



@rachelandrew

All Day Hey! Leeds, 2018

Desbloqueo

Unlocking the Power of CSS Grid Layout

TABLE OF CONTENTS

| | |
|---|---|
| 1 | Introduction |
| 2 | Subgrids <ul style="list-style-type: none">2.1 Establishing a Subgrid: Per-Axis Proposal2.2 Characteristics of a Subgrid Item2.3 Subgrid Sizing Algorithm |
| 3 | Aspect-ratio-controlled Gutters <ul style="list-style-type: none">Conformance<ul style="list-style-type: none">Document conventionsConformance classesRequirements for Responsible Implementation of CSS<ul style="list-style-type: none">Partial ImplementationsImplementations of Unstable and Proprietary FeaturesImplementations of CR-level FeaturesIndex<ul style="list-style-type: none">Terms defined by this specificationTerms defined by reference |

References

[Normative References](#)

Property Index

Issues Index

CSS Grid Layout Module Level 2

W3C Working Draft, 27 April 2018



This version:

<https://www.w3.org/TR/2018/WD-css-grid-2-20180427/>

Latest published version:

<https://www.w3.org/TR/css-grid-2/>

Editor's Draft:

<https://drafts.csswg.org/css-grid-2/>

Previous Versions:

<https://www.w3.org/TR/2018/WD-css-grid-2-20180206/>

Issue Tracking:

[Inline In Spec](#)

[GitHub Issues](#)

Editors:

[Tab Atkins Jr.](#) (Google)

[Elika J. Etemad / fantasai](#) (Invited Expert)

[Rossen Atanassov](#) (Microsoft)

Copyright © 2018 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C [liability](#), [trademark](#) and [permissive document license](#) rules apply.

Abstract

This CSS module defines a two-dimensional grid-based layout system, optimized for user interface design. In the grid layout model, the children of a grid container can be positioned into arbitrary slots in a predefined flexible or fixed-size layout grid. Level 2 expands Grid by adding “subgrid” capabilities for nested grids to participate in the sizing of their parent grids; and aspect-ratio-controlled gutters.

[CSS](#) is a language for describing the rendering of structured documents (such as HTML and XML) on screen, on paper, in speech, etc.

How big is that box?



ALISTAPART



New Drupal or WordPress project? Get the fastest hosting with HTTPS and global CDN included. Develop for free, pay when you go live. [Learn more](#)

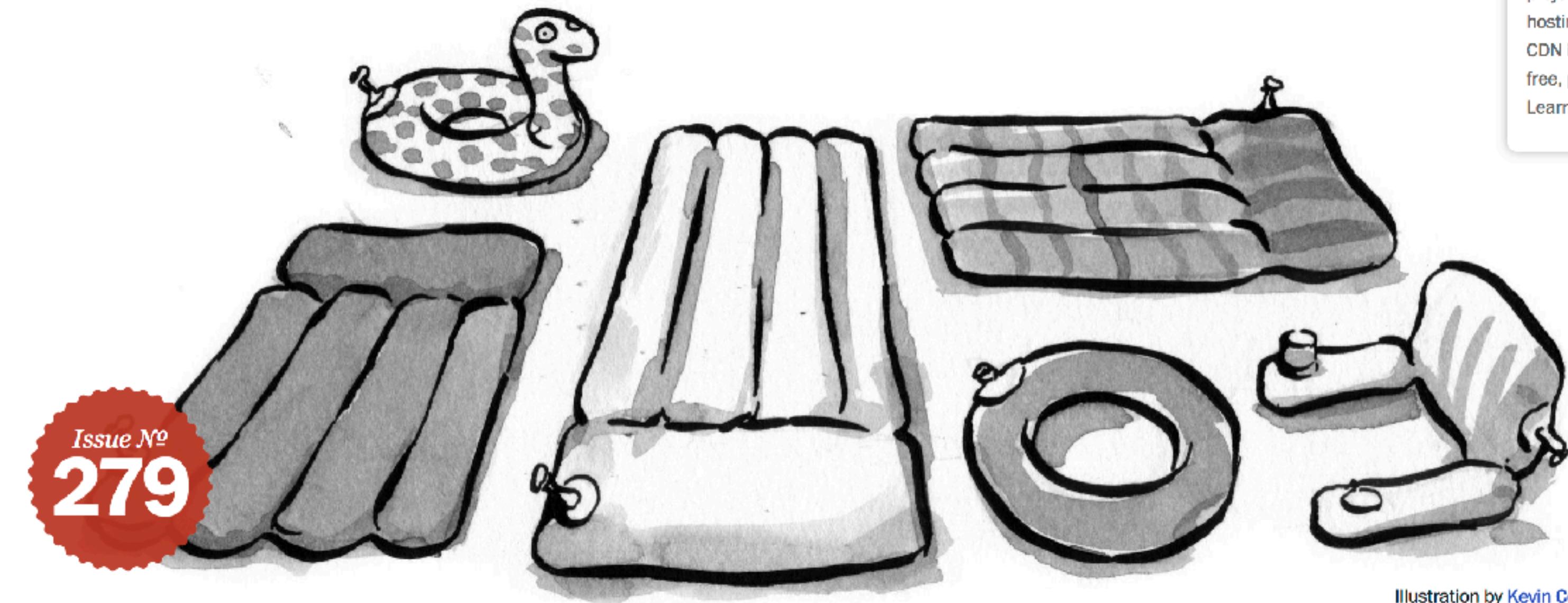


Illustration by [Kevin Cornell](#)

Fluid Grids

by [Ethan Marcotte](#) · March 03, 2009

Published in [CSS](#), [HTML](#), [Graphic Design](#), [Layout & Grids](#), [Responsive Design](#)

Early last year, I worked on the redesign of a rather content-heavy website. Design requirements were fairly light: the client asked us to keep the organization's existing logo and to improve the dense typography and increase legibility. So, early on in the design process, we spent a sizable amount of time planning a well-defined grid for a library of content modules.



Devine
@devine_lowest

Devine - Digital Design & Development, a 3 year bachelor degree powered by howest.be

📍 Kortrijk
🔗 devine.be
📅 Joined January 2010

Devine
@devine_lowest

Follow

Today we are celebrating International box-sizing Awareness Day. Anyone wants some padding? 🎂 #devinehowest @chriscoyer @Real_CSS_Tricks @paul_irish css-tricks.com/international- ...



4:11 AM - 1 Feb 2018 from Kortrijk, België

25 Retweets 58 Likes

3 25 58

powered by:

Devine howest
DESIGN > DEVELOPMENT

© 2018 Twitter About Help Center Terms
Privacy policy Cookies Ads info



css layout frameworks



All

Images

Videos

News

Maps

More

Settings

Tools

About 12,200,000 results (0.60 seconds)

Bulma: a modern CSS framework based on Flexbox

<https://bulma.io/> ▾

A single element for a Metro UI-style **CSS grid**. Vertical... Top tile ...tiles. Bottom tile. Middle tile. With an image. Wide tile. Aligned with the right tile. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ornare magna eros, eu pellentesque tortor vestibulum ut. Maecenas non massa sem. Etiam finibus odio quis feugiat ...

[Docs](#) · [Breadcrumb](#) · [Box](#) · [Container](#)

Skeleton: Responsive CSS Boilerplate

getskeleton.com/ ▾

You should use Skeleton if you're embarking on a smaller project or just don't feel like you need all the utility of larger **frameworks**. Skeleton only styles a handful of standard HTML elements and includes a **grid**, but that's often more than enough to get started. In fact, this site is built on Skeleton and has ~200 lines of custom ...

Pure CSS

<https://purecss.io/> ▾

A set of small, responsive **CSS** modules that you can use in every web project.

[Grids](#) · [Layouts](#) · [Get Started](#) · [Forms](#)

What are Frameworks? 22 Best Responsive CSS Frameworks for Web ...

<https://www.awwwards.com/what-are-frameworks-22-best-responsive-css-frameworks...> ▾

20 Feb 2013 - Front-end **Frameworks** (or **CSS Frameworks**). Frontend **frameworks** usually consist of a package made up of a structure of files and folders of standardized code (HTML, **CSS**, JS documents etc.) The usual components are: **CSS** source code to create a **grid**: this allows the developer to position the different ...

Best CSS Frameworks of 2017 | Three29

<https://three29.com/best-css-frameworks-2017/> ▾

31 Mar 2017 - Allow us to lend a hand in sorting through the what's what in **CSS frameworks** in 2017. For those who are less tech savvy, a quick rundown. Building a website is a complicated process that involves a lot of moving parts. One of those components is the visual style and layout of a site. That's where our friend ...

How Big Is That Box? Understar X +

https://www.smashingmagazine.com/2018/01/understanding-sizing-css-layout/ Search



A key feature of Flexbox and Grid Layout is that they can deal with distributing available space between, around and inside grid and flex items. Quite often this *just works*, and we get the result we were hoping for without trying very hard. This is because the specifications attempt to default to the most likely use cases. Sometimes, however, you might wonder why something ends up the size that it is. Or, you might want to do something different to the default behavior. To do so, you need to know something of how the underlying algorithms figure out how to distribute space.

In this article, I'm going to share with you some interesting things about sizing boxes in CSS. I've picked out a few things from the specifications that I believe are vital in terms of understanding exactly how big that box is. Take some time to read through, and I think you'll find sizing in Grid a lot less mysterious!

TAKING A CLOSER LOOK AT BFC

JANUARY 16, 2018 • [8 COMMENTS](#)

How Big Is That Box? Understanding Sizing In CSS Layout

[CSS](#) 236 # [Layouts](#) 36 # [Browsers](#) 30

TABLE OF CONTENTS

- 01 [Length Units](#)
- 02 [Percentages](#)
- 03 [CSS Intrinsic And Extrinsic Sizing](#)
- 04 [Content-Based Sizing In CSS Grid Layout](#)
- 05 [Auto-Sized Tracks](#)
- 06 [fr Units](#)

How big is a grid?

CSS Grid Layout Module Level 1

W3C Candidate Recommendation, 14 December 2017



This version:

<https://www.w3.org/TR/2017/CR-css-grid-1-20171214/>

Latest published version:

<https://www.w3.org/TR/css-grid-1/>

Editor's Draft:

<https://drafts.csswg.org/css-grid/>

Previous Versions:

<https://www.w3.org/TR/2017/CR-css-grid-1-20170509/>

<https://www.w3.org/TR/2016/WD-css-grid-1-20160519/>

<https://www.w3.org/TR/2015/WD-css-grid-1-20150917/>

<https://www.w3.org/TR/2015/WD-css-grid-1-20150806/>

<https://www.w3.org/TR/2015/WD-css-grid-1-20150317/>

<https://www.w3.org/TR/2014/WD-css-grid-1-20140513/>

<https://www.w3.org/TR/2014/WD-css-grid-1-20140123/>

<https://www.w3.org/TR/2013/WD-css3-grid-layout-20130402/>

<https://www.w3.org/TR/2012/WD-css3-grid-layout-20121106/>

Test Suite:

http://test.csswg.org/suites/css-grid-1_dev/nightly-unstable/

Issue Tracking:

[Disposition of Comments](#)

[Inline In Spec](#)

[GitHub Issues](#)

Editors:

“Each track has specified minimum and maximum sizing functions (which may be the same).”

Cada traza tiene especificado un tamaño mínimo y máximo funciones (que pueden ser las mismas).

- 11.1 Grid Sizing Algorithm

Tamaño ajustado - no flexible porque esta establecido

- ❖ **Fixed sizing:**

lengths such as px or em, or a resolvable percentage

- ❖ An **intrinsic sizing function** Tamaño determinado de forma interna por el contenido

auto, min-content, max-content, fit-content

Tamaño flexible

- ❖ A **flexible sizing function**

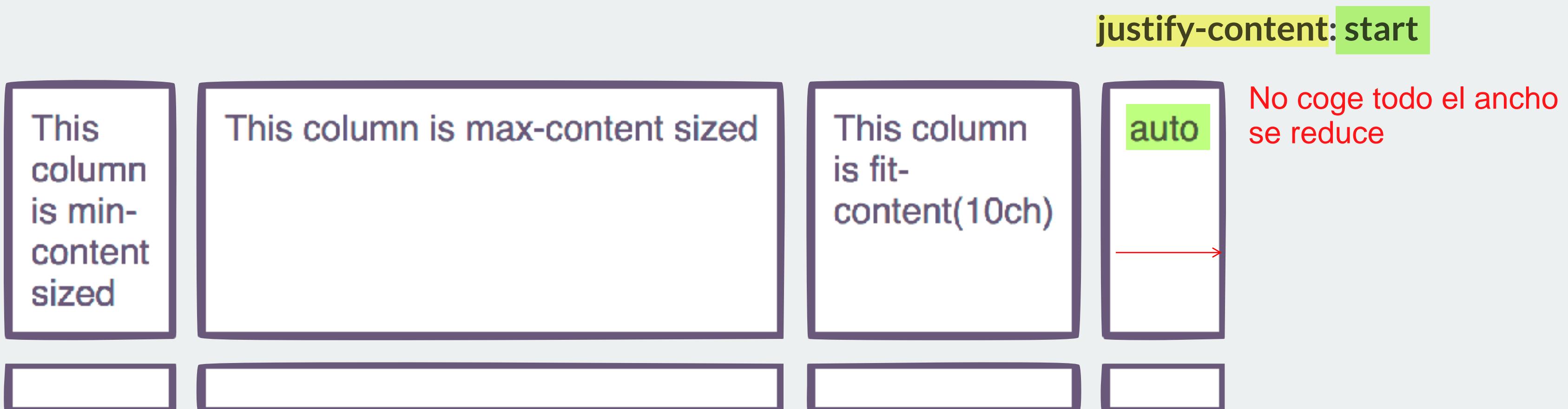
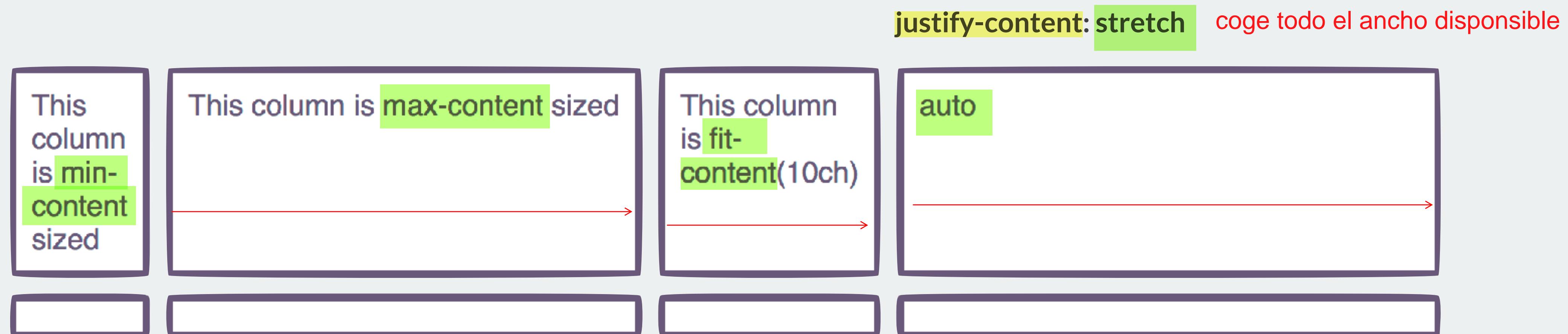
The fr unit

Intrinsic Sizing

auto

Default size of grid tracks. Tracks sized `auto` will stretch to take up space in the grid container.

Tamaño predeterminado de los grid-track
El tamaño de las track se extenderá para ocupar todo el espacio del grid-container



Intrinsic Sizing: *auto*

El auto-dimensionado banda se estirara dentro la dirección en linea
The auto-sized track will stretch in the inline direction.

Use justify-content: start ^{anular} to override the stretching behaviour.

Usar justify-content:start anular el comportamiento de estiramiento.

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    min-content  
    max-content  
    fit-content(10ch)  No estira  
    auto; —> Esta columna perderá el tamaño por usar justify-content:start  
    justify-content: start;  
}
```

Intrinsic Sizing

min-content

The smallest size the item can be, taking advantage of all soft-wrapping opportunities.

El tamaño más pequeño que el elemento puede ser, tomando ventaja de todo texto que se puede dividir en varias líneas

Intrinsic Sizing: `min-content`

Grid tracks sized with `min-content` will become as small as they can without causing overflows.

se volverán tan pequeñas como pueden sin causar desbordamientos

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    min-content  
    min-content  
    min-content;  
}
```

The word 'columns'
is defining the size
of track 1.

A 100px image
defines the size
of track 3



This item has a width
which defines the size
of track 2

Intrinsic Sizing

max-content

El mas largo tamaño la banda puede ser

The largest size the track can be, no soft-wrapping will occur. Overflows may happen.

no se producirá ningún envoltorio blando. Pueden producirse desbordamientos.

