Does CSS Grid Replace Flexbox?

BY **ROBIN RENDLE** ON MARCH 31, 2017
FLEXBOX, **GRID**

No. Well. Mostly No.

Grid is much newer than Flexbox and has a bit less browser support. That's why it makes perfect sense if people are wondering if CSS grid is here to replace Flexbox.

To put a point on it:

1. Grid can do things Flexbox can't do.
2. Flexbox can do things Grid can't do.
3. They can work together: a grid item can be a flexbox container. A flex item can be a grid container.

Even though Grid is pretty new, we have lots of articles about it, including a getting started article, an article about a basic layout done multiple ways, and a complete guide.

If you hadn't heard the trumpets blaring, March 2017 was the banner month for Grid. It was released, completely unprefixed and ready-to-go, in Chrome, Opera, Firefox, and Safari. Even Edge supports it, albiet an older version of the spec, which you can get some support for by using Autoprefixer.

So.

> Does Grid replace Flexbox?

It seems a little complicated at first. Especially if you're just starting to get a grip on the weird, alien syntax of Flexbox. Now there is ***a whole other*** syntax to learn? Sheesh.

Here's some things Grid is ***specifically*** better at than Flexbox:

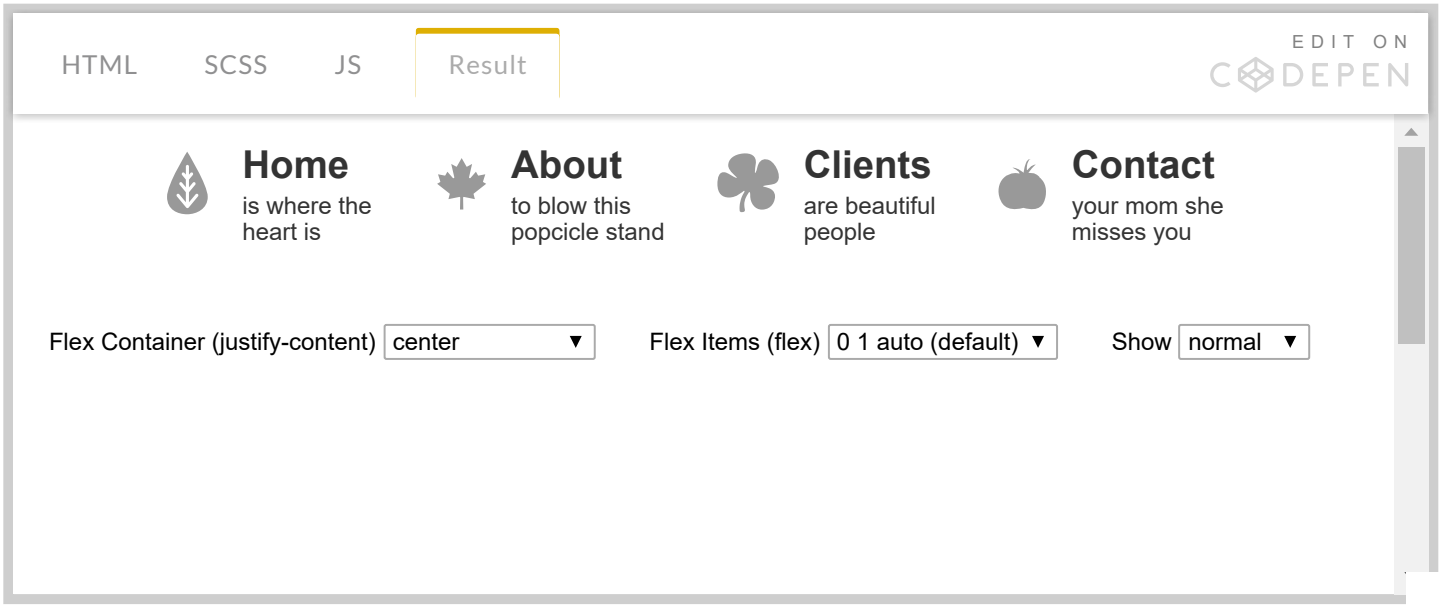
- Making whole page layouts. It's better for that even just considering layout performance reasons.
- Making literal grids. Like X columns with Y gap between them homegrown framework stuff. `grid-gap` is wonderful, as gutters are the main pain point of grid systems.
- Requiring less media query intervention with really powerful functionality like auto layout, `minmax()`, `repeat()`, and `auto-fill`. Here's a neat idea.

