CSS grid wrapping

Asked 2 years, 7 months ago Active 14 days ago Viewed 56k times

82

33

Is it possible to make a CSS grid wrap without using media queries?

In my case, I have a non-deterministic number of items that I want placed in a grid and I want that grid to wrap. Using Flexbox, I'm unable to reliably space things nicely. I'd like to avoid a bunch of media queries too.

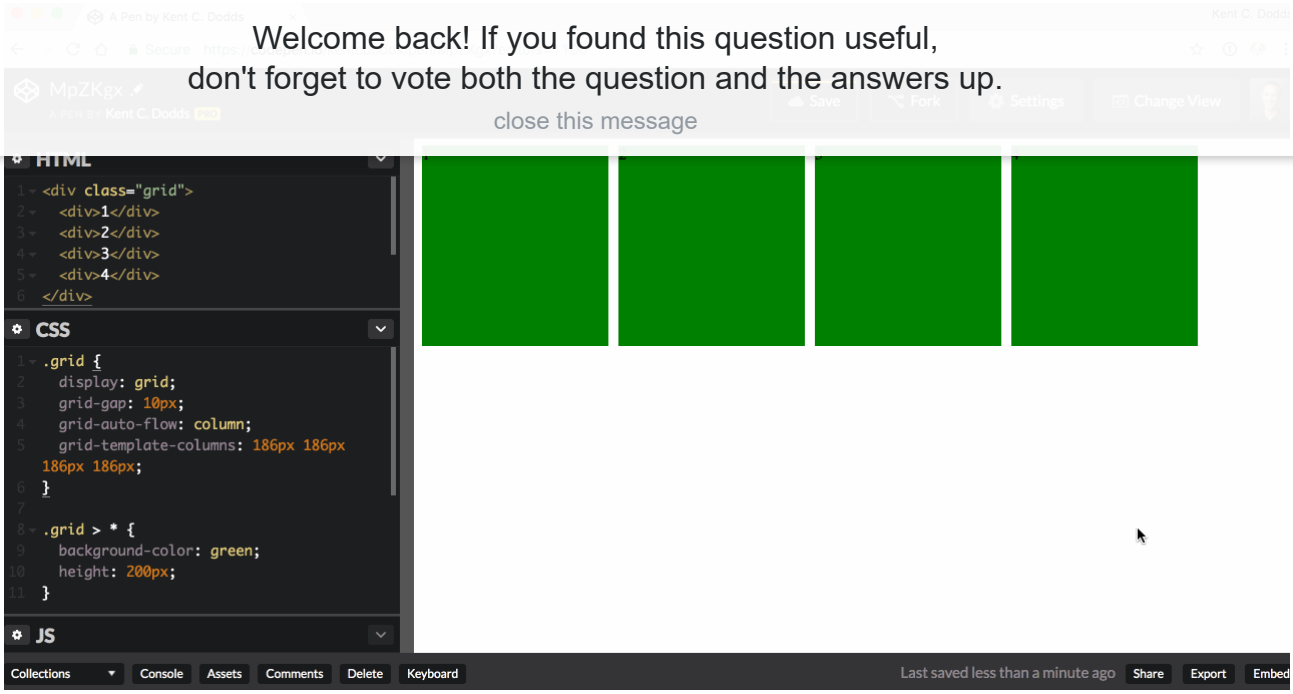
Here's some sample code:

```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-auto-flow: column;
  grid-template-columns: 186px 186px 186px 186px;
}

.grid > * {
  background-color: green;
  height: 200px;
}
```

```
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

Run code snippet Copy snippet to answer Expand snippet

And here's a GIF image:



As a side-note, if anyone can tell me how I could avoid specifying the width of all the items like I am with `grid-template-columns` that would be great. I'd prefer the children to specify their own width.

html css css-grid

edited Nov 3 at 12:30
Peter Mortensen
24.1k ● 19 ● 89 ● 118

asked Mar 30 '17 at 22:21
kentcdodds
16.6k ● 23 ● 89 ● 159

1 `grid-template-columns: auto auto auto auto;` does work in this case? =) – Jakub Chlebowicz Mar 30 '17 at 22:26

5 Answers

140

Use either `auto-fill` or `auto-fit` as the repetition number of the `repeat()` notation.