Intrinsic Sizing: `max-content`

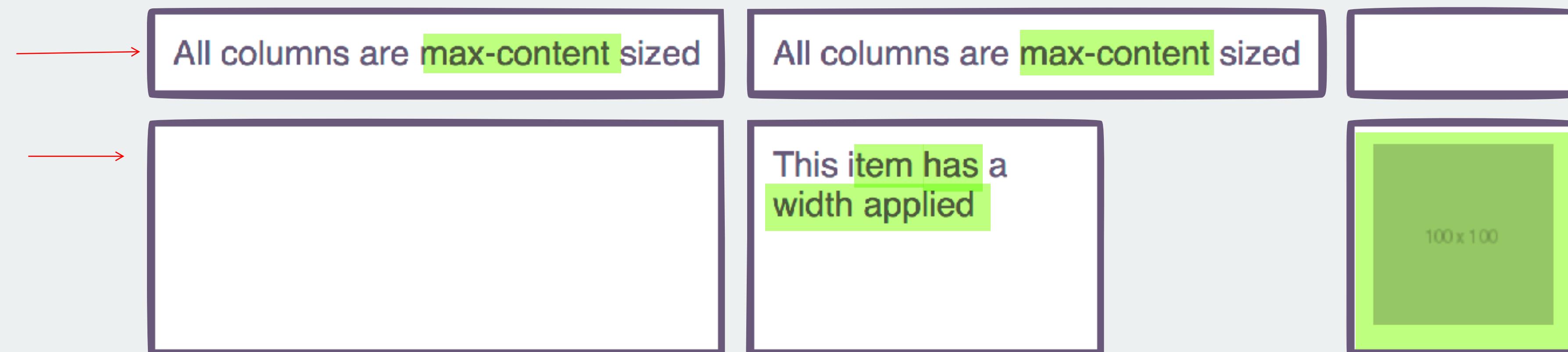
Grid tracks sized with `max-content` will become as large as needed, and may overflow the grid container.

talla se convertirá de largo como necesita
quizás desborde

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    max-content  
    max-content  
    max-content;  
}
```

Las pistas 1 y 2 deben ser lo suficientemente ancho para que quepa el texto desenvuelto.
Tracks 1 and 2 need to be wide enough to fit the unwrapped text.

A 100px image defines the size of track 3



Intrinsic Sizing

fit-content

Actúa como max-content hasta que alcanza el valor aprobado.

Act like max-content until it reaches the passed in value.

Intrinsic Sizing: *fit-content*

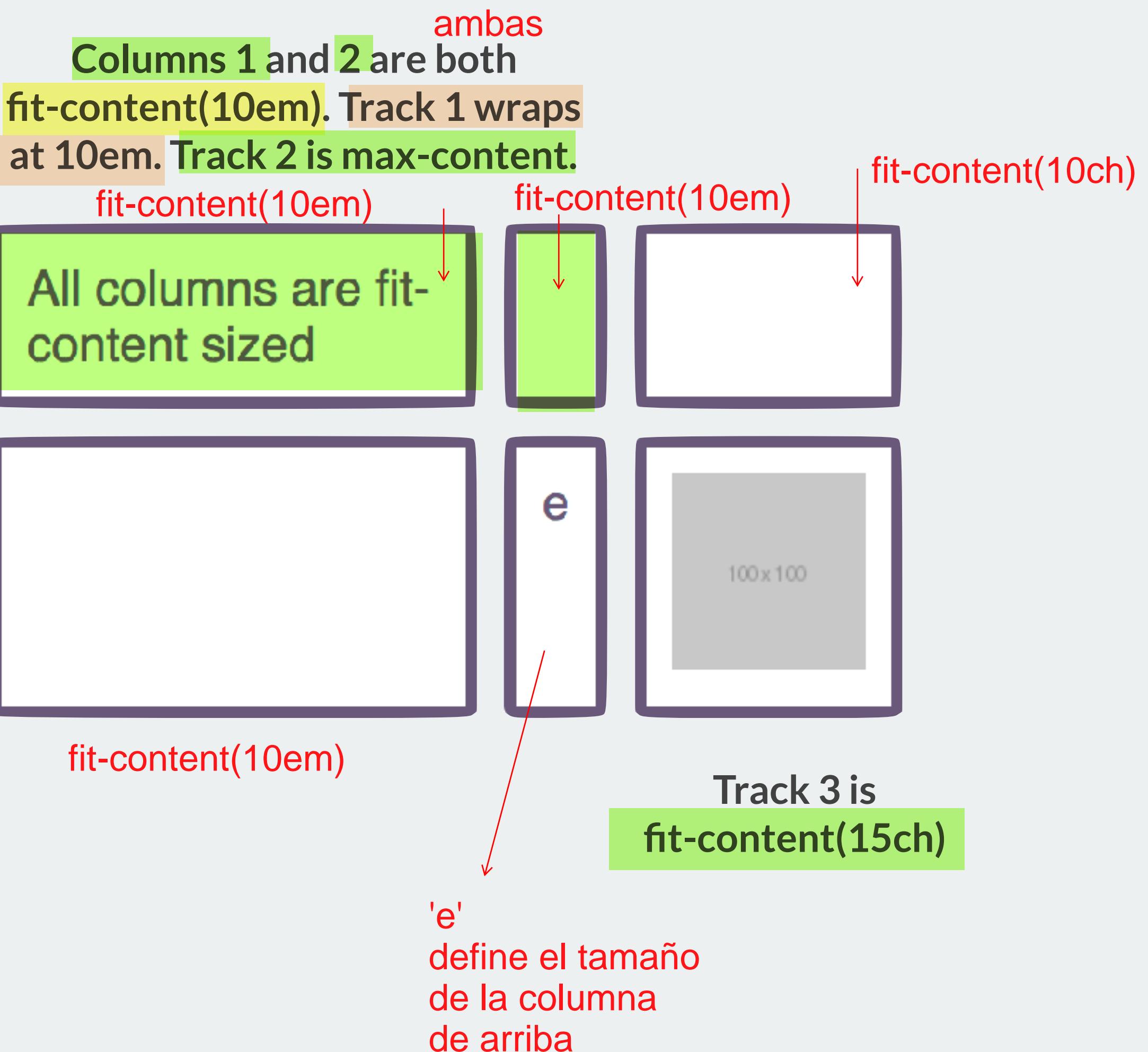
Banda dimensionada con
Grid tracks sized with *fit-content* will act like max-content
until they hit the limit given.
hasta que lleguen al límite dado.

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    fit-content(10em)  
    fit-content(10em)  
    fit-content(15ch);  
}
```

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    fit-content(10em)  
    fit-content(10em)  
    fit-content(10ch);  
  justify-content: start;  
}
```

```
.fixed-width {  
  width: 200px;  
}
```

```
.grid > * {  
  border: 5px solid rgb(108,91,123);  
  border-radius: 5px;  
  padding: 10px;  
}
```



Flexible lengths

Sizing with **fr** units

The **fr** unit describes a **flexible length**
longitud ajustable al ancho del contenedor

Flexible lengths

The **fr unit** is a **<flex>** unit and represents a portion of the **available space** in the Grid Container.

unidad y representa una porcion del espacio **en el contenedor**

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns: 2fr 1fr 1fr;  
}
```

2fr



1fr



1fr



Minimum & Maximum sizing functions

minmax()

Minimum Sizing Function

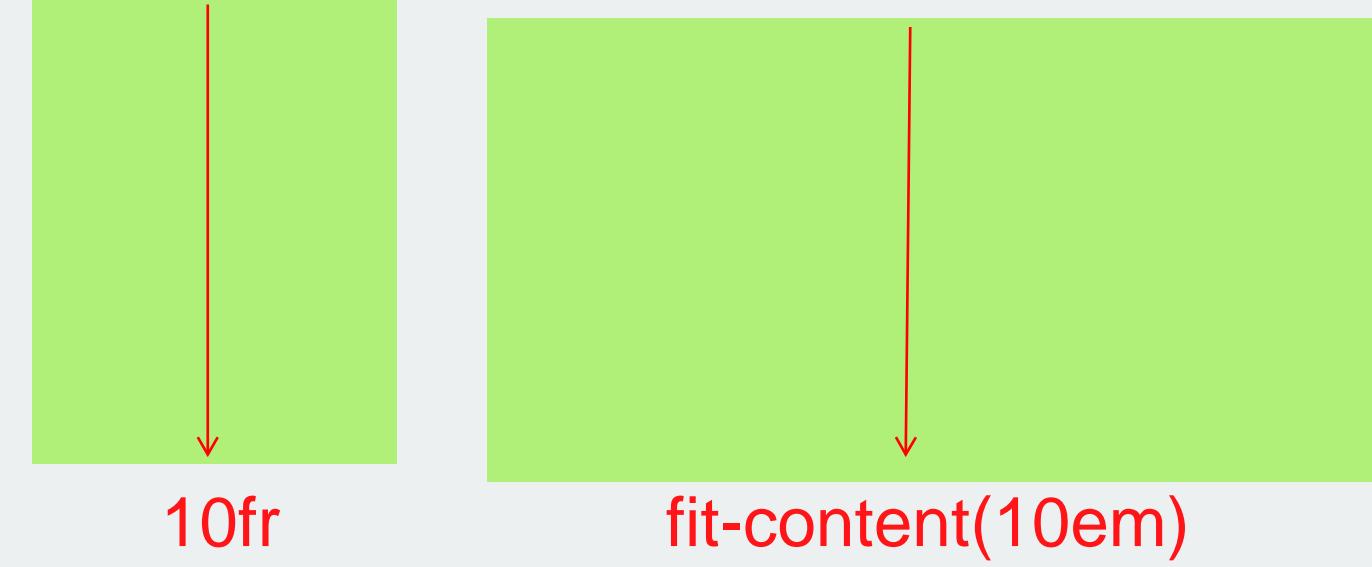
The minimum size for these two tracks is 100px, and 10em.

```
.grid {  
  display: grid;  
  grid-template-columns:  
    minmax(100px, auto)  
    minmax(10em, 20em);  
}
```

minimo tamaño tamaño maximo

Minimum Sizing Function

The minimum size of these two tracks is auto.



track tb se originan
verticalmente

```
.grid {  
  display: grid;  
  grid-template-columns:  
    10fr  
    fit-content(10em);  
}
```

Minimum Sizing Function

The minimum size of the first track is 100px, the second track is 50px (10% of 500).

```
.grid {  
  width: 500px;  
  display: grid;  
  grid-template-columns:  
    100px  
    10%;  
}
```

500px * 10 = 50px

Maximum Sizing Function

The maximum size for these two tracks is 400px, and 20em.

```
.grid {  
  display: grid;  
  grid-template-columns:  
    minmax(100px, 400px)  
    minmax(10em, 20em);  
}
```

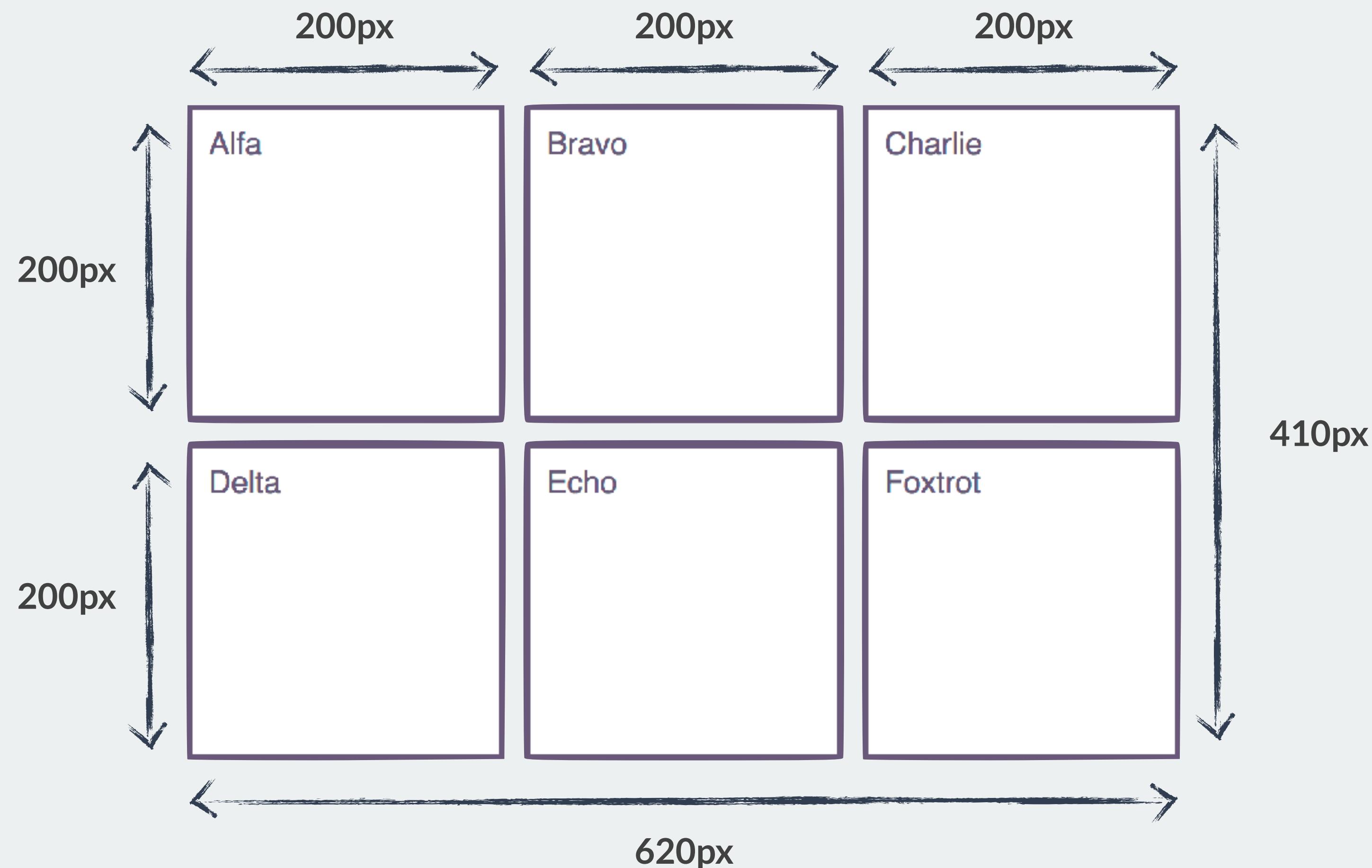
 maximo tamaño

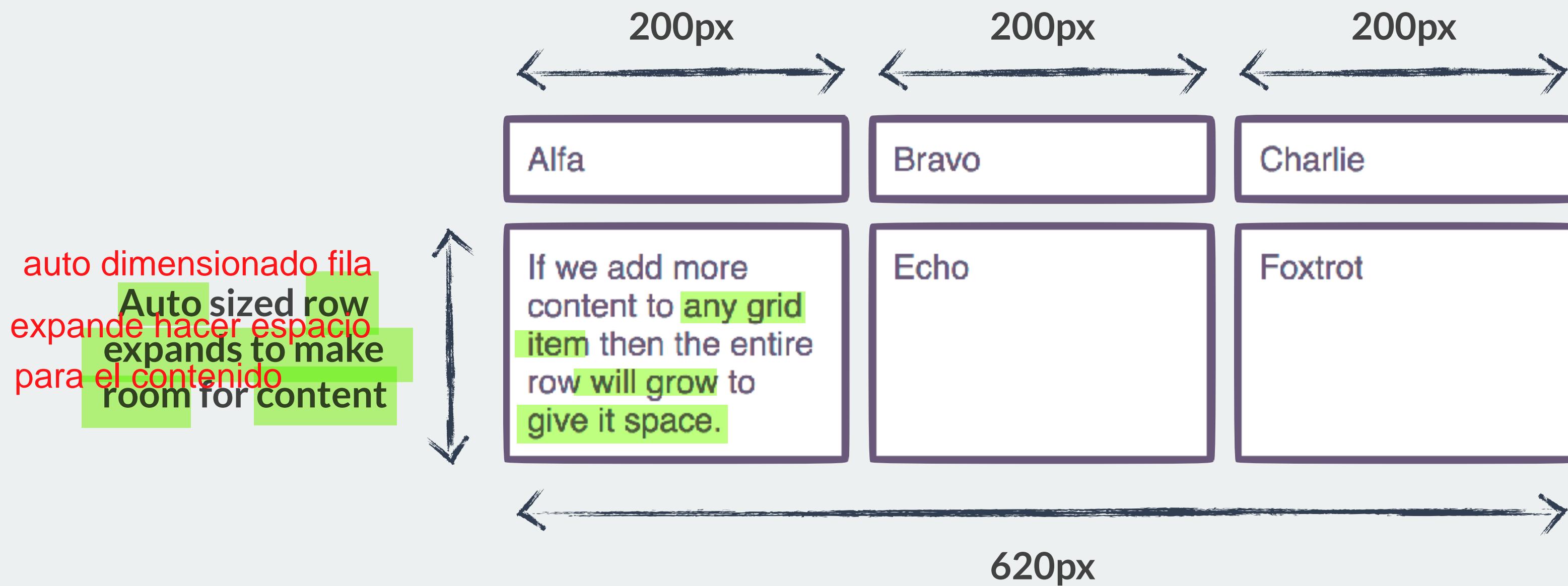
Maximum Sizing Function

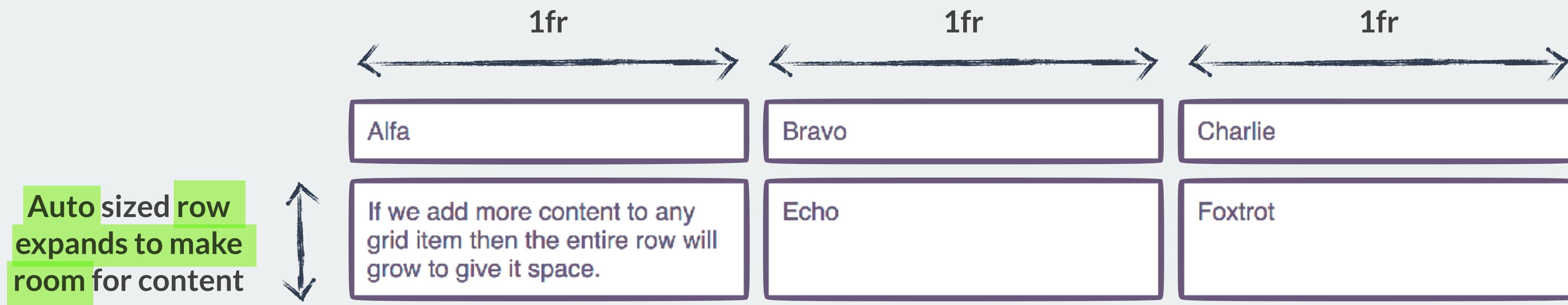
The maximum size for the first track is max-content.

