

21일 (수)

1. Codecademy Making a Website Responsive - Learn CSS: Grid

Lesson 1. Grid Essentials

- Intro

그리드는 홈페이지 레이아웃을 위한 강력한 도구이다.

- `grid-template-columns`
- `grid-template-rows`
- `grid-template`
- `grid-template-area`
- `grid-gap`
- `grid-row-start` / `grid-row-end`
- `grid-column-start` / `grid-column-end`
- `grid-area`

이번 레슨에서 배우게 될 그리드에 대한 프로퍼티들

- **Creating a Grid**

그리드를 만들기 위해서, 그리드 컨테이너 와 그리드 아이템들을 생성해야 한다. 그리드 컨테이너 html 태그에서 `display: grid or inline-grid` 를 추가해야한다. (flexbox 와 같다.)

```
5
6 ▼ <body>
7 ▼   <div class="grid">
8       <div class="box a">A</div>
9       <div class="box b">B</div>
10      <div class="box c">C</div>
11      <div class="box d">D</div>
12      <div class="box e">E</div>
13      <div class="box f">F</div>
14      <div class="box g">G</div>
15  </div>
16 </body>
17
```

A	
B	
C	
D	
E	
F	
G	

A
B
C
D
E
F
G

display : grid 를 적용한 모습

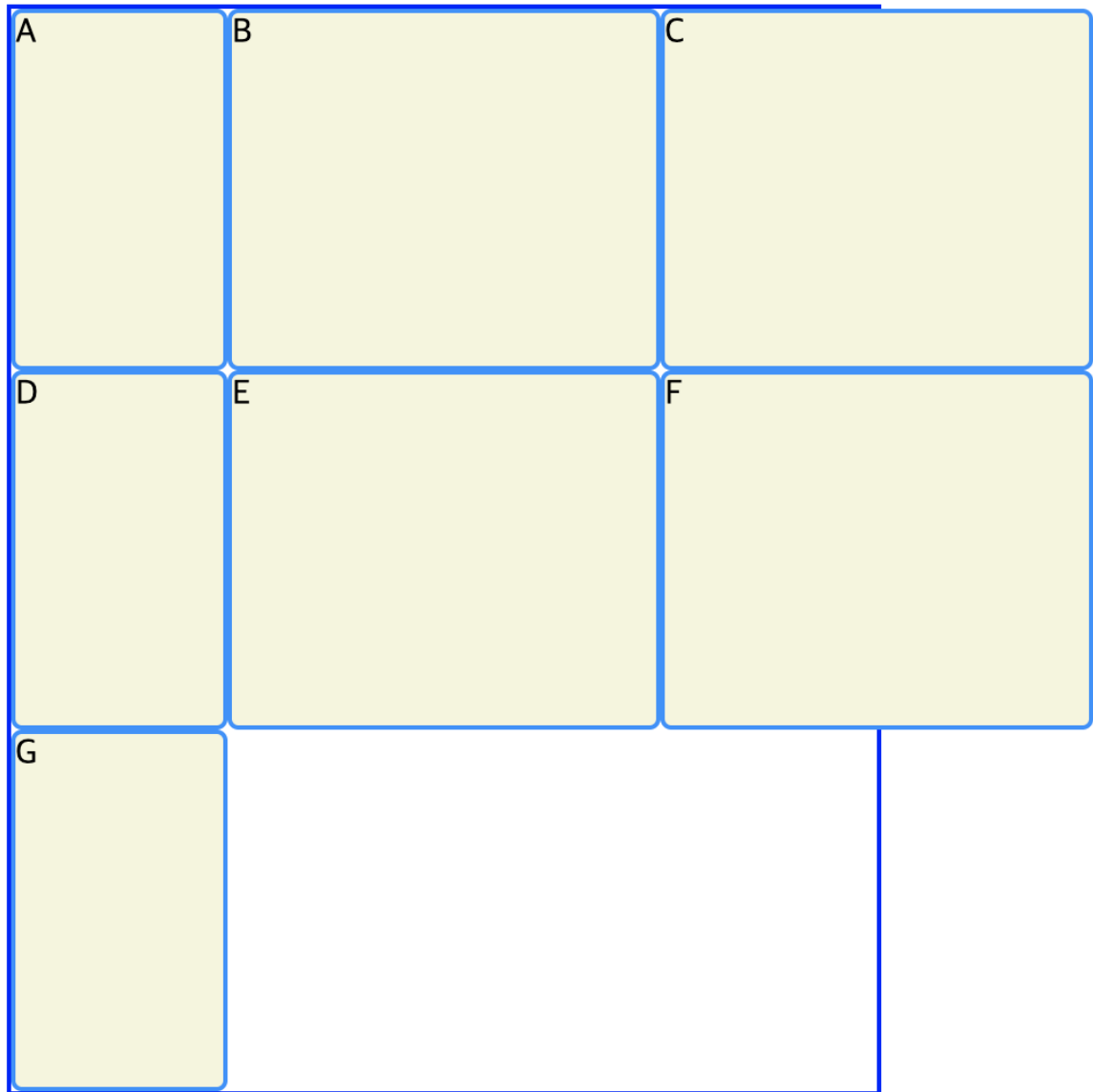
- **Creating Columns**

```
.grid {  
  display: grid;  
  width: 500px;  
  grid-template-columns: 100px 200px;  
}
```

그리드의 컬럼은 grid-template-columns 프로퍼티를 사용하여 생성한다. 각 100px, 200px 의 크기의 두 개의 컬럼을 생성하는 것을 의미한다. 이 프로퍼티의 디폴트 값은 1이다.

```
.grid {  
  display: grid;  
  width: 100px;  
  grid-template-columns: 20px 40%  
60px;  
}
```

위 예시는 3개의 컬럼을 생성한 것이며, px 대신 %로 써도 좋다.



