CSS Grid Layout

C	S	S	
G	R	I	D

CODE VIEW

```
<style>
.container > div {
    background: #eee;
    padding: 1em;
    }
.container > div:nth-child(odd) {
    background: #ddd;
    }
</style>
<body>
<div class="container">
    <div>Grid item one.</div>
    <div>Grid item two.</div>
    <div>Grid item three.</div>
    <div>Grid item four.</div>
</div>
</body>
```

GRID BASICS

GETTING STARTED

We'll begin by adding a little CSS to automatically shade the grid items in alternating shades of gray. Use the CSS code to the left to take care of that.

Building a Grid Container and Grid Items

Next, well use HTML to create a generic element (a **div** with a class of **.container**) to serve as our **grid container**. All CSS Grid layouts start with a grid container. Inside the grid container will be a series of **grid items** ("elements" or "boxes") that will contain the content of this page. Use the HTML code to the left.

Below is what you get in a browser.

Grid item one.

Grid item two.

Grid item three.

Grid item four.

```
.container {
  display: grid;
  grid-template-columns: 70% 30%;
  grid-column-gap: 1em;
  grid-row-gap: 1em;
  }
```

El contenedor comienza declarando un tipo de visualización de 'cuadrícula' con display: grid.
A continuación, utilizamos columnas de plantilla de cuadrícula para indicar cuántas columnas queremos. En este caso, hay dos valores que siguen a la propiedad, lo que significa que será una cuadrícula de dos columnas.
La primera columna usará el 70% del espacio y la segunda columna usará el 30%.

CONTROLLING THE GRID CONTAINER

So far, we haven't done anything with CSS Grid. We'll be using the **grid container** to define the number of columns and how rows will be built.

← The container starts by declaring a display type of "grid" with display: grid. Next, we use grid-template-columns to indicate how many columns we want. In this case there are two values following the property meaning this will be a two column grid. The first column will use 70% of the space and the second column will use 30%.

NOTE: Units could be percentages, pixels, rems or even a combination such as 120px 20% 2fr 1fr;. *More on fr below.*

grid-column-gap creates a gap between the columns and grid-row-gap creates a gap between the rows. You can control both with simply grid-gap: 1em.

THE "IMPLICIT" GRID

Tenemos 2 columnas definidas y cuatros 'Grid-items' o DIVs dentro del HTML , los 2 primeros elementos se colocan en la 1º fila y como caben más , saltan a una 2º fila

Cuantas más columnas tengamos , más DIV se iran colocando para ocupar ese ancho de la fila aunque sobresalga del contenedor principal

Salta a la siguente fila porque no hay mas columnas para agregar más DIVs

Viewed in a browser, you'll see the 70%/30% split. Also notice that because there are four elements (divs) in the HTML, only the first two elements are on the first row.

Los dos elementos finales se colocaron automáticamente en la segunda fila a través del algoritmo de colocación automática de CSS Grid. Esto se conoce como la cuadrícula implícita. No tuvimos que decir explícitamente en qué filas deberían continuar los elementos. Estaba implícito que los siguientes elementos tendrían que moverse hacia abajo a la siguiente fila.

The final two elements automatically placed themselves on the second row via CSS Grid's auto-placement algorithm. This is referred to as the **implicit grid**. We didn't have to explicitly say what rows the elements should go on. It was implied that the next elements would have to move down to the next row.

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

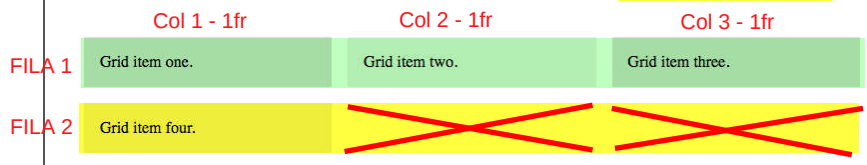
CSS Grid nos trajo una nueva unidad llamada 'fractional' o fr. Un 'fr' es una unidad de tamaño flexible que se puede utilizar con otras unidades, como porcentaje, o solo. 1fr significa que el elemento debe ocupar una fracción del espacio restante o sin asignar.

