UD3.2 CSS CON GRID

ÍNDEX

1. CONCEPTOS BÁSICOS DE DISPOSICIÓN GRID

El diseño de cuadrícula CSS (GRID) introduce un sistema de cuadrícula bidimensional en CSS. Las cuadrículas pueden utilizarse para distribuir las principales áreas de la página o chicos elementos de la interfaz de usuario. Estos apuntes presentan el CSS Grid Layout.

1.1.¿Qué es una cuadrícula?

Una cuadrícula es un conjunto de líneas horizontales y verticales que se cruzan y definen columnas y filas. Los elementos pueden colocarse en la cuadricula dentro de estas líneas de columnas y filas. El diseño de cuadricula CSS tiene las siguientes características:

Carriles fijos y flexibles

Puede crear una cuadricula con tamaños de pista fijos, utilizando píxeles, por ejemplo. Esto establece la cuadricula en el píxel especificado que se ajusta al diseño que desea. También puede crear una cuadricula utilizando tamaños flexibles con porcentajes o con la unidad `fr` diseñada para este fin.

Colocación de elementos

Puede colocar elementos en un lugar preciso de la cuadrícula utilizando números de línea, nombres o apuntando a una zona de la cuadrícula. Grid también contiene un algoritmo para controlar la colocación de elementos que no tienen una posición explícita en la cuadrícula.

Creación de pistas adicionales para albergar contenidos

Puede definir una cuadricula explícita con el diseño de cuadricula. La especificación Grid Layout es lo suficientemente flexible como para añadir filas y columnas adicionales cuando sea necesario. Se incluyen funciones como añadir "tantas columnas como quepan en un contenedor".

Control de alineación

La cuadrícula contiene funciones de alineación para que podamos controlar cómo se alinean los elementos una vez colocados en un área de la cuadrícula, y cómo se alinea toda la cuadrícula.

Control del solapamiento de contenidos

Se puede colocar más de un elemento en una celda o área de la cuadrícula y pueden superponerse parcialmente entre sí. Esta superposición puede controlarse con la propiedad z-index.

Grid es una potente especificación que, cuando se combina con otras partes de CSS como flexbox (cercano documento de teoría), puede ayudarte a crear diseños que antes eran imposibles de construir en CSS. Todo empieza con la creación de una cuadrícula en su **contenedor de cuadrícula**.

2. CONTENEDOR DE CUADRICULA

Creamos un *contenedor de cuadricula* declarando `display: grid` o `display: inline-gridon` un elemento. Cuando lo hagamos, todos los *hijos directos* de este elemento se convertirán en *elementos de cuadricula*.

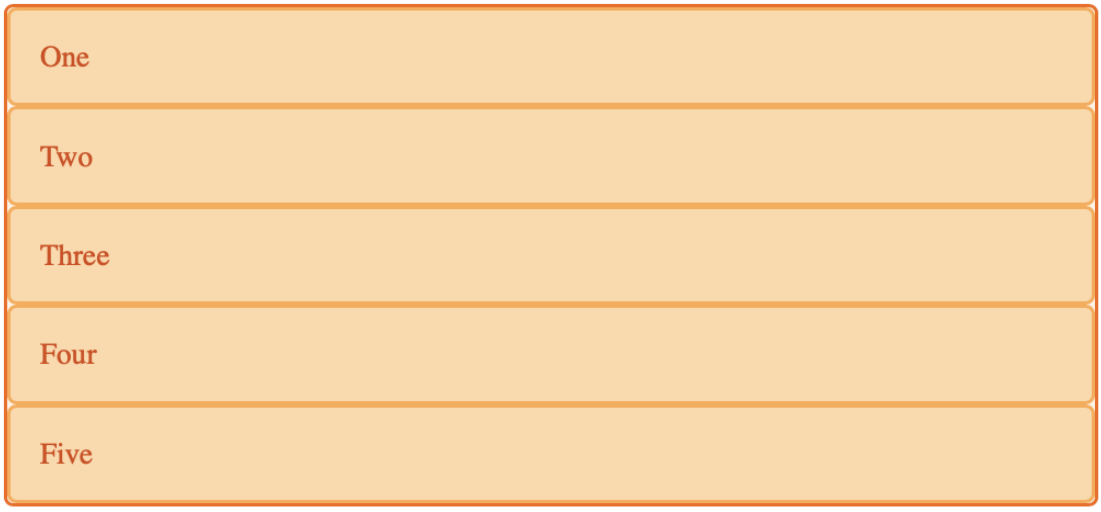
En este ejemplo, tengo un div contenedor con una clase 'wrapper' y, dentro hay cinco elementos hijo.