그리드 컨테이너보다 더 넓은 컬럼을 생성하게 되면 이렇게 overflow 가 된다. 이러한 해결방법을 뒤에서 배워보자.

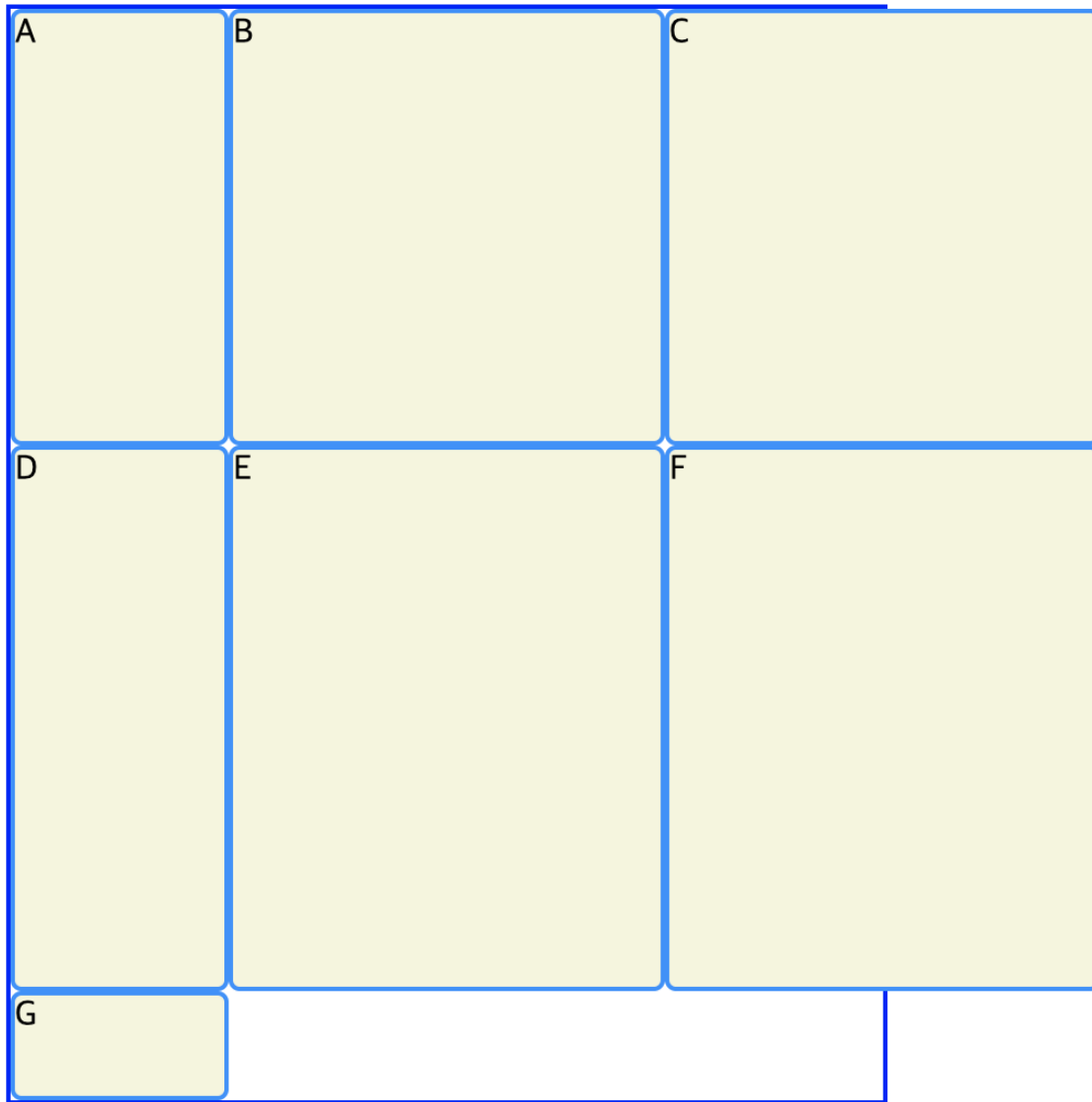
- **Creating Rows**

```
.grid {  
  display: grid;  
  width: 1000px;  
  height: 500px;  
  grid-template-columns: 100px 200px;  
  grid-template-rows: 10% 20% 600px;  
}
```

row 또한 위 프로퍼티를 사용하면 된다. %로 정의될 때, column은 width 의 % , row 는 height 의 % 로 적용된다.

예시

```
1 ▼ .grid {  
2   display: grid;  
3   border: 2px blue solid;  
4   width: 400px;  
5   height: 500px;  
6   grid-template-columns: 100px 50%  
   200px;  
7   grid-template-rows: 40% 50% 50px;  
8 }  
9
```

- **Grid Template**

앞서 적용했던 프로퍼티들의 솔핸드 버전이다.

```
.grid {
  display: grid;
  width: 1000px;
  height: 500px;
  grid-template: 200px 300px / 20%
  10% 70%;
}
```

'/' 을 기준으로 앞은 row , 뒤는 column 이다.

