







CSS Avanzado - Flexbox y Diseño Web

Dominando el diseño moderno

Fundamentos de la Web - Clase 4

Objetivos de Hoy

¿Qué aprenderemos?

-  Dominar propiedades CSS de color y texto
-  Entender el modelo de caja (box model)
-  Diferenciar display: block, inline e inline-block
-  Usar Flexbox para layouts modernos
-  Crear galerías de imágenes profesionales
-  Desarrollar barras de navegación interactivas




Repaso CSS Básico

Propiedades de Color

¿Cuándo necesito esto?


Escenario: "Quiero que mi título sea azul con fondo gris claro"

Principios esenciales:

-  `color` → Cambia el color del texto
-  `background-color` → Cambia el color de fondo
-  3 formas de definir colores:
 - **Nombres:** `red`, `blue`, `slategray`
 - **RGB:** `rgb(33, 150, 243)`
 - **Hexadecimal:** `#2196F3`

Ejemplo: Aplicando Colores

```
/* Método 1: Nombres de color */
h1 {
  color: slategray;
  background-color: lightgray;
}
/* Método 2: RGB */
h2 {
  color: rgb(33, 150, 243);      /* Azul */
  background-color: rgb(245, 245, 245); /* Gris claro */
}
/* Método 3: Hexadecimal (más común) */
h3 {
  color: #2196F3;               /* Azul */
  background-color: #f5f5f5;    /* Gris claro */
}
```

 Ejemplo completo: `examples/01-colores-css.html`

Propiedades de Texto

¿Cuándo necesito esto?


Escenario: "Quiero centrar un título y ponerlo en negrita"

Propiedades esenciales:

- `text-align` → Alineación (left, center, right)
- `font-weight` → Grosor (normal, bold, 100-900)
- `font-style` → Estilo (normal, italic)
- `text-decoration` → Decoración (none, underline, line-through)
- `font-family` → Tipografía (serif, sans-serif, monospace)

Ejemplo: Estilizando Texto

```
h1 {  
    text-align: center;          /* Centrado */  
    font-weight: bold;           /* Negrita */  
    font-family: sans-serif;     /* Tipografía moderna */  
}  
p {  
    text-align: justify;         /* Justificado */  
    font-weight: normal;        /* Peso normal */  
}  
a {  
    text-decoration: none;       /* Sin subrayado */  
    color: #2196F3;             /* Color azul */  
}  
a:hover {  
    text-decoration: underline; /* Subrayado al pasar el mouse */  
}
```

 Ejemplo completo: `examples/01-colores-css.html`

Modelo de Caja (Box Model)

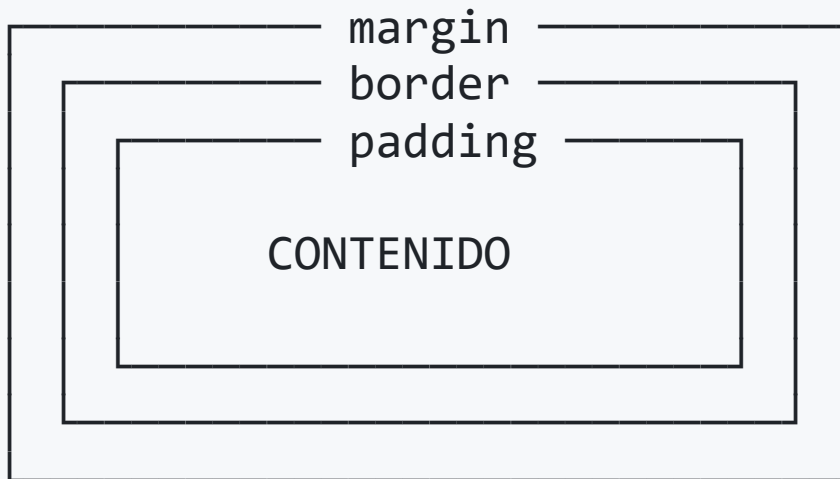


¿Qué es el Modelo de Caja?

¿Cuándo necesito esto?