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```

Hago del .wrapper un contenedor de cuadricula.

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

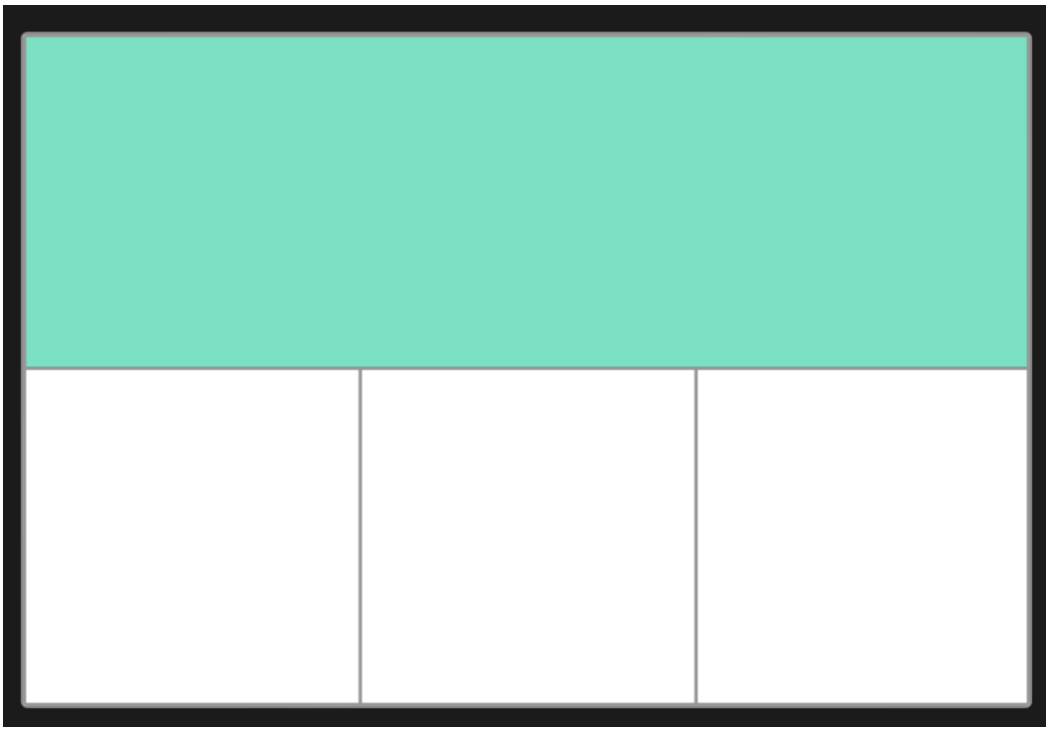
Todos los hijos directos son ahora elementos de la cuadricula. En un navegador web, no verás ninguna diferencia en cómo se muestran estos elementos antes de convertirlos en una cuadricula, ya que grid ha creado una cuadricula de una sola columna para los elementos.



Si queremos que esto se parezca más a una cuadrícula, debemos añadir pistas de columna.

3. SENDEROS DE QUADRICULA

Definimos filas y columnas en nuestra cuadricula con las propiedades `grid-template-rows` y `grid-template-columnos`. Estas definen las líneas de la cuadrícula. Un trazado de *cuadrícula* es el espacio entre dos líneas adyacentes de la cuadrícula. La imagen de abajo muestra una pista resaltada - esta es la pista de la primera fila de nuestra cuadricula.



Los trazados de la cuadricula se definen en la cuadricula explícita utilizando las propiedades `grid-template-columns y grid-template-rows.`

3.1. Ejemplo básico

Puedo añadir a nuestro ejemplo anterior mediante la adición de la propiedad `grid-template-columns`, a continuación, definir el tamaño de las pistas de columna.

