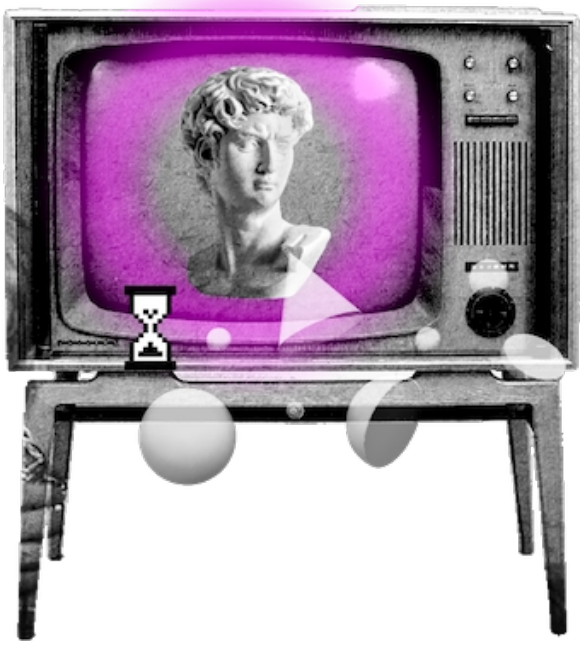
NUCBA

DISEÑO WEB

CSS GRID



APUNTES

CSS VII - Grid

¿Qué es Grid?

Grid es una **herramienta de diseño en CSS** que nos permite **crear diseños complejos y estructuras de manera fácil y rápida**. Básicamente, Grid nos permite dividir nuestro sitio web en **filas y columnas**, y luego colocar elementos dentro de esas filas y columnas. Es como si tuviéramos una tabla invisible que nos ayuda a organizar todo el contenido de nuestra página web.

Para crear un Grid en CSS, primero debemos definir un contenedor que envuelva todo el contenido que queremos organizar. Luego, con algunas propiedades CSS específicas, podemos definir el número de filas y columnas que queremos para nuestro Grid, así como su tamaño y espaciado.

Una de las ventajas más grandes de utilizar Grid es que podemos crear diseños adaptables y responsivos sin tener que utilizar media queries o código adicional. Con Grid, podemos establecer diferentes tamaños y posiciones de los elementos según el tamaño de la pantalla, lo que nos ayuda a garantizar que nuestro sitio web se vea bien en cualquier dispositivo.

Además, Grid es compatible con la mayoría de los navegadores modernos, lo que significa que podemos utilizar esta herramienta sin preocuparnos demasiado por la compatibilidad.

Estructura básica de Grid

Veamos cuales son las principales partes que componen un diseño de Grid.

Filas

Una **fila** en Grid es **una serie de celdas que se extienden horizontalmente en nuestro diseño**. Podemos definir el número de filas que queremos para nuestro Grid, así como su tamaño y espaciado. Las filas nos ayudan a organizar nuestro contenido en bloques horizontales y a definir el espacio entre ellos.

Columnas

Una **columna** en Grid es **una serie de celdas que se extienden verticalmente en nuestro diseño**. Al igual que con las filas, podemos definir el número de columnas que queremos para nuestro Grid, así como su tamaño y espaciado. Las columnas nos ayudan a organizar nuestro contenido en bloques verticales y a definir el espacio entre ellos.

Celdas

Una **celda** en Grid es **el espacio en el que colocamos nuestro contenido**. Cada celda ocupa una posición específica en el Grid, y podemos definir su tamaño y su ubicación en función de las filas y columnas que hemos definido previamente. Las celdas nos permiten colocar nuestro contenido en la posición exacta que queremos dentro de nuestro diseño.

Flexbox vs Grid

Tanto **Grid** como **Flexbox** son herramientas de diseño en CSS que nos permiten organizar nuestro contenido en diseños flexibles y adaptables. Sin embargo, hay algunas diferencias clave entre ambas herramientas.

Flexbox es una herramienta que nos permite crear diseños flexibles y adaptables en una dimensión (horizontal o vertical). Con Flexbox, podemos alinear y distribuir elementos en una fila o en una columna, pero no podemos crear diseños complejos que involucren múltiples filas y columnas.