Escenario: "Mi botón se ve pegado al borde y muy pequeño, necesito darle espacio"

Cada elemento HTML es una caja con 4 capas:



← Espacio externo





← Borde

← Espacio interno

← width x height

Propiedades del Modelo de Caja:

Principios esenciales:

1.  **Contenido**
 - `width` → Ancho
 - `height` → Alto
2.  **Padding (Espacio interno):** Separa el contenido del borde
3.  **Border (Borde):** Línea alrededor del elemento
4.  **Margin (Espacio externo):** Separa el elemento de otros elementos



Ejemplo: Creando un Botón Espaciado

Paso 1: Contenido base

```
button {  
  width: 150px;  
  height: 40px;  
  background-color: #2196F3;  
  color: white;  
}
```

Resultado: Botón básico, pero muy pegado



Ejemplo: Creando un Botón Espaciado

Paso 2: Agregar espacio interno (padding)

```
button {  
  width: 150px;  
  height: 40px;  
  padding: 10px 20px; /* Arriba-abajo 10px | Izq-der 20px */  
  background-color: #2196F3;  
  color: white;  
}
```

Resultado: El texto ya no está pegado a los bordes



Ejemplo: Creando un Botón Espaciado

Paso 3: Agregar borde redondeado

```
button {  
  width: 150px;  
  height: 40px;  
  padding: 10px 20px;  
  background-color: #2196F3;  
  color: white;  
  border: 2px solid #1976D2; /* Grosor | Tipo | Color */  
  border-radius: 5px;        /* Esquinas redondeadas */  
}
```

Resultado: Botón con bordes suaves



Ejemplo: Creando un Botón Espaciado

Paso 4: Agregar espacio externo (margin)

```
button {  
  width: 150px;  
  height: 40px;  
  padding: 10px 20px;  
  background-color: #2196F3;  
  color: white;  
  border: 2px solid #1976D2;  
  border-radius: 5px;  
  margin: 10px; /* Separa del resto de elementos */  
}
```

✅ ¡Botón completo y espaciado!

📁 Ejemplo completo: `examples/02-modelo-caja-boton.html`

Sintaxis Corta: Padding y Margin

4 formas de escribir padding/margin:

```
/* 1. Un valor → Todos los lados iguales */  
padding: 20px;
```

```
/* 2. Dos valores → Vertical | Horizontal */  
padding: 10px 20px; /* Arriba-abajo: 10px | Izq-der: 20px */
```

```
/* 3. Tres valores → Arriba | Horizontal | Abajo */  
padding: 10px 20px 30px; /* Arriba: 10px | Lados: 20px | Abajo: 30px */
```

```
/* 4. Cuatro valores → Arriba | Derecha | Abajo | Izquierda (sentido reloj) */  
padding: 10px 20px 30px 40px;
```

 **Tip:** Usa la sintaxis corta para código más limpio

? Pregunta Rápida

```
.caja {  
  width: 200px;  
  padding: 20px;  
  border: 5px solid black;  
  margin: 10px;  
}
```

¿Cuál es el ancho TOTAL que ocupa la caja en la página?

- A) 200px
- B) 240px
- C) 250px
- D) 270px

Piensa unos segundos... 🤔



Respuesta:

```
.caja {  
  width: 200px;      /* Contenido: 200px */  
  padding: 20px;     /* +20px izq + 20px der = +40px */  
  border: 5px solid; /* +5px izq + 5px der = +10px */  
  margin: 10px;      /* No cuenta para el ancho visual */  
}
```

Respuesta correcta: C) 250px ¿Por qué?

- Contenido: 200px
- Padding: $20\text{px} \times 2 = 40\text{px}$
- Border: $5\text{px} \times 2 = 10\text{px}$
- **Total visual: 250px**

⚠ Nota: El margin NO se suma al ancho visual, pero sí separa de otros elementos




Display: Block vs Inline

Tipos de Display

¿Cuándo necesito esto?

