**CSS tricks**

[**1.** **Grid css** 2](#_Toc104465676)

[**2.** **Tạo hình với css clip-path** 3](#_Toc104465677)

[**3.** **Công thức dàn layout với flexbox ko bị rớt hàng và tạo biến để linh động responsive** 4](#_Toc104465678)

[**4.** **Làm icon overlay background cùng color** 5](#_Toc104465679)

[**5.** **So sánh làm Table bằng div – flexbox với <td> <tr>** 6](#_Toc104465680)

1. **Grid css**

Container:

Lun phải có **display:grid**

Có **grid-template-columns**,

Có thể có thêm **grid-template-rows**



**Grid-template-columns**:

* **repeat**: lập lại các giá trị cho các phần tử theo cột.
* **Auto-fit**: tự động tính width để fix với các màn hình.
* **Minmax**: input1: số min: width **ko thể nhỏ hơn** , input2: số max width **ko thể lơn hơn**

(nếu **width màn hình** < **tổng width của các div** thì sẽ xuống hàng)

Vd: 4div, min: 250px.

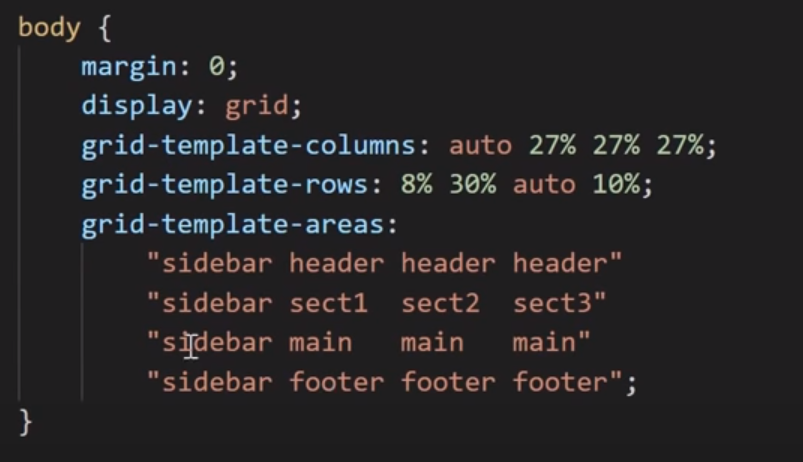
Nếu màn hình từ 1000px trở lên thì 4 div trên 1 row.

Nếu dưới 1000px thì 1 div sẽ rớt xuống row dưới.

Nếu dưới 750px thì 2 div rớt.

Nếu dưới 500px thì 3 div rớt xuống row dưới.

Nên auto-fit sẽ giúp các div lấp khoảng trống của row đều nhau.



**Grid-template-areas**:

Để tạo ra 1 cái bản đồ cho cái layout.

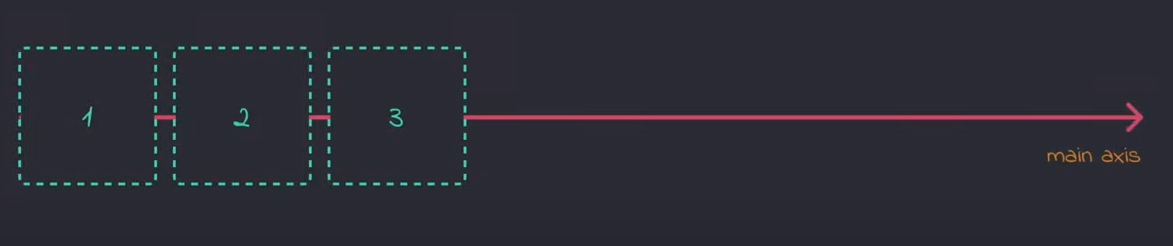
Nhìn rõ ràng, nhưng phải định nghĩa thêm **grid-template-rows** nữa, nên chỉ phù hợp với layout tổng(layout cố định các section)

