IT Academy

by KIBERNUM

El Modelo de Cajas (Box Model)

📌 Propósito de la clase

En esta sesión, exploraremos el modelo de cajas en CSS y cómo afecta la organización y presentación de los elementos en una página web. Además, aprenderemos sobre los distintos tipos de posicionamiento y cuál es su impacto en la disposición del contenido. Finalmente, analizaremos los layouts más utilizados en el diseño web y sus ventajas y desventajas.

Al finalizar esta clase, serás capaz de:

✅ Comprender el modelo de cajas en CSS, identificando sus propiedades principales y cómo influyen en la estructura de los elementos.

✅ Diferenciar los tipos de cajas en CSS, reconociendo las diferencias entre elementos de bloque y línea y su impacto en el diseño.

✅ Inspeccionar y analizar elementos en el navegador, utilizando herramientas de desarrollo para visualizar márgenes, rellenos y bordes.

✅ Aplicar correctamente propiedades de posicionamiento, incluyendo estático, relativo, absoluto, fijo y flotante, para controlar la disposición de los elementos en una página.

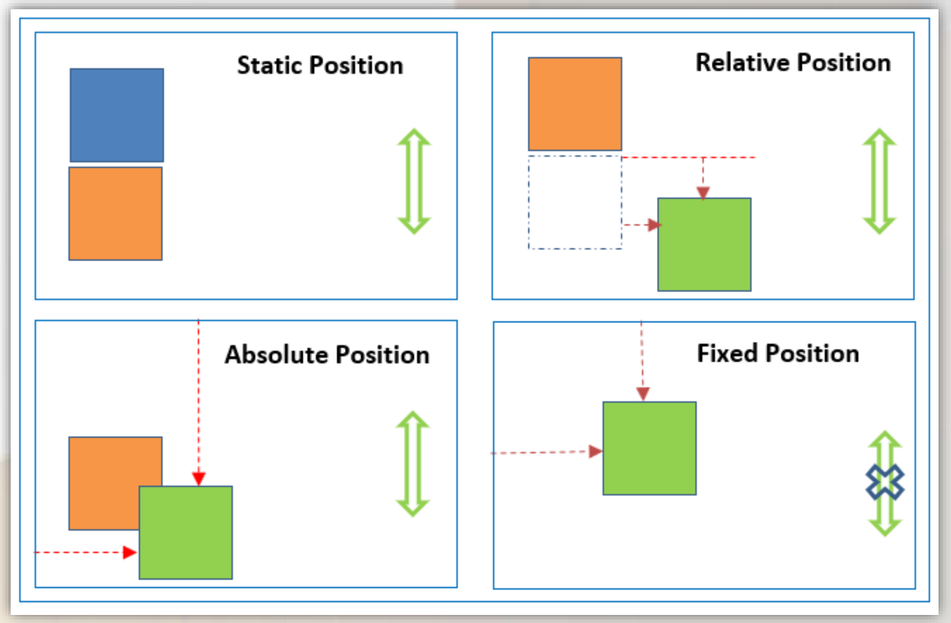
✅ Definir qué es un layout y sus tipos (fluido, fijo, elástico y absoluto), comprendiendo sus ventajas y desventajas en distintos escenarios de diseño.


Posicionamiento en CSS

📌 ¿Qué es el posicionamiento en CSS?

El posicionamiento en CSS es una de las claves para controlar con precisión dónde van a aparecer los elementos de tu página web. Cuando dices que es parecido a **“decidir dónde colocar los muebles en una habitación”,** no es solo una metáfora divertida: realmente define el espacio y la distribución en tu **“casa digital”.**

Para entenderlo mejor, imagina que cada elemento (una cabecera, un párrafo, una imagen, un botón) es un mueble distinto que quieres acomodar. El modelo de posicionamiento de CSS te ofrece distintas maneras de **“anclar”** esos muebles.



