Lab 05. CSS (3)

인터넷과웹기초

Contents

- Flexbox
  - Wrapping of items
  - Aligning items
  - Ordering items
  - Controlling ratios of items
  - Use cases
- Grid
  - Grid layout
  - Line-based placement
  - Grid template areas
- Media query

Flexbox

Flexbox

- Basic Concepts
  - Flexbox = Flexible Box module
  - 1차원 레이아웃 모델
  - Flexbox 인터페이스 내의 아이템 간 공간 배분 및 정렬
  - Flex Container (display: flex;) → 점선 박스
  - Flex items (direct children) → a, b, c 박스
  - Flex-direction: row(default), column


```
<style>
    .box {
    display: flex;
    }
</style>

<div class="box">
    <div>A</div>
    <div>B</div>
    <div>C</div>
</div>
```

Flexbox

- Flex-direction
  - Change the direction of flex items


```
.box {
    display: flex;
    flex-direction: row;
}
```


```
.box {
    display: flex;
    flex-direction: row-reverse;
}
```


```
.box {
    display: flex;
    flex-direction: column;
}
```


```
.box {
    display: flex;
    flex-direction: column-reverse;
}
```

Flexbox

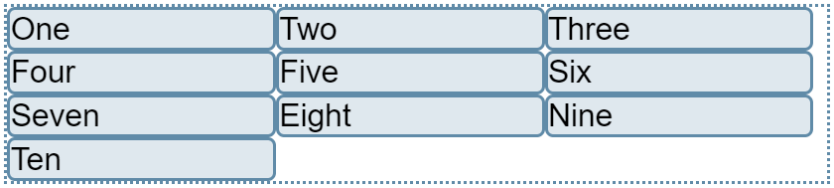
- order
  - this property lay the items out in ordinal groups.
  - The lower the order value, the more it is placed first.
  - value: integer

```
.box {
    display: flex;
    flex-direction: row;
}
.box :nth-child(1) { order: 2; }
.box :nth-child(2) { order: 3; }
.box :nth-child(3) { order: 1; }
.box :nth-child(4) { order: 3; }
.box :nth-child(5) { order: 1; }
```

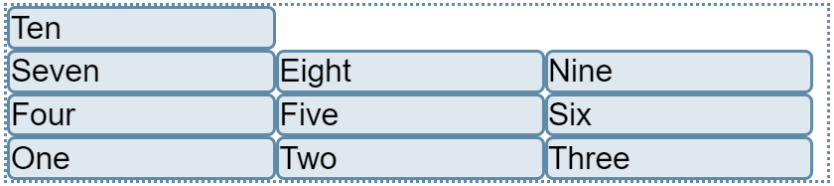
Wrapping of items

- flex-wrap
  - this property sets whether flex items are forced onto one line or can wrap onto multiple lines.

```
flex-wrap: nowrap; /* Default value */
flex-wrap: wrap;
flex-wrap: wrap-reverse;
```




```
.box {
    display: flex;
    flex-wrap: wrap;
}
.box > * {
    width: 160px;
}
```




```
.box {
    display: flex;
    flex-wrap: wrap-reverse;
}
.box > * {
    width: 160px;
}
```

Aligning items

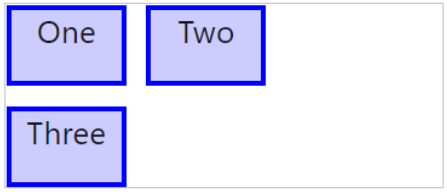
- justify-content
  - this property controls alignment of all items on the **main axis**.

```
/* Positional alignment */
justify-content: center;     /* Pack items around the center */
justify-content: start;      /* Pack items from the start */
justify-content: end;        /* Pack items from the end */
justify-content: flex-start; /* Pack flex items from the start */
justify-content: flex-end;   /* Pack flex items from the end */
justify-content: left;       /* Pack items from the left */
justify-content: right;      /* Pack items from the right */

/* Distributed alignment */
justify-content: space-between; /* Distribute items evenly
                                   The first item is flush with the start,
                                   the last is flush with the end */
justify-content: space-around;  /* Distribute items evenly
                                   Items have a half-size space
                                   on either end */
justify-content: space-evenly;  /* Distribute items evenly
                                   Items have equal space around them */
```

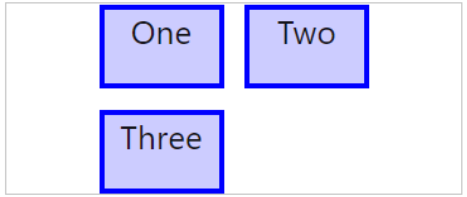
Aligning items

- Justify-content
  - examples

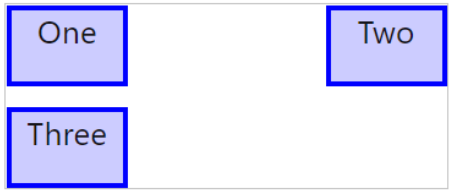
```
.box {
    display: flex;
    justify-content: start;
}
```



