CSS Grid

CSS Grid es un módulo de CSS que permite crear layouts bidimensionales de manera precisa y flexible. A diferencia de Flexbox, que se enfoca en una dimensión (horizontal o vertical), CSS Grid facilita la organización tanto en filas como en columnas, proporcionando un control más avanzado sobre el diseño de las interfaces.

Ventajas de CSS Grid:

- Permite crear layouts complejos y organizados en dos dimensiones (filas y columnas) de forma más sencilla.
- Facilita la creación de layouts responsivos y adaptables.
- Proporciona mayor control sobre el tamaño y el posicionamiento de los elementos en un contenedor.
- **Índice**

Conceptos básicos de CSS Grid

CSS Grid se basa en dos conceptos principales:

1. **Contenedor Grid (Grid Container)**: Es el elemento que actúa como el contenedor principal de la cuadrícula. Se declara con `display: grid;` en CSS.
2. **Ítems Grid (Grid Items)**: Son los elementos hijos directos del contenedor grid. Estos elementos se organizan en filas y columnas dentro del grid.

Propiedades del Contenedor Grid

Las propiedades principales que se aplican al contenedor grid para definir su estructura son:

- `display: grid;`: Define el contenedor como un grid bidimensional.
- `grid-template-columns`: Establece las columnas del grid.
- `grid-template-rows`: Define las filas del grid.
- `grid-gap`: Define el espacio entre las filas y columnas del grid.
    - `grid-row-gap`: Espacio entre filas.
    - `grid-column-gap`: Espacio entre columnas.
    - ***Nota***: Tambien puedo declarar el espacio entre filas y columnas de la siguiente forma:
        - `gap: 15px 30px;`: 15px entre filas, 30px entre columnas.

Definición de columnas en CSS Grid

La propiedad `grid-template-columns` define cómo se distribuyen las columnas dentro del contenedor grid. Se pueden especificar tamaños fijos, fracciones (`fr`), porcentajes o incluso repeticiones automáticas. Ejemplos comunes:

- `grid-template-columns: 50% 50%;`: Define dos columnas con un ancho del 50% cada una.
- `grid-template-columns: calc(50% - 7.5px) calc(50% - 7.5px);`: Usa una función `calc()` para calcular el ancho de las columnas.
- `grid-template-columns: repeat(4, 24%);`: Utiliza la función `repeat()` para crear 4 columnas con un ancho del 24% cada una.
- `grid-template-columns: 1fr 1fr 1fr;`: Establece tres columnas iguales usando unidades fraccionales (`fr`).
- `grid-template-columns: 1fr 1fr 2fr;`: Define tres columnas, donde la última tiene el doble de ancho que las dos primeras.
- `grid-template-columns: repeat(3, 1fr);`: Es una forma simplificada de definir tres columnas fraccionales.

Definición de filas en CSS Grid

La propiedad `grid-template-rows` define las filas en el contenedor grid. Similar a las columnas, se pueden usar tamaños específicos, unidades fraccionales o repeticiones:

- `grid-template-rows: repeat(2, 1fr) 3fr repeat(2, 1fr);`: Define dos filas con una fracción cada una, una fila más alta de tres fracciones, y luego dos filas adicionales con una fracción cada una.

Posicionamiento de elementos en CSS Grid

CSS Grid permite posicionar elementos de forma precisa utilizando las propiedades `grid-column` y `grid-row`, así como sus subpropiedades: `grid-column-start`, `grid-column-end`, `grid-row-start` y `grid-row-end`. Estas subpropiedades proporcionan un control más detallado para definir en qué punto debe comenzar y terminar un elemento dentro de la **rejilla (grid)**. A continuación, se explica el uso de cada una:

Propiedades para Posicionar Columnas

1. `grid-column-start`:
    - Define en qué línea de **columna** comienza un elemento.
    - Ejemplo:

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

        - En este caso, el elemento comienza en la **columna 2** de la rejilla.
2. `grid-column-end`:
    - Define en qué línea de **columna** termina un elemento.
    - Ejemplo:

```
.item {
  grid-column-end: 4;
}
```

- El elemento se extiende hasta la **columna 4**, ocupando las columnas 2 y 3.

3. **Combinación de `grid-column-start` y `grid-column-end`**:
   - Se pueden usar juntas para definir la **extensión horizontal** exacta de un elemento.
   - Ejemplo:

```
.item {
  grid-column-start: 1;
  grid-column-end: 3;
}
```

- El elemento comenzará en la **columna 1** y se extenderá hasta la **columna 3**, ocupando **dos columnas** en total.

4. **Shorthand: `grid-column`**:
   - `grid-column` es una propiedad **abreviada** que combina `grid-column-start` y `grid-column-end`.
   - Ejemplo:

```
.item {
  grid-column: 2 / 4;
}
```

- El elemento comienza en la **columna 2** y termina en la **columna 4**, ocupando **dos columnas** en total.

Propiedades para Posicionar Filas

1. `grid-row-start`:
   - Define en qué línea de **fila** comienza un elemento.
   - Ejemplo:

```
.item {
  grid-row-start: 1;
}
```

- El elemento comienza en la **fila 1** de la rejilla.

2. `grid-row-end`:
    - Define en qué línea de **fila** termina un elemento.
    - Ejemplo:

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

        - El elemento se extiende hasta la **fila 3**, ocupando las filas 1 y 2.

3. **Combinación de** `grid-row-start` **y** `grid-row-end`:
    - Se pueden usar juntas para definir la **extensión vertical** exacta de un elemento.
    - Ejemplo:

```
.item {
  grid-row-start: 2;
  grid-row-end: 5;
}
```

        - El elemento comienza en la **fila 2** y se extiende hasta la **fila 5**, ocupando **tres filas** en total.

4. **Shorthand:** `grid-row`:
    - `grid-row` es una propiedad **abreviada** que combina `grid-row-start` y `grid-row-end`.
    - Ejemplo:

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

        - El elemento se extiende desde la **fila 1** hasta la **fila 4**, ocupando **tres filas** en total.

Unidades fraccionales en Grid

La unidad `fr` es una de las características clave de CSS Grid y permite asignar espacio de manera proporcional en el contenedor grid:

- Un valor de `1fr` indica que el ítem ocupará una parte proporcional del espacio disponible en relación con otros ítems que también usen `fr`.
- Si todos los ítems tienen un valor de `1fr`, el espacio se distribuye equitativamente entre ellos.
- Un valor de `2fr` hará que el ítem ocupe el doble de espacio que los elementos con `1fr`.

Estas unidades son útiles para crear layouts flexibles y responsivos sin necesidad de usar porcentajes o tamaños fijos.