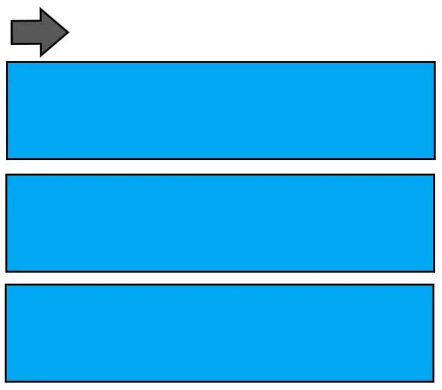
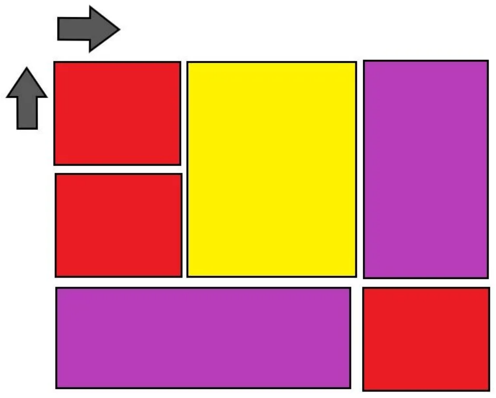
Day-6: CSS Grid

Introduction: Flex Vs Grid

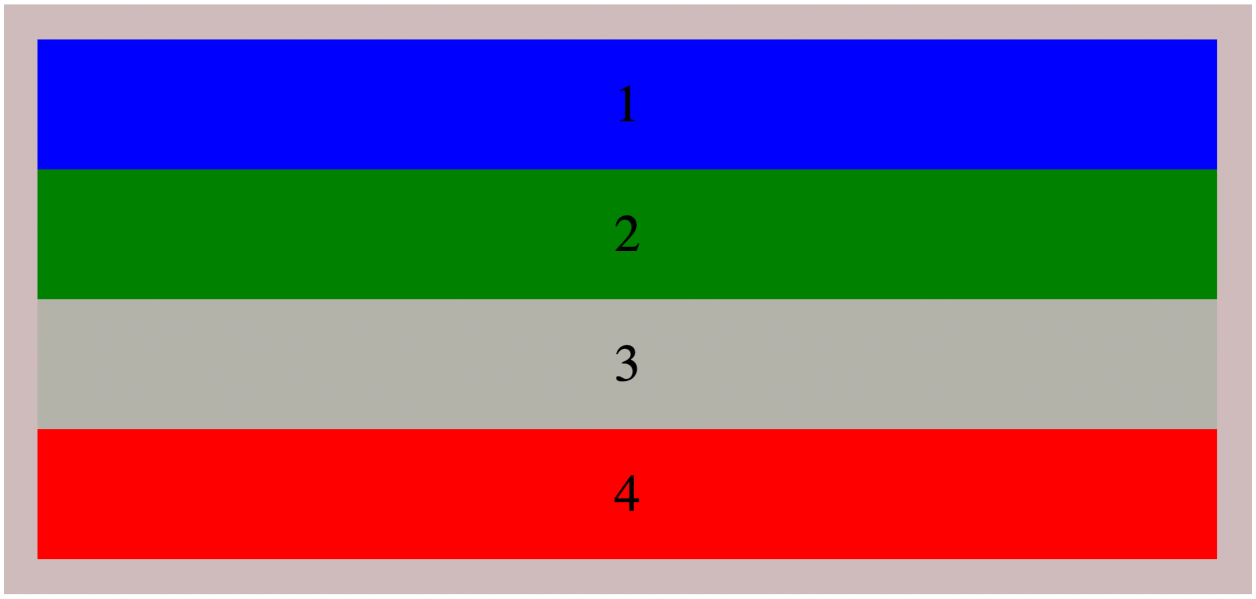

Flexbox is one-dimensional


CSS Grid is two-dimensional

Basic Grid :

```
<style> #container { display: grid; } </style> </head> <body> <div id="contai
ner"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>
6</div> </div> </body> Note: some styles are hidden like background color etc
```