Escenario: "Quiero 3 cajas una al lado de la otra, pero se apilan verticalmente"

Principios esenciales:

-  **block** → Ocupa todo el ancho, siempre en nueva línea
 - Ejemplos: `<div>` , `<p>` , `<h1>` , `<section>`
-  **inline** → Solo ocupa su espacio, puede estar en la misma línea
 - Ejemplos: `` , `<a>` , `` , ``
-  **inline-block** → Híbrido: en línea pero con dimensiones
 - Se puede controlar width/height



Ejemplo: 3 Columnas Lado a Lado

Problema: Por defecto se apilan (block)

```
<div class="columna">Columna 1</div>  
<div class="columna">Columna 2</div>  
<div class="columna">Columna 3</div>
```

```
.columna {  
  width: 200px;  
  background-color: lightblue;  
  padding: 20px;  
  /* display: block; ← Por defecto */  
}
```

Resultado: ❌ Las 3 columnas se apilan verticalmente



Ejemplo: 3 Columnas Lado a Lado

Solución: inline-block

```
.columna {  
    display: inline-block; /* ¡Ahora están en línea! */  
    width: 200px;  
    background-color: lightblue;  
    padding: 20px;  
    vertical-align: top; /* ¡Importante! Alinea arriba */  
}
```

Resultado:  Las 3 columnas están lado a lado



Ejemplo completo: `examples/03-columns-inline-block.html`

⚠ Problema de inline-block

El "espacio fantasma"

Cuando usas `inline-block`, el HTML crea espacios entre elementos:

```
<div class="columna">Col 1</div>  
<div class="columna">Col 2</div> ← Este espacio se renderiza  
<div class="columna">Col 3</div>
```

Problemas:

- ✗ Espacios inesperados entre columnas
- ✗ Difícil calcular anchos exactos con porcentajes
- ✗ Incómodo de controlar

💡 Solución moderna: ¡Flexbox lo resuelve! →

? Pregunta Rápida

¿Qué display tiene cada elemento por defecto?

```
<p>Este es un párrafo</p>  
<a href="#">Este es un enlace</a>  
<div>Este es un div</div>  
<span>Este es un span</span>
```

- A) Todos son block
- B) `<p>` y `<div>` son block; `<a>` y `` son inline
- C) Todos son inline
- D) `<div>` es block; todos los demás son inline



Piensa unos segundos... 🤔

✓ Respuesta

```
<p>Este es un párrafo</p>           <!-- block -->
<a href="#">Este es un enlace</a> <!-- inline -->
<div>Este es un div</div>          <!-- block -->
<span>Este es un span</span>       <!-- inline -->
```

Respuesta correcta: B)

¿Por qué?

-  **Block:** `<p>` , `<div>` , `<h1>-<h6>` , `<section>` , `<article>`
-  **Inline:** `<a>` , `` , `` , `` , ``

¡Bien hecho si acertaste! 🎉



Flexbox - La Solución Moderna

¿Por qué Flexbox?

¿Cuándo necesito esto?

- ✓ Alinear elementos horizontal o verticalmente
- ✓ Distribuir espacio entre elementos
- ✓ Centrar contenido (¡fácil y rápido!)
- ✓ Crear layouts sin "espacios fantasma"
- ✓ Diseños responsivos

Flexbox resuelve los problemas de `inline-block` y más 💪



Principios Esenciales de Flexbox

4 conceptos clave:

1. Contenedor flex (padre)

- `display: flex;` → Activa Flexbox

2. Eje principal (main axis)

- Horizontal por defecto (izquierda → derecha)

3. `justify-content`

- Distribuye elementos en el eje principal

4. `align-items`

- Alinea elementos en el eje transversal (vertical)

Ejemplo: Centrar un Logo

Escenario: Logo debe estar centrado horizontal y verticalmente

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

Paso 1: Activar Flexbox

```
.header {
  display: flex;
  height: 100px;
  background-color: #333;
}
```

Resultado: Logo arriba a la izquierda (comportamiento por defecto)



Ejemplo: Centrar un Logo

Paso 2: Centrar horizontalmente

```
.header {  
  display: flex;  
  justify-content: center; /* Centrado horizontal */  
  height: 100px;  
  background-color: #333;  
}
```

Resultado: Logo centrado horizontalmente, pero arriba



Ejemplo: Centrar un Logo

Paso 3: Centrar verticalmente

```
.header {  
  display: flex;  
  justify-content: center; /* Horizontal */  
  align-items: center;    /* Vertical */  
  height: 100px;  
  background-color: #333;  
}
```

✅ ¡Logo perfectamente centrado!