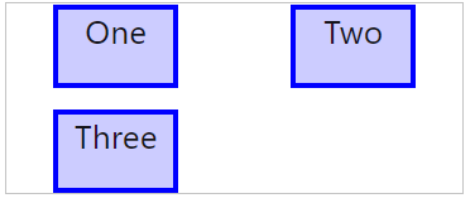

```
.box {
    display: flex;
    justify-content: center;
}
```




```
.box {
    display: flex;
    justify-content: space-between;
}
```




```
.box {
    display: flex;
    justify-content: space-around;
}
```



Aligning items

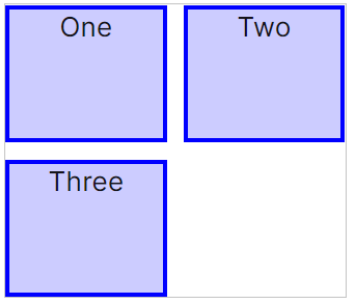
- align-items
  - this property controls alignment of all items on the **cross axis**.

```
/* Basic keywords */
align-items: normal;
align-items: stretch;

/* Positional alignment */
/* align-items does not take left and right values */
align-items: center; /* Pack items around the center */
align-items: start; /* Pack items from the start */
align-items: end; /* Pack items from the end */
align-items: flex-start; /* Pack flex items from the start */
align-items: flex-end; /* Pack flex items from the end */

/* Global values */
align-items: inherit;
align-items: initial;
align-items: revert;
align-items: unset;
```

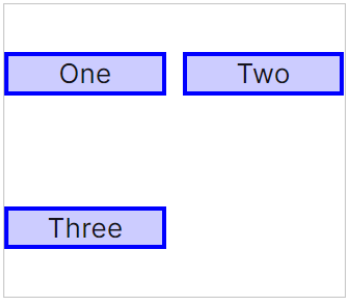
Aligning items

- align-items
  - examples

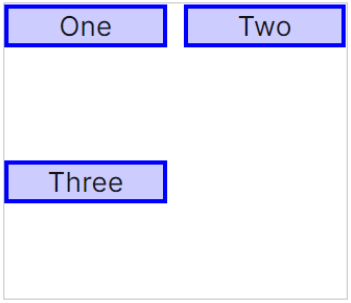
```
.box {
    display: flex;
    align-items: stretch;
}
```



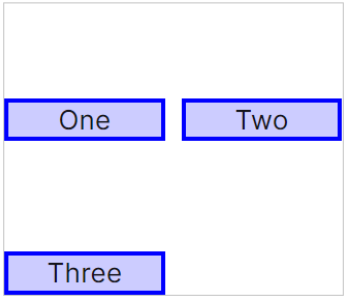

```
.box {
    display: flex;
    align-items: center;
}
```




```
.box {
    display: flex;
    align-items: start;
}
```



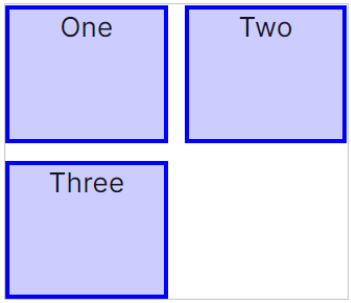

```
.box {
    display: flex;
    align-items: end;
}
```



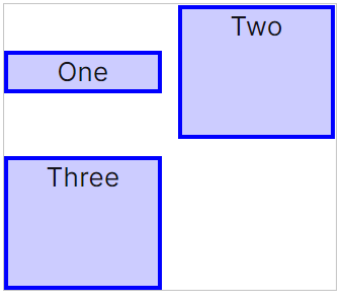
Aligning items

- align-self
  - align-items 와 동일하지만, 아이템 하나에 대해서 적용시킴

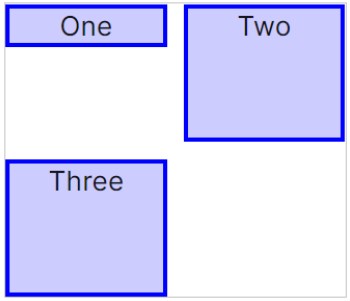
```
#One {
    align-items: stretch;
}
```



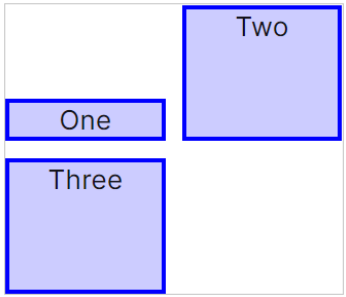

```
#One {
    align-items: center;
}
```




```
#One {
    align-items: start;
}
```



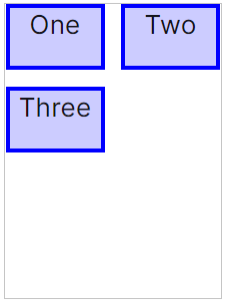

```
#One {
    align-items: end;
}
```



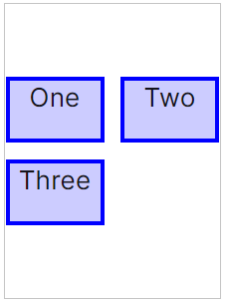
Aligning items

- align-content
  - this property sets the space between content items along **cross axis**.