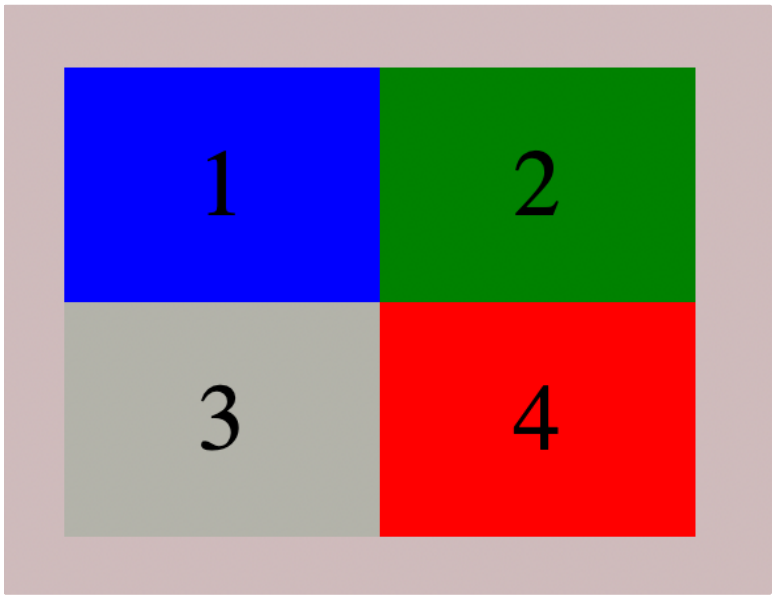
Output:



grid-template-columns

- Defines the columns and rows of the grid with a space-separated list of values.

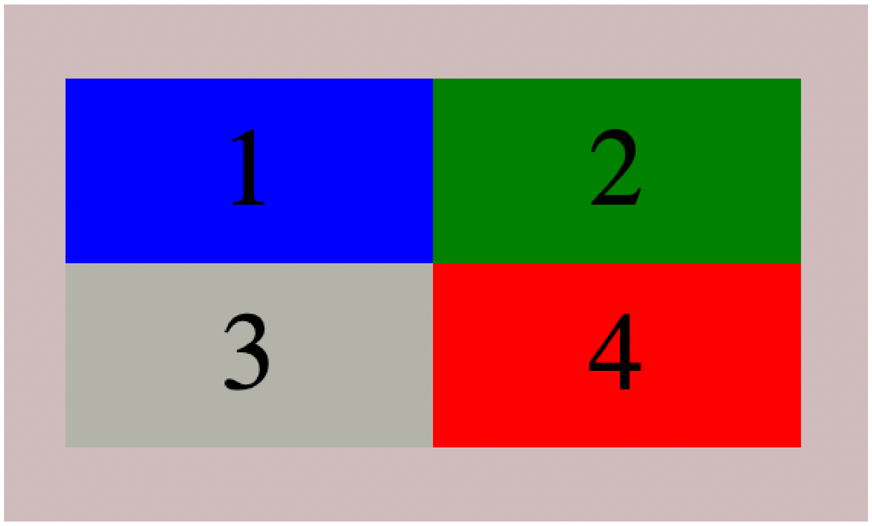
```
<style> #container { display: grid; grid-template-columns: 100px 100px; // 10
0px represents column size } </style> </head> <body> <div id="container"> <di
v>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div>
</div> </body> Note: some styles are hidden like background color etc
```



grid-template-rows

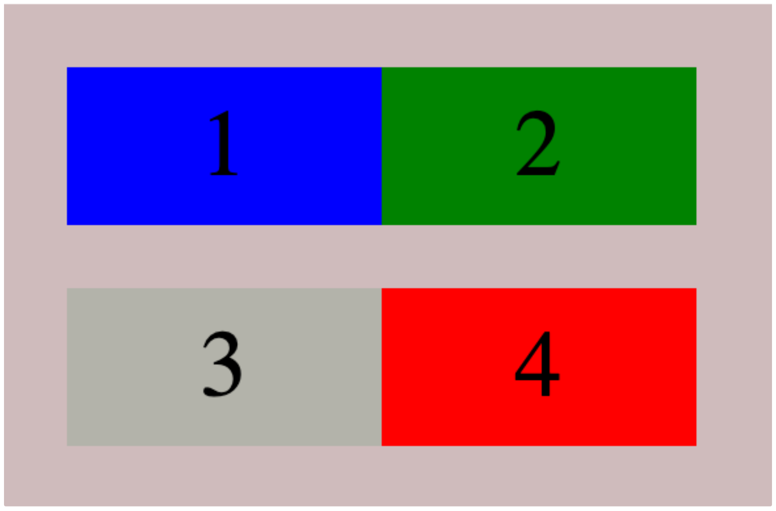
- We can also specify height of each row by using `grid-template-rows` property

```
<style> #container { display: grid; grid-template-columns: 100px 100px; // 10
0px represents column size grid-template-rows: 50px 50px ; // 100px represent
s row height } </style> </head> <body> <div id="container"> <div>1</div> <div
>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> </body>
Note: some styles are hidden like background color etc
```