Khi xuống mobile thì phải định nghĩa lại template này

So sánh flex vs grid css

**Flex**

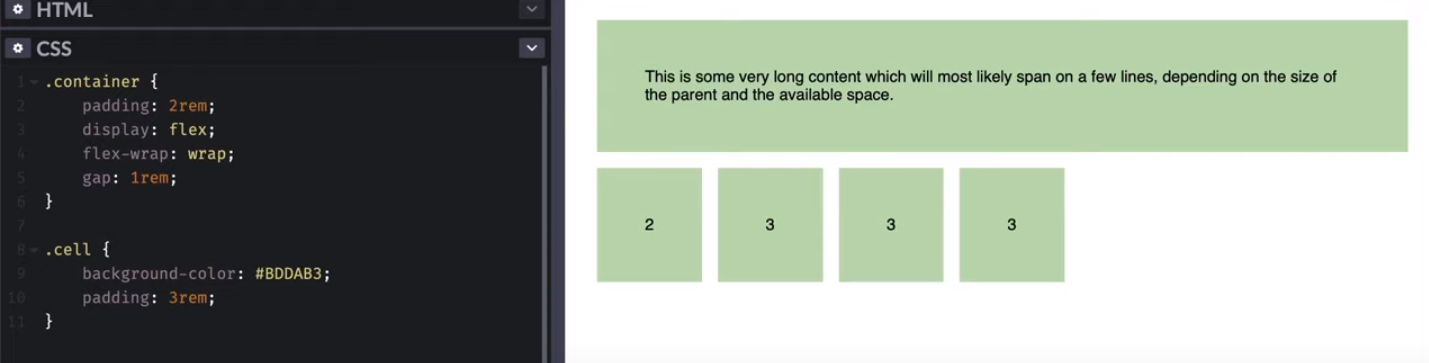
1 direction



Nên muốn đổi hưởng thì phải dùng flex-direction: row/coloumn

Flexbox cũng có thể dùng trong grid system.