```
repeat( [ <positive-integer> | auto-fill | auto-fit ] , <track-list> )
```

`auto-fill`

> When `auto-fill` is given as the repetition number, if the grid container has a definite size or max size in the relevant axis, then the number of repetitions is the largest possible positive integer that does not cause the grid to overflow its grid container.

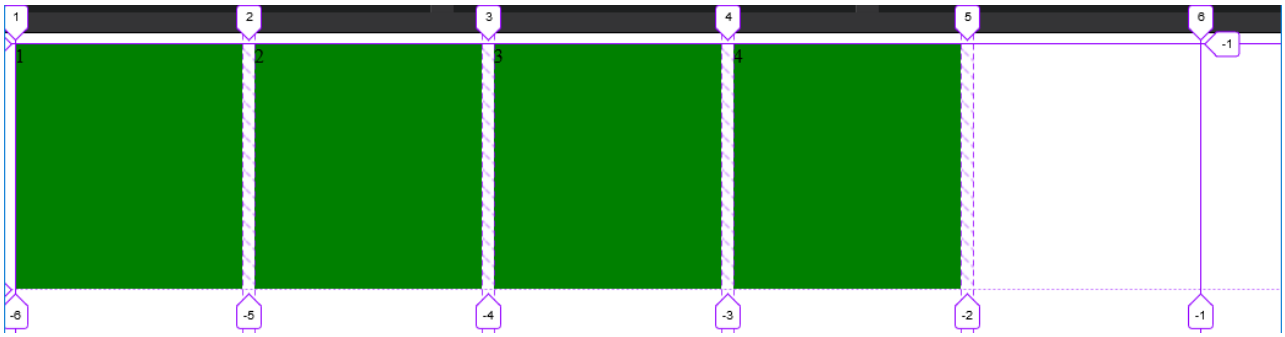
```
.grid {
  display: grid;
  grid-gap: 10px;
```

```
  height: 200px;
}
```

```
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

Run code snippet | Copy snippet to answer | Expand snippet

The grid will repeat as many tracks as possible without overflowing its container.



In this case, given the example above *(see image)*, only 5 tracks can fit the grid-container without overflowing. There are only 4 items in our grid, so a fifth one is created as an empty track within the remaining space.

The rest of the remaining space, track #6, ends the explicit grid. This means there was not enough space to place another track.

`auto-fit`

The `auto-fit` keyword behaves the same as `auto-fill`, except that after grid item placement algorithm any empty tracks within the remaining space will be collapsed to `0`.

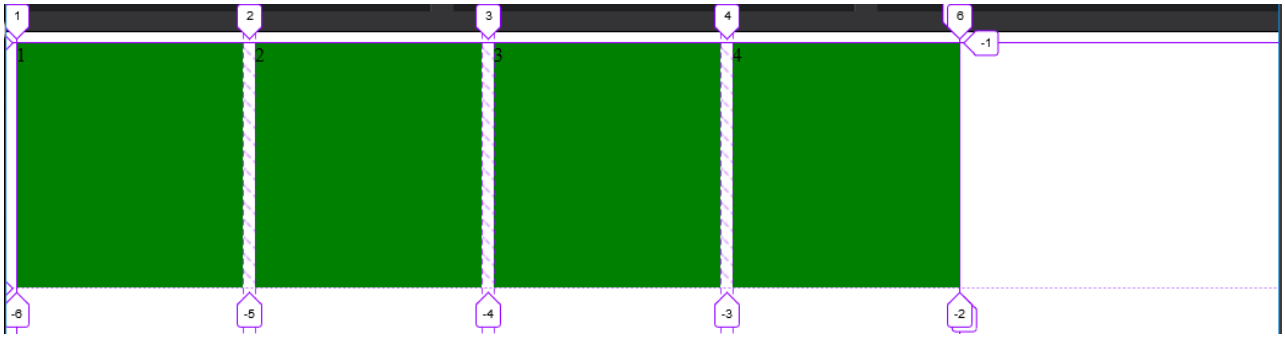
```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(auto-fit, 186px);
}

.grid>* {
  background-color: green;
  height: 200px;
}
```

```
<div class="grid">
  <div>1</div>
  <div>2</div>
```

Run code snippet | Copy snippet to answer | Expand snippet

The grid will still repeat as many tracks as possible without overflowing its container, but the empty tracks will be collapsed to `0`.

A collapsed track is treated as having a fixed track sizing function of `0px`.



Unlike the `auto-fill` image example, the empty fifth track is collapsed, ending the explicit grid right after the 4th item.

`auto-fill` VS `auto-fit`

The difference between the two is noticeable when the `minmax()` function is used.

Use `minmax(186px, 1fr)` to range the items from `186px` to `186px` plus a fraction of the leftover space in the grid container.

When using `auto-fill`, the items will grow once there is no space to place empty tracks.

```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(auto-fill, minmax(186px, 1fr));
}

.grid>* {
  background-color: green;
  height: 200px;
}
```

```
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

Run code snippet | Copy snippet to answer | Expand snippet

```css
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(auto-fit, minmax(186px, 1fr));
}

.grid>* {
  background-color: green;
  height: 200px;
}
```

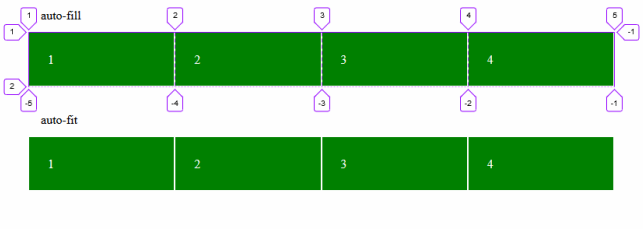
```html
<div class="grid">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
</div>
```

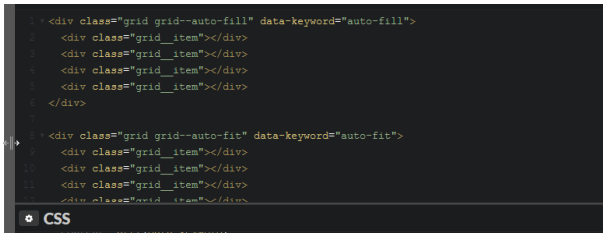
Run code snippet | Copy snippet to answer | Expand snippet

Playground:

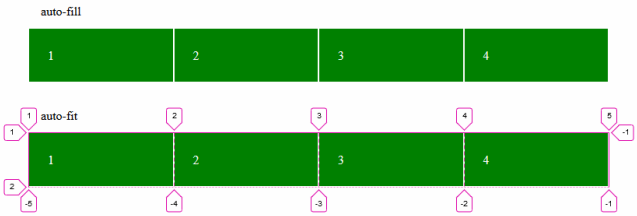
CodePen

Inspecting auto-fill tracks





Inspecting auto-fit tracks



edited Sep 19 '18 at 21:20

answered Mar 30 '17 at 22:38

Ricky
13.1k ● 4 ● 26 ● 38

1 Is there any way to make it center the ones that are on the next line? – kentcdodds Mar 30 '17 at 23:00

Just like you would with flex box, on the `display: grid` element use `justify-content: center` – Spittal Oct 4 '17 at 21:33

Dear @Ricky How to make it so that the first parameter of minmax, eg. `repeat(auto-fill, minmax(186px, 1fr));` is not pixels, but just as long as the div has text inside? – mesqueeb Apr 28 '18 at 4:11

1 @mesqueeb It is not possible, a definite size is needed. You can take a look at this answer for more details. – Ricky Jun 19 '18 at 20:12

1 Is there any way to make it so that when it has to go to the next line, two of the items go down instead of only one? So like from 4 to 2 to 1 and not have it 4 to 3 to 2 to 1? – sammiepls Jan 30 at 8:12

14

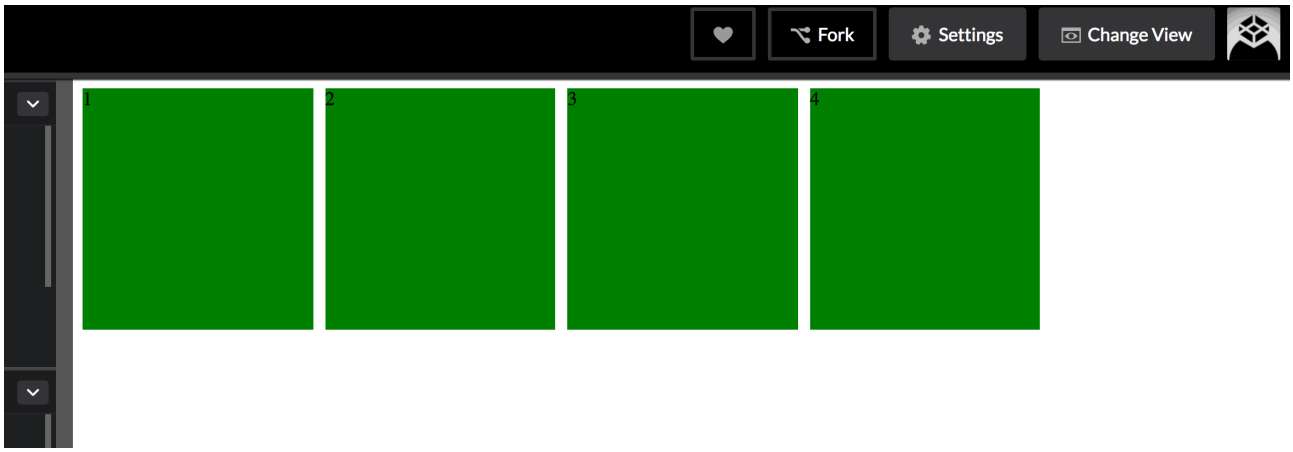
You want either `auto-fit` or `auto-fill` inside the `repeat()` function:

```
grid-template-columns: repeat(auto-fit, 186px);
```

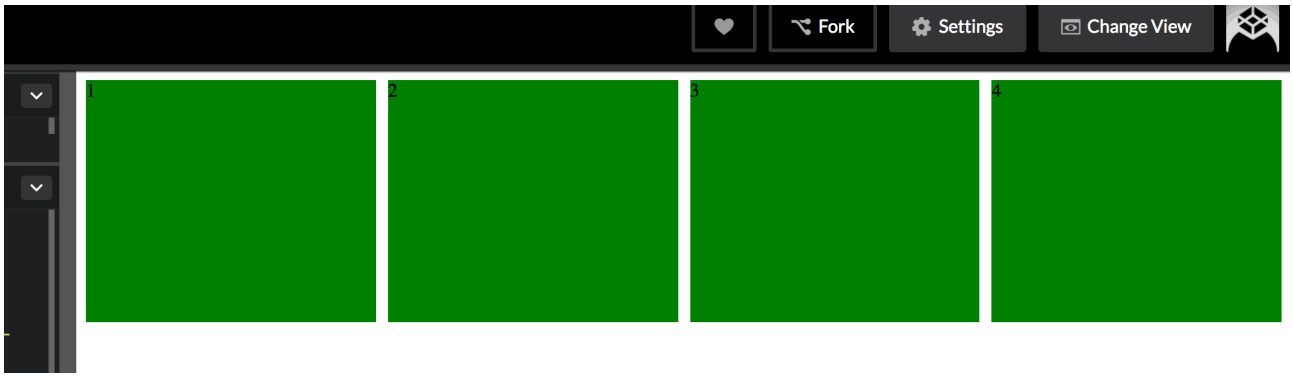
The difference between the two becomes apparent if you also use a `minmax()` to allow for flexible column sizes:

```
grid-template-columns: repeat(auto-fill, minmax(186px, 1fr));
```

This allows your columns to flex in size, ranging from 186 pixels to equal-width columns stretching across the full width of the container. `auto-fill` will create as many columns as will fit in the width. If, say, five columns fit, even though you have only four grid items, there will be a fifth empty column:



Using `auto-fit` instead will prevent empty columns, stretching yours further if necessary:



edited Nov 3 at 12:37 answered Mar 30 '17 at 23:52

6

You may be looking for `auto-fill`:

```
grid-template-columns: repeat(auto-fill, 186px);
```

Demo: http://codepen.io/alanbuchanan/pen/wJRMox

To use up the available space more efficiently, you could use `minmax`, and pass in `auto` as the second argument:

```
grid-template-columns: repeat(auto-fill, minmax(186px, auto));
```

Demo: http://codepen.io/alanbuchanan/pen/jBXWLR

If you don't want the empty columns, you could use `auto-fit` instead of `auto-fill`.

edited Nov 3 at 12:35

Peter Mortensen
24.1k ● 19 ● 89 ● 118

answered Mar 30 '17 at 22:37

alanbuchanan
2,475 ● 2 ● 22 ● 48

2 Is there any way to make it center the ones that are on the next line? – kentcdodds Mar 30 '17 at 23:00

2

I had a similar situation. On top of what you did, I wanted to center my columns in the container while not allowing empty columns to for them left or right:

```
.grid {
    display: grid;
    grid-gap: 10px;
    justify-content: center;
    grid-template-columns: repeat(auto-fit, minmax(200px, auto));
}
```

answered Mar 28 at 15:54

farrellw
527 ● 7 ● 7

Re *"while not allowing empty columns to for them"*: Do you mean *"while not allowing empty columns **either** for them"* (*to* → *too* → *either*)? Or *"while not allowing empty columns for them"* (without *to*)? Or something else (it doesn't seem to compute)? – Peter Mortensen Nov 3 at 12:44

0

Here's my attempt. *Excuse the fluff,* I was feeling extra creative.

My method is a parent `div` with **fixed dimensions**. The rest is just fitting the content inside that div accordingly.

This will **rescale** the images regardless of the aspect ratio. There will be **no hard cropping** either.

```
body {
    background: #131418;
    text-align: center;
    margin: 0 auto;
}

.my-image-parent {
    display: inline-block;
    width: 300px;
    height: 300px;
    line-height: 300px; /* Should match your div height */
    text-align: center;
    font-size: 0;
}

/* Start demonstration background fluff */
    .bg1 {background: url(https://unsplash.it/801/799);}
    .bg2 {background: url(https://unsplash.it/799/800);}
    .bg3 {background: url(https://unsplash.it/800/799);}
    .bg4 {background: url(https://unsplash.it/801/801);}
    .bg5 {background: url(https://unsplash.it/802/800);}
    .bg6 {background: url(https://unsplash.it/800/802);}
    .bg7 {background: url(https://unsplash.it/802/802);}
    .bg8 {background: url(https://unsplash.it/803/800);}
    .bg9 {background: url(https://unsplash.it/800/803);}
    .bg10 {background: url(https://unsplash.it/803/803);}
    .bg11 {background: url(https://unsplash.it/803/799);}
    .bg12 {background: url(https://unsplash.it/799/803);}
    .bg13 {background: url(https://unsplash.it/806/799);}
    .bg14 {background: url(https://unsplash.it/805/799);}
    .bg15 {background: url(https://unsplash.it/798/804);}
    .bg16 {background: url(https://unsplash.it/804/799);}
    .bg17 {background: url(https://unsplash.it/804/804);}
    .bg18 {background: url(https://unsplash.it/799/804);}
    .bg19 {background: url(https://unsplash.it/798/803);}
    .bg20 {background: url(https://unsplash.it/803/797);}
/* end demonstration background fluff */

.my-image {
    width: auto;
    height: 100%;
    vertical-align: middle;
    background-size: contain;
    background-position: center;
    background-repeat: no-repeat;
}
```

```
<div class="my-image-parent">
    <div class="my-image bg1"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg2"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg3"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg4"></div>
</div>
```

```
    <div class="my-image bg6"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg7"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg8"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg9"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg10"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg11"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg12"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg13"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg14"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg15"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg16"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg17"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg18"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg19"></div>
</div>

<div class="my-image-parent">
    <div class="my-image bg20"></div>
</div>
```