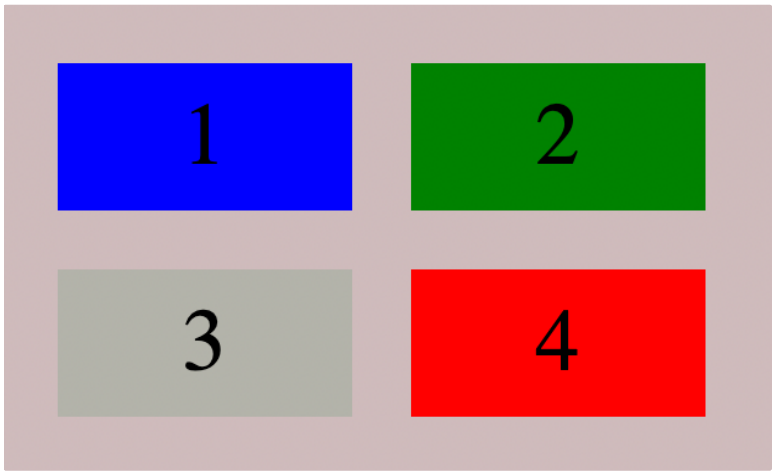
- To get gap between each item, we can use `grid-row-gap`

```
<style> #container { display: grid; grid-template-columns: 100px 100px; grid-
template-rows: 50px 50px; grid-row-gap:20px } </style> </head> <body> <div id
="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</di
v> <div>6</div> </div> </body>
```



- `grid-column-gap`

```
<style> #container { display: grid; grid-template-columns: 100px 100px; grid-template-rows: 50px 50px; grid-row-gap:20px; grid-column-gap:20px } </style> </head> <body> <div id="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> </body>
```



- Shorthand notation for grid-gap

```
<style> #container { display: grid; grid-template-columns: 100px 100px; grid-template-rows: 50px 50px; /* grid-row-gap:20px; grid-column-gap: 20px; */ gap:20px 20px } </style> </head> <body> <div id="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> </div> </body>
```

repeat():

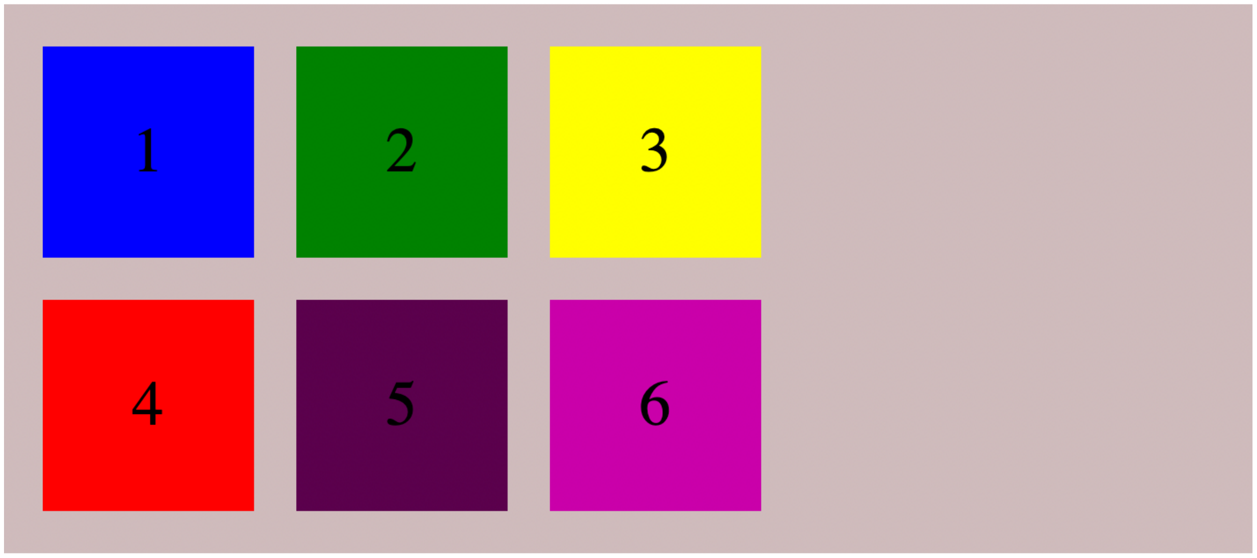
- The `repeat()` CSS function represents a repeated fragment of the track list, allowing a large number of columns or rows that exhibit a recurring pattern to be written in a more compact form.
- Syntax : `repeat(no_of_times,size)`
- for eg:
  - grid-template-columns: 100px 100px can be written as `repeat(2,100px)`
- Now the above code can be changed as

```
<style> #container { display: grid; grid-template-columns: repeat(2,100px); grid-template-rows: repeat(2,50px); gap:20px 20px } </style> </head> <body> <div id="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> </div> </body>
```