📁 Ejemplo completo: `examples/05-centrar-logo.html`



Ejemplo: Adiós Espacios Fantasma

```
<div class="container">
  <div class="columna">Columna 1</div>
  <div class="columna">Columna 2</div>
  <div class="columna">Columna 3</div>
</div>
```

```
.container {
  display: flex; /* ¡Solo esto! */
}
.columna {
  width: 200px;
  background-color: lightblue;
  padding: 20px;
  /* No más inline-block ni vertical-align */
}
```

✓ Sin espacios fantasma, perfectamente alineado

📁 Ejemplo completo: `examples/04-columnas-flex.html`

justify-content: Distribución Horizontal

Controla cómo se distribuyen los elementos:

```
.container {  
  display: flex;  
  justify-content: flex-start;    /* ← Izquierda (defecto) */  
  /* justify-content: center; */  /* ← Centro */  
  /* justify-content: flex-end; */ /* ← Derecha */  
  
  /* Distribución con espacio: */  
  /* justify-content: space-between; */ /* Espacio entre (bordes pegados) */  
  /* justify-content: space-around; */ /* Espacio alrededor */  
  /* justify-content: space-evenly; */ /* Espacio uniforme */  
}
```


justify-content: Visualización

```
flex-start:    [A][B][C]_____
center:        _____[A][B][C]_____
flex-end:      _____[A][B][C]
space-between:  [A]_____ [B]_____ [C]
space-around:   __[A]_____ [B]_____ [C]__
space-evenly:  __[A]_____ [B]_____ [C]__
```

💡 **Tip:** Usa `space-between` para navbars, `space-evenly` para galerías



align-items: Alineación Vertical

Controla cómo se alinean verticalmente:

```
.container {  
  display: flex;  
  height: 200px;  
  
  align-items: flex-start; /* ← Arriba */  
  /* align-items: center; */ /* ← Centro */  
  /* align-items: flex-end; */ /* ← Abajo */  
  /* align-items: stretch; */ /* ← Estira (defecto) */  
}
```



Ejemplo: NavBar Básica

Escenario: Logo a la izquierda, links a la derecha

```
<nav class="navbar">
  <div class="logo">MiSitio</div>
  <div class="links">Inicio | Servicios | Contacto</div>
</nav>
```

```
.navbar {
  display: flex;
  justify-content: space-between; /* ¡Clave! */
  align-items: center;
  padding: 20px;
  background-color: #2196F3;
  color: white;
}
```

✓ Logo izquierda, Links derecha, ambos centrados verticalmente

? Pregunta Rápida

Quiero centrar un botón perfectamente en una caja de 300px × 200px

```
<div class="caja">  
  <button>Haz clic</button>  
</div>
```

¿Qué CSS necesito?

- A) Solo `text-align: center`
- B) `display: flex; justify-content: center;`
- C) `display: flex; align-items: center;`
- D) `display: flex; justify-content: center; align-items: center;`

Piensa unos segundos... 🤔

✓ Respuesta

```
.caja {  
  display: flex;  
  justify-content: center; /* Centrado horizontal */  
  align-items: center;    /* Centrado vertical */  
  width: 300px;  
  height: 200px;  
}
```

Respuesta correcta: D) ¿Por qué?

- `justify-content: center` → Centra horizontalmente
- `align-items: center` → Centra verticalmente
- ¡Necesitas AMBAS para centrado perfecto!

¡Excelente si acertaste! 🎉



Diseño con Imágenes



Galerías con Flexbox

¿Cuándo necesito esto?

Escenario: "Galería de productos/imágenes alineadas y del mismo tamaño"

Principios esenciales:

1. Todas las imágenes mismo tamaño → `width` + `height`
2. Flexbox para alinear → `display: flex`
3. Espacio uniforme → `justify-content: space-evenly`
4. Mantener proporción → `object-fit: cover`
5. Múltiples filas → `flex-wrap: wrap`



Ejemplo: Galería 4 Imágenes

HTML simple:

```
<div class="galeria">  
    
    
    
    
</div>
```




Ejemplo: Galería 4 Imágenes

Paso 1: Contenedor Flex

```
.galeria {  
  display: flex;  
  justify-content: space-evenly; /* Espacio uniforme */  
  padding: 20px;  
  background-color: #f5f5f5;  
}
```

Resultado: Imágenes en fila, pero tamaños irregulares



Ejemplo: Galería 4 Imágenes

Paso 2: Imágenes uniformes

```
.galeria img {  
  width: 150px;  
  height: 150px;  
  border-radius: 10px; /* Esquinas redondeadas */  
  object-fit: cover; /* ¡Importante! Mantiene proporción */  
}
```

✓ ¡Galería perfecta con imágenes uniformes!

📁 Ejemplo completo: `examples/06-galeria-imagenes.html`



object-fit: Manejo de Proporciones

Controla cómo la imagen se ajusta:

```
img {  
  width: 200px;  
  height: 200px;  
  
  object-fit: cover;    /* ← Cubre todo, recorta si es necesario */  
  /* object-fit: contain; */ /* Cabe completo, puede dejar espacios */  
  /* object-fit: fill; */   /* Estira para llenar (distorsiona) */  
}
```



Recomendación: Usa `cover` para galerías uniformes

flex-wrap: Múltiples Filas

Para galerías con muchas imágenes:

```
.galeria {  
  display: flex;  
  flex-wrap: wrap;      /* Permite múltiples filas */  
  gap: 20px;           /* Espacio entre imágenes */  
  justify-content: center;  
}  
  
.galeria img {  
  width: 200px;  
  height: 200px;  
  object-fit: cover;  
}
```

✓ Las imágenes se distribuyen automáticamente en varias filas

? Pregunta Rapida

Tengo 6 imágenes de diferentes tamaños en una galería

```
.galeria {  
  display: flex;  
}  
.galeria img {  
  width: 150px;  
  height: 150px;  
  /* ¿Qué falta? */  
}
```

¿Qué propiedad necesito para que NO se distorsionen?

- A) background-size: cover
- B) object-fit: cover
- C) border-radius: 50%
- D) No se puede evitar

Piensa unos segundos



✓ Respuesta

```
.galeria img {  
  width: 150px;  
  height: 150px;  
  object-fit: cover; /* ← ¡Esta es la clave! */  
}
```

Respuesta correcta: B)

¿Por qué?

- `object-fit: cover` ajusta la imagen sin distorsionar
- Recorta lo necesario para llenar el espacio
- Mantiene la proporción original

¡Perfecto si acertaste! 🎉






Barra de Navegación Profesional

Anatomía de una NavBar

¿Cuándo necesito esto?

Escenario: "Menú de navegación con logo a la izquierda y links a la derecha"

Componentes esenciales:

1.  HTML semántico → `<nav>`
2.  Flexbox → Logo vs Links
3.  Lista horizontal → Links en línea
4.  Interactividad → Estados hover
5.  Alineación → `space-between` + `center`



Ejemplo NavBar: Paso 1 - HTML

Estructura semántica:

```
<nav class="navbar">
  <div class="logo">
    <a href="#">MiSitio</a>
  </div>
  <ul class="nav-links">
    <li><a href="#">Inicio</a></li>
    <li><a href="#">Servicios</a></li>
    <li><a href="#">Nosotros</a></li>
    <li><a href="#">Contacto</a></li>
  </ul>
</nav>
```



Ejemplo NavBar: Paso 2 - Container Flex

Navbar como contenedor flex:

```
.navbar {  
  display: flex;  
  justify-content: space-between; /* Logo izq, Links der */  
  align-items: center;           /* Centrado vertical */  
  background-color: #2196F3;  
  padding: 15px 30px;  
}
```

Resultado: Logo a la izquierda, links a la derecha



Ejemplo NavBar: Paso 3 - Logo Estilo

Estilizando el logo:

```
.logo a {  
  color: white;  
  font-size: 24px;  
  font-weight: bold;  
  text-decoration: none; /* Sin subrayado */  
  font-family: 'Arial', sans-serif;  
}
```

Resultado: Logo destacado y limpio



Ejemplo NavBar: Paso 4 - Links Horizontales

Lista como contenedor flex:

```
.nav-links {  
  display: flex;           /* ¡Links en línea! */  
  list-style: none;       /* Sin viñetas */  
  gap: 20px;              /* Espacio entre items */  
  margin: 0;  
  padding: 0;  
}  
  
.nav-links a {  
  color: white;  
  text-decoration: none;  
  font-size: 16px;  
}
```

Resultado: Links alineados horizontalmente



Ejemplo NavBar: Paso 5 - Hover Interactivo

Estados hover:

```
.nav-links a {  
  color: white;  
  text-decoration: none;  
  transition: color 0.3s ease; /* Transición suave */  
}  
  
.nav-links a:hover {  
  color: #FFD700; /* Dorado al pasar el mouse */  
}  
  
.logo a:hover {  
  color: #FFD700;  
}
```

✅ ¡NavBar completa e interactiva!