Ahora he creado una cuadricula con tres pistas de columna de 200 píxeles de ancho. Los elementos hijo se colocarán en esta cuadricula uno en cada celda de la cuadricula.

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```

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

3.2.La unidad fr

Los trazados pueden definirse utilizando cualquier unidad de longitud. Grid también introduce una unidad de longitud adicional para ayudarnos a crear trazados de cuadrícula flexibles. La nueva unidad `fr` representa una fracción del espacio disponible en el contenedor de la cuadricula. La siguiente definición de cuadricula crearía tres tracks de igual anchura que crecen y decrecen según el espacio disponible.

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```

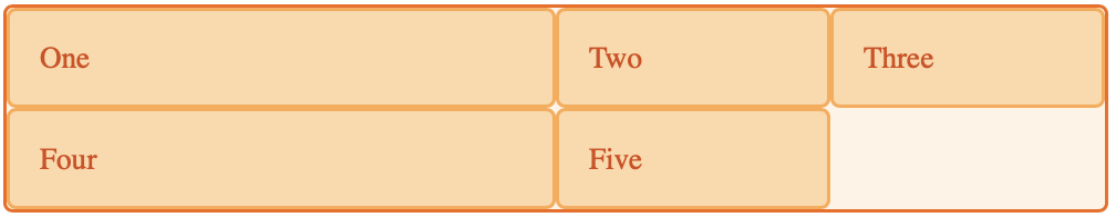
```
.wrapper {
 display: grid;
 grid-template-columns: 1fr 1fr 1fr;
}
```

3.3. Tamaños desiguales

En el siguiente ejemplo, creamos una definición con una pista de 2fr y luego dos pistas de 1fr. El espacio disponible se divide en cuatro. Se asignan dos partes a la primera pista y una parte a cada una de las dos pistas siguientes.

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```

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



3.4. Mezcla de tamaños flexibles y absolutos

En este último ejemplo, mezclamos pistas de tamaño absoluto con unidades `fr`. La primera pista es de 500 píxeles, por lo que la anchura fija el resto del espacio disponible. El espacio restante se divide en tres y se asigna en proporción a las dos pistas flexibles.

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```

```
.wrapper {
 display: grid;
 grid-template-columns: 500px 1fr 2fr;
}
```

3.5. Listados de pistas con notación repeat()

Las rejillas grandes con muchas pistas pueden utilizar la notación `repeat()`, para repetir todo o una sección del listado de pistas. Por ejemplo, la definición de la cuadricula:

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

También se puede escribir como:

```
.wrapper {
 display: grid;
 grid-template-columns: repeat(3, 1fr);
}
```

La notación de repetición se puede utilizar para una parte de la lista de pistas. En el siguiente ejemplo he creado una cuadricula con una pista inicial de 20 píxeles, luego una sección de repetición de 6 pistas de `1fr` y luego una pista final de 20 píxeles.

```
.wrapper {
 display: grid;
 grid-template-columns: 20px repeat(6, 1fr) 20px;
}
```

La notación de repetición toma el listado de pistas y lo utiliza para crear un patrón repetitivo de pistas. En el siguiente ejemplo, mi parrilla constará de 10 pistas, una pista de `1fr` y, a continuación, una pista `de 2fr`. Este patrón se repetirá cinco veces.

```
.wrapper {
 display: grid;
 grid-template-columns: repeat(5, 1fr 2fr);
}
```

3.6. Rejillas implícitas y explícitas

Al crear nuestra cuadricula de ejemplo definimos específicamente nuestras pistas de columnas con la propiedad `grid-template-columnos`, pero la cuadricula también creó filas por sí misma. Estas filas forman parte de la cuadricula implícita. Mientras que la cuadricula explícita consiste en cualquier fila y columna definida con `grid-template-columns` o `grid-template-rows`.

