My favourite 3 lines of CSS

February 6, 2023 2:18 pm

In Every Layout, we wrote about The Stack. It’s a marriage of Heydon’s Lobotomised Owl selector and my method of managing Flow and Rhythm with CSS Custom Properties.

```
.stack > * + * {
  margin-block-start: 1.5rem;
}
```

Let’s break the selector down: every **direct sibling child** element of `.stack` has `margin-block-start` added to it. This is achieved by the Lobotomised Owl selector, but the `>` combinator is added to prevent margin being added recursively. In writing modes that are left-to-right or right-to-left—such as English or Arabic—the margin is added to the top of the element.

I’ve been obsessed with this particular snippet for *years* now, and in recent years, I’ve enhanced it further with the flow utility:

```
.flow > * + * {
  margin-block-start: var(--flow-space, 1em);
}
```

As you can see, this is very similar to The Stack. The only difference aside from the name, is the use of Custom Properties—specifically the fallback value.

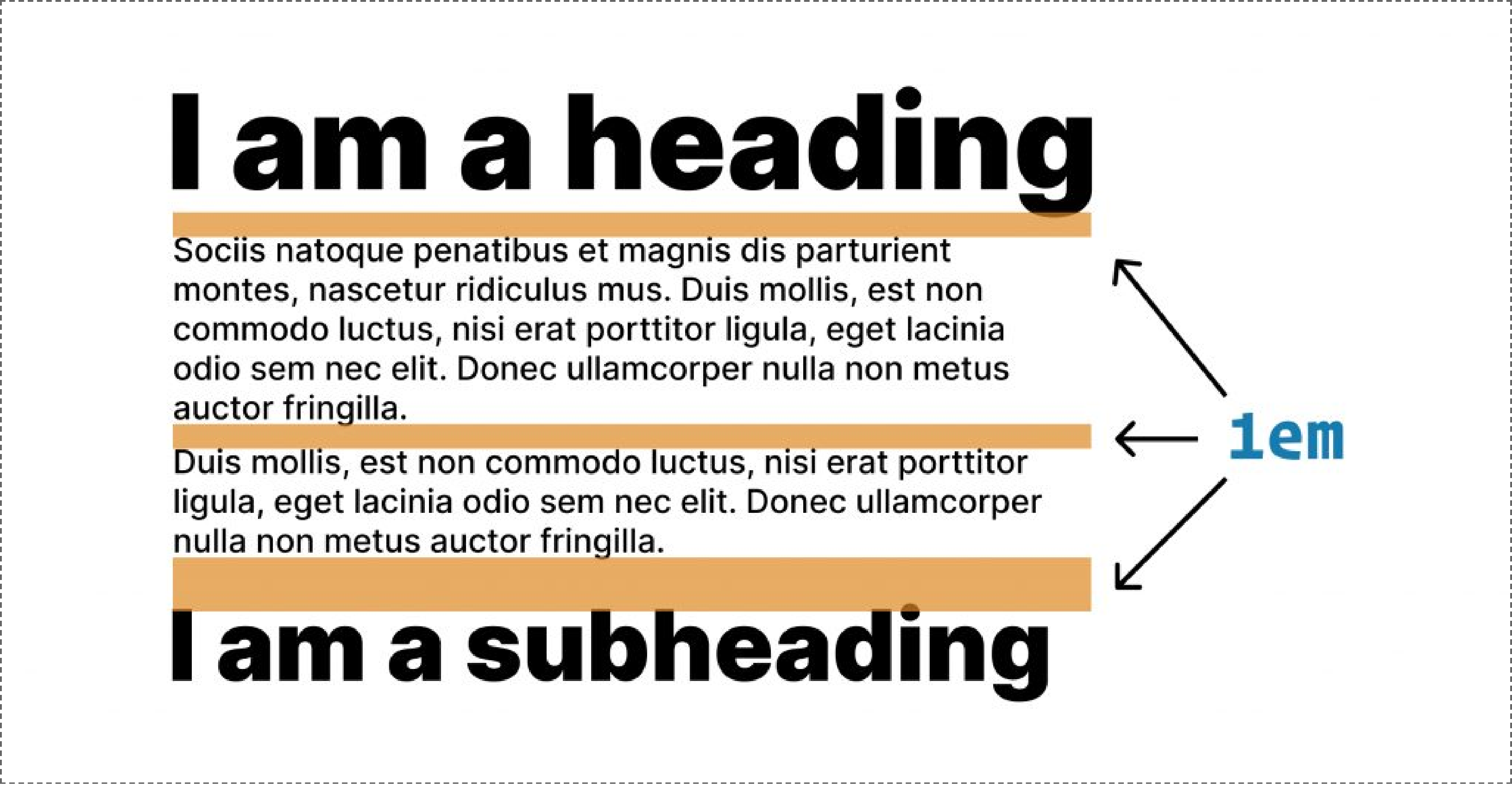
How fallback values work in CSS Custom Properties

If you try to grab a CSS Custom Property value that hasn’t been defined, the initial or inherited value will be used instead, but if the Custom Property is invalid and you don’t provide a fallback, it’ll fail.

