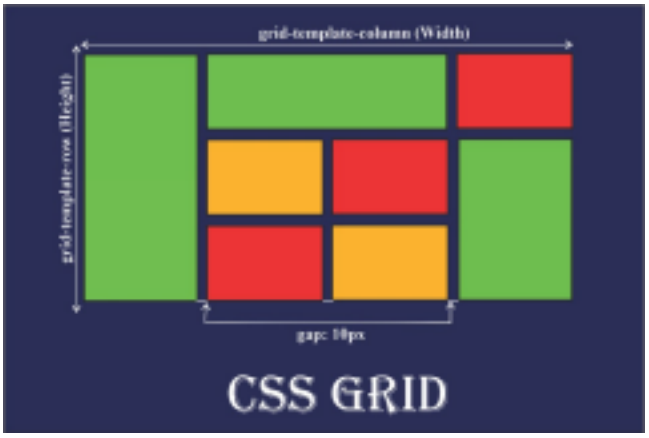
Capítulo 5. GRID LAYOUT

Que es el grid layout

El grid layout es un sistema bidimensional, que transforma un elemento HTML en una cuadrícula. Los elementos hijos de este pueden ser posicionados dentro de las celdas de esta cuadrícula, o en áreas definidas arbitrariamente con reglas CSS.



5.1 Grid layout – un ejemplo básico

En el HTML tenemos un div id="cuadrícula", dentro del cual aparecen anidados otros 5 div class="ítem".

```
<div id="cuadricula">
        <div class="item"><p>1</p></div>
        <div class="item"><p>2</p></div>
        <div class="item"><p>3</p></div>
        <div class="item"><p>4</p></div>
        <div class="item"><p>5</p></div>
</div>
```


En el CSS lo más importante son estas líneas de código:

```
#cuadricula {
 /*transforma la #cuadrícula en un contenedor grid:*/
 display: grid;
 /*esto genera 3 columnas iguales cuya anchura es 1fr */
```

```
 grid-template-columns:1fr 1fr 1fr;
 /*establece el espaciado entre los ítems del grid*/
 grid-gap:1em;
}
```

La primera línea de código transforma la #cuadricula en un contenedor grid:

```
display: grid;
```

La segunda línea de código crea la estructura de columnas:

```
grid-template-columns:1fr 1fr 1fr;
```

La propiedad grid-template-columns establece:

- La #cuadricula tiene tres columnas.
- La anchura de cada columna dentro del contenedor grid es de una fracción de grid (1fr), o sea, en este caso, la tercera parte del ancho de la #cuadricula.

En la tercera línea de código, la propiedad grid-gap establece el tamaño del espacio que tiene que haber entre columnas, en este caso una distancia de 1em:

```
grid-gap:1em;
```

2

Propiedades del contenedor grid

Propiedad	Posibles valores	Descripción	Defecto

display	*grid | inline-grid | subgrid;*	Transforma un elemento HTML en un contenedor grid. Observación: las propiedades column, float, clear, y vertical-align no tienen ningún efecto dentro de un contenedor grid.	
grid-template-columns **grid-template-rows**	*[linea] 1fr [linea] 1fr [linea] 1fr;*	Crea la estructura de columnas (columns) o filas (rows) del contenedor grid. Toma una lista nombres de líneas (entre corchetes, opcional) y valores separadas por espacios en blanco.	none
grid-template-areas	*grid-template-areas:"a a a" "b c c" "b c c";*	Define las áreas del contenedor grid. Toma como valor una lista de nombres para las áreas del contenedor o none (no define ninguna área)	none
grid-rows-gap **grid-column-gap**	*grid-rows-gap: 1em;* *grid-columns-gap:.5em;*	Definen el espacio que tiene que haber entre las columnas y las filas del contenedor grid.	0
grid-gap	*grid-gap: grid-rows-gap / grid-columns-gap*	es un método abreviado para grid row-gap (espaciado de fila) y grid-column-gap (espaciado de columna)	0
grid-auto-flow	*row | column | row dense | column dense*	Los ítems que no están explícitamente colocados en el contenedor grid, se disponen automáticamente en filas (rows). Para reordenar los ítems del contenedor grid utilizamos la propiedad grid-auto flow.	row
grid-auto-columns **grid-auto-rows**	*grid-auto-columns:100 px;* *grid-auto-rows:100px;*	Establece el tamaño de los ítems situados fuera del grid explícito.	auto