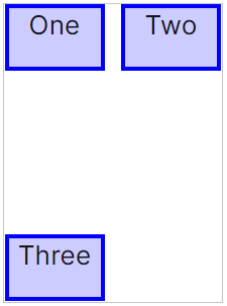
```
.box {
    display: flex;
    align-content: start;
}
```



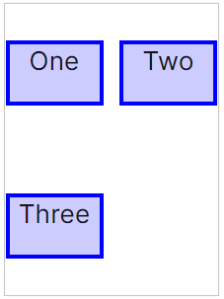

```
.box {
    display: flex;
    align-content: center;
}
```




```
.box {
    display: flex;
    align-content: space-between;
}
```



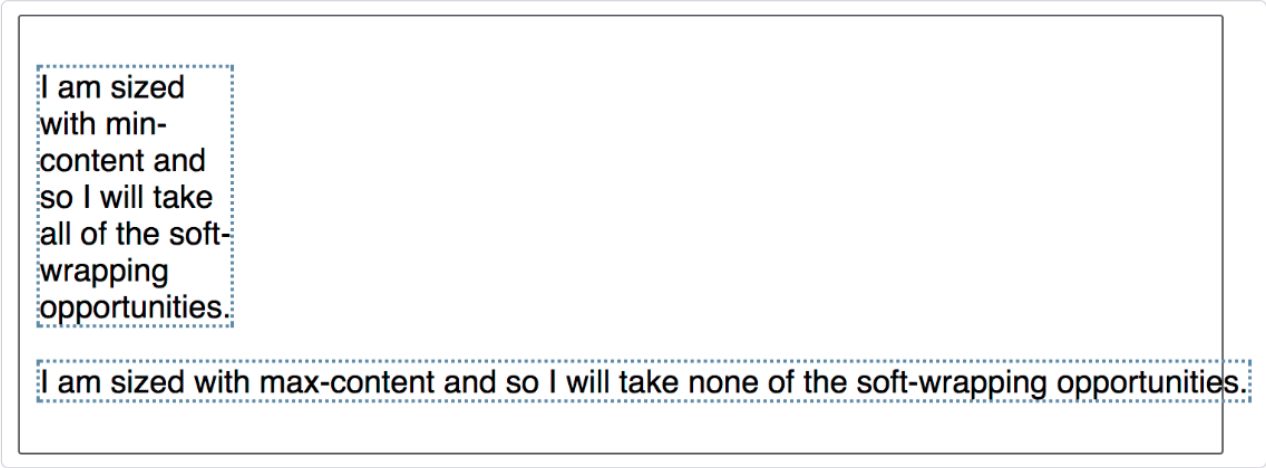

```
.box {
    display: flex;
    align-content: space-around;
}
```



Controlling ratio of items

- min-content,
  - the longest word in the string is dictating the size.
- max-content
  - It gets as big as it possibly can be

```
#min-content {
    width: min-content;
}
#max-content {
    width: max-content;
}
```



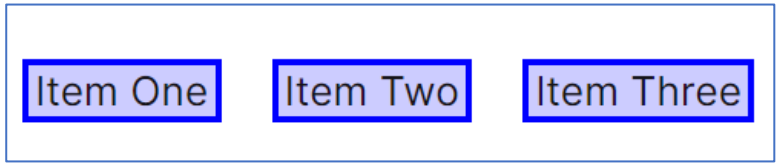
Controlling ratio of items

- flex-basis
  - this property specifies the initial size of the **flex item** before any space distribution happens.

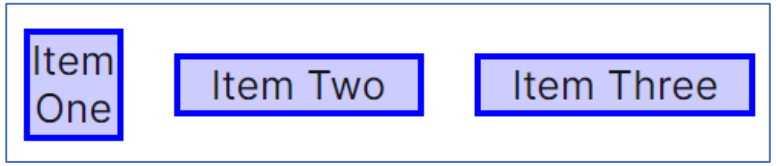
```
/* Specify <'width'> */
flex-basis: 10em;
flex-basis: 3px;
flex-basis: auto; /* initial value*/

/* Intrinsic sizing keywords */
flex-basis: fill;
flex-basis: max-content;
flex-basis: min-content;
flex-basis: fit-content;

/* Automatically size based on the flex item's
content */
flex-basis: content;
```