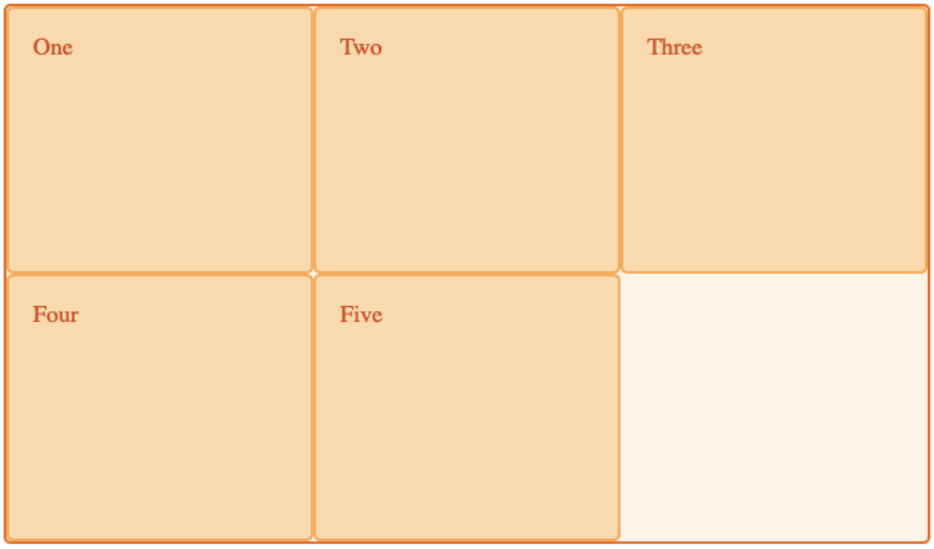
Si coloca algo fuera de la cuadricula definida -o debido a la cantidad de contenido, se necesitan más pistas de cuadricula- entonces la cuadricula crea filas y columnas en la cuadricula implícita. Por defecto, el tamaño de estas filas se ajusta automáticamente en función del contenido que contienen.

También puede definir un tamaño fijo para las pistas creadas en la cuadricula implícita con las propiedades `grid-auto-rows` y `grid-auto-columns`.

En el siguiente ejemplo, utilizamos `grid-auto-rows` para garantizar que las pistas creadas en la cuadricula implícita tengan 200 píxeles de altura.

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```

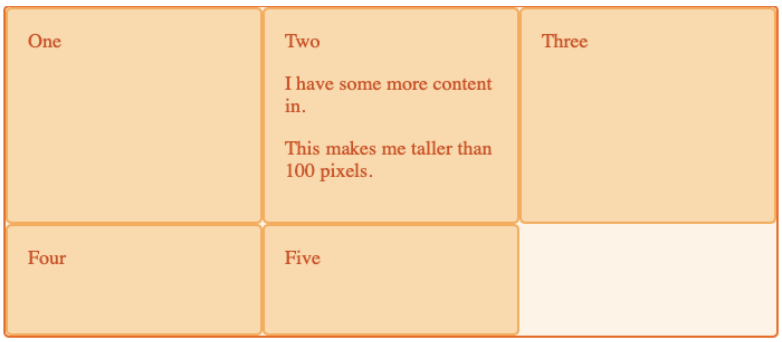
```
.wrapper {
 display: grid;
 grid-template-columns: repeat(3, 1fr);
 grid-auto-rows:  200px;
}
```



3.7. Procedimentaldo de vías y minmax

Al configurar una cuadricula explícita o definir el tamaño de las filas o columnas creadas automáticamente, puede ser que queramos dar a las filas un tamaño mínimo, pero también asegurarnos de que se expanden para ajustarse a cualquier contenido que se afianza. Por ejemplo, puede ser que queramos que mis filas nunca se contraigan por debajo de 100 píxeles, pero si mi contenido se estira hasta 300 píxeles de altura, entonces me gustaría que la fila se estire hasta esa altura.

Grid tiene una solución para ello con la función minmax`()`. En el siguiente ejemplo estoy usando minmax() en el valor de `grid-auto-rows`. Esto significa que las filas creadas automáticamente tendrán un mínimo de 100 píxeles de alto, y un máximo de auto . Usando `auto` significa que el tamaño se verá en el tamaño del contenido y se estirará para dar espacio para el elemento más alto en una celda, en esta fila.