- **Fraction (비, 비율)**

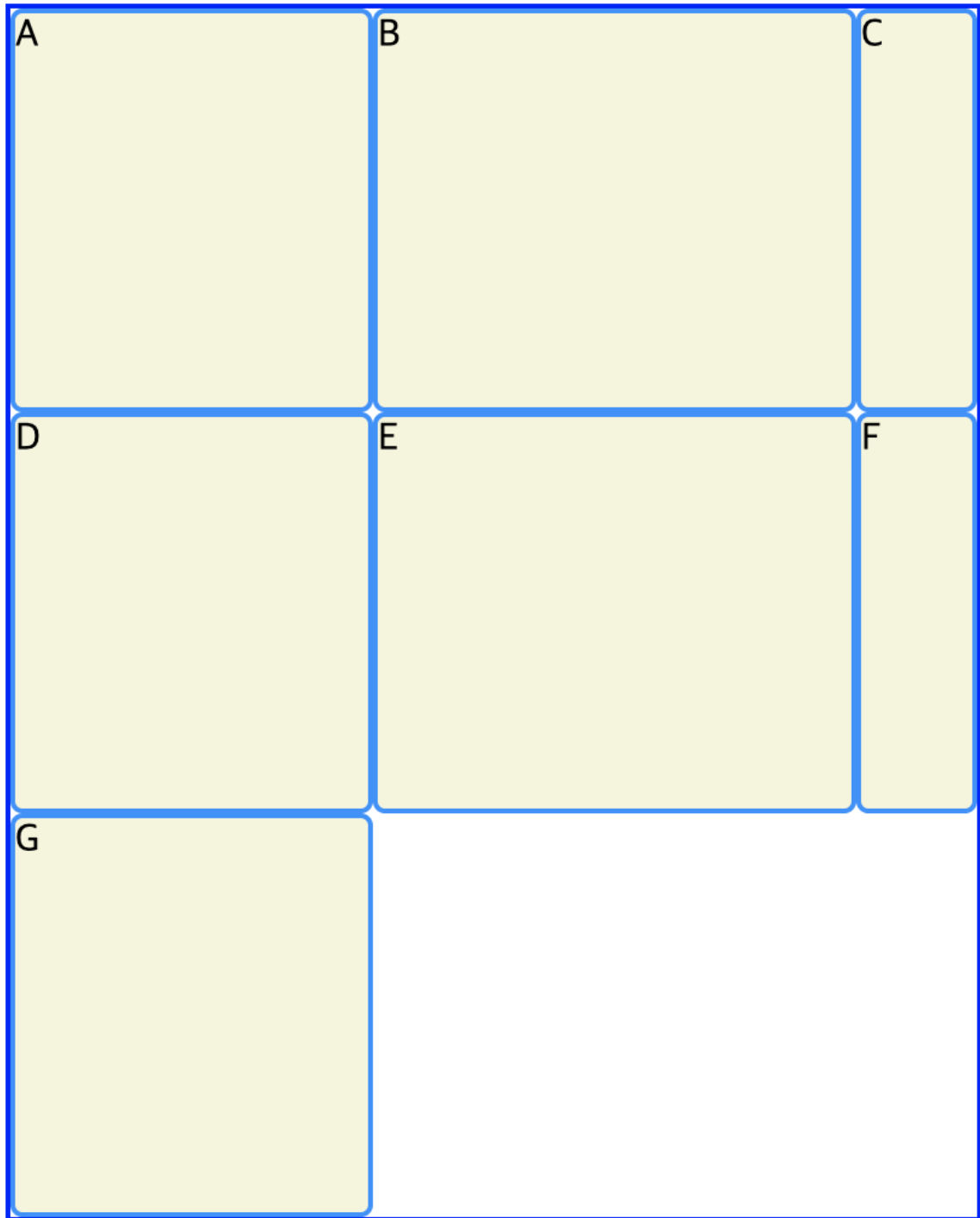
css 에서 반응형 유닛인 %, em, rem 등과 같이 또 다른 그리드에서 쓰이는 반응형 유닛이 있다. 바로 fr(fraction) 이다.

반응형으로 만들 뿐만 아니라 오버플로우를 방지한다.

```
.grid {
  display: grid;
  width: 1000px;
  height: 400px;
  grid-template: 2fr 1fr 1fr / 1fr
  3fr 1fr;
}
```

```
1 ▼ .grid {  
2     display: grid;  
3     border: 2px blue solid;  
4     width: 400px;  
5     height: 500px;  
6     grid-template: 1fr 1fr 1fr/ 3fr  
50% 1fr;  
7 }  
8
```

fr 을 써도 여전히 % 나 px 는 사용이 가능하다. %나 px 를 제외한 나머지 비율로 fr을 나눈다.



- **repeat 함수**

```
.grid {  
  display: grid;  
  width: 300px;  
  grid-template-columns: repeat(3,  
100px);  
}
```

이처럼 row 나 column의 갯수를 정하는 프로퍼티에 repeat() 함수를 사용할 수 있다.
repeat은 여러번 반복해야하는 수고를 덜어준다.

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

위에 repeat 과 동일한 커맨드

```
grid-template-columns: repeat(2, 20px  
50px)
```

이처럼 멀티인자를 가질 수도 있다. 이럴경우 20px 50px 20px 50px 와 동일한 출력값을 얻게 된다.

예시

```

1 ▼ .grid {
2     display: grid;
3     border: 2px blue solid;
4     width: 400px;
5     height: 500px;
6     grid-template: 1fr 1fr 1fr / 3fr 50%
                      1fr;
7 }

```

```

1 ▼ .grid {
2     display: grid;
3     border: 2px blue solid;
4     width: 400px;
5     height: 500px;
6     grid-template: repeat(3, 1fr) / 3fr
                      50% 1fr;
7 }

```

- minmax 함수

지금까지 배운 그리드는 모두 사이즈가 고정된 그리드였다. 이번에 배울 minmax 함수는 그리드를 반응형으로 사이즈를 크고줄일 수 있는 기능을 가지고 있다.

```
.grid {
  display: grid;
  grid-template-columns: 100px
minmax(100px, 500px) 100px;
}
```

이런식으로 원하는 그리드에 minmax(min, max) 값을 넣으면 된다. 두번째 컬럼 그리드만 브라우저 사이즈에 따라 변하게 된다.

minmax 를 사용하기 위해 width 를 지정하면 안된다!!(?)