Controlling ratio of items

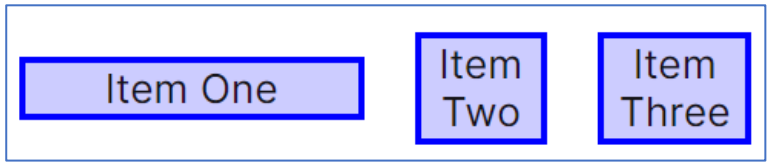
- flex-basis
  - examples




```
#One {
    flex-basis: auto;
}
```




```
#One {
    flex-basis: 0;
}
```




```
#One {
    flex-basis: 200px;
}
```

Controlling ratio of items

- flex-grow
  - this property specifies **the flex grow factor**, which determines how much the flex item will grow relative to the rest of the flex items in the flex container when the positive free space is distributed.

Controlling ratio of items

- flex-grow
  - examples

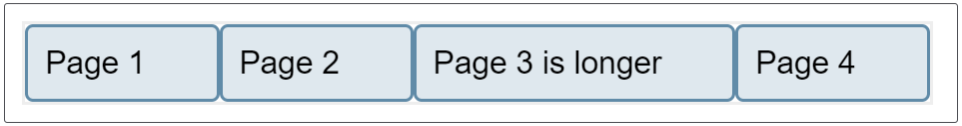
```
.box {
    display: flex;
}
```


```
.box {
    display: flex;
}

.box > * {
    flex: 1 1 0;
}
```

Use cases

- Navigation

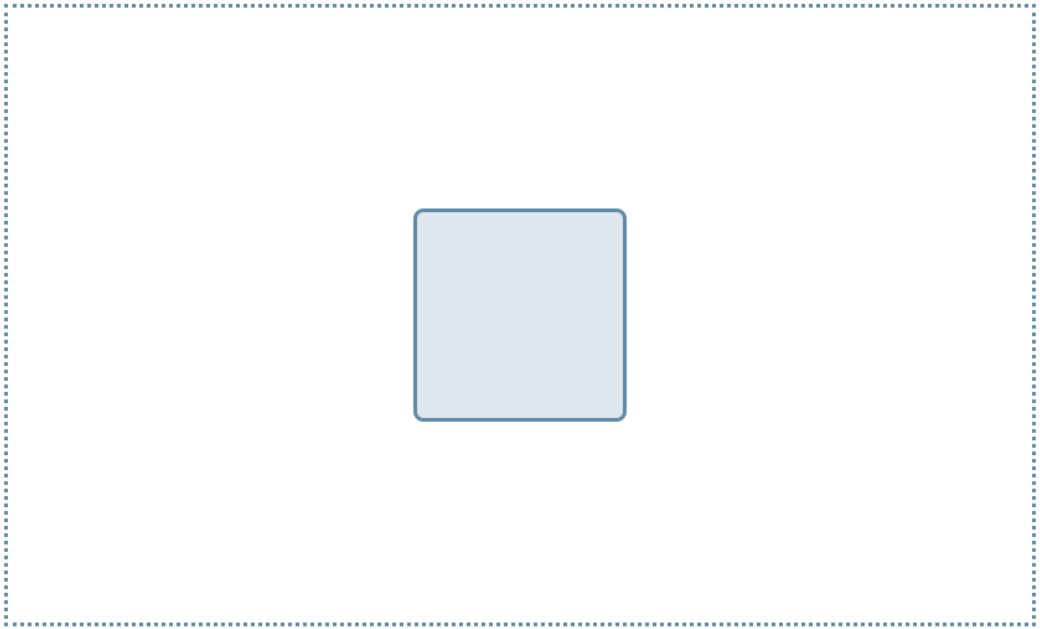

```
nav ul {
  display: flex;
  justify-content: space-between;
}
<nav>
    <ul>
        <li><a href="#">Page 1</a></li>
        <li><a href="#">Page 2</a></li>
        <li><a href="#">Page 3 is longer</a></li>
        <li><a href="#">Page 4</a></li>
    </ul>
</nav>
```




```
nav ul {
  display: flex;
}
nav li {
  flex: auto ;
}
```

Use cases

- Center item



```
.box {
  display: flex;
  align-items: center;
  justify-content: center;
}

.box div {
  width: 100px;
  height: 100px;
}
```

```
<div class="box">
  <div></div>
</div>
```

Use cases

- Form control


```
.wrapper {
  display: flex;

}
.wrapper input[type="text"] {
    flex: 1 1 auto;
}
```

```
<form class="example">
    <div class="wrapper"><input type="text"
id="text"><input type="submit" value="Send"></div>
</form>
```

flex: 증가너비 감소너비 기본너비;