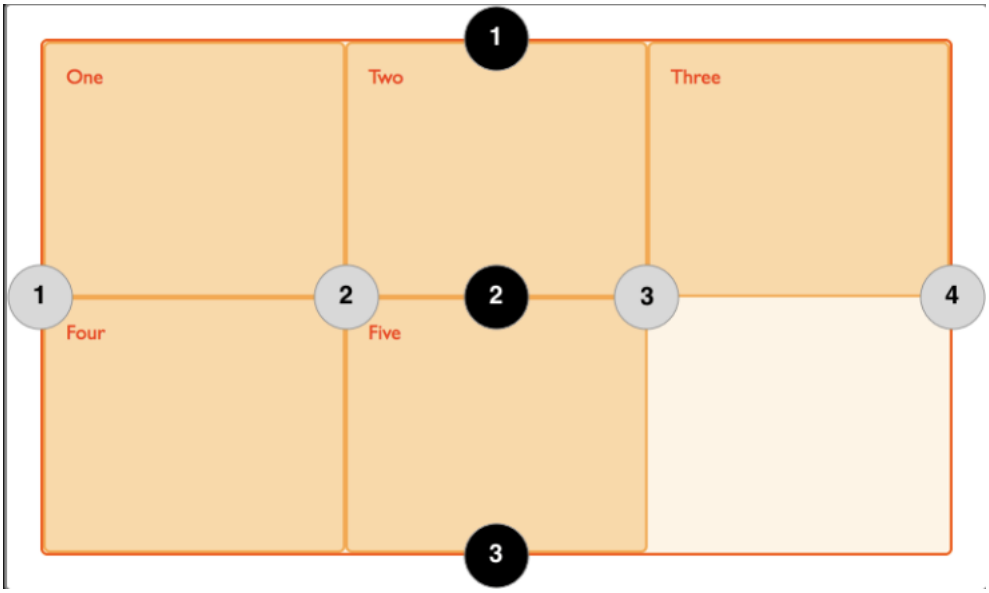



```
.wrapper {
display: grid;
grid-template-columns: repeat(3, 1fr);
grid-auto-rows: minmax(100px, acte);
}
```

```
<div class="wrapper">
<div>Un</div>
<div>
Dos
<p>Tinc una mica més de contingut en.</p>
<p>Això em fa més alt de 100 píxels.</p>
</div>
<div>Tres</div>
<div>Quatre</div>
<div>Cinc</div>
</div>
```

4. LÍNEAS DE CUADRÍCULA

Hay que tener en cuenta que cuando definimos una cuadricula definimos las pistas de la cuadricula, no las líneas. La cuadricula entonces nos da líneas numeradas para usar cuando posicionamos elementos. En nuestra cuadrícula de tres columnas y dos filas tenemos cuatro líneas de columna.



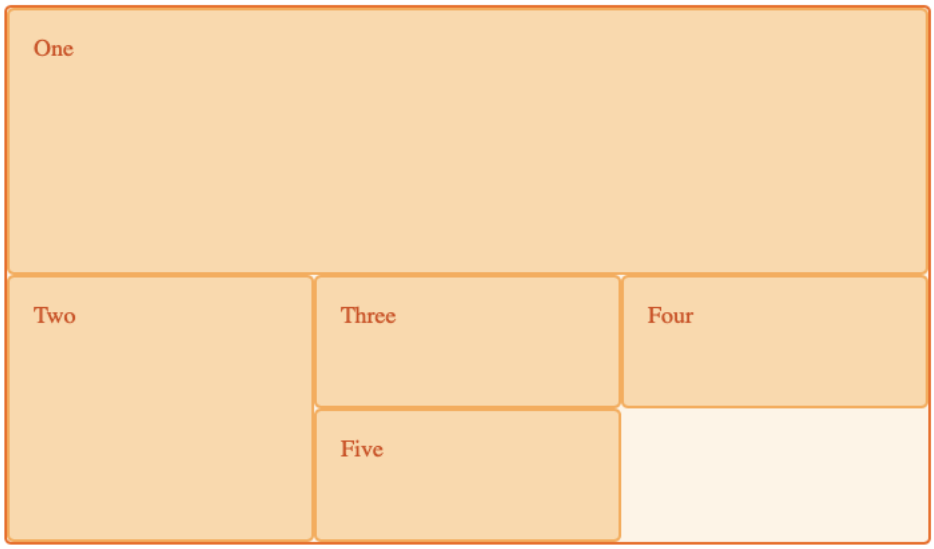
Las líneas se numeran según el modo de escritura del documento. En una lengua de izquierda a derecha, la línea 1 está en el lado izquierdo de la cuadrícula. En un idioma de derecha a izquierda, está en el lado derecho de la cuadrícula. Las líneas también pueden nombrarse, y veremos cómo hacerlo en una guía posterior de esta serie.