Por otro lado, **Grid** es una herramienta que nos permite crear diseños complejos y estructuras de manera fácil y rápida en dos dimensiones (horizontal y vertical). Con **Grid**, podemos dividir nuestro sitio web en filas y columnas, y luego colocar elementos dentro de esas filas y columnas. Esto nos permite crear diseños más complejos y estructurados que los que podemos lograr con **Flexbox**.

Otra diferencia clave entre **Grid** y **Flexbox** es la forma en que se definen los elementos en cada herramienta. En **Flexbox**, los elementos se definen en una sola dimensión y se distribuyen en esa dimensión. En cambio, en **Grid**, los elementos se definen en dos dimensiones y se colocan en una celda específica dentro del **Grid**.

No obstante, ambas herramientas son muy utilizadas e incluso pueden combinarse para sacar provecho a lo mejor de ambas herramientas y crear diseños modernos y adaptables de una manera óptima.

Propiedades de CSS GRID para elementos padre

A continuación, vamos a ver las principales propiedades de Grid para **elementos padre.**

Display Grid

“display: grid;” es una propiedad en CSS que se utiliza para convertir un contenedor en una **cuadrícula.** Sin esta propiedad, no podríamos utilizar las demás que nos permiten dar forma a nuestra grilla, por lo tanto, es una propiedad obligatoria para poder trabajar con **Grid.**

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

Grid-template-columns

La propiedad **"grid-template-columns"** es una propiedad en CSS que nos permite definir el ancho y la cantidad de columnas en una cuadrícula de diseño "grid" (cuadrícula). Con esta propiedad, podemos crear diseños en los que los elementos se posicionan en columnas, y cada columna tiene un ancho específico.

Por ejemplo, si queremos definir 3 columnas de nuestra cuadrícula, lo hacemos de la siguiente manera:

```
.contenedor{
  display:grid;
  grid-template-columns: 50px 100px 50px;

}
```

En este ejemplo, tanto la primera como la tercera columna tendrán 50px de **ancho**, mientras que la del medio tendrá 100px.

Grid-template-rows

La propiedad "grid-template-rows" es una propiedad en CSS que nos permite definir la altura y la cantidad de filas en una cuadrícula de diseño "grid". Con esta propiedad, podemos crear diseños en los que los elementos se posicionan en filas, y cada fila tiene una altura específica.

Por ejemplo, si queremos definir 3 filas de nuestra cuadrícula, lo hacemos de la siguiente manera:

```
.contenedor{
  display:grid;
  grid-template-rows: 50px 100px 50px;

}
```

En este ejemplo, la primera y tercera fila tendrán 50px de **alto**, mientras que la segunda tendrá 100px.

Grid-column-gap

La propiedad **"grid-column-gap"** es una propiedad en CSS que nos **permite establecer el espacio entre las columnas en una cuadrícula de diseño "grid"**. Con esta propiedad, podemos controlar la separación horizontal entre las columnas de una cuadrícula.

```
.contenedor{
  display:grid;
  grid-column-gap: valor;
}
```

Donde "valor" puede ser una unidad de medida, como píxeles, em, rem, entre otras.

Grid-row-gap

La propiedad **"grid-row-gap"** es una propiedad en CSS que nos **permite establecer el espacio entre las filas en una cuadrícula de diseño "grid"**. Con esta propiedad, podemos controlar la separación vertical entre las filas de una cuadrícula.

```
.contenedor{
  display:grid;
  grid-row-gap: valor;
}
```

Donde "valor" puede ser una unidad de medida, como píxeles, em, rem, entre otras.

Repeat

La función **"repeat"** en CSS Grid nos **permite definir un patrón repetitivo de valores para las propiedades "grid-template-columns" y "grid-template-rows"**. En lugar de tener que escribir manualmente cada valor individual para cada columna o fila, podemos usar "repeat" para repetir un patrón de valores.

Por ejemplo, en lugar de escribir lo siguiente:

```
.contenedor{
  display:grid;
  grid-template-columns: 100px 100px 100px 100px;
}
```

podemos utilizar **repeat** para no escribir 4 veces 100px, como veremos a continuación:

```
.contenedor{
  display:grid;
  grid-template-columns: repeat(4, 100px);
}
```

Fr como unidad de medida

La unidad **"fr"** en CSS Grid es **una unidad de medida que se utiliza para definir el tamaño de las columnas o filas en una cuadrícula**. La unidad **"fr"** es una unidad flexible, lo que significa que el tamaño de la cuadrícula se divide en partes iguales en función de los valores **"fr"** especificados.