- Now lets try to build a layout which has 2 rows and 3 columns with different backgrounds

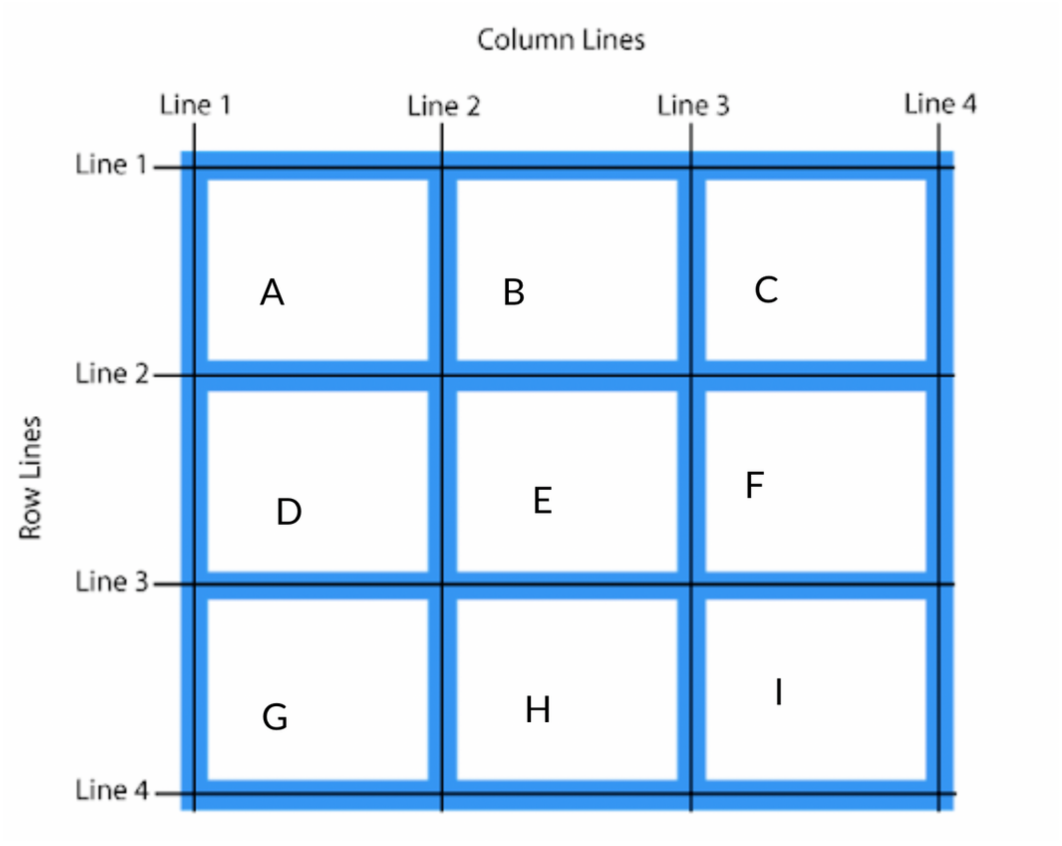
```
#container { display: grid; grid-template-columns: 100px 100px 100px; grid-template-rows: 100px 100px; gap:20px 20px } </style> </head> <body> <div id="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> </body>
```

Output:

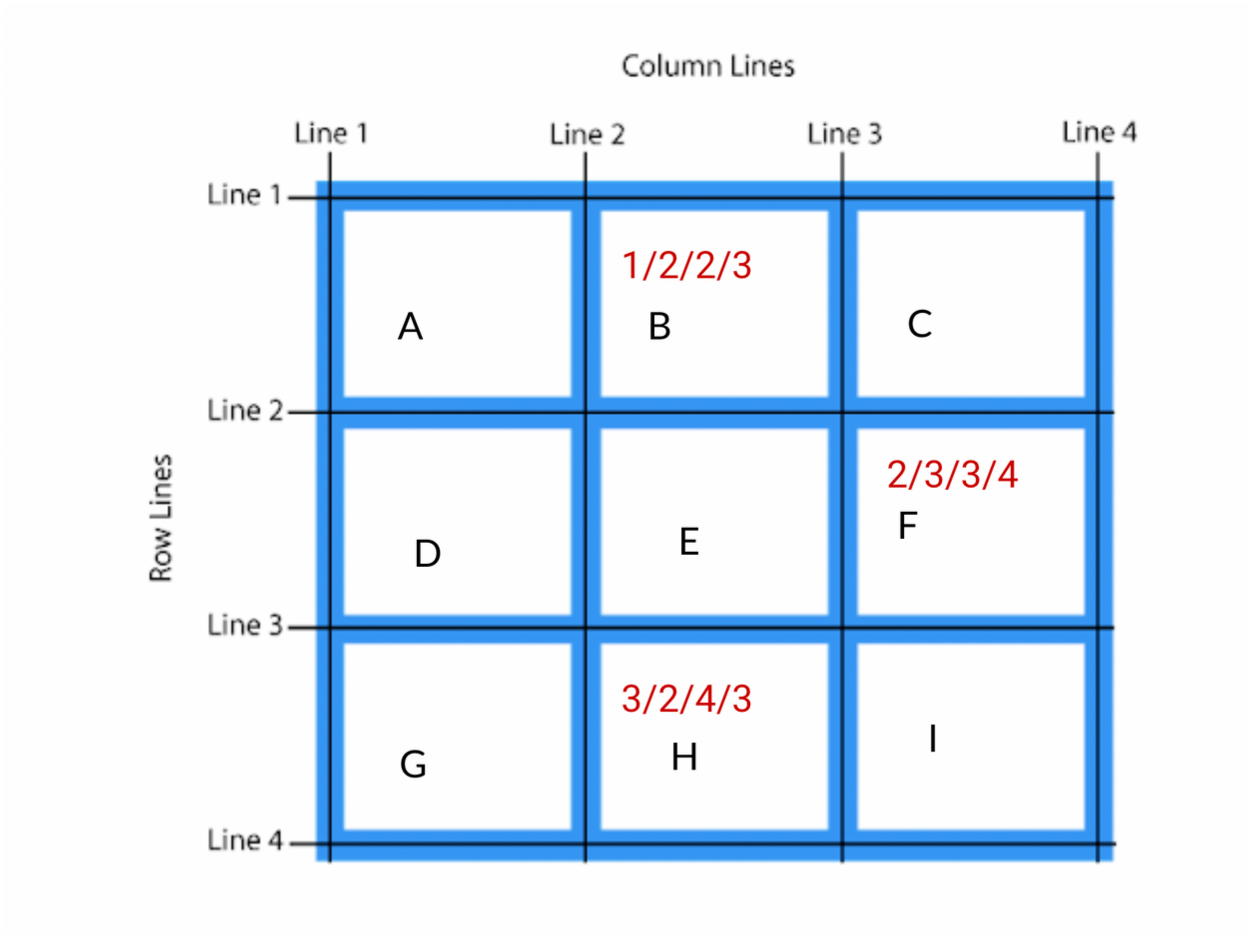


Live code: Codepen

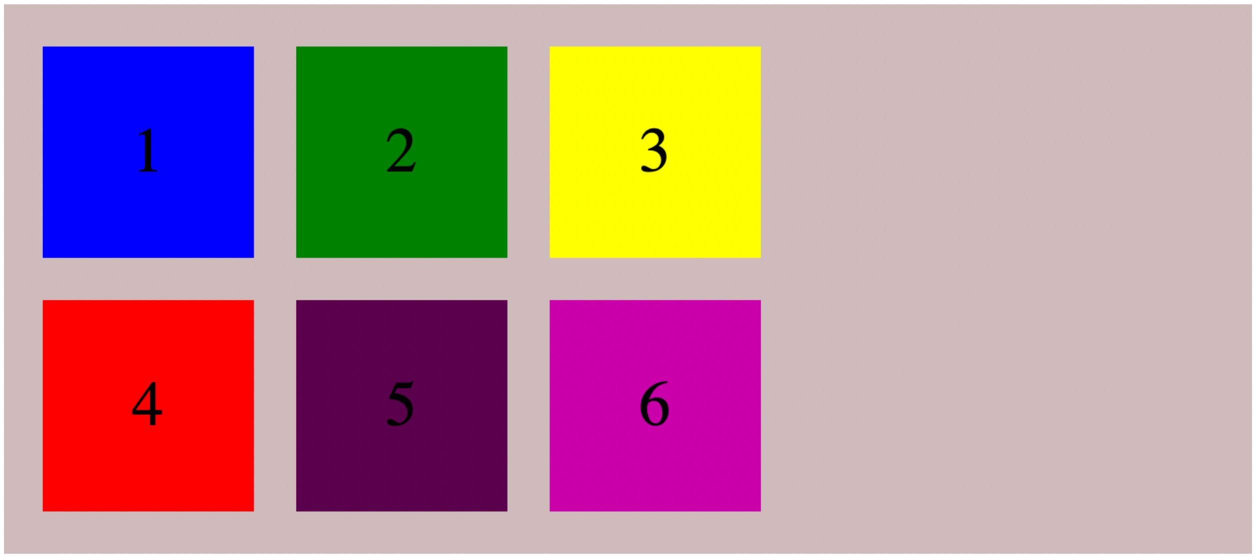
Line-based placement



- for block, B what are these 4 values?
  - row-start-line - 1
  - column-start-line - 2
  - row-end-line - 2
  - column-end-line - 3
- Similarly for H and F

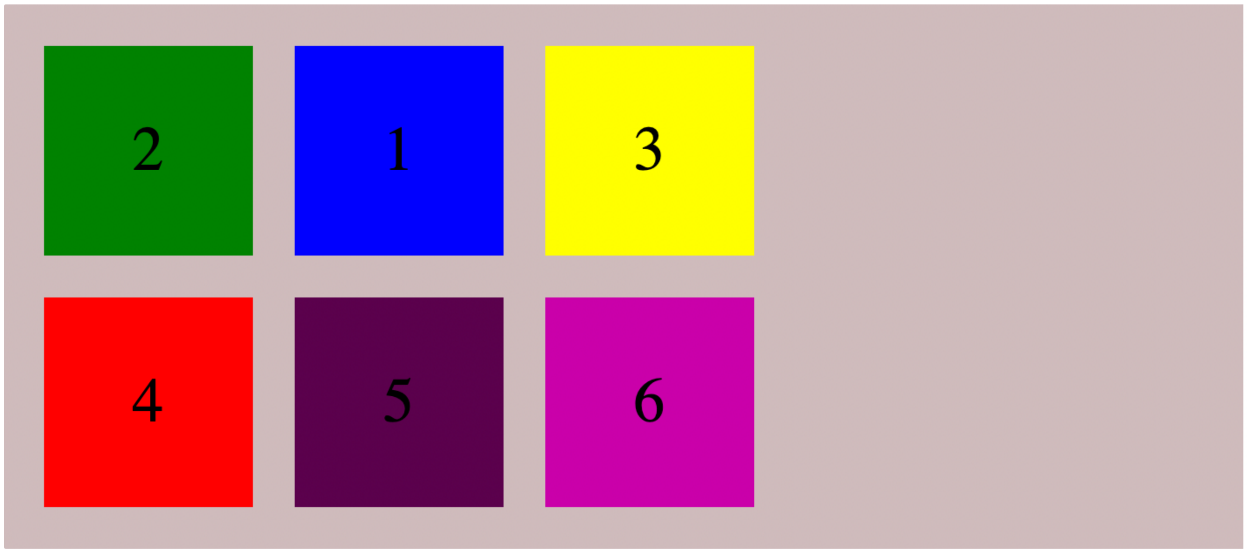


- From this layout If you want to move green box to first place, how can you acheive this using line based placement method?



```
<style> #container { display: grid; grid-template-columns: 100px 100px 100px;
grid-template-rows: 100px 100px; gap:20px 20px } #container > div:nth-child
(2) { grid-area:1/1/2/2; } </style> </head> <body> <div id="container"> <div>
1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </d
iv> </body>
```

Output:



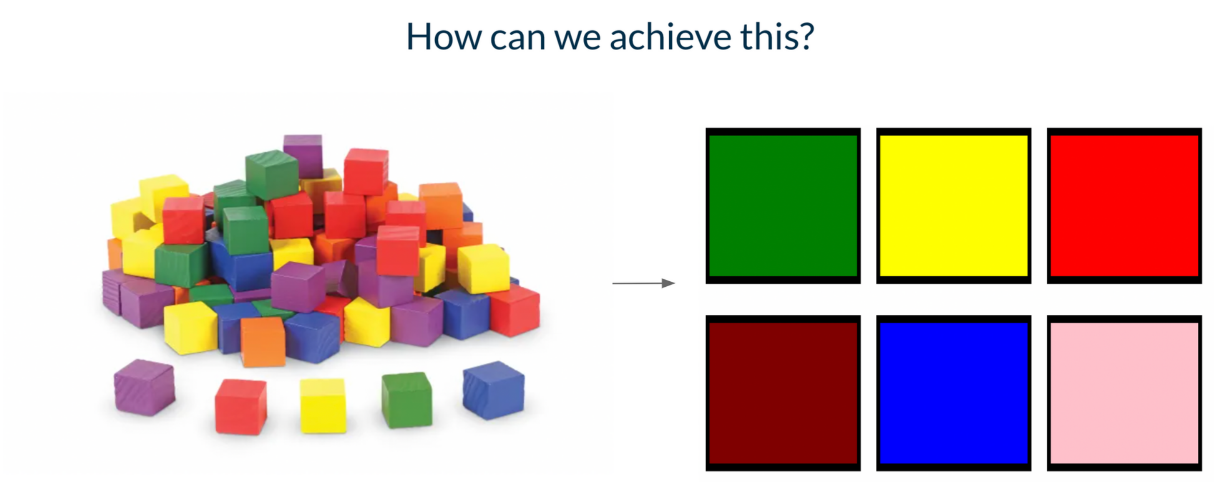
Live code: Codepen

- Can you really count these lines when this layout has a large number of columns.

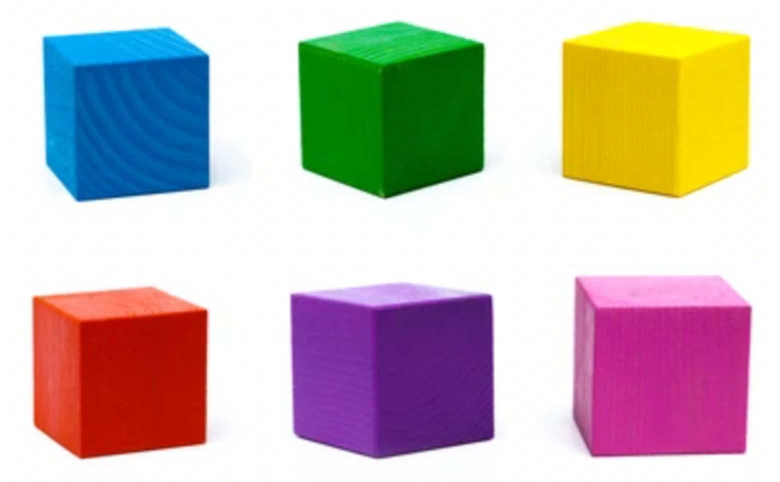
Grid-template-area method

- Since you face difficulty in line-based placement, we will now introduce to the template-area method.
- Did you play building blocks games in childhood

- If the answer is yes, how can you build this layout using blocks



- Answer: Just picking those colored blocks and placing it

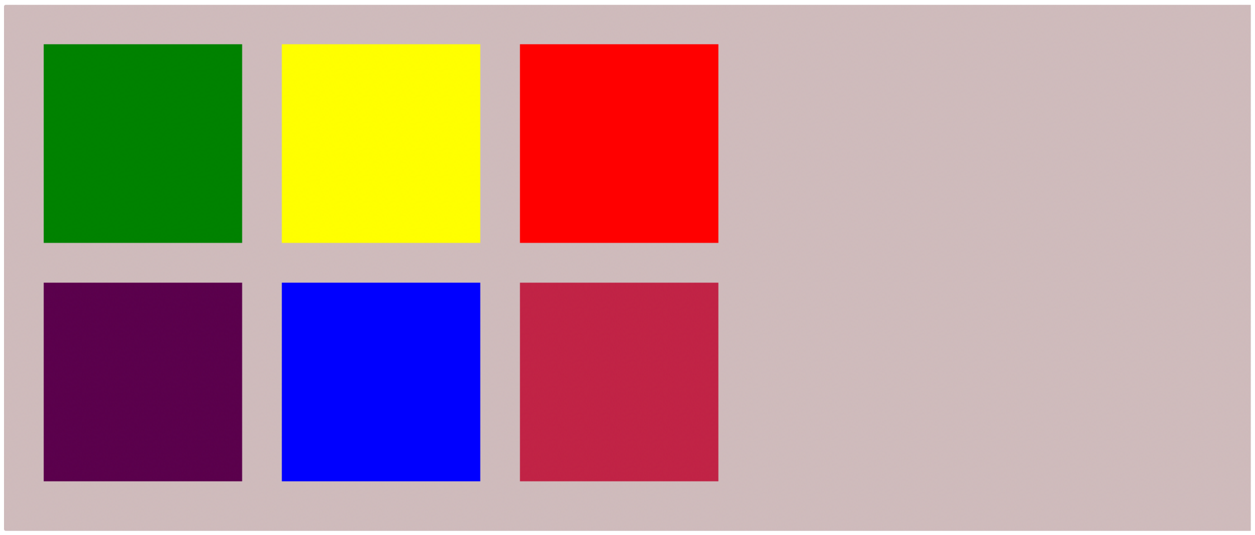


- For building that what might be the steps student has followed
    - Identifying colors they need to pick up, based on label(color)
    - Placing those colored blocks wherever it is needed
    - In this case, we have placed boxes in this order