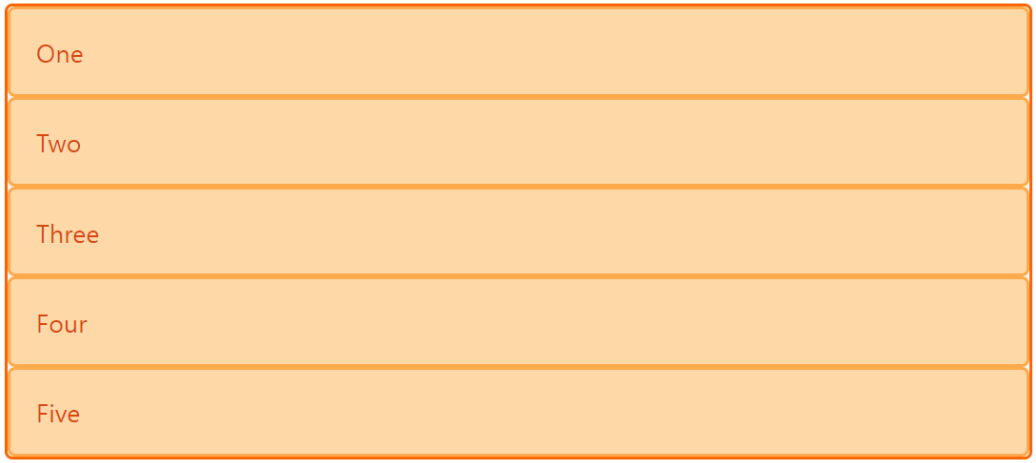
Grid

Grid

- Basic concepts
  - 2차원 레이아웃 모델

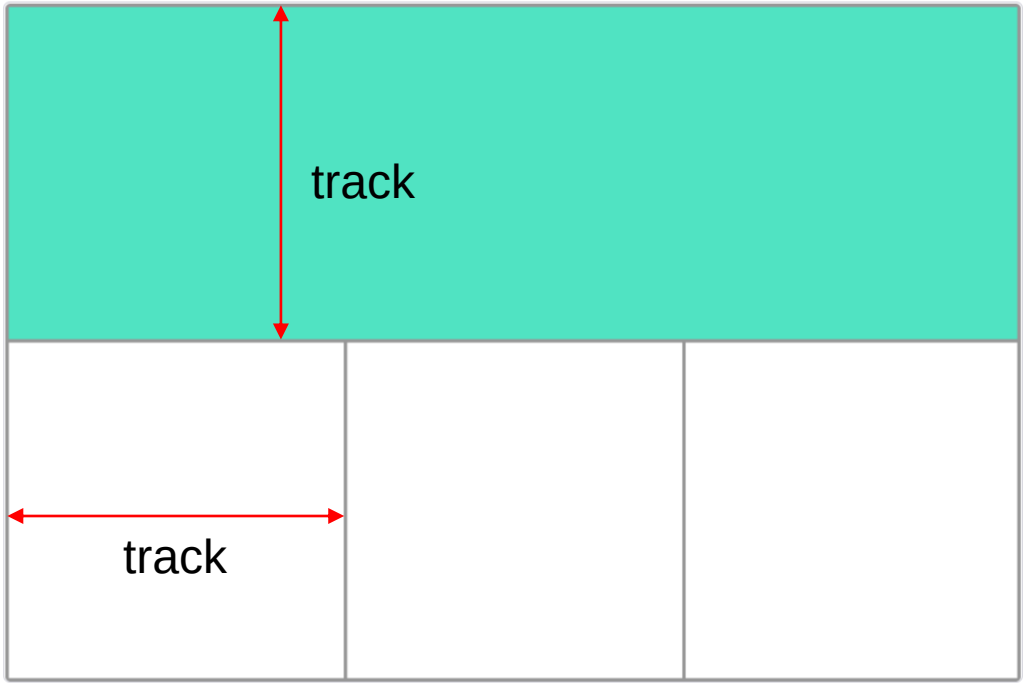
```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

```
.wrapper {
  display: grid;
}
```



Grid

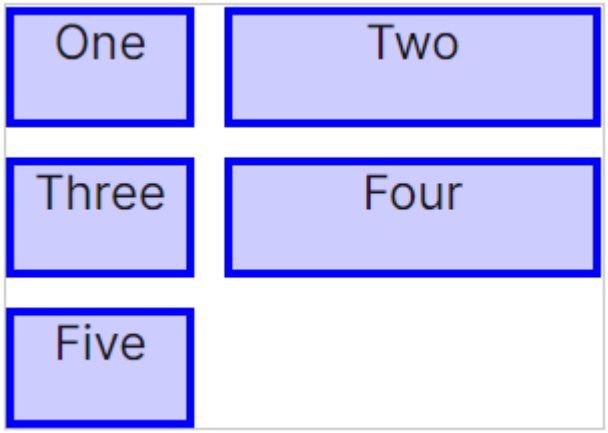
- what is the track
  - A grid track is the space between any two lines on the grid.



Grid layout

- grid-template-columns
  - defines the size of the column tracks.
  - **`fr`** represents a fraction of the available space in the grid container.