Por ejemplo, si queremos definir una cuadrícula con dos columnas, una con un ancho del 25% y otra con un ancho del 75%, podemos escribirlo así:

```
.contenedor{
  display:grid;
  grid-template-columns: 1fr 3fr;
}
```

Esto indica que la primera columna tendrá un tamaño igual a 1/4 del ancho total de la cuadrícula (25%), mientras que la segunda columna tendrá un tamaño igual a 3/4 del ancho total de la cuadrícula (75%).
Se puede usar en combinación con otras unidades de medida.

Grid-template-areas

La propiedad **"grid-template-areas"** en CSS Grid nos **permite definir un diseño de cuadrícula utilizando un sistema de áreas**. En lugar de definir el tamaño y la posición de cada celda de la cuadrícula de manera individual, podemos **agruparlas** en áreas que compartan una misma etiqueta, y definir el tamaño y la posición de estas áreas en la cuadrícula.

Por ejemplo, si queremos crear una cuadrícula de 4x4, donde la primera fila tenga un ancho de 100 píxeles y las tres filas siguientes tengan un ancho de 50 píxeles cada una, podemos definirlo así:

```
.contenedor{
  display:grid;
  grid-template-rows: 100px 50px 50px 50px;
}
```

Sin embargo, si queremos crear un diseño más complejo, podemos utilizar la propiedad "grid-template-areas". Por ejemplo, si queremos crear una cuadrícula donde la primera fila tenga una celda de ancho completo y dos celdas de ancho medio, y las tres filas siguientes tengan tres celdas de ancho completo, podemos definirlo así:

```
.contenedor{
  display:grid;
  grid-template-areas:
  "header header header"
  "nav    main   main"
  "nav    main   main"
  "footer footer footer";
}
```

En este caso, estamos definiendo **cuatro áreas** diferentes: "header", "nav", "main" y "footer". Cada área está compuesta por una o varias celdas de la cuadrícula, y podemos definir el tamaño y la posición de cada área en la cuadrícula utilizando la propiedad **"grid-template-areas"**.

La propiedad **"grid-template-areas"** utiliza una sintaxis donde cada área está representada por una **etiqueta entre comillas**, y cada fila de la cuadrícula está separada por un **salto de línea**. Para definir una celda vacía en la cuadrícula, podemos utilizar el carácter **"."** (punto).

Es importante destacar que la propiedad **"grid-template-areas"** debe estar acompañada de otras propiedades que definan el tamaño y la posición de las filas y columnas de la cuadrícula, como "grid-template-rows" y "grid-template-columns". Además, cada área debe estar compuesta por celdas contiguas, sin celdas vacías que las separen.

Para que el uso de esta propiedad este completo, debemos asignar las areas a un elemento, lo cual haremos con la propiedad de los elementos hijos **grid-area**, que veremos más adelante en este documento.

Justify-items

La propiedad **justify-items** en CSS Grid nos **permite alinear los elementos de una cuadrícula a lo largo del eje horizontal**. Es decir, nos permite controlar cómo se posicionan los elementos dentro de sus celdas a lo largo de la fila en la que se encuentran.

Por defecto, **todos los elementos de una cuadrícula se alinean al inicio de su celda correspondiente**. Esto significa que si hay espacio sobrante en la celda, el elemento estará a la izquierda de la celda. Sin embargo, con la propiedad **justify-items**, podemos cambiar este comportamiento.

Hay varios valores que podemos asignar a justify-items:

- **start**: este es el valor por defecto y alinea los elementos al inicio de la celda.
- **end**: alinea los elementos al final de la celda.
- **center**: alinea los elementos en el centro de la celda.
- **stretch**: hace que los elementos se estiren para llenar toda la celda.
- **baseline**: alinea los elementos en la línea base de la celda.

Por ejemplo, si queremos centrar todos los elementos de nuestra cuadrícula a lo largo del eje horizontal, podemos hacerlo de la siguiente manera:

```
.contenedor{
  display:grid;
  justify-items: center;
}
```

Align-items

La propiedad **align-items** en CSS Grid nos **permite controlar la alineación vertical de los elementos en la cuadrícula**.

Podemos establecer la propiedad align-items en cualquier contenedor de cuadrícula para alinear todos los elementos hijos verticalmente.

