Grid area placement

Story time!

A young man has a well baked cake. He has 3 children asking him to give them some portions. Ideally, who shares the cake?

The young man!

The young man cuts out the areas and gives them to his children.

Here’s why I told that sweet story.

Like the cake, the entire grid area is owned by what element?

The grid container!

Like the young man, the grid container has 3 children too `.aside`, `.main` and `.footer`.

Now the grid container gets to choose how the entire area portions are shared.

One more thing.

Because the children all have names, the young man may say, *“hey Brian, here’s your portion”*, or *“hey Emma, have this”*

It is easy to identify who owns what portion of the cake by assigning the portions to a named individual.

I have no cure for you if you live in a country where people go without names 😁

The grid items all have names! We did so using the `grid-area` property.

Now, let’s share the cake!

The grid-template-areas property

Now the grid container must share the “cake” i.e assign what area portions goes to who.

There are many ways to do what I am about to explain to you, but the `grid-template-areas` property is the easiest to reason about.

It is the bit you need to know for efficiency.

How does the grid-template-areas property work?

Take a look at the code below:

```
body {
      grid-template-areas: "sidebar  content"
                           "footer   footer";
}
```

😳 what the heck is that ?

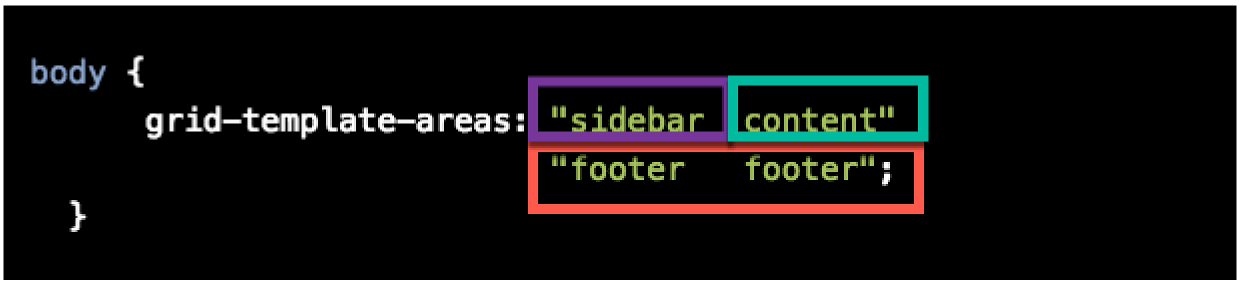
That, my dear is the `grid-template-areas property` in action. There’s no need to get overwhelmed. In this lesson, I will explain how it works—in clear terms.

The grid-template-areas property (what a long property name) provides a very visual structure of the grid.

Take a look at the code again:

```
body {
      grid-template-areas: "sidebar  content"
                           "footer   footer";
  }
```

