







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?

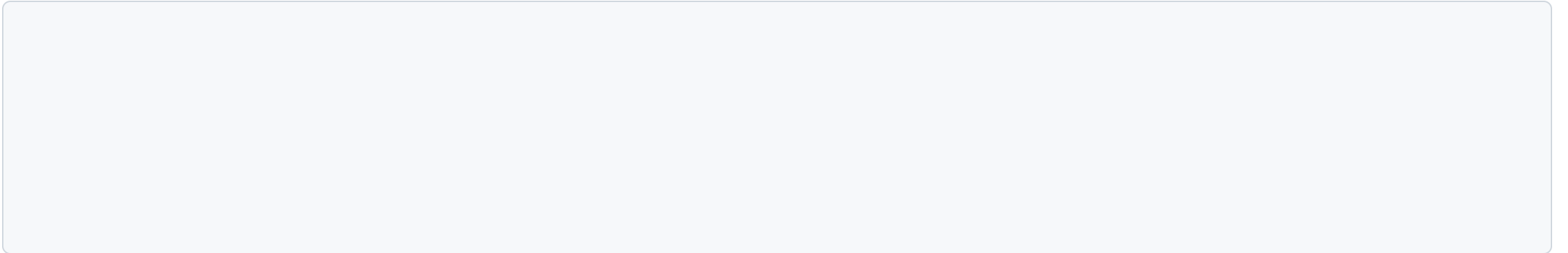
Escenario: "Quiero que mi título sea azul con fondo gris claro"

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



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

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

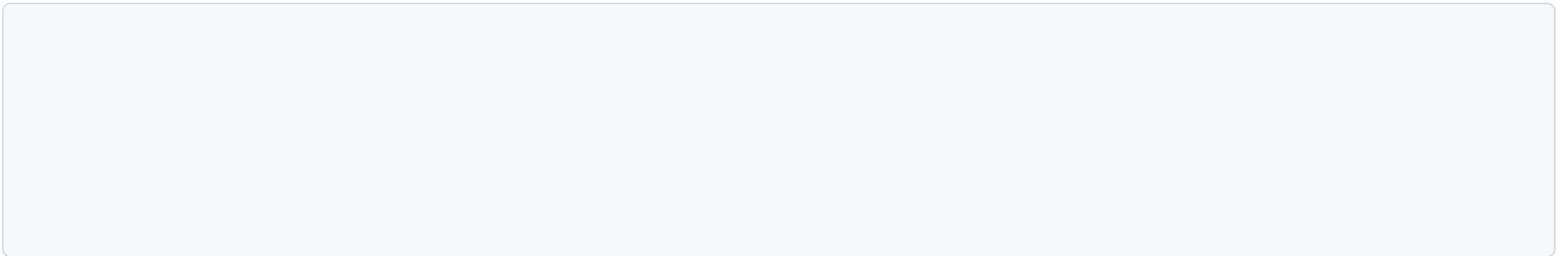
Modelo de Caja (Box Model)

¿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:



Propiedades del Modelo de Caja:

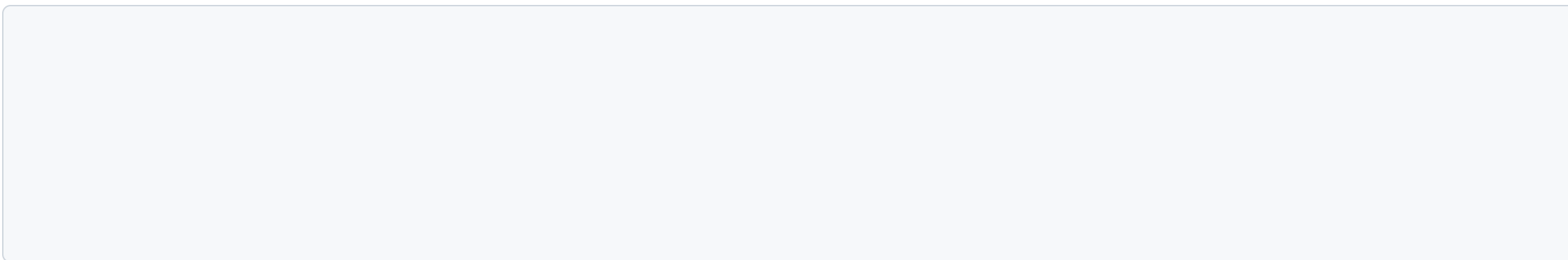
Principios esenciales:

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

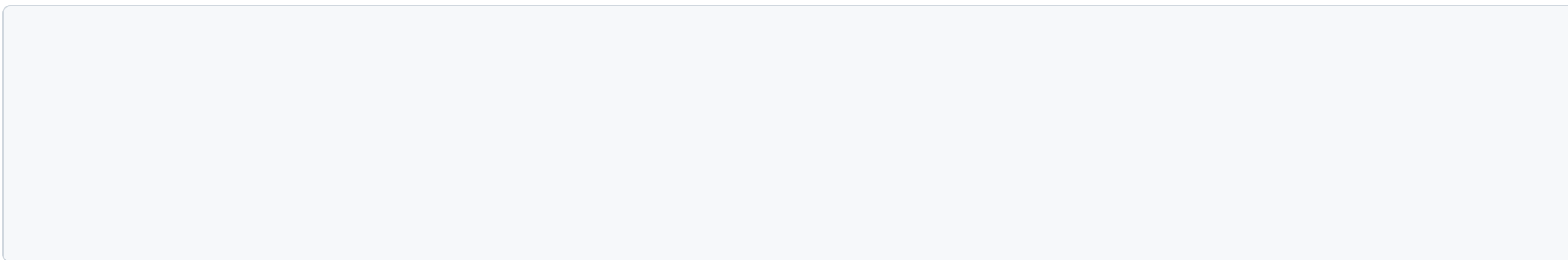


Resultado: Botón básico, pero muy pegado



Ejemplo: Creando un Botón Espaciado

Paso 2: Agregar espacio interno (padding)