4.1. Posicionamiento de artículos contra líneas

Exploraremos la colocación basada en líneas con todo detalle en un artículo posterior. El siguiente ejemplo muestra cómo hacerlo de manera sencilla. Al colocar un elemento, nos dirigimos a la línea - en lugar de la pista.

En el siguiente ejemplo estoy colocando los dos primeros elementos en nuestra cuadricula de tres columnas, utilizando las propiedades `grid-column-start`, `grid-column-end`, `grid-row-start` y `grid-row-end`. Trabajando de izquierda a derecha, el primer elemento se coloca contra la línea de columna 1, y se extiende hasta la línea de columna 4, que en nuestro caso es la línea del extremo derecho de la cuadricula. Comienza en la línea de fila 1 y termina en la línea de fila 3, por lo que abarca dos pistas de fila.

El segundo elemento empieza en la línea 1 de la columna de la cuadrícula y abarca una pista. Este es el valor por defecto, por lo que no es necesario especificar la línea final. También abarca dos pistas de fila desde la línea 3 hasta la línea 5. Los otros elementos se colocarán en los espacios vacíos de la cuadrícula.




```
<div class="wrapper">
 <div class="box1">Un</div>
 <div class="box2">Dos</div>
 <div class="box3">Tres</div>
 <div class="box4">Quatre</div>
 <div class="box5">Cinc</div>
</div>
```

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

.box1 {
 grid-column-start: 1;
 grid-column-end: 4;
 grid-row-start: 1;
 grid-row-end: 3;
}

.box2 {
 grid-column-start: 1;
 grid-row-start: 3;
 grid-row-end: 5;
}
```

4.2. Abreviaturas de posicionamiento de líneas

Los valores a mano alzada utilizados anteriormente pueden comprimirse en una línea para las columnas con `grid-column`, y en una línea para las filas con `grid-row`. El siguiente ejemplo daría el mismo posicionamiento que en el código anterior, pero con mucho menos CSS. El valor antes de la barra oblicua (`/`) es la línea de inicio, el valor después de la línea final.

Puede omitir el valor final si el área solo abarca una pista.

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

.box1 {
 grid-column
: 1 / 4;
 grid-row: 1 / 3;
}

.box2 {
 grid-column: 1;
 grid-row: 3 / 5;
}
```

5. CEL· LAS DE LA CUADRÍCULA

Una *celda de cuadrícula es* la unidad más chiquita de una cuadrícula. Conceptualmente es como una celda de mesa. Como vimos en nuestros ejemplos anteriores, una vez que una cuadrícula se define como padre, los elementos hijos se colocarán en una celda cada uno de la cuadrícula definida. En la siguiente imagen, he resaltado la primera celda de la cuadrícula.

5.1.Zonas cuadriculadas

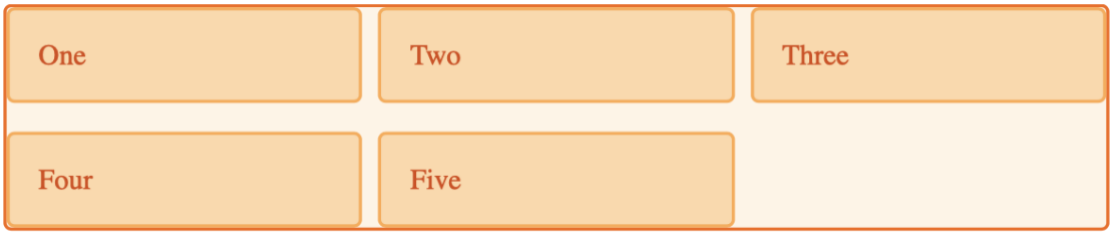
Los elementos pueden abarcar una o más celdas tanto por fila como por columna, y esto crea un *área de cuadrícula*. Las áreas de cuadrícula deben ser rectangulares; no es posible crear un área en forma de L, por ejemplo. El área de cuadrícula resaltada abarca dos filas y dos columnas.