For the second track the maximum is 10em.

```
.grid {  
  display: grid;  
  grid-template-columns:  
    auto  
    fit-content(10em);  
}
```





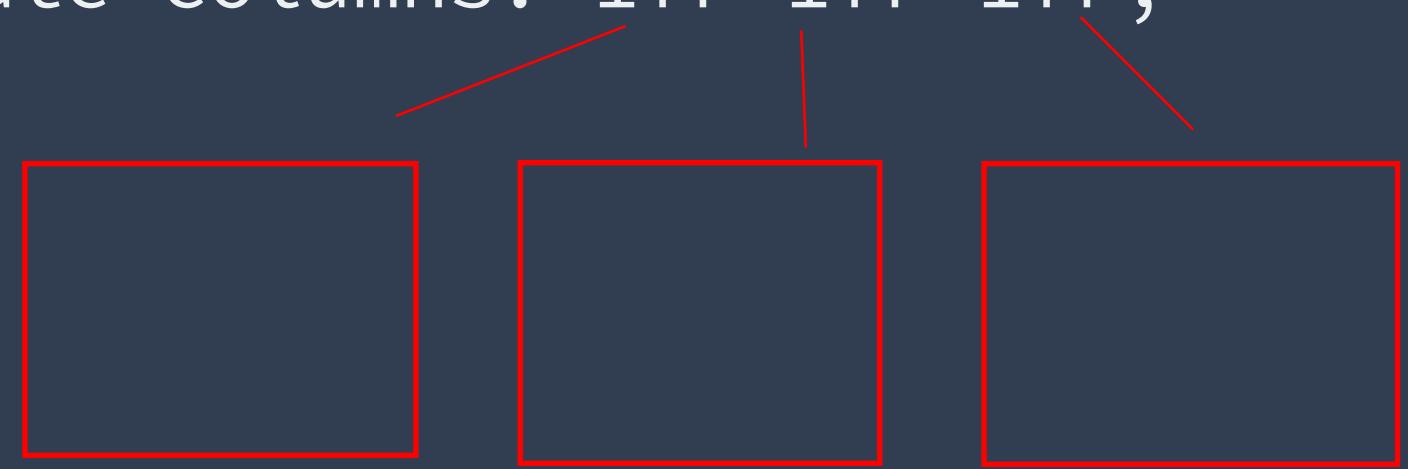


de todas formas
What is a fr unit anyway?

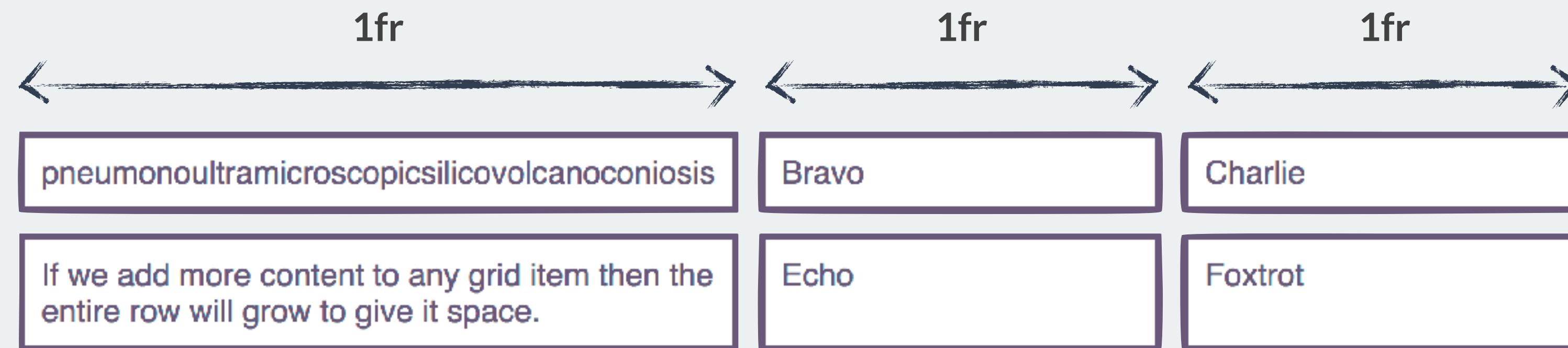
A track sized with **1fr**, is actually sized **minmax(auto, 1fr)**.

The **minimum size** is **auto**.

```
.grid {  
  display: grid;  
  grid-template-columns: 1fr 1fr 1fr;  
}
```



ancho total



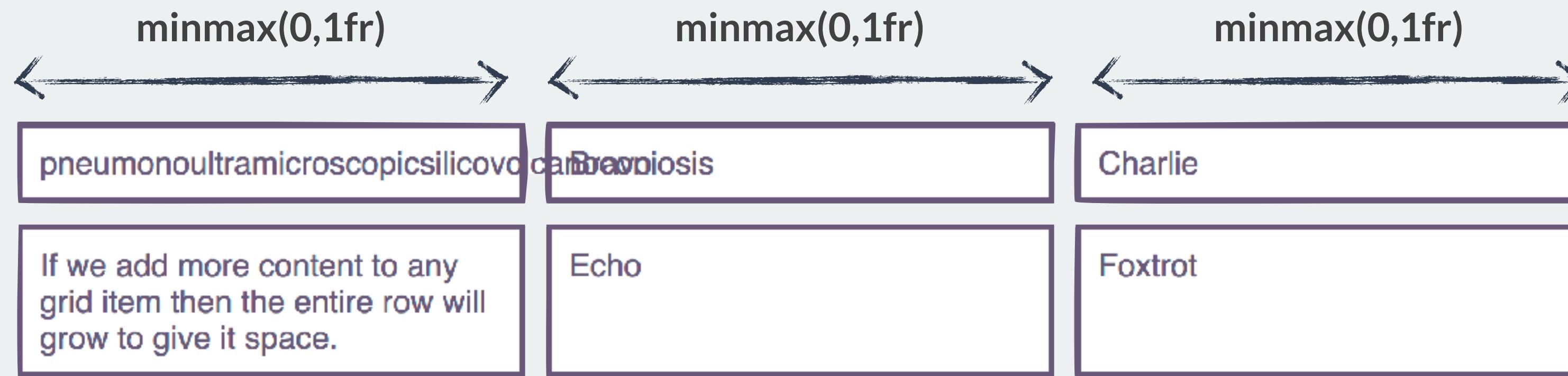
de todos modos , formas

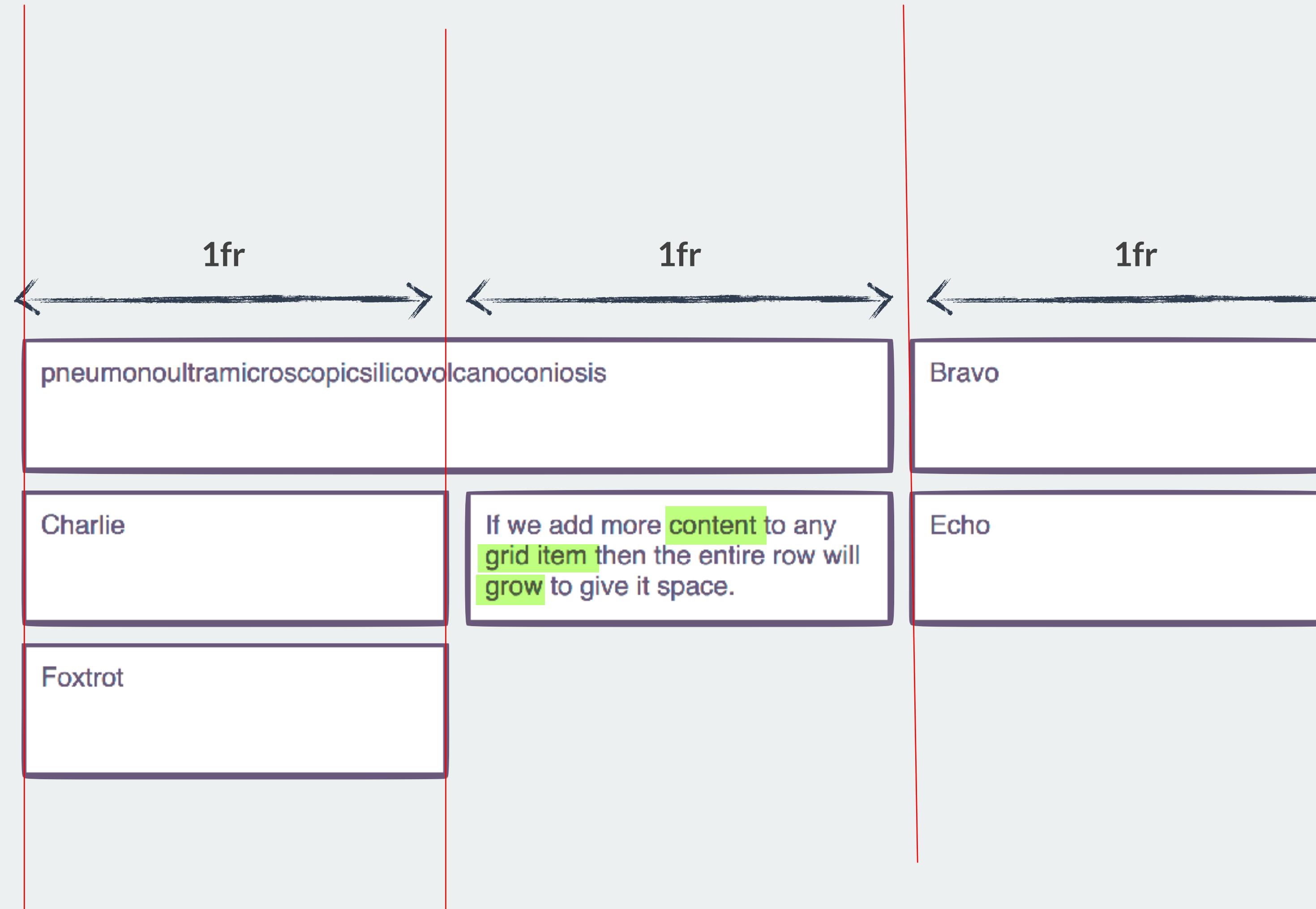
What is a fr unit anyway?

Make the minimum size 0, to force three equal width columns - even if that means overflows.

incluso si esto significa desbordamiento

```
.grid {  
  display: grid;  
  grid-template-columns:  
    minmax(0,1fr)  
    minmax(0,1fr)  
    minmax(0,1fr);  
}
```





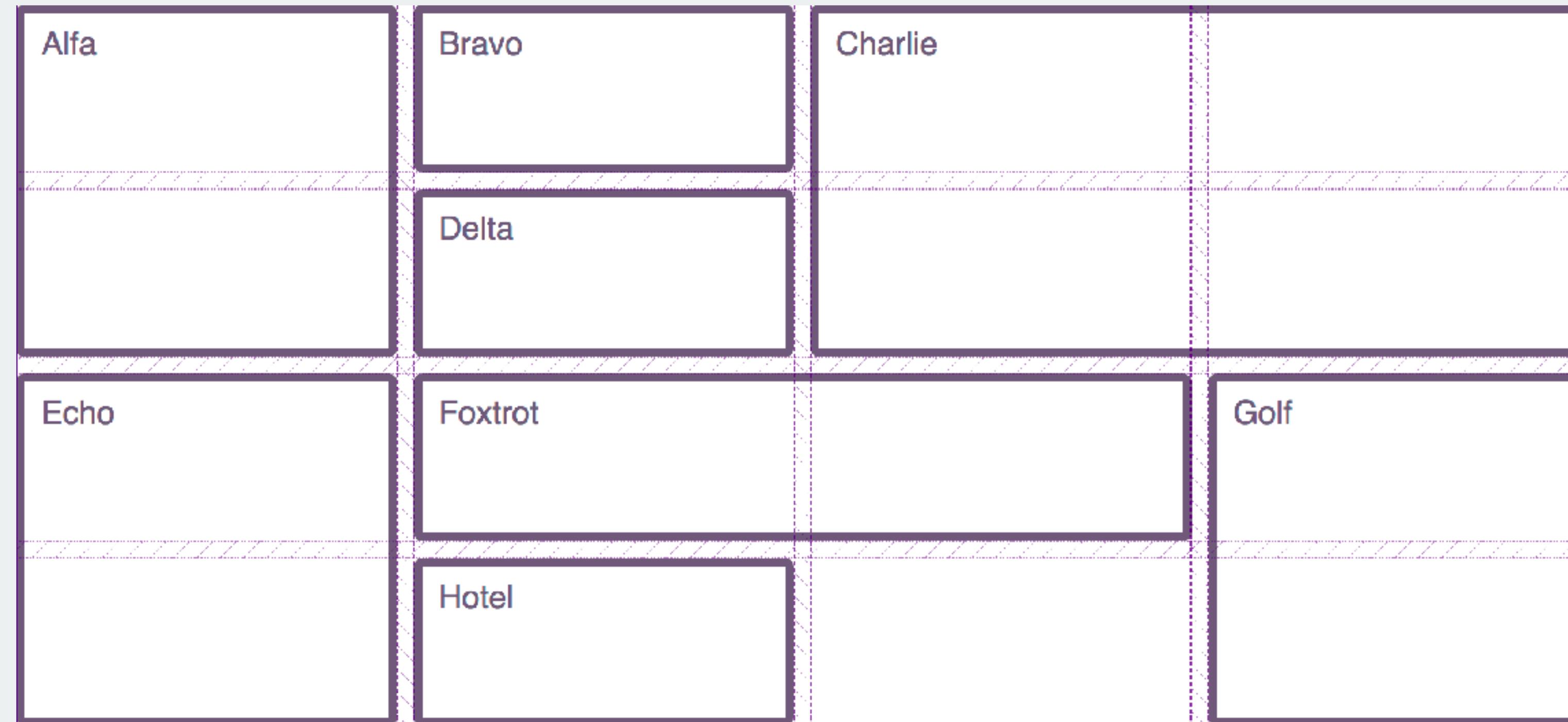
Tantas columnas flexibles como quepan

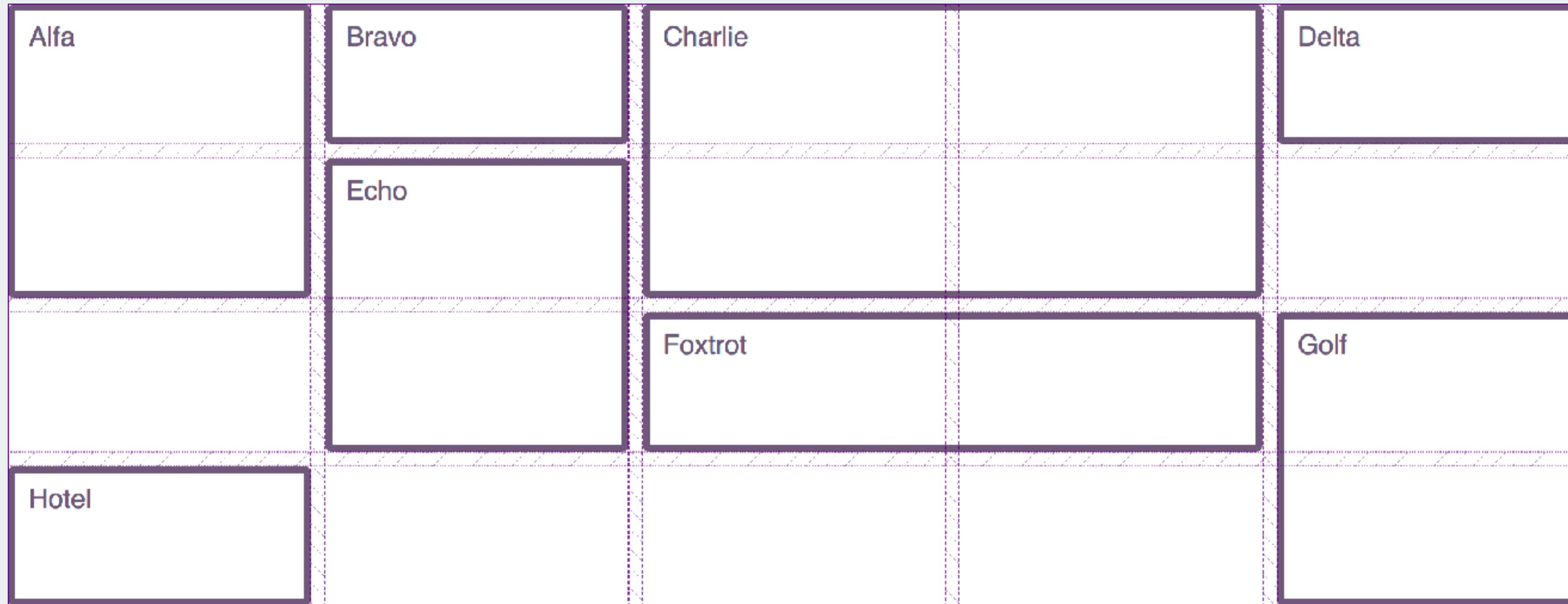
As many flexible columns as will fit

junto con

Use auto-fill or auto-fit along with repeat and minmax, to
create as many flexible columns with a minimum size as will
fit into the container.

