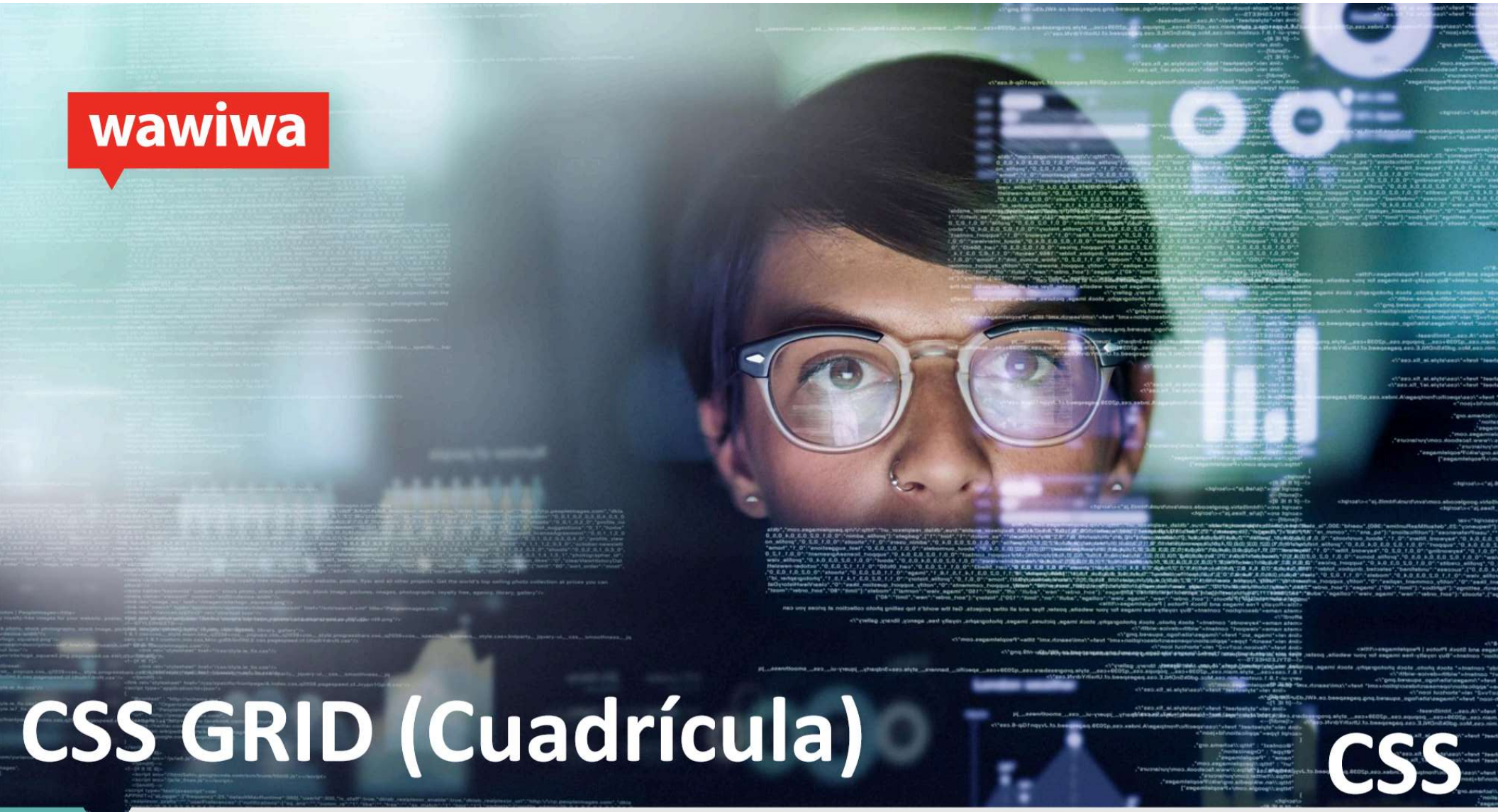


wawiwa

Flex grid (cuadrícula flexible) - more on mdn

CSS Grid Layout (diseño de cuadrícula) es un sistema de diseño bidimensional para la web. Te permite mostrar contenido en filas y columnas. Tiene muchas funcionalidades que facilitan la creación de diseños complejos de manera sencilla. Este artículo explica todo lo que tienes que saber para empezar con diseño de páginas.