Hay varios valores que se pueden utilizar con la propiedad align-items, pero los más comunes son:

- **stretch** (valor por defecto): Estira los elementos para que se ajusten a la altura de la fila en la que se encuentran.
- **center**: Centra los elementos verticalmente en la celda de la cuadrícula.
- **start**: Alinea los elementos con el borde superior de la celda de la cuadrícula.
- **end**: Alinea los elementos con el borde inferior de la celda de la cuadrícula.

```
.contenedor{
  display:grid;
  align-items: center;
}
```

En el ejemplo anterior, utilizamos **align-items** para centrar verticalmente los elementos dentro de su propia celda.

Justify-content

La propiedad **justify-content** en CSS Grid nos **permite definir cómo se van a distribuir y alinear las columnas que hay dentro de nuestro contenedor**.

Los posibles valores para esta propiedad son:

- **start**: este valor alinea las columnas hacia el inicio del contenedor (es decir, hacia la izquierda).
- **end**: este valor alinea las columnas hacia el final del contenedor (es decir, hacia la derecha).
- **center**: este valor alinea las columnas al centro del contenedor en el eje horizontal.
- **stretch**: este valor estira las columnas para llenar todo el ancho del contenedor.
- **space-between**: este valor distribuye las columnas de manera uniforme en el contenedor, dejando un espacio en blanco entre ellas y en los bordes del contenedor.
- **space-around**: este valor distribuye las columnas de manera uniforme en el contenedor, dejando un espacio en blanco igual entre ellas y en los bordes del contenedor.

Es importante recordar que la propiedad justify-content sólo afecta a la distribución y alineación de las columnas dentro del contenedor, y no a la posición de los elementos dentro de cada celda.

Veamos cómo podemos usarlo para alinear y distribuir las columnas al centro del contenedor en el eje horizontal:

```
.contenedor{
  display:grid;
  grid-template-columns: repeat(3, 1fr);
  justify-content: center;
}
```

En este ejemplo, estamos definiendo un contenedor de cuadrícula con tres columnas de tamaño igual usando repeat(3, 1fr). Luego, establecemos **justify-content** en **center** para alinear y distribuir las columnas al centro del contenedor en el eje horizontal.

Align-content

Si el tamaño total de los elementos de la cuadrícula es **menor que el tamaño del contenedor de la cuadrícula**, la propiedad **align-content controla cómo se alinean y distribuyen los elementos de la cuadrícula a lo largo del eje de bloque**. Si el tamaño total de los elementos de la cuadrícula es **mayor que el tamaño del contenedor de la cuadrícula**, la propiedad **align-content controla cómo se distribuyen los elementos de la cuadrícula a lo largo del eje de bloque**.

Hay varias opciones que se pueden usar con la propiedad align-content, incluyendo:

- **stretch**: Esta es la opción predeterminada y estira los elementos de la cuadrícula para llenar el contenedor de la cuadrícula a lo largo del eje de bloque.
- **center**: Los elementos de la cuadrícula se alinean en el centro del contenedor de la cuadrícula a lo largo del eje de bloque.
- **start**: Los elementos de la cuadrícula se alinean en el inicio del contenedor de la cuadrícula a lo largo del eje de bloque.
- **end**: Los elementos de la cuadrícula se alinean en el final del contenedor de la cuadrícula a lo largo del eje de bloque.
- **space-between**: Los elementos de la cuadrícula se distribuyen uniformemente a lo largo del eje de bloque, con espacios iguales entre ellos.
- **space-around**: Los elementos de la cuadrícula se distribuyen uniformemente a lo largo del eje de bloque, con espacios iguales alrededor de ellos.
- **space-evenly**: Los elementos de la cuadrícula se distribuyen uniformemente a lo largo del eje de bloque, con espacios iguales entre ellos y alrededor de ellos.

La propiedad **align-content** se aplica al contenedor de la cuadrícula y no a los elementos de la cuadrícula individualmente. Se utiliza de la misma manera que **justify-content.**

Place-items

La propiedad **place-items** en CSS Grid es una propiedad que nos **permite alinear tanto horizontal como verticalmente los elementos de un grid**.

Esta propiedad **es una abreviatura de las propiedades justify-items y align-items**. Si se utiliza la propiedad place-items en un grid, estamos definiendo tanto la alineación horizontal como la vertical para todos los elementos del grid.