justify-items

justify-items: start | end | center | stretch;

align-items

Alinea horizontalmente (justify items) y verticalmente (align items) los ítems dentro de las

stretch

		celdas o de las áreas del contenedor grid.
3		
justify-content align-content *start* | *end* | *center* | *stretch* |	*space-around* | *space between* | *space-evenly;* Alinea las columnas y las filas del	grid relativo al contenedor grid. start

5.2 Las propiedades grid-template-columns y grid-template-rows.

En el ejemplo anterior hemos visto la propiedad grid-template-columns en acción. También hay otra propiedad: grid-template-rows (*rows == filas*) utilizada para crear la estructura de filas del contenedor grid.

```
#cuadricula {
 display: grid;
 grid-template-columns:1fr 1fr 1fr;
 grid-template-rows:6em max-content;
 grid-gap:1em;
 }
```

En este caso la propiedad grid-template-rows crea una primera fila (*row*) que tiene una altura de 6em, y una segunda fila cuya altura depende del contenido de los ítems de esta (max-content):

5.3 La propiedad grid-template-areas

(una propiedad del contenedor grid)

4

Un área del contenedor grid es un espacio definido entre cuatro líneas: dos líneas verticales y dos horizontales. Dentro de esta área pueden haber, o no, uno o más ítems grid. Para poder referenciar estas áreas necesitamos definirlas utilizando la propiedad grid template-areas.

```
.grid {
 /* transforma el elemento en un contenedor grid*/
 display: grid;
 /*define las columnas y las filas del contenedor grid*/
 grid-template-columns:1fr 10em 10em 1fr;
 grid-template-rows:5em 10em 5em;
 /*define las áreas del grid*/
 grid-template-areas:"h h h h"
 "l . c r"
 "l f f f";
 }
```

Miremos con atención la última regla CSS: el valor de grid-template-areas es una sucesión de tres cadenas de texto, una para cada fila definida con grid-template-rows. Dentro de cada cadena de texto aparecen exactamente 4 valores: uno para cada columna definida con grid template-columns. Los nombres utilizados en este caso son h, l, ., c, r y f, pero podemos utilizar los nombres que deseamos.

Podemos utilizar estos nombres para definir la posición y la extensión de los ítems grid utilizando la propiedad grid-area:

```
.h{grid-area: h;}
.l{grid-area: l;}
.c{grid-area: c;}
```

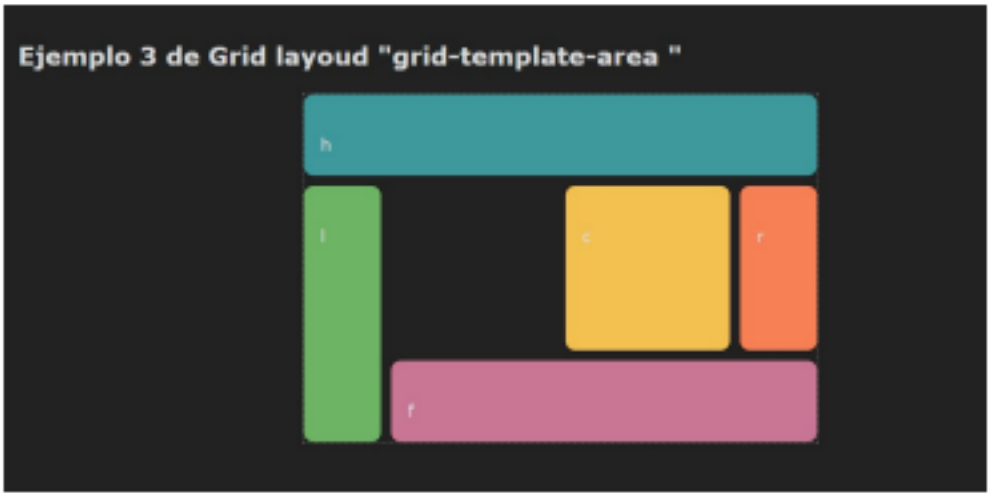
```
.r{grid-area: r;}
.f{grid-area: f;}
```