5.2.Canalones

Se pueden crear *callejuelas* entre las celdas de la cuadrícula utilizando las propiedades `column-gap` y `row-gap`, o la abreviatura `gap`. En el siguiente ejemplo, estoy creando un espacio de 10 píxeles entre columnas y un espacio de `1em` entre filas.

```
.wrapper {
 display: grid;
 grid-template-columns: repeat(3, 1fr);
 column-gap: 10px;
 row-gap: 1em;
}
```

```
<div class="wrapper">
 <div>Un</div>
 <div>Dos</div>
 <div>Tres</div>
 <div>Quatre</div>
 <div>Cinc</div>
</div>
```



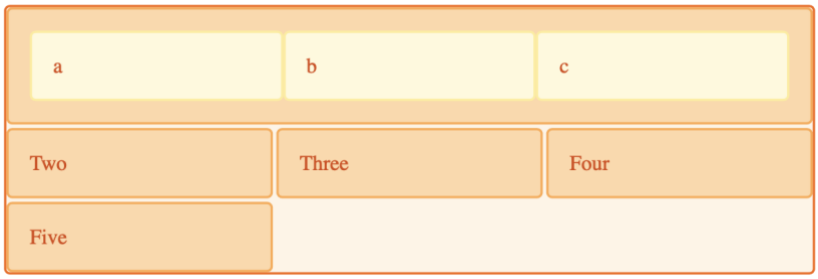
Cualquier espacio utilizado por los huecos se tendrá en cuenta antes de asignar espacio a las pistas `fr` de longitud flexible, y los huecos actúan a efectos de tamaño como una pista de cuadrícula normal, aunque no se puede colocar nada en un hueco. En términos de posicionamiento basado en líneas, el vacío actúa como una línea gruesa.

6. REJILLAS NIADAS

Un elemento de la cuadricula puede convertirse en un contenedor de la cuadricula. En el siguiente ejemplo, tengo la cuadricula de tres columnas que creo anteriormente, con nuestros dos elementos posicionados. En este caso el primer elemento tiene algunos sub-elementos. Como estos elementos no son hijos directos de la cuadricula, no participan en el diseño de la cuadricula, por lo que se muestran en un flujo de documento normal.

6.1. Anidamiento sense subrejilla

Si configure `box1` para `display:grid`, puedo darle una definición de pista y también se convertirá en una cuadrícula. Los elementos se colocarán en esta nueva cuadrícula.




```
.box1 {
 grid-column-start: 1;
 grid-column-end: 4;
 grid-row-start: 1;
 grid-row-end: 3;
 display: grid;
 grid-template-columns: repeat(3, 1fr);
}
```

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

.wrapper {
  border: 2px solid #f76707;
  border-radius: 5px;
  gap: 3px;
  background-color: #fff4e6;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

```
.box {
  border: 2px solid #ffa94d;
  border-radius: 5px;
  background-color: #ffd8a8;
  padding: 1em;
  color: #d9480f;
}

.box1 {
  grid-column: 1 / 4;
}

.nested {
  border: 2px solid #ffec99;
  border-radius: 5px;
  background-color: #fff9db;
  padding: 1em;
}
```

En este caso la cuadricula niada no tiene ninguna relación con la cuadricula padre. Como puede ver en el ejemplo, no ha heredado el `gap` de la cuadricula principal y las líneas de la cuadricula niada no se alinean con las líneas de la cuadricula principal.

6.2.Subcuadrícula

Además de las cuadrículas normales, *las subcuadrículas* nos permiten crear cuadrículas anadas que utilizan la definición de pista de la cuadrícula padre.