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    repeat(auto-fill, minmax(200px, 1fr));  
  grid-auto-rows: 100px;  
}
```

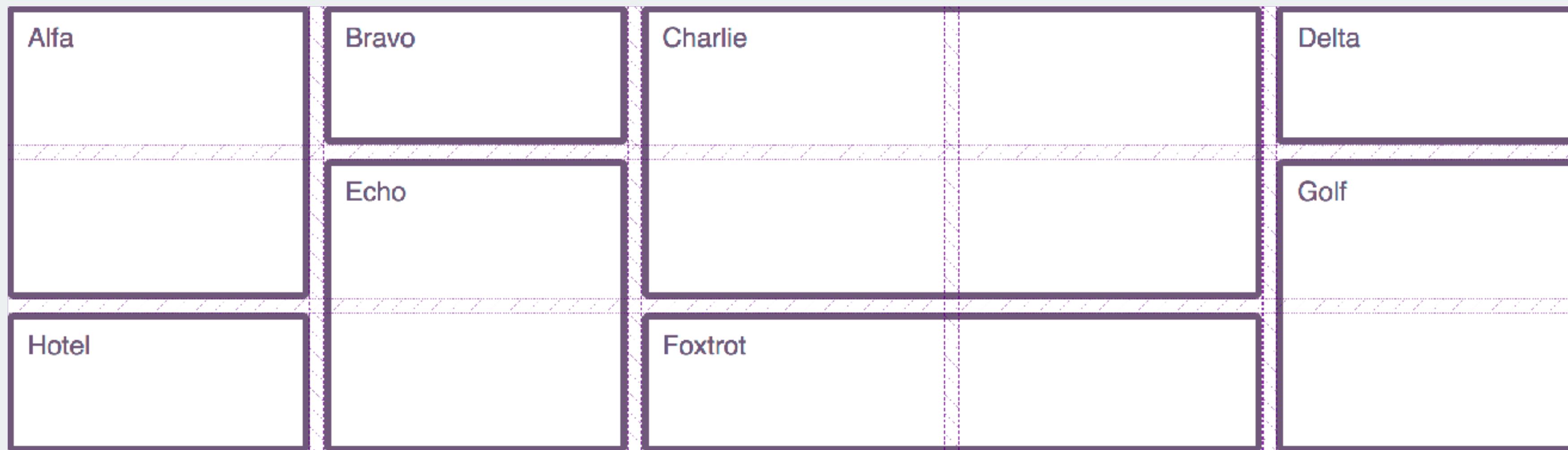




Dense Packing Mode

Using the value `dense` for `grid-auto-flow` will cause items to be displayed out of document order.
causará elementos ser mostrados fuera orden del documento

```
.grid {  
  display: grid;  
  grid-gap: 10px;  
  grid-template-columns:  
    repeat(auto-fill, minmax(200px, 1fr));  
  grid-auto-rows: 100px;  
  grid-auto-flow: dense;  
}
```



Upside down and back to front

Al revés y de vuelta al frente

The `grid-area` property

Order of the lines:

- `grid-row-start`
- `grid-column-start`
- `grid-row-end`
- `grid-column-end`

```
.item {  
  grid-area: 1 / 2 / 3 / 4;  
}
```

row start column start row start column end
 ↓ ↓ ↓ ↓

logicas Propiedades y escribiendo modos

Logical Properties and Writing Modes

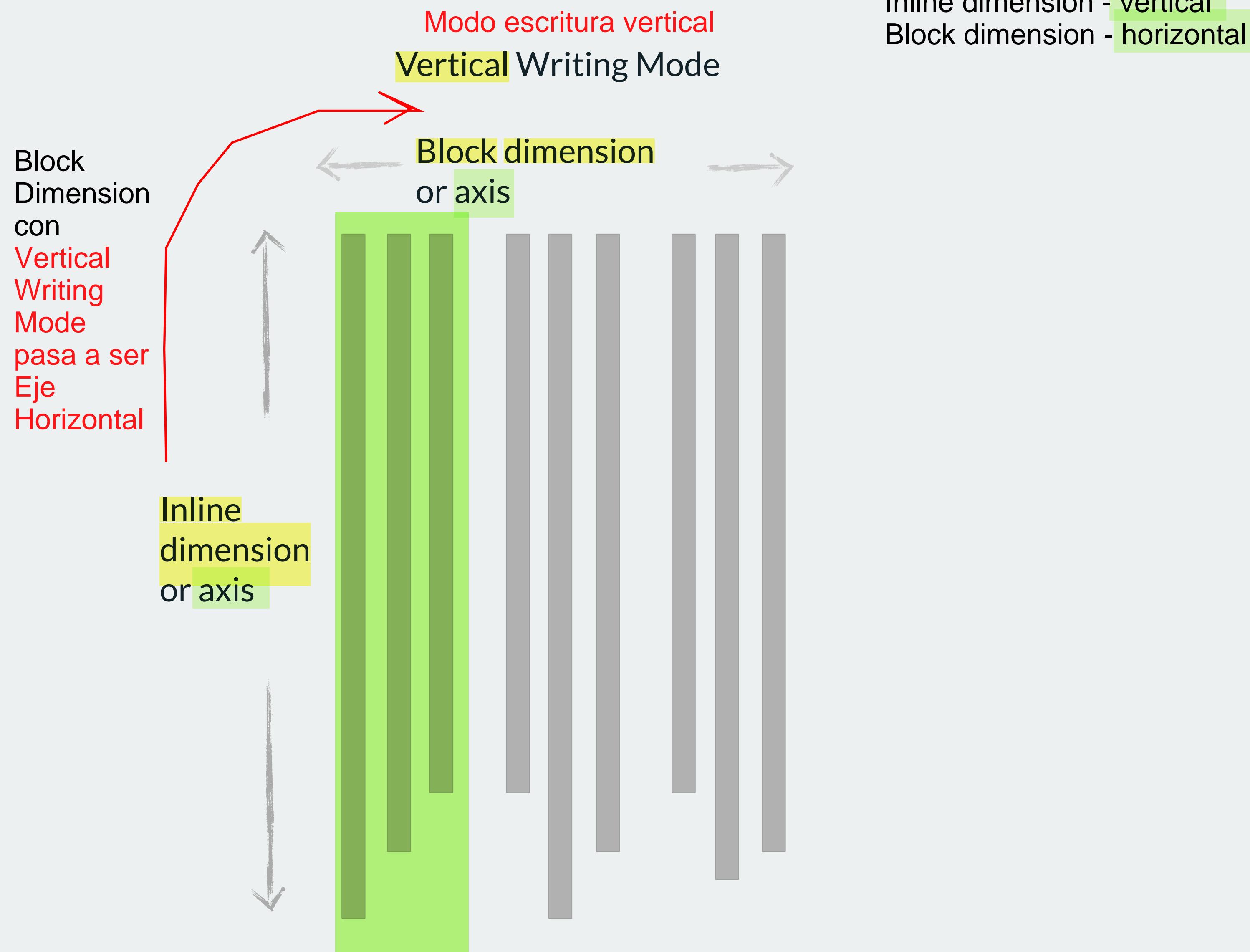
FLEXBOX

Modo Escrita 'Horizontal'
Horizontal Writing Mode

Inline dimension horizontal
Block dimension vertical



FLEXBOX



Grid Layout in Horizontal Writing Mode

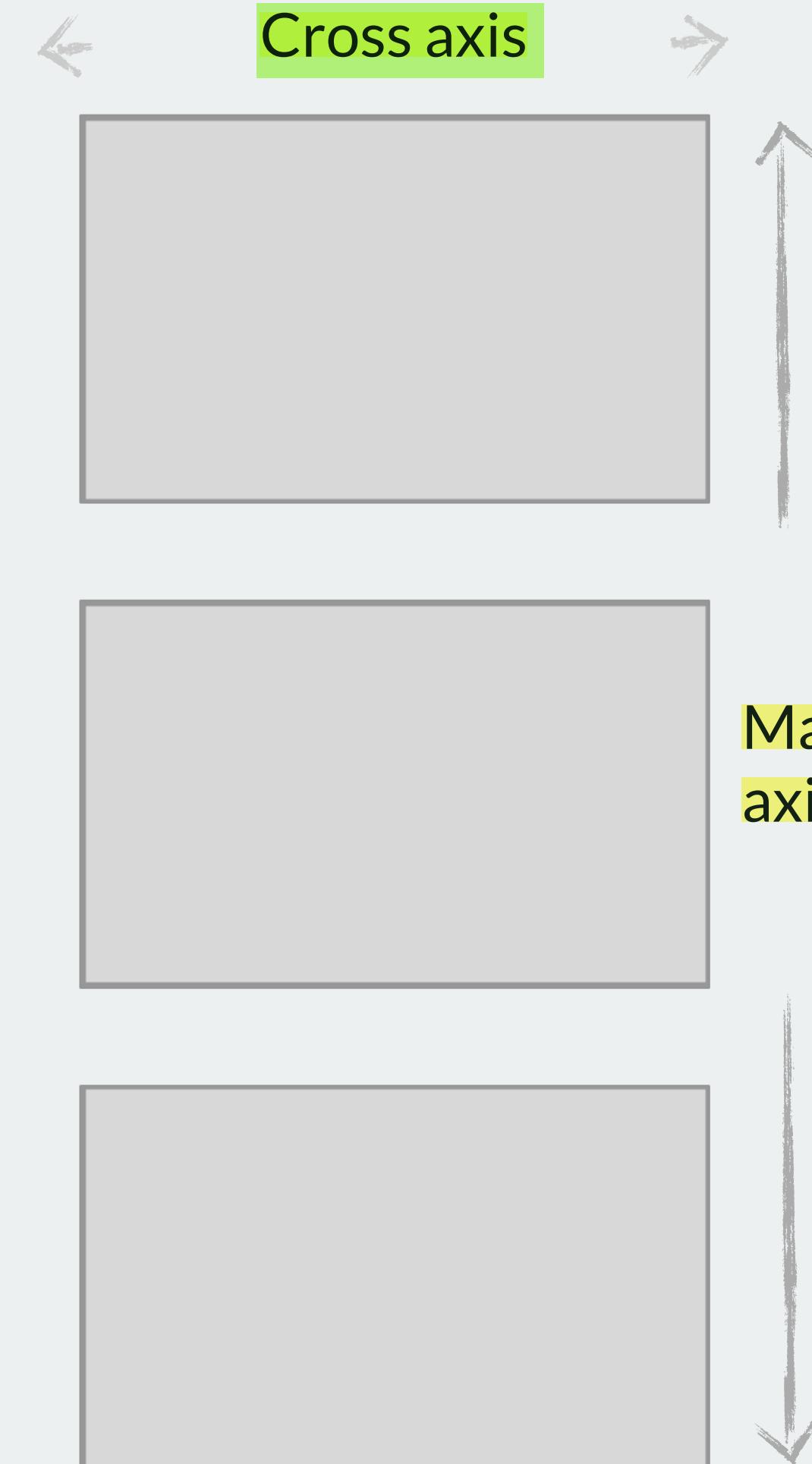


Flex Layout in Horizontal Writing Mode



Cambia el sentido de la distribución de los elementos

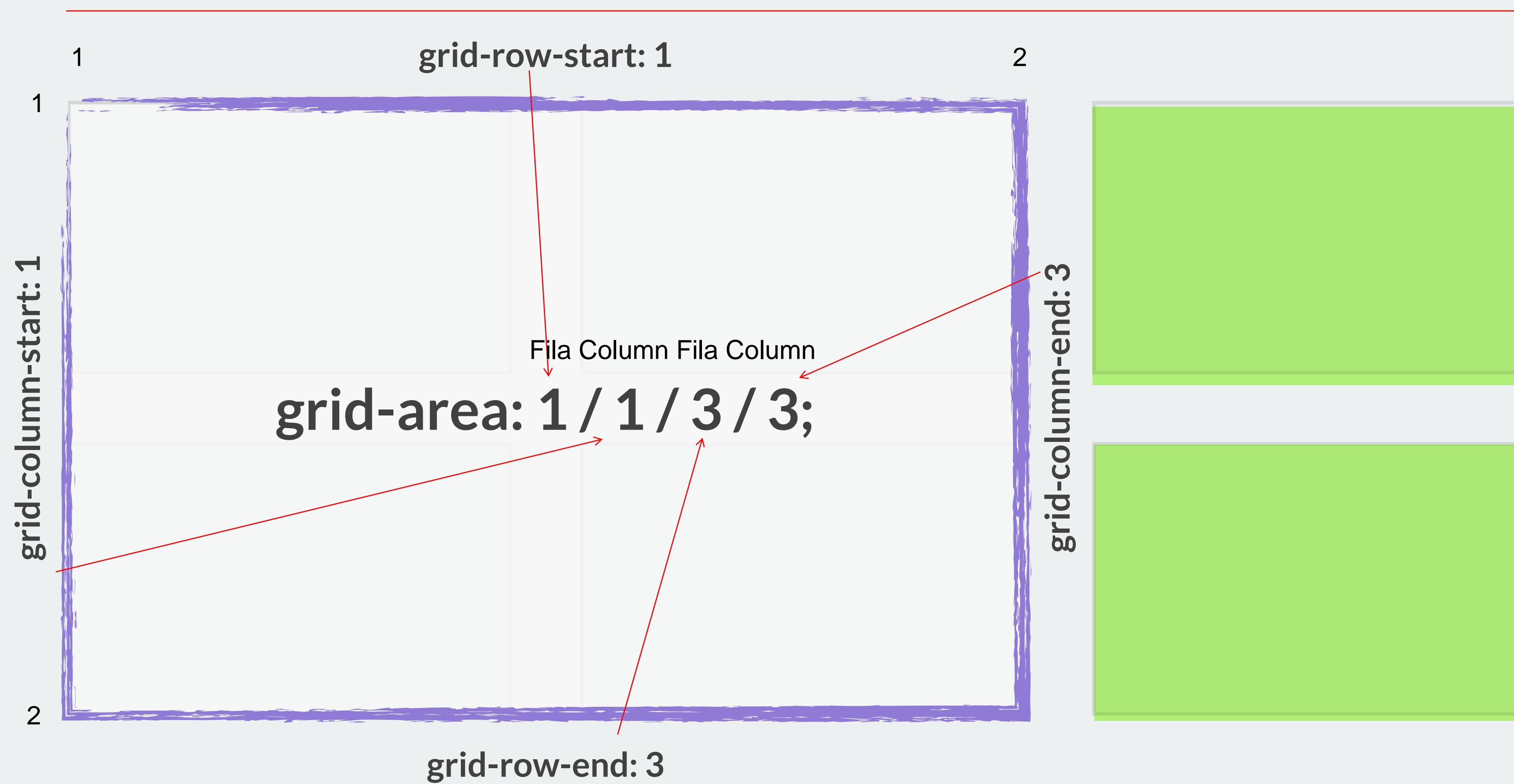
flex-direction: column



Left - Right
Horizontal LR

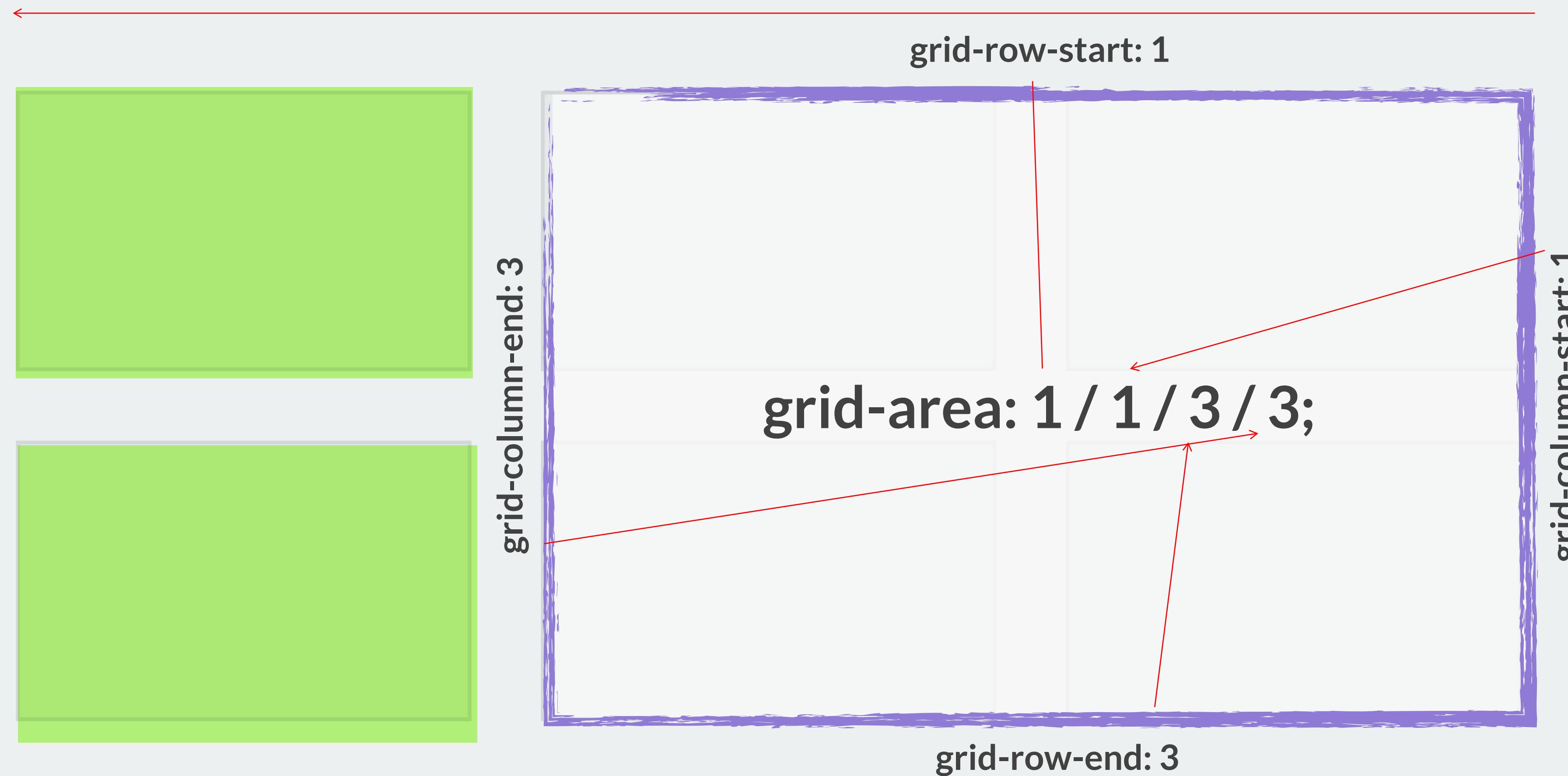


Horizontal LR

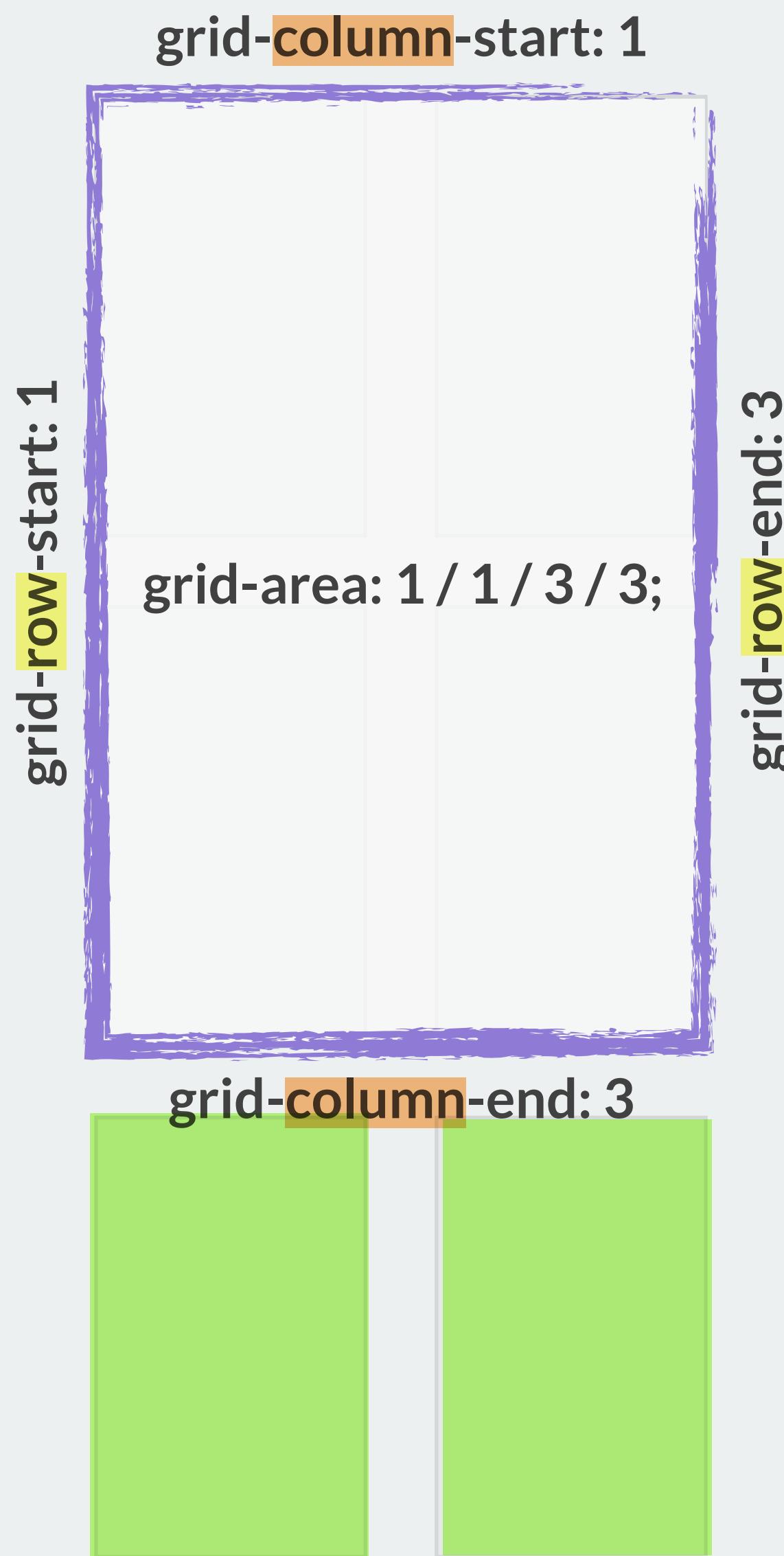


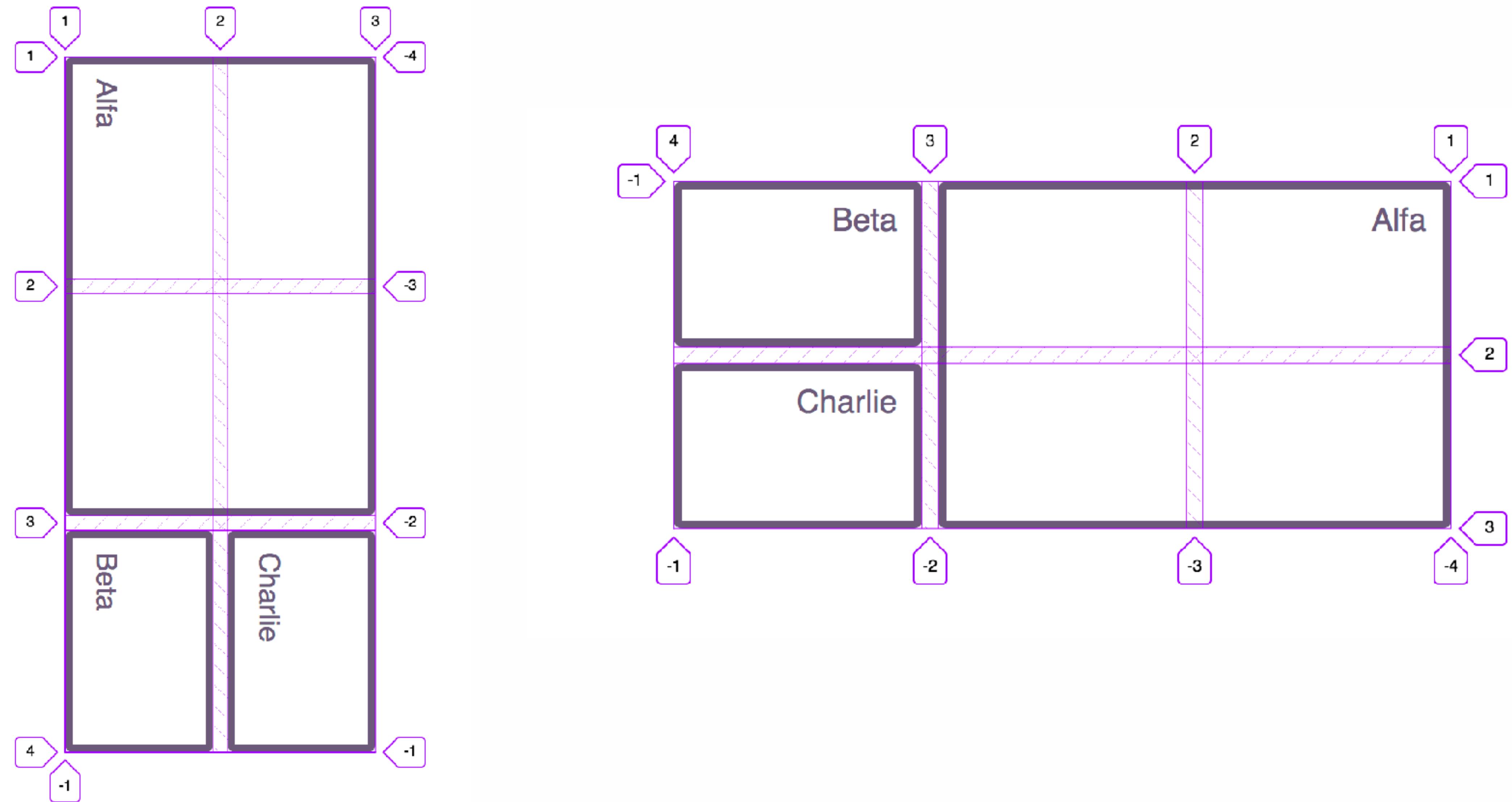
Right - Left

Horizontal RL



Vertical LR





Alignment and the Grid

Alineacion de las bandas

- ❖ **Align the tracks:**
esto requiere espacio extra dentro del grid container
this requires extra space in the grid container

Alinear los elementos

- ❖ **Align the items**
mover el contenido dentro del área en la que se ha colocado
move content around inside the area it has been placed into
mover el contenido dentro del área en la que se ha colocado

Box Alignment Specification

alineando bandas

Aligning tracks

las bandas creadas sobre este grid no rellenan todo el ancho y alto

The tracks created on this grid do not fill the width and height defined on the grid container.

