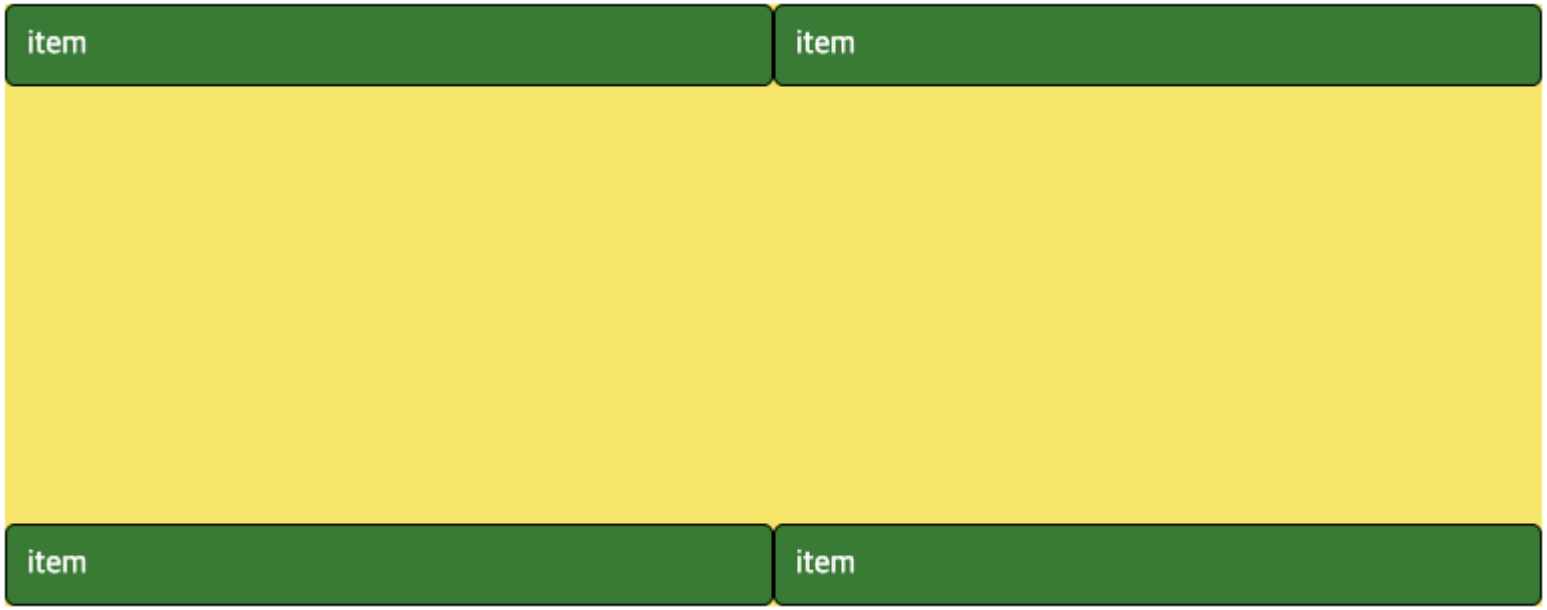
align-content: end

end는 grid의 열 축의 열 끝 가장자리에 grid container의 행을 정렬합니다.


```
section {
  display: grid;
  align-content: end;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 300px;
}
```

align-content: space-between

- grid container의 첫 번째 행을 열 시작 가장자리에 정렬합니다.
- 컨테이너의 마지막 행을 열 끝 가장자리에 맞춥니다.
- 첫 번째 행과 마지막 행 사이의 각 행 쌍 사이에 균일한 간격을 만듭니다.




```
section {
  display: grid;
  align-content: space-between;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 300px;
}
```

align-content: space-between

space-around는 grid container 행의 각 측면에 동일한 간격을 할당합니다.
따라서 첫 번째 행 앞과 마지막 행 뒤의 공간은 각 행 사이 너비의 절반입니다.


```
section {
  display: grid;
  align-content: space-around;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 300px;
}
```

align-content: space-evenly

space-evenly는 grid container의 양쪽 끝과 해당 행 사이에 균일한 간격을 할당합니다.


```
section {
  display: grid;
  align-content: space-evenly;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 300px;
}
```

align-items

align-items는 모든 grid item들에 대한 기본 align-self 값을 지정합니다.

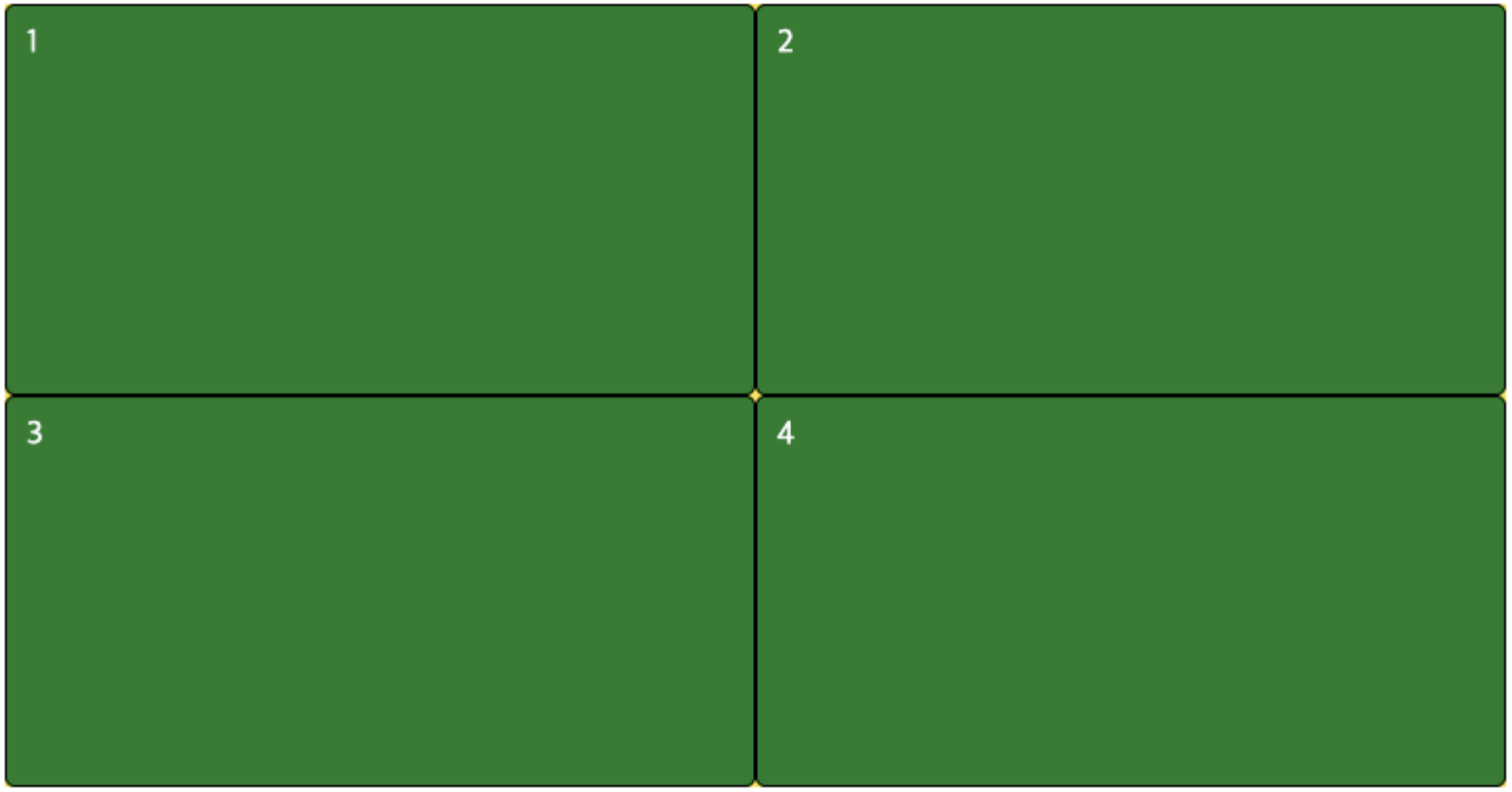
`align-items` 속성 값:

- `stretch`

- `start`
- `center`
- `end`

align-items: stretch

Stretch는 align-items의 기본값입니다. grid container의 item들을 늘려 개별 셀의 열(블록) 축을 채웁니다.




```
section {
  display: grid;
  align-items: stretch;
```

```
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 400px;
}
```

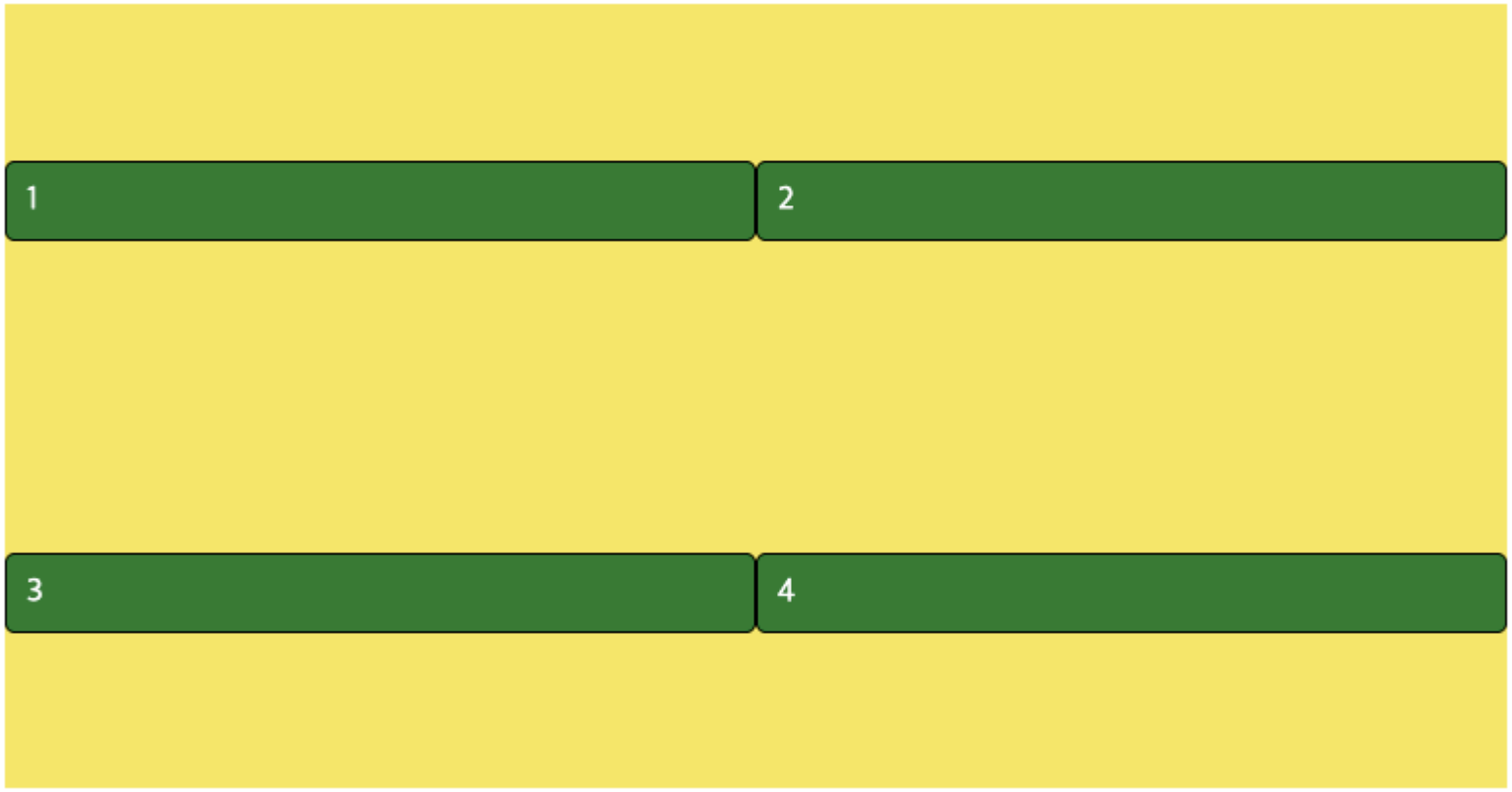
align-items: start

start는 grid container의 item들을 개별 셀의 열 축의 열 시작 가장자리에 정렬합니다.

```
section {
  display: grid;
  align-items: start;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 400px;
}
```