Para utilizarlas, editamos el ejemplo de cuadricula niada anterior para cambiar la definición de la pista `grid-template-columns: repeat(3, 1fr)`, por `grid-template-columns: subgrid`. La cuadricula niada utilizará entonces las pistas de la cuadricula padre para maquetar los elementos.

```
.box1 {
grid-column-start: 1;
grid-column-end: 4;
grid-row-start: 1;
grid-row-end: 3;
display: grid;
grid-template-columns: subgrid;
}
```

7. SUPERPOSICIÓN DE ELEMENTOS CON Z-INDEX

Los elementos de la cuadrícula pueden ocupar la misma celda, y en este caso podemos utilizar la propiedad `z-index` para controlar el orden en que se apilan los elementos superpuestos.

7.1.Superposición sin z-index

Si volvemos a nuestro ejemplo con elementos posicionados por número de línea, podemos cambiar esto para hacer que dos elementos se superponen.

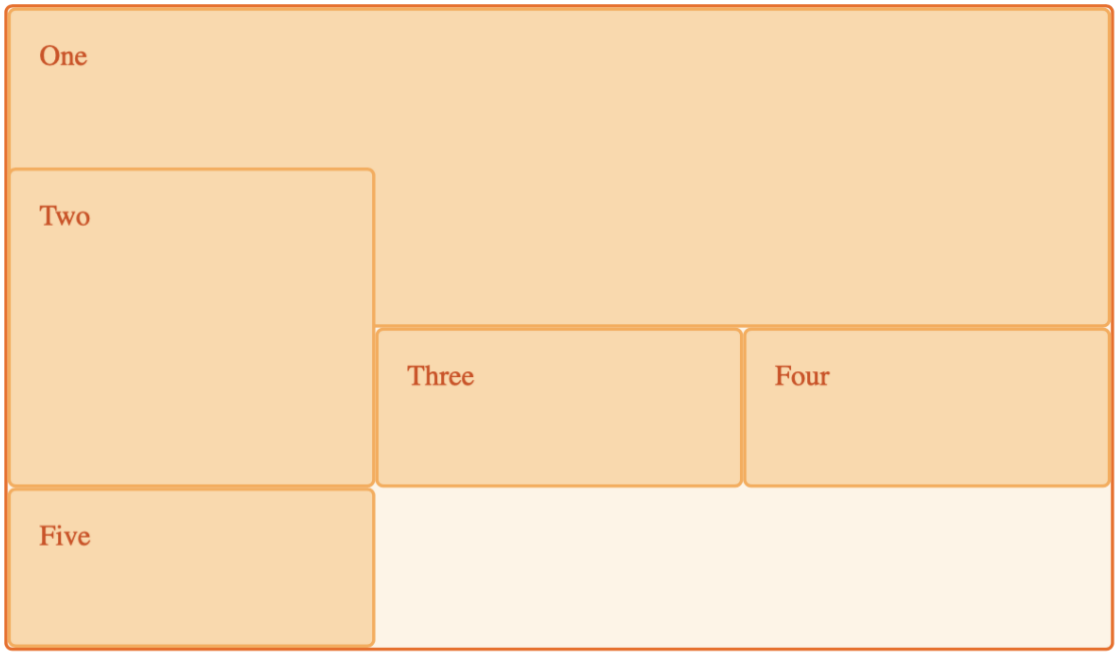
```
<div class="wrapper">
 <div class="box box1">Un</div>
 <div class="box box2">Dos</div>
 <div class="box box3">Tres</div>
 <div class="box box4">Quatre</div>
 <div class="box box5">Cinc</div>
</div>
```

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

.box1 {
 grid-column-start: 1;
 grid-column-end: 4;
 grid-row-start: 1;
 grid-row-end: 3;
}

.box2 {
 grid-column-start: 1;
 grid-row-start: 2;
 grid-row-end: 4;
}
```

El elemento `box2` está ahora superpuesto a `box1`, se muestra en la parte superior ya que viene más tarde en el orden de origen.



7.2.Controlar el orden

Podemos controlar el orden en el que los elementos se apilan utilizando la propiedad `z-index` igual que los elementos posicionados. Si damos a `box2` un `z-index` inferior al de box1, `se` mostrará debajo de `box1` en la pila.

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

.box1 {
 grid-column-start: 1;
 grid-column-end: 4;
 grid-row-start: 1;
 grid-row-end: 3;
 z-index: 2;
}

.box2 {
 grid-column-start: 1;
 grid-row-start: 2;
 grid-row-end: 4;
 z-index: 1;
}
```