- **Grid Gap**

그리드 아이템 사이의 갭을 넣기 위해 grid-column-gap, grid-row-gap 프로퍼티를 사용해보자.

이러한 프로퍼티는 그리드의 시작과 끝에 공간을 추가하지않는다는 것을 알고 있자. fr 은 이러한 갭까지 모두 빼고나서 비를 계산한다.

```
.grid {
  display: grid;
  width: 320px;
  grid-template-columns: repeat(3,
1fr);
  grid-column-gap: 10px;
}
```

```
.grid {  
  display: grid;  
  width: 320px;  
  grid-template-columns: repeat(3,  
1fr);  
  grid-gap: 20px 10px;  
}
```

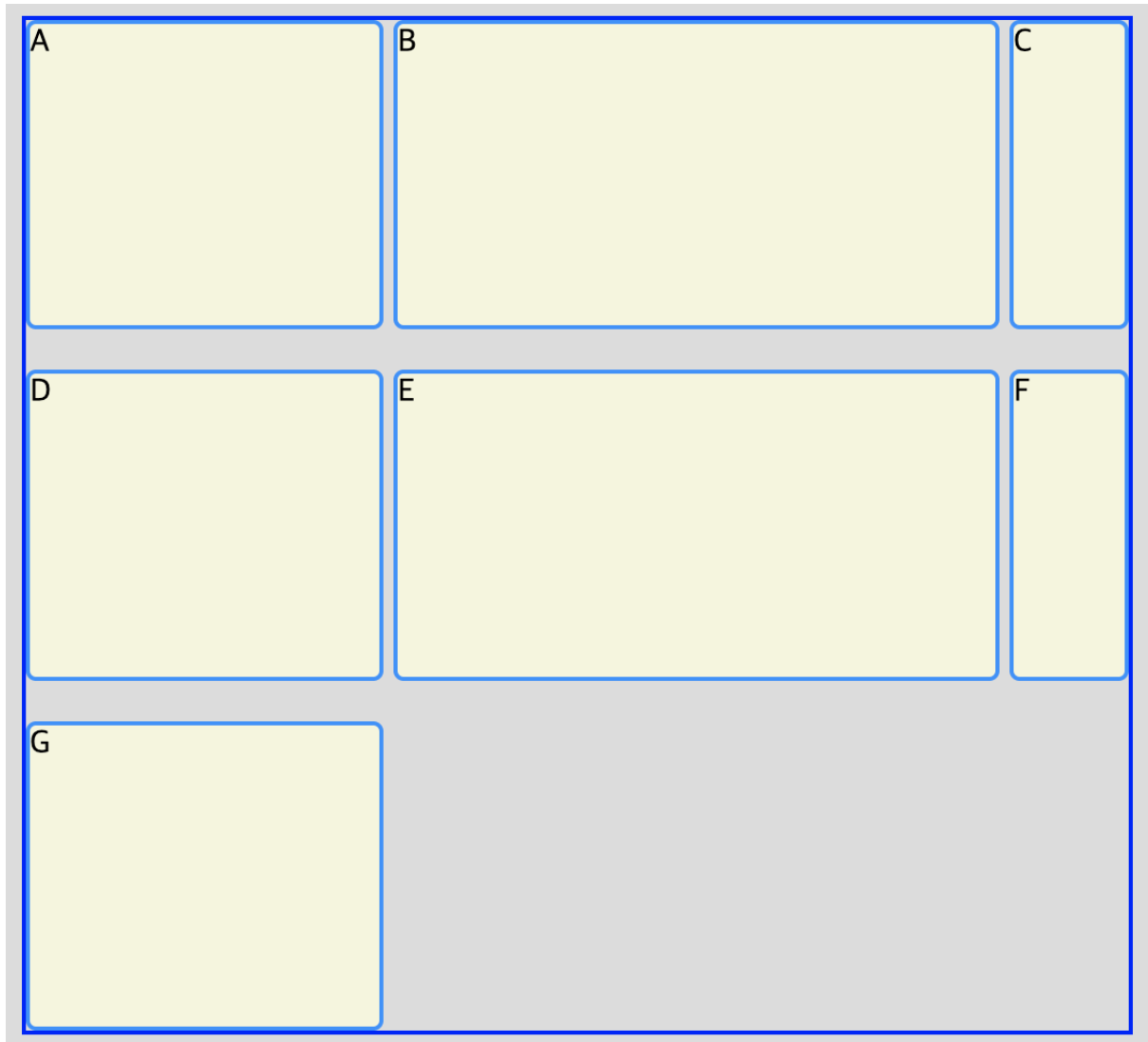
grid-gap 솔핸드 작성 순서는 앞에서부터 row , column 순이다.