```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```




```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 2fr;
}
```

Grid layout

- repeat() notation
  - repeat all or a section of the track listing.


```
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

=

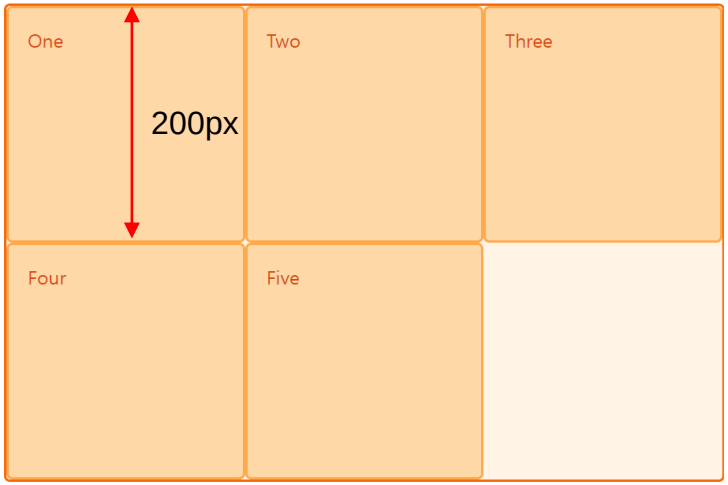
```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

Grid layout

- The implicit and explicit grid
  - When creating grid defined our column tracks with the **grid-template-columns**, the grid also created rows on its own.
  - at this case, explicit grid defined with **grid-template-columns**
  - use **grid-auto-rows** to define implicit grid (rows)

```
<div class="wrapper">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

```
.wrapper {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 200px;
}
```

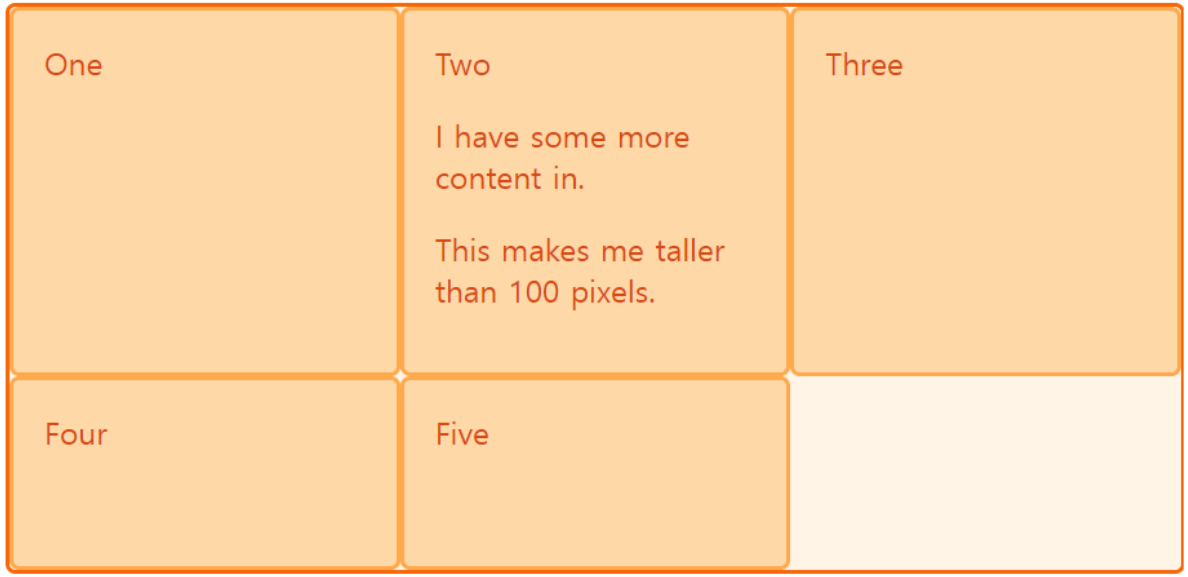


Grid layout

- using minmax()
  - `minmax(min, max)`

