ISEMED
MARÍA AUXILIADORA
SAN PEDRO SULA

Tablas y Columnas Avanzadas

Diseño Web II

Duodécimo grado

1. La Evolución de la Maquetación Web

La historia del diseño web ha estado marcada por una constante búsqueda de flexibilidad y eficiencia. En sus orígenes, los diseñadores y desarrolladores dependían en gran medida de las **tablas HTML** para estructurar la maquetación de una página. Aunque diseñadas para representar datos tabulados, las tablas se convirtieron en la herramienta de facto para crear un diseño visual, anidando celdas dentro de celdas para definir columnas, cabeceras y pies de página. Esta práctica, si bien funcional, tenía serias desventajas: el código resultante era verboso, difícil de mantener y, lo más importante, no era responsivo. Los diseños de tablas eran rígidos y no se adaptaban bien a los diferentes tamaños de pantalla, lo que resultaba en barras de desplazamiento horizontales y una mala experiencia de usuario.

Posteriormente, surgieron soluciones basadas en CSS, como la propiedad float, que permitían colocar elementos uno al lado del otro. Esta era una técnica más semántica, ya que separaba el contenido (HTML) de la presentación (CSS), pero aún presentaba desafíos. Los floats requerían un manejo cuidadoso para evitar que los elementos se superpusieran o rompieran el diseño, obligando a usar técnicas de "limpieza" como *clearing floats*. Era una solución manual y a menudo frágil para layouts complejos.

La verdadera revolución llegó con la introducción de **CSS Grid** y **Flexbox**. Estos dos módulos de CSS, concebidos para la maquetación, ofrecieron por primera vez herramientas nativas y poderosas para construir diseños complejos y responsivos. A diferencia de sus predecesores, no eran trucos o adaptaciones, sino soluciones robustas que permitieron a los desarrolladores un control sin precedentes sobre la disposición de los elementos en la página.

2. CSS Grid Layout:

CSS Grid es un sistema de maquetación bidimensional que permite a los desarrolladores organizar elementos en la interfaz tanto en **filas** como en **columnas** al mismo tiempo. Es la solución ideal para el diseño de la estructura principal de una página (el *macro-layout*), como un **dashboard**, un panel de control, o una página de inicio con múltiples secciones. Grid es la primera herramienta de CSS que ofrece un verdadero sistema de cuadrícula, similar al usado en el diseño gráfico, directamente en la web.

2.1. Propiedades del Contenedor (Grid Container)

Para comenzar, se aplican las siguientes propiedades al elemento padre que contendrá a todos los elementos de la cuadrícula:

- **`display: grid`**;: El pilar fundamental. Esta propiedad activa el contexto de cuadrícula para todos los elementos hijos directos.
- **`grid-template-columns:`** Define el número y el ancho de las columnas de la cuadrícula. Acepta múltiples valores, lo que permite crear columnas de diferentes tamaños.
- **`grid-template-rows:`** De forma análoga, define el número y la altura de las filas.

2.2. Unidades de Medida en CSS Grid

Las unidades de medida juegan un papel crucial en la flexibilidad de Grid. Las más comunes son:

- **Unidades Absolutas (`px`):** Ideales para elementos con un tamaño fijo, pero restan adaptabilidad.
- **Unidades Relativas (`%, em, rem, vw, vh`):** Relacionan el tamaño con el contenedor, la fuente raíz o el *viewport*.
- **La Unidad `fr` (fracción):** Esta es la unidad estrella de CSS Grid. Representa una fracción del espacio disponible en la cuadrícula. Es la base del diseño fluido. Por ejemplo, `grid-template-columns: 1fr 2fr 1fr;` creará tres columnas: la primera y la tercera ocuparán una porción del espacio, mientras que la del medio ocupará el doble.
- **La función `minmax()`:** Una de las funciones más poderosas de Grid. Permite definir un rango de tamaño para una banda. `grid-template-columns: minmax(100px, 1fr) 1fr`; crea una columna que nunca será menor de 100 píxeles pero se expandirá para ocupar una fracción del espacio si hay más espacio disponible.

2.3. Controlando el Espacio y la Distribución

- `gap`: Simplifica la tarea de añadir espacio entre las bandas de la cuadrícula. Se puede usar de forma abreviada (`gap: 20px`;) o específica (`row-gap, column-gap`).
- **Alineación:** Grid hereda muchas de las propiedades de alineación de Flexbox, pero las aplica en dos dimensiones.
    - `justify-items y align-items`: Controlan la alineación de los elementos individualmente dentro de su celda, en los ejes horizontal y vertical, respectivamente.
    - `justify-content y align-content:` Controlan la alineación de la cuadrícula completa dentro del contenedor si esta no ocupa todo el espacio.