align-items: center

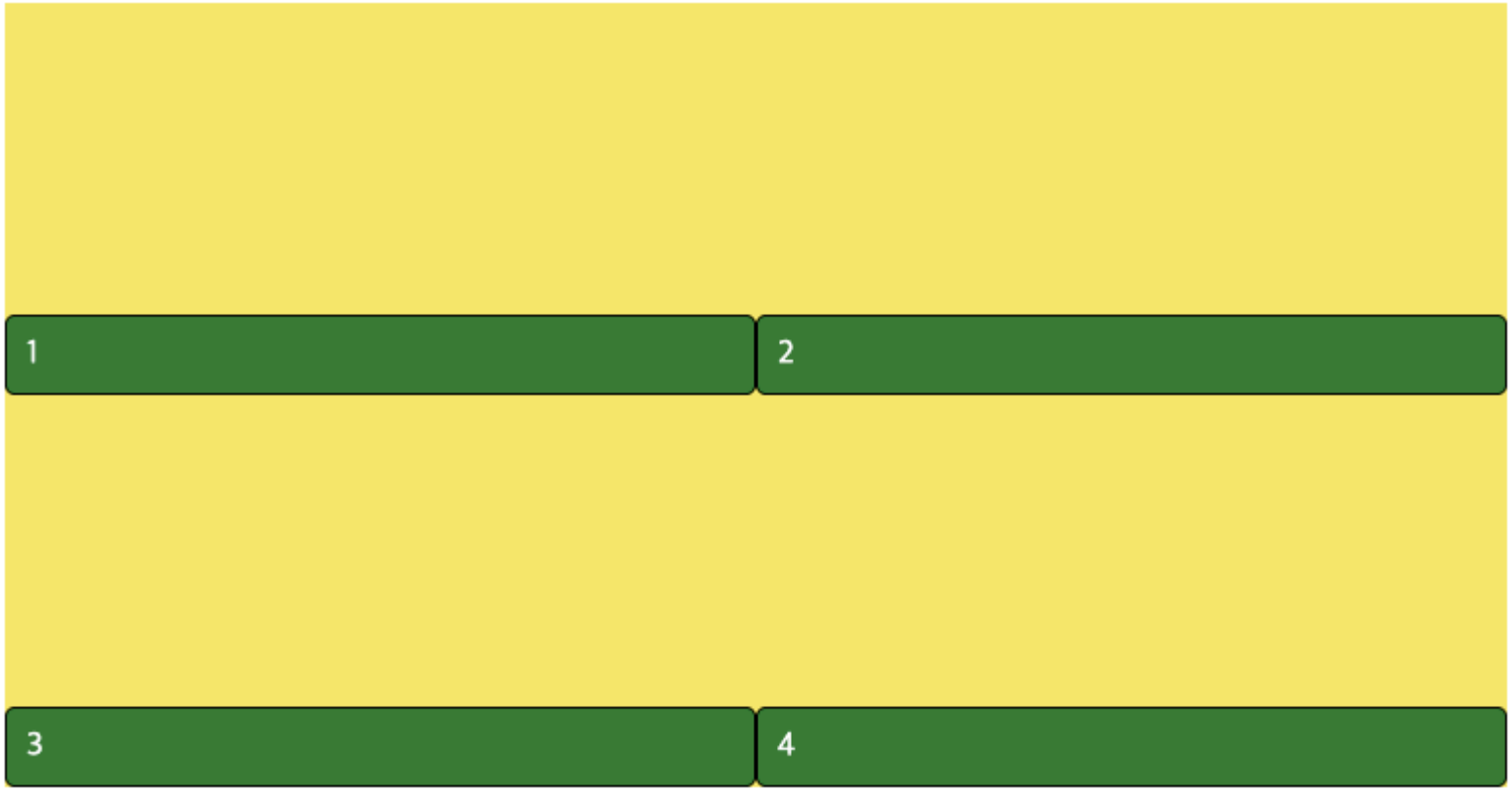
`center`는 grid container의 item들을 개별 셀의 열 축 중앙에 정렬합니다.




```
section {
  display: grid;
  align-items: center;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 400px;
}
```

align-items: end

`end` 는 grid container의 item들을 개별 셀 열 축의 열 끝 가장자리에 정렬합니다.




```
section {
  display: grid;
  align-items: end;
  grid-template-columns: 1fr 1fr;
  background-color: #F6E96B;
  margin: 10px;
  height: 400px;
}
```

Grid Item 속성

grid item의 속성은 브라우저가 grid box model 내에서 지정된 item들을 배치하는 방법을 지정합니다.

참고: container가 아닌 item에 대한 grid item의 속성을 정의합니다.

grid item 속성:

- `justify-self`
- `align-self`
- `grid-column-start`
- `grid-column-end`
- `grid-column`
- `grid-row-start`
- `grid-row-end`
- `grid-row`
- `grid-area`
- `grid-template-areas`

justify-self

justify-self는 브라우저가 셀의 행(inline) 축을 따라 선택한 그리드 아이템을 배치하는 방법을 지정합니다.

justify-self 속성값 :

- `stretch`
- `start`
- `center`
- `end`

justify-self: stretch

Stretch는 justify-self의 기본값입니다. 선택한 그리드 아이템을 늘려 해당 셀의 행(inline) 축을 채웁니다.


```
.grid-item1 {
  justify-self: stretch;
}
```

justify-self: start

start는 선택한 그리드 아이템을 셀 행 축의 행 시작 가장자리에 배치합니다.


```
.grid-item1 {
  justify-self: start;
}
```

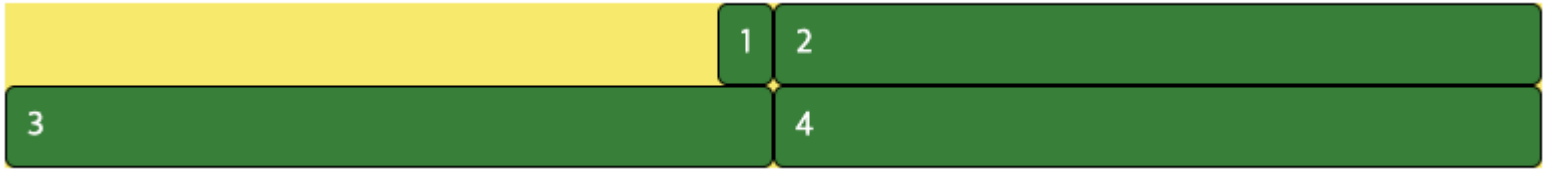
justify-self: center

center는 선택한 그리드 아이템을 셀 행 축의 중간에 배치합니다.


```
.grid-item1 {
  justify-self: center;
}
```

justify-self: end

end는 선택한 그리드 아이템을 셀 행 축의 행의 끝 가장자리에 배치합니다.




```
.grid-item1 {
  justify-self: end;
}
```

align-self

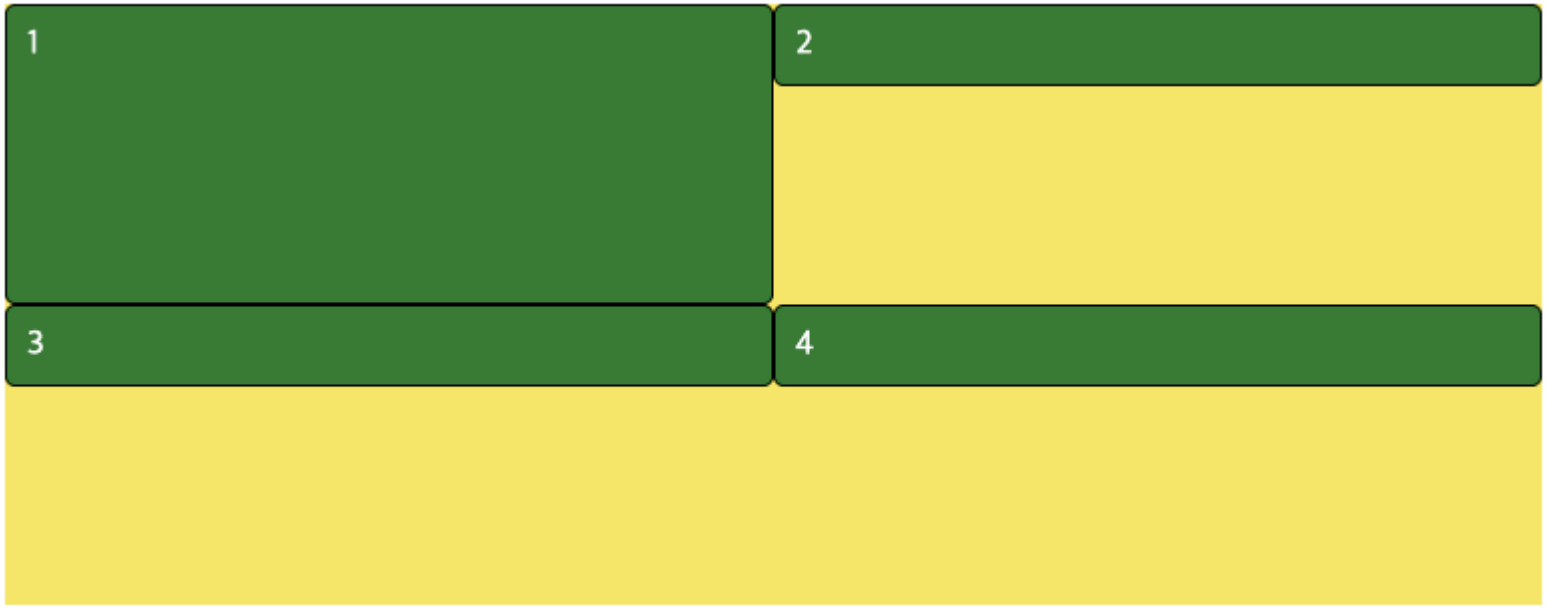
align-self는 브라우저가 셀의 열(block) 축을 따라 선택한 그리드 아이템을 배치하는 방법을 지정합니다.

align-self의 속성 값:

- `stretch`
- `start`
- `center`
- `end`

align-self: stretch

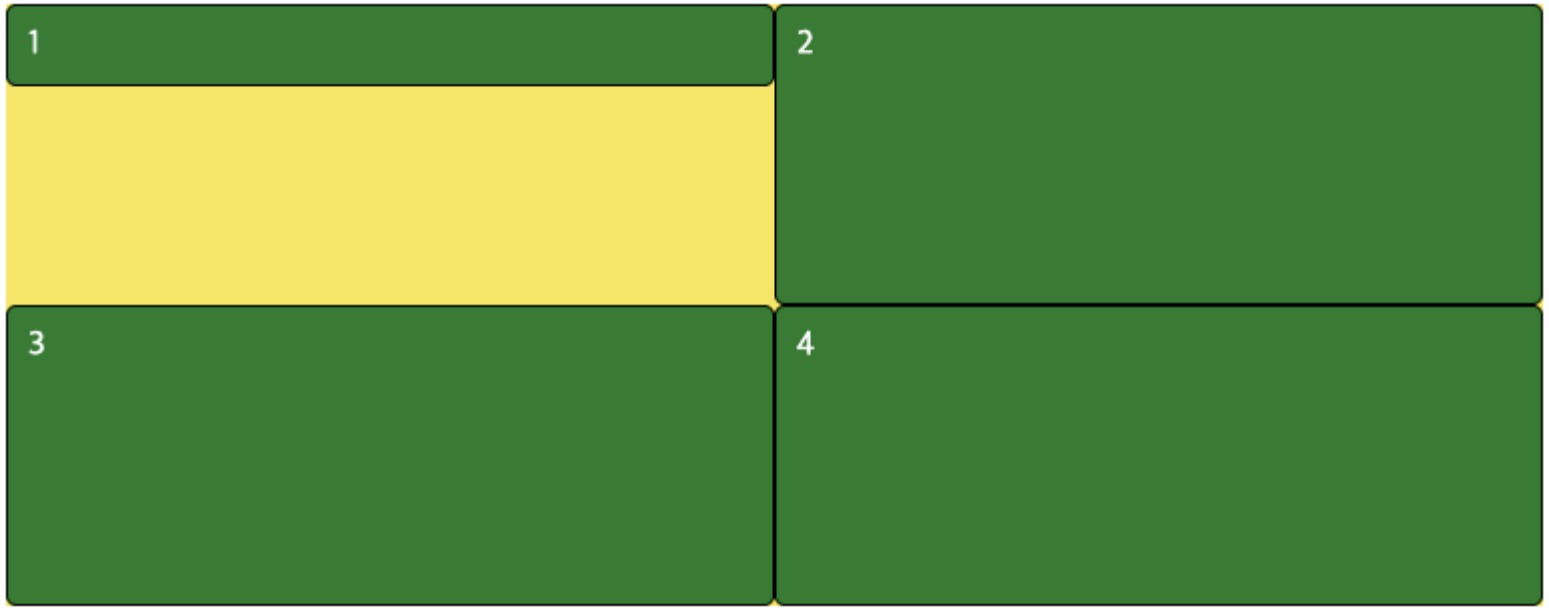
stretch는 align-self의 기본값입니다. 선택한 그리드 아이템을 늘려 해당 셀의 열(block) 축을 채웁니다.




```
.grid-item1 {
  align-self: stretch;
}
```

align-self: start

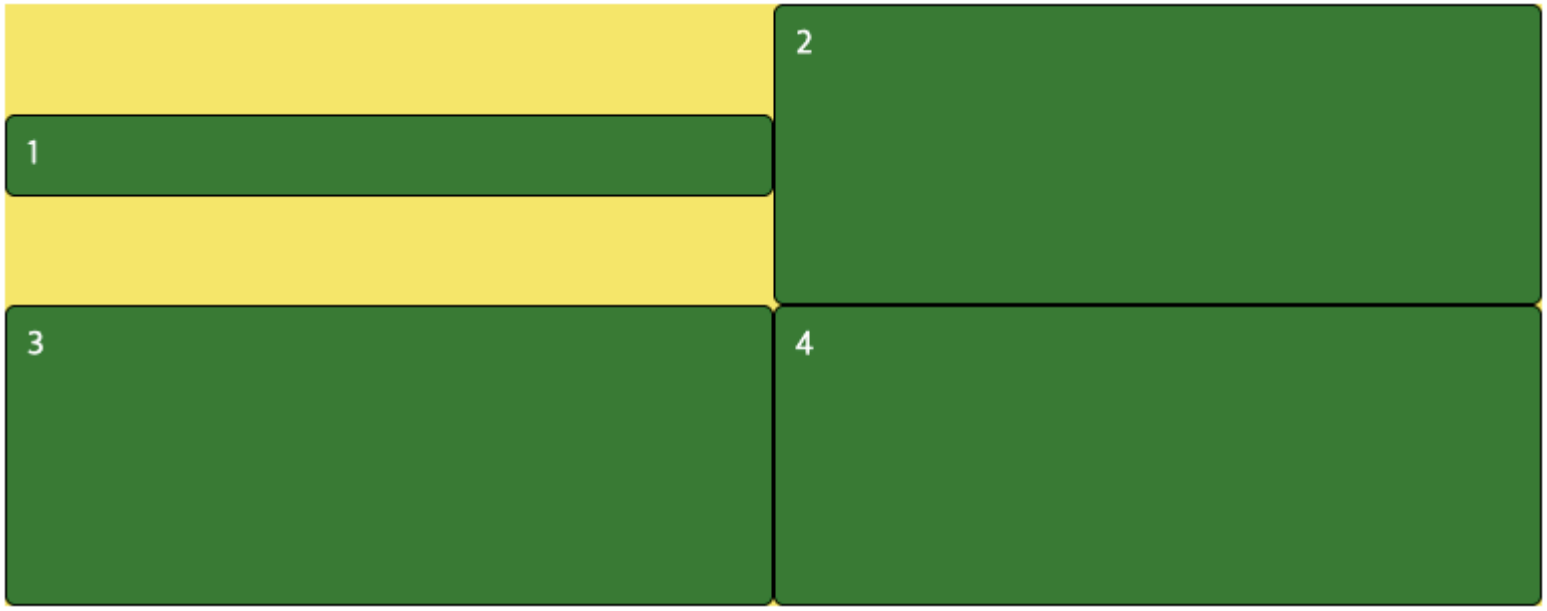
start는 선택한 그리드 아이템을 셀 열 축의 열 시작 가장자리에 배치합니다.




```
.grid-item1 {
  align-self: start;
}
```

align-self: center

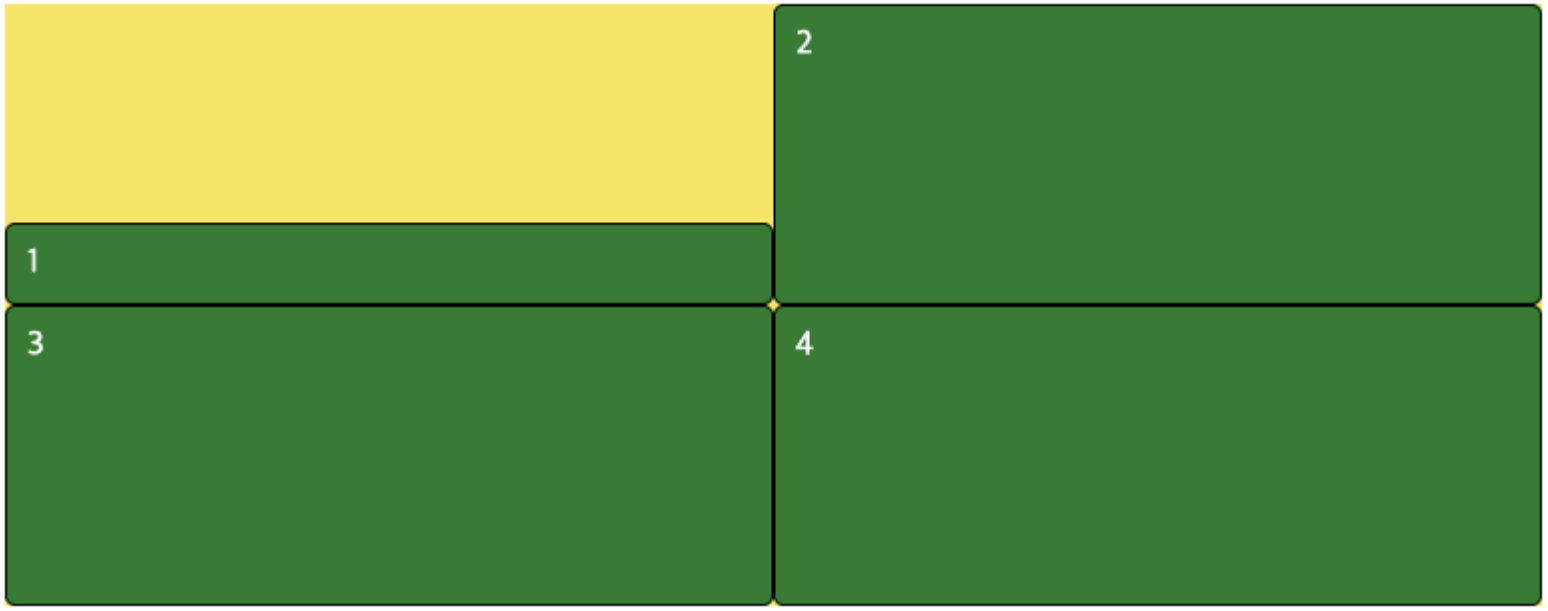
center는 선택한 그리드 아이템을 셀 열 축의 중간에 배치합니다.




```
.grid-item1 {
  align-self: center;
}
```

align-self: end

end는 선택한 그리드 아이템을 셀 열 축의 열의 끝 가장자리에 배치합니다.




```
.grid-item1 {
  align-self: end;
}
```

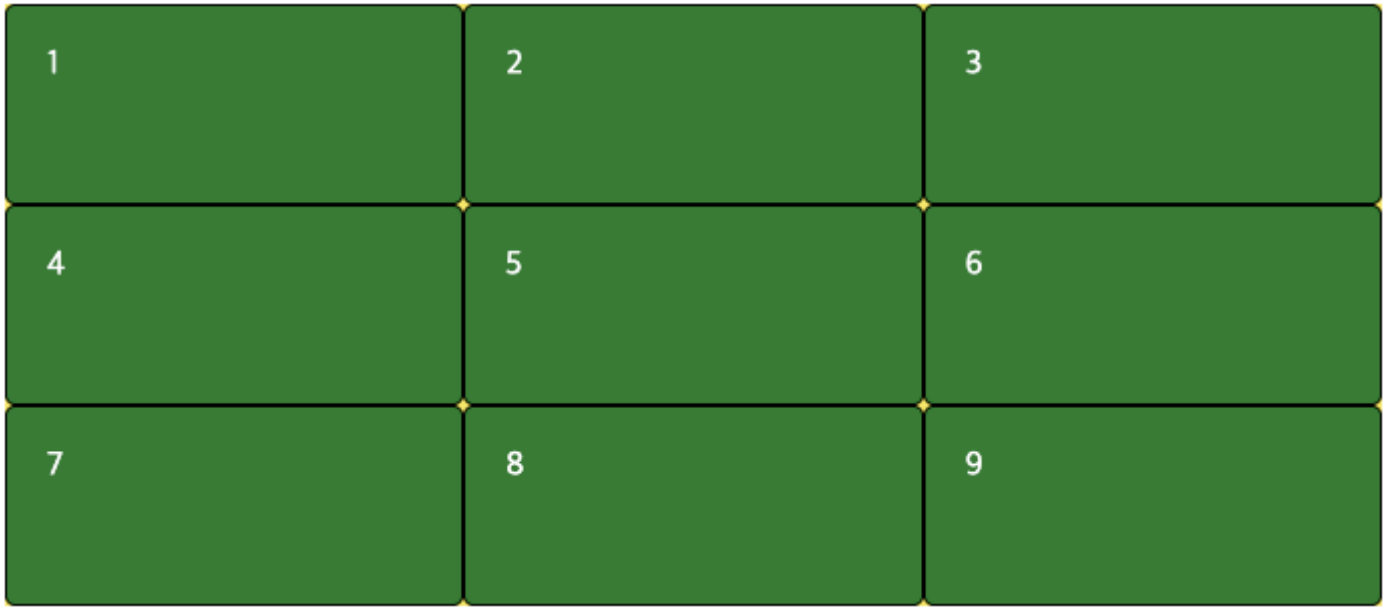
grid-column-start

grid-column-start는 선택한 그리드 아이템이 grid container의 열(block) 축을 따라 시작 또는 확장되어야 하는 위치를 지정합니다.