Si no utilizamos una de las áreas definidas, esta se queda vacía. En este caso el nombre del área vacía es un punto.

```
.grid {
 display: grid;
 grid-template-columns:1fr 10em 10em 1fr;
 grid-template-rows:5em 10em 5em;
 /*********************************/

5
 grid-template-areas:"h h h h"
 "l . c r"
 "l f f f";
 /*********************************/
 grid-gap:.5em;

 margin: 1em auto;
 border: 1px dashed #777;
 width:31em;
}
```



5.4 La propiedad grid-template

La propiedad grid-template es el método abreviado (*shorthand*) para declarar las

propiedades grid-template-rows, grid-template-columns y grid-template-areas en una sola línea de código. La especificación nos dice que su valor puede ser:

```
none |
 [ <'grid-template-rows'> / <'grid-template-columns'> ] | [
<line-names>? <string> <track-size>? <line-names>? ]+ [ / <explicit
track-list> ]?
```

Veamos qué quiere decir.

none

El valor de grid-template puede ser none. En este caso no hay filas ni columnas ni líneas. Tampoco hay áreas del grid definidas.

```
grid-template: none;
```

[<'grid-template-rows'> / <'grid-template-columns'>]

Esto define primero las filas (*rows*) y después las columnas (*columns*) en una sola declaración.
Ejemplos:

```
grid-template: 1fr 2fr / 1fr 1fr 1fr;
```

es equivalente de:

```
grid-template-rows: 1fr 2fr;
grid-template-columns: 1fr 1fr 1fr;
```

Veamos otro ejemplo. Esta vez también definimos las líneas del grid. (Como ya se sabe el nombre de las líneas que definimos aparece entre corchetes.)

```
grid-template: [r1] 1fr [r2] 2fr [r3]
 / [c1] 1fr [c2] 1fr [c3] 1fr [c4];
```

es equivalente de:

```
grid-template-rows:[r1] 1fr [r2] 2fr [r3];
grid-template-columns: [c1] 1fr [c2] 1fr [c3] 1fr [c4];
```

5.5 La propiedad grid-gap

(una propiedad del contenedor grid)

La propiedad grid-gap define el espacio que tiene que haber entre las columnas y las filas del contenedor grid, y es un método abreviado para grid-row-gap (espaciado de fila) y grid-column gap (espaciado de columna).

Por ejemplo esta regla;

.grid{grid-gap:1em;}
Establece que la distancia entre las celdas del contenedor grid es de 1em. O sea: la distancia entre las columnas es igual a la distancia entre las filas del contenedor.

Esta otra regla:

```
.grid{grid-gap:1em .5em;}
```

Establece que la distancia entre las filas del grid es de 1em mientras que la distancia entre las columnas es de .5em.
Los locuaces pueden escribir lo mismo utilizando dos reglas CSS:

```
.grid{
 grid-row-gap:1em;
 grid-column-gap: .5em;
 }
```

7
Enlace

5.6 La propiedad grid-auto-flow

(una propiedad del contenedor grid)

Podemos reordenar los ítems del contenedor grid utilizando la propiedad grid-auto-flow.

La propiedad grid-auto-flow puede tomar una de estas valores:

```
.contenedor{
```


```
grid-auto-flow: row | column | row dense | column dense
```

}

El valor por defecto de grid-auto-flow es **row**, lo que quiere decir que los elementos se disponen automáticamente en filas.

Si grid-auto-flow: column los elementos se disponen en columnas.