3. Flexbox: El Maestro de la Alineación Unidimensional

Flexbox (Flexible Box Layout) es el sistema de maquetación de referencia para la **alineación y distribución de elementos en una única dirección**, ya sea en una **fila** o en una **columna**. Es perfecto para el *micro-layout*, como la barra de navegación de un sitio, una galería de fotos o un grupo de tarjetas.

3.1. Propiedades y Conceptos Clave de Flexbox

Flexbox opera con dos ejes: el **eje principal** (*main axis*) y el **eje transversal** (*cross axis*).

- `display: flex;:` Declara el contenedor como un **contenedor flexible** y sus hijos directos como **elementos flexibles**.
- `flex-direction:` Define el eje principal. Puede ser `row` (horizontal), `row-reverse, column` (vertical), o `column-reverse`.
- `justify-content:` Alinea los elementos a lo largo del eje principal. Las opciones como `flex-start, flex-end, center, space-between y space-around` ofrecen un control granular sobre la distribución horizontal o vertical.
- `align-items:` Alinea los elementos a lo largo del eje transversal.
- `flex-wrap:` Maneja el desbordamiento de contenido. Si los elementos no caben en una sola línea, esta propiedad permite que se envuelvan en múltiples líneas (wrap).
- **Propiedades de los elementos hijos:**
    - `flex-grow:` Define la capacidad de un elemento para crecer y ocupar el espacio disponible.
    - `flex-shrink:` Define la capacidad de un elemento para encogerse y evitar el desbordamiento.
    - `flex-basis:` Especifica el tamaño inicial de un elemento flexible antes de que el espacio restante se distribuya.

4. La Sinergia Perfecta: Integrando Grid y Flexbox

La verdadera potencia del diseño web moderno reside en la combinación estratégica de Grid y Flexbox. No son tecnologías competidoras, sino herramientas complementarias que resuelven diferentes problemas de maquetación de manera eficiente.

Escenario de Maquetación	Herramienta Recomendada	Razón
Diseño de página completo (Dashboard, blog, *landing page*)	**CSS Grid**	Permite una maquetación bidimensional y un control total de la estructura principal.
Barra de navegación con enlaces espaciados	**Flexbox**	La alineación de elementos en una única fila es su punto fuerte.
Galería de imágenes o tarjetas	**CSS Grid** con repeat() y auto-fit	Maneja la disposición en filas y columnas de manera fluida y responsiva.
Contenido de un solo componente (Ej. un *card* con imagen y texto)	**Flexbox**	Organiza y alinea los elementos internos de forma sencilla.

La práctica actual de desarrollo web es utilizar Grid para el diseño general (el "esqueleto" de la página) y luego usar Flexbox dentro de las celdas de esa cuadrícula para alinear los elementos de los componentes. Esta modularidad hace que el código sea más legible, mantenible y escalable. La combinación de ambas técnicas permite un control preciso sobre cada aspecto del diseño, desde la macro-estructura hasta el último detalle de alineación.

5. El Futuro de la Maquetación Web

La evolución de las tecnologías de maquetación demuestra un claro avance hacia la simplicidad, la flexibilidad y la semántica. CSS Grid y Flexbox han liberado a los desarrolladores de las limitaciones de las tablas y los hacks de **`float`.** El conocimiento profundo de estas herramientas no solo es un requisito para el desarrollo frontend moderno, sino que también fomenta la creación de diseños más robustos, accesibles y con un mejor rendimiento.
A medida que los navegadores continúan mejorando su soporte, la comunidad de desarrolladores ha adoptado estas tecnologías como el nuevo estándar. La capacidad de construir diseños complejos sin un solo truco de maquetación ha transformado por completo la forma en que se aborda la creación de interfaces web.

Tradicionalmente, esto habría requerido una compleja red de tablas anidadas. Con Grid, se define un *Grid Container*, se establecen las filas y columnas y se nombran las áreas. El posicionamiento de cada elemento (barra lateral, encabezado, área de gráficos) se vuelve tan sencillo como asignarle el nombre de área correspondiente, lo que también facilita la reubicación de elementos en diferentes tamaños de pantalla utilizando *media queries* de manera eficiente.
La adopción de Grid y Flexbox también tiene un impacto positivo en la accesibilidad y la optimización para motores de búsqueda (SEO), ya que el código HTML se mantiene limpio y centrado en el contenido, mientras que el diseño es manejado enteramente por CSS. Esto permite que los lectores de pantalla y los rastreadores web interpreten el contenido de manera más lógica y eficiente.