¿Qué es "grid layout"?

Un “grid” es una colección de líneas horizontales y verticales creando un patrón con el cual podemos alinear nuestros elementos de diseño. Pueden ayudarnos a crear diseños en los cuales nuestros elementos no van a saltar de un lado a otro o cambiar su ancho mientras nos movemos de página a página, dando una mejor consistencia a nuestros sitios web.

Un “grid” típicamente va a tener **columnas**, **filas** y espacios entre cada fila y columna. A los espacios comúnmente se les llama ”**gutters”**.

wawiwa

Definiendo un grid

Para definir un grid se usa el valor `grid` en la propiedad `display` . Como con “Flexbox”, este habilita un diseño de cuadrícula (Grid Layout); todos los hijos directos de este contenedor se convierten en elementos del grid (“grid items”). Y este es el CSS dentro de tu archivo:

```
.container { display: grid; }
```

Para ver algo que se parece más a una cuadrícula, debemos agregar algunas columnas a la cuadrícula. Agreguemos tres columnas de 200 píxeles. Puedes usar cualquier unidad de longitud o porcentaje para usar estas columnas.

```
.container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
}
```

wawiwa

Antes de Grid - Fase 1

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```


```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```

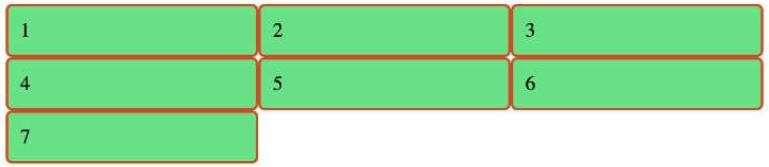
wawiwa

Después de Grid Fase 2

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Para ver algo que se parece más a una cuadrícula, debemos agregar algunas columnas a la cuadrícula. Agreguemos tres columnas de 200 píxeles. Puedes usar cualquier unidad de longitud o porcentaje para usar estas columnas. Agrega la segunda declaración a tus reglas de CSS, después recarga la página. Debes de ver que los elementos se han reacomodado para que haya uno en cada celda del grid

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

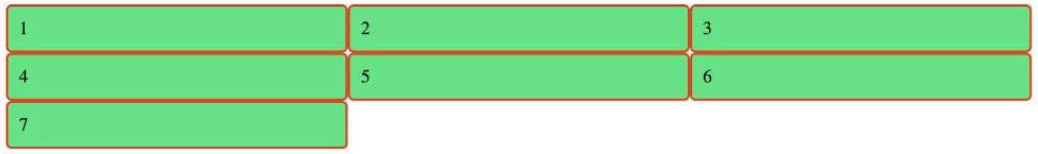
Después de Grid - Fase 3

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

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

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Cuadrículas flexibles con la unidad frt

Además de crear cuadrículas usando longitudes y porcentajes, podemos usar la unidad `fr` para dimensionar columnas y filas. Esta unidad representa una fracción del espacio disponible del contenedor del grid. Cambia tu listado a la siguiente definición, creando tres `1fr` pistas.