**Content shapes the layout** – nội dung định hình layout

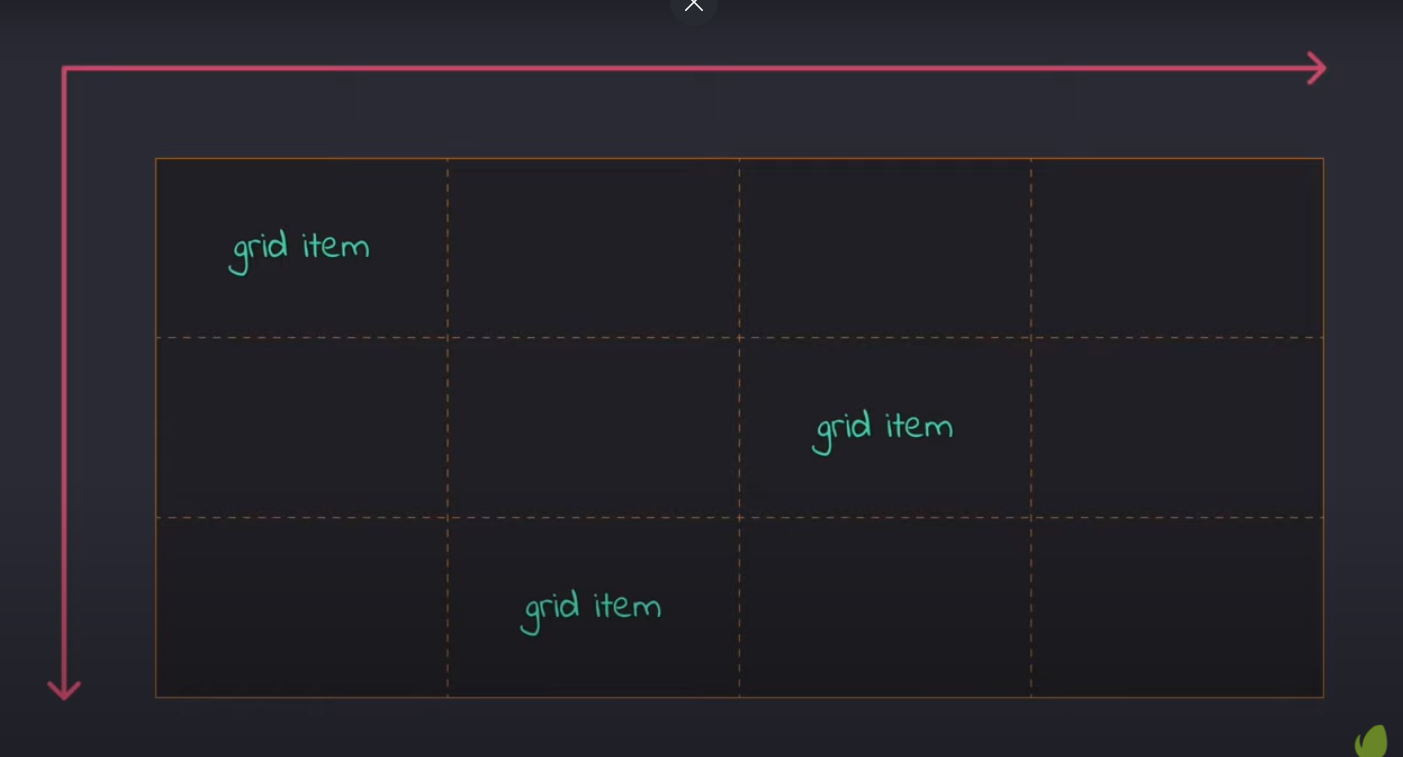


Nội dung được chạy từ trái sang phải cho full width.

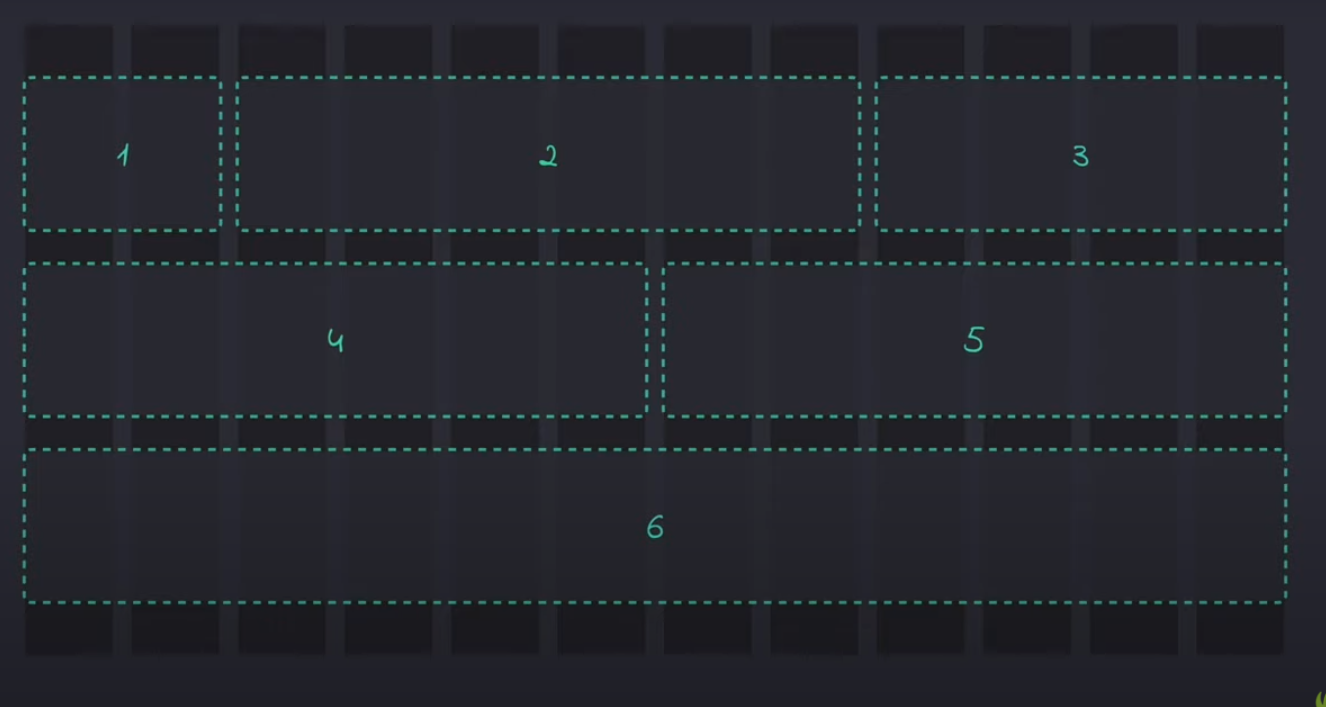
Flex-box ko thể responsive được (phải dùng @media) , nó chỉ wrap các items ở trong thôi. Khi ko width ko đủ chỗ thì flex-box wrap sẽ tự động xuống hàng.