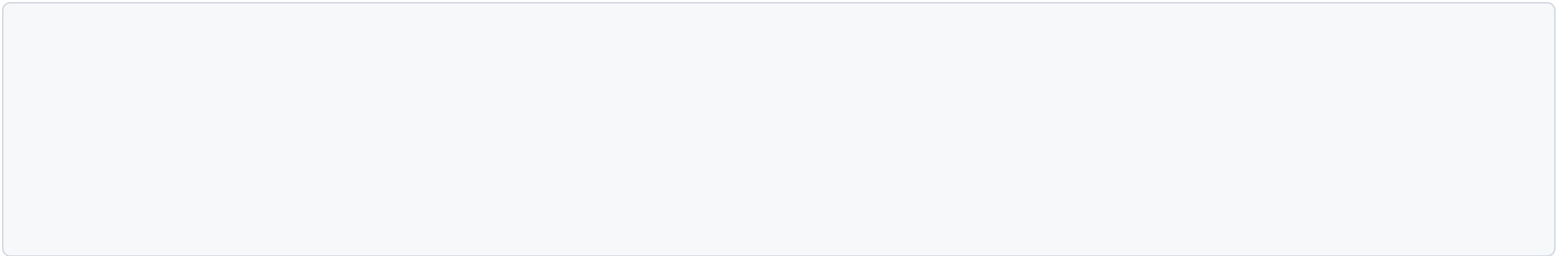


Resultado: El texto ya no está pegado a los bordes



Ejemplo: Creando un Botón Espaciado

Paso 3: Agregar borde redondeado

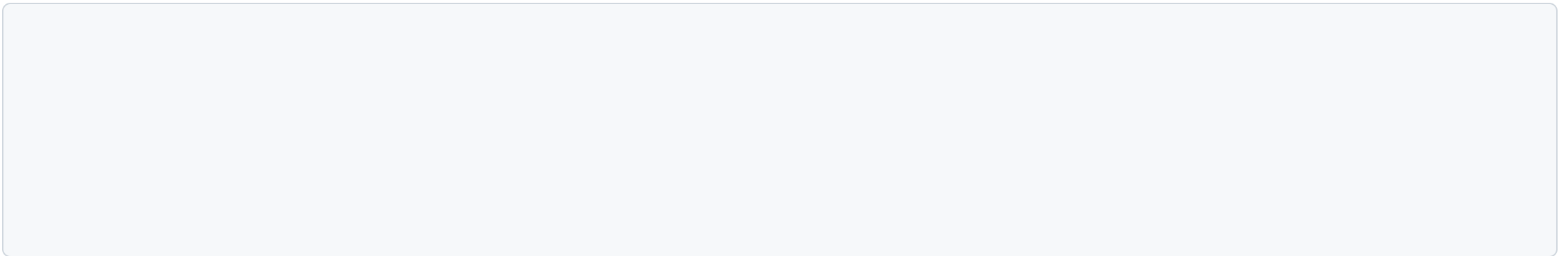


Resultado: Botón con bordes suaves



Ejemplo: Creando un Botón Espaciado

Paso 4: Agregar espacio externo (margin)

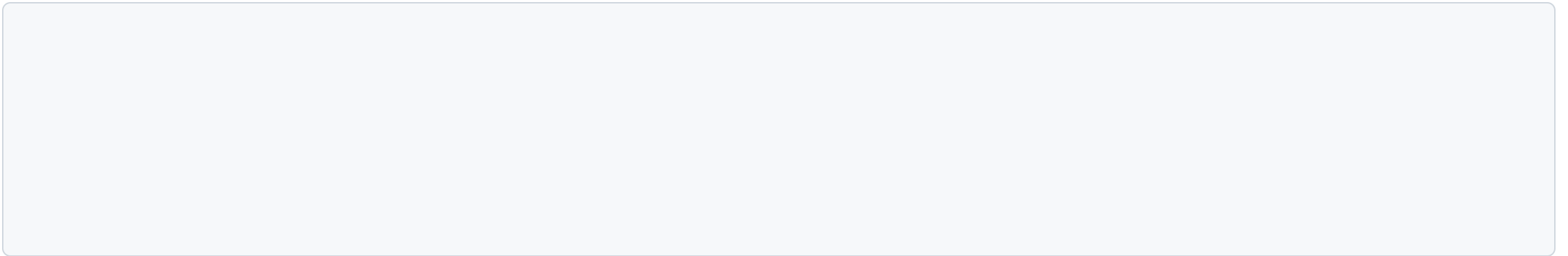


✅ ¡Botón completo y espaciado!

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

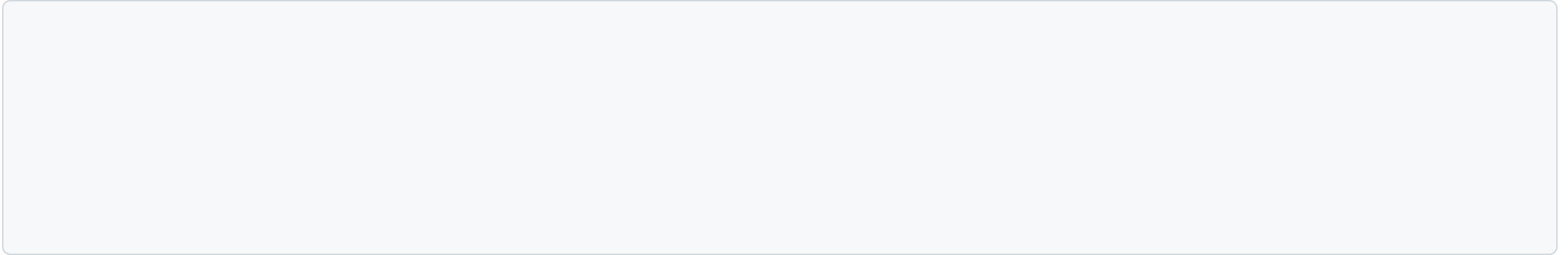
Sintaxis Corta: Padding y Margin

4 formas de escribir padding/margin:



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

? Pregunta Rápida



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

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

Piensa unos segundos... 🤔



Respuesta:

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"