Enlace

Para entender lo que puede hacer la palabra clave dense, necesitamos hacer unos pequeños cambios al ejemplo anterior:

```
.item:nth-child(2n +1){grid-row-end:span 2;}
 .item:nth-child(2n){grid-column-end:span 2;}
```

8

Esto hace que algunos ítems sean más anchos o más altos que los demás, y por lo tanto ya no encajan perfectamente y aparecen huecos en la estructura del grid. Y aquí la cosa se pone interesante.

Si añadimos la palabra clave dense:

```
.grid{
 grid-auto-flow: row dense;
}
```

el algoritmo auto-flow intenta calcular de nuevo la posición de cada ítem por tal de hacerlos encajar.

Enlace

5.7 Justificar y alinear dentro del contenedor grid

Cuando hablamos de grid, **justificar** (*justify*) actúa horizontalmente mientras que **alinear** (*align*) actúa verticalmente .

Las propiedades justify-items & align-items

(dos propiedades del contenedor grid)

Utilizamos las propiedades justify-items y align-items para alinear horizontalmente (justify items) y verticalmente (align-items) los ítems dentro de las celdas o de las áreas del contenedor grid.
Estos atributos pueden tomar uno de estos valores:

```
.contenedor{
 justify-items: start | end | center | stretch;
 align-items: start | end | center | stretch;
}
```

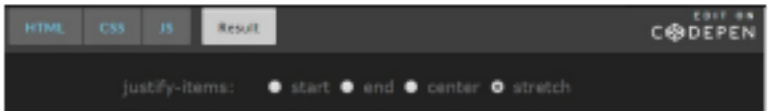
En idiomas como el castellano, con un sistema de escritura de izquierda a derecha direction: ltr; (*left to right*),

justify-items: start alinea los ítems a la izquierda,
justify-items: end alinea los ítems a la derecha.

La otra propiedad
align-items: start alinea los ítems a la parte superior de la celda o área
align-items: end alinea los ítems a la parte inferior de la celda o área.

Si utilizamos estas palabras clave:
center: el ítem aparece en el centro (*center*) de la celda o área, stretch: el ítem aparece estirado (*stretched*) para ocupar toda la celda o área.

Alineación de los ítems dentro de



las celdas del grid:

Enlace

Alineación de los ítems dentro de las áreas del grid:

Enlace

10

5.8 Las propiedades justify-content y align-content

(dos propiedades del contenedor grid)

Hay situaciones en las cuales el contenedor es más grande que el grid. Esto puede pasar si dimensionamos el grid relativo a la ventana, pero las columnas y las filas tienen dimensiones fijas (en px o en em). En estos casos podemos alinear las columnas y las filas del grid relativo al contenedor utilizando las propiedades justify-content y align-content.
Los valores que pueden tomar estas dos propiedades son:

```
.contenedor{
 justify-content: start | end | center | stretch | space-around |
space-between | space-evenly;
```

```
 align-content: start | end | center | stretch | space-around | space
between | space-evenly;
 }
```

start (el valor por defecto): los elementos aparecen agrupados al principio (*start*) del eje horizontal (para justify-content) o vertical (para align-content).

end: los elementos aparecen agrupados al final (*end*) del eje horizontal (para justify-content) o vertical (align-content).

center: los elementos aparecen agrupados al centro (*center*).

stretch: los elementos aparecen estirados (*stretched*) para ocupar el espacio restante.

space-around: los elementos aparecen distribuidos uniformemente, y con un espacio igual entre ellos. (medio espacio entre los bordes del contenedor y los ítems).

space-between: los elementos aparecen distribuidos uniformemente: al principio, en el centro y al final del eje (sin espacio entre los bordes del contenedor y los ítems).

space-evenly: los elementos aparecen distribuidos uniformemente, y con un espacio igual entre ellos y entre ellos y los bordes del contenedor.

