CSS Parte 2: Layout y Posicionamiento

Introducción

En este módulo aprenderemos cómo posicionar y organizar elementos en una página web utilizando CSS. Estos conceptos son fundamentales para crear diseños web profesionales y funcionales.

1. Display (Visualización)

La propiedad `display` determina cómo se muestra un elemento en la página y cómo interactúa con los elementos cercanos. Es una de las propiedades más importantes para controlar el layout.

1.1. `display: block`

Los elementos con `display: block`:

- Ocupan todo el ancho disponible de su contenedor
- Comienzan en una nueva línea
- Respetan propiedades de ancho y alto
- Tienen márgenes por defecto

Elementos que son block por defecto: `<div>`, `<p>`, `<h1>` a `<h6>`, `<ul>`, `<ol>`, `<li>`, `<section>`, `<article>`, etc.

```
.bloque {
  display: block;
  width: 200px;
  height: 100px;
  background-color: lightblue;
  margin: 10px;
}
```

1.2. `display: inline`

Los elementos con `display: inline`:

- Solo ocupan el espacio necesario para su contenido
- No comienzan en una nueva línea
- Ignoran propiedades de ancho y alto
- Respetan márgenes horizontales pero no verticales

Elementos que son inline por defecto: `<span>`, `<a>`, `<strong>`, `<em>`, `<img>`, etc.

```
.enlinea {
  display: inline;
  /* width y height no funcionarán */
  background-color: lightpink;
```

```
  margin: 10px; /* solo los márgenes horizontales se aplicarán */
}
```

1.3. `display: inline-block`

Esta propiedad combina características de los dos anteriores:

- No comienza en nueva línea (como inline)
- Respeta propiedades de ancho y alto (como block)
- Solo ocupa el espacio necesario (como inline)
- Respeta todos los márgenes

```
.enlineabloque {
  display: inline-block;
  width: 150px;
  height: 75px;
  background-color: lightgreen;
  margin: 10px;
}
```

1.4. `display: none`

Oculta completamente el elemento, como si no existiera en el documento:

- No ocupa espacio
- No es visible
- No es accesible

```
.oculto {
  display: none;
}
```

2. Box Model (Modelo de Caja)

El modelo de caja es un concepto fundamental en CSS que describe cómo cada elemento HTML se representa como una caja rectangular con diferentes capas.

2.1. Componentes del Box Model

Cada elemento tiene, de dentro hacia fuera:

1. **Content (Contenido)**: El área donde se muestra el texto o imagen
2. **Padding (Relleno)**: Espacio transparente entre el contenido y el borde
3. **Border (Borde)**: Línea que rodea el padding y contenido
4. **Margin (Margen)**: Espacio transparente fuera del borde


Box Model

2.2. Propiedades del Box Model

```
.caja {
  /* Contenido */
  width: 300px;
  height: 150px;

  /* Padding (arriba, derecha, abajo, izquierda) */
  padding: 20px;
  /* O individualmente */
  padding-top: 10px;
  padding-right: 20px;
  padding-bottom: 10px;
  padding-left: 20px;

  /* Border */
  border: 2px solid black;
  /* O individualmente */
  border-width: 2px;
  border-style: solid;
  border-color: black;

  /* Margin */
  margin: 30px;
  /* O individualmente */
  margin-top: 15px;
  margin-right: 30px;
  margin-bottom: 15px;
  margin-left: 30px;
}
```

2.3. Box-sizing

Por defecto, `width` y `height` solo definen el tamaño del contenido. Para incluir padding y border en estas dimensiones:

```
.caja {
  box-sizing: border-box;
  width: 300px; /* Ahora incluye padding y border */
}
```

Esta propiedad es tan útil que muchos desarrolladores la aplican a todos los elementos:

```
* {
  box-sizing: border-box;
}
```

2.4. Colapso de márgenes

Un fenómeno importante: cuando dos márgenes verticales se encuentran, solo se aplica el mayor de ellos. Esto ocurre entre elementos adyacentes.

```
.caja1 {
  margin-bottom: 30px;
}

.caja2 {
  margin-top: 20px;
}
/* El espacio entre ellas será de 30px, no de 50px */
```

3. Flexbox

Flexbox es un modo de diseño unidimensional que permite distribuir el espacio entre elementos de forma dinámica y flexible. Es especialmente útil para crear diseños responsivos.

3.1. Conceptos básicos

- **Contenedor flex**: El elemento padre que tiene `display: flex`
- **Elementos flex**: Los hijos directos del contenedor flex
- **Eje principal**: Dirección principal del contenedor (horizontal o vertical)
- **Eje cruzado**: Dirección perpendicular al eje principal

3.2. Propiedades del contenedor flex

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

  /* Dirección del eje principal */
  flex-direction: row; /* Horizontal (predeterminado) */
  /* Otras opciones: */
  flex-direction: row-reverse; /* Horizontal invertido */
  flex-direction: column; /* Vertical */
  flex-direction: column-reverse; /* Vertical invertido */

  /* Permite o no el salto de línea */
  flex-wrap: nowrap; /* Sin salto (predeterminado) */
  /* Otras opciones: */
  flex-wrap: wrap; /* Con salto */
  flex-wrap: wrap-reverse; /* Con salto invertido */

  /* Atajo para direction y wrap */
  flex-flow: row wrap;

  /* Alineación en el eje principal */
  justify-content: flex-start; /* Al inicio (predeterminado) */
```