```
<div class="wrapper">
  <div>One</div>
  <div>Two
    <p>I have some more content in.</p>
    <p>This makes me taller than 100 pixels.</p>
  </div>
  <div>Three</div>
  <div>Four</div>
  <div>Five</div>
</div>
```

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

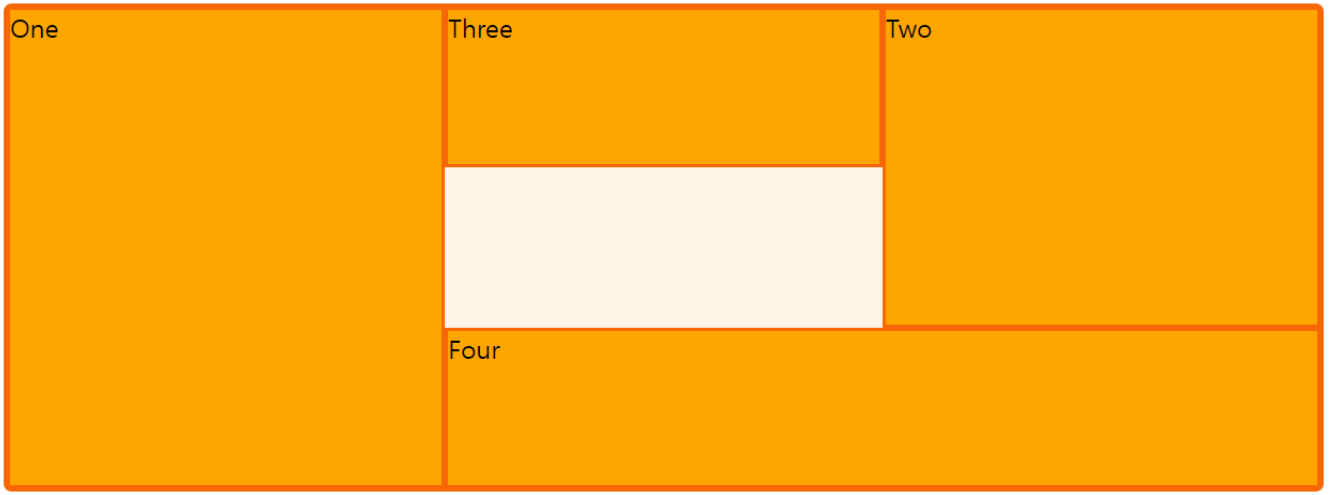


Line-based placement

- Positioning items by line number

```
.box1 {
   grid-column-start: 1;
   grid-column-end: 2;
   grid-row-start: 1;
   grid-row-end: 4;
}
.box2 {
   grid-column-start: 3;
   grid-column-end: 4;
   grid-row-start: 1;
   grid-row-end: 3;
}
```

```
.box3 {
   grid-column-start: 2;
   grid-column-end: 3;
   grid-row-start: 1;
   grid-row-end: 2;
}
.box4 {
   grid-column-start: 2;
   grid-column-end: 4;
   grid-row-start: 3;
   grid-row-end: 4;
}
```

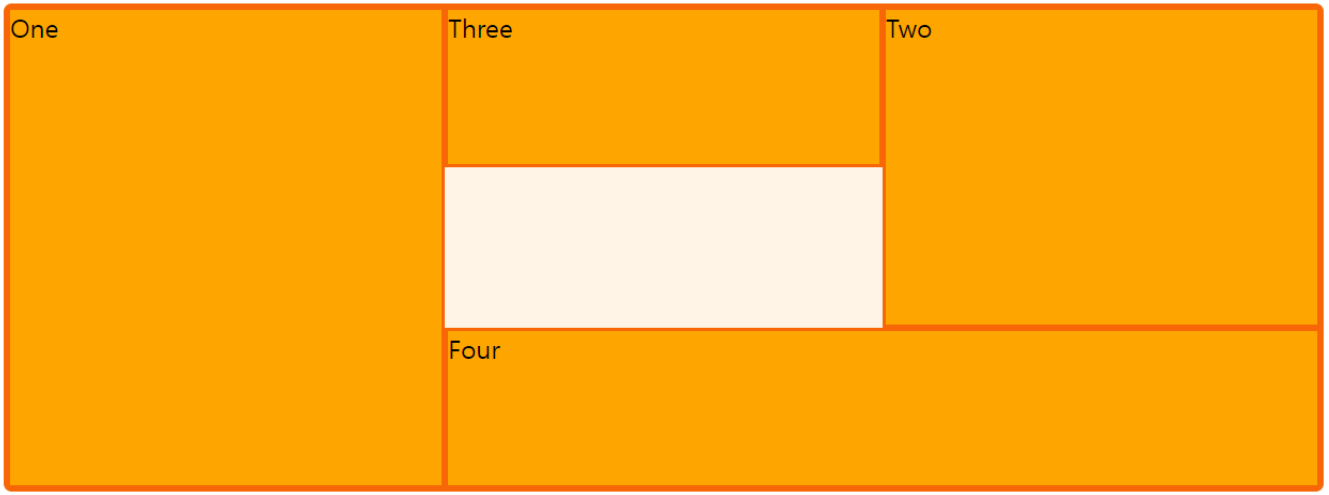


Line-based placement

- Positioning items by line number
  - using `grid-column` and `grid-row`