Si esto no parece muy claro, pruebe esta demostración:

Enlace

5.9 Propiedades de los ítems grid

Propiedad	Posibles valores	Descripción
grid-column-start grid-column-end grid-row-start grid-row-end:	*el nombre o el número de la línea*	Definen la posición y la extensión de un elemento (ítem) dentro del contenedor grid.
grid-column	*grid-row: grid-row-start/grid-row-end;*	Declaraciones abreviadas para grid-row-start/grid-row-end, y grid-column-start/grid-column

grid-row	*grid-column: grid-column start/grid-column-end;*	end respectivamente.
grid-area	*grid-area: grid-row-start / grid-column-start /* *grid-row-end / grid-column end;*	Otra manera abreviada de definir la posición y la extensión de un ítem dentro del contenedor grid

justify-self align-self	*start* | *end* | *center* | *stretch;*	se utilizan para alinear los elementos relativo al eje de la fila (justify-self) o de la columna (align-self).
order	*número integro*	Define el orden para los ítems del grid

5.10 Las propiedades grid-column y grid-row

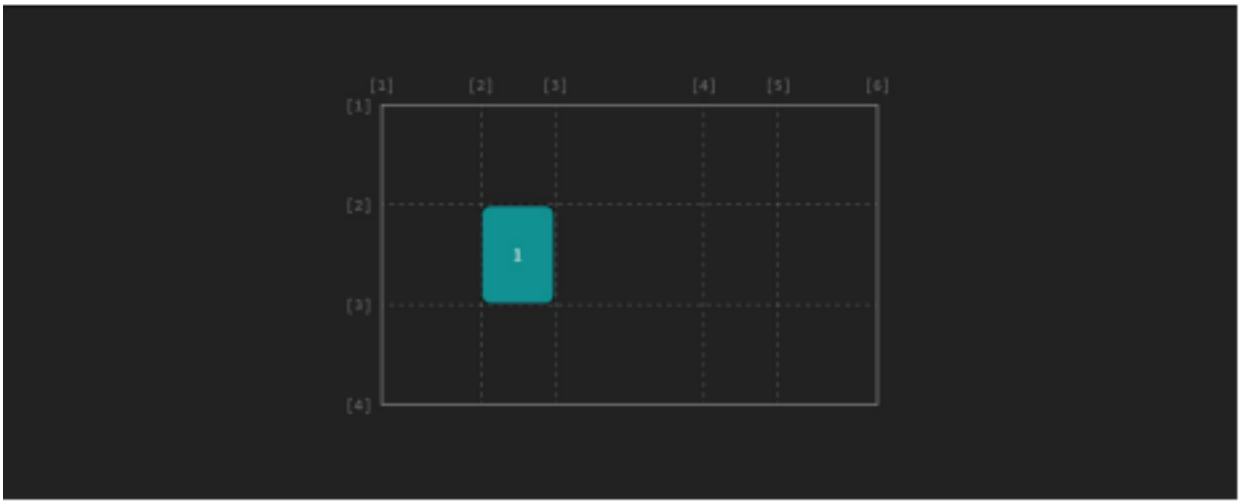
(dos propiedades de los ítems grid)

A veces necesitamos definir la posición y la extensión de un elemento (ítem) dentro del contenedor grid. Por ejemplo, para decir que el elemento se sitúa entre la segunda y la tercera línea vertical y entre la segunda y la tercera línea horizontal tenemos que escribir estas reglas:

```
.item{
        grid-column-start: 2;
        grid-column-end: 3;
        grid-row-start: 2;
        grid-row-end: 3;
}
```

Podemos escribir lo mismo utilizando solo dos líneas de código:


```
.item{
        grid-column: 2 / 3;
        grid-row: 2 / 3;
}
```



En estos ejemplos los números utilizados son los nombres de las líneas verticales y horizontales definidas con grid-template-columns y/o grid-template-rows.

Ejemplo:

También podemos utilizar la palabra clave span (*se extiende*). Por ejemplo, span 1 quiere decir que el ítem se extiende exactamente una celda, fuera el que fuera el tamaño de esta.

```
.item{
        grid-column: 2 / span 1;
grid-row: 2 / span 1;
}
```

Todavía más: podemos escribirlo todo utilizando una sola línea de código. Pero para esto necesitamos utilizar otra propiedad: grid-area.

