Unidad 4: Grids

Propiedades Básicas de CSS Grid Aplicadas al Contenedor (Padre)

En este módulo, exploraremos las propiedades fundamentales de CSS Grid que se aplican al contenedor (padre). Estas propiedades permiten definir la estructura y disposición de los elementos hijos dentro de una grilla. A continuación, se detallan cada una de estas propiedades junto con ejemplos de código CSS y su resultado visual.

display: grid

La propiedad `display: grid` convierte un elemento en un contenedor de grilla, lo que permite organizar sus elementos hijos en filas y columnas.

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

grid-template-columns

La propiedad **`grid-template-columns`** define la estructura de las columnas en la grilla. Puedes especificar el tamaño de cada columna usando unidades como **`px`**, **`%`**, **`fr`**, entre otras.

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

Ejemplo Visual

Columna 1	Columna 2	Columna 3
1fr	2fr	1fr

grid-template-rows

La propiedad `grid-template-rows` define la estructura de las filas en la grilla, similar a cómo `grid-template-columns` define las columnas.

```
.container {
    display: grid;
    grid-template-rows: 100px 200px;
}
```

Ejemplo Visual

Fila 1 (100px)

Fila 2 (200px)

---

grid-template-areas

La propiedad `grid-template-areas` permite nombrar áreas dentro de la grilla para facilitar el posicionamiento de los elementos hijos. Se usan nombres de áreas definidos en los elementos hijos.

```
.container {
    display: grid;
    grid-template-areas:
        "header header"
        "sidebar main";
}
```

```
.header {
    grid-area: header;
}
.sidebar {
    grid-area: sidebar;
}
.main {
    grid-area: main;
}
```

Ejemplo Visual

Header	Header
Sidebar	Main

column-gap

La propiedad `column-gap` (anteriormente `grid-column-gap`) define el espacio entre las columnas de la grilla.

```
.container {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    column-gap: 20px;
}
```

Ejemplo Visual

| Columna 1 | | Columna 2 | | Columna 3 | |----------| 20px |----------| 20px |----------|

row-gap

La propiedad `row-gap` (anteriormente `grid-row-gap`) define el espacio entre las filas de la grilla.

```
.container {
    display: grid;
    grid-template-rows: 100px 100px;
    row-gap: 20px;
}
```

Ejemplo Visual

| Fila 1 (100px) | | | 20px | | Fila 2 (100px) |

Ejemplo Completo

A continuación, se presenta un ejemplo completo que combina varias de estas propiedades para ilustrar cómo se pueden usar juntas para crear una grilla compleja.

```
.container {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    grid-template-rows: 100px 200px;
    grid-template-areas:
        "header header header"
        "sidebar main .";
    column-gap: 20px;
    row-gap: 10px;
}
.header {
    grid-area: header;
}
.sidebar {
```

```
    grid-area: sidebar;

}

.main {

    grid-area: main;

}
```

Ejemplo Visual Completo

Header	Header	Header
Sidebar	Main	

Explicación:

- En este ejemplo, tenemos un contenedor de grilla con tres columnas y dos filas.
- Las áreas se definen con nombres como "header", "sidebar" y "." (un punto vacío).
- Se aplica un espacio de 20px entre columnas y 10px entre filas.
- Cada elemento hijo (header, sidebar, main) se posiciona en el área correspondiente usando la propiedad grid-area.
- Se aplican colores de fondo diferentes a cada área para visualizar la estructura de la grilla.

Items y Áreas de la Grilla

Flexibilización de Ítems Individuales en CSS Grid

En este módulo, aprenderemos cómo flexibilizar y posicionar ítems individuales dentro de una grilla de CSS Grid. Utilizaremos propiedades como `grid-column`, `grid-row`, y `grid-area` para ajustar la posición y el tamaño de elementos específicos. A continuación, se explica cada propiedad junto con ejemplos de código CSS y su resultado visual.

grid-column

La propiedad **`grid-column`** se utiliza para definir en qué columna comienza un ítem y cuántas columnas ocupa. Esta propiedad combina **`grid-column-start`** y **`grid-column-end`**.

```
.item {
```

grid-column: 2 / 4;

}

Ejemplo Visual

Columna 1	Columna 2	Columna 3	Columna 4
	Ítem	**Ítem**	

En este ejemplo, el ítem comienza en la columna 2 y termina en la columna 4, ocupando dos columnas.

---

grid-row

La propiedad `grid-row` se utiliza para definir en qué fila comienza un ítem y cuántas filas ocupa. Esta propiedad combina `grid-row-start` y `grid-row-end`.