    ```
    blue green yellow red maroon pink
    ```

```
<style> #container { display: grid; grid-template-columns: 100px 100px 100px;
grid-template-rows: 100px 100px 100px 100px; /* gap:row-gap column-gap */ ga
p: 20px; grid-template-areas: "grn ylw rd" "mrn blu dpink" } #container > di
v:nth-child(1) { background-color: blue; grid-area:blu /* label - name */ } #
container > div:nth-child(2) { background-color: green; grid-area:grn /* labe
l - name */ } #container > div:nth-child(3) { background-color: yellow; grid-
area:ylw } #container > div:nth-child(4) { background-color: red; grid-area:r
d } #container > div:nth-child(5) { background-color: rgb(83, 12, 74); grid-a
rea:mrn } #container > div:nth-child(6) { background-color: rgb(177, 52, 73);
grid-area:dpink } </style> </head> <body> <div id="container"> <div></div> <d
iv></div> <div></div> <div></div> <div></div> <div></div> </div> </body>
```

Output:



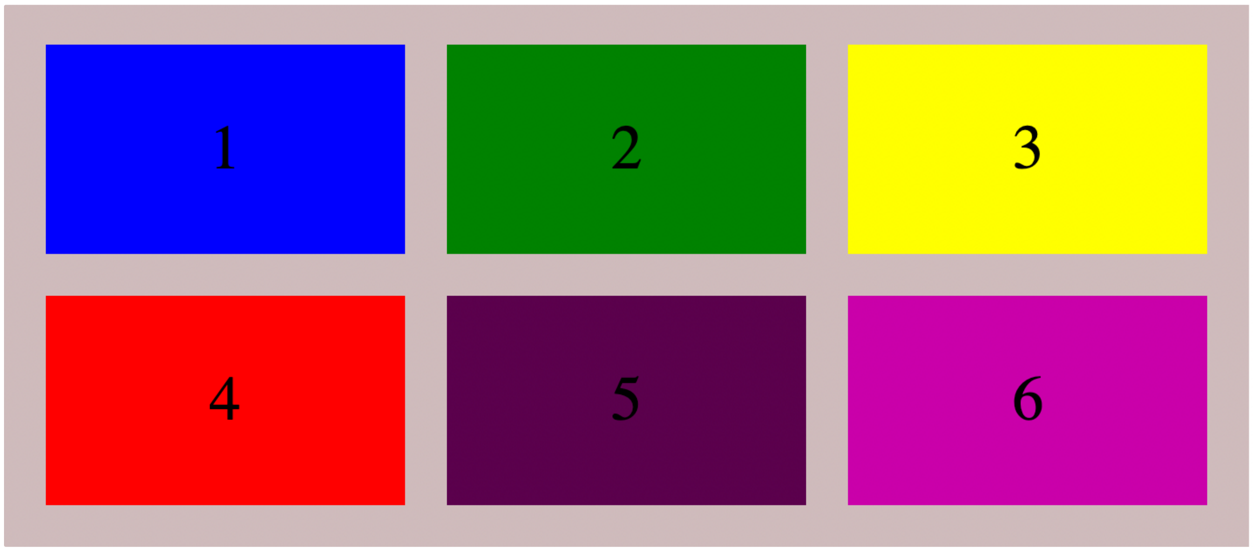
- Codepen

Units of grid-template-columns:

```
Pixels grid-template-columns: 100px 100px or repeat(2,100px) Percentages grid-template-columns: 50% 50% or repeat(2,50%) // each column will take 50% width in reference to its parent grid-template-columns: 1fr 1fr or repeat(2,1fr) // each column will take 1 fraction width in reference to its parent
```

```
<style> #container { display: grid; grid-template-columns: repeat(3,1fr); grid-template-rows: 100px 100px; gap:20px 20px } </style> </head> <body> <div id="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> </body>
```

Output: Dividing each block into 3 fractions



```
<style> #container { display: grid; grid-template-columns: repeat(3,1fr); grid-template-rows: 100px 100px; gap:20px 20px } </style> </head> <body> <div id="container"> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> </div> </body>
```

Output: Dividing each block into 4 fractions