La propiedad grid-area

(una propiedad de los ítems grid)

La propiedad grid-area es una manera abreviada para definir la posición y la extensión de un ítem dentro del contenedor grid. La sintaxis es la siguiente:

14

```
.item{
        grid-area: <grid-row-start> / <grid-column-start> / < grid-row-end> /
< grid-column-end>
}
```

O sea: podemos tomar esto:

```
.item{
        grid-row-start: 2;
        grid-column-start: 2;
        grid-row-end: 3;
        grid-column-end: span 3;
 }3333
```

Y transformarlo en esto:

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

Ejemplo:

5.11 Las propiedades justify-self y align-self

15
(dos propiedades de los ítems grid)

Tanto justify-self como align-self son propiedades de los ítems (NO del contenedor) grid, y se utilizan para alinear los elementos relativos al eje de la fila (justify-self) o de la columna (align-self).

Los valores que pueden tomar estas dos propiedades son:

align-self: start | end | center | stretch;

justify-self: start: alinea el contenido del elemento a la izquierda.

align-self: start: alinea el contenido del elemento a la parte superior.

justify-self: end: alinea el contenido del elemento a la derecha.

align-self: end: alinea el contenido del elemento a la parte inferior.

justify-self: center

align-self: center: alinea el contenido del elemento en el centro.

justify-self: stretch

align-self: stretch: el contenido aparece estirado (*stretched*) para ocupar todo el espacio.

Ejemplo:

16

La propiedad order

(una propiedad de los ítems grid)

Los ítems de un contenedor grid, como todos los elementos HTML aparecen en el mismo orden que en el código. Podemos alterar este orden utilizando la propiedad order.

Por defecto los elementos no tienen definido un order (orden). Si definimos el atributo order ="1" para un ítem cualquiera, este aparecerá en la última posición dentro de la caja, ya que 1 es más grande que nada.

En el siguiente ejemplo primero definimos el orden para cada elemento del grid: el primer ítem: order:1; el segundo ítem: order:2 . . . etc.

```
var items = document.querySelectorAll("#cuadricula .item");
 for(var i = 0; i < items.length; i++){
 items[i].style.order = i+1;
        }
```

Cada vez que el input type="number" cambia, el valor del atributo order del primer ítem cambia también, y por lo tanto cambia también la posición de este.

17

```
orderInput.addEventListener("input", function(){
 items[0].style.order = orderInput.value;
 },false);
```

Ejemplo:

5.12 Algunas palabras clave

Una fracción de grid

Aunque podemos utilizar cualquier otras unidades para longitud disponibles en CSS (px, %, em, rem . . .etc), es importante saber que el grid tiene una unidad específica: fr que representa una fracción del espacio disponible dentro del contenedor grid.

Veamos un caso muy sencillo. Si queremos crear una estructura de 3 columnas , podemos escribir:

grid-template-columns:1fr 1fr 1fr;

En este caso cada una de las tres columnas tiene una anchura de

33.33%. Si queremos que la columna central sea más ancha podemos

escribir:

```
grid-template-columns:1fr 2fr 1fr;
```


Esta vez dividimos el espacio disponible por 4 (1fr + 2fr + 1fr == 4fr), y en este caso la primera y la tercera columna tendrán una anchura de 25%, mientras que la columna central tendrá una anchura de 50%.

No es absolutamente necesario utilizar números enteros. También podemos usar números decimales. Por ejemplo, esta es una declaración perfectamente válida, i el resultado es idéntico al de antes:

```
grid-template-columns:0.5fr 1fr 0.5fr;
```

Tampoco es necesario utilizar exclusivamente fracciones de grid (fr). Podemos utilizar cualquier combinación de unidades de longitud, según sea necesario.

En este caso:

```
grid-template-columns:1fr 100px 10em;
```

la primera columna ocupa todo el espacio disponible dentro del contenedor grid, y es equivalente a:

```
grid-template-columns:calc(100% - 100px - 10em) 100px 10em;
```

En este caso, también podemos utilizar la palabra clave auto con el mismo efecto:

```
grid-template-columns:auto 100px 10em;
```

El método repeat()

Para no repetirnos tanto (grid-template-rows:1fr 1fr 1fr;) podemos utilizar el método repeat():

```
grid-template-columns: repeat(3, 1fr);
```

donde el primer argumento (3) especifica cuantas veces tiene que repetirse la sucesión especificada por el segundo argumento (1fr).