And another huge one: Grid can position elements in 2 dimensions. Both columns and rows. That's a first. Rachel Andrew once made that very clear:

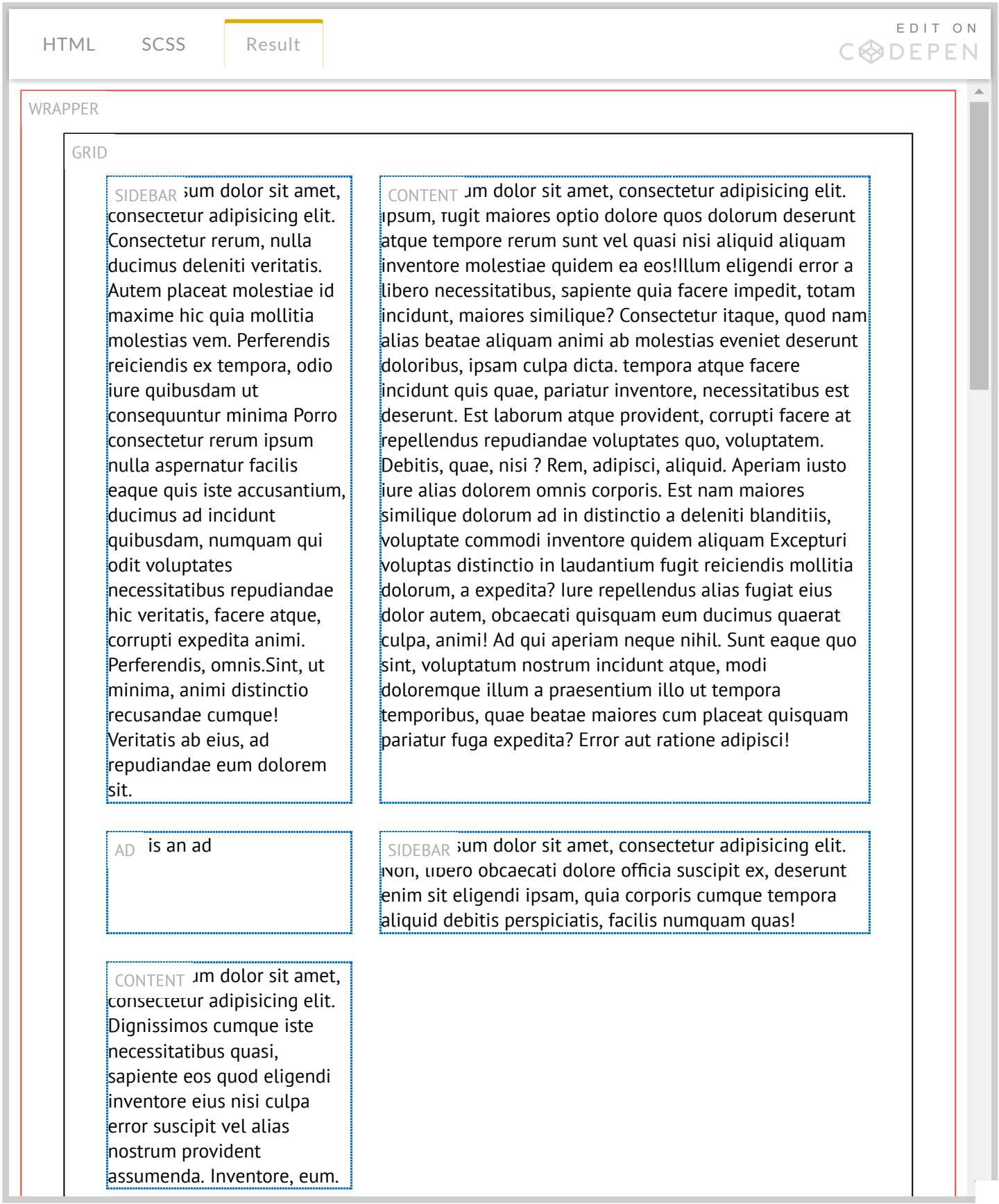
> Flexbox is essentially for laying out items in a single dimension – in a row OR a column. Grid is for layout of items in two dimensions – rows AND columns.