**grid**

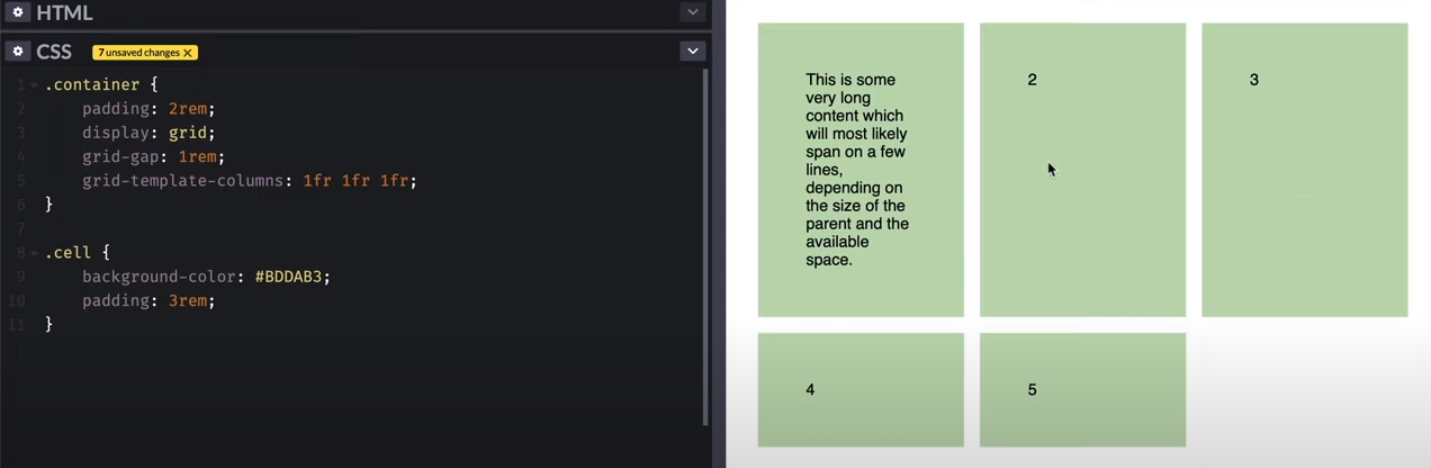
2 directions



12 coloumns grid in grid system



**Layout shapes the content** – layout định hình cho nội dung



Nội dung chỉ chạy ở trong 1 section mà grid đã định nghĩa, ko chạy hết width

Grid-css có thể tự responsive dc(ko dùng @media) với thuộc tính **auto-fit/auto-fill**.