.item {

grid-row: 1 / 3;

}

Ejemplo Visual

Fila 1 (

Ítem

)

Fila 2 (

Ítem

)

Fila 3

En este ejemplo, el ítem comienza en la fila 1 y termina en la fila 3, ocupando dos filas.

---

grid-area

La propiedad **`grid-area`** es una forma abreviada de definir **`grid-row-start`**, **`grid-column-start`**, **`grid-row-end`** y **`grid-column-end`** en una sola propiedad.

.item {

grid-area: 1 / 2 / 3 / 4;

}

Ejemplo Visual

Fila 1 (Col 2-4:

Ítem

)

Fila 2 (Col 2-4:

Ítem

)

Fila 3

En este ejemplo, el ítem comienza en la fila 1, columna 2, y termina en la fila 3, columna 4.

Ejemplo Completo

A continuación, se presenta un ejemplo completo que combina varias de estas propiedades para ilustrar cómo se pueden usar juntas para posicionar elementos específicos dentro de una grilla.

```
.container {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    grid-template-rows: repeat(3, 100px);
    gap: 10px;
}
.item1 {
    grid-column: 1 / 3;
}
.item2 {
    grid-row: 2 / 4;
}
.item3 {
    grid-area: 1 / 3 / 3 / 5;
}
```

Ejemplo Visual Completo

Item 1	**Item 1**	**Item 3**	**Item 3**
Item 1	**Item 1**	**Item 3**	**Item 3**
	Item 2	**Item 3**	**Item 3**
	Item 2		

En este ejemplo:

- `Item 1` ocupa las columnas 1 y 2 de la primera fila.
- `Item 2` ocupa las filas 2 y 3 de la segunda columna.
- `Item 3` ocupa desde la fila 1, columna 3 hasta la fila 2, columna 4.

Estos conceptos y ejemplos proporcionan una base sólida para comenzar a trabajar con CSS Grid y flexibilizar ítems individuales dentro de una grilla.

Propiedades de Distribución en CSS Grid

En este módulo, exploraremos cuatro propiedades fundamentales de **CSS Grid** que controlan la distribución de los elementos en la grilla: `justify-items`, `align-items`, `justify-content` y `align-content`. Estas propiedades nos permiten alinear y distribuir los ítems y las áreas de la grilla de manera precisa. A continuación, te presentamos una explicación actualizada de cada propiedad junto con ejemplos claros y visuales que ilustran sus efectos.

`justify-items`

La propiedad `justify-items` alinea los elementos hijos dentro de sus celdas a lo largo del eje horizontal (izquierda a derecha). Los valores posibles son:

- `start` (inicio)
- `end` (fin)
- `center` (centro)
- `stretch` (ocupa todo el ancho disponible)

Ejemplo de Código

```
.container {

    display: grid;

    grid-template-columns: repeat(3, 1fr);

    justify-items: center;

}
```

Visualización Actualizada

Cada ítem está centrado horizontalmente en su celda:

| Ítem | Ítem | Ítem |

`align-items`

La propiedad `align-items` alinea los elementos hijos dentro de sus celdas a lo largo del eje vertical (arriba a abajo). Los valores posibles son:

- `start` (arriba)
- `end` (abajo)
- `center` (centro)
- `stretch` (ocupa toda la altura disponible)

Ejemplo de Código

```
.container {
    display: grid;
    grid-template-rows: repeat(3, 100px);
    align-items: end;
}
```

Visualización Actualizada

Cada ítem está alineado al fondo de su celda:

```
-------------

    Ítem

-------------

    Ítem

-------------

    Ítem

-------------
```

`justify-content`

La propiedad `justify-content` alinea y distribuye las columnas de la grilla a lo largo del eje horizontal del contenedor. Los valores posibles son:

- `start` (inicio)
- `end` (fin)
- `center` (centro)
- `space-between` (espacio entre columnas)
- `space-around` (espacio alrededor de columnas)
- `space-evenly` (espacios iguales entre y alrededor de columnas)

Ejemplo de Código

```
.container {

    display: grid;

    grid-template-columns: repeat(3, 1fr);

    justify-content: space-between;

}
```

Visualización Actualizada

Las columnas están distribuidas con espacio entre ellas:

|Ítem| |Ítem| |Ítem|

`align-content`

La propiedad `align-content` alinea y distribuye las filas de la grilla a lo largo del eje vertical del contenedor. Solo tiene efecto cuando el contenedor de la grilla tiene un espacio mayor al necesario para las filas. Los valores posibles son:

- `start` (arriba)
- `end` (abajo)
- `center` (centro)
- `space-between` (espacio entre filas)

- `space-around` (espacio alrededor de filas)
- `space-evenly` (espacios iguales entre y alrededor de filas)

Ejemplo de Código

```
.container {
    display: grid;
    grid-template-rows: repeat(3, 100px);
    align-content: space-evenly;
}
```

Visualización Actualizada

Las filas están distribuidas con espacios iguales entre y alrededor:

Fila 1

(space)

Fila 2

(space)

Fila 3

Ejemplo Completo

Combinemos varias propiedades para mostrar cómo afectan la disposición de la grilla.

Ejemplo de Código

```
.container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(2, 100px);
    justify-items: center;
```

```
    align-items: start;

    justify-content: space-around;

    align-content: space-between;

}
```

Visualización Actualizada

El resultado es una grilla organizada de la siguiente forma:

```
  Columna 1      Columna 2      Columna 3

-----------------------------------------

  Ítem           Ítem           Ítem

-----------------------------------------

                 (space)

  Ítem           Ítem           Ítem

-----------------------------------------
```

En este ejemplo:

- `justify-items: center` alinea los ítems horizontalmente al centro de sus celdas.
- `align-items: start` alinea los ítems verticalmente al inicio de sus celdas.
- `justify-content: space-around` distribuye las columnas con espacio alrededor.
- `align-content: space-between` distribuye las filas con espacio entre ellas.

Introducción al Diseño Responsive y su Importancia en el Desarrollo Web

El diseño responsive es un enfoque en el desarrollo web que garantiza que los sitios web se vean y funcionen bien en una amplia variedad de dispositivos y tamaños de pantalla, desde teléfonos móviles hasta computadoras de escritorio. La importancia del diseño responsive radica en la creciente diversidad de dispositivos utilizados para acceder a la web, lo que hace esencial que los sitios sean accesibles y fáciles de usar en cualquier dispositivo.

Importancia del Diseño Responsive

- **Mejora la Experiencia del Usuario**: Un sitio web que se adapta a diferentes tamaños de pantalla proporciona una mejor experiencia de usuario, lo que puede aumentar el tiempo de permanencia y la satisfacción del usuario.
- **Optimización para Motores de Búsqueda**: Google y otros motores de búsqueda favorecen los sitios web que son móviles y responsivos, lo que puede mejorar el ranking en los resultados de búsqueda.
- **Mayor Alcance**: Un diseño responsive garantiza que el sitio sea accesible para una mayor audiencia, incluidos los usuarios de dispositivos móviles, tabletas y computadoras de escritorio.
- **Reducción de Costos de Desarrollo**: En lugar de crear y mantener múltiples versiones de un sitio para diferentes dispositivos, un diseño responsive permite gestionar un solo sitio que se adapta a todos los dispositivos.

Uso de Media Queries en CSS

Las *media queries* son una característica de CSS que permiten aplicar estilos específicos en función de las características del dispositivo, como el ancho y la altura de la pantalla. A continuación, se presentan ejemplos de cómo utilizar *media queries* para adaptar el diseño a diferentes tamaños de pantalla.

Si acaso te estás preguntando las medidas exactas para el responsive, la respuesta es que no hay. Ya que son estándares que se llevan y pueden variar. De igual forma, a medida que vayas practicando vas a notar que si trabajas bien flex, grid, medidas relativas, etc., el diseño se va a adaptar de una forma excelente.

Ejemplo Básico de Media Query

```
/* Estilos para dispositivos móviles */

body {

    font-size: 16px;

    padding: 10px;

}

/* Media query para tabletas */

@media (min-width: 600px) {

    body {

        font-size: 18px;

        padding: 20px;
```

```
    }
}

/* Media query para computadoras de escritorio */
@media (min-width: 1024px) {
    body {
        font-size: 20px;
        padding: 30px;
    }
}
```

Escenarios de Diseño Responsive

1. Ajuste de la Barra de Navegación

En este ejemplo, ajustamos el diseño de una barra de navegación para que sea más adecuada en dispositivos móviles y de escritorio.

```
/* Estilos para dispositivos móviles */
.navbar {
    display: flex;
    flex-direction: column;
    align-items: center;
}

/* Media query para tabletas y computadoras de escritorio */
@media (min-width: 768px) {
    .navbar {
        flex-direction: row;
        justify-content: space-between;
```

```
    }
}
```