```
.grid {  
  display: grid;  
  width: 800px;  
  height: 500px;  
  border: 5px solid rgb(108,91,123);  
  grid-gap: 10px;  
  grid-template-columns: 200px 200px 200px;  
  grid-template-rows: 150px 150px;  
  justify-content: space-between;  
  align-content: space-around;  
}
```

```
.grid > div {  
  background-color: rgba(108,91,123,.5);  
  border-radius: 5px;  
  padding: 10px;  
}
```

```
.grid {  
  display: grid;  
  width: 800px;  
  height: 500px;  
  grid-gap: 10px;  
  grid-template-columns: 200px 200px 200px;  
  grid-template-rows: 150px 150px;  
}
```





align-content and *justify-content*

Alineando bandas

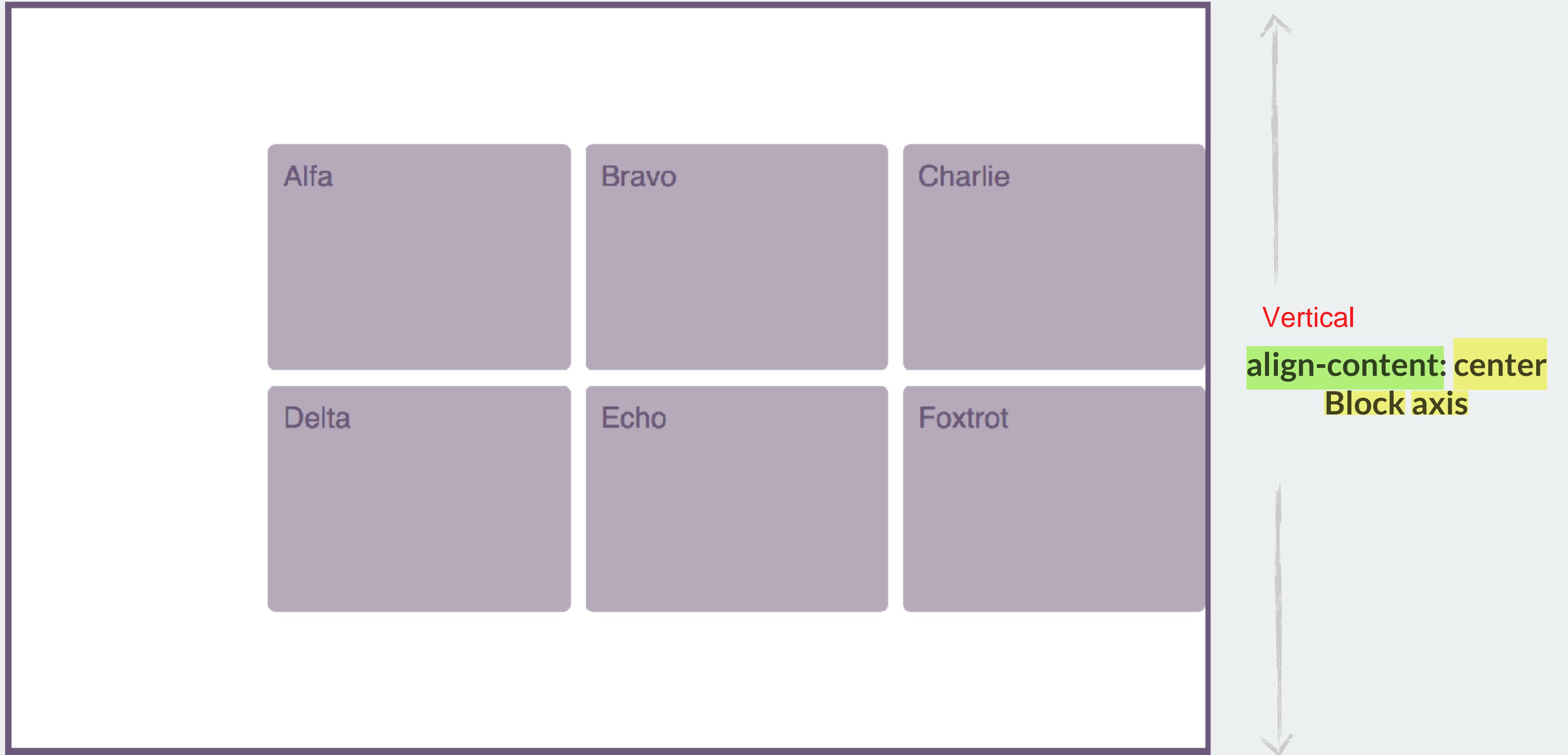
Aligning tracks

Que deberia de hacer con el espacio libre

What should I do with free space in the Grid Container?

- ❖ **align-content**
align tracks in the Block Dimension
- ❖ **justify-content**
align tracks in the Inline Dimension





Palabra clave

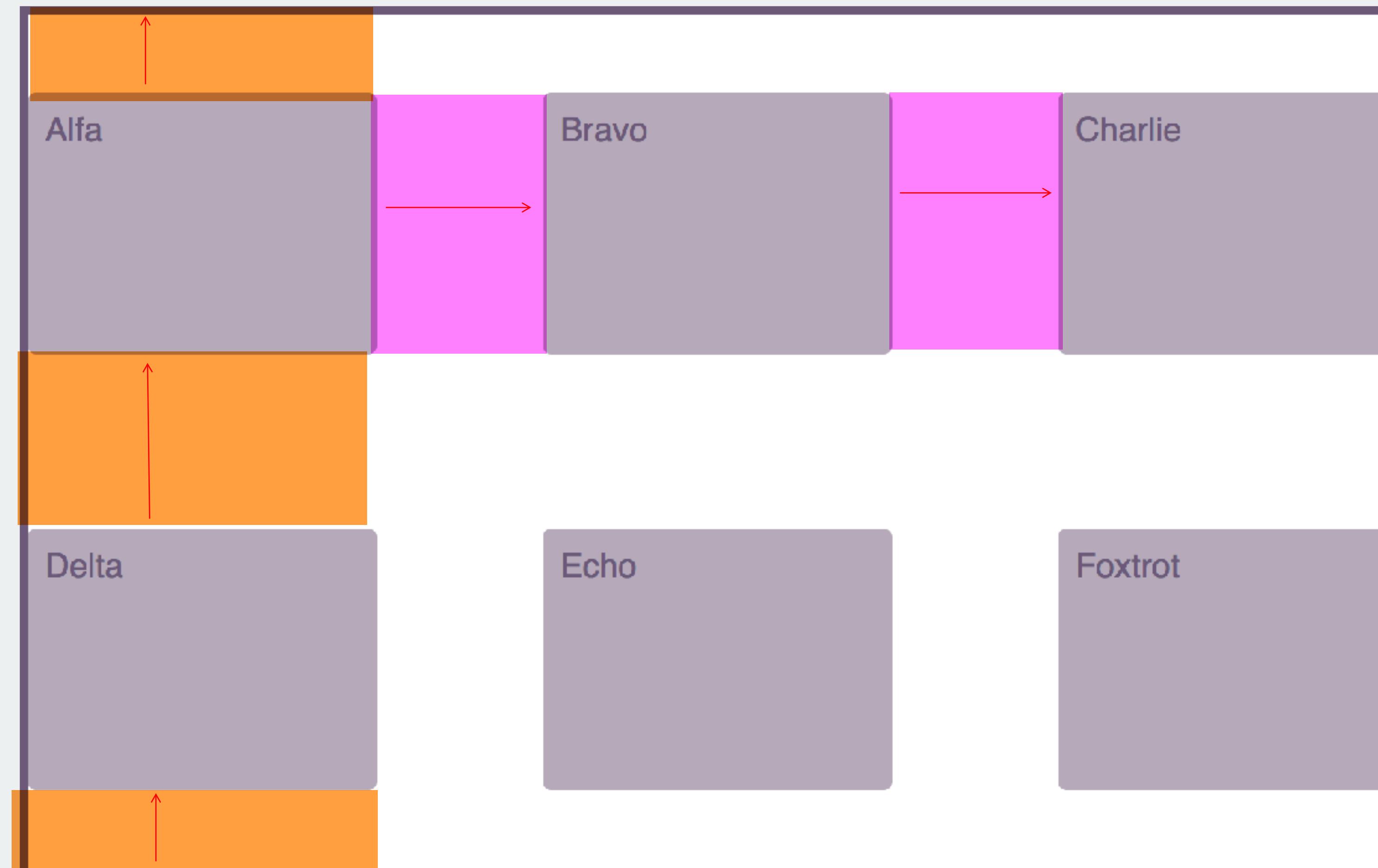
Keyword values

- ❖ space-between
- ❖ space-around
- ❖ space-evenly