Por ejemplo, si definimos la propiedad place-items: center, todos los elementos del grid se alinearán tanto horizontal como verticalmente al centro.

```
.contenedor{
  display:grid;
  grid-template-columns: repeat(4, 150px);
  grid-template-rows: repeat(3, 100px);
  place-items: center;
}
```

Place-content

La propiedad **place-content** en CSS Grid es **una abreviatura para establecer tanto la alineación horizontal (justify-content) como la alineación vertical (align-content) en una sola declaración**.

Las palabras clave disponibles para place-content son las mismas que para justify-content y align-content, incluyendo start, end, center, space-between, space-around, space-evenly y stretch.

Por ejemplo, para centrar el contenido tanto horizontal como verticalmente dentro de un contenedor de cuadrícula, se puede utilizar la siguiente declaración:

```
.contenedor{
  display:grid;
  place-content: center;
}
```

Esta propiedad es especialmente útil cuando se trabaja con cuadrículas que no están completamente llenas de elementos, ya que puede ayudar a ajustar el contenido en el centro del contenedor.

Grid-auto-columns

La propiedad **grid-auto-columns** en CSS Grid es una propiedad que se utiliza para **especificar el tamaño predeterminado de las columnas en una cuadrícula cuando no se ha definido explícitamente un número de columnas** a través de la propiedad "grid-template-columns".

Por ejemplo, si tenemos una cuadrícula con tres columnas definidas, pero agregamos más elementos de los que caben en esas columnas, la propiedad "grid-auto-columns" se utilizará para especificar el tamaño de las columnas adicionales necesarias para acomodar estos elementos.

Podemos definir el valor de esta propiedad utilizando diferentes unidades de medida, como píxeles, porcentajes, fracciones, etc. Además, también podemos utilizar palabras clave como **"max-content"** o **"min-content"** para indicar el tamaño máximo o mínimo del contenido en una columna.

Es importante tener en cuenta que la propiedad "grid-auto-columns" solo se aplicará si no se ha definido explícitamente un número de columnas a través de la propiedad "grid-template-columns". Si se ha definido un número de columnas, la propiedad "grid-auto-columns" no tendrá ningún efecto.

```
.contenedor{
  display: grid;
  grid-template-columns: 100px 100px;
  grid-auto-columns: 50px;
  grid-gap: 10px;
}
```

En este ejemplo, hemos creado un grid con 2 columnas definidas con un ancho de 100px cada una, pero también hemos utilizado la propiedad grid-auto-columns para especificar que cualquier columna adicional que se agregue al grid tenga un ancho de 50px.

Al agregar más elementos dentro del contenedor, se crearán nuevas columnas automáticamente con un ancho de 50px cada una, como hemos especificado con la propiedad grid-auto-columns.

Grid-auto-rows

La propiedad **grid-auto-rows** en CSS Grid nos **permite establecer la altura predeterminada para las filas que no están explícitamente definidas en nuestro grid**. Esto significa que, si nuestro grid tiene una cantidad fija de filas definidas con **"grid-template-rows"** y agregamos más elementos a nuestro grid, se crearán automáticamente nuevas filas con la altura especificada en **"grid-auto-rows"**.

Un ejemplo sencillo de cómo usar la propiedad "grid-auto-rows" sería:

```
.contenedor{
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: 100px 100px;
  grid-auto-rows: 50px;
}
```

En este ejemplo, hemos definido un grid con 3 columnas y 2 filas explícitamente definidas de 100 píxeles de alto. Además, hemos establecido "grid-auto-rows" en 50 píxeles. Si agregamos más elementos al grid, se crearán automáticamente nuevas filas con una altura de 50 píxeles.

Grid-auto-flow

La propiedad **grid-auto-flow** en CSS Grid nos **permite definir cómo se comporta la colocación de los elementos en las celdas que quedan disponibles en la cuadrícula** después de colocar los elementos **definidos explícitamente con la propiedad grid-area**.

En otras palabras, esta propiedad nos permite definir c**ómo se distribuyen los elementos que no tienen un área definida en la cuadrícula**, es decir, aquellos elementos que se agregan automáticamente. Podemos elegir entre colocar los elementos nuevos en filas o columnas, de forma consecutiva o de forma paralela a los elementos existentes, y hacia arriba o hacia abajo, de acuerdo con la dirección de escritura del documento.