USING THE "FR" UNIT

CSS Grid brought us a new unit called "fractional" or fr. An "fr" is a flexible sizing unit that can be used with other units, such as percentage, or alone. 1fr means the element should take up one fraction of the remaining or unassigned space.

← In the example to the left, we've decided to use 3 columns, each using 1 fraction (1fr) of the available space. Notice too that instead of using both grid-column-gap and grid-row-gap, we have instead handled both with the shorthand property grid-gap.

The result in a browser is shown below. Notice that with four elements, the fourth box is moved down to the second row.

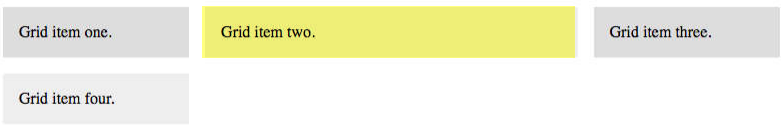



```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-gap: 1em;
    }
```

Changing Ratios of Fractionals

← In this example, we have changed the second column to using 2 fractions of the available space. We are still using just 3 columns, but the second column is now twice as wide as the first and third columns. As before, the fourth element is moved down to the second row via the "implicit grid."

The result in a browser is shown below.



Seguir por aqui
05/02/2020 ,22:17:38

```
<div class="container">
    <div>Grid item one.</div>
    <div>Grid item two.</div>
    <div>Grid item three.</div>
    <div>Grid item four.</div>
    <div>Grid item five.</div>
    <div>Grid item six.</div>
    <div>Grid item seven.</div>
    <div>Grid item eight.</div>
</div>
```

Add More Elements

We are going to need a few more boxes (elements) for content to illustrate the next couple of examples.

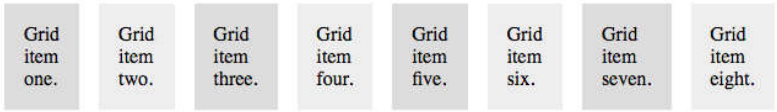
← Add four more elements to the HMTL as shown to the left.

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

An Eight Column Grid

If you want an eight column layout, you can simply write our 1fr eight times, as shown to the left.

The result in a browser is shown below.



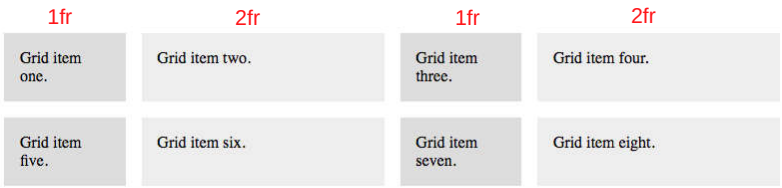

```
.container {
    display: grid;
    grid-template-columns: repeat(2, 1fr 2fr);
    grid-gap: 1em;
    }
```

Repeating Fractional Units

A better alternative to writing out 1fr eight times is to use a new function in CSS Grid called repeat(). Now, if we need 8 equal width columns, instead of writing it out as grid-template-columns: 1fr 1fr 1fr 1fr 1fr 1fr 1fr 1fr, we can simply write out grid-template-columns: repeat(8, 1fr).

← In the example to the left, we are repeating the pattern 1fr 2fr two times.

The result in a browser is shown below.




```
<div class="container">
    <div>Grid item one. Grid item one. Grid item one. Grid item one. Grid item one. Grid item one. Grid item one. Grid item one.</div>
    <div>Grid item two.</div>
    <div>Grid item three.</div>
    <div>Grid item four.</div>
    <div>Grid item five.</div>
    <div>Grid item six.</div>
    <div>Grid item seven.</div>
    <div>Grid item eight.</div>
</div>
```

Add More Content

We are going to need more content in one of our elements to see the effect of the next lesson.