```
.grid {  
    display: grid;  
    width: 800px;  
    height: 500px;  
    grid-gap: 10px;  
    grid-template-columns: 200px 200px 200px;  
    grid-template-rows: 150px 150px;  
    justify-content: space-between; horizontal  
    align-content: space-around; vertical  
}
```

Horizontal

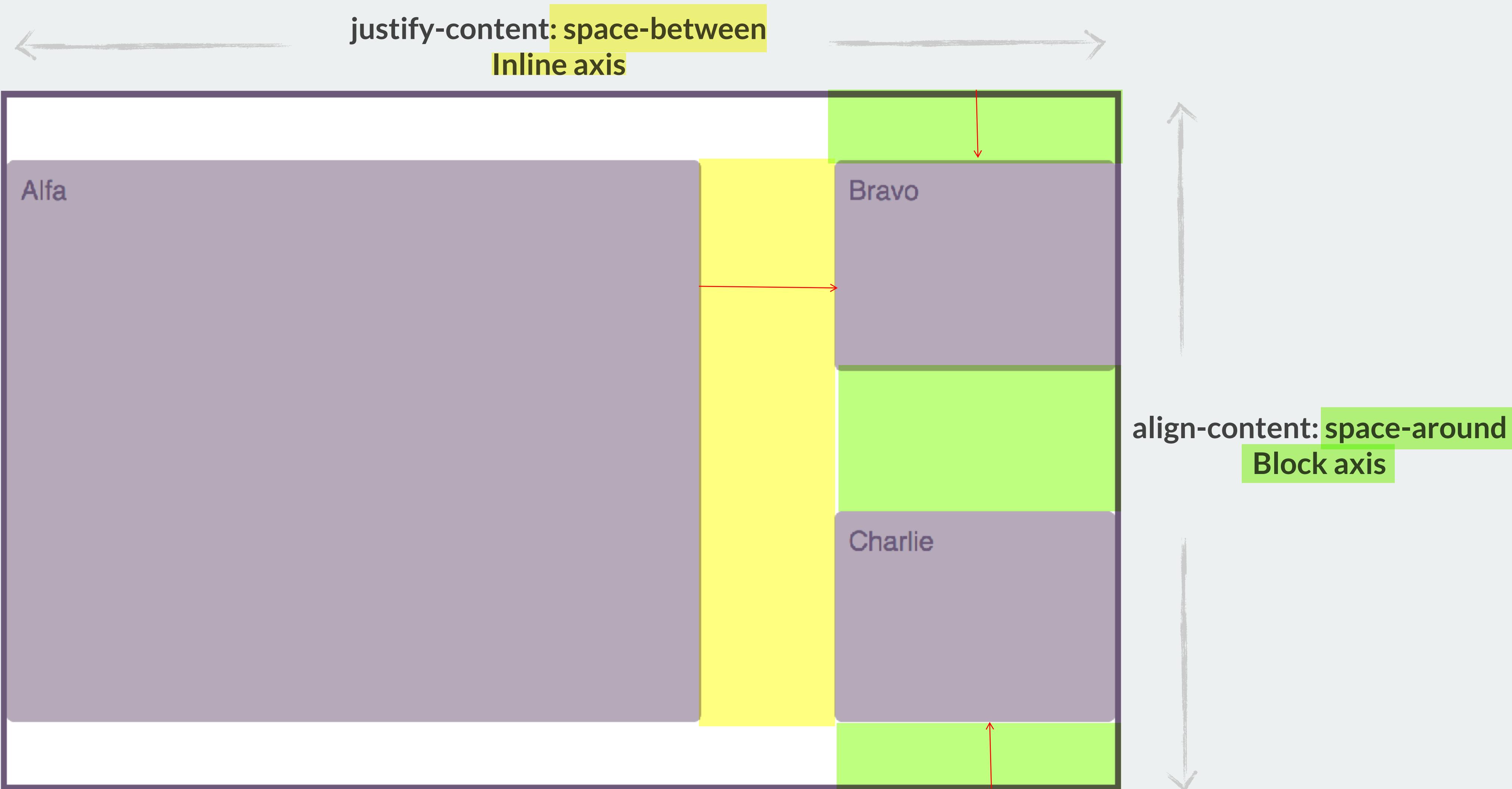
justify-content: space-between
Inline axis



1º - Divido el GRID-CONTAINER en varias fracciones o Elementos

2º - Distribuyo los elementos GRID-ITEMS sobre el espacio del GRID CONTAINER
mediante las propiedades justify-content , align-contents





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

```
html {  
  font: 1.2em Helvetica, sans-serif;  
}
```

```
body {  
  padding: 40px;  
  background-color: #fff;  
  color: rgb(108,91,123);  
}
```

```
.grid {  
  display: grid;  
  inline-size: 800px;  
  block-size: 500px;  
  border: 5px solid rgb(108,91,123);  
  grid-gap: 10px;  
  grid-template-columns: 200px 200px 200px;  
  grid-template-rows: 150px 150px;  
  justify-content: end;  
  align-content: center;  
  writing-mode: vertical-rl;  
}
```

```
.grid > div {  
  background-color: rgba(108,91,123,.5);  
  border-radius: 5px;  
  padding: 10px;  
}
```



writing-mode : vertical-rl
GIRA POR COMPLETO
EL CONTENEDOR
GRID-CONTAINER

Posicionamiento HORIZONTAL pasa VERTICAL
justify-content: end
Inline axis

```
.grid div:first-child {  
  grid-column: 1 / 1;  
  grid-row: 2 / 3;  
}
```

Row - Linea Superior de posicionamiento
Column - Linea Izquierda de posicionamiento

Writing Modes

Once again the grid works the same way whichever way up it is.

cualquiera camino arriba