In the context of the `.flow` utility, this would be pretty bad, because if you’ve removed element’s default margin, like I do in a reset, you could end up with no space at all.

The fallback value for `.flow` gives us two benefits:

1. There’s a sensible fallback for if `--flow-space` is accidentally set as an invalid value
2. If `--flow-space` doesn’t exist, the `margin-top-value` will be `1em`, which will be relative to the element’s computed font size


I am a heading
Sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.
Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec ullamcorper nulla non metus auctor fringilla.
1em
I am a subheading

The space is 1em by default, so if the font size is larger, there will be more space. This is a handy way of maintaining flow and rhythm if you're using a type scale.

Why do I use flow over The Stack?

I get asked this *a lot* and it's a valid question. The reason is twofold.

Firstly, I use this utility **everywhere** on projects. At our studio, it is in every project we do, and probably will be forever. The reason is that because we use fluid type and fluid space, the `.flow` utility's flexible, fallback-based setup works in harmony with our methodology of letting the browser do all the hard work for us.

Secondly, in contexts like prose content (often long-form content like this article), we like to manage how certain HTML elements space themselves. For example, we might want to reduce the space of elements that directly follow headings and increase the space around `figure` elements.

This snippet is from this site you’re looking at now:

```css
.prose :is(h2 + *, h3 + *, h4 + *) {
  --flow-space: var(--space-s);
}
```

What the snippet does is:

1. Look for elements that directly follow a **h2**, **h3** or **h4**.
2. Assign **--flow-space** with the small item from the spacing scale

As for **.prose** itself, I’ve set the following:

```css
.prose {
  --flow-space: var(--space-m-l);
}
```

By default, I tend to leave **--flow-space** as undefined, but in **.prose** contexts, I like a nice chunky bit of space between elements, so because Custom Properties are affected by specificity and the cascade, I create that more specific value.

With the spacing value used for **--flow-space***—which computes to around* **2.5rem***—we get consistent space between every element.*

Why use margin and not gap?

Again, I get asked this *a lot*. I remember when Safari finally pulled their finger out with **gap**, I sent Heydon a message saying we should make **everything** on Every Layout **gap**, which we were rather excited about. While we were putting actual thought into the second edition—which introduced **gap** to the layouts—we quickly realised we didn't want to change The Stack.

This was partly down to the fact that in order to use **gap**, we would have to make The Stack, either a flex or grid parent. This could cause all sorts of problems for people that grabbed the layout and dropped it in existing projects. This makes up a very large portion of the people who use Every Layout in the real world.

Along with that, all of the control is put on the parent and not the elements themselves. Back in the context of **.flow** and **--flow-space** with a fallback value: this wouldn't be possible with **gap**. This is because **gap** is set on the parent (**.flow**) and controls the space between child elements. The parent is in complete control and the child elements have no say in what **gap** is at any given moment.

Lastly, along with the flexibility of letting **--flow-space** be affected by the cascade, and if it's not set, letting **.flow** inherit spacing values from the child element it affects. The other key reason I don't like to use **gap** is Logical Properties. I personally want everything we build at the studio to use Logical Properties as much as possible, because you get much better multi-language and reading mode support out of the box. They're really well supported now, too!

Wrapping up

I honestly should have retired after making this iteration of **.flow**:

```
.flow > * + * {
    margin-block-start: var(--flow-space, 1em);
}
```

Especially now that we use that and the rest of the layouts from Every Layout at Set Studio. We're essentially just colouring in front-ends at this point—apart from the odd occasion when we create a complex specific layout, where we use CSS grid layout to its full potential.

The main reason I’ve written this post though, is to send people to it when they ask why I use **`.flow`** or use **`margin`**, but also, I hope you’ve seen how damn powerful CSS Custom Properties are. They’re certainly more than just CSS variables, that’s for sure.

+ Liked by Yann Brelière Factorial.io / ,🤡 @factorial_io@social.factorial.io, Roland Franke and **164 others**

👋 Hello, I’m Andy and I’ll help you build fast & visually stunning websites.

I’m the founder of Set Studio, a creative agency that specialises in building stunning websites that work for **everyone**. If you’ve got a project in mind, get in touch.

Back to blog

RSS Feed My Setup Set Studio