📁 Ejemplo completo: `examples/07-navbar-completa.html`



Variaciones de NavBar

Puedes personalizar:

```
/* NavBar oscura */
.navbar {
  background-color: #1a1a1a;
}

/* NavBar con sombra */
.navbar {
  box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

/* Links con fondo al hover */
.nav-links a:hover {
  background-color: rgba(255,255,255,0.1);
  padding: 5px 10px;
  border-radius: 5px;
}
```

? Pregunta Rápida

Para una navbar con logo izquierda y links derecha:

```
<nav class="navbar">  
  <div class="logo">Logo</div>  
  <ul class="links">...</ul>  
</nav>
```

¿Qué justify-content uso?

- A) flex-start
- B) center
- C) space-between
- D) space-evenly

Piensa unos segundos... 🤔



Respuesta

```
.navbar {  
  display: flex;  
  justify-content: space-between; /* ← ¡Correcto! */  
  align-items: center;  
}
```

Respuesta correcta: C)

¿Por qué?

- `space-between` → Primer elemento al inicio, último al final
- Logo queda a la izquierda
- Links quedan a la derecha
- Espacio máximo entre ellos

¡Genial si acertaste! 🎉



Resumen y Recursos



Conceptos Clave de Hoy

Lo más importante:

- ✓ **Modelo de Caja:** margin → border → padding → contenido
- ✓ **Display:** block (nueva línea), inline (misma línea), inline-block (híbrido)
- ✓ **Flexbox:** `display: flex` en el padre
- ✓ **justify-content:** Distribución horizontal (space-between, center)
- ✓ **align-items:** Alineación vertical (center, flex-start, flex-end)
- ✓ **object-fit: cover:** Imágenes sin distorsión






Lo Más Importante

Recuerda:

- 💡 Flexbox es tu mejor amigo para layouts modernos
- 💡 `justify-content` controla el eje principal (horizontal)
- 💡 `align-items` controla el eje transversal (vertical)
- 💡 `space-between` es perfecto para navbars
- 💡 `object-fit: cover` para galerías profesionales
- 💡 Siempre usa `display: flex` en el contenedor padre

Recursos para Seguir Aprendiendo






Documentación y herramientas:

-  [Flexbox Froggy](#) - ¡Aprende jugando!
-  [CSS-Tricks Flexbox Guide](#) - Guía completa
-  [MDN Web Docs - Flexbox](#) - Documentación oficial
-  [Flexbox Defense](#) - Juego de estrategia
-  [Flexbox Playground](#) - Experimenta en vivo

Próximos Pasos

¿Qué viene después?

En las próximas clases veremos:





-  CSS Grid - Layouts bidimensionales
-  Media Queries - Diseño responsive
-  Animaciones CSS - Transiciones y efectos
-  Posicionamiento - Absolute, relative, fixed
-  Diseño Mobile-First - Del celular al desktop

¡Flexbox es la base para todo esto!

Práctica para Casa

Proyecto sugerido: Portfolio Personal

Crea una página con:

1.  **Navbar** con tu nombre y secciones (Sobre mí, Proyectos, Contacto)
2.  **Galería** de proyectos (mínimo 6 imágenes)
3.  **Cards** con información usando flexbox
4.  **Colores** personalizados y hover effects

Recursos: Usa los ejemplos de `examples/` como referencia

Consejos Finales

Buenas prácticas:

- ✓ Usa Flexbox para layouts unidimensionales (filas o columnas)
- ✓ Combina con CSS Grid cuando necesites layouts bidimensionales
- ✓ Evita abusar de divs - usa HTML semántico
- ✓ Usa DevTools (F12) para experimentar con Flexbox en vivo
- ✓ Mobile-first - piensa primero en móvil
- ✓ Practica, practica, practica - la única forma de dominar

 ¡Felicidades!

Ya dominas Flexbox y diseño moderno con CSS

Ahora puedes crear layouts profesionales y responsive 

? Preguntas

¿Alguna duda sobre Flexbox o el modelo de caja?

 ¡Excelente Trabajo!

¡Nos vemos en la siguiente clase con más CSS avanzado!

No olvides completar todos los ejercicios 