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```

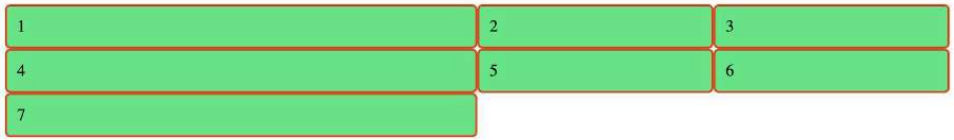


wawiwa

Después de Grid - Fase 4

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

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

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

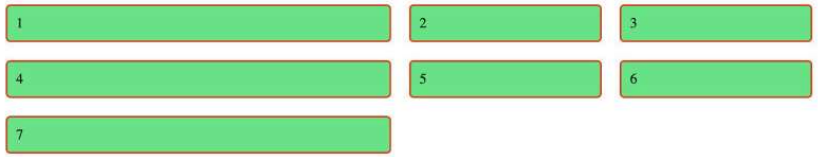
Después de Grid - Fase 5

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container{
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 20px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Gaps between tracks

To create gaps between tracks we use the properties `column-gap` for gaps between columns, `row-gap` for gaps between rows and `gap` as a shorthand for both.

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



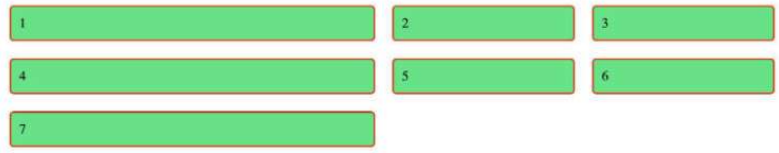
wawiwa

Después de Grid - Fase 6

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container {
    display: grid;
    grid-template-columns: 2fr 1fr 1fr;
    gap: 20px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Nota: Las propiedades de gap (`column-gap`, `row-gap` y `gap`) solían ser prefijas por `grid-`, pero esto se cambió para que puedan ser usados en múltiples métodos de diseño. Las versiones prefijadas se mantienen como un alias, así que será seguro usarlas por un tiempo. Para estar del lado seguro, podrías duplicar y usar las 2 propiedades para que tu código sea a prueba de balas.

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

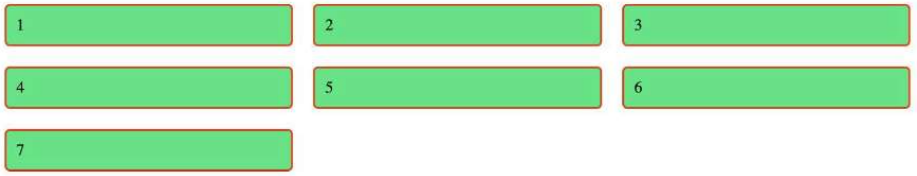
Después de Grid - Fase 7

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 20px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Repitiendo el listado de pistas

Puedes repetir todas o casi una sección de tu listado de pistas usando la función `repeat()` de CSS. Cambia tu listado de pistas a esto

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

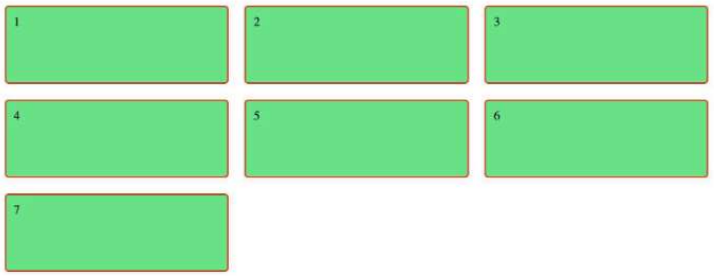
Después de Grid - Fase 8

```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-auto-rows: 100px;
    gap: 20px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Hasta ahora solo hemos especificado pistas de columnas, sin embargo, se están creando filas para contener nuestro contenido. Esto es un ejemplo de la cuadrícula explícita frente a la implícita. La cuadrícula explícita es la que creas utilizando `grid-template-columns` o `grid-template-rows`. La cuadrícula implícita se crea cuando el contenido se coloca fuera de esa cuadrícula, como en nuestras filas.