grid-column-start의 속성 값:

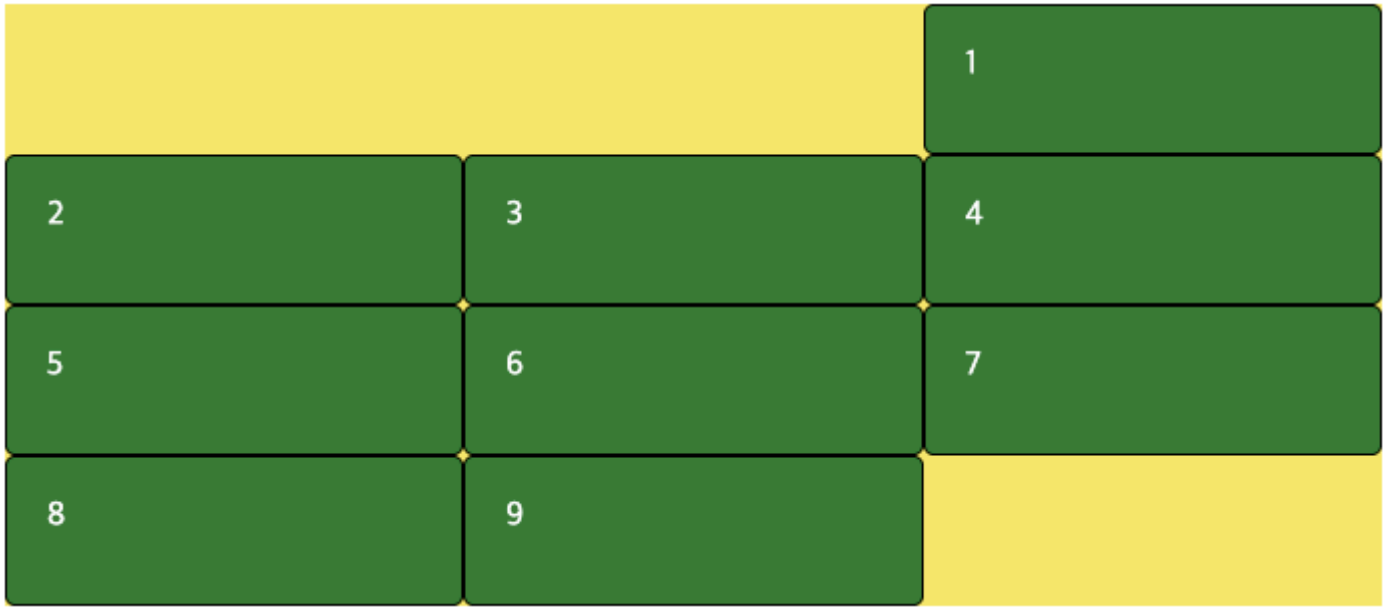
- `auto`
- `<column-line-number>`
- `span <number-of-columns>`

Ex1) 선택한 그리드 아이템을 일반적인 열 흐름에 따라 자동으로 시작되게 하기



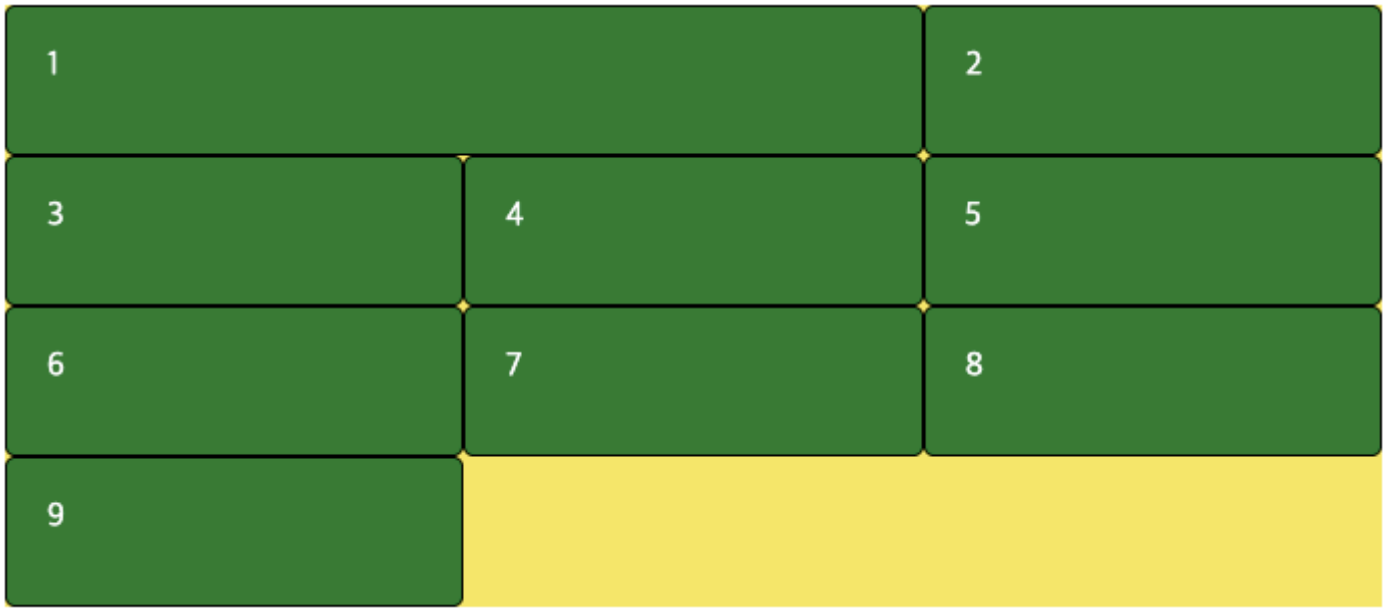

```
.grid-item1 {
  grid-column-start: auto;
}
```

Ex2) 선택한 그리드 아이템을 **3**열에서 시작되게 하기




```
.grid-item1 {
  grid-column-start: 3;
}
```

Ex3) 선택한 그리드 아이템을 2열에 걸쳐 확장하기




```
.grid-item1 {
  grid-column-start: span 2;
}
```

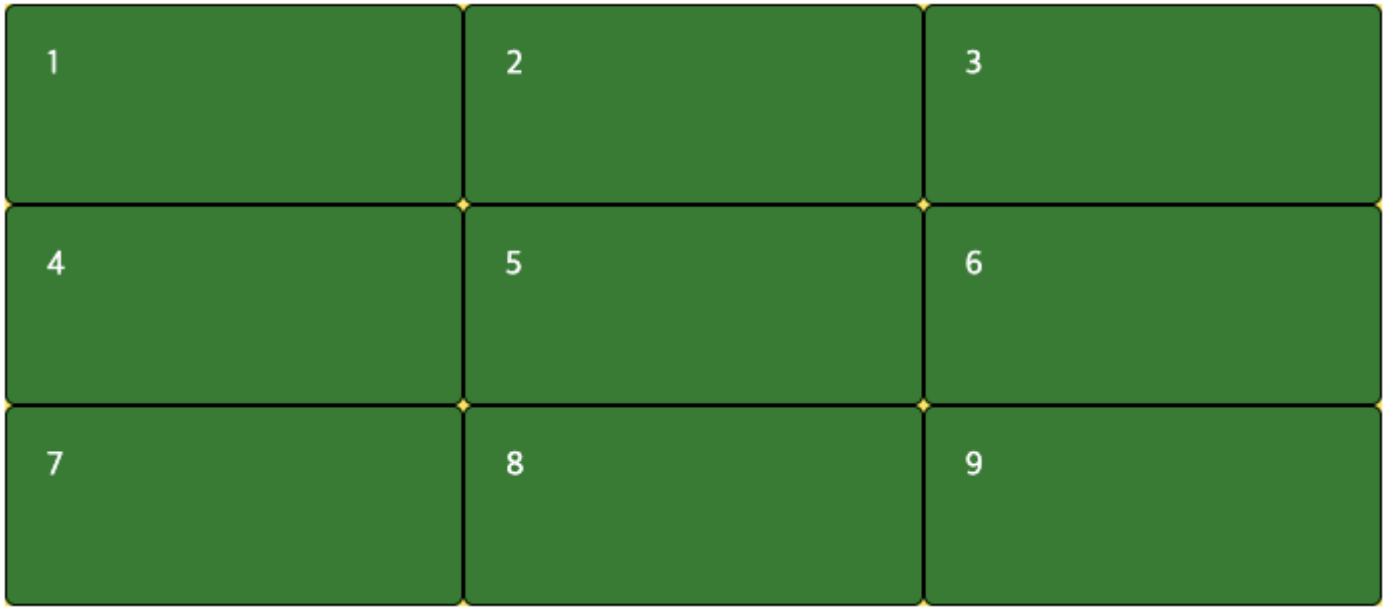
grid-column-end

grid-column-end는 선택한 그리드 아이템이 grid container의 열(block) 축을 따라 끝나거나 확장되어야 하는 위치를 지정합니다.

grid-column-end의 속성 값:

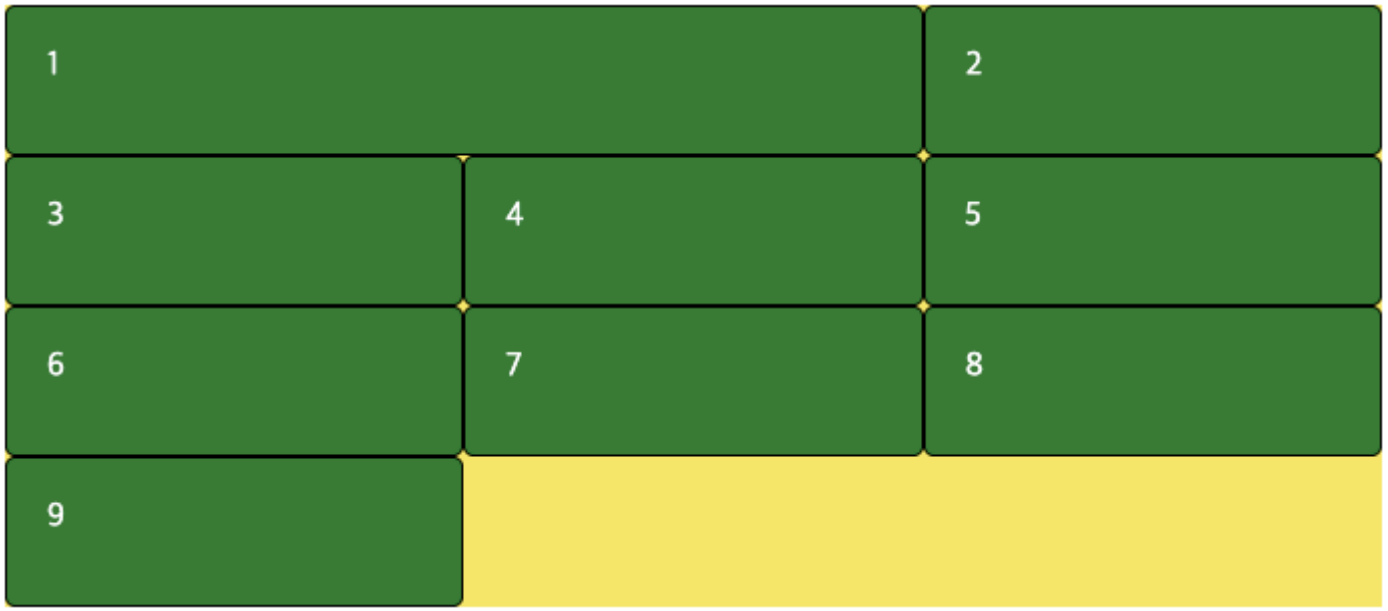
- `auto`
- `<column-line-number>`
- `span <number-of-columns>`