```
  /* Otras opciones: */
  justify-content: flex-end; /* Al final */
  justify-content: center; /* Centrado */
  justify-content: space-between; /* Espacio entre elementos */
  justify-content: space-around; /* Espacio alrededor de elementos */
  justify-content: space-evenly; /* Espacio equidistante */

  /* Alineación en el eje cruzado */
  align-items: stretch; /* Estira los elementos (predeterminado) */
  /* Otras opciones: */
  align-items: flex-start; /* Al inicio */
  align-items: flex-end; /* Al final */
  align-items: center; /* Centrado */
  align-items: baseline; /* Alineado por línea base del texto */

  /* Alineación de múltiples líneas (solo con wrap) */
  align-content: flex-start; /* Líneas al inicio */
  /* Otras opciones similares a justify-content */
}
```

3.3. Propiedades de los elementos flex

```
.elemento-flex {
  /* Orden de aparición (por defecto es 0) */
  order: 1;

  /* Capacidad de crecer respecto a otros elementos */
  flex-grow: 0; /* No crece (predeterminado) */
  flex-grow: 1; /* Crece proporcionalmente */

  /* Capacidad de encogerse */
  flex-shrink: 1; /* Se encoge (predeterminado) */
  flex-shrink: 0; /* No se encoge */

  /* Tamaño base antes de distribuir espacio */
  flex-basis: auto; /* Tamaño basado en contenido */
  flex-basis: 200px; /* Tamaño específico */

  /* Atajo para grow, shrink y basis */
  flex: 0 1 auto; /* Valores predeterminados */
  flex: 1; /* Equivale a flex: 1 1 0% */

  /* Alineación individual (sobrescribe align-items) */
  align-self: auto; /* Hereda del contenedor (predeterminado) */
  /* Mismas opciones que align-items */
}
```

4. Ejemplos prácticos

4.1. Ejemplo 1: Navegación horizontal con Flexbox

```html
<nav class="menu-principal">
  <a href="#">Inicio</a>
  <a href="#">Productos</a>
  <a href="#">Servicios</a>
  <a href="#">Contacto</a>
</nav>
```

```css
.menu-principal {
  display: flex;
  background-color: #333;
  padding: 15px;
}

.menu-principal a {
  color: white;
  text-decoration: none;
  padding: 10px 15px;
  margin-right: 5px;
}

.menu-principal a:hover {
  background-color: #555;
}
```

4.2. Ejemplo 2: Layout de tarjetas

```html
<div class="contenedor-tarjetas">
  <div class="tarjeta">
    <h3>Tarjeta 1</h3>
    <p>Contenido de la primera tarjeta.</p>
  </div>
  <div class="tarjeta">
    <h3>Tarjeta 2</h3>
    <p>Contenido de la segunda tarjeta.</p>
  </div>
  <div class="tarjeta">
    <h3>Tarjeta 3</h3>
    <p>Contenido de la tercera tarjeta.</p>
  </div>
</div>
```

```css
.contenedor-tarjetas {
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
```

```
}

.tarjeta {
  flex-basis: 30%;
  box-sizing: border-box;
  padding: 20px;
  margin-bottom: 20px;
  border: 1px solid #ddd;
  border-radius: 5px;
}
```

4.3. Ejemplo 3: Layout centrado verticalmente

```
<div class="contenedor-centrado">
  <div class="caja-centrada">
    <h2>Contenido centrado</h2>
    <p>
      Este contenido está perfectamente centrado vertical y
horizontalmente.
    </p>
  </div>
</div>
```

```
.contenedor-centrado {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 400px;
  background-color: #f5f5f5;
}

.caja-centrada {
  max-width: 500px;
  padding: 30px;
  background-color: white;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

5. Ejercicios prácticos

Ejercicio 1: Crear un menú de navegación

Crea un menú de navegación horizontal utilizando flexbox que tenga 4 elementos y esté centrado en la página.

Ejercicio 2: Layout de dos columnas

Crea un layout con una barra lateral (sidebar) y un área de contenido principal usando flexbox.

Ejercicio 3: Galería de imágenes

Crea una galería de imágenes responsive con flexbox donde las imágenes se distribuyan uniformemente y cambien de disposición según el espacio disponible.

6. Consejos para principiantes

1. **Usa `box-sizing: border-box`** para un cálculo más intuitivo de dimensiones.
2. **Experimenta con Flexbox** en proyectos pequeños para entender cómo funciona.
3. **Usa las herramientas de desarrollo del navegador** para inspeccionar el box model de los elementos.
4. **Comienza con layouts simples** antes de intentar estructuras complejas.
5. **Practica regularmente** creando diferentes tipos de layouts.

7. Glosario de términos

- **Contenedor**: Elemento que contiene otros elementos
- **Elemento hijo**: Elemento contenido dentro de otro elemento
- **Eje principal**: En flexbox, la dirección principal definida por flex-direction
- **Eje cruzado**: En flexbox, la dirección perpendicular al eje principal
- **Responsive**: Diseño que se adapta a diferentes tamaños de pantalla

Conclusion

Dominar los conceptos de layout y posicionamiento en CSS es fundamental para crear diseños web efectivos. El display, el box model y flexbox son las herramientas que te permitirán controlar cómo se organizan los elementos en tu página. Con práctica, serás capaz de crear diseños cada vez más complejos y profesionales.