El valor predeterminado de esta propiedad es **"row"**, lo que significa que los elementos se agregarán a filas nuevas a medida que se necesiten. Pero si deseamos cambiar este comportamiento, podemos usar los valores **"column"**, **"dense"** o **"row dense"** para cambiar la dirección, la densidad o ambas, respectivamente.

A continuación, presentamos un ejemplo sencillo de cómo podemos usar la propiedad grid-auto-flow para colocar elementos automáticamente en una cuadrícula:

```
.contenedor{
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-flow: column;
}
```

En este ejemplo, hemos definido una cuadrícula de tres columnas y hemos establecido la propiedad grid-auto-flow en "column". Esto significa que los elementos que no tienen un área definida en la cuadrícula se agregarán en una nueva columna, en lugar de en una nueva fila.

Propiedades para elementos hijos

A continuación, veremos las propiedades más relevantes de los elementos hijos en CSS Grid

Grid-column-start

La propiedad **grid-column-start** en CSS Grid nos permite **establecer en qué línea comienza una celda dentro de una grilla**.

Cada línea en una grilla tiene un número asignado, comenzando por el número 1 en el borde izquierdo superior de la grilla. Por lo tanto, al usar grid-column-start, podemos especificar en qué línea de la grilla debe comenzar la celda.

Un ejemplo sencillo de cómo usar **grid-column-start** sería el siguiente:

Supongamos que tenemos una grilla de 3 columnas y 3 filas, y queremos establecer que un elemento div comience en la segunda columna. Para hacerlo, agregaríamos la propiedad **grid-column-start** al selector de CSS del elemento div y estableceríamos su valor en 2:

```
.contenedor {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 100px);
  grid-gap: 10px;
}

.contenedor div {
  grid-column-start: 2; /* comienza en la segunda columna */
}
```

En este ejemplo, hemos creado una grilla de 3 columnas y 3 filas, con una separación de 10 píxeles entre las celdas. Luego, agregamos la propiedad **grid-column-start: 2** al selector CSS del elemento div dentro de la grilla. Esto establece que la celda que contiene el elemento div comenzará en la segunda línea de la grilla, es decir, en la segunda columna.

Grid-column-end

La propiedad **grid-column-end** en CSS Grid es utilizada para **definir en qué línea de la cuadrícula terminará un elemento en el eje de las columnas**.

Esta propiedad acepta valores numéricos para indicar la línea de la cuadrícula en la que termina el elemento, así como también palabras clave que tienen un significado especial,

como **"span"** (se coloca span *número*, donde el número indica la cantidad de columnas o filas que se extiende) para indicar cuántas columnas abarca el elemento.
Por ejemplo, si queremos que un elemento div ocupe dos columnas en la cuadrícula, podemos utilizar la propiedad **"grid-column-end"** con el valor **"span 2"**, lo que indica que el elemento terminará en la segunda línea de la cuadrícula:

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

.contenedor div {
  grid-column-end: span 2;
}
```

En este ejemplo, la clase ".contenedor" crea una cuadrícula con tres columnas iguales, y el selector “.contenedor div” hace que el elemento ocupe dos de esas columnas. El valor "span 2" indica que el elemento se extenderá por dos columnas en la cuadrícula.

Es importante tener en cuenta que si se utiliza tanto la propiedad **"grid-column-start"** como **"grid-column-end"** en un mismo elemento, se creará un rango de columnas en la cuadrícula que será ocupado por el elemento.

Grid-row-start

La propiedad **grid-row-start** en CSS Grid **indica en qué fila empieza un elemento en la cuadrícula**.

Por ejemplo, si tenemos una cuadrícula con 3 filas y 3 columnas y queremos colocar un elemento en la segunda fila y la primera columna, podemos usar las propiedades grid-row-start y grid-column-start. Para colocarlo en la segunda fila, debemos asignarle el valor 2 a grid-row-start.

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px 100px 100px;
}

.item {
  grid-row-start: 2;

}
```

En este ejemplo, creamos una cuadrícula con 3 columnas y 3 filas de 100 píxeles de altura cada una. Luego, asignamos la clase "item" a un elemento que queremos colocar en la segunda fila de la cuadrícula, utilizando **grid-row-start**.

Grid-row-end

La propiedad **grid-row-end** es una propiedad en CSS Grid que nos permite **definir la línea final de una celda en la dirección de las filas**. En otras palabras, nos permite establecer en qué línea de fila debe terminar un elemento de cuadrícula.