2. Reorganización del Contenido

Ajustamos la disposición del contenido para que se muestre en una sola columna en dispositivos móviles y en dos columnas en pantallas más grandes.

```
/* Estilos para dispositivos móviles */
.container {
    display: flex;
    flex-direction: column;
}
/* Media query para tabletas y computadoras de escritorio */
@media (min-width: 768px) {
    .container {
        flex-direction: row;
        justify-content: space-between;
    }
}
```

3. Ajuste de Imágenes Responsivas

Las imágenes también deben ajustarse para adaptarse a diferentes tamaños de pantalla.

```
/* Estilos para dispositivos móviles */
img {
    width: 100%;
    height: auto;
```

```
}

/* Media query para computadoras de escritorio */

@media (min-width: 1024px) {

    img {

        width: 50%;

    }

}
```

Ejemplo Completo

A continuación, se presenta un ejemplo completo que combina varios de estos conceptos para crear un diseño responsive.

```
<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Ejemplo de Diseño Responsive</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 0;

            padding: 10px;

        }

        .navbar {

            display: flex;
```

```
    flex-direction: column;
    align-items: center;
    background-color: #333;
    color: white;
    padding: 10px;
}
.navbar a {
    color: white;
    padding: 10px;
    text-decoration: none;
}
.container {
    display: flex;
    flex-direction: column;
    margin-top: 20px;
}
img {
    width: 100%;
    height: auto;
}
/* Media query para tabletas */
@media (min-width: 600px) {
    body {
        font-size: 18px;
```

```
            padding: 20px;
        }
        .navbar {
            flex-direction: row;
            justify-content: space-between;
        }
        .container {
            flex-direction: row;
            justify-content: space-between;
        }
    }
    /* Media query para computadoras de escritorio */
    @media (min-width: 1024px) {
        body {
            font-size: 20px;
            padding: 30px;
        }
        img {
            width: 50%;
        }
    }
  </style>
</head>
<body>
```

```
    <div class="navbar">
        <a href="#">Inicio</a>
        <a href="#">Acerca de</a>
        <a href="#">Contacto</a>
    </div>
    <div class="container">
        <div class="content">
            <h1>Bienvenidos a nuestro sitio</h1>
            <p>Este es un ejemplo de diseño responsive que se adapta a diferentes tamaños de pantalla.</p>
        </div>
        <img src="https://via.placeholder.com/800" alt="Ejemplo de imagen">
    </div>
</body>
</html>
```

Con este enfoque, el sitio web se adapta adecuadamente a diferentes dispositivos, proporcionando una experiencia de usuario consistente y optimizada.

Enfoque de Desarrollo Mobile First

El enfoque de desarrollo *mobile first* implica diseñar y desarrollar un sitio web inicialmente para dispositivos móviles y luego adaptarlo progresivamente para pantallas más grandes como tabletas y computadoras de escritorio. Este enfoque garantiza que el contenido y la funcionalidad principal estén accesibles y optimizados para los usuarios móviles, que representan una gran parte del tráfico web actual.

Pasos para Implementar un Diseño Mobile First

1. Planificación y Estructura

Antes de comenzar con el código, planifica el contenido y la estructura del sitio. Define qué elementos son esenciales y deben ser prioritarios en la versión móvil.

2. Escribir el HTML

Crea el marcado HTML con una estructura semántica. Asegúrate de que el HTML sea claro y accesible, y que todos los elementos importantes estén presentes sin estilos específicos.

```
<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Ejemplo Mobile First</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Título del Sitio</h1>

    </header>

    <main>

        <section>

            <h2>Sección Principal</h2>

            <p>Contenido relevante para dispositivos móviles.</p>

        </section>

    </main>
```

```
    <footer>
        <p>Pie de página</p>
    </footer>
</body>
</html>
```

3. Estilos Básicos para Móviles

Escribe los estilos CSS comenzando por los dispositivos móviles. Utiliza una hoja de estilos base que incluya las reglas fundamentales para pantallas pequeñas.

```
/* styles.css */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}
header, main, footer {
    padding: 10px;
    text-align: center;
}
h1 {
    font-size: 1.5em;
}
h2 {
    font-size: 1.2em;
}
```

```
p {
    font-size: 1em;
}
```

4. Uso de Media Queries

A medida que el diseño se ajusta para pantallas más grandes, utiliza *media queries* para aplicar estilos adicionales y reorganizar el contenido según sea necesario.