예시

```
▼ .grid {  
  display: grid;  
  border: 2px blue solid;  
  height: 500px;  
  grid-template: repeat(3, 1fr) / 3fr  
minmax(50px, 300px) 1fr;  
  grid-row-gap: 20px;  
  grid-column-gap: 5px;  
}
```



```
▼ .grid {  
  display: grid;  
  border: 2px blue solid;  
  height: 500px;  
  grid-template: repeat(3, 1fr) / 3fr minmax  
(50px, 300px) 1fr;  
  //grid-row-gap: 20px;  
  //grid-column-gap: 5px;  
  grid-gap: 20px 5px;  
}  
  
▼ .box {  
  background-color: beige;  
  color: black;  
  border-radius: 5px;  
  border: 2px dodgerblue solid;  
}
```

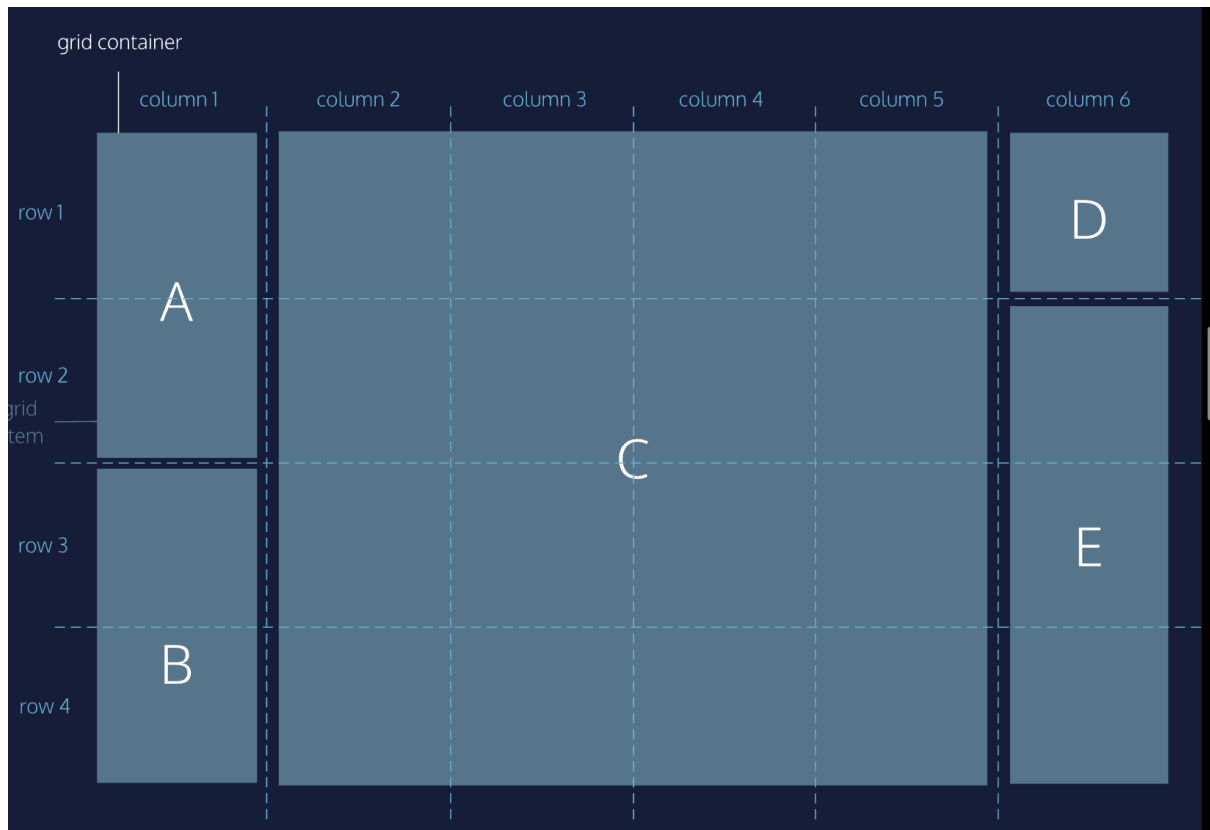


위와 같이 그리드의 가로세로 가장 끝부분은 gap 이 추가가 되지 않는다.

지금까지 배운 프로퍼티들은 모두 그리드 컨테이너에 지정하는 프로퍼티들이다.

- **Grid Items**

지금까지 그리드 컨테이너를 어떻게 지정하는지에 대해 배웠다.



앞으로 그리디 아이템을 이용하여 다음과 같이 여러개의 그리디 아이템을 합쳐 멋진 구성을 만들어 볼 것이다.

- **Multiple Row Items**

grid-row-start, grid-row-end 프로퍼티를 배워보자. 위 프로퍼티를 이용하여 멀티플 로우를 만들 수 있다. 이제 그리드 컨테이너에 css 를 적용하지말것!

grid-row-start 는 시작점을 뜻하며, end 는 끝나는 지점인데 원하는 지점보다 1이 더 큰 숫자를 써야하는 점을 주의하자!

start 가 end 보다 더 클 수 있으며, 마이너스 값을 가질 수도 있다. 이에 대해서는 documentation 을 참고하자

```
.item {  
  grid-row-start: 1;  
  grid-row-end: 3;  
}
```

예시

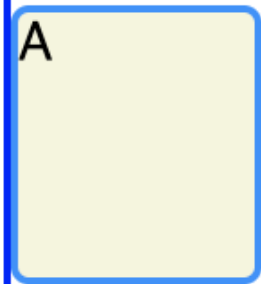
```
▼ .grid {  
  display: grid;  
  border: 2px blue solid;  
  height: 500px;  
  width: 500px;  
  grid-template: repeat(4, 1fr 2fr) / repeat(4,  
3fr 2fr);  
  grid-gap: 5px;  
}  
  
.a {  
  
}
```

가로 세로 8개의 그리드를 가지고 있는 그리드 컨테이너이다. 하지만 박스는 1개 뿐이다.