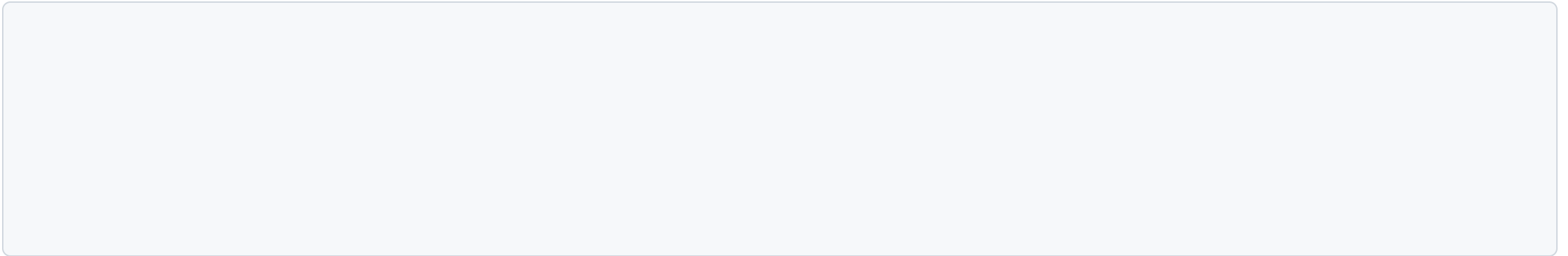
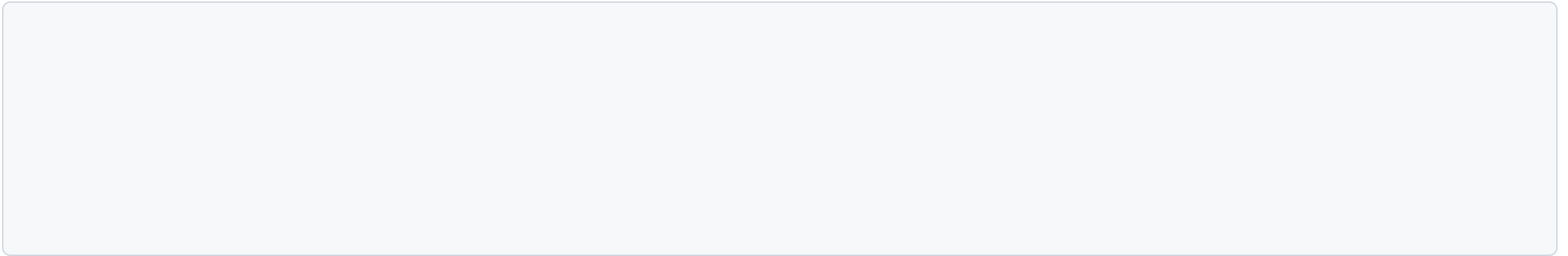
Principios esenciales:

-  **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)

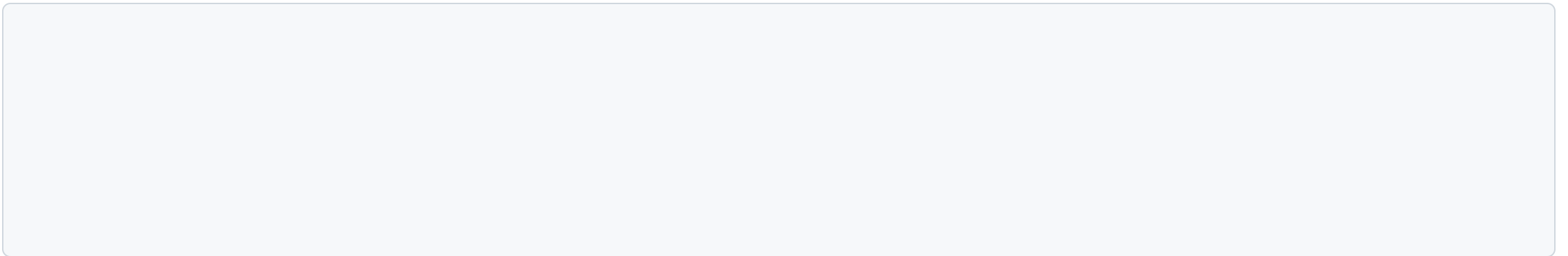


Resultado: **✗** Las 3 columnas se apilan verticalmente



Ejemplo: 3 Columnas Lado a Lado

Solución: inline-block



Resultado:  Las 3 columnas están lado a lado

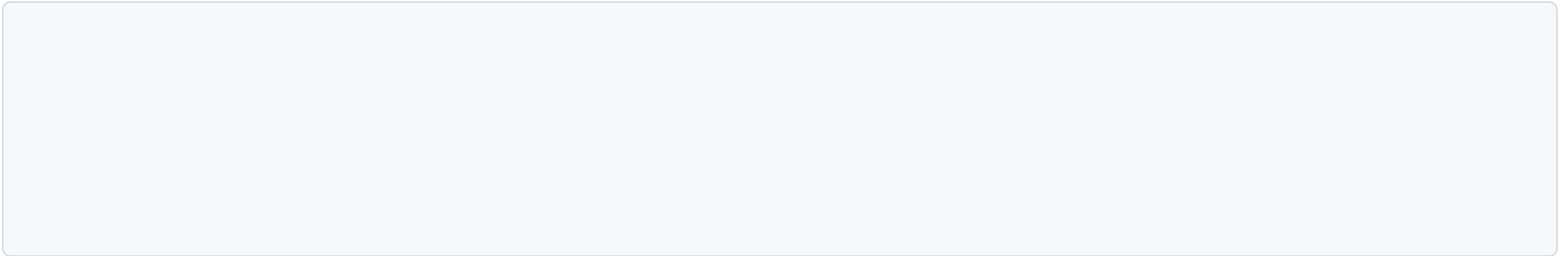


Ejemplo completo: `examples/03-columns-inline-block.html`

⚠ Problema de inline-block

El "espacio fantasma"

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



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?



- 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

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

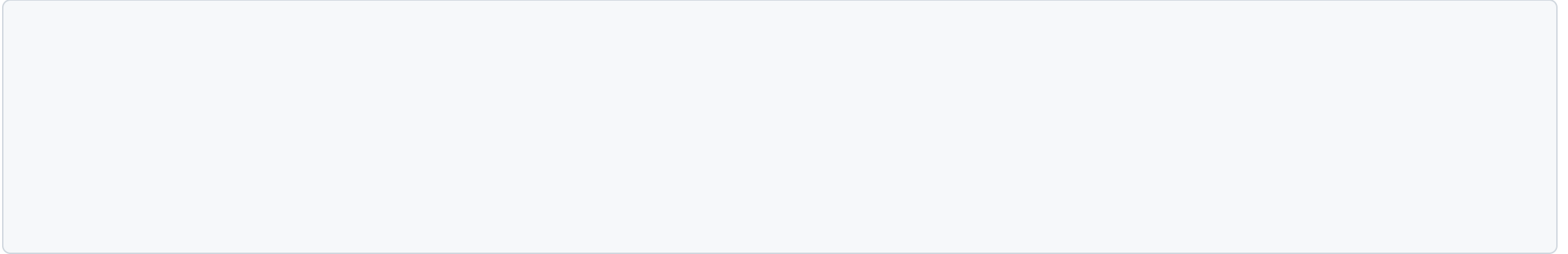
4. `align-items`

- Alinea elementos en el eje transversal (vertical)