1. **Tạo hình với css clip-path**

[CSS clip-path Editor (codepen.io)](https://codepen.io/stoumann/pen/abZxoOM)

Note: div có gán clip-path thì phải là display: block/inline-block và phải có width, height

Vài hình cơ bản:

clip-path: polygon(100% 0,0 100%,100% 100%); 

clip-path: polygon(0% 0%, 75% 0%, 100% 50%, 75% 100%, 0% 100%); 

clip-path: polygon(100% 0%, 75% 50%, 100% 100%, 25% 100%, 0% 50%, 25% 0%); 

clip-path: polygon(0% 5%,100% 5%,100% 25%,0% 25%,0% 40%,100% 40%,100% 60%,0% 60%,0% 75%,100% 75%,100% 95%,0% 95%); 

clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%); 

clip-path: polygon(20% 0%, 0% 20%, 30% 50%, 0% 80%, 20% 100%, 50% 70%, 80% 100%, 100% 80%, 70% 50%, 100% 20%, 80% 0%, 50% 30%); 

clip-path: polygon(25% 0%, 100% 0%, 75% 100%, 0% 100%);

1. **Công thức dàn layout với flexbox ko bị rớt hàng và tạo biến để linh động responsive**

.content\_\_wrapper{ /\* container \*/

    --col:4;

    --spacing: 25px;

    --width: calc(100% / var(--col));

display: flex;

flex-flow: row wrap;

    margin-left: calc(-1 \* var(--spacing)); /\* dùng số âm để item ko thụt sâu vào trong và để cho đủ chỗ cho các items \*/

}

.content\_\_wrapper\_\_item{ /\* child của wrapper \*/

        flex-basis: calc(calc(var(--width) - var(--spacing)));

        margin-left: var(--spacing);

        margin-bottom: var(--spacing);

    }

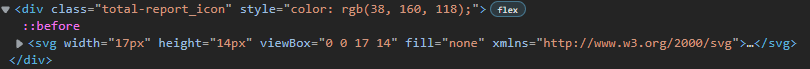
Biến này có thể đặt ở trong thằng Cha. Và các phần tử Child có thể dùng. CSSOM

* Note: giờ có thể dùng property **gap** để thay thế cho margin-left (content\_\_wrapper) và bỏ margin-left, margin-right (content\_\_wrapper\_\_item)

1. **Làm icon overlay background cùng color**

Ý tưởng:



1 div: prop sẽ dynamic **style={{color: ...}}** có border radius chứa ::before và svg

element.style {

    color: #26a076;

}

.total-report\_icon {

    position: relative;

    width: 40px;

    height: 40px;

    display: flex;

    justify-content: center;

    align-items: center;

    border-radius: 50%;

    overflow: hidden;

}

::before có props: **opacity, absolute, background-color: currentColor**

.total-report\_icon::before{

    content: '';

    position: absolute;

    top: 0;

    left: 0;

    bottom: 0;

    right: 0;

    opacity: 0.4;

    background-color: currentColor;

}

Svg props: **fill: currentColor**

.total-report\_icon svg{

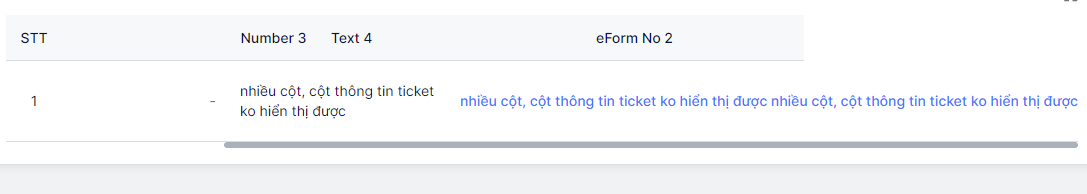
    fill: currentColor;

}

1. **So sánh làm Table bằng div – flexbox với <td> <tr>**

Flexbox

* Dễ tùy chỉnh, responsive dc
* Khó cho việc sticky nếu row quá dài
* Khó custom width của cell bằng nhau
* Cell trên và dưới sẽ ko dãn đều nhau



* Solution: phải lấy dc width của cell dài nhất để đặt cho các cols

<td> <tr>

* Khó responsive
* Khó custom border
* Khó custom cell và nội dung trong cells
* Cell sẽ dài chung với nhau, ko cần set width cho cell