Let's see some demos.

Say we're building a horizontal navigation component — that's the perfect use case for Flexbox because it sets content in only one direction. In Chris' demo below you can mess around with those properties and see just how powerful Flexbox is:



But there are some instances where those flexbox properties, such as `justify-content` or `align-items`, should be used in conjunction with Grid properties. Take this demo for instance:



First we've made a .wrapper that is set to `display: flex;` . That way we can set a max-width on our `.grid` and use `justify-content: center;` to place it in the middle of the viewport. Then we can make our grid with the right number of columns:

css

```
.wrapper {
  display: flex;
  justify-content: center;
}

.grid {
  display: grid;
  max-width: 800px;
  grid-template-columns: 1fr 2fr;
}
```

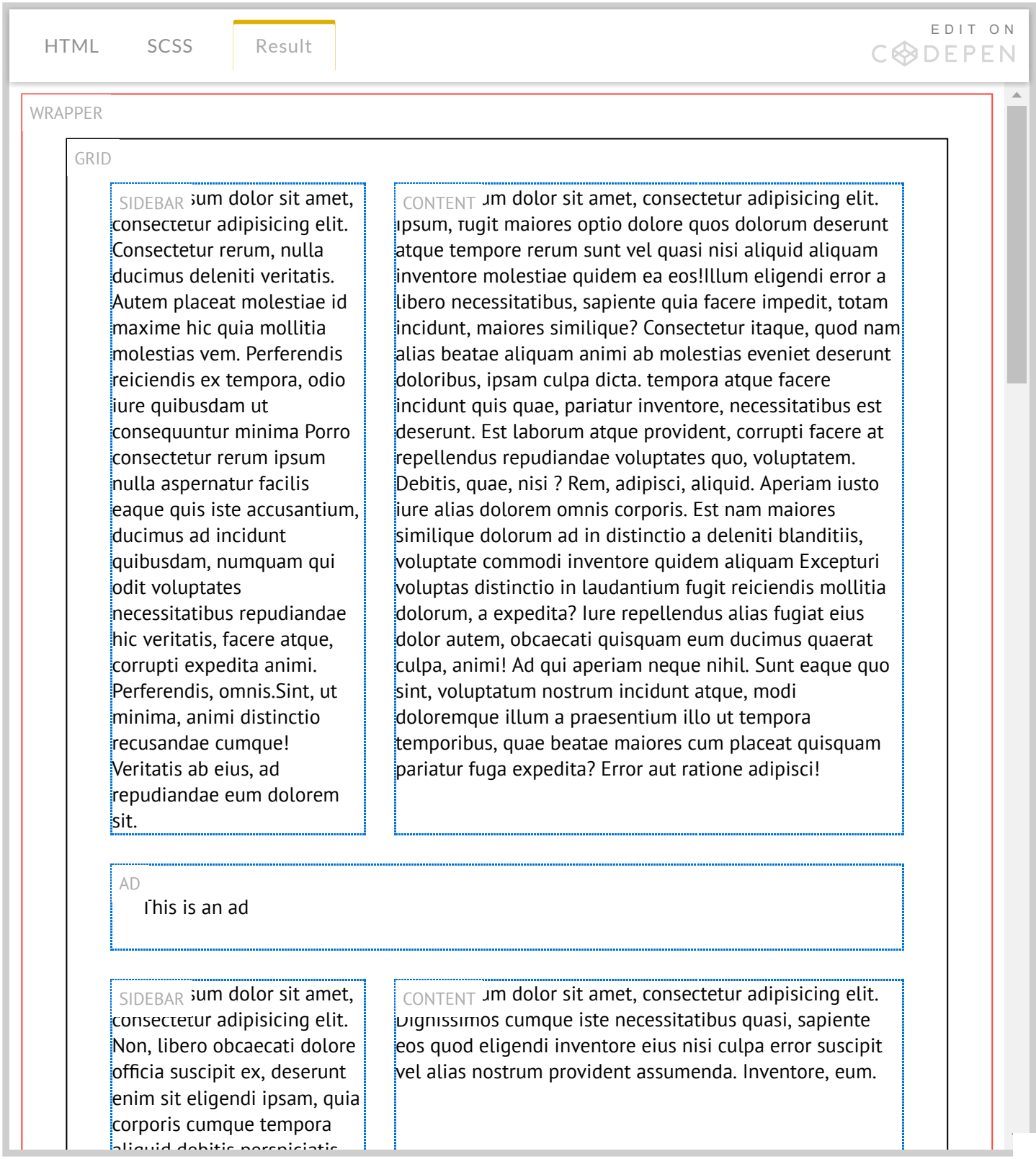
That's just the first step.