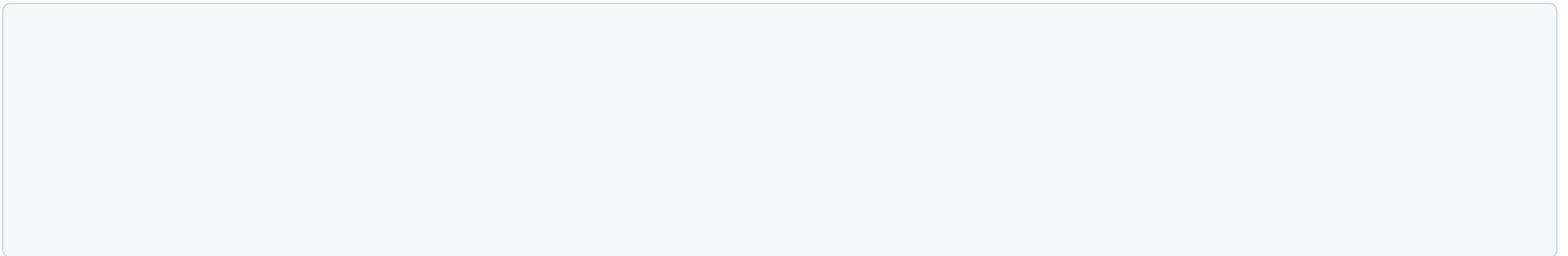


Ejemplo: Centrar un Logo

Escenario: Logo debe estar centrado horizontal y verticalmente



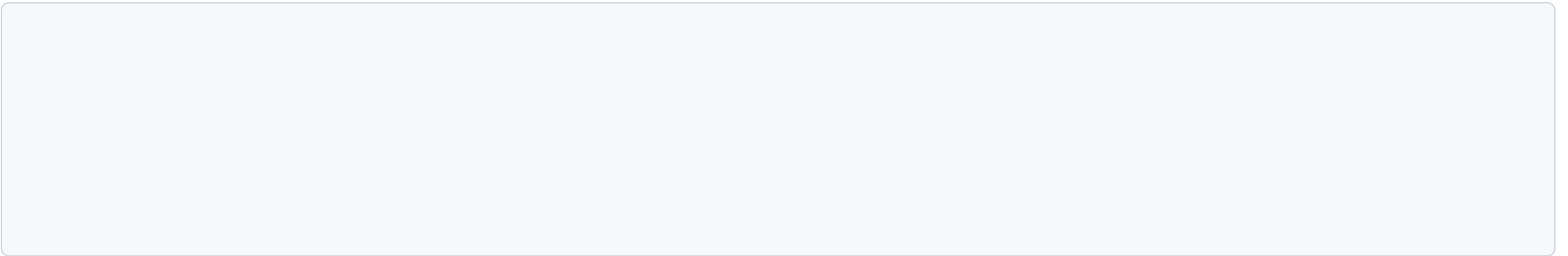
Paso 1: Activar Flexbox



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

Ejemplo: Centrar un Logo

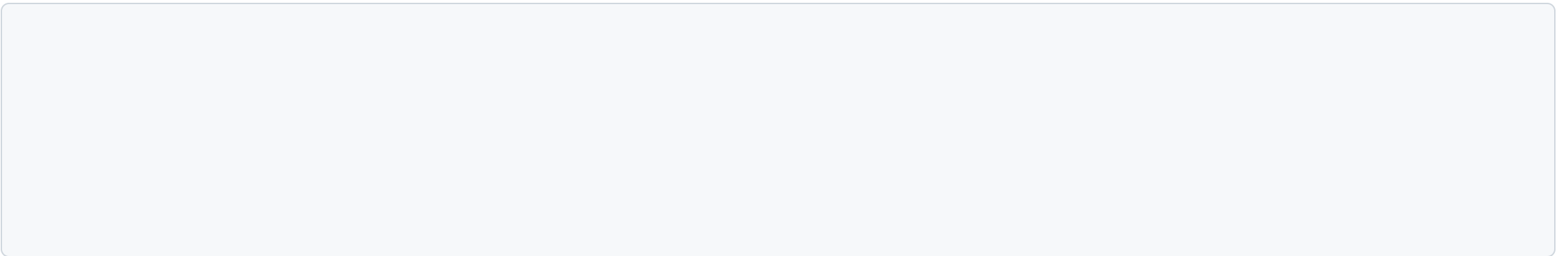
Paso 2: Centrar horizontalmente



Resultado: Logo centrado horizontalmente, pero arriba

Ejemplo: Centrar un Logo

Paso 3: Centrar verticalmente

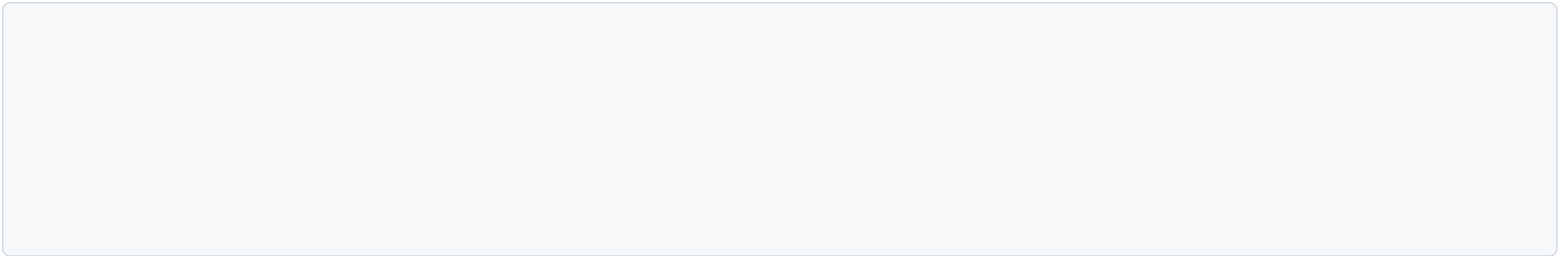
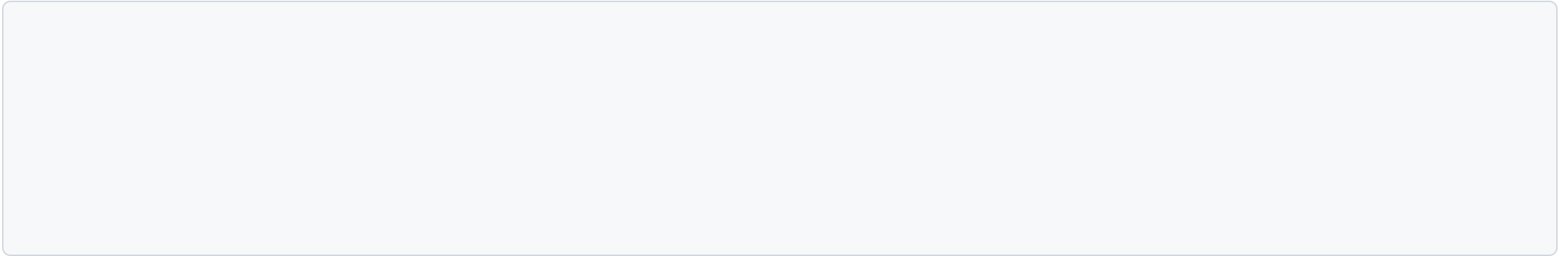


✅ ¡Logo perfectamente centrado!

