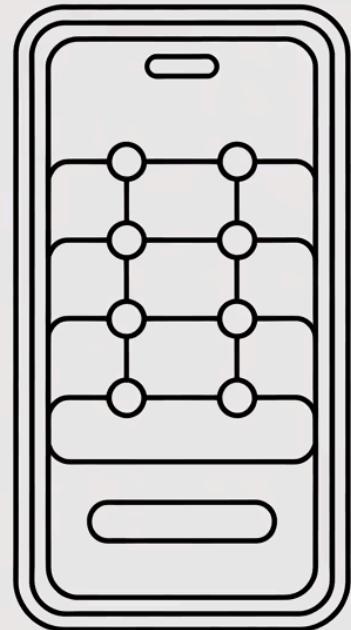


Introducción al Desarrollo web

Curso: Desarrollo Frontend con React



Connect the dots



Layouts en CSS con Flexbox y Grid

Agenda



Fundamentos de maquetación web

Introducción



Flexbox

Propiedades y usos principales



Grid

Sistema de maquetación bidimensional



Comparativa y casos prácticos

Cuándo utilizar cada sistema



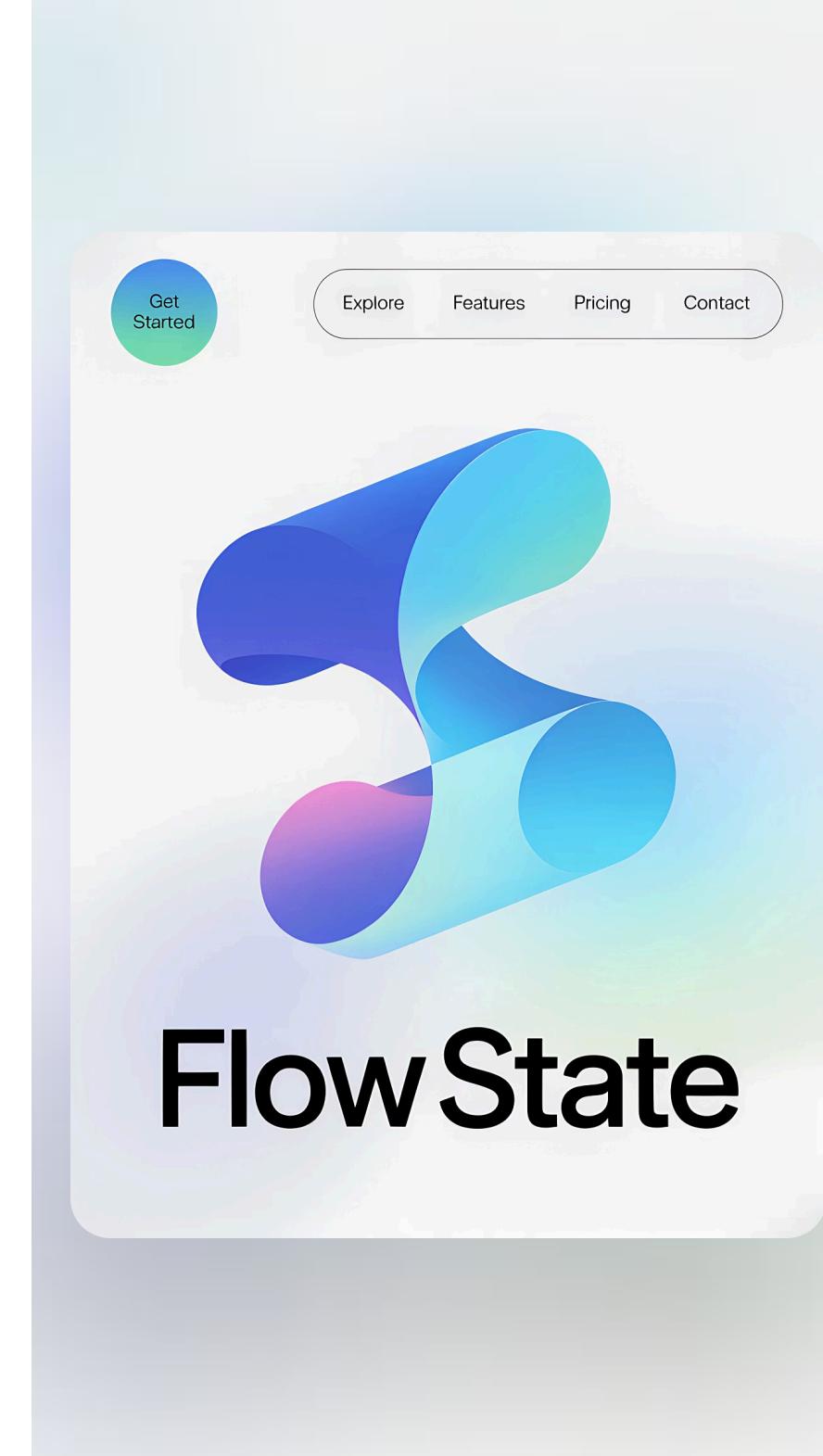
Diseño responsive

Adaptación a diferentes dispositivos



Buenas prácticas

Consejos y errores comunes

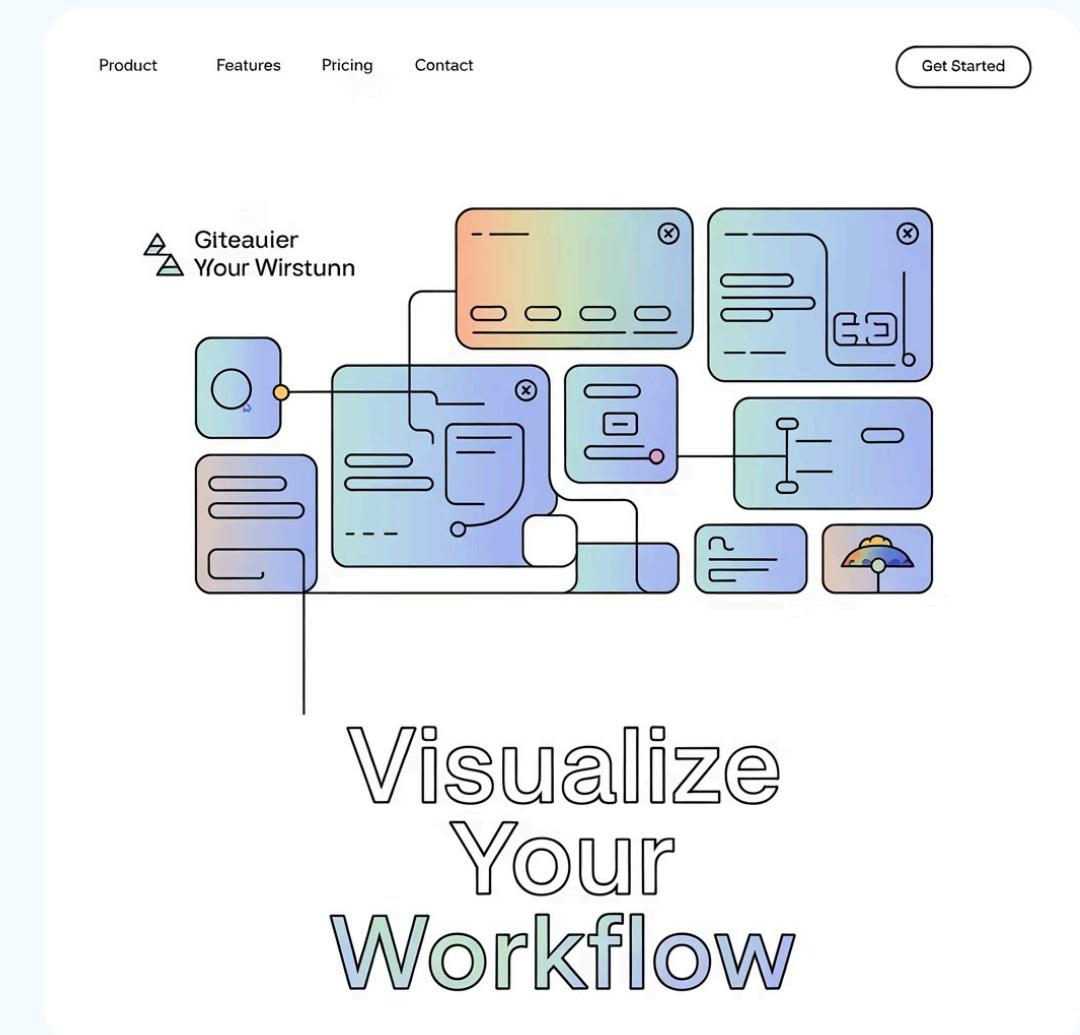


¿Qué es la maquetación web?

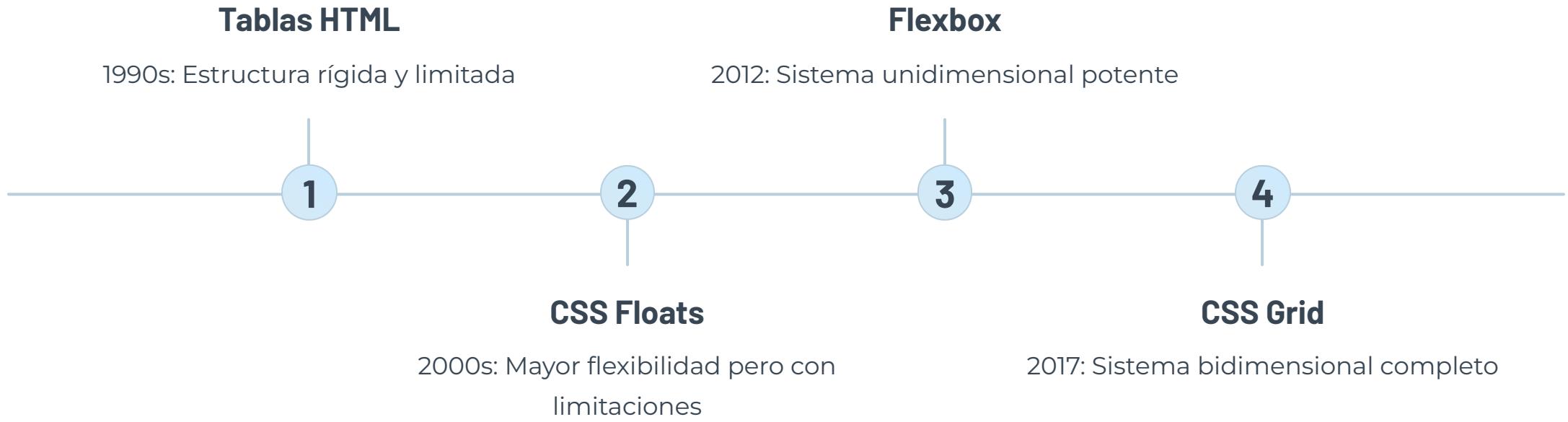
La maquetación es el proceso de **organizar elementos** visuales en una página web para crear una experiencia atractiva y funcional.

Es como la [arquitectura digital](#) que permite:

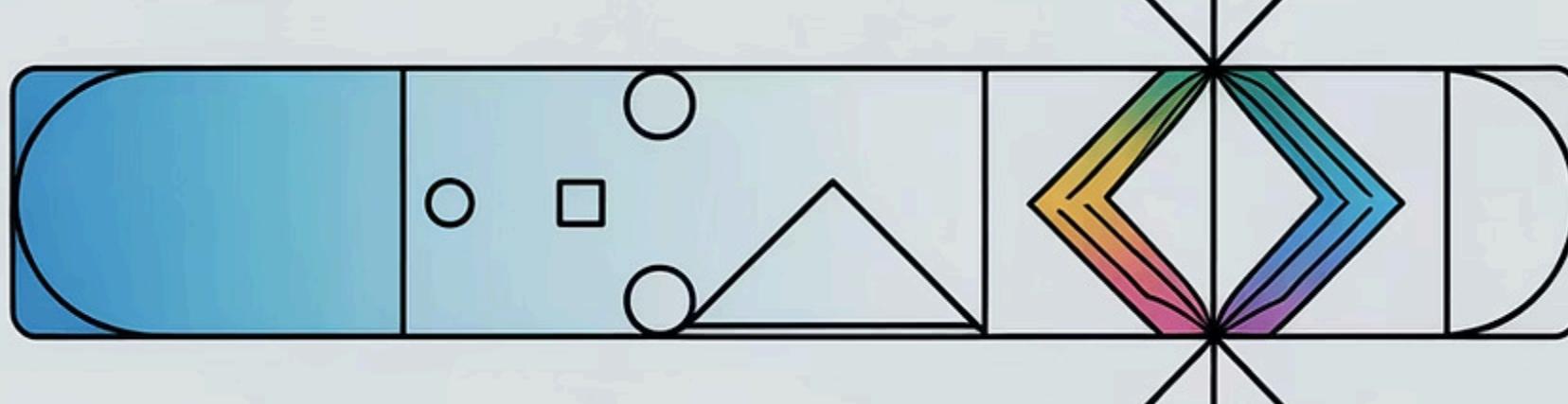
- Distribuir contenido de forma coherente
- Crear jerarquías visuales
- Facilitar la navegación
- Adaptar el diseño a diferentes dispositivos



Evolución de la maquetación web



Hemos pasado de soluciones improvisadas a [sistemas robustos de maquetación](#) específicamente diseñados para el entorno web.



Introducción a Flexbox

Flexbox es un sistema de maquetación **unidimensional** diseñado para distribuir elementos en una sola dirección (fila o columna).

Ideal para componentes de interfaz y patrones de diseño pequeños o medianos:

- Menús de navegación
- Barras de herramientas
- Tarjetas de contenido
- Formularios

Terminología de Flexbox

Contenedor Flex (Flex Container)

Elemento padre con `display: flex`

Elementos Flex (Flex Items)