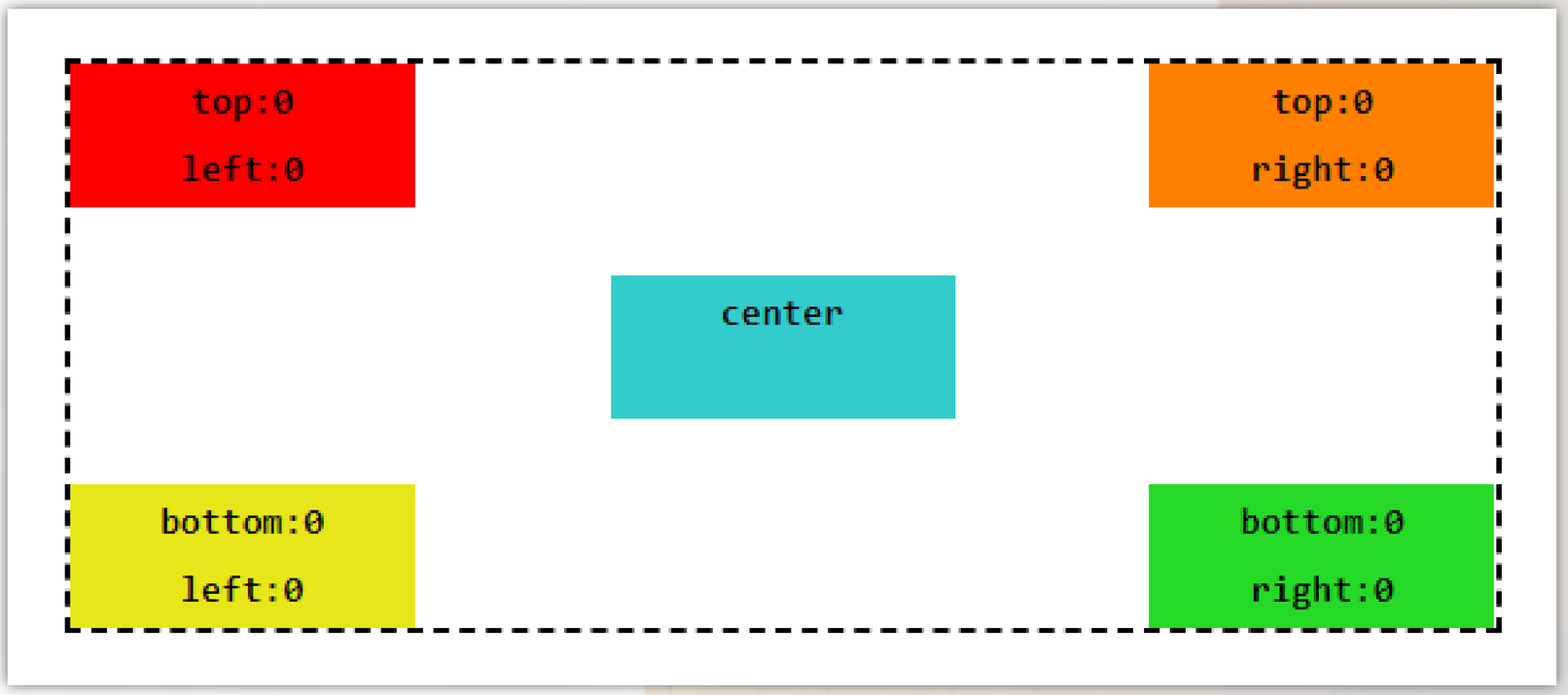
Fuente: https://www.csssolid.com/css-position.html

IT Academy
by KIBERNUM

📍 Coordenadas en el Posicionamiento CSS

El posicionamiento en CSS permite definir dónde aparecerán los elementos dentro de una página web. Para lograr esto, CSS utiliza un sistema de coordenadas, que se basa en cuatro propiedades principales:

- top (arriba)
- right (derecha)
- bottom (abajo)
- left (izquierda)



Fuente: https://dev.to/lupitacode/guia-completa-y-practica-sobre-posicionamiento-css-position-absolute-3ghn

Estas propiedades permiten mover elementos dentro del flujo de la página, dependiendo del tipo de posicionamiento (relative, absolute, fixed o sticky).

Cómo funcionan las coordenadas en CSS

Cuando un elemento tiene una propiedad de posicionamