# Módulo 3: Layout con CSS – Display, Position y Flexbox (9:40 – 10:30)

## 🎯 Objetivos del módulo

* Entender cómo se comportan los elementos HTML por defecto.
* Conocer y aplicar distintos valores de la propiedad display.
* Aprender las bases del posicionamiento con position.
* Utilizar **Flexbox** para crear layouts dinámicos, modernos y responsivos.

## 📦 ¿Qué es un layout?

Es la **disposición visual** de los elementos en la página. Un layout define cómo se organizan los encabezados, párrafos, imágenes, menús, tarjetas, botones, etc.

CSS nos da varias herramientas para **controlar la ubicación y el comportamiento de los elementos**, empezando por dos propiedades clave:

* display
* position

Y para layouts modernos: **Flexbox y Grid**

## 🔸 Propiedad display

Controla **cómo se presenta un elemento en el flujo del documento**.

### 🧩 Tipos más comunes

| Valor | Comportamiento |
| --- | --- |
| block | Ocupa todo el ancho disponible. |
| inline | Ocupa solo el espacio necesario. |
| inline-block | Como inline pero permite aplicar ancho/alto. |
| none | Elimina el elemento del flujo visual. |
| flex | Activa el modelo de Flexbox. |
| grid | Activa el modelo de CSS Grid. |

### Ejemplo:

html

CopiarEditar

<p style="display: block;">Párrafo block</p>

<span style="display: inline;">Span inline</span>

<div style="display: inline-block; width: 200px;">Caja inline-block</div>

## 🔸 Propiedad position

Permite **colocar elementos en ubicaciones específicas** en la página.

### 📌 Valores clave:

| Valor | Descripción |
| --- | --- |
| static | (Por defecto) sigue el flujo normal del HTML. |
| relative | Se mueve respecto a su posición original. |
| absolute | Se posiciona respecto a su **contenedor posicionado**. |
| fixed | Se queda fijo en la ventana, sin moverse al hacer scroll. |
| sticky | Se comporta como relative, pero se vuelve fixed al hacer scroll. |

### Ejemplo visual:

css

CopiarEditar

.caja-absoluta {

position: absolute;

top: 50px;

left: 100px;

}

### 🧠 Importante: contexto de posicionamiento

Un elemento absolute **se posiciona respecto al primer ancestro que tenga un** position **distinto de** static. Si no hay ninguno, se toma el <body>.

## 🔹 ¿Qué es Flexbox?

**Flexbox** (Flexible Box Layout) es un sistema de diseño unidimensional que permite **alinear elementos en filas o columnas**, de forma eficiente y adaptable.

### ¿Por qué usar Flexbox?

* Centrado vertical y horizontal fácil.
* Distribución automática de espacio.
* Reordenamiento de elementos.
* Compatible con diseño responsivo.

## 🧰 Activando Flexbox

Para usar Flexbox, hay que declarar display: flex en un **contenedor**.

css

CopiarEditar

.contenedor {

display: flex;

}

Los hijos se convierten en **ítems flexibles** y responden a reglas como:

### 🧭 Dirección (main axis)

css

CopiarEditar

flex-direction: row; /\* predeterminado: horizontal \*/

flex-direction: column; /\* vertical \*/

### 🎯 Alineación principal

css

CopiarEditar

justify-content: flex-start; /\* Izquierda \*/

justify-content: center; /\* Centro horizontal \*/

justify-content: space-between; /\* Espacio entre elementos \*/

justify-content: space-around; /\* Espacio igual alrededor \*/

### 🧍 Alineación transversal

css

CopiarEditar

align-items: stretch; /\* predeterminado \*/

align-items: center; /\* centro vertical \*/

align-items: flex-start; /\* arriba \*/

align-items: flex-end; /\* abajo \*/

## 📐 Ejemplo completo

### HTML

html

CopiarEditar

<div class="contenedor">

<div class="caja">Caja 1</div>

<div class="caja">Caja 2</div>

<div class="caja">Caja 3</div>

</div>

### CSS

css

CopiarEditar

.contenedor {

display: flex;

justify-content: center;

align-items: center;

height: 300px;

background-color: lightgray;

}

.caja {

width: 100px;

height: 100px;

margin: 10px;

background-color: teal;

color: white;

display: flex;

align-items: center;

justify-content: center;

}

🖼 Resultado: 3 cajas centradas horizontal y verticalmente.

## 🧪 Actividad práctica guiada

Crear un layout con las siguientes condiciones:

1. Un contenedor de 3 cajas.
2. Las cajas deben estar alineadas horizontalmente al centro.
3. Cada caja debe tener su color y texto.
4. Probar cambiar flex-direction: column.

## 📌 Buenas prácticas

* Siempre definir height si vas a usar align-items: center.
* Agrupar estilos comunes en clases (evitar repetir).
* Para navegadores antiguos, verificar compatibilidad o usar display: inline-block.