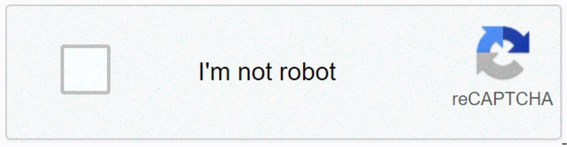

Continue

Grid- template- columns internet explorer

This is the first in a three-part series all on how to use CSS grid in a way that will work not only in modern browsers, but also in Internet Explorer (IE). Imagine writing CSS grid code without having to write an alternative layout! Many of us believe that this is a distant future that is many years away. If the only thing that holds you back from reality is IE11 (check caniuse.com), then you're in luck! This day is today! Or at least it will be when you finish reading this series. ☺ To get started, this first part is going to demystify some common misunderstandings around the native application of IE11's grid. In part 2, I'm going to blow the lid off the myth that using CSS grid in IE11 is extremely difficult. No more crappy alternative layout for IE11! In part 3, I'll show you a cool flexbox technique that I use to create simple auto-positioning grids. These fake auto-positioning grids have fixed pixel-based grid gaps that align perfectly with those created using a real CSS grid. They work perfectly in IE11, fully respond, and updating their column count based on screen width requires only changing a width value in a media query. That's enough intro lets you get started on these misunderstandings! IE has an indirect grid in the CSS grid, the explicit grid is what you manually define with all grid-template-* properties. The indirect grid is how the browser handles placing cells placed outside the explicit grid. When you use css grid in modern browsers, the indirect grid is pretty obvious, since the grid will continue to create and place the grid cells automatically in new rows without having to set them. IE doesn't have automatic placement, so it's easy to assume that IE doesn't have a silent grid. It does though – you just need to be a little clearer when it comes to using it. Refer to the Explicit Pen vs. Daniel Tonon (@daniel-tonon) grid in CodePen. The most common use case for using the IE indirect grid is to use it to create your rows for you. If you want all lines in the grid to be dictated by the height of their content, you don't have to bother setting -ms-grid-rows (the IE version of the grid-standard-lines). Rows will be created automatically for you in IE when you place grid cells. An important thing to note though is that modern browsers have access to the property grid-auto-series and grid-auto-columns. These properties allow you to control how the program handles the size of rows and columns in the indirect grid. IE has no meaning of these properties. Indirect IE rows and columns can only ever be sorted as automatic. IE has repeat function Don't be afraid! There is no need to write out 1fr 20px 12 times for your precious 12-column grid (IE does not have native grid-gap support). IE comes prepackaged with full iteration() functionality. He's hiding behind a different pension. The modern syntax for repeating values in columns and rows is repeated(12, 1fr 20px) which means repeat 1fr 1fr pattern 12 times. The IE version of the syntax is (1fr 20px)[12]. The IE version has identical functionality, just a different syntax. /* This grid... */ .grid-one { appearance: -ms-grid? display: grid; -ms-grid-columns: 1fr 20px 1fr 20px 1fr 1fr 1fr 1fr 1fr; grid-standard-columns: 1fr 20px 1f 20px 1f is exactly the same as this grid: */ .grid-two { display: -ms-grid? display: grid; /* IE repeat syntax */ -ms-grid-columns: 1fr (20px 1fr)[3]; /* Modern recurring syntax */ grid-template-columns: 1fr repeat(3, 20px 1fr); } In addition to syntax, there is only one difference between how modern browsers and ie implement the repeat() function. Since IE does not support automatic placement, these keywords are meaningless enough to IE so avoid using them. minmax() is natively supported in IE minmax() is a css sizing mode that is exclusive to css grid (currently , anyway). Its functionality is quite self-evident. You can give it a minimum value in the first parameter and a maximum value in the second parameter. The column or row to which this is applied can then shrink and expand between these two values as the available space increases and decreases. Try changing the size of the code below to see it in action. See Daniel Tonon's demo pen minmax(@daniel-tone) in CodePen. People often think that this awesome little feature is not supported in IE, but is actually inherently supported. I guess this is due to at least one of these two reasons: It's cool functionality and they think IE can't have cool things like this because IE sucks. Everyone has seen this magic code snippet and had their dreams shattered when they realized it wouldn't work in IE: grid-standard-columns: repeat (auto-fit, minmax (250px, 1fr)). (If you've never seen that clip before, watch this short video and prepare to have your mind blown.) Since the magic quote doesn't work in IE, people probably assume that nothing in the quote is IE-friendly. In fact, the only reason why the code snippet is not replicated in IE is because IE does not support automatic keyword application and auto-placement. minimal content and max-content are both 100% IE friendly IE has full native support for both minimal content and max-content values. minimum content is a keyword value that shrinks the column/row down to the minimum possible width at which the content can shrink. If applied to a this essentially means that the column will be the width of the longest word. See the demo with Daniel Tonon's minimal pen content (@daniel-tonon) in CodePen. Maximum content, on the other hand, will increase the column/row to the maximum possible width that its content occupies and no further. If you've ever set up the space: now read a large portion of the text, it will appear very similar. See Daniel Tonon's demo max-content pen (@daniel-tonon) on CodePen. I didn't expect IE to support these values mainly mainly logic one below minmax(). I was glad I was surprised when it turned out wrong while investigating IE11 network support. Combined with minmax (), there aren't many grids that a modern browser can do that IE can't handle (as long as the grids don't include auto-positioning). fit-content() is not IE friendly, but ... fit-content() does not work inherently in IE. ☹ Fortunately, fit-content() is kind of a shorthand syntax and when you write it out the long way, IE supports it! 🎉 The long way to write it is by applying automatic (or more specifically, minmax (minimum content, maximum content)) to the column/row in the grid template, and then setting maximum width: [value] to the child element. Here is an example of how you can use the fit-content() function in a modern browser: /* This is not IE-friendly */ .grid { display: grid; grid-template-columns: 100px fit-content(300px) 1fr; } .cell { grid-column: 2; } What fit-content() is basically saying here is, makes the middle column assume the maximum possible width of its content (i.e. maximum content value) until it reaches 300px at what point, do not grow unless forced to. Refer to the modern example 1 of /1 of /90>>) of Daniel Tonon's (@daniel-tonon) Pen fit-content() file in CodePen. If you're reading this on mobile, check out this Codepen demos section in landscape orientation for a better understanding. In order for IE to behave in the same way, you must write the code like this: /* IE-friendly 'fit-content()' alternative */ .grid { display: -ms-grid; display: grid; -ms-grid-columns: 100px auto 1fr; grid-template-columns: 100px auto 1fr; } .cell { -ms-grid-column: 2; grid-column: 2; max-width: 30px; } See the IE hack fit-content pen by Daniel Tonon (@daniel-tonon) in CodePen. Note that this is not a foolproof method of reproducing customization content() in IE. If there is another grid cell that has content that takes up more width than the maximum width setting of the other cell, the grid column will expand to fit the largest cell. It will not be fastened to 300px as it would with fit-content (). See the hack of Pen Broken customization content by Daniel Tonon (@daniel-tonon) in CodePen. Compare this to the modern fit-content() mode that tightens the entire column: See the modern example of /fit-content() pen() by Daniel Tonon (@daniel-tonon) in CodePen. While I'm on the subject, I need to clear up a common misconception around the application-content () function itself. The misconception is that the column (or row) to which it is applied can never exceed the value you provided the function. That's not the way it is. If a column is set to width (300px) and a grid cell within this column is set to 400px width, the column width will be 400px, not 300px, as many people can expect. See the modern pen broken customization content by Daniel Tonon (@daniel-tonon) on CodePen. IE auto != Modern auto The automatic value in IE behaves a little differently than auto in modern browsers. In IE, automatic strictly equals minmax (minimum content, max-content). A column or row row automatic in IE can never shrink any less than the minimum content. Also, it can never grow larger than the maximum content. Modern tours treat the automatic price a little differently. For the most part, when the price is used on its own, they still treat auto like minmax (min-content, max-content). There's a small but critical difference though: auto in modern browsers is able to stretch out alignment-content and justify-content properties. Ie does not allow this. When automatic column size is used in a modern browser, automatic behavior is more like 1fr if there is not enough content to fill the column. IE does not. IE will always shrink the column down to the size of the maximum content. So if you have this code that defines your grid: .grid { appearance: -ms-grid; appearance: grid; -ms-grid-columns: auto auto; grid-standard-columns: auto auto; auto auto; auto you will end up with it in modern browsers: Modern browser auto columns width with little content. ... and this in IE: IE car width columns with little content. Columns shrink in content. Okay, the modern browser looks a lot like what we get when we use 1fr. Can we use minmax (min-content, 1fr) for IE then? This will stretch it out as it does in modern browsers won't it? Yes, but then I run on this issue: .grid { display: -ms-grid; appearance: grid; -ms-grid-columns: minmax(min-content, 1fr) minmax (min-content, 1fr); grid-standard-columns: auto auto; } Modern browsers: Modern browser auto-width columns with a lot of content in a cell. Columns have different widths. IE: IE minmax (min-content, 1fr) columns with multiple content in a cell. Columns have equal widths. FYI, minmax (minimum content, 1fr) is essentially 1fr and 1fr and does not equal auto. I also tried minmax (minimum content, 100%) in IE, but it just led to the same looking grid using 1fr. As far as I can tell, it is not possible to replicate the modern function of the auto browser in IE. There is another crucial difference between IE and modern versions of auto value though. Since the IE version of auto explicitly equals minmax (minimum content, maximum content), auto cannot be used in minmax() expressions. minmax() cannot be used within another minmax() function so automatic is excluded. Modern browsers are a little more thin. They recognize that automatic can mean different things in different contexts. If used on its own, it is essentially equal to minmax (minimum content, but with the added ability to stretch. When used as a maximum value, the automatic is the same as the maximum content. When used as a minimum value, it is essentially equal to a minimum content. If, at this time, you swear never to automatically use grid templates again, you may want to reconsider. There are only three cases where the automatic will behave differently between modern browsers and IE. These are: auto is used in grid-standard-columns without a fr drive in the same statement. are automatically used in grid template series without a FR unit in the same statement and the height of the grid is from the height of the grid cells. auto is used in a minmax() mode. That's it, that's it! These are the only times that automatic will behave differently in IE about how it behaves in modern browsers. auto is much easier to write than minmax (min-content, max-content), so it's not really worth condemning over a few minor browser inconsistencies that are easily avoidable. (Update)