Por ejemplo, si tenemos una cuadrícula de tres filas y tres columnas y queremos que un elemento ocupe las dos primeras filas, podemos usar la propiedad "grid-row-end" para indicar que la celda debe terminar en la línea 3, que es la línea que sigue a la segunda fila.

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 100px 100px 100px;
}

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

En este ejemplo, hemos creado una cuadrícula con tres filas y tres columnas. Luego, hemos aplicado la propiedad "grid-row-end" a un elemento de la cuadrícula llamado ".item". Con "grid-row-end: 3", estamos diciendo que el elemento debe terminar en la línea 3 de las filas, lo que significa que ocupará las dos primeras filas.

Es importante tener en cuenta que la propiedad "grid-row-end" no define la altura de la celda, sino simplemente la línea en la que termina. La altura se define en función de la posición de la celda en la cuadrícula y el contenido que contiene.

Grid-column

La propiedad **grid-column** en CSS Grid nos permite **definir en qué columnas debería ser colocado un elemento**. Podemos definir los números de columna o nombres de línea de la cuadrícula que se cruzan para especificar la ubicación de un elemento.

Por ejemplo, si queremos que un elemento se extienda desde la primera columna hasta la tercera columna, podemos usar la siguiente regla de CSS:

```css
.grid-item {
  grid-column: 1 / 4;
}
```

Grid-row

La propiedad **grid-row** es utilizada para **definir la ubicación de un elemento en las filas de la cuadrícula de CSS**. Esta propiedad toma dos valores, el primero indica la fila donde el elemento debe comenzar, y el segundo indica la fila donde el elemento debe terminar.

El valor de grid-row se especifica como una serie de números enteros separados por una barra diagonal (/). El primer número representa la fila donde el elemento comienza, mientras que el segundo número representa la fila donde el elemento termina.

Es importante destacar que las filas se cuentan desde arriba hacia abajo. Por ejemplo, si una cuadrícula tiene 3 filas, la primera fila es la número 1 y la última fila es la número 3.

```css
.container {
  display: grid;
  grid-template-rows: 50px 50px 50px;
}

.item {
  grid-row: 1 / 3; /* El elemento ocupa desde la fila 1 hasta la fila 3 */
}
```

En este ejemplo, la cuadrícula tiene 3 filas, cada una de 50 píxeles de alto. El elemento con clase .item se ubica en las filas 1 y 2, ya que se establece **grid-row: 1 / 3**.

Grid-area

La propiedad **grid-area** en CSS Grid es una abreviatura que nos permite indicar todas las propiedades anteriores en una única línea, es decir, **grid-row-start , grid-column-start, grid-row-end y grid-column-end.**

```
.item {
  grid-area: 1 / 2 / 4 / 5;
}
```

En este ejemplo, la celda se inicia en la primera fila y la segunda columna, y termina en la cuarta fila y la quinta columna, abarcando así tres filas y tres columnas. La cuadrícula se divide en un total de 12 filas y 12 columnas.

Además, como mencionamos anteriormente en este documento, una vez que tenemos un diseño de cuadrícula definido a través de **grid-template-areas**, podemos asignar elementos a áreas específicas de la cuadrícula utilizando la propiedad **grid-area**.

```
.contenedor{
  display:grid;
  grid-template-areas:
  "header header header"
  "nav    main   main"
  "nav    main   main"
  "footer footer footer";
}
```

Por ejemplo, si queremos colocar un elemento con la clase logo en la celda superior izquierda de nuestro diseño anterior, podríamos hacerlo así:

```
.logo {
  grid-area: header header;
}
```

Aquí estamos asignando el elemento .logo a la celda header header de la cuadrícula, que corresponde a la esquina superior izquierda , dejándonos espacio para un elemento más en el header, como podrían ser los enlaces de navegación.

Por lo tanto, podemos pensar en **grid-template-areas** como la definición de las áreas nombradas en nuestra cuadrícula y en **grid-area** como la asignación de elementos específicos a esas áreas. Juntas, estas propiedades nos permiten crear diseños complejos y flexibles en CSS Grid.

Nucba tip: *Sigan investigando sobre los temas aquí expuestos. Practiquen, apoyense en la documentación oficial y sobre todo mantengan la constancia y la curiosidad a medida que vayan aprendiendo cosas nuevas.*

#HappyCoding 🚀