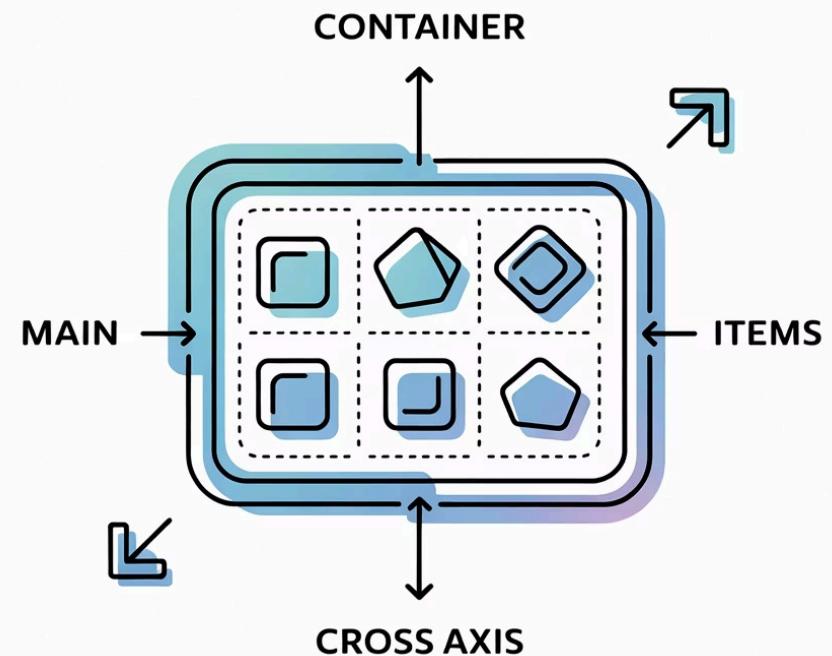
Hijos directos del contenedor flex

Eje Principal (Main Axis)

Dirección primaria de distribución

Eje Secundario (Cross Axis)

Dirección perpendicular al eje principal



Propiedades principales de Flexbox

display: flex

Define un contenedor flexible

```
.contenedor {  
  display: flex;  
}
```

flex-direction

Establece la dirección del eje principal

```
.contenedor {  
  flex-direction: row | row-reverse |  
  column | column-reverse;  
}
```

justify-content

Alineación en el eje principal

```
.contenedor {  
  justify-content: flex-start | flex-end |  
  center | space-between | space-around;  
}
```

align-items

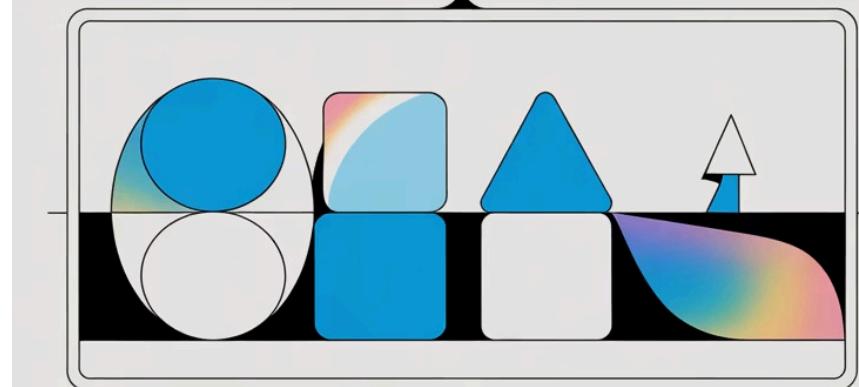
Alineación en el eje secundario

```
.contenedor {  
  align-items: flex-start | flex-end |  
  center | stretch | baseline;  
}
```

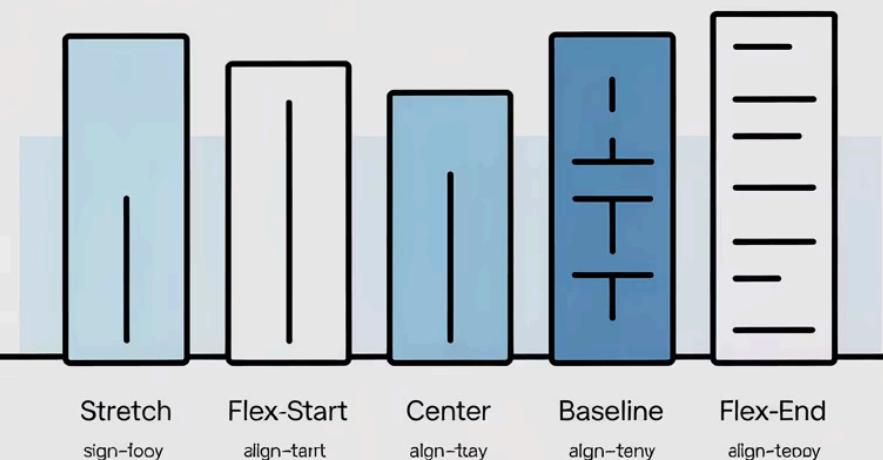
Flexbox: justify-content

Controla la alineación de los elementos en el [eje principal](#):

- **flex-start:** Elementos al inicio
- **flex-end:** Elementos al final
- **center:** Elementos centrados
- **space-between:** Espacio entre elementos
- **space-around:** Espacio alrededor de elementos
- **space-evenly:** Espacio uniforme



Align-Items



Flexbox: align-items

Controla la alineación de los elementos en el [eje secundario](#):

- **flex-start**: Elementos al inicio
- **flex-end**: Elementos al final
- **center**: Elementos centrados
- **stretch**: Estirados para ocupar todo el espacio (default)
- **baseline**: Alineados por la línea base del texto

Propiedades de los elementos flex

flex-grow

Factor de crecimiento

```
.item {  
  flex-grow: 0; /* No crece */  
  flex-grow: 1; /* Crece */  
  flex-grow: 2; /* Crece el doble */  
}
```

flex-basis

Tamaño inicial

```
.item {  
  flex-basis: auto; /* Tamaño natural */  
  flex-basis: 0; /* Sin tamaño inicial */  
  flex-basis: 200px; /* Tamaño fijo */  
}
```

flex-shrink

Factor de reducción

```
.item {  
  flex-shrink: 1; /* Puede reducirse */  
  flex-shrink: 0; /* No se reduce */  
}
```

flex (atajo)

Combina grow, shrink y basis

```
.item {  
  flex: 0 1 auto; /* Default */  
  flex: 1; /* flex: 1 1 0% */  
  flex: auto; /* flex: 1 1 auto */  
}
```

Ejercicio Práctico: Creando una Tarjeta de Producto con Flexbox

En este ejercicio práctico, te guiaremos a través de la creación de una tarjeta de producto responsive utilizando Flexbox. Aprenderás a organizar y distribuir elementos de manera flexible dentro de un contenedor, aplicando las propiedades clave de Flexbox.

El ejercicio se desarrollará paso a paso en los siguientes módulos, donde cubriremos desde la estructura HTML básica hasta la aplicación avanzada de estilos con Flexbox para lograr un diseño dinámico y adaptable.

Paso 1: Estructura HTML Básica

En este primer paso, crearemos la estructura HTML para nuestra tarjeta de producto. Esta incluirá todos los elementos necesarios: imagen, título, descripción, precio y botón.

Crea un archivo index.html y añade la siguiente estructura dentro del `<body>`:

```
<div class="product-card">
  
  <div class="product-details">
    <h3 class="product-title">Auriculares Inalámbricos</h3>
    <p class="product-description">Disfruta de un sonido nítido y llamadas manos libres con estos auriculares de alta calidad.</p>
  </div>
  <div class="product-actions">
    <span class="product-price">$59.99</span>
    <button class="add-to-cart">Añadir al Carrito</button>
  </div>
</div>
```

Conceptos HTML aplicados:

- Contenedor principal (.product-card) que agrupará todos los elementos
- Imagen del producto con atributo alt para accesibilidad
- Div para agrupar los detalles del producto (título y descripción)
- Div separado para las acciones (precio y botón de compra)
- Uso de clases semánticas para facilitar el estilado posterior

Paso 2: Estilos Base y Aplicación de Flexbox

Ahora daremos estilo a la tarjeta y la convertiremos en un contenedor Flexbox. Queremos que los elementos internos se apilen verticalmente.

