UT4_3.- HOJAS DE ESTILO. AVANZADO - GRID

RESULTADOS DE APRENDIZAJE ASOCIADOS
1. Reconoce las características de lenguajes de marcas analizando e interpretando fragmentos de código. 2. Utiliza lenguajes de marcas para la transmisión de información a través de la Web analizando la estructura de los documentos e identificando sus elementos
CRITERIOS DE EVALUACIÓN
- De RA1 – desde CEA hasta CEK - De RA2 – desde CEA hasta CEH

UT4_3.- HOJAS DE ESTILO. AVANZADO - GRID

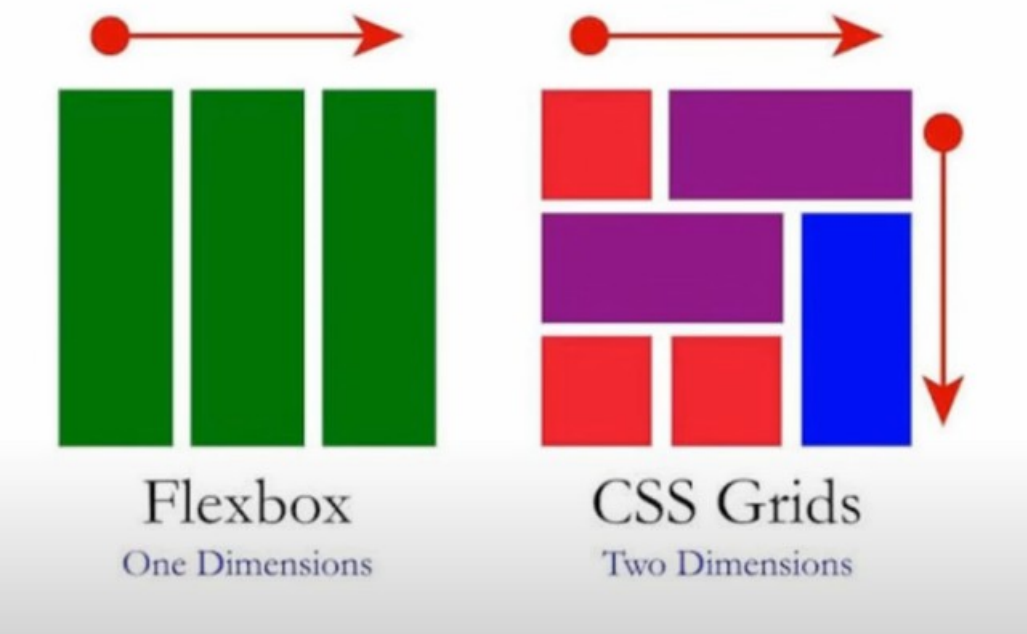
Índice de contenido

1.- Introducción

CSS puede ser un desafío, especialmente para aquellos que son nuevos en el tema. Uno de los problemas más comunes **es cómo colocar y distribuir elementos en una página web**. Mientras que herramientas como el posicionamiento, los *floats* y los elementos en bloque o en línea pueden ser útiles, a menudo son insuficientes para crear diseños y estructuras modernas.

Para abordar esta problemática, se ha desarrollado el sistema de elementos flexibles Flex.

Aunque es un gran avance, Flex está diseñado para estructuras de una sola dimensión, por lo que aún se requiere algo más poderoso para crear estructuras web multidimensionales de manera rápida y sencilla.



Entra en escena **Grid CSS**, una herramienta que toma lo mejor del sistema grid y lo mejora con numerosas características adicionales.