Ex1) 선택한 그리드 아이템을 일반적인 열 흐름에 따라 자동으로 끝에 배치하기



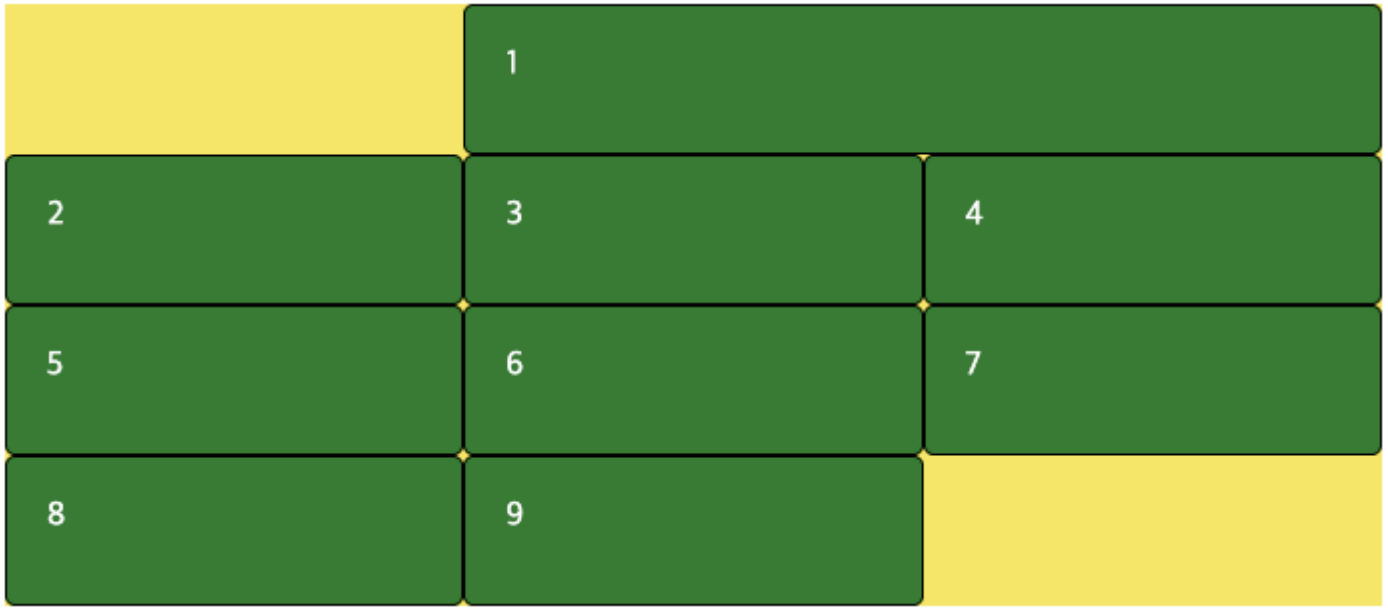

```
.grid-item1 {
  grid-column-end: auto;
}
```

Ex2) 선택한 그리드 아이템을 열의 **3**번째 라인에서 끝나게 하기




```
.grid-item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}
```

Ex3) 선택한 그리드 아이템을 2열에 걸쳐 확장하기




```
.grid-item1 {
  grid-column-start: 2;
  grid-column-end: span 2;
}
```

grid-column

grid-column은 grid-column-start 및 grid-column-end 속성의 약어입니다.

```
grid-column: grid-column-start / grid-column-end;
```

```
.grid-item1 {
  grid-column-start: 1;
  grid-column-end: 3;
```

```
}

.grid-item1 {
  grid-column: 1 / 3;
}
```

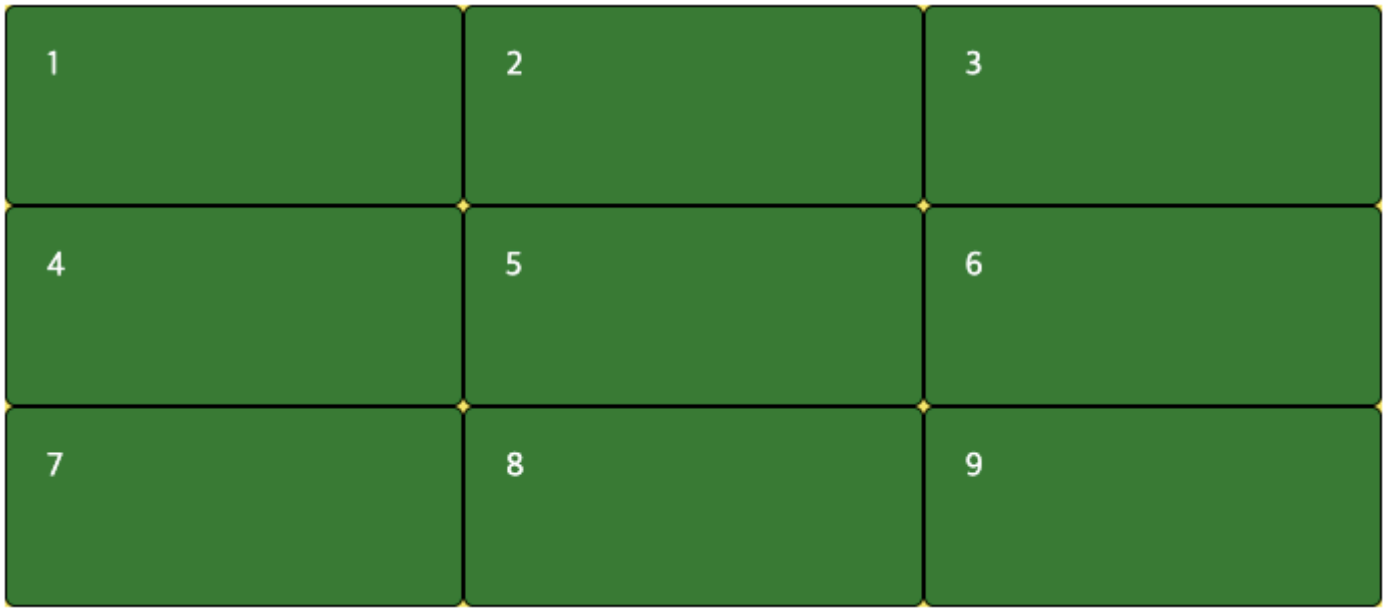
grid-row-start

grid-row-start는 선택한 그리드 아이템이 grid container의 행(inline) 축을 따라 시작 또는 확장되어야 하는 위치를 지정합니다.

grid-row-start의 속성 값:

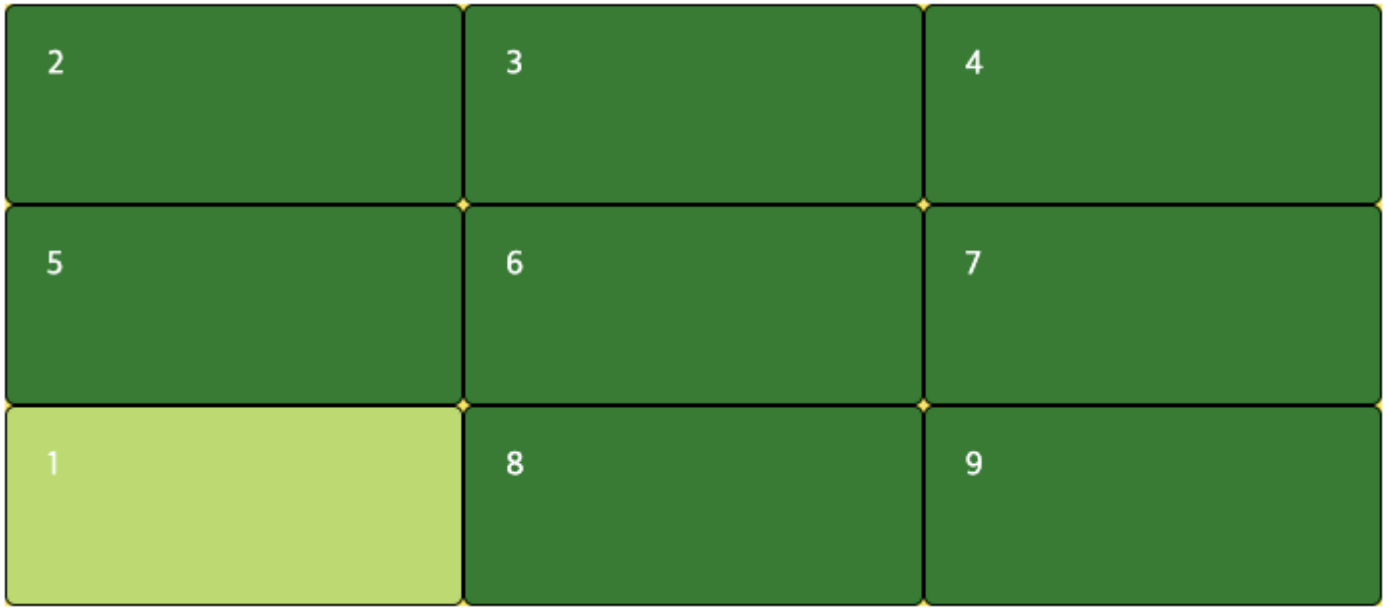
- `auto`
- `<row-line-number>`
- `span <number-of-rows>`