Crea un archivo `style.css` y enlázalo a tu HTML. Añade los siguientes estilos:

```
.product-card {  
    width: 300px;  
    border: 1px solid #ddd;  
    border-radius: 8px;  
    overflow: hidden;  
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
    padding: 15px;  
    display: flex; /* ¡Aquí está Flexbox! */  
    flex-direction: column; /* Apila los elementos verticalmente */  
    align-items: center; /* Centra los elementos horizontalmente */  
    gap: 15px; /* Espacio entre los elementos flex */  
}  
  
.product-image {  
    max-width: 100%;  
    height: auto;  
    border-radius: 4px;  
}  
  
.product-details {  
    text-align: center;  
}  
  
.product-title {  
    margin-top: 0;  
    margin-bottom: 5px;  
    color: #333;  
}  
  
.product-description {  
    font-size: 0.9em;  
    color: #666;  
}
```

Conceptos de Flexbox aplicados:

- `display: flex` convierte `.product-card` en un contenedor flex
- `flex-direction: column` organiza los hijos directos en una columna
- `align-items: center` centra los elementos en el eje transversal
- `gap` añade espacio uniforme entre los elementos flex

Paso 3: Distribuyendo las Acciones con Flexbox

En este último paso, usaremos Flexbox dentro del contenedor `.product-actions` para alinear el precio y el botón horizontalmente.

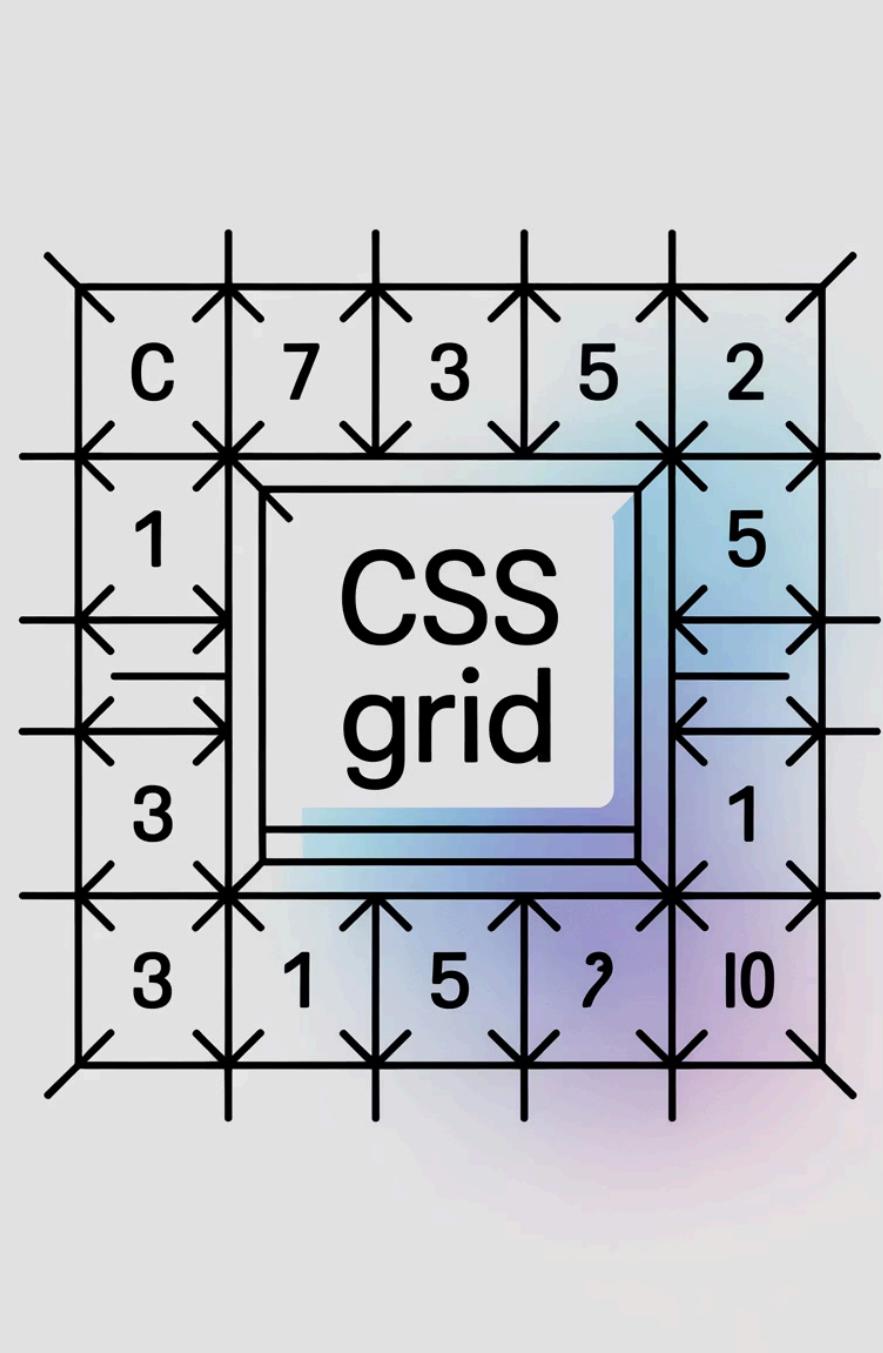
Añade estos estilos a tu archivo `style.css`:

```
.product-actions {  
    width: 100%;  
    display: flex; /* Otro contenedor flex */  
    justify-content: space-between; /* Distribuye el espacio entre los elementos */  
    align-items: center; /* Alinea verticalmente el precio y el botón */  
}  
  
.product-price {  
    font-size: 1.2em;  
    font-weight: bold;  
    color: #007bff;  
}  
  
.add-to-cart {  
    background-color: #28a745;  
    color: white;  
    border: none;  
    padding: 10px 15px;  
    border-radius: 5px;  
    cursor: pointer;  
    font-size: 1em;  
}  
  
.add-to-cart:hover {  
    background-color: #218838;  
}
```

Conceptos de Flexbox aplicados:

- `display: flex` convierte `.product-actions` en un contenedor flex
- `justify-content: space-between` separa el precio y el botón a los extremos
- `align-items: center` alinea verticalmente ambos elementos en el centro
- `width: 100%` asegura que el contenedor ocupe todo el ancho disponible

¡Felicitaciones! Has creado una tarjeta de producto responsive usando Flexbox. Experimenta cambiando los valores de las propiedades para ver cómo afectan el diseño.