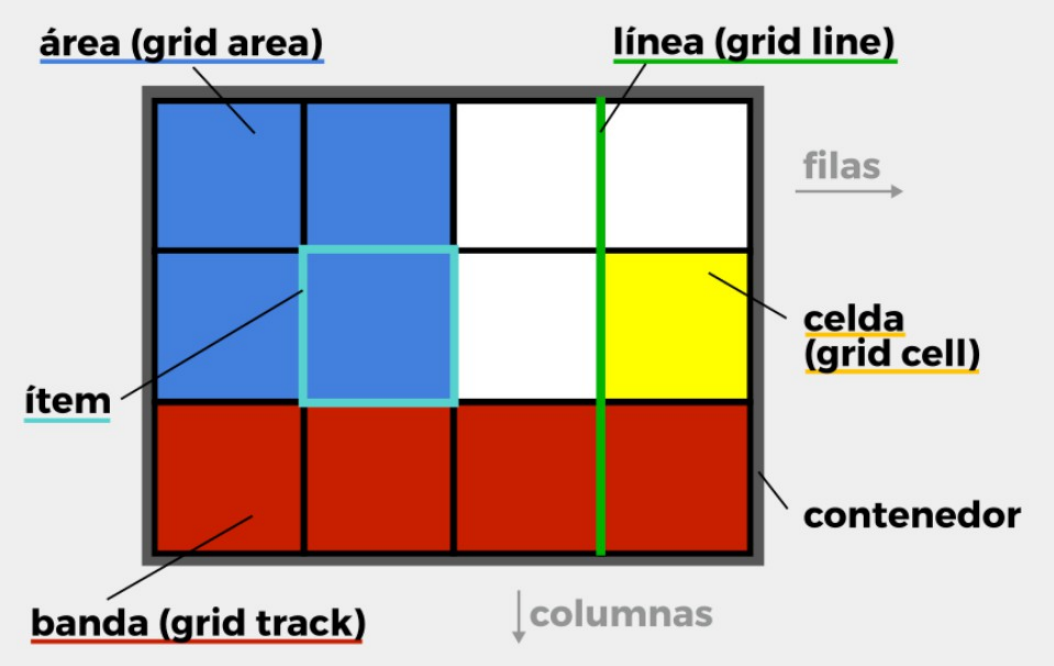
Con Grid CSS, es posible crear cuadrículas flexibles y potentes en cuestión de minutos, gracias a una nueva familia de propiedades CSS.

Antes de comenzar a trabajar con Grid CSS, **es recomendable tener un conocimiento previo del sistema de maquetación mediante Flex**, ya que Grid toma muchos de sus conceptos y filosofía.

Manual:
https://www.w3schools.com/css/css_grid.asp

2.- Conceptos

Para crear diseños basados en Grid CSS necesitaremos tener en cuenta una serie de conceptos que utilizaremos a partir de ahora y que definiremos a continuación:



- **Contenedor:** El elemento padre contenedor que definirá la cuadrícula o rejilla.

- **Ítem:** Cada uno de los hijos que contiene la cuadrícula (elemento contenedor).

- **Celda (grid cell):** Cada uno de los cuadritos (unidad mínima) de la cuadrícula.

- **Área (grid area):** Región o conjunto de celdas de la cuadrícula.

- **Banda (grid track):** Banda horizontal o vertical de celdas de la cuadrícula.

- **Línea (grid line):** Separador horizontal o vertical de las celdas de la cuadrícula.

3.- Modalidades de Grid

Para activar la cuadrícula grid hay que utilizar sobre el elemento contenedor la propiedad **display** y especificar uno de los dos valores que queramos utilizar: **grid** o **inline-grid**.

Tipo de elemento	Descripción
`inline-grid`	Establece una cuadrícula con ítems **en línea**, de forma equivalente a `inline-block`.
`grid`	Establece una cuadrícula con ítems **en bloque**, de forma equivalente a `block`.

4.- Definir filas y columnas

En Grid CSS, la forma principal de definir una cuadrícula es indicar el tamaño de sus filas y sus columnas de forma explícita. Para ello, sólo tenemos que usar las propiedades CSS ***grid-template-columns*** y ***grid-template-rows***:

Propiedad	Valor	Descripción
grid-template-columns	[*col1*] [*col2*] ...	Establece el SIZE de cada columna (*col 1, col 2...*).
grid-template-rows	[*fila1*] [*fila2*] ...	Establece el SIZE de cada fila (*fila 1, fila 2...*).

5.- Grid por áreas

Los Grids por área ofrecen una forma avanzada de especificar la posición precisa de cada elemento en una cuadrícula. Utilizando la propiedad **grid-template-areas** en el contenedor padre y **grid-area** en los elementos hijos, es posible nombrar y posicionar cada área de manera clara y sencilla.

Por ejemplo, con el siguiente código HTML y CSS, se puede crear una cuadrícula con una cabecera en la parte superior, un menú lateral a la izquierda, un área principal a la derecha y un pie de página en la parte inferior:

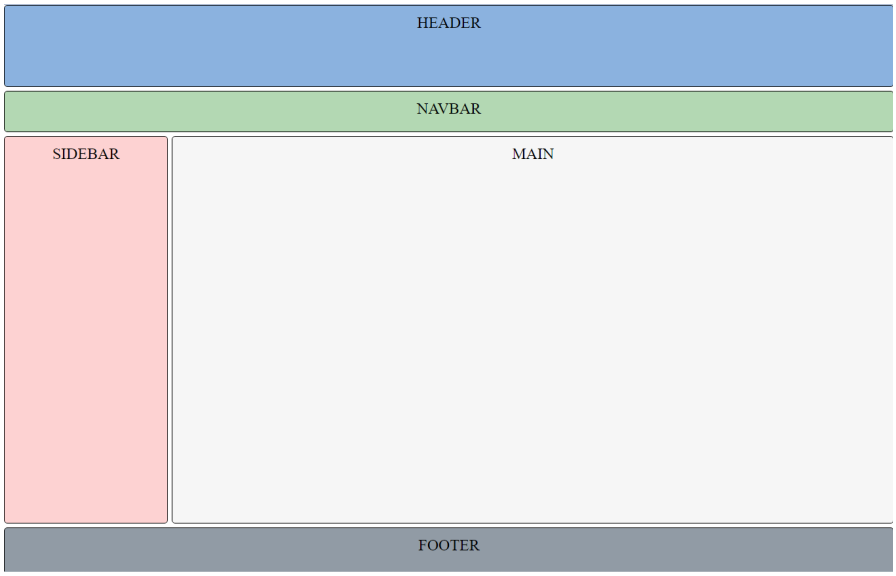
```
<body class="grid-container">
    <header class="header">HEADER</header>
    <nav class="navbar">NAVBAR</nav>
    <aside class="sidebar">SIDEBAR</aside>
    <article class="main">MAIN</article>
    <footer class="footer">FOOTER</footer>
</body>
```

```css
/**La propiedad de CSS box-sizing indica cómo se deben calcular las medidas de un elemento **/
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
  }

  /** Para que ocupe el 100% del alto **/
  html {
    height: 100%;
  }

  body{
    font-size:1.2rem;
    min-height:100%
  }

  /*Efectos para el grid */
  .grid-container > * {
    border:solid 1px black;
    border-radius: 4px;
    padding: 10px;
    text-align: center;
  }
```

```css
/** Plantilla del grid **/
.grid-container{
    /** Ajustamos el ancho de las columnas: 1ª columna un ancho de 200px
    y la 2ª un ancho automático **/

    grid-template-columns: 200px auto;
    /* Ajustamos el alto de las columnas */
    grid-template-rows: 100px 50px auto 100px;
    display:grid;
    gap:5px;

    grid-template-areas:
        "header header"
        "navbar navbar"
        "sidebar main"
        "footer footer";
}
```

```css
.header {
  grid-area: header;
  background-color: #85aedd;
}

.navbar {
  grid-area: navbar;
  background-color: #afd6af;
}

.sidebar {
  grid-area: sidebar;
  background-color: #fdcfcf;
}

.main {
  grid-area: main;
  background-color: #f5f5f5;
}

.footer {
  grid-area: footer;
  background-color: #8c96a0;
}
```

La salida será:



Sin embargo, es importante tener en cuenta que cada celda debe contener texto, de lo contrario la cuadrícula se adaptará a su contenido (que no existe) y parecerá que no existe.

En resumen, los Grids por área ofrecen una forma intuitiva y poderosa de crear cuadrículas altamente personalizadas con una gran flexibilidad en la disposición y posición de cada área.

6.- Unidad de fracción restante

En el ejemplo anterior se han utilizado ***píxeles*** como unidades de las celdas de la cuadrícula, sin embargo, también podemos utilizar otras unidades (o incluso combinarlas): **porcentajes**, la palabra clave **auto** (que obtiene el tamaño restante) o la unidad especial de grid **fr** (fracción restante):

```
<div class="container">
    <div class="item1 color-1"> item-1 </div>
    <div class="item color-2"> item-2 </div>
    <div class="item color-3"> item-3 </div>
    <div class="item color-4"> item-4 </div>
    <div class="item color-5"> item-5 </div>
  </div>
```

```
.container{
  display: grid;
  /* 3 filas */
  grid-template-columns: 1fr 3fr 2fr;
  /* 2 columnas */
  grid-template-rows: 9rem 5rem;
}
```

6.1.- Mezclando columnas fijas y flexibles.

```css
.container{
  display: grid;
  /* 3 filas */
  grid-template-columns: 29rem 3fr 2fr;
  /* 2 columnas */
  grid-template-rows: 9rem 5rem;
}
```

7.- Huecos en grid

Por defecto, la cuadrícula tiene todas sus celdas pegadas a sus celdas contiguas. Aunque sería posible darle un margin a las celdas dentro del contenedor, existe una forma más apropiada: los huecos (gutters).

Para especificar los huecos (espacio entre celdas) podemos utilizar las propiedades **column-gap** y/o **row-gap**. En ellas indicaremos el tamaño de dichos huecos:

Propiedad	Descripción
column-gap	Establece el SIZE de los huecos entre columnas (*líneas verticales*).
row-gap	Establece el SIZE de los huecos entre filas (*líneas horizontales*).

```
.grid {
  column-gap: 100px;
  row-gap: 10px;
}
```

Ejemplo:

grid gaps (huecos)

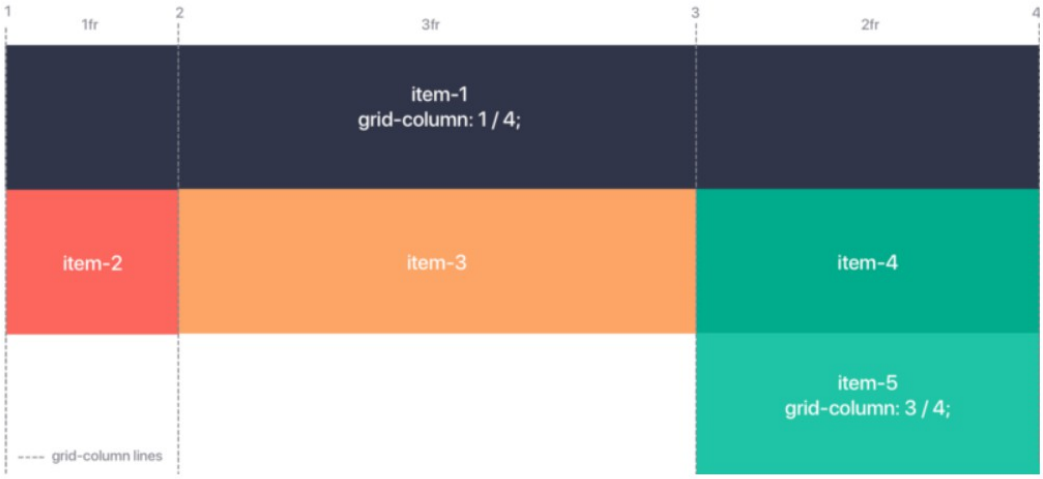
8.- Celdas irregulares

Las propiedades para crear esa distribución irregular de una celda del grid son las siguientes:

Propiedad	Descripción
grid-column-start	Indica en que columna empezará el ítem de la cuadrícula.
grid-column-end	Indica en que columna terminará el ítem de la cuadrícula.
grid-row-start	Indica en que fila empezará el ítem de la cuadrícula.
grid-row-end	Indica en que fila terminará el ítem de la cuadrícula.

Ejemplo (para el mismo body que antes):

```css
.container{
    display: grid;
    grid-template-columns: 1fr 3fr 2fr;
    grid-template-rows: 9rem 9rem 9rem;
}

/*Celdas irregulares*/
.item-1 {
 grid-column: 1 / 4;
}
.item-5 {
 grid-column: 3 / 4;
}

.color-1{
    background-color: black;
    color: white;
}
```



Más información:

https://lenguajecss.com/css/maquetacion-y-colocacion/grid-css/