19

```
#cuadricula {
        display: grid;99 /*grid-template-columns:1fr 1fr 1fr;*/
        grid-template-columns: repeat(3, 1fr);
 grid-gap:1em;
}
```

También podemos repetir una sucesión de columnas:

```
grid-template-columns: repeat(3, 1fr .5fr);
```

y esto es equivalente a:

```
grid-template-columns: 1fr .5fr 1fr .5fr 1fr .5fr;
```

O podemos utilizar una combinación, por ejemplo, algo así:

```
grid-template-columns: repeat(3, 1fr .5fr) 1fr;
```

que es equivalente a:

```
grid-template-columns: 1fr .5fr 1fr .5fr 1fr .5fr 1fr;
```

Las palabras clave max-content y min-content

Si utilizamos max-content para definir una columna o una fila, esto quiere decir que esta tendrá el *tamaño* (anchura para columnas o altura para filas) mínimo necesario para encajar el contenido máximo.

Si utilizamos min-content, esto quiere decir que el carril (fila o columna) tendrá el tamaño mínimo necesario para que no haya overflow.

Ejemplo:

20

En el siguiente ejemplo la propiedad grid-template-rows crea una primera fila (*row*) que tiene una altura de 6em, y una segunda fila cuya altura depende del contenido de los ítems de esta (max-content).

De otra parte, el contenido del div naranja es una lista cuyos elementos <li> pueden ser editados.

```
<li contenteditable="true">
```

En el CSS los elementos li tienen white-space: nowrap; para prevenir los saltos de línea.

```
li{white-space: nowrap;}
```

Por favor edite los elementos de lista para ver min-content in acción. Empiece por el elemento de lista más largo.

Ejemplo:

Las palabras clave start y end

Dos de las palabras clave muy utilizadas: start (*inicio*) y end (*final*) pueden crear una cierta confusión. Es importante saber que en idiomas como el castellano, con un sistema de escritura de izquierda a derecha direction: ltr; (*left to right*), la palabra clave start se refiere a la izquierda para columnas y arriba para filas. De la misma manera la palabra clave end se refiere a la derecha del contenedor grid para columnas y abajo para filas.

Ejemplo:

22

Líneas y carriles

Cuando definimos las columnas y las filas (*tracks* o carriles) de un contenedor grid, automáticamente definimos las líneas (*lines*) de este.

En idiomas como el castellano, con un sistema de escritura de izquierda a derecha (direction: ltr;) , la primera línea vertical coincide con el borde izquierdo del contenedor grid, y la podemos referenciar utilizando el número 1. De la misma manera la primera línea horizontal

coincide con el borde superior del contenedor grid, y de nuevo la podemos referenciar utilizando el número 1.

Si no nos gusta utilizar números, o, si por alguna razón consideramos que pueden crear confusión, podemos dar nombres a las líneas utilizando esta sintaxis:

grid-template-columns: [col-1-a] 1fr [col-1-z col-2-a] 1fr [col-2-z col-3-a] 1fr [col 3-z];

En este ejemplo la primera línea se llama col-1-a, pero la podemos referenciar también utilizando el número 1. La segunda línea tiene dos nombres: col-1-z y col-2-a que aparecen en el código separados por un espacio en blanco ([col-1-z col-2-a]). Exactamente como antes, también la podemos referenciar por el número de orden, 2 en este caso.

¿Y para qué necesitamos todos estos nombres? Los necesitamos al definir la posición y extensión de los ítems grid, con las propiedades grid-area, grid-row-start, grid-column-start, grid row-end y/o grid-column-end.

Ejemplos

Estos son algunos ejemplos creados con Grid-Layout:

https://silocreativo.com/labs/escape-from-a-christmas-tale/es/

https://labs.jensimmons.com/2017/01-011.html

https://codepen.io/julesforrest/full/QaBvPe