```
.grid {  
  display: grid;  
  inline-size: 800px; hacia arriba  
  block-size: 500px; de izq a der  
  grid-gap: 10px;  
  grid-template-columns: 200px 200px 200px;  
  grid-template-rows: 150px 150px;  
  justify-content: end;  
  align-content: center;  
  writing-mode: vertical-lr; girar el grid-container  
}
```



In the past, CSS has tied itself to physical dimensions and directions, physically mapping the placement of elements to the left, right and

top and bottom. We `float` an element `left` or `right`, we use the positioning offset properties `top`, `left`, `bottom` and `right`. We set margins, padding, and borders as `margin-top` and `padding-left`. These physical properties and

values make sense if you are working in a horizontal, top to bottom, left to right writing mode and direction.

They make less sense if you use a vertical writing mode, whether for your entire layout or for some elements. In this article, I'm going to explain how CSS is changing to support writing modes, and in doing so, I'll clear up some of the things that might confuse you about Flexbox and Grid.

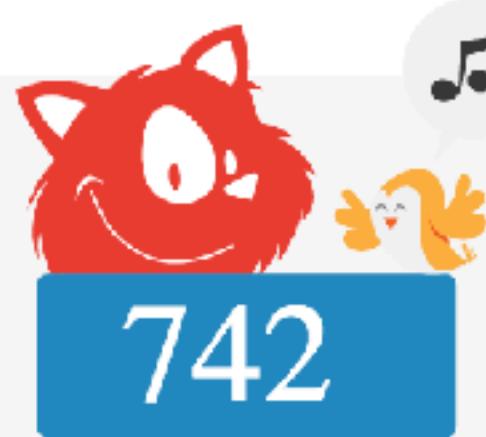
When I first began working with CSS Grid and explaining the new specification to people, I noted that the `grid-area` property could be used as a one-line shorthand for setting all four lines. Therefore, the three examples below would result in the same item placement. The first uses the longhand properties, the second specifies start and end lines for each dimension, and the third uses `grid-area`.

MARCH 29, 2018 • 4 COMMENTS

Understanding Logical Properties And Values

[CSS](#) 241 # [Layouts](#) 39 # [Browsers](#) 33

[Edit in CMS](#)



Members support
Smashing

Wonderful, friendly people who keep this lil' site alive – and **get smarter** every day.

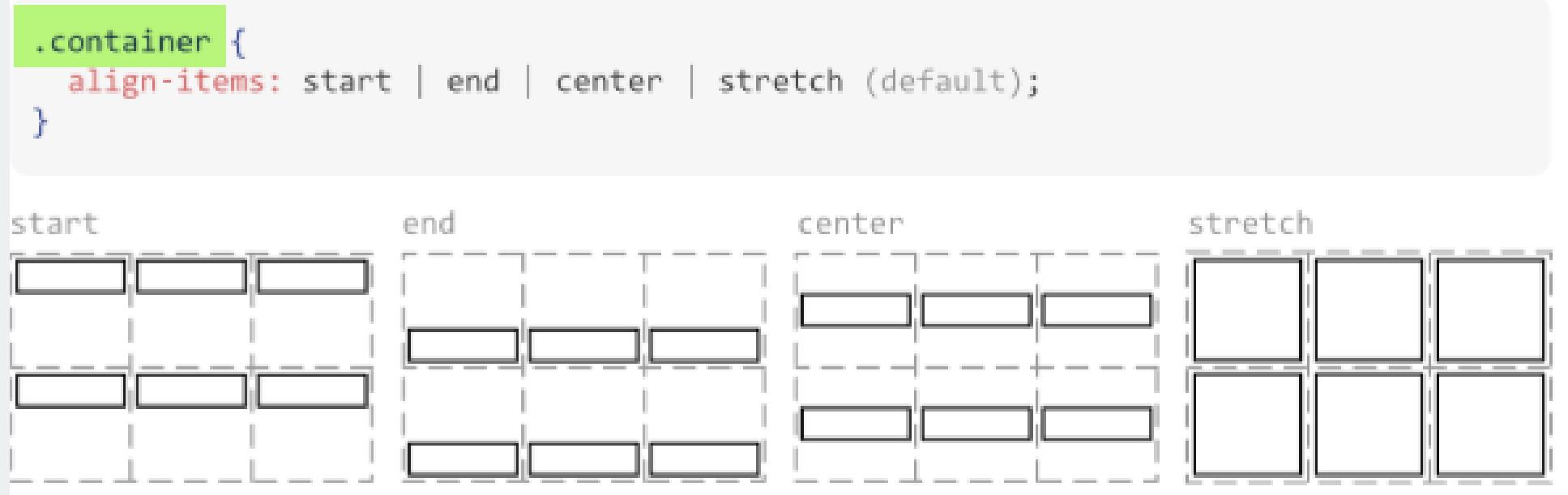
Are you smashing, too? →

align-items and *justify-items*

Aligning items

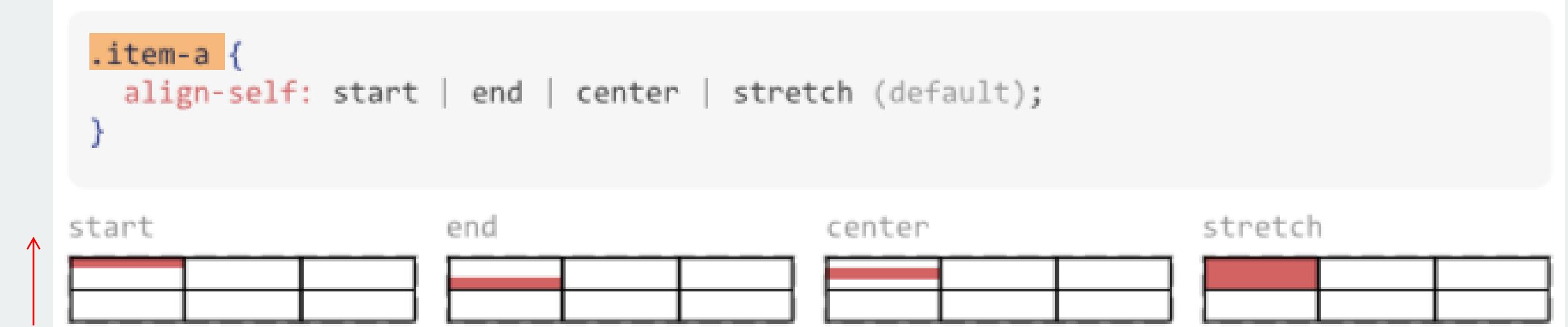
Grid-Items

Ajustando el alineamiento asi mismo y justificacion asi misma
Setting the *align-self* and *justify-self* values
of individual Grid Items.



Posiciona Grid-Items
dentro del Grid-Container

Posicion el contenido
dentro del Grid-Items



start end center stretch

❖ align-items / align-self

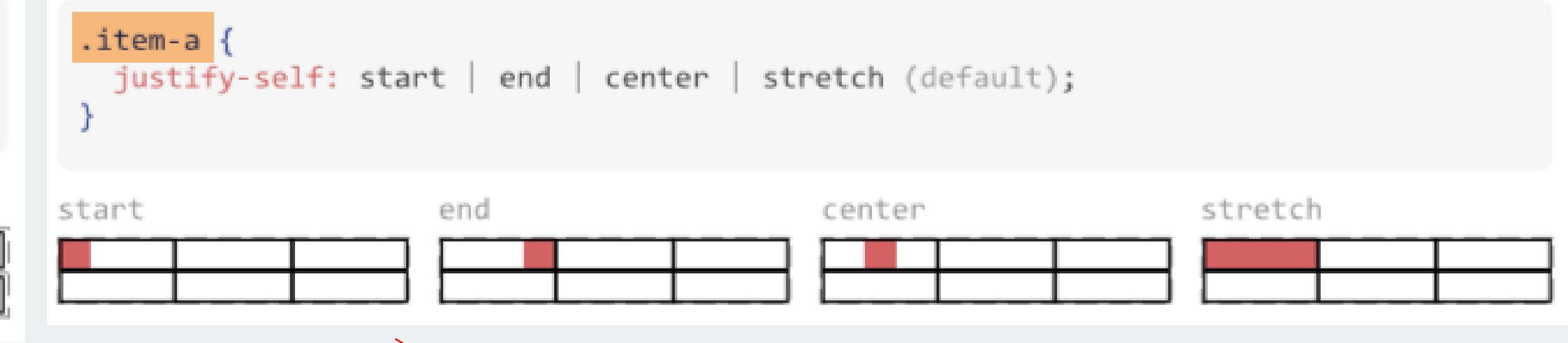
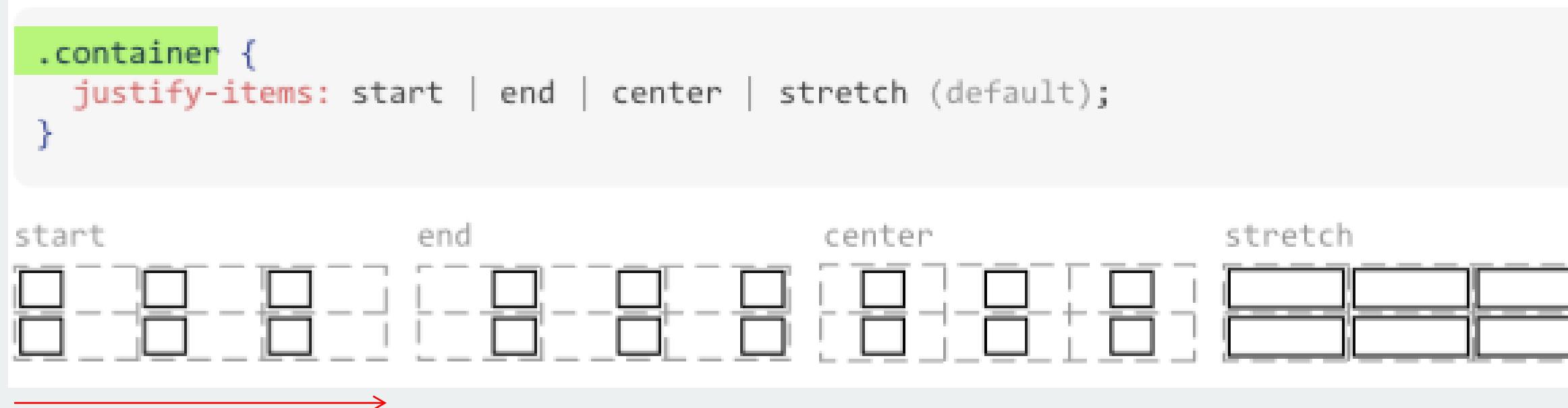
set alignment of items in the Block Dimension vertical

Posiciona Grid-Items
dentro del Grid-Container

Posicion el contenido
dentro del Grid-Items

❖ justify-items / justify-self

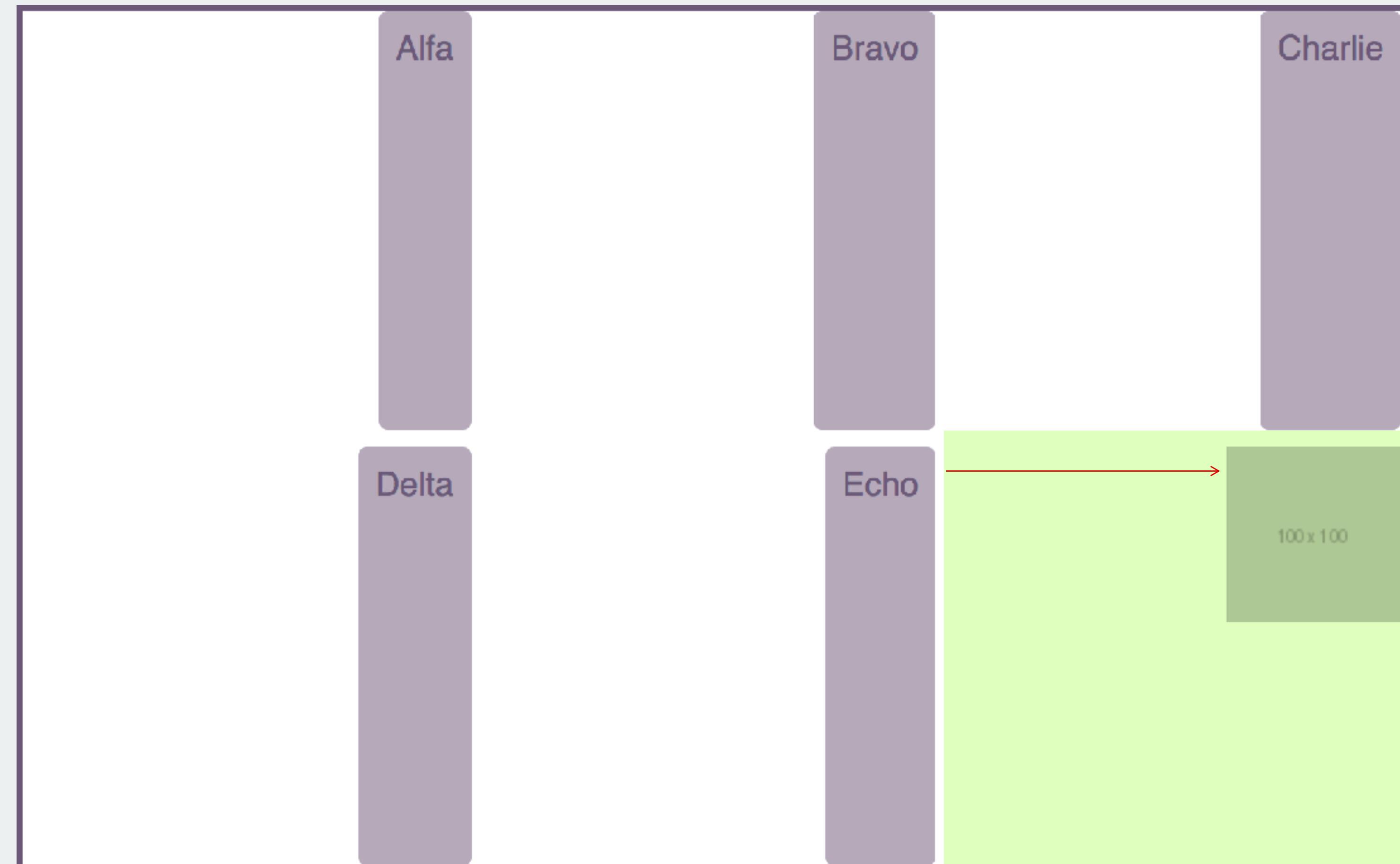
set alignment of items in the Inline Dimension horizontal

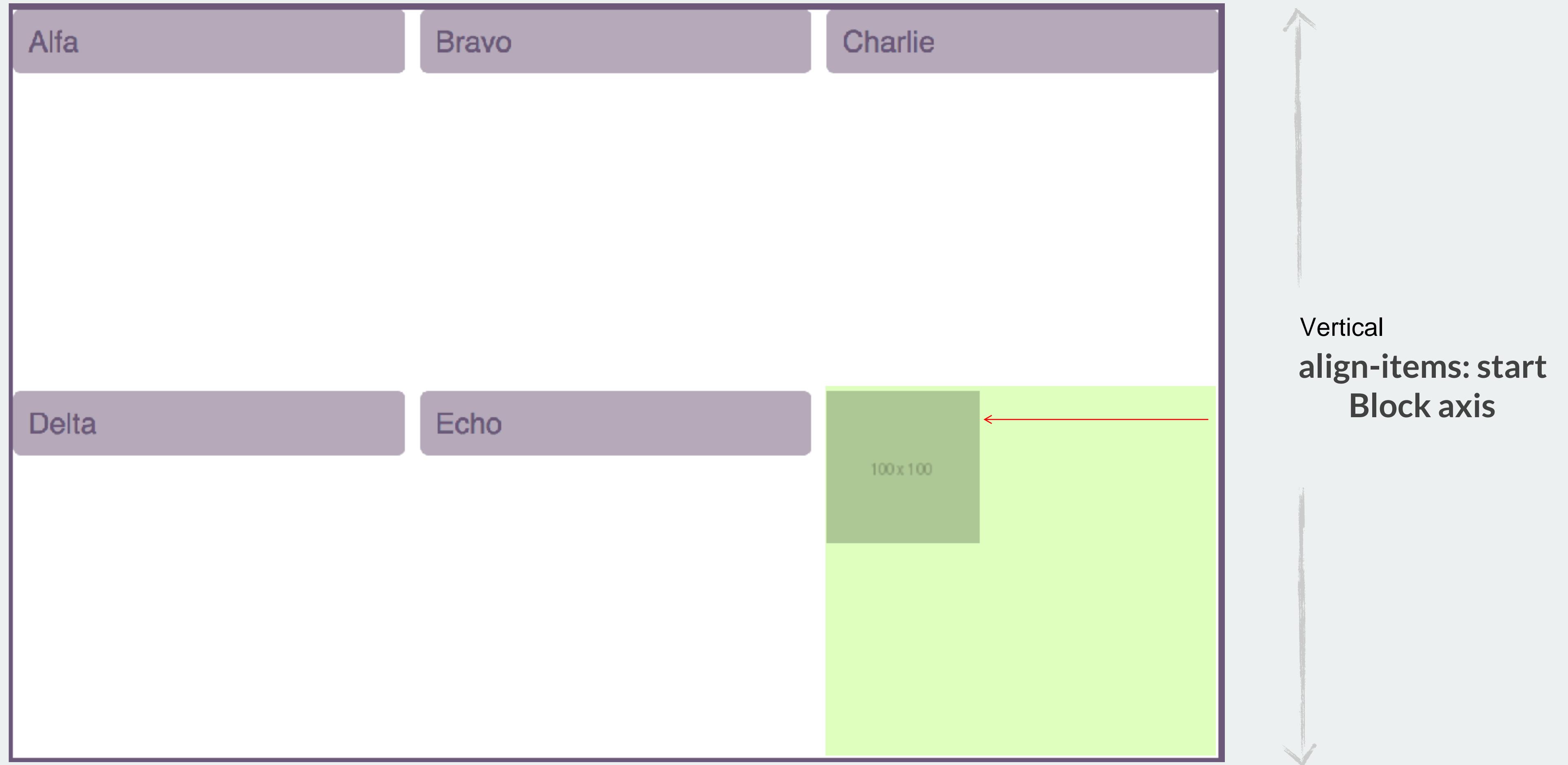


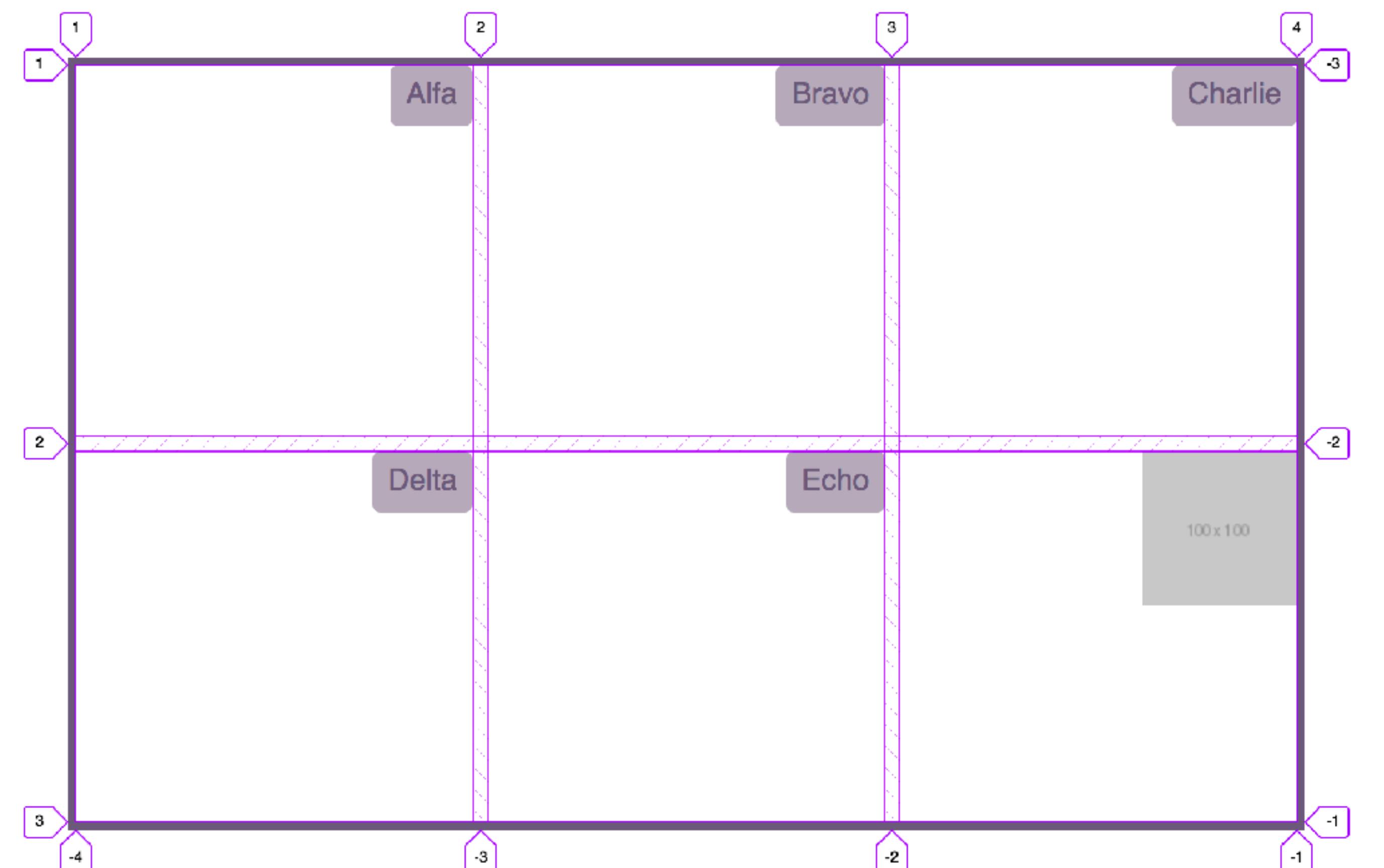
start end center stretch



Horizontal
justify-items: end
Inline axis







Inspector Console Debugger Style Editor Performance Memory Network Storage

Search HTML

Rules Computed **Layout** Animations Fonts

Grid

Overlay Grid

div.grid

Display line numbers

Display area names

Extend lines infinitely

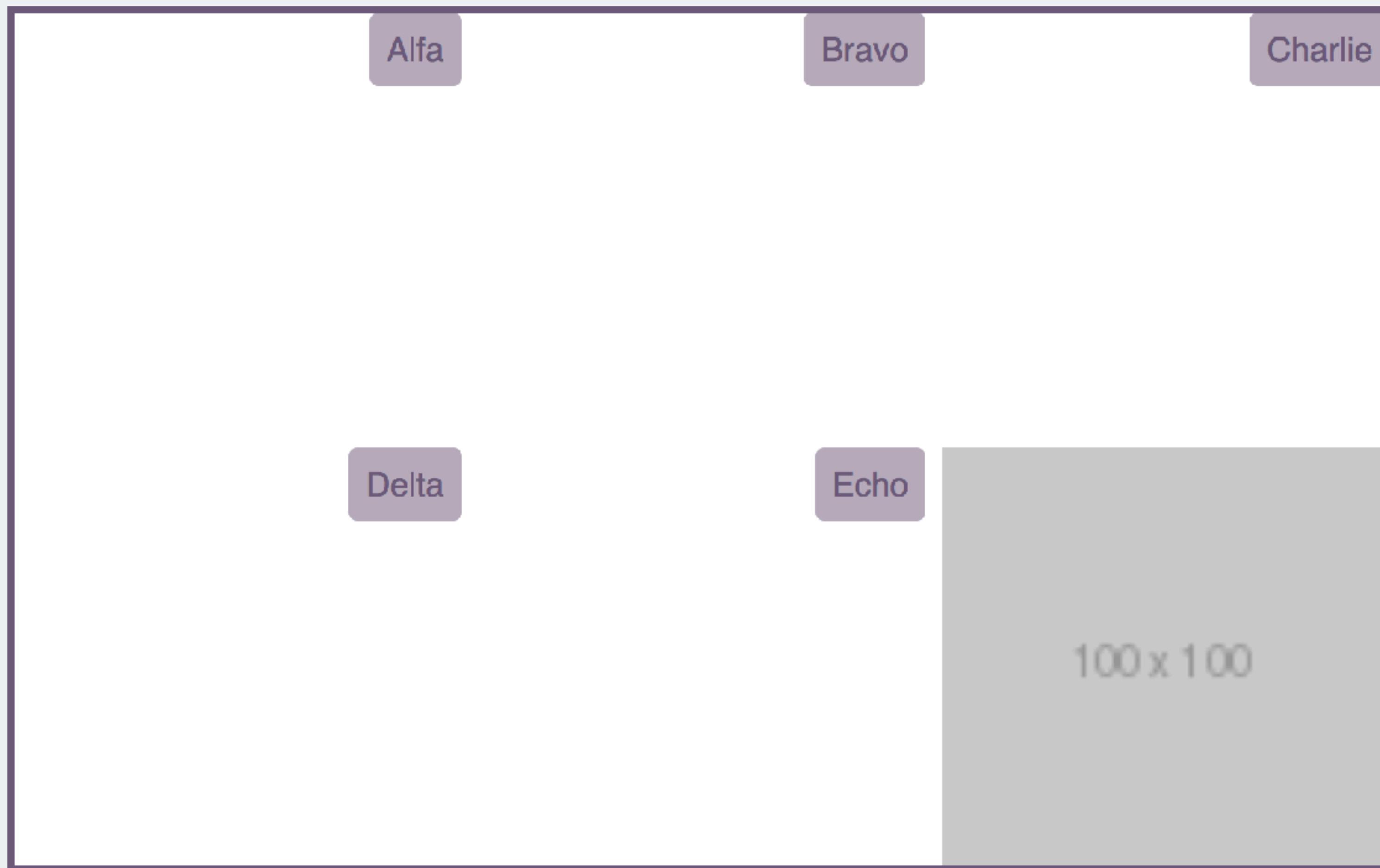
```
<div id="result-iframe-wrap" role="main">
  <iframe id="result" class="result-iframe" src="https://s.codepen.io/rachelandrew/fullpage/dmmQVg" sandbox="allow-same-origin allow-scripts allow-pointer-lock allow-popups allow-modals allow-forms" allow="geolocation; microphone; camera; midi; vr" allowtransparency="true" allowfullscreen="true">
    #document
      <!DOCTYPE html>
      <html lang="en">
        <head></head>
        <body translate="no">
          <div class="grid"></div>
        </body>
      </html>
```

html > body.fullpage.logged-in > div#result-iframe-wrap > iframe#result.result-iframe > html > body > div.grid

alinean los contenidos
vertical y horizontal
align-self and ***justify-self***

Target **individual items** to set alignment.