Ex1) 선택한 그리드 아이템을 일반적인 행 흐름에 따라 시작되게 하기



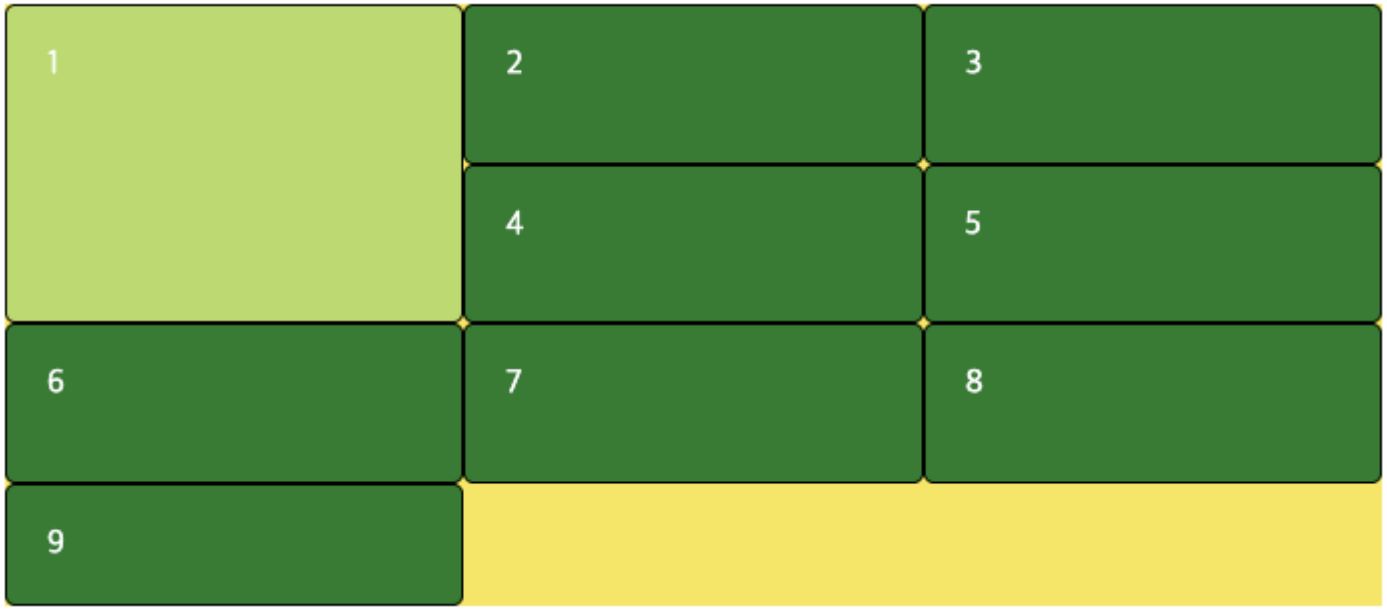

```
.grid-item1 {
  grid-row-start: auto;
}
```

Ex2) 선택한 그리드 아이템을 3행에서 시작되게 하기




```
.grid-item1 {
  grid-row-start: 3;
  background-color: #BEDC74;
}
```

Ex3) 선택한 그리드 아이템을 2행에 걸쳐 확장하기




```
.grid-item1 {
  grid-row-start: span 2;
  background-color: #BEDC74;
}
```

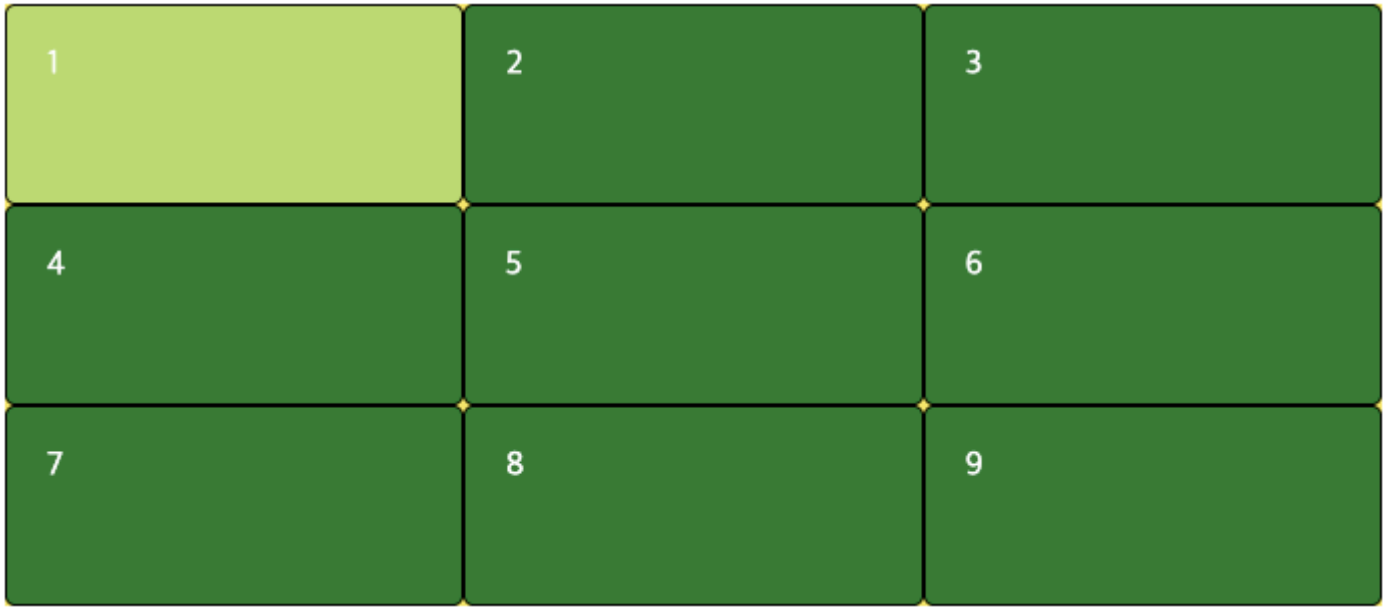
grid-row-end

grid-row-end는 선택한 그리드 아이템이 grid container의 행(inline) 축을 따라 종료 또는 확장되어야 하는 위치를 지정합니다.

grid-row-end의 속성 값:

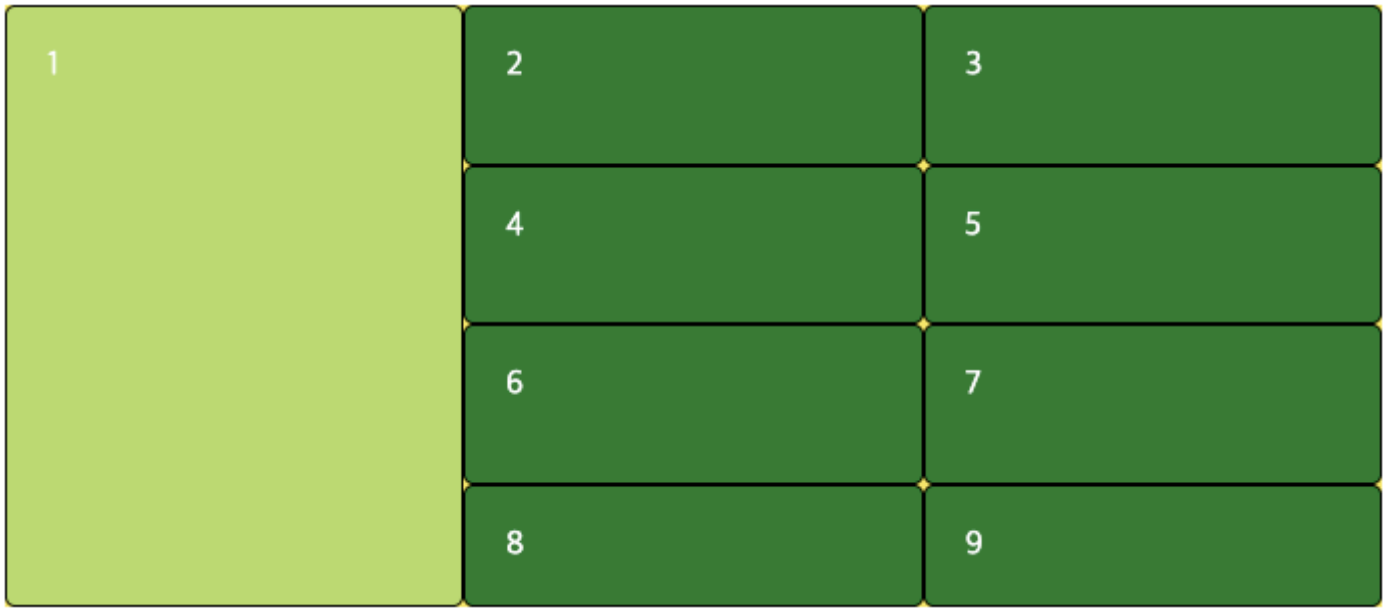
- `auto`
- `<row-line-number>`
- `span <number-of-rows>`

Ex1) 선택한 그리드 아이템을 일반적인 행 흐름에 따라 자동으로 끝에 배치하기



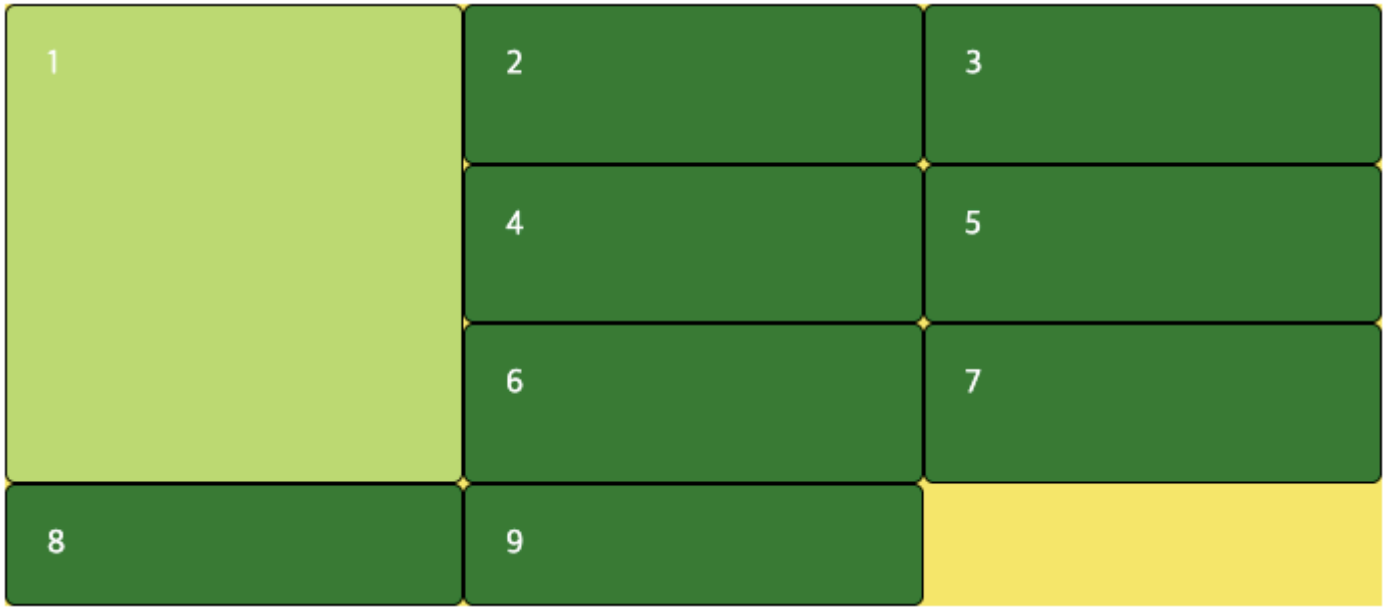

```
.grid-item1 {
  grid-row-end: auto;
  background-color: #BEDC74;
}
```

Ex2) 선택한 그리드 아이템을 행의 **5**번 라인에서 끝나게 하기




```
.grid-item1 {
  grid-row-start: 1;
  grid-row-end: 5;
  background-color: #BEDC74;
}
```

Ex3) 선택한 그리드 아이템을 3행에 걸쳐 확장하기




```
.grid-item1 {
  grid-row-end: span 3;
  background-color: #BEDC74;
}
```

grid-row

grid-row은 grid-row-start 및 grid-row-end 속성의 약어입니다.

```
grid-row: grid-row-start / grid-row-end;

.grid-item1 {
  grid-row-start: 1;
  grid-row-end: 5;
}
```

```
.grid-item1 {
  grid-row: 1 / 5;
}
```

grid-area

grid-area 속성의 사용 목적:

1. `grid-column-start`, `grid-column-end`, `grid-row-start`, `grid-row-end` 속성에 대한 약어입니다.
2. 그리드 아이템의 이름을 지정합니다.

```
.your-grid-item {
  grid-area: grid-row-start / grid-column-start / grid-row-end / grid-column-end;
}
```

```
.grid-item1 {
  grid-row-start: 3;
  grid-row-end: 5;
  grid-column-start: 1;
  grid-column-end: span 2;
}

.grid-item1 {
  grid-area: 3 / 1 / 5 / span 2;
}
```

```
.your-grid-item {
  grid-area: item-name;
```

```
}
```

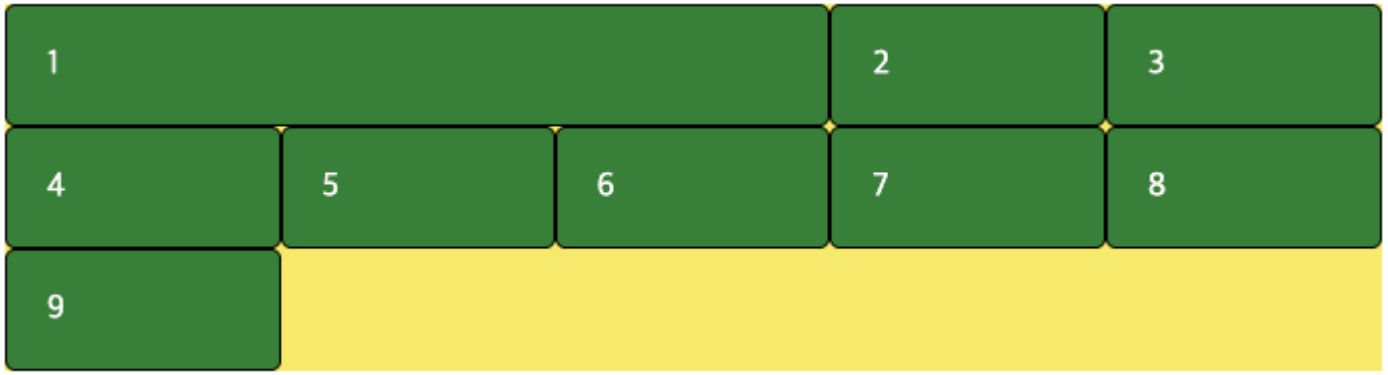
```
.grid-item1 {
  grid-area: firstDiv;
}

.grid-item2 {
  grid-area: middleDiv;
}

.grid-item2 {
  grid-area: lastDiv;
}
```

```
<section>
  <div class="grid-item1">1</div>
  <div class="grid-item2">2</div>
  <div class="grid-item3">3</div>
</section>
```

grid-template-area

grid-template-areas는 grid container 내에 명명된 그리드 item들을 배치할 영역을 지정합니다.
grid item의 이름을 지정하려면 CSS grid-area 속성을 사용합니다.

Ex1) 이름이 지정된 그리드 item을 세 개의 열에 배치하는 방법

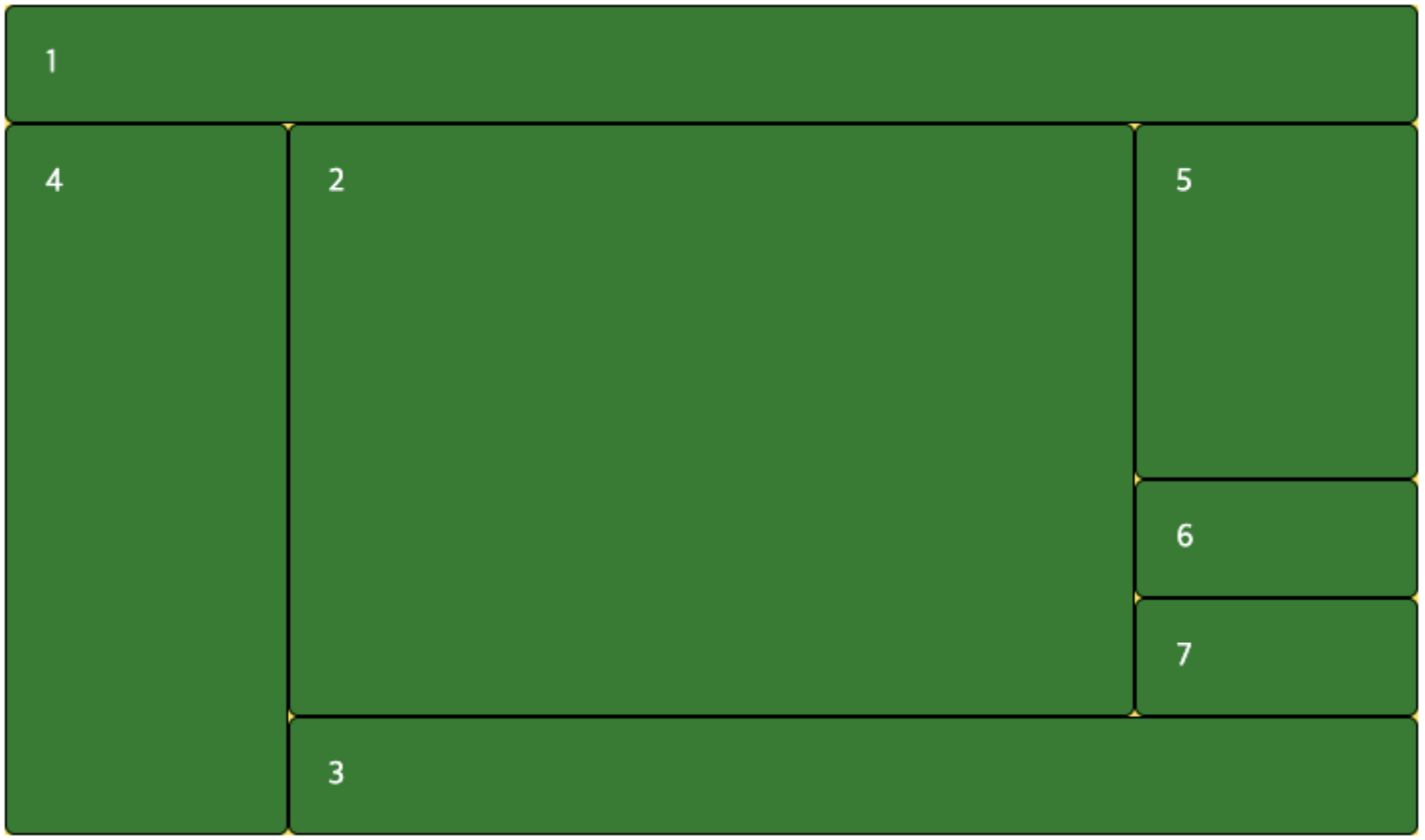



```
.grid-item1 {
  grid-area: firstDiv;
}

section {
  display: grid;
  grid-template-areas: "firstDiv firstDiv firstDiv . .";
  background-color: #f6e96b;
  margin: 50px;
}
```

주의사항:

- 따옴표("")는 각 grid 행을 정의합니다.
- 마침표 기호(.)는 이름이 지정되지 않은 그리드 항목을 정의합니다.
- 공백(whitespace)을 사용하여 그리드 열을 분리합니다.

Ex2) 여러 개의 명명된 그리드 item의 배치를 지정하는 방법




```
.grid-item1 {
  grid-area: header;
}

.grid-item2 {
  grid-area: article;
}

.grid-item3 {
  grid-area: footer;
}
```

```css
.grid-item4 {
  grid-area: sidebar;
}

.grid-item5 {
  grid-area: ads1;
}

.grid-item6 {
  grid-area: ads2;
}

.grid-item7 {
  grid-area: ads3;
}

section {
  display: grid;
  grid-template-columns: repeat(5, 1fr);
  grid-template-rows: repeat(7, 1fr);
  grid-template-areas:
    "header header header header header"
    "sidebar article article article ads1"
    "sidebar article article article ads1"
    "sidebar article article article ads1"
    "sidebar article article article ads2"
    "sidebar article article article ads3"
    "sidebar footer footer footer footer";
  background-color: #f6e96b;
  margin: 30px;
}
```

grid-template-areas 속성에 대해 알아야 할 중요 사항

1. grid-template-areas는 빈 셀을 허용하지 않습니다.

grid-template-areas 속성을 사용하려면 모든 grid 셀에 대한 item을 제공해야 합니다.

```
grid-template-areas:
  "header header"
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads2"
  "sidebar article article article ads3"
  "sidebar footer footer footer footer";
```

위는 첫 번째 행이 불완전하기 때문에 잘못된 grid-template-areas 값입니다.

2. 점을 사용하여 빈 셀을 지정할 수 있습니다

일부 셀을 비워둘 경우 점(.) 또는 간격을 두지 않은 여러 점(...)을 사용합니다.

```
grid-template-areas:
  "header header . . ."
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads2"
  "sidebar article article article ads3"
  "sidebar footer footer footer footer";
```

3. grid-template-areas는 여러 위치에 항목을 배치하는 것을 허용하지 않습니다.

grid-template-areas 속성은 grid container 내에 item을 두 번 배치할 수 없습니다.

```
grid-template-areas:
  "header header header header header"
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads2"
  "sidebar article article article ads3"
  "sidebar footer header header header";
```

위는 header 항목이 두 개의 그리드 영역을 차지하기 때문에 잘못된 grid-template-areas 값입니다.

4. grid-template-areas은 직사각형 영역만 허용합니다.

grid-template-areas 속성은 직사각형이 아닌 영역(예: T자형 또는 L자형)을 만들 수 없습니다.

```
grid-template-areas:
  "header header header header header"
  "sidebar ads1 ads1 ads1 ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads1"
  "sidebar article article article ads2"
  "sidebar article article article ads3"
  "sidebar footer footer footer footer";
```

위는 ads1 항목이 직사각형이 아닌 그리드 영역을 생성하기 때문에 잘못된 grid-template-areas 값입니다.

CSS minmax() 함수를 사용하여 Grid의 최소, 최대 크기를 정의하는 방법

CSS minmax()

`minmax()` 두개의 인수를 받습니다.

```
minmax(minimum-size, maximum-size)
```

- minimum-size는 특정 길이에 대한 최소 크기를 지정합니다.
- maximum-size는 특정 길이에 대한 최대 크기를 지정합니다.
- minmax()의 인수는 음수가 아닌 CSS 길이를 나타내는 단위이거나, auto, min-content, max-content 중 하나일 수 있습니다.
- maximum-size가 minimum-size보다 작을 경우, 브라우저는 maximum-size를 무시하고 minmax() 함수를 min()으로 처리합니다.
- minimum-size에 fr 단위는 사용하지 않습니다.