← Add more text to the first grid item, as shown to the left.

ADDING AND CONTROLLING GRID ROWS

Como aprendimos anteriormente , nosotros no tenemos que explicitamente declarar/exponer el numero de filas Filas son implicitas creadas via CSS GRID

As we learned earlier, we don't have to explicitly state the number of rows. Rows are "implicitly" created via CSS Grid's "auto placement algorithm."

'algoritmo de colocación automática.'

Tampoco es necesario especificar una altura para ninguno de los elementos o filas de la cuadrícula.

There is also no need to specify a height for any of the grid elements or rows. That's because CSS Grid automatically expands the height of an element (box) to fit the content inside.

que elemento grid uno ha expandido

Observe en el siguiente ejemplo que el elemento de cuadrícula uno ha expandido su altura para acomodar el mayor contenido que contiene

Notice in the example below that grid item one has expanded its height to accommodate the increased content it holds. The other elements on that row also expand to the same height.

su altura a acomodar el incrementado.contenido que ello tiene.

Los otros elementos sobre esa fila también se expanden a la misma altura.

esto es algo que era muy dificil hacer en los viejos dias de usar layout float

This is something that was very difficult to do in the old days of "floated" layouts. (It was referred to as the Holy Grail Layout problem). With CSS Grid, it's a breeze.

grid-auto-rows : especifica el tamaño de una nueva fila creada de forma implicita

```
.container {
    display: grid;
    grid-template-columns: repeat(2, 1fr 2fr);
    grid-auto-rows: minmax(120px, auto);
    grid-gap: 1em;
    }
```

Cada vez que el tamaño de los elementos de las columnas es superior al ancho de estos se crea una nueva fila con las propiedades de minmax()

Hay momentos en los que uno quiere controlar la altura mínima de una fila y al mismo tiempo permitir que el contenido crezca en altura si se agrega más contenido. Para eso, usamos la propiedad grid-auto-rows: con una función llamada minmax ().

grid-template-columns: repeat(2, 1fr 2fr)

Creamos grupos de 2 elementos grid-items

cada grupo tiene una columna de 1fr y otra columna de 2fr

Controlling Row Heights

hay veces cuando uno desea controlar el minimo alto

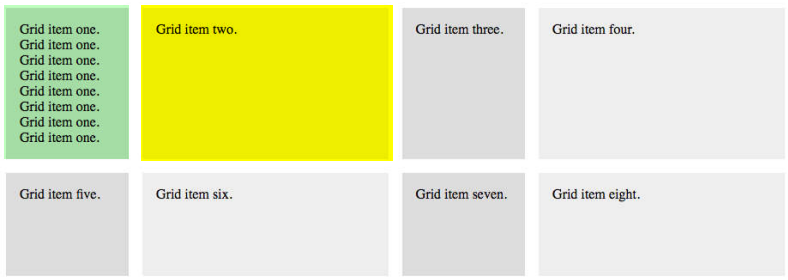
There are times when one wants to control the minimum height of a row while still allowing the content to grow in height if more content is added. For that, we use the grid-auto-rows: property with a function called minmax().

de una fila mientras aun permite el contenido creciendo en alto si mas contenido es añadido

← Add the line to the left and notice the effect on the rows. The first number (120px) corresponds to the minimum height of a row. The second number corresponds to the maximum height a row can be (auto). As you can see below, the content in the first grid item was more than 120 pixels tall, so the content expanded to the height it needed. However, none of the grid items on the second row needed more than 120 pixels, so that row is set to that height and it slightly increased the height of the second row.

ligeramente

1fr 2fr 1fr 2fr



CSS Grid Layout
Steven D. Anderson, Ph.D.
James Madison University
School of Media Arts & Design
anderssd@jmu.edu