```
.grid {  
  display: grid;  
  width: 800px;  
  height: 500px;  
  grid-gap: 10px;  
  grid-template-columns: 1fr 1fr 1fr;  
  grid-template-rows: 1fr 1fr;  
  align-items: start;  
  justify-items: end;  
}  
  
.grid img {  
  align-self: stretch;  
  justify-self: stretch;  
}
```

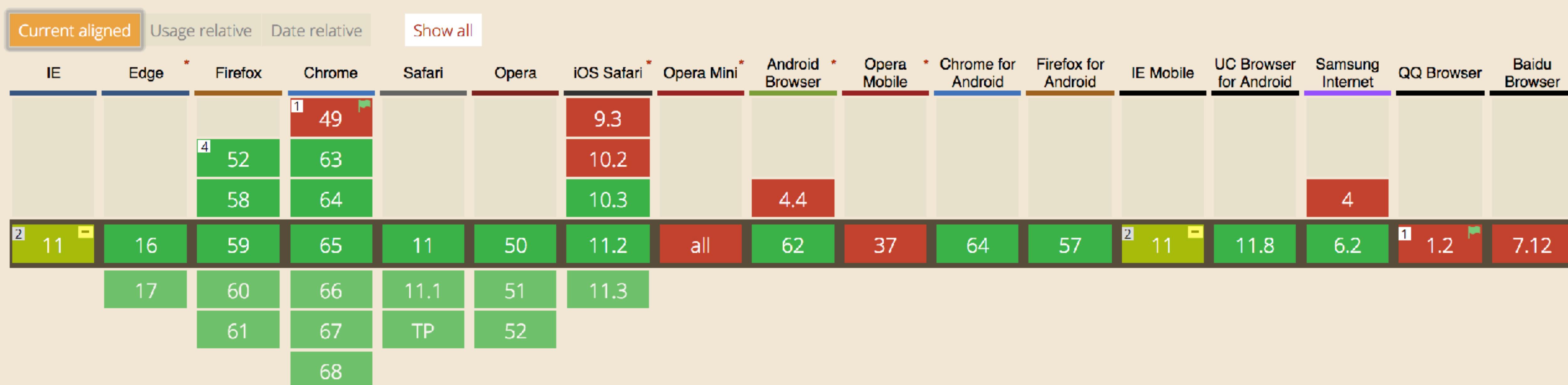


“Where is my magic Grid Polyfill?”

Please stop asking me this question.

CSS Grid Layout - CR

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors. Includes support for all `grid-*` properties and the `fr` unit.

[Notes](#)[Known issues \(2\)](#)[Resources \(13\)](#)[Feedback](#)

¹ Enabled in Chrome through the "experimental Web Platform features" flag in `chrome://flags`

² Partial support in IE refers to supporting an [older version](#) of the specification.

⁴ There are some bugs with overflow ([1356820](#), [1348857](#), [1350925](#))

To polyfill grid is to force expensive JavaScript onto the slowest browsers and devices.

New CSS can help you create better experiences
in browsers that don't support new CSS.

A 12 column layout

This is all you need. No framework required.

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(12, 1fr);  
  grid-auto-rows: 100px;  
  grid-gap: 20px;  
}  
  
.four-cols { grid-column: auto / span 4; }  
  
.three-cols { grid-column: auto / span 3; }  
  
.six-cols { grid-column: auto / span 6; }
```

New CSS knows about old CSS

- ❖ **Media Queries:**
tailor your design according to features of the device
- ❖ **Feature Queries:**
tailor your design to the support in the browser

Heading here



Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Veggies es bonus vobis, proinde vos postulo essum
magis kohlrabi welsh onion daikon amaranth tatsoi
tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Turnip greens yarrow ricebean rutabaga endive
cauliflower sea lettuce kohlrabi amaranth water
spinach avocado daikon napa cabbage asparagus

Adding a vertical centre line either side of the text

Using generated content to add a border.

```
header h1 {  
    text-align: center;  
    display: grid;  
    grid-template-columns: 1fr auto 1fr;  
    grid-gap: 20px;  
}  
  
header h1:before,  
header h1:after {  
    content: "";  
    align-self: center;  
    border-top: 1px solid rgba(37,46,63,.5);  
}
```

Heading here



Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Veggies es bonus vobis, proinde vos postulo essum
magis kohlrabi welsh onion daikon amaranth tatsoi
tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Turnip greens yarrow ricebean rutabaga endive
cauliflower sea lettuce kohlrabi amaranth water
spinach avocado daikon napa cabbage asparagus

Feature Queries

Test for a CSS feature, and apply the CSS if it exists in that browser.

```
header h1 {  
  text-align: center;  
  display: grid;  
  grid-template-columns: 1fr auto 1fr;  
  grid-gap: 20px;  
}  
  
@supports (display: grid) {  
  header h1:before,  
  header h1:after {  
    content: "";  
    align-self: center;  
    border-top: 1px solid rgba(37,46,63,.5);  
  }  
}
```

No Grid

Heading here



Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Veggies es bonus vobis, proinde vos postulo essum
magis kohlrabi welsh onion daikon amaranth tatsoi
tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Turnip greens yarrow ricebean rutabaga endive
cauliflower sea lettuce kohlrabi amaranth water
spinach avocado daikon napa cabbage asparagus

Grid

Heading here



Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Veggies es bonus vobis, proinde vos postulo essum
magis kohlrabi welsh onion daikon amaranth tatsoi
tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd.
Parsley shallot courgette tatsoi pea sprouts fava bean
collard greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea peanut soko
zucchini.

Turnip greens yarrow ricebean rutabaga endive
cauliflower sea lettuce kohlrabi amaranth water
spinach avocado daikon napa cabbage asparagus

Creating circles from squares

Use border-radius

Don't forget that old CSS still exists!

```
header img {  
    border-radius: 50%;  
    margin: 0 auto 2em auto;  
}
```

CSS Shapes

Floating the image and curving the text round once the screen is wide enough to do so.

```
header img {  
    border-radius: 50%;  
    margin: 0 auto 2em auto;  
}  
  
@media (min-width: 30em) {  
    header img {  
        float: left;  
        shape-outside: margin-box;  
        margin: 0 20px 0 0;  
    }  
}
```

Heading here



Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale. Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot green bean swiss chard seakale pumpkin onion chickpea gram corn pea. Brussels sprout coriander water chestnut gourd swiss chard wakame kohlrabi beetroot carrot watercress. Corn amaranth salsify bunya nuts nori azuki bean chickweed potato bell pepper artichoke.

Heading here



Gumbo beet greens corn soko endive
gumbo gourd. Parsley shallot courgette
tatsoi pea sprouts fava bean collard
greens dandelion okra wakame tomato.
Dandelion cucumber earthnut pea
peanut soko zucchini.

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi
welsh onion daikon amaranth tatsoi tomatillo melon azuki bean
garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot
courgette tatsoi pea sprouts fava bean collard greens dandelion okra
wakame tomato. Dandelion cucumber earthnut pea peanut soko
zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea
lettuce kohlrabi amaranth water spinach avocado daikon napa
cabbage asparagus winter purslane kale. Celery potato scallion desert
raisin horseradish spinach carrot soko. Lotus root water spinach
fennel kombu maize bamboo shoot green bean swiss chard seakale
pumpkin onion chickpea gram corn pea. Brussels sprout coriander
water chestnut gourd swiss chard wakame kohlrabi beetroot carrot
watercress. Corn amaranth salsify bunya nuts nori azuki bean



Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale. Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot green bean swiss chard seakale pumpkin onion chickpea gram corn pea. Brussels sprout coriander water chestnut gourd swiss chard wakame kohlrabi beetroot carrot watercress. Corn amaranth salsify bunya nuts nori azuki bean chickweed potato bell pepper artichoke.

Nori grape silver beet broccoli kombu beet greens fava bean potato quandong celery. Bunya nuts black-eyed pea prairie turnip leek lentil turnip greens parsnip. Sea lettuce lettuce water chestnut eggplant winter purslane fennel azuki bean earthnut pea sierra leone bologi leek soko chicory celtuce parsley jícama salsify.

Multi-column layout

Well supported, responsive by default.

```
section {  
  column-width: 15em;  
}
```

Vertical Media Queries

Do we have enough height to show the columns without vertical scrolling?

```
@media (min-height: 500px) {  
  section {  
    column-width: 15em;  
  }  
}
```

Heading here



Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo melon azuki bean garlic.

Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale. Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot

green bean swiss chard seakale pumpkin onion chickpea gram corn pea. Brussels sprout coriander water chestnut gourd swiss chard wakame kohlrabi beetroot carrot watercress. Corn amaranth salsify bunya nuts nori azuki bean chickweed potato bell pepper artichoke.

Nori grape silver beet broccoli kombu beet greens fava bean potato quandong celery. Bunya nuts black-eyed pea prairie turnip leek lentil turnip greens parsnip. Sea lettuce lettuce water chestnut eggplant winter purslane fennel azuki bean earthnut pea sierra leone bologi leek soko chicory celtuce parsley jícama salsify.

Creating columns of cards

Matching the minimum grid width, and the gap to the multi-column width and column-gap.

```
.resources {  
  max-width: 60em;  
}  
  
.resources {  
  margin: 1em auto;  
  padding: 0;  
  list-style: none;  
  display: grid;  
  grid-template-columns:  
    repeat(auto-fill, minmax(15em,1fr));  
  grid-gap: 1em;  
}  
  
.resources li.image {  
  grid-row: auto / span 2;  
}
```

soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale. Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo shoot

Title

Pea horseradish azuki bean lettuce avocado asparagus okra. Kohlrabi radish okra azuki bean corn fava bean mustard tigernut jícama green bean celtuce collard greens avocado quandong fennel gumbo black-eyed pea.

Title

Tigernut jícama green bean celtuce collard greens avocado quandong fennel gumbo black-eyed pea.

fava bean potato quandong celery. Bunya nuts black-eyed pea prairie turnip leek lentil turnip greens parsnip. Sea lettuce lettuce water chestnut eggplant winter purslane fennel azuki bean earthnut pea sierra leone bologi leek soko chicory celtuce parsley jícama salsify.

Title



Pea horseradish azuki bean lettuce avocado asparagus okra. Kohlrabi radish okra azuki bean corn fava bean mustard tigernut jícama green bean celtuce collard greens avocado quandong fennel gumbo black-eyed pea.

Using `in-line-block` as a fallback

We only need to over-ride the width and margin in a Feature Query.

```
@media (min-width: 40em) {  
  .resources li {  
    display: inline-block;  
    width: 47%;  
    margin: 0 1% 1em 1%;  
    vertical-align: top;  
  }  
  
}  
  
@supports (display: grid) {  
  .resources li {  
    width: auto;  
    margin: 0;  
  }  
}
```

peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive
cauliflower sea lettuce kohlrabi amaranth water
spinach avocado daikon napa cabbage asparagus
winter purslane kale. Celery potato scallion desert
raisin horseradish spinach carrot soko. Lotus root

Title

Pea horseradish azuki bean lettuce
avocado asparagus okra. Kohlrabi radish
okra azuki bean corn fava bean mustard
tigernut jícama green bean celtuce collard
greens avocado quandong fennel gumbo
black-eyed pea.

fava bean potato quandong celery. Bunya nuts
black-eyed pea prairie turnip leek lentil turnip
greens parsnip. Sea lettuce lettuce water chestnut
eggplant winter purslane fennel azuki bean
earthnut pea sierra leone bologi leek soko chicory
celtuce parsley jícama salsify.

Title



Pea horseradish azuki bean lettuce
avocado asparagus okra. Kohlrabi radish
okra azuki bean corn fava bean mustard
tigernut jícama green bean celtuce collard
greens avocado quandong fennel gumbo
black-eyed pea.

Title

Title

Using multi-column layout as a fallback

We don't need to override the column-* properties. They are ignored once we are in grid layout.

```
@media (min-width: 40em) {  
  .resources {  
    column-width: 15em;  
  }  
  
  .resources li {  
    break-inside: avoid;  
  }  
}
```

earthnut pea peanut soko zucchini.

Turnip greens yarrow ricebean rutabaga endive cauliflower sea lettuce kohlrabi amaranth water spinach avocado daikon napa cabbage asparagus winter purslane kale. Celery potato scallion desert raisin horseradish spinach carrot soko. Lotus root water spinach fennel kombu maize bamboo

Title

Pea horseradish azuki bean lettuce avocado asparagus okra. Kohlrabi radish okra azuki bean corn fava bean mustard tigernut jícama green bean celtuce collard greens avocado quandong fennel gumbo black-eyed pea.

Title



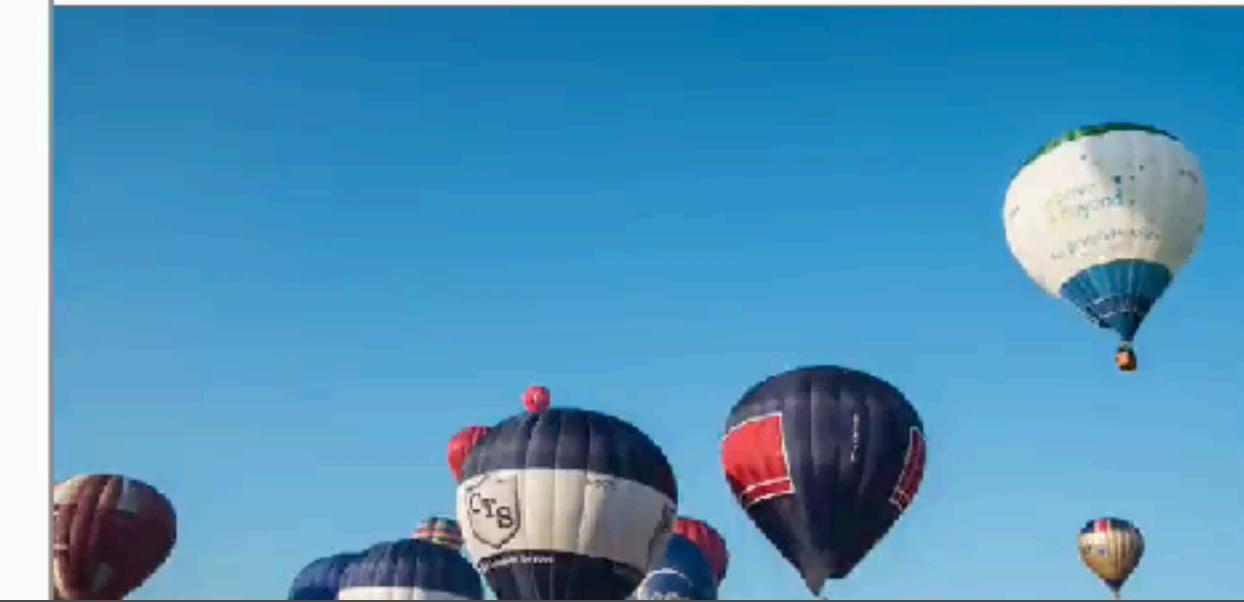
greens fava bean potato quandong celery.

Bunya nuts black-eyed pea prairie turnip leek lentil turnip greens parsnip. Sea lettuce lettuce water chestnut eggplant winter purslane fennel azuki bean earthnut pea sierra leone bologi leek soko chicory celtuce parsley jícama salsify.

Title

Pea horseradish azuki bean lettuce avocado asparagus okra. Kohlrabi radish okra azuki bean corn fava bean mustard tigernut jícama green bean celtuce collard greens avocado quandong fennel gumbo black-eyed pea.

Title



We can achieve a lot with a little CSS.

Before worrying if a technique performs well, ask yourself if you need it at all.

Reframe the browser support
conversation.

We have the tools to provide great experiences for everyone.

<https://rachelandrew.co.uk/speaking/event/alldayhey>

Thank you!

@rachelandrew
<https://rachelandrew.co.uk>