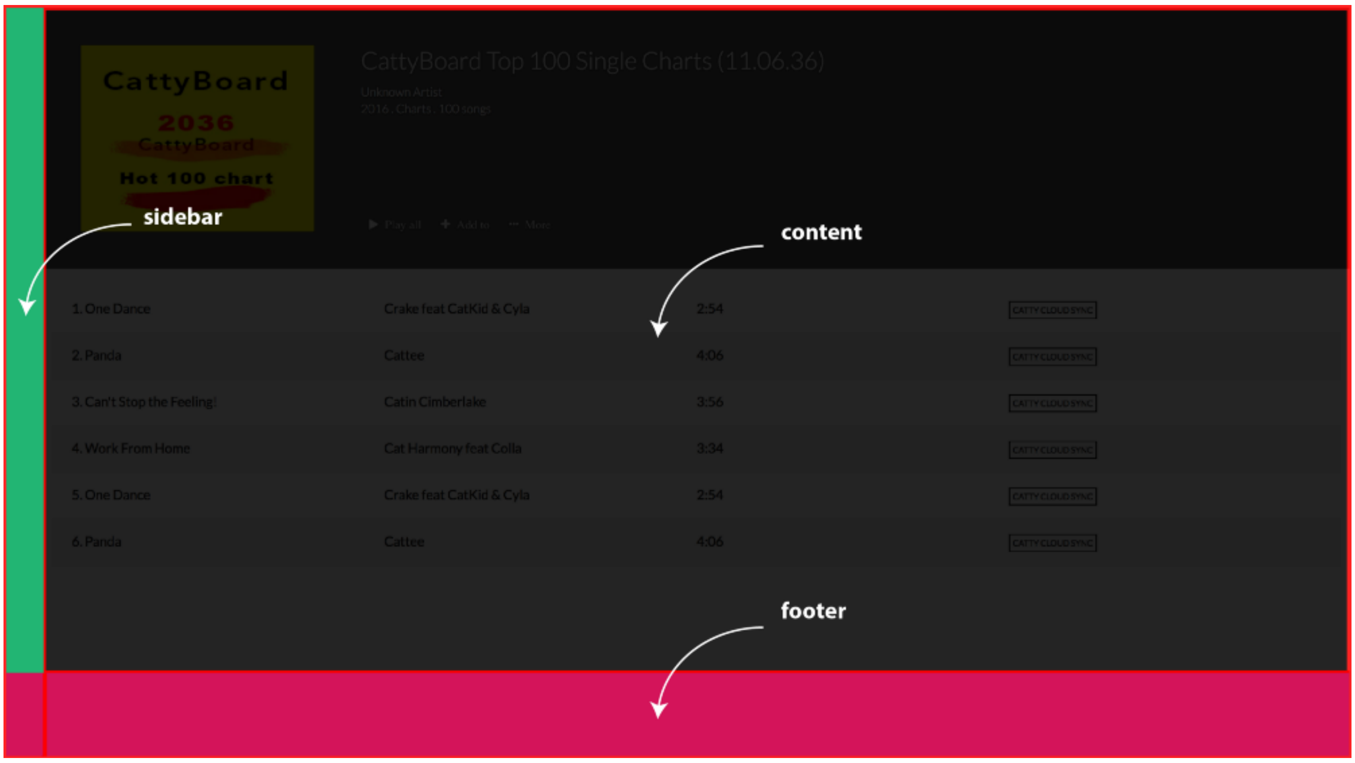
Note that the entries in the property value are the names of the grid items!



```
body {
      grid-template-areas: "sidebar  content"
                           "footer   footer";
  }
```

Look at how the declaration matches the structure of the grid. The footer spanning the entire width with the sidebar and content sitting side by side.

`sidebar`, `content` and `footer` refer to the names of the grid items. The declaration above has succesfully shared the area with respect to the names of the grid items—brilliant!



CattyBoard Top 100 Single Charts (11.06.36)

Unknown Artist
2016 . Charts . 100 songs

1. One Dance	Crake feat CatKid & Cyla	2:54	CATTY CLOUD SYNC
2. Panda	Cattee	4:06	CATTY CLOUD SYNC
3. Can't Stop the Feeling!	Catin Cimberlake	3:56	CATTY CLOUD SYNC
4. Work From Home	Cat Harmony feat Colla	3:34	CATTY CLOUD SYNC
5. One Dance	Crake feat CatKid & Cyla	2:54	CATTY CLOUD SYNC
6. Panda	Cattee	4:06	CATTY CLOUD SYNC

The image above will help you understand how the area portions have been split.

The footer takes the entire bottom area. The sidebar and content take first and second top areas respectively.

Awesome!

At this point here's the complete grid definition we have:

```
body {
      display: grid;
      grid-template-columns: 40px 1fr;
      grid-template-rows: 1fr 90px;
      grid-template-areas: "sidebar  content"
                           "footer  footer";
  }
```

I have added colors for visual aid. The red section represents the `footer`, the other two sections, the `.main` section and `.sidebar`.

The result of that is this:

Output

HTML

CSS (SCSS)

Look! That is a playground 😋

Be sure to check the code tabs above.