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



Ejemplo: Adiós Espacios Fantasma

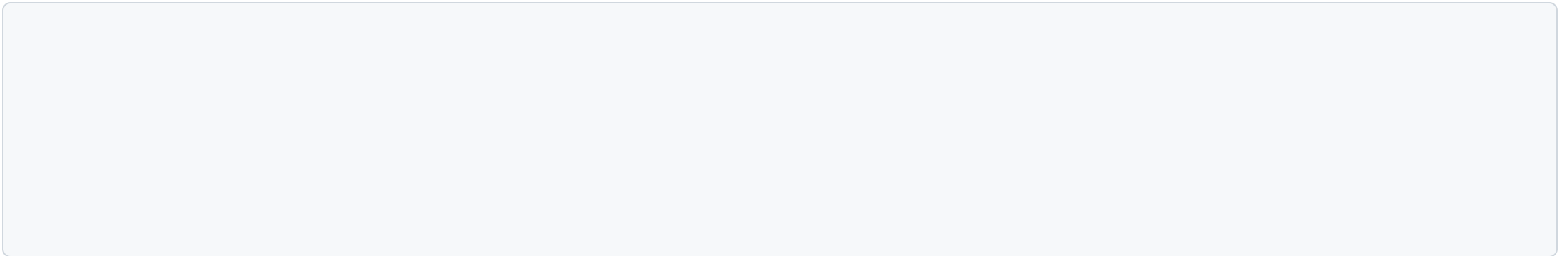


✓ Sin espacios fantasma, perfectamente alineado

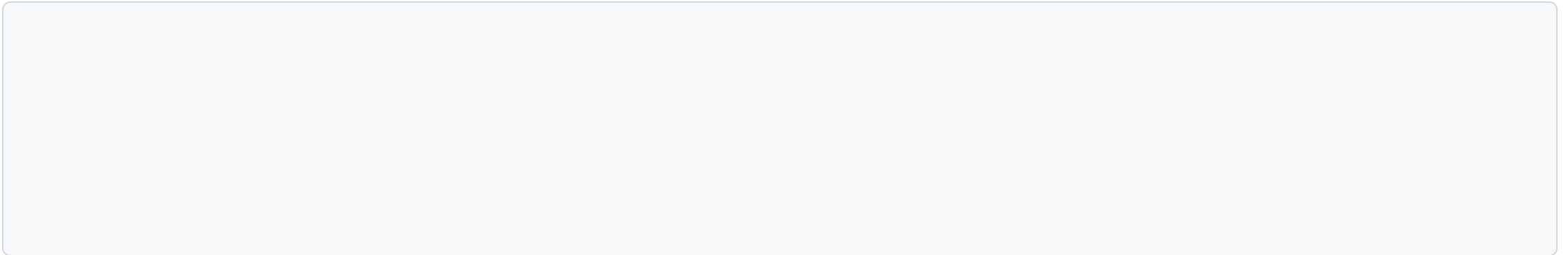
📁 Ejemplo completo: `examples/04-columns-flex.html`

justify-content: Distribución Horizontal

Controla cómo se distribuyen los elementos:



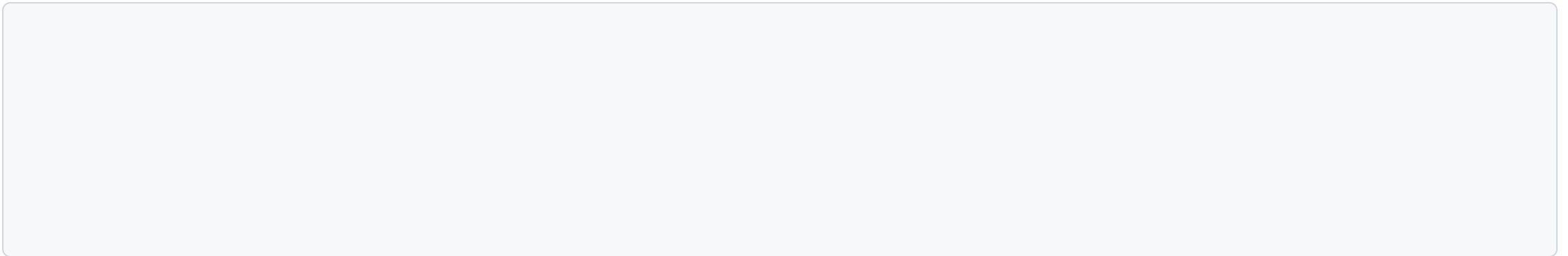
justify-content: Visualización



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

align-items: Alineación Vertical

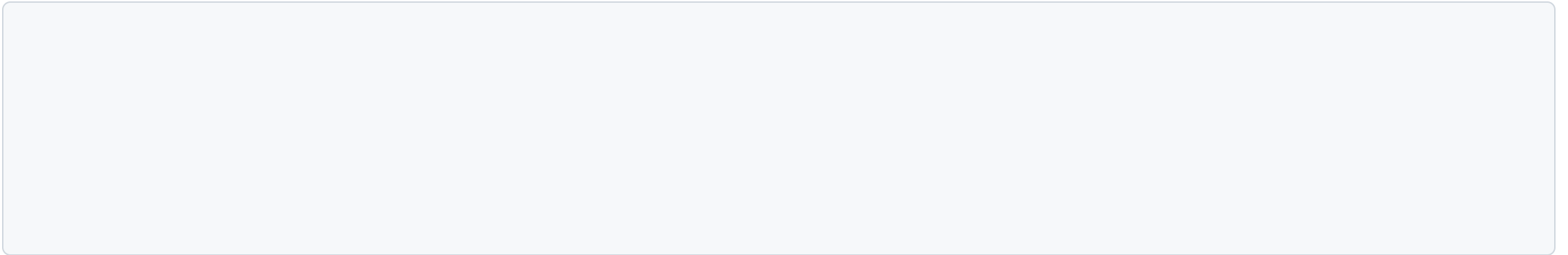
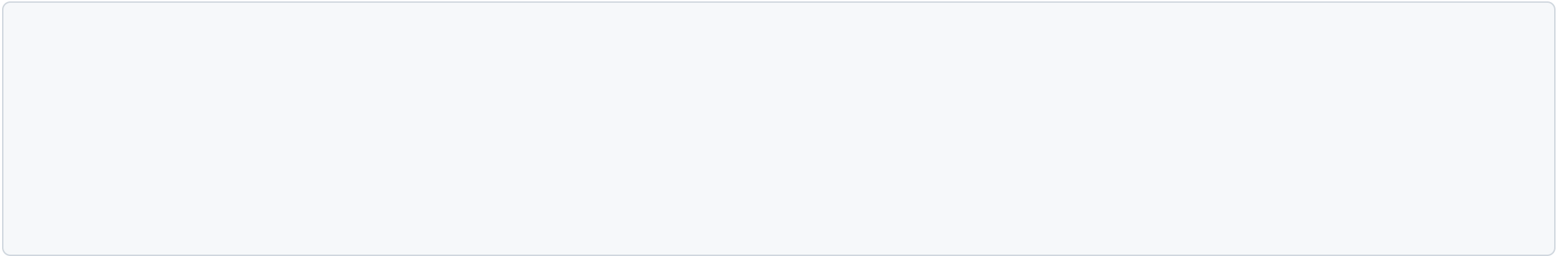
Controla cómo se alinean verticalmente:





Ejemplo: NavBar Básica

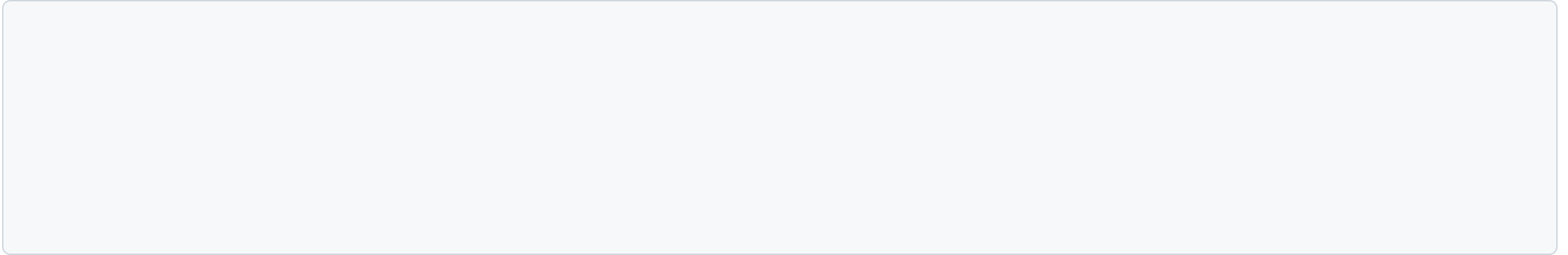
Escenario: Logo a la izquierda, links a la derecha



✓ Logo izquierda, Links derecha, ambos centrados verticalmente

? Pregunta Rápida

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



¿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

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"

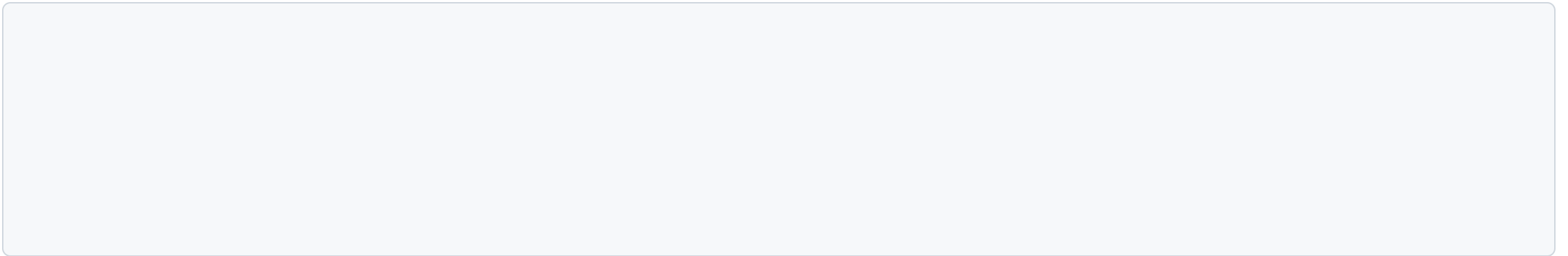
Principios esenciales:

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`



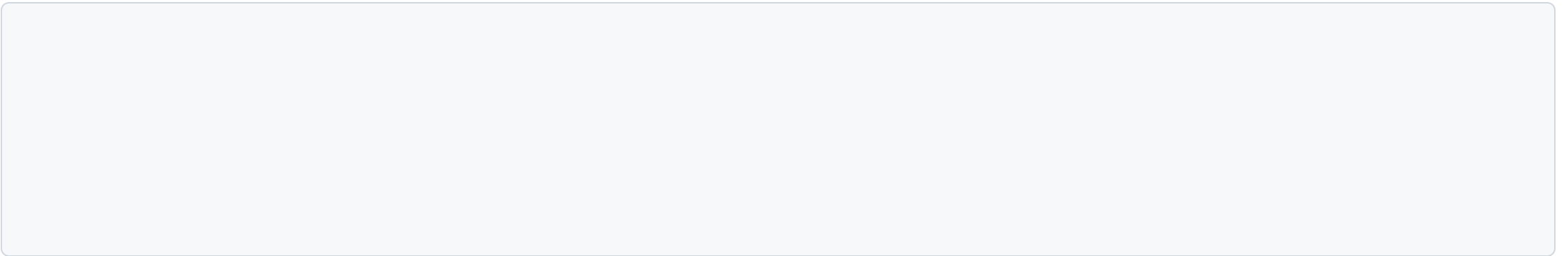
Ejemplo: Galería 4 Imágenes

HTML simple:



Ejemplo: Galería 4 Imágenes

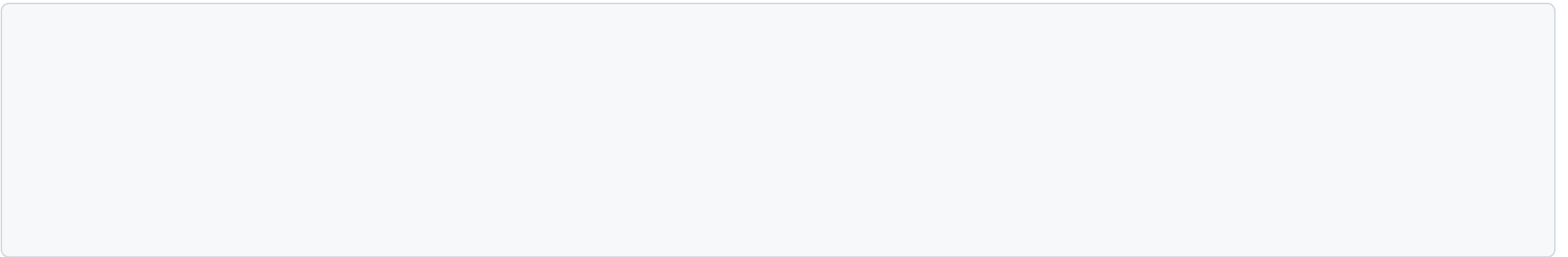
Paso 1: Contenedor Flex



Resultado: Imágenes en fila, pero tamaños irregulares

Ejemplo: Galería 4 Imágenes

Paso 2: Imágenes uniformes



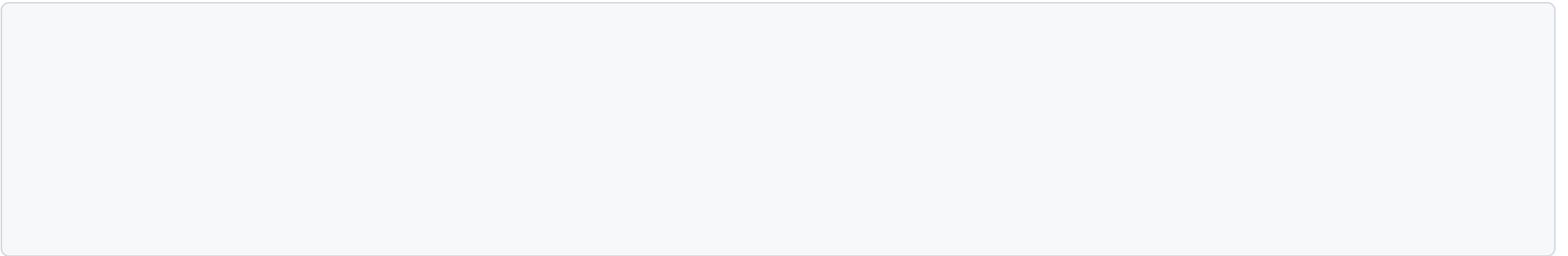
✓ ¡Galería perfecta con imágenes uniformes!

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



object-fit: Manejo de Proporciones

Controla cómo la imagen se ajusta:

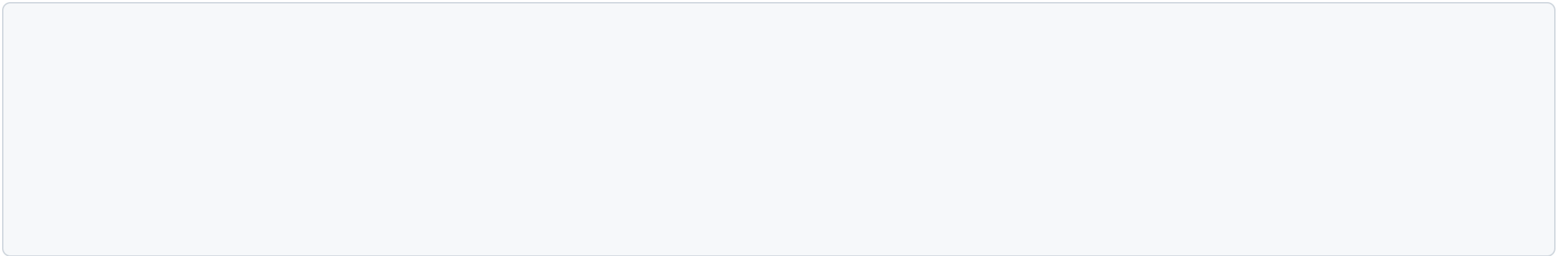


Recomendación: Usa `cover` para galerías uniformes



flex-wrap: Múltiples Filas

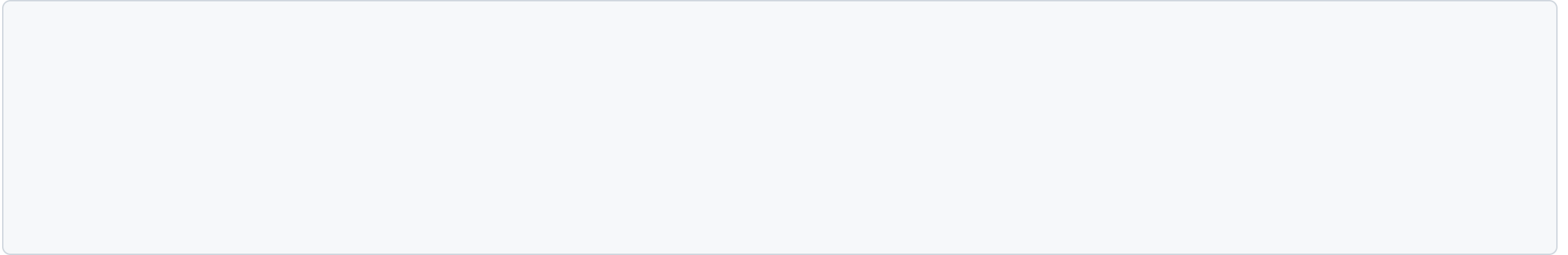
Para galerías con muchas imágenes:



- ✓ Las imágenes se distribuyen automáticamente en varias filas

? Pregunta Rápida

Tengo 6 imágenes de diferentes tamaños en una galería



¿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

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"

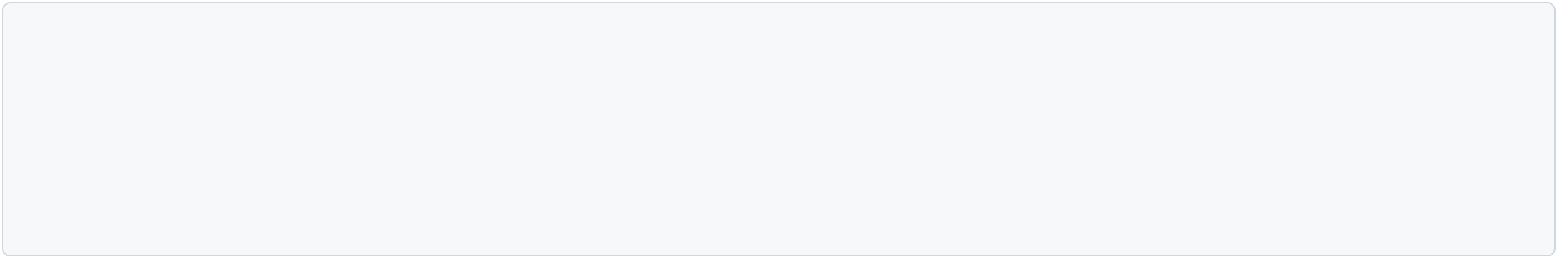
Componentes esenciales:

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

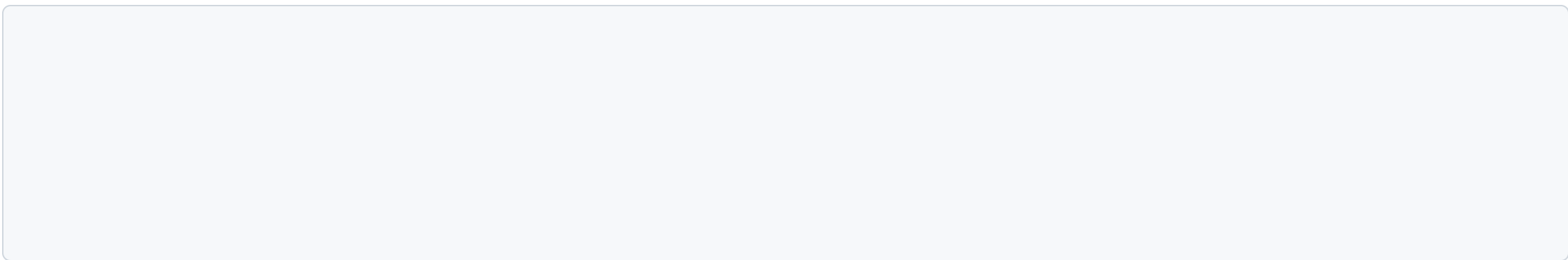
Estructura semántica:





Ejemplo NavBar: Paso 2 - Container Flex

Navbar como contenedor flex:

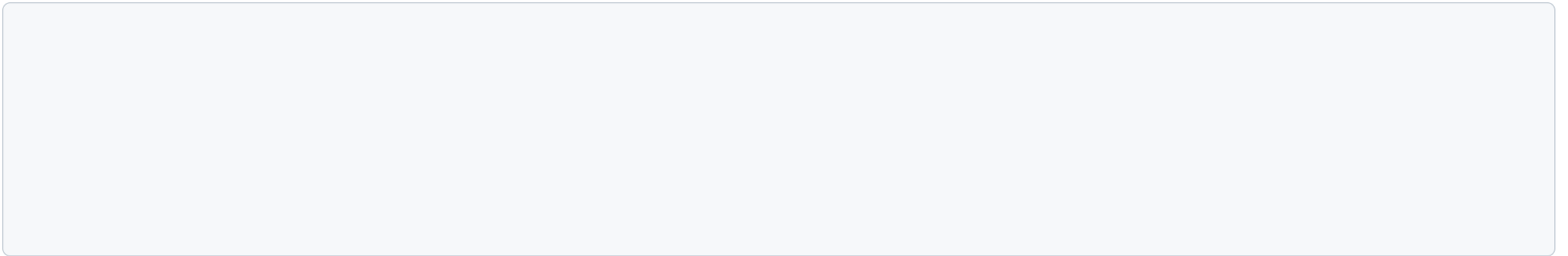


Resultado: Logo a la izquierda, links a la derecha



Ejemplo NavBar: Paso 3 - Logo Estilo

Estilizando el logo:

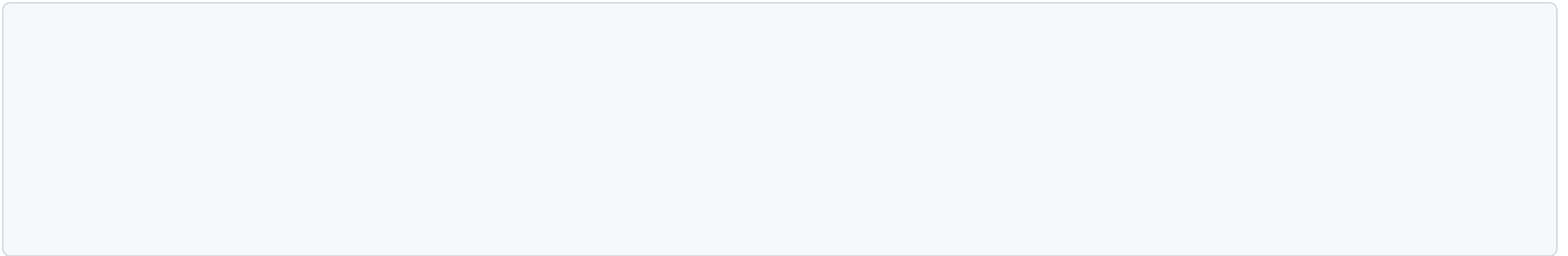


Resultado: Logo destacado y limpio



Ejemplo NavBar: Paso 4 - Links Horizontales

Lista como contenedor flex:

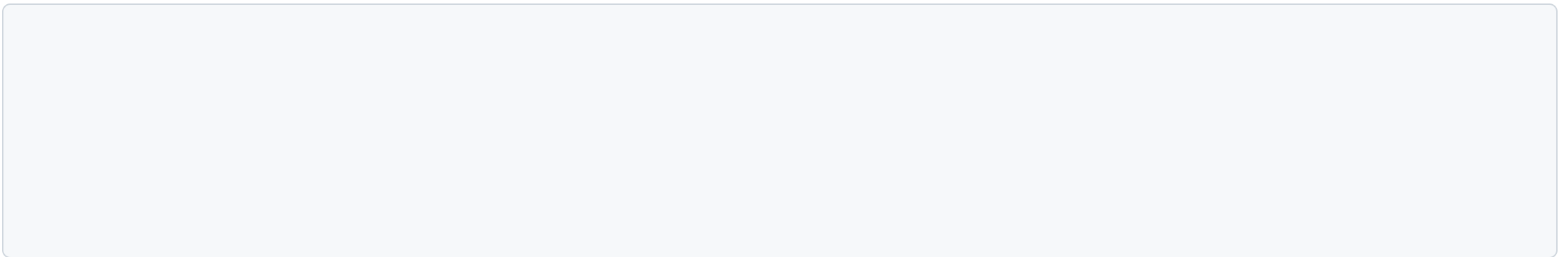


Resultado: Links alineados horizontalmente



Ejemplo NavBar: Paso 5 - Hover Interactivo

Estados hover:



✅ ¡NavBar completa e interactiva!

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

Variaciones de NavBar

Puedes personalizar:

? Pregunta Rápida

Para una navbar con logo izquierda y links derecha:

¿Qué justify-content uso?

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

Piensa unos segundos... 🤔



Respuesta

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 