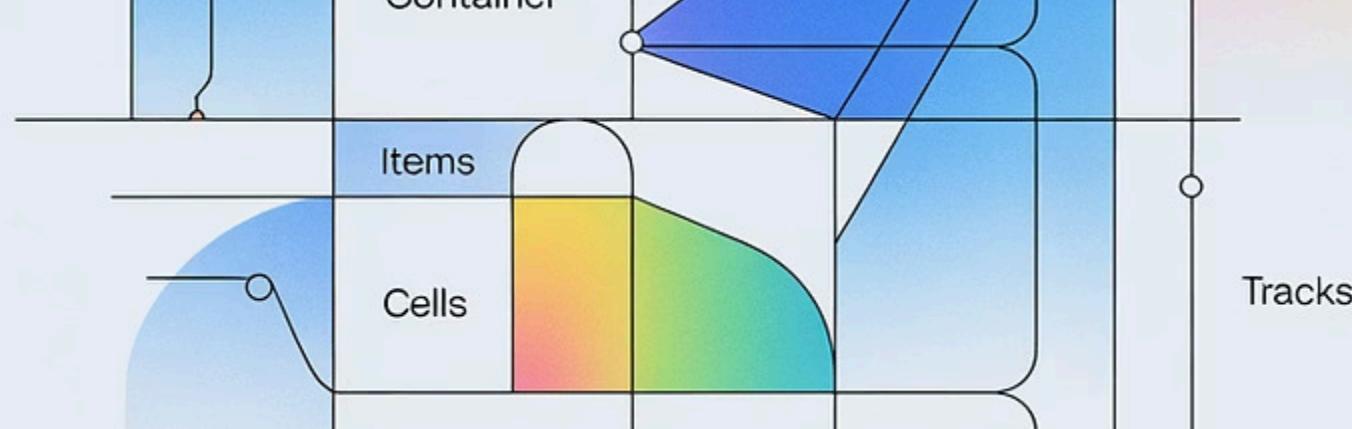


Introducción a CSS Grid

CSS Grid es un sistema de maquetación **bidimensional** diseñado para crear layouts complejos con filas y columnas.

Ideal para:

- Layouts completos de páginas
- Galerías y sistemas de tarjetas
- Diseños asimétricos
- Solapamiento de elementos



Terminología de Grid

Contenedor Grid

Elemento con `display: grid`

Elementos Grid

Hijos directos del contenedor

Líneas Grid

Líneas divisorias horizontales y verticales

Pista (Track)

Filas o columnas entre dos líneas

Celda

Intersección entre fila y columna

Área

Conjunto rectangular de celdas

Gap

Espacio entre filas y columnas

Grid explícito e implícito

Definido vs. generado automáticamente

Propiedades principales de Grid

display: grid

Define un contenedor grid

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

grid-template-columns

Define columnas del grid

```
.contenedor {  
    grid-template-columns: 100px 200px auto;  
    /* O usando fr: */  
    grid-template-columns: 1fr 2fr 1fr;  
}
```

grid-template-rows

Define filas del grid

```
.contenedor {  
    grid-template-rows: auto 200px 100px;  
    /* O usando fr: */  
    grid-template-rows: 1fr 2fr;  
}
```

gap

Espacio entre filas y columnas

```
.contenedor {  
    column-gap: 10px;  
    row-gap: 15px;  
    /* O simplificado: */  
    gap: 15px 10px; /* filas columnas */  
}
```

La unidad fr en Grid

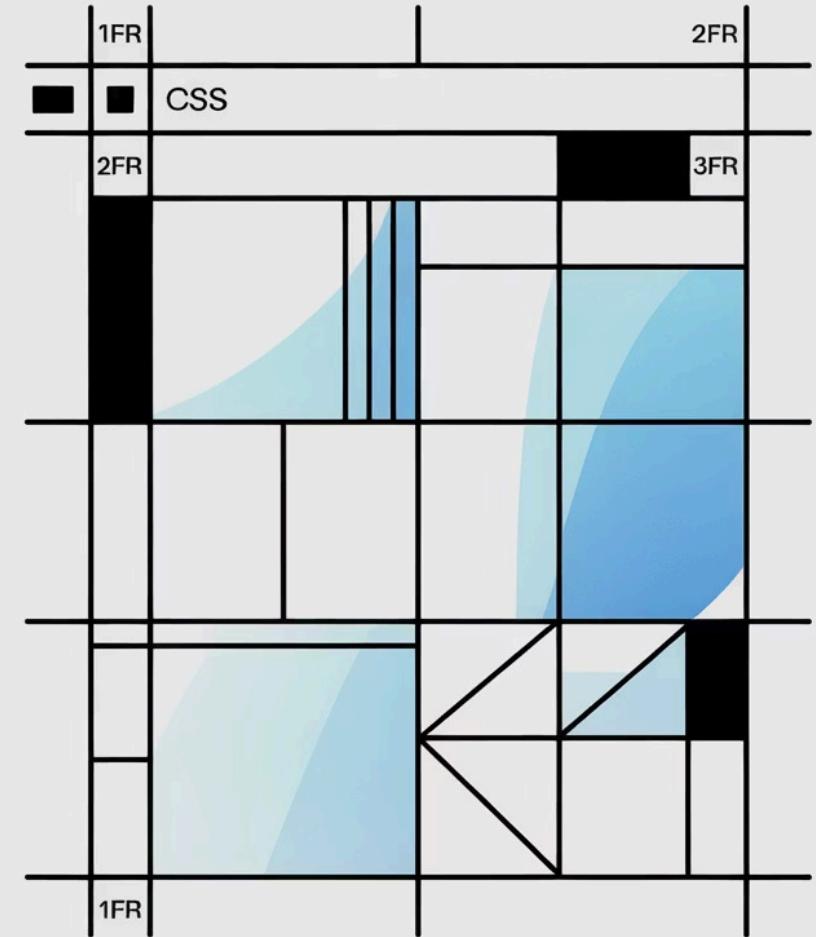
La unidad `fr` (fracción) distribuye el espacio disponible proporcionalmente.

Ejemplos de uso:

```
/* Tres columnas iguales */  
grid-template-columns: 1fr 1fr  
1fr;  
  
/* O más simple: */  
grid-template-columns: repeat(3,  
1fr);  
  
/* Columna central el doble de  
ancha */  
grid-template-columns: 1fr 2fr  
1fr;
```

Combinando fr con otras unidades:

```
/* Primera columna fija,  
segunda flexible */  
grid-template-columns: 200px  
1fr;  
  
/* Tres columnas, mínimo  
200px,  
distribución proporcional */  
grid-template-columns:  
minmax(200px, 1fr)  
minmax(200px, 2fr)  
minmax(200px, 1fr);
```



Funciones útiles de Grid

repeat()

Repite patrones de pistas

```
/* 4 columnas de 1fr */  
grid-template-columns: repeat(4, 1fr);  
  
/* Patrón repetido */  
grid-template-columns: repeat(3, 100px 1fr);  
/* Equivale a: 100px 1fr 100px 1fr 100px 1fr */
```

minmax()