Ejemplo de Media Queries

```
/* Media query para tabletas */
@media (min-width: 600px) {
    body {
        padding: 20px;
    }
    header, main, footer {
        padding: 20px;
        text-align: left;
    }
    h1 {
        font-size: 2em;
    }
    h2 {
        font-size: 1.5em;
    }
    p {
        font-size: 1.2em;
```

```
    }
}
/* Media query para computadoras de escritorio */
@media (min-width: 1024px) {
    body {
        max-width: 1200px;
        margin: 0 auto;
    }
    header, main, footer {
        display: flex;
        justify-content: space-between;
        padding: 30px;
    }
    h1 {
        font-size: 2.5em;
    }
    h2 {
        font-size: 2em;
    }
    p {
        font-size: 1.5em;
    }
}
```

5. Pruebas y Optimización

Prueba el diseño en varios dispositivos y tamaños de pantalla para asegurarte de que se vea bien y funcione correctamente en todos ellos. Optimiza el rendimiento para asegurar tiempos de carga rápidos, especialmente en dispositivos móviles.

Ejemplo Completo

HTML

```
<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Ejemplo Mobile First</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <h1>Título del Sitio</h1>

    </header>

    <main>

        <section>

            <h2>Sección Principal</h2>

            <p>Contenido relevante para dispositivos móviles.</p>

        </section>

    </main>

    <footer>
```

```
        <p>Pie de página</p>
    </footer>
</body>
</html>
```

CSS

```
/* styles.css */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}
header, main, footer {
    padding: 10px;
    text-align: center;
}
h1 {
    font-size: 1.5em;
}
h2 {
    font-size: 1.2em;
}
p {
    font-size: 1em;
```

```
}
/* Media query para tabletas */
@media (min-width: 600px) {
    body {
        padding: 20px;
    }
    header, main, footer {
        padding: 20px;
        text-align: left;
    }
    h1 {
        font-size: 2em;
    }
    h2 {
        font-size: 1.5em;
    }
    p {
        font-size: 1.2em;
    }
}
/* Media query para computadoras de escritorio */
@media (min-width: 1024px) {
    body {
        max-width: 1200px;
```

```
        margin: 0 auto;
    }
    header, main, footer {
        display: flex;
        justify-content: space-between;
        padding: 30px;
    }
    h1 {
        font-size: 2.5em;
    }
    h2 {
        font-size: 2em;
    }
    p {
        font-size: 1.5em;
    }
}
```

Con este enfoque, garantizas que el sitio web se vea y funcione bien en todos los dispositivos, mejorando la experiencia del usuario y asegurando accesibilidad y usabilidad óptimas.

Mobile First con Flexbox

Paso 1: Estructura Base en HTML

El HTML debe contener una estructura limpia y semántica.

```
<!DOCTYPE html>
```

```
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Mobile First</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Mi Sitio Web</h1>
        <nav>
            <ul>
                <li><a href="#">Inicio</a></li>
                <li><a href="#">Servicios</a></li>
                <li><a href="#">Contacto</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section>
            <h2>Bienvenido</h2>
            <p>Este es un ejemplo práctico de Mobile First.</p>
        </section>
    </main>
```

```
    <footer>
        <p>&copy; 2024 Mi Sitio Web</p>
    </footer>
</body>
</html>
```

Paso 2: Estilos Base para Móviles

Define estilos CSS optimizados para pantallas pequeñas.

```
/* styles.css */
/* Estilos generales */
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    text-align: center;
}
header, main, footer {
    padding: 10px;
}
nav ul {
    list-style: none;
    padding: 0;
}
nav ul li {
```

```
    margin: 5px 0;
}
nav ul li a {
    text-decoration: none;
    color: #007BFF;
}
```

Paso 3: Adaptación para Pantallas Más Grandes

Usa *media queries* para añadir estilos adicionales para tabletas y computadoras de escritorio.

```
/* Media query para tabletas */
@media (min-width: 600px) {
    nav ul {
        display: flex;
        justify-content: center;
        gap: 15px;
    }
}
/* Media query para computadoras de escritorio */
@media (min-width: 1024px) {
    body {
        max-width: 1200px;
        margin: 0 auto;
    }
    header, footer {
```

```
        display: flex;
        justify-content: space-between;
        align-items: center;
    }
    nav ul {
        justify-content: flex-end;
    }
}
```

Ejemplo en Acción

1. **Versión Móvil:** Los elementos están centrados y ocupan todo el ancho de la pantalla.
2. **Versión Tableta:** El menú de navegación se muestra como una fila alineada al centro.
3. **Versión Escritorio:** El diseño se adapta, maximizando el espacio horizontal con un menú alineado a la derecha.