A

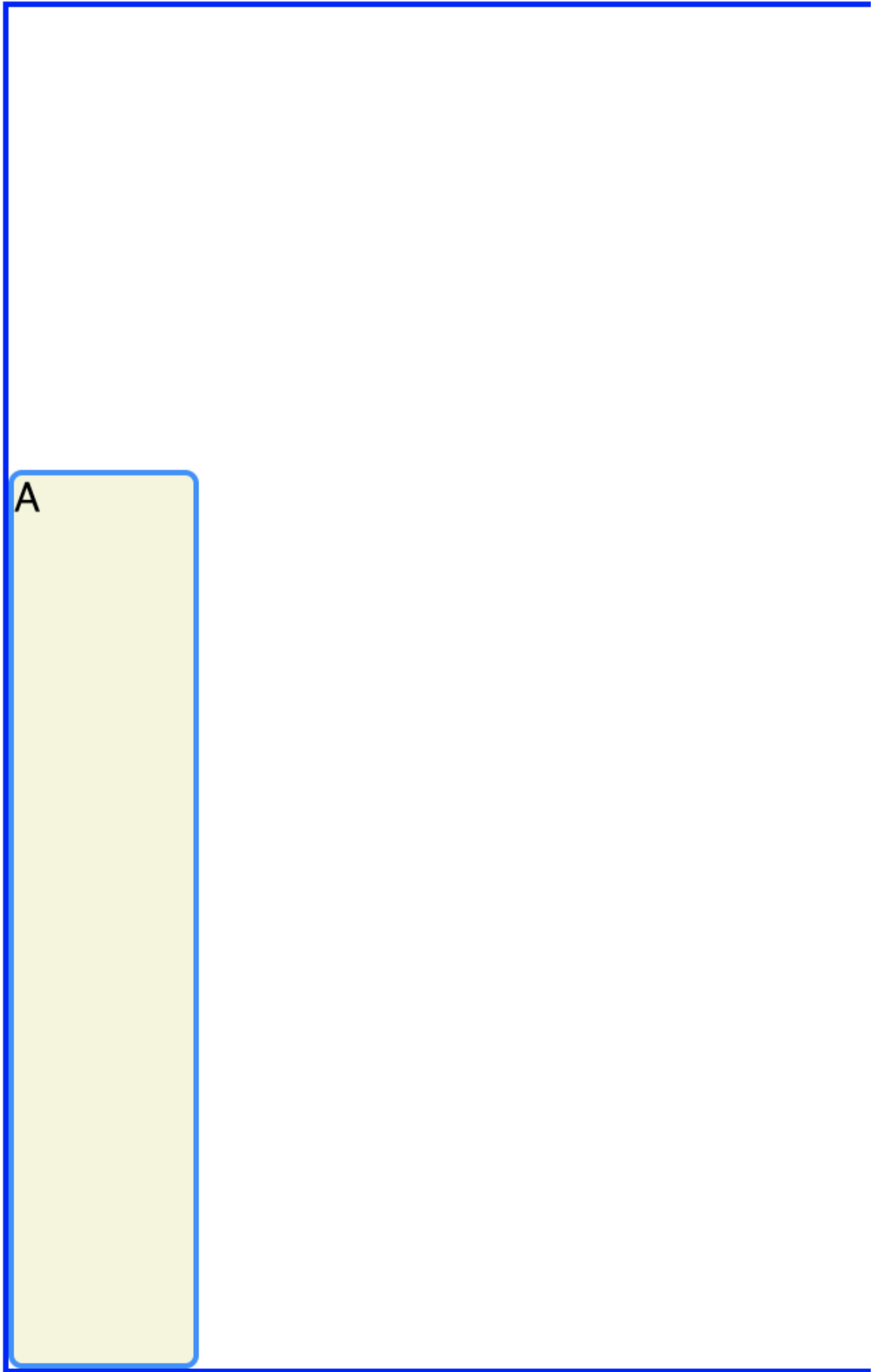
```
▼ .grid {  
  display: grid;  
  border: 2px blue solid;  
  height: 500px;  
  width: 500px;  
  grid-template: repeat(4, 1fr 2fr) / repeat(4,  
3fr 2fr);  
  grid-gap: 5px;  
}  
  
▼ .a {  
  grid-row-start: 4;  
}  
  
▼ .box {
```

a 클래스에 start 를 적용한 모습 4부터 시작



```
1 ▼ .grid {
2     display: grid;
3     border: 2px blue solid;
4     height: 500px;
5     width: 500px;
6     grid-template: repeat(4, 1fr 2fr) / repeat(4,
7         3fr 2fr);
8     grid-gap: 5px;
9 }
10 ▼ .a {
11     grid-row-start: 4;
12     grid-row-end: 9;
13 }
14
```

end 를 9로 지정했다. (end 는 1 더 큰 숫자를 써야한다.)