Define tamaño mínimo y máximo

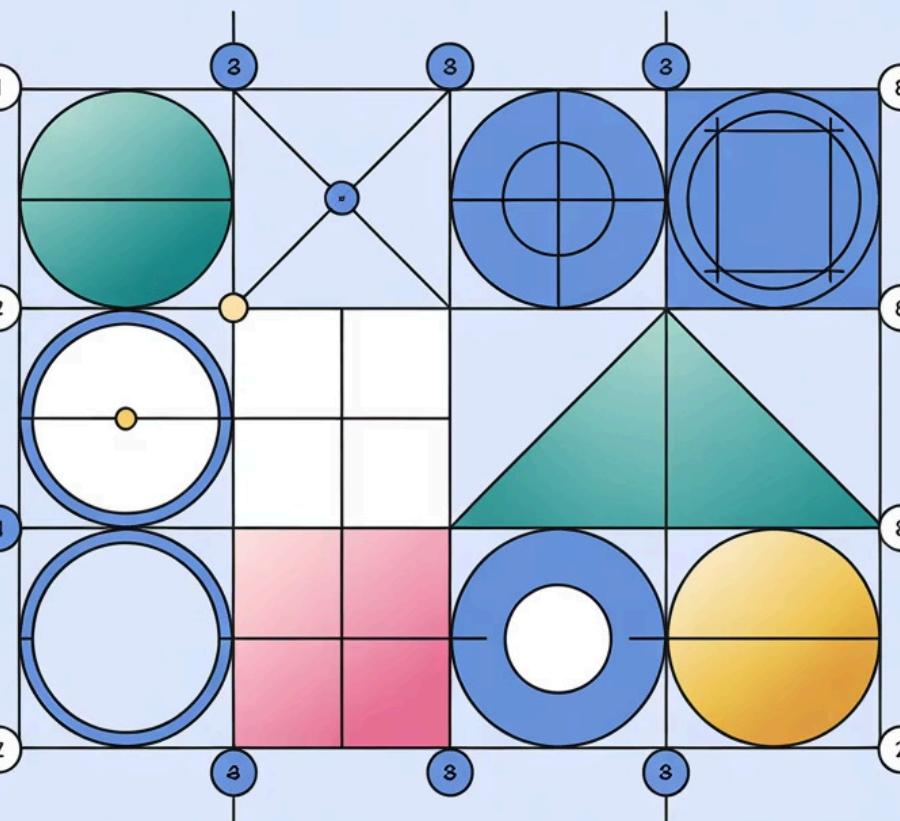
```
/* Mínimo 100px, máximo 1fr */  
grid-template-columns: minmax(100px, 1fr) 3fr;  
  
/* Filas automáticas mínimo 100px */  
grid-auto-rows: minmax(100px, auto);
```

auto-fill y auto-fit

```
/* Tantas columnas de 200px como quepan */  
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));  
  
/* Como auto-fill pero colapsando columnas vacías */  
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
```

CSS Grid Layout

Loreum ipsum dolor sit amet, consectetur adipiscing elit. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusamus et iusto odio dignissimos.

[Download Guide](#)

Colocación de elementos en Grid

Por línea:

```
.item {  
    grid-column-start: 1;  
    grid-column-end: 3;  
    grid-row-start: 2;  
    grid-row-end: 4;  
  
    /* O simplificado:  
     * grid-column: 1 / 3; /* inicio / fin  
     */  
    grid-row: 2 / 4; /* inicio / fin */  
  
    /* Con span:  
     * grid-column: 1 / span 2; /*  
     * ocupa 2 celdas */  
}
```

Por áreas nombradas:

```
.grid {  
    grid-template-areas:  
        "header header header"  
        "sidebar content content"  
        "footer footer footer";  
}  
  
.header { grid-area: header; }  
.sidebar { grid-area: sidebar; }  
.content { grid-area: content; }  
.footer { grid-area: footer; }
```

Ejercicio Práctico: Layout de Página Web con CSS Grid

En este ejercicio, te embarcarás en la creación de un layout de página web responsive utilizando las potentes capacidades de CSS Grid. Aprenderás a estructurar elementos clave como la cabecera, la barra lateral, el contenido principal y el pie de página, sentando las bases para diseños web modernos y flexibles. Cada paso detallado de este proceso se desarrollará en los próximos módulos, permitiéndote construir tu propio layout de forma gradual y efectiva.

Paso 1: Estructura HTML Básica

En este primer paso, crearemos la estructura HTML básica para nuestro layout de página web. Definiremos los elementos estructurales principales que luego posicionaremos en el grid.

Crea un archivo index.html y añade la siguiente estructura dentro del <body>:

```
<div class="grid-container">
  <header class="header">Cabecera</header>
  <aside class="sidebar">Barra Lateral</aside>
  <main class="main">Contenido Principal</main>
  <footer class="footer">Pie de Página</footer>
</div>
```

Conceptos HTML aplicados:

- Contenedor principal (.grid-container) que será nuestro contenedor de grid
- Uso de clases descriptivas para facilitar el estilado posterior
- Elementos semánticos: <header>, <aside>, <main>, <footer>
- Estructura lógica que respeta la jerarquía del contenido

Esta estructura HTML define los elementos fundamentales que luego organizaremos usando CSS Grid.

Paso 2: Configurar el Contenedor Grid

En este paso, convertiremos nuestro contenedor en un grid y definiremos la estructura básica de columnas y filas.

Crea un archivo style.css y enlázalo a tu HTML. Dentro de style.css, aplica las propiedades de Grid al .grid-container:

```
.grid-container {  
    display: grid;  
    grid-template-columns: 250px 1fr; /* 250px para sidebar, 1fr para main */  
    grid-template-rows: auto 1fr auto; /* auto para header/footer, 1fr para main/sidebar */  
    gap: 20px; /* Espacio entre las celdas del grid */  
    min-height: 100vh; /* Para que el grid ocupe toda la altura de la ventana */  
}
```

Conceptos de Grid aplicados:

- **display: grid** convierte el div en un contenedor de grid
- **grid-template-columns** define el número y tamaño de las columnas
- **grid-template-rows** define el número y tamaño de las filas
- **gap** añade espacio entre los elementos del grid
- **min-height: 100vh** asegura que el layout ocupe la altura completa del viewport

La unidad **fr** (fracción) distribuye el espacio disponible proporcionalmente, mientras que **auto** se ajusta al contenido.

Paso 3: Definir Áreas con grid-template-areas

En este paso, asignaremos nombres a las áreas de nuestro grid para facilitar la colocación de los elementos de forma visual e intuitiva.

Añade la propiedad grid-template-areas a tu .grid-container:

```
.grid-container {  
    /* ... propiedades anteriores ... */  
    grid-template-areas:  
        "header header"  
        "sidebar main"  
        "footer footer";  
}
```

Conceptos de Grid aplicados:

- **grid-template-areas** permite visualizar y organizar el layout de forma intuitiva
- Las áreas con el mismo nombre se fusionan automáticamente
- "footer footer" significa que el footer ocupa ambas columnas
- Cada cadena de texto representa una fila del grid
- "header header" significa que el header ocupa ambas columnas
- Los nombres dentro de las comillas representan las áreas
- "sidebar main" significa que sidebar ocupa la primera columna y main la segunda

Esta técnica hace que el código sea más legible y fácil de mantener.

Paso 4: Posicionar Elementos Individuales

En este paso final, asignaremos cada elemento HTML a su área correspondiente utilizando la propiedad grid-area.

Añade estos estilos a tu archivo style.css:

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

Para visualizar mejor el layout, también puedes añadir algunos estilos básicos:

```
body {
  margin: 0;
  font-family: sans-serif;
}

.header, .sidebar, .main, .footer {
  padding: 20px;
  border-radius: 8px;
  color: white;
  text-align: center;
}

.header { background-color: #f44444; /* Rojo */ }
.sidebar { background-color: #5E98F1; /* Azul */ }
.main { background-color: #5CC97B; /* Verde */ }
.footer { background-color: #B05EF1; /* Púrpura */ }
```

Conceptos de Grid aplicados:

- **grid-area** conecta los elementos HTML con las áreas definidas en grid-template-areas
- Cada elemento se coloca automáticamente en su área correspondiente
- No necesitas especificar posiciones numéricas, solo el nombre del área
- El layout se adapta automáticamente si cambias las definiciones de las áreas

¡Felicitaciones! Has creado un layout de página web completo utilizando CSS Grid.