```
.box1 {
   grid-column: 1 / 2;
   grid-row: 1 / 4;
}
.box2 {
   grid-column: 3 / 4;
   grid-row: 1 / 3;
}
```

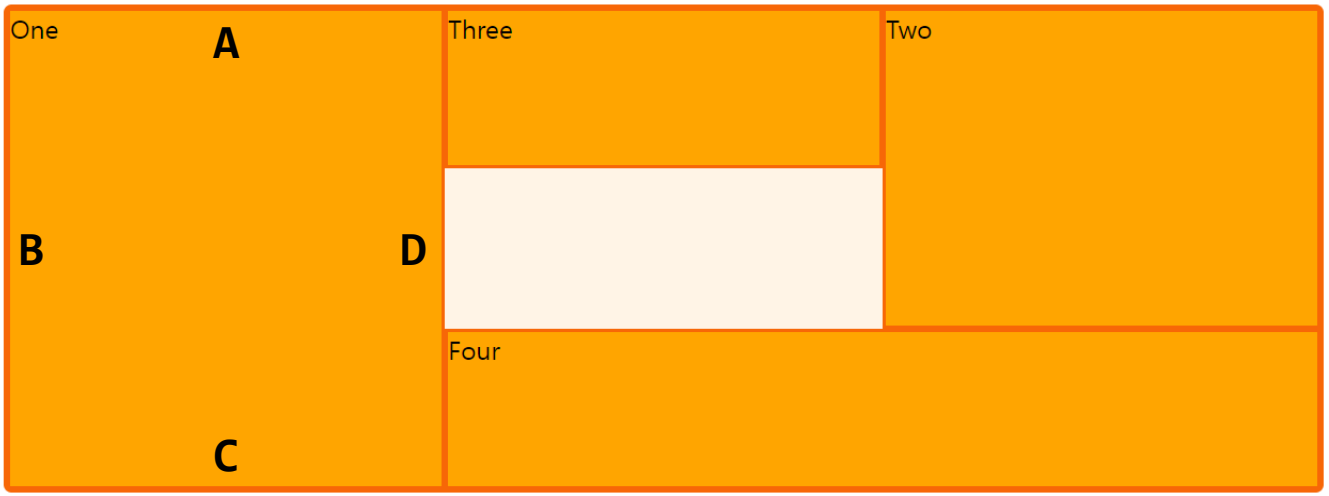
```
.box3 {
   grid-column: 2 / 3;
   grid-row: 1 /  2;
}
.box4 {
   grid-column: 2 / 4;
   grid-row: 3 / 4;
}
```



Line-based placement

- Positioning items by line number
  - using `grid-area`

```
.box1 {
   grid-area: 1 / 1 / 4 / 2;
}
.box2 {
   grid-area: 1 / 3 / 3 / 4;
}
.box3 {
   grid-area: 1 / 2 / 2 / 3;
}
.box4 {
   grid-area: 3 / 2 / 4 / 4;
}
```



`grid-area: A / B / C / D;`

Grid template areas

- Naming a grid area
  - `grid-area`
  - `grid-template-areas`

```
.header {
    grid-area: hd;
}
.footer {
    grid-area: ft;
}
.content {
    grid-area: main;
}
.sidebar {
    grid-area: sd;
}
```

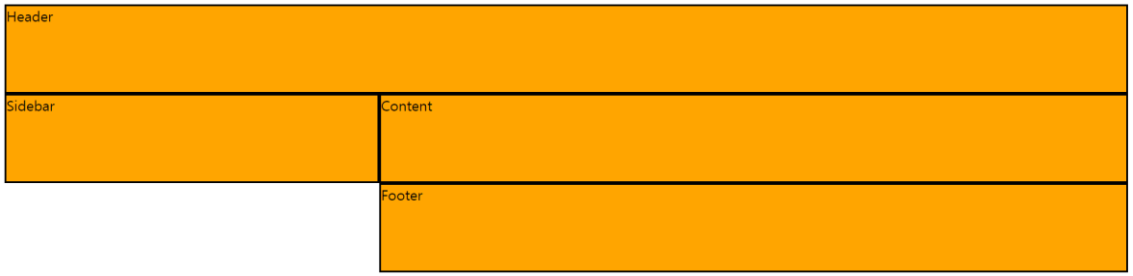
```
.wrapper {
    display: grid;
    grid-template-columns: repeat(9, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-template-areas:
      "hd hd hd hd   hd   hd   hd   hd   hd"
      "sd sd sd main main main main main main"
      "ft ft ft ft   ft   ft   ft   ft   ft";
}
```

Grid template areas

- Leaving a grid cell empty

```
.header {
    grid-area: hd;
}
.footer {
    grid-area: ft;
}
.content {
    grid-area: main;
}
.sidebar {
    grid-area: sd;
}
```

```
.wrapper {
    display: grid;
    grid-template-columns: repeat(9, 1fr);
    grid-auto-rows: minmax(100px, auto);
    grid-template-areas:
      "hd hd hd hd   hd   hd   hd   hd   hd"
      "sd sd sd main main main main main main"
      " .  .  . ft   ft   ft   ft   ft   ft";
}
```



과제 설명