Now let's make that `.ad` take up the whole row of our grid. Well, we can do that by specifically targeting our `<div>` without messing up the rest of the children of our grid:

css

```
.ad {
  grid-column-start: 1;
  grid-column-end: 3;
}
```

What we're saying with the code above is *"start this div on the first column and end on the very last column."* That should look something like this:



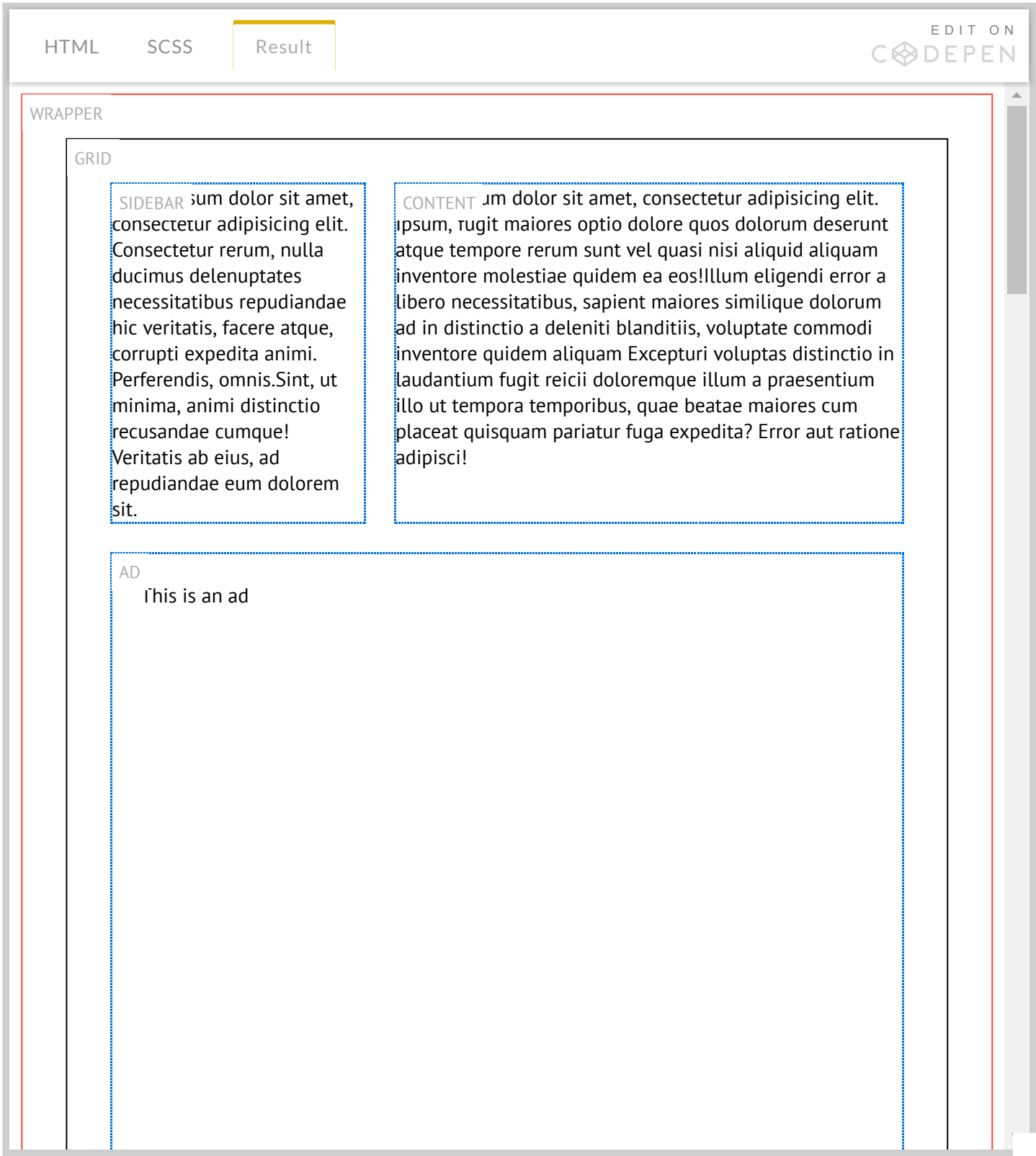
Further, let's say that we always wanted our ad block to be twice the size of our first row – we can totally do that! We just need to use the `grid-template-rows` property:

CSS

```
.grid {
  display: grid;
  max-width: 800px;
  grid-template-columns: 1fr 2fr;
```

```
  grid-template-rows: 1fr 2fr 1fr;
}
```

That's the equivalent of saying: *"there are three rows in our grid. Always make sure that the second row is twice the size of the first and third."* In other words, we're creating relationships between rows and other rows whilst also defining the number of columns in our grid!



With CSS Grid we can set relationships horizontally (with `grid-template-columns`) and vertically (with `grid-template-rows`) but at the same time. Flexbox, on the other hand, is stuck

doing either vertical or horizontal layouts (with `flex-direction`) – but that doesn't mean we should ditch it.

Potentially Confusing: a "2D" Layout with Flexbox

What can get a little confusing is that it's not impossible to make multi-dimensional layouts in just Flexbox. For example:

HTML SCSS Result EDIT ON CODEPEN

```scss
.layout {
  display: flex;
  flex-wrap: wrap;
}
.child {
  min-height: 100px;
  background: orange;
  flex: calc(100% / 3);
  text-align: center;
  line-height: 100px;
  border: 5px solid white;
}

* {
  box-sizing: border-box;
}
```

a a a

a a a

a

There are certainly rows and columns there, even a final element that spans multiple columns. It's easy to build and flexible. It's done through allowing `flex-wrap` on the container, having the children's `flex-basis` be 1/3 of the width of the container, and allowing `flex-grow` to stretch children if need by. (The `flex` property is shorthand for `flex-grow`, `flex-shrink`, and `flex-basis`.)

It's not wrong, it's just one way of doing something like this, and you could probably make an argument for it being less intuitive and adaptable.

Potentially Confusing: a "1D" Layout with Grid

Grid *can* do 2D layout, meaning defining both `grid-template-rows` and `grid-template-columns`, but it doesn't have to use both. Here's a demo of using it just to lay boxes in a horizontal row:

HTML SCSS Result EDIT ON CODEPEN

```scss
.layout {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  // same as:
  // grid-template-columns: repeat(3, 1fr);
  grid-gap: 8px;
}
.child {
  min-height: 100px;
  background: orange;
  line-height: 100px;
  text-align: center;
}
```

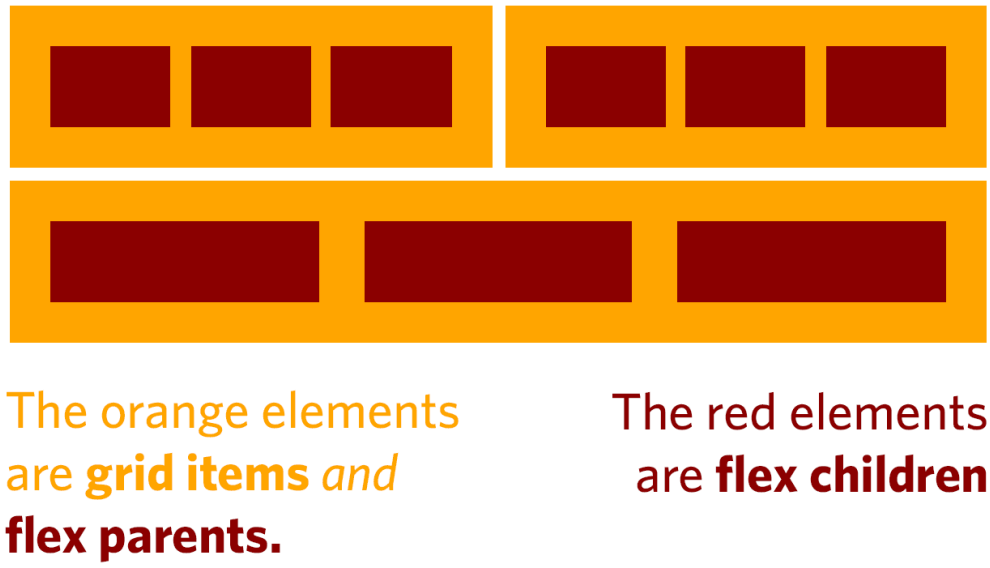
a a a

This isn't wrong either. In fact, you could easily make the argument that this is easier to understand and more succinctly expressed than doing it with flexbox. For example, no layout properties are needed on the child elements at all. But you could also argue that purely 1D layout like this is more powerful in Flexbox, because Flexbox allows us to move those elements around easier (e.g. move them all to one side or another, change their order, space them out evenly, etc).

About That Nesting

One more time to lock it in!

This is good to remember: grid items can be flex parents.

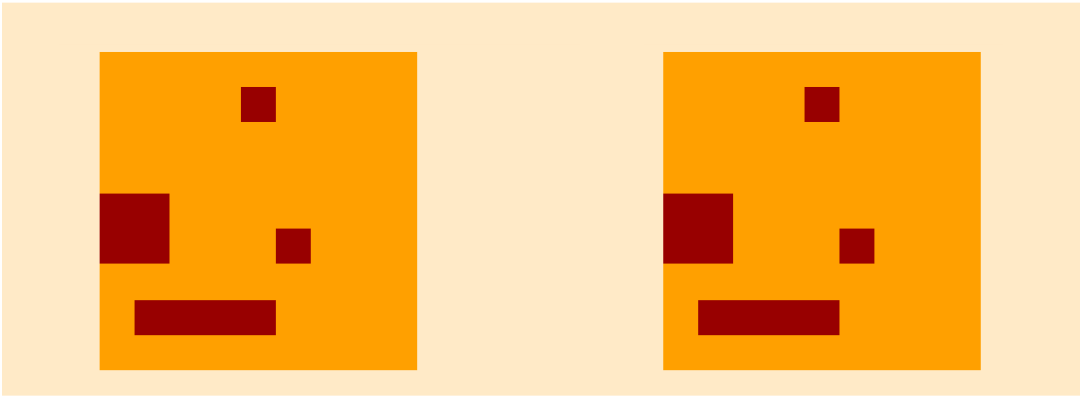


HTML SCSS Result EDIT ON CODEPEN

```scss
.layout {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 8px;
}
.child {
  background: orange;
  text-align: center;
  padding: 25px;

  display: flex;
  justify-content: space-between;
}
```

And, flex items can be grid parents:



HTML SCSS Result

EDIT ON CODEPEN

```scss
.layout {
  display: flex;
  justify-content: space-around;
}

.child {
  width: 200px;
  height: 200px;
  background: orange;

  display: grid;
  grid-template-columns: repeat(9, 1fr);
  grid-template-rows:    repeat(9, 1fr);
```

Comments

Rachel Andrew

APRIL 2, 2017

"Potentially Confusing: a "2D" Layout with Flexbox" – that isn't a 2d layout. It's a wrapped flex layout. As soon as you want to make that final box line up with the boxes in the first column you realise what you have isn't two-dimensional.

Flex wrapping doesn't make it two dimensional. Each row (in your example) is a flex container itself, space distribution happens across each row individually. Which is why it isn't two-dimensional.

More here https://rachelandrew.co.uk/archives/2017/03/31/grid-is-all-about-the-container/

You can also space grid tracks out evenly and so on, box alignment is shared by both the flexbox and grid specifications. For examples of that see http://gridbyexample.com/video/align-grid/

Charles

APRIL 3, 2017

Hence the scare quotes on "2D".

Rachel Andrew

APRIL 4, 2017

Perhaps just me but I feel it would be better to provide correct info in the text than infer something else with punctuation.

Josh Habdas

APRIL 5, 2017

Flexbox still doesn't even properly support `flex-flow` column wrap for creating multi-column dropdown menus with a constrained height. And now this. Get ready for a wild ride! https://github.com/philipwalton/flexbugs

Curtis Schofield

APRIL 5, 2017

I am pretty confused by the introduction of "2D" ..

Thank you for the other reference , however, I am now 2x confused. :P

David

APRIL 12, 2017

So what exactly can Flexbox do that Grid can't ?

Can you elaborate on "Flexbox allows us to move those elements around easier (e.g. move them all to one side or another, change their order, space them out evenly, etc)" ?
How ? I though doing that with grid was easy too.