Comparativa: Flexbox vs Grid

Flexbox

- **Unidimensional:** filas O columnas
- **Basado en contenido:** se adapta al tamaño de los elementos
- **Control de flujo:** orden, dirección, alineación
- **Distribución:** excelente para distribuir espacio

Ideal para: componentes UI, navegación, elementos en línea

Grid

- **Bidimensional:** filas Y columnas
- **Basado en layout:** define estructura primero
- **Control de ubicación:** áreas, líneas, celdas
- **Precisión:** excelente para layouts complejos

Ideal para: layouts completos, sistemas de cuadrícula, galerías

No compiten entre sí: son [herramientas complementarias](#) para diferentes casos de uso.

¿Cuándo usar cada uno?

Usa Flexbox cuando:

- Necesites organizar elementos en una sola dimensión (fila o columna)
- El tamaño de los elementos deba adaptarse a su contenido
- Trabajes con componentes como menús, barras de herramientas o tarjetas
- Necesites alinear o distribuir elementos fácilmente

Usa Grid cuando:

- Necesites crear una estructura bidimensional de filas y columnas
- Necesites precisión en la colocación de elementos
- Trabajes con layouts complejos de páginas completas
- Requieras que elementos ocupen múltiples filas o columnas
- Desees crear sistemas de cuadrícula o rejillas

Y recuerda: ¡puedes usar ambos, juntos! Grid para el layout general y Flexbox para componentes específicos.

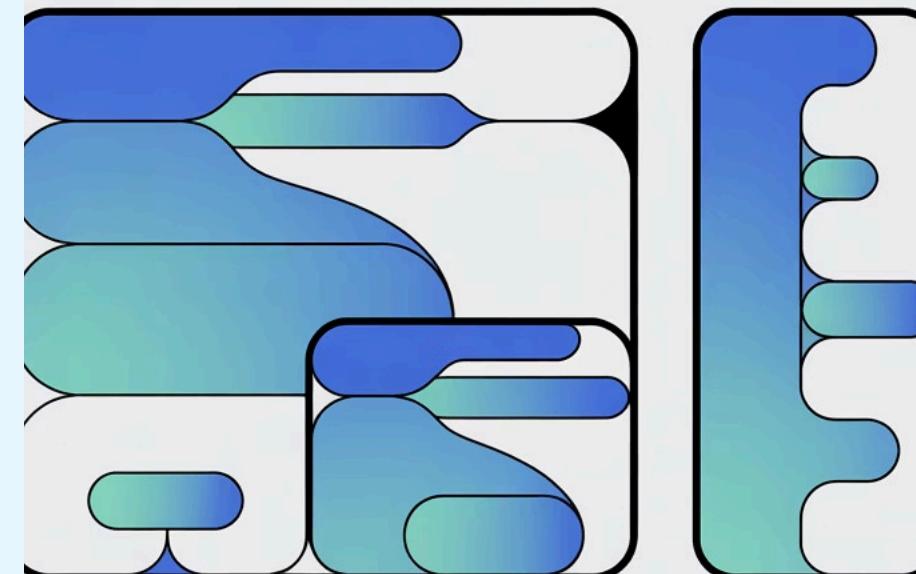
Diseño responsivo con Flexbox

Flexbox es naturalmente adaptable, pero se puede mejorar con media queries:

```
.contenedor {  
    display: flex;  
    flex-wrap: wrap; /* Permite que los elementos salten a la siguiente línea */  
    gap: 20px;  
}  
  
.item {  
    flex: 1 1 300px; /* Grow, shrink, basis */  
    /* Mínimo 300px, pero puede crecer y encogerse */  
}  
  
/* En móviles, apilamos los elementos */  
@media (max-width: 600px) {  
.item {  
    flex-basis: 100%; /* Ocupar todo el ancho */  
}
```

Responsive Web web design

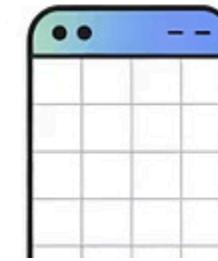
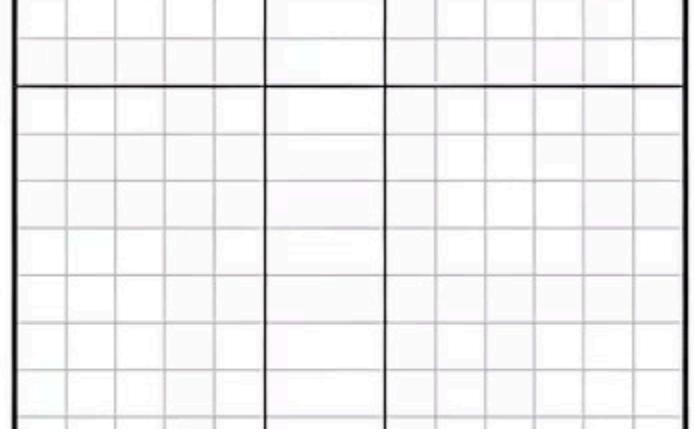
Lespectawíjoli wiile cōuritt angpaft stroviin gnicin or eswniilv,
ard respocit awilioe st anofotshestine iia glvatine.

[Try it free](#)[Try it free](#)

Despñected

Tablet

Mobile

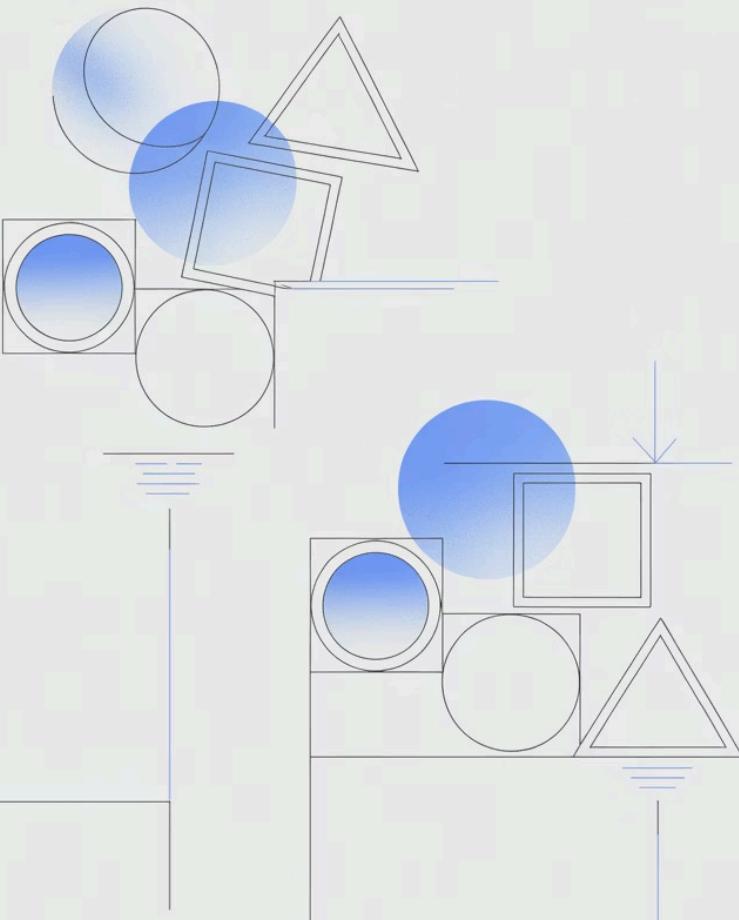


Diseño responsivo con Grid

Grid ofrece opciones potentes para crear layouts adaptables:

```
.grid {  
  display: grid;  
  /* Tantas columnas de mínimo 300px como quepan */  
  grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
  gap: 20px;  
}  
  
/* Alternativa con media queries para más control */  
@media (max-width: 900px) {  
  .grid {  
    grid-template-columns: repeat(2, 1fr);  
  }  
}  
  
@media (max-width: 600px) {  
  .grid {  
    grid-template-columns: 1fr;  
  }  
}
```

Técnicas avanzadas: Alineación y espaciado



Flexbox

- **gap:** Espacio entre elementos
- **align-content:** Distribución de líneas en el eje secundario
- **align-self:** Alineación individual en el eje secundario
- **flex-wrap:** Control del salto de línea

Estas propiedades avanzadas permiten un [control preciso](#) sobre la distribución espacial.