Por defecto,, las pistas creadas en la cuadrícula implícita tienen un tamaño `automático,` lo que en general significa que son lo suficientemente grandes como para acomodar su contenido. Si deseas darle un tamaño a las pistas de la cuadrícula implícita, puedes utilizar las propiedades grid-auto-rows y grid-auto-columns. Si agregas grid-auto-rows con un valor de 100px a tu CSS, verás que esas filas creadas ahora tienen 100 píxeles de altura.

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

Después de Grid - Fase 9 min max

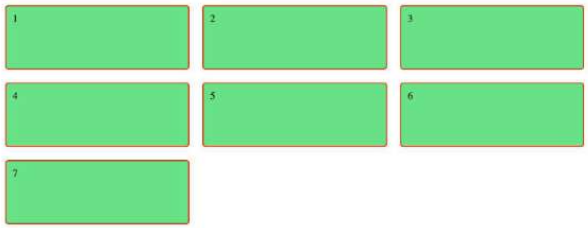
```
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container {
    display: grid;
    grid-template-columns: repeat(auto-fill,
    minmax(200px, 1fr));
    grid-auto-rows: minmax(100px, auto);
    gap: 20px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

La función minmax()

Nuestras pistas de 100 píxeles de altura no serán muy útiles si agregamos contenido a esas pistas que sea más alto que 100 píxeles, lo que causaría desbordamiento. Sería mejor tener pistas que tengan al menos 100 píxeles de altura y que aún puedan expandirse si se agrega más contenido. Un hecho bastante básico sobre la web es que nunca sabes realmente cuán alto será algo: contenido adicional o tamaños de fuente más grandes pueden causar problemas con los diseños que intentan ser perfectos en píxeles en todas las dimensiones.

La función `minmax()` nos permite configurar un tamaño mínimo y máximo a casa pista, por ejemplo, `minmax(100px, auto)`. El tamaño mínimo es 100 pixeles, pero el máximo es `auto`, que se expande para acomodar más contenido. Intenta cambiar `grid-auto-rows` para usar un valor minmax:

```
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

Después de Grid - Fase 10

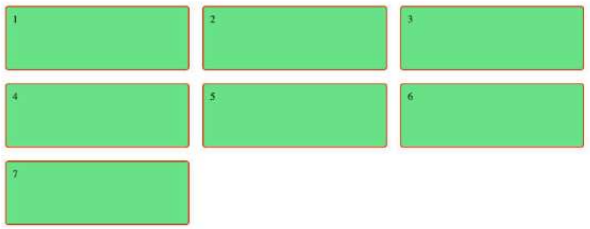
```css
body {
    width: 90%;
    max-width: 900px;
    margin: 20px auto;
}

.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-auto-rows: minmax(100px, auto);
    gap: 20px;
}

.container > div {
    border-radius: 5px;
    padding: 10px;
    background-color: rgb(10, 233, 126);
    border: 2px solid rgb(230, 65, 15);
}
```

Todas las columnas que quepan

Podemos combinar algunas de las lecciones que hemos aprendido sobre la lista de pistas, la notación de repetición y `minmax()` para crear un patrón útil. A veces es útil poder pedirle a la cuadrícula que cree tantas columnas como quepan en el contenedor. Hacemos esto estableciendo el valor de `grid-template-columns` usando la función `repeat()`, pero en lugar de pasar un número, pasamos la palabra clave `auto-fill`. Para el segundo parámetro de la función, usamos `minmax()` con un valor mínimo igual al tamaño mínimo de pista que nos gustaría tener y un máximo de 1fr.

Prueba esto es tu archivo usando el siguiente:

```html
<body>
    <div class="container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
        <div>4</div>
        <div>5</div>
        <div>6</div>
        <div>7</div>
    </div>
</body>
```



wawiwa

Ejemplo Final: link

https://github.com/mdn/learning-area/blob/main/css/css-layout/grids/8-placement-starting-point.html

wawiwa

¿Preguntas?