**7. LAYOUT CON GRID**

CSS Grid Layout es una herramienta poderosa para crear diseños de página web complejos y responsivos. A diferencia de Flexbox, que se centra en la distribución de elementos en un solo eje (horizontal o vertical), CSS Grid permite trabajar en dos ejes (horizontal y vertical) simultáneamente, facilitando la creación de estructuras de diseño más avanzadas.

**7.1. Conceptos Básicos de CSS Grid**

* **Grid Container**: El elemento que contiene el grid. Se define con display: grid.
* **Grid Items**: Los elementos hijos directos del grid container. Estos son los elementos que se organizan en el grid.

**7.2. Propiedades del Grid Container**

* **grid-template-columns** y **grid-template-rows**: Definen el número y tamaño de las columnas y filas.

**CSS**

.container {

display: grid;

grid-template-columns: repeat(3, 1fr); /\* Tres columnas de igual tamaño \*/

grid-template-rows: auto; /\* Filas automáticas \*/

}

* **gap**: Define el espacio entre las filas y columnas.

**CSS**

.container {

gap: 10px; /\* Espacio de 10px entre filas y columnas \*/

}

* **grid-template-areas**: Permite nombrar áreas del grid para facilitar la organización.

**CSS**

.container {

grid-template-areas:

"header header header"

"sidebar main main"

"footer footer footer";

}

**7.3. Propiedades de los Grid Items**

* **grid-column** y **grid-row**: Permiten que un elemento ocupe varias columnas o filas.

**CSS**

.item1 {

grid-column: 1 / 3; /\* Ocupa las dos primeras columnas \*/

grid-row: 1; /\* Primera fila \*/

}

* **grid-area**: Asigna un área nombrada a un elemento.

**CSS**

.header {

grid-area: header;

}

**7.4. Ejemplo Completo de Layout con Grid**

**CSS**

/\* Contenedor Grid \*/

.container {

display: grid;

grid-template-columns: 1fr 2fr 1fr;

grid-template-rows: auto;

grid-template-areas:

"header header header"

"sidebar main main"

"footer footer footer";

gap: 10px;

height: 100vh; /\* Altura del contenedor \*/

}

/\* Elementos Grid \*/

.header {

grid-area: header;

background-color: lightblue;

padding: 20px;

text-align: center;

}

.sidebar {

grid-area: sidebar;

background-color: lightgreen;

padding: 20px;

}

.main {

grid-area: main;

background-color: lightcoral;

padding: 20px;

}

.footer {

grid-area: footer;

background-color: lightgoldenrodyellow;

padding: 20px;

text-align: center;

}

**---------------------------------------------------------------------------------------------**

**HTML**

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

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

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

<title>Layout con Grid</title>

</head>

<body>

<div class="container">

<div class="header">Header</div>

<div class="sidebar">Sidebar</div>

<div class="main">Main Content</div>

<div class="footer">Footer</div>

</div>

</body>

</html>

**7.5 Explicación del Ejemplo**

En este ejemplo, el contenedor .container utiliza display: grid para activar el modo Grid. Se definen tres columnas (grid-template-columns: 1fr 2fr 1fr) y se nombran áreas específicas (grid-template-areas) para organizar el diseño. Los elementos hijos .header, .sidebar, .main, y .footer se asignan a estas áreas utilizando la propiedad grid-area.

* **Header**: Ocupa toda la primera fila.
* **Sidebar**: Ocupa la primera columna de la segunda fila.
* **Main**: Ocupa las dos últimas columnas de la segunda fila.
* **Footer**: Ocupa toda la tercera fila.

**7.6 Resumen**

CSS Grid es una herramienta extremadamente flexible y poderosa para crear layouts complejos y responsivos. Permite definir tanto filas como columnas, y organizar los elementos de manera precisa utilizando áreas nombradas. Esto facilita la creación de diseños avanzados y adaptativos, mejorando la experiencia del usuario en diferentes dispositivos y tamaños de pantalla.