- **Grid Row**

위에 배운 grid-row-start와 end 의 속기법이 존재한다.

```
.item {  
  grid-row-start: 4;  
  grid-row-end: 6;  
}
```

```
.item {  
  grid-row: 4 / 6;  
}
```

- **Grid Column**

역시 grid-column-start , end 가 존재한다. 또한 속기법으로 grid-column 또한 존재한다.

```
.item {  
  grid-column: 4 / 6;  
}
```

4~5까지 column 을 합친다.

```
.item {  
  grid-column: 4 / span 2;  
}
```

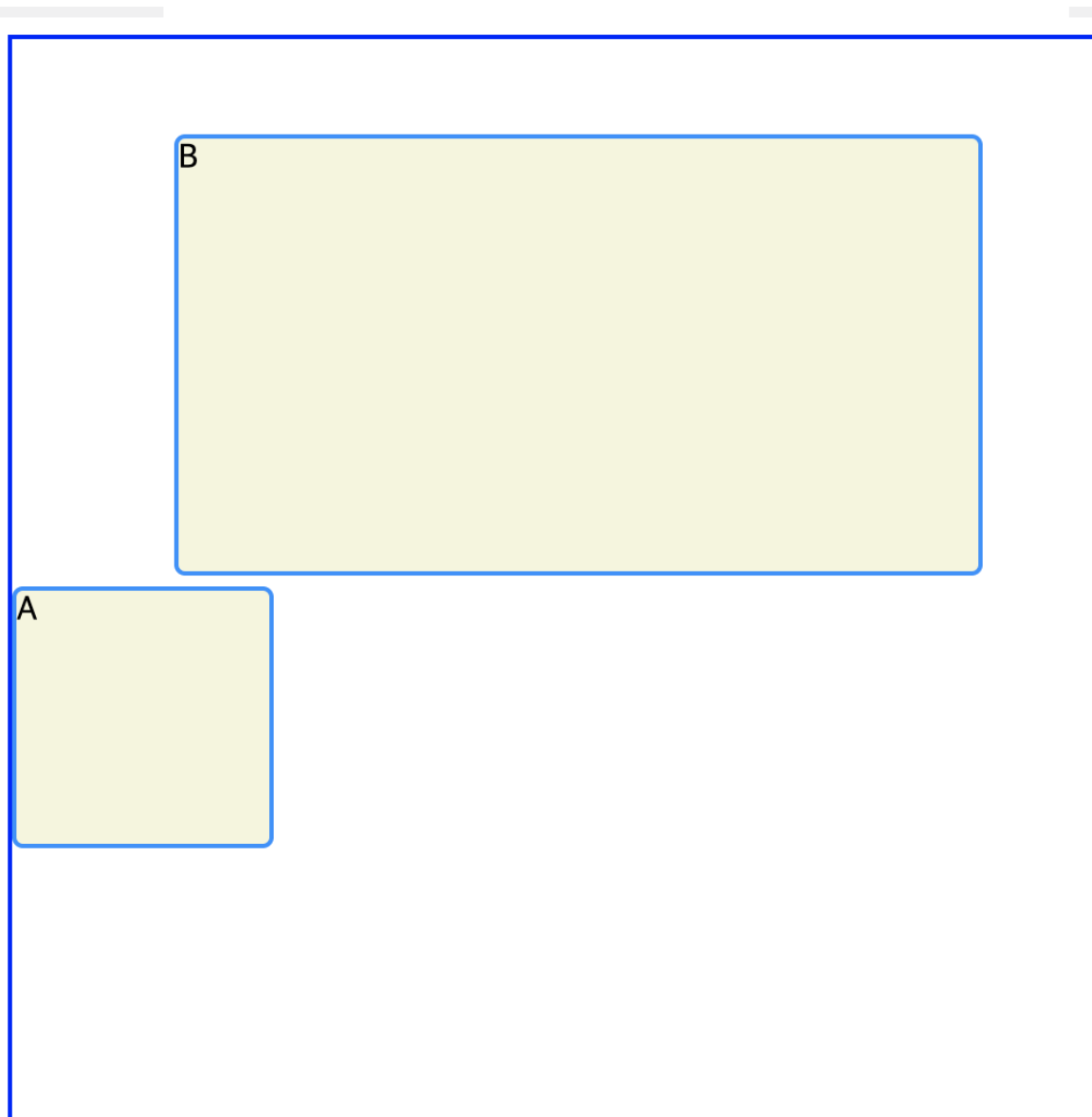
span 을 사용하면 1을 더하지않아서 생기는 오류를 줄일 수 있다. 위 예시는 4부터 시작해서 2개의 그리드로 만든다는 것이다.

```
.item {  
  grid-column-start: span 2;  
  grid-column-end: 6;  
}
```

예시

```
1 ▼ .grid {
2     display: grid;
3     border: 2px blue solid;
4     height: 500px;
5     width: 500px;
6     grid-template: repeat(4, 1fr 2fr) / repeat(4,
7     3fr 2fr);
8     grid-gap: 5px;
9 }
```

```
9
10 ▼ .a {
11     grid-row: 5 / 7;
12     grid-column: span 2 / 3;
13 }
14
15 ▼ .b {
16     grid-row: 2 / span 3;
17     grid-column: 2 / span 6;
18 }
19
```



- **Grid Area**

위에 배운 `grid-row-start`, `grid-row-end`, `grid-column-start`, `end` 이 4가지 모두를 한번에 적을 수 있는 프로퍼티가 있다. 바로 `grid-area` 다.