Grid

- **place-items:** Atajo para justify-items + align-items
- **place-content:** Atajo para justify-content + align-content
- **place-self:** Atajo para justify-self + align-self
- **grid-auto-flow:** Control de la colocación automática

Buenas prácticas en maquetación

Mobile-first

Diseña primero para móviles y luego expande para pantallas más grandes. Es más fácil añadir complejidad que simplificar.

Usar unidades relativas

Prefiere rem, em, %, vh/vw, fr sobre píxeles fijos para mejorar la adaptabilidad y accesibilidad.

No abusar de media queries

Aprovecha las capacidades responsivas de Flexbox y Grid antes de recurrir a media queries para cada cambio.

Mantener la semántica

El orden visual no debe comprometer la estructura lógica y semántica del documento HTML.

Un buen layout se adapta a diferentes dispositivos y respeta la [accesibilidad](#) y la [usabilidad](#).

Errores comunes a evitar

Sobrecomplicar layouts

No uses Grid para un simple menú horizontal ni anides contenedores innecesariamente. Elige la herramienta adecuada para cada tarea.

Ignorar el orden del DOM

Modificar radicalmente el orden visual respecto al orden del HTML puede crear problemas de accesibilidad, especialmente para lectores de pantalla.

Olvidar los navegadores antiguos

Implementa fallbacks para IE11 y navegadores que no soporten completamente Flexbox o Grid. Considera usar @supports para funcionalidades avanzadas.

No probar en dispositivos reales

Los emuladores no capturan todos los problemas. Siempre prueba tus layouts en diferentes dispositivos físicos y orientaciones cuando sea posible.

Herramientas y recursos

Documentación

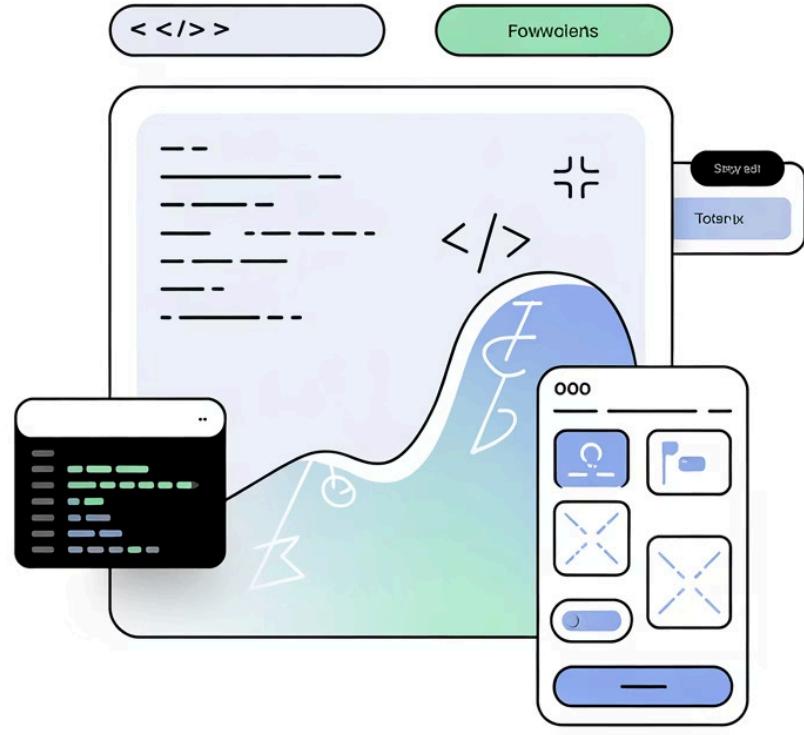
- MDN Web Docs: Guías completas de CSS
- CSS-Tricks: Guías visuales de Flexbox y Grid

Frameworks y sistemas

- Bootstrap 5 (basado en Flexbox)
- Tailwind CSS
- CSS Grid Layout Module

Generadores y visualizadores

- CSS Grid Generator
- Flexbox Playground
- Grid Inspector en Firefox DevTools

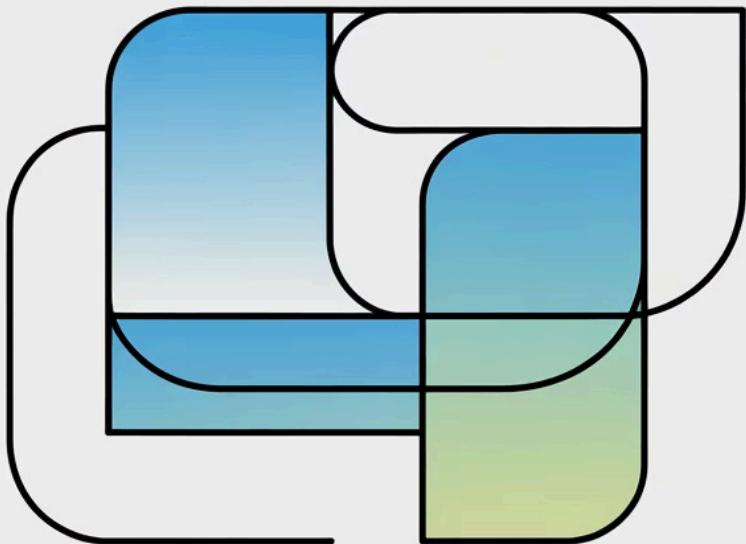


Master CSS Layout

Unlock the power of flexible design

Mastering Layouts

Start building –



Conclusiones

Flexbox y Grid son complementarios

No se trata de elegir uno u otro, sino de conocer las fortalezas de cada uno y usarlos donde brillan más.

Dominar ambos es esencial

Un desarrollador web moderno necesita manejar con soltura tanto Flexbox como Grid para crear interfaces efectivas.

La práctica es clave

Experimenta, construye proyectos y explora las posibilidades para consolidar tu dominio de estas herramientas.

Con Flexbox y Grid, puedes crear prácticamente cualquier layout que imagines. ¡El límite es tu creatividad!

#EDCOUNIANDES

<https://educacioncontinua.uniandes.edu.co/>

Contacto: educacion.continua@uniandes.edu.co

© - Derechos Reservados: La presente obra, y en general todos sus contenidos, se encuentran protegidos por las normas internacionales y nacionales vigentes sobre propiedad Intelectual, por lo tanto su utilización parcial o total, reproducción, comunicación pública, transformación, distribución, alquiler, préstamo público e importación, total o parcial, en todo o en parte, en formato impreso o digital y en cualquier formato conocido o por conocer, se encuentran prohibidos, y solo serán lícitos en la medida en que se cuente con la autorización previa y expresa por escrito de la Universidad de los Andes.