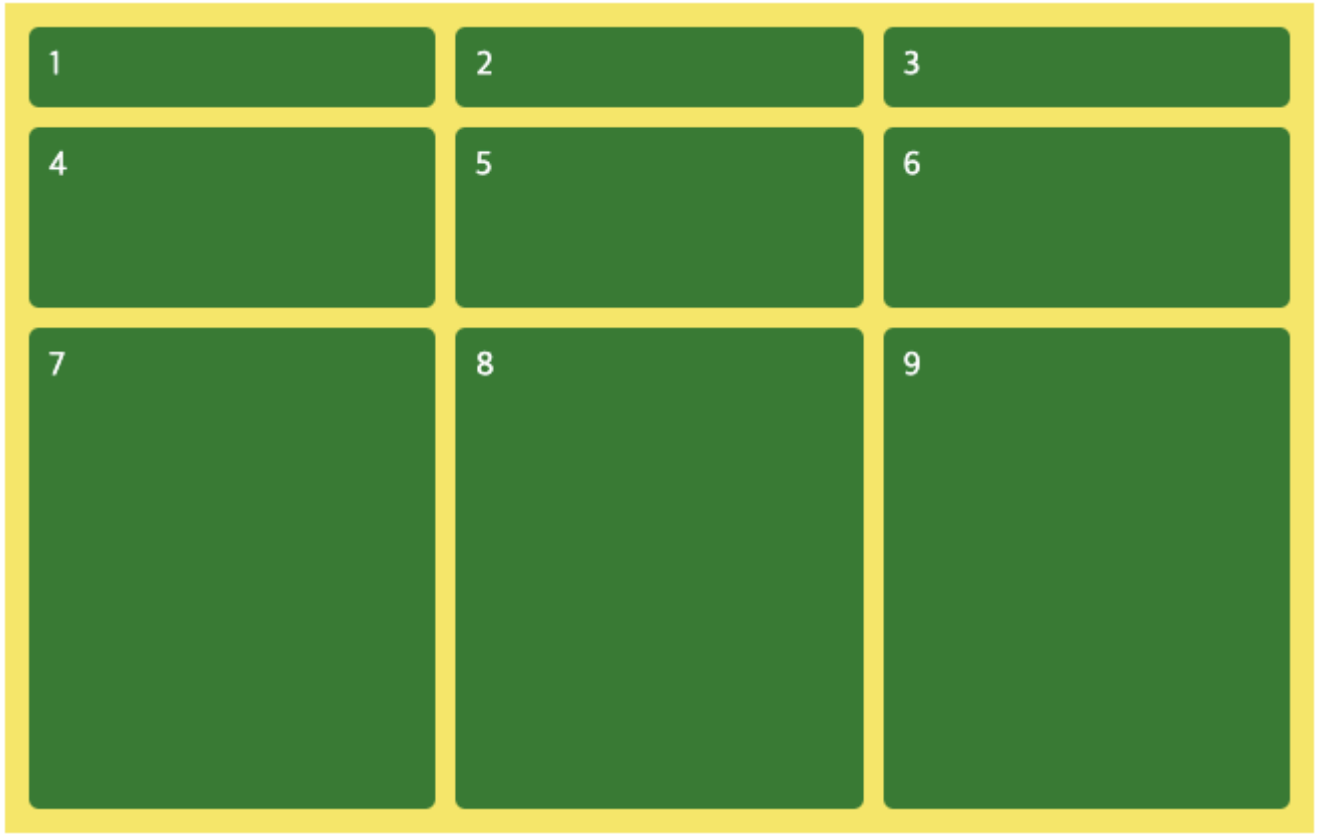
CSS minmax() 사용법

다음 CSS 속성들의 값으로 minmax()를 사용할 수 있습니다:

- `grid-template-columns`
- `grid-template-rows`
- `grid-auto-columns`
- `grid-auto-rows`

CSS minmax() 예제

Ex1) 최소 70px, 최대 250px 높이의 행 그리드 크기를 정의하는 방법




```
section {
  display: grid;
  grid-template-rows: 50px 100px minmax(70px, 250px);
  grid-template-columns: auto auto auto;
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

Ex2) 최소 30%, 최대 70% 너비의 열 그리드 크기를 정의하는 방법


```
section {
  display: grid;
  grid-template-rows: auto auto auto;
  grid-template-columns: 1fr minmax(30%, 70%) 1fr;
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

CSS repeat() 함수를 이용하여 반복되는 패턴의 Grid Track을 정의하는 방법

repeat() CSS 기능을 통해 반복되는 패턴의 grid track을 지정할 때 보다 간결하고 가독성 있는 값을 작성할 수 있습니다.

- Track은 grid container의 열(또는 행)을 나타냅니다.
- repeat()은 grid-template-columns, grid-template-rows의 속성의 값으로 사용할 수 있습니다.

CSS `repeat()`

repeat()은 두개의 인수를 받습니다.

```
repeat(number-of-repetition, track-list-to-repeat)
```

인수1: number-of-repetition

number-of-repetition 인수는 브라우저가 지정한 트랙 목록(두 번째 인수)을 반복하는 횟수를 지정합니다.

아래를 값으로 받을 수 있습니다:

- 1 이상의 정수 값
- `auto-fill`
- `auto-fit`

auto-fill vs auto-fit

auto-fill과 auto-fit은 overflow를 일으키지 않고 grid container를 채우기 위해 필요한 만큼의 track을 자동으로 생성합니다.

두 값의 차이점은 auto-fit은 빈 트랙을 0px로 축소한다는 것입니다. 그러나 auto-fill은 빈 트랙과 채워진 트랙을 모두 표시합니다.

빈 트랙이란 grid item이 없는 열 또는 행입니다.

인수1: track-list-to-repeat

track-list-to-repeat 인수는 grid container의 가로축 또는 세로축에 걸쳐 반복할 트랙 패턴을 지정합니다.
즉, track-list-to-repeat는 grid container 내에서 브라우저가 반복해야 하는 트랙의 크기를 지정하는 하나 이상의 값으로 구성됩니다.

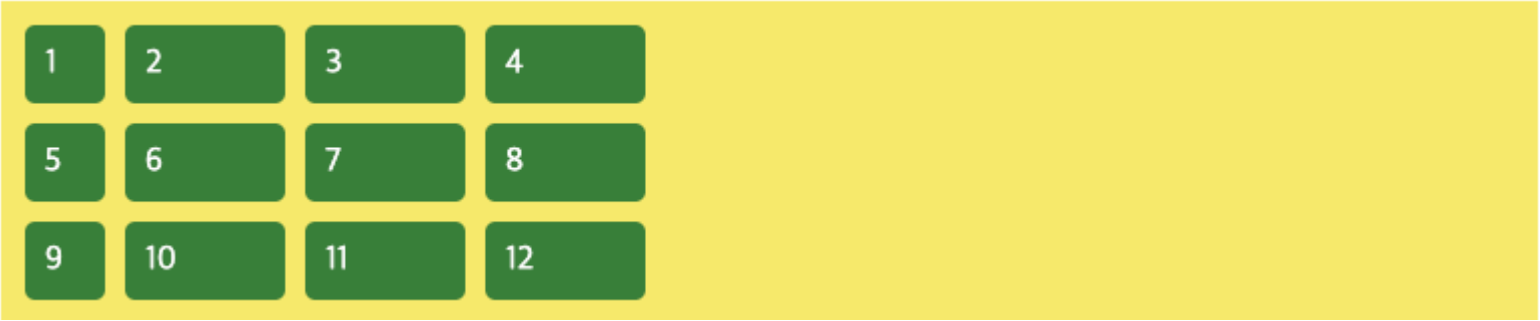
참고: number-of-repetition가 `auto-fill` 또는 `auto-fit`일 경우 track-list-to-repeat 인수로는 고정된 크기만 사용할 수 있습니다.

CSS repeat() 예제

Ex1) 각 열의 너비가 **70px**인 **3**열 **grid container** 만들기

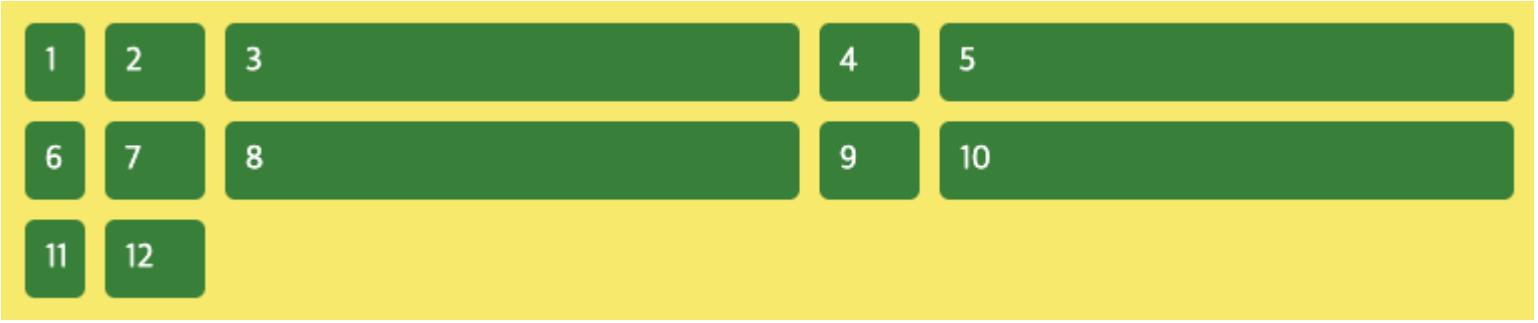

```css
section {
  display: grid;
  grid-template-columns: repeat(3, 70px);
  /* grid-template-columns: 70px 70px 70px; */
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

Ex2) 하나의 열의 너비가 **50px**이고, 나머지 열이 **90px**인 **4**열 **grid container** 만들기




```
section {
  display: grid;
  grid-template-columns: 50px repeat(3, 90px);
  /* grid-template-columns: 50px 90px 90px 90px; */
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

Ex3) 40px 너비의 한개의 열과 두 개의 60px 1fr 열의 너비를 갖는 5열 grid container 만들기

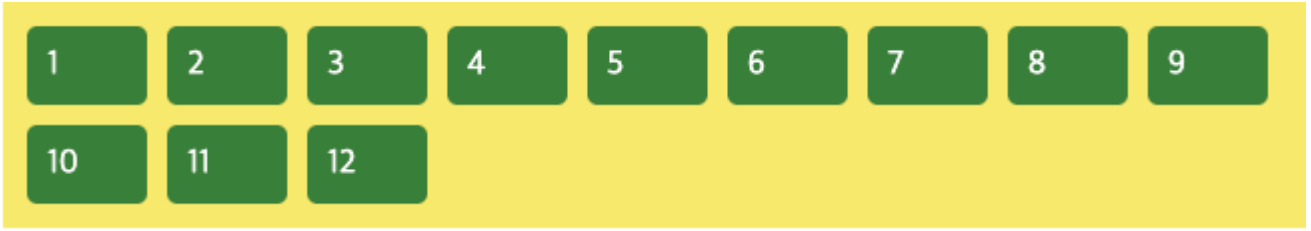



```
section {
  display: grid;
  grid-template-columns: 40px repeat(2, 60px 1fr);
```

```
  /* grid-template-columns: 40px 60px 1fr 60px 1fr; */
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

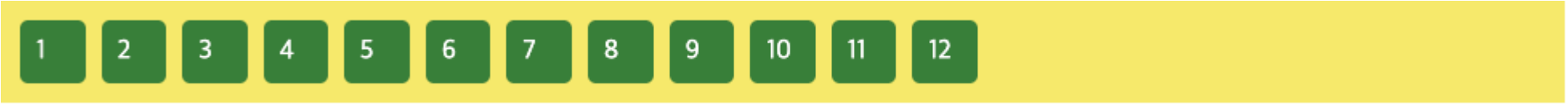
참고: grid container에서 사용 가능한 공간의 비율을 기준으로 세 번째 및 다섯 번째 열을 확장하기 위해 fr 단위를 사용했습니다.

Ex4) grid container에 70px 너비의 열을 자동으로 채우는 방법




```
section {
  display: grid;
  grid-template-columns: repeat(auto-fill, 70px);
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

Ex5) grid container에 최소 50px, 최대 1fr 너비의 열을 자동으로 채우는 방법



```css
section {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(50px, 1fr));
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```

Ex6) grid container에 최소 50px, 최대 1fr 너비의 열을 자동으로 맞추는 방법


```css
section {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(50px, 1fr));
  background-color: #f6e96b;
  margin: 10px;
  padding: 7px;
}
```