masonry layouts with CSS

Masonry layouts are interesting and a challenge to do with existing web Javascript since there's no way to do it natively with HTML and/or CSS.

> Masonry is a grid layout based on columns. Unlike other grid layouts, it doesn't have fixed height rows. Basically, Masonry layout optimizes the use of space inside the web page by reducing any unnecessary gaps. Without this type of layout, certain restrictions are required to maintain the structure of layout.
>
> From: Understanding Masonry Layout — Sitepoint (2014)

The earliest example I remember of masonry layout is the Pinterest home page.

Home Today Following

How are you planning for Diwali?

Get ideas

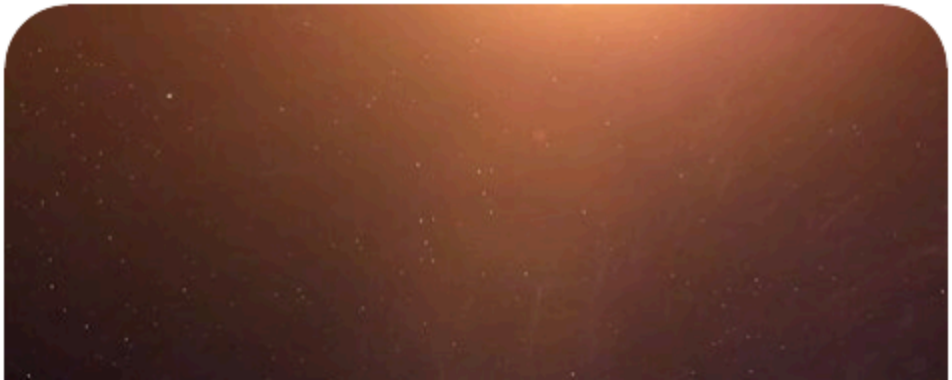

Figure 1: Pinterest home page showing masonry layout where the images will display next to each other regardless of height

I was surprised to see that the CSS Working Group's CSS Grid Layout Module Level 3 defines a masonry layout based on CSS Grid.

Rachel Andrew wrote Native CSS Masonry Layout In CSS Grid about grid-based masonry and its current support in Firefox. It made me wonder how will it work and how can we implement a fallback for when it doesn't. The code has only been implemented in Firefox as of writing this post, and it looks like a promising solution to the challenge of creating masonry layouts.

At its most fundamental level, grid-based masonry requires us to tell the browser what axis we want to use for masonry and use the `masonry` value for the opposite axis. The most common case is to use rows for the masonry axis, like in this example

```
.container {
  display: grid;
  gap: 10px;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: masonry;
}
```

This will produce the layout we want. All content packed up together, regardless of the height of the elements displayed.

What I like the most about this solution is that it allows other aspects of CSS grid to work together with masonry. Using Rachel's article as an example, we can have some items span more than one column using code like this:

```
.wide-2 {
  grid-column-end: span 2;
}
```

And assign the class to the child elements that we want to span two columns in our grid. Note that this will cause other elements of our grid to be repositioned to think carefully if you want to use this technique.

We can also use masonry as the value of `grid-template-columns` in which case it causes overlap and a different result to what you probaly expect. There is more overlap and the space is not used as efficiently as in the row masonry layout.

Don't forget that you have to format the children element so that they will fit within the space the grid assigns them.

Fallbacks and Alternatives

If you're working with CSS, you can use something like this to only use native masonry grids in browsers that support it. You will have to provide an alternative layout for browsers that don't, like a grid layout without masonry.

```
@supports (grid-template-rows: masonry) {
  /* masonry code here */
}
```

If the masonry layout is important to the design you will have to use a Javascript solution like like Masonry and query for support using JavaScript's CSS.support, something like the following code:

```
const supportsMasonry = CSS.supports('grid-template-rows', 'masonry');

if (supportsMasonry) {
  console.log('Native masonry is supported, do nothing');
} else {
  console.log('Native masonry not supported');
  console.log('Loading alternative library');
```

```
  const newScript = document.createElement("script");
  newScript.src = "https://unpkg.com/masonry-layout@4.2.2/dist/masonry.pkg
  document.body.appendChild(newScript);
}
```

Then initialize the Masonry library with something like this:

```
const elem = document.querySelector('.grid');
const msnry = new Masonry( elem, {
  // options
  itemSelector: '.grid-item',
  columnWidth: 200
});
'.grid-item'
```

For the the other steps you need to complete to configure Masonry check the library's getting started documentation.

Conclusions

When (and if) more browsers decide to implement the specification, masonry will have a huge impact on design and it will reduce our dependency on Javascript APIs to do layout. Rachel's article goes into more detail about the interaction between masonry and other parts of the grid specification that may or may not be working in Firefox's implementation. Testing and playing with the spec as it is today means you get the chance to provide feedback and change the spec to acommodate your needs.