Problem with % units in CSS

Do you know how to use % units in CSS? Do you have any problems using percentages in CSS? Let's find out with TipsMake.com.com!

Do you know **how to use % units in CSS** ? Do you have any problems **using percentages in CSS** ? Let's find out with TipsMake.com.com!

Not too long ago, we relied entirely on percentages for width and height. Using percentages means that your layout and elements can have height and width based on the viewport. However, modern CSS is constantly evolving, even now programmers do not necessarily use % units.

Join TipsMake.com.com to learn about the problems you will encounter when using percentages and modern CSS techniques instead of percentages. They will give you the same results without any hitches.

Example of a simple grid pattern

To illustrate the problem with percentages, consider this HTML layout:

```
?? ?? ?? ??
```

The outer element is a basic **container div with two child divs** inside. Each child has a grid-item class. To turn this container into a grid with 2 columns (two boxes), you need to apply the following CSS code:

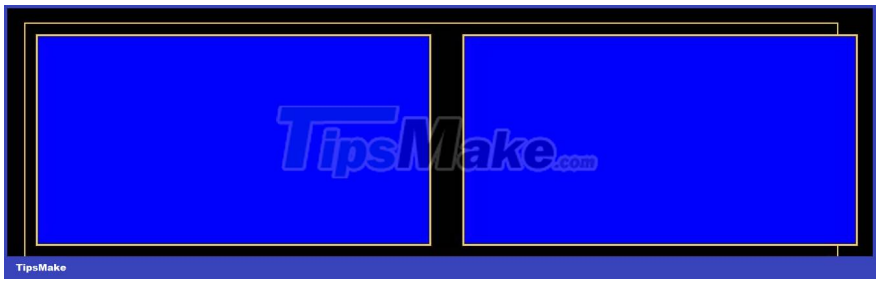
```
body { ??background-color: black; ??align-items: center; ??
justify-content: flex-start; } .my-grid { ??display: grid; ??
grid-template-columns: 50% 50%; ??margin: 3rem; ??border: 2px solid gold;
```

```
??padding: 1rem; } .grid-item { ??border: 3px solid gold; ??
padding: 10rem 0; ??background: blue; }
```

Therefore, each column (grid item) has a yellow background color. On the parent class, set **grid-template-column** to 50% for each column. As a result, both boxes take up 50% of the total width of the container element.

Result:

But there is a problem with this arrangement. First, if you decide to add a space to the parent grid, the child element may overflow. For example, if we added **gap: 3px** to the **.my-grid** block in CSS, its layout would look like this:



As you can see in the image above, the box on the right has been moved outside the container. Sometimes, you might not notice it because the gap is small enough to cause weird alignment errors. However, if there is a larger gap, the overlap becomes quite obvious.

Whenever you are using percentages and adding margins or spacing, there is a high chance that you will encounter such errors. But, why does that error occur?

Caused by each column taking up 50% of the parent element. The example above has 50% plus 50% of that distance (3px), pushing the box outside the container.

Note that this error does not only occur with a 50-50 ratio. You can set the first column to 75%, the second to 25% and the error will still occur. This is why you need to use the following solution more often.

Solution with fractional values

The solution here is to use fractional values ??instead of percentages. So instead of setting the first column to 75% and the second to 50%, you can set the first column to 3fr and the second to 1fr:

```
grid-template-columns: 3fr 1fr
```

This kite maintains the same proportions as in the first example. But the advantage of using fr units is that it uses fractions of the available distances. In this case, the first column will take up 3 parts of the distance, and the second column will take up a part, excluding the distance.

Another advantage of using frs on percentage or other absolute simplicity like px or rem is that you can combine them with fixed values. For example:

```
grid-template-columns: 1fr 10rem;
```

With the above code, you will have a fixed value that never changes according to the screen size. That's because the column on the right will always remain at **10rem** , and the column on the left will take up the remaining space (minus the distance).

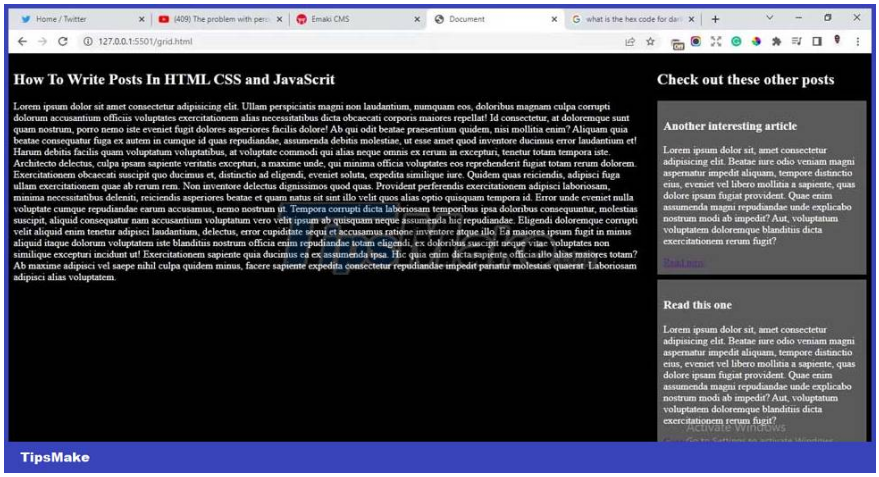
Sometimes you don't need to use percentages. But you have to use them in a smart way, so that they still meet the situation. That usually means pairing them with a value of fr.

Other practical examples

Imagine you have a page consisting of a main content area and a subpage (for related posts). The main content takes up 3 parts of the screen, and the secondary takes up the remaining space minus the distance:

```
.container { ??width: 100%; ??display: grid; ??
grid-template-columns: 3fr 1fr; ??gap: 1.5rem; } .card { ??
background-color: #5A5A5A; ??padding: 10px; ??margin-bottom: .5rem; }
```

Result:



Usually, you will move the sidebar to the bottom (or top) part of the page when the screen becomes too small. This means setting up media queries that stack everything when the view reaches a certain breakpoint.

Here's how you can stack everything into one column when the viewport reaches 55em or less:

```
@media(max-width: 55em) { ??.container { ????display: flex; ????
flex-direction: column; ??} }
```

Result:

Now you don't want each tag to extend the length of the entire viewport. Instead of letting the card show one after another. The best way to achieve this is to use CSS grids. But instead of setting fixed width values ??(like 50%) for the grid-template-column, use the repeat() function like this:

```
.sidebar-grid { ????display: grid; ????
grid-template-columns: repeat(auto-fit, minmax(25rem, 1fr)); ????
align-content: start; ????gap: 2rem; }
```

This CSS sets the minimum size to 25rem and the maximum to 1fr. This approach is much better than setting a fixed width because it relies on the intrinsic size. In other words, it allows the browser to figure things out based

on available parameters.

Now when reducing the browser window to a specific width, the box grid will automatically resize to two boxes per line.

When the screen gets smaller, it has one box per line. So this browser stacks everything on top of everything else. This all happens when you resize the window. You can use a browser feature like Chrome DevTools to understand how CSS works here and how window resizing changes layout.

The best part here is that you don't need a container query or anything fancy to be the response element. Just put in a grid and use min-max() to set fractional values ??instead of fixed sizes.

Above is an alternative to percentage units in CSS. Hope the article is useful to you.