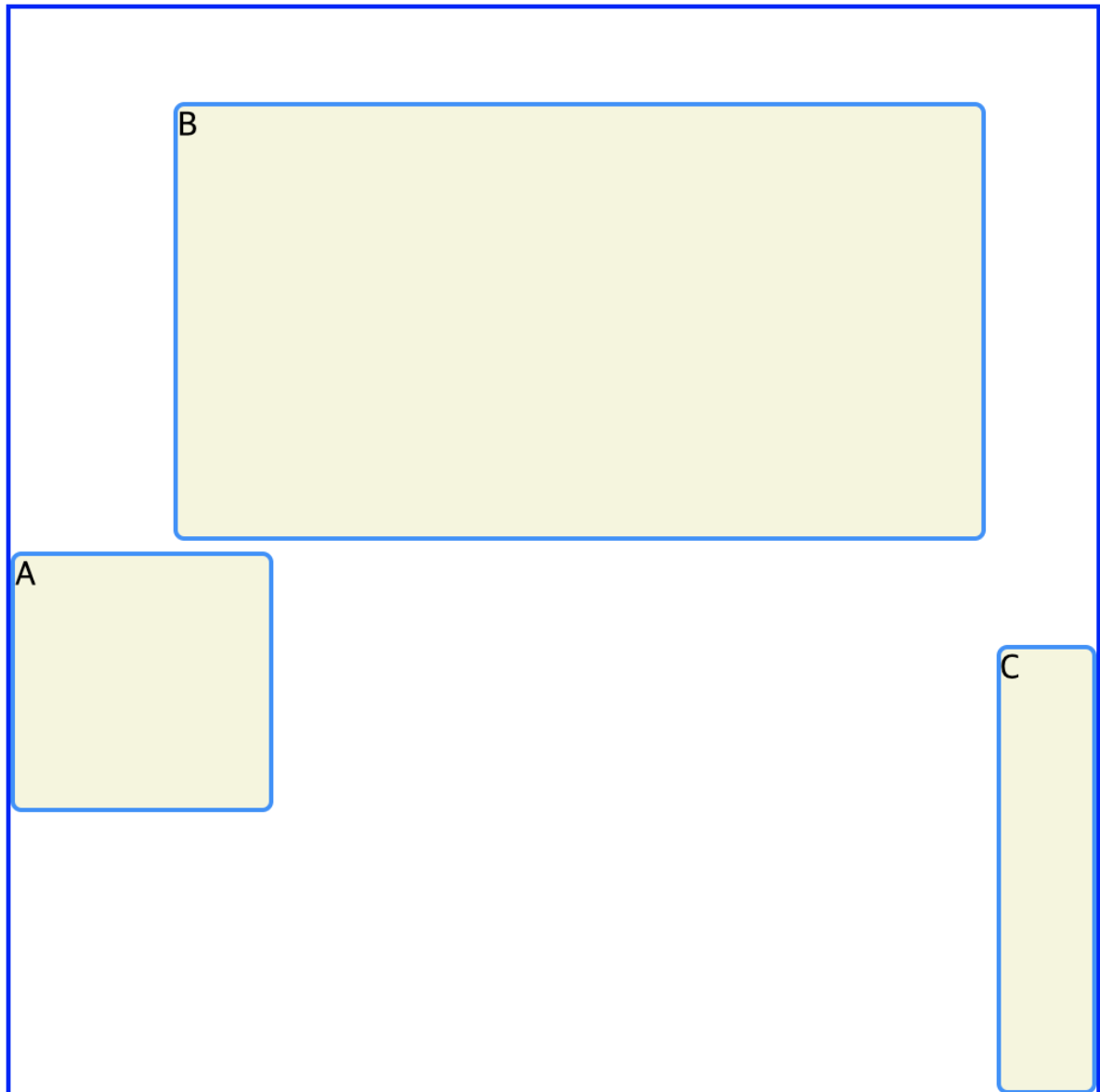
```
.item {  
  grid-area: 2 / 3 / 4 / span 5;  
}
```

`grid-area` takes four values separated by slashes. The order is important! This is how `grid-area` will interpret those values.

1. `grid-row-start`
2. `grid-column-start`
3. `grid-row-end`
4. `grid-column-end`

위와 같은 순서대로 써야한다!

```
▼ .a {  
  //grid-row: 5 / 7;  
  //grid-column: 1 / span 2;  
  grid-area: 5 / 1 / span 2 / span 2;  
}  
  
▼ .b {  
  //grid-row: 2 / span 3;  
  //grid-column: 2 / span 6;  
  grid-area: 2 / 2 / span 3 / span 6;  
}  
  
▼ .c {  
  grid-area: 6 / 8 / span 3 / span 1;  
}
```



2. REPL 이란?






















간단하게 테스트하고 출력하는 커맨드 환경이며

Read Eval Print Loop 라는 뜻으로 입력(read) , 평가(eval), 출력(print), 반복(loop) 이다.

repl 은 보통 CLI 위에서 작동한다. 윈도우는 명령프롬프트(CMD), 파워셸(PowerShell) 리눅스와 맥에서는 터미널 환경에서 사용이 가능하다.

REPL 을 사용하는 이유는 컴파일 과정 없이 즉석에서 코드를 입력해서 결과를 바로 알 수 있는 방식이기 때문이다. 보통 크롬의 개발자 모드에서 console 이 대표적인 REPL 환경이다.

3. VS Code 미니로그 뜻

	Methods and Functions	<code>method</code> , <code>function</code> , <code>constructor</code>
	Variables	<code>variable</code>
	Fields	<code>field</code>
	Type parameters	<code>typeParameter</code>
	Constants	<code>constant</code>
	Classes	<code>class</code>
	Interfaces	<code>interface</code>
	Structures	<code>struct</code>
	Events	<code>event</code>
	Operators	<code>operator</code>
	Modules	<code>module</code>
	Properties and Attributes	<code>property</code>
	Values and Enumerations	<code>value</code> , <code>enum</code>
	References	<code>reference</code>
	Keywords	<code>keyword</code>
	Files	<code>file</code>
	Folders	<code>folder</code>
	Colors	<code>color</code>
	Unit	<code>unit</code>
	Snippet prefixes	<code>snippet</code>
	Words	<code>text</code>

오늘의 단어

- take up : (이미 끝난 데서 시작하여) 계속하다.
- take up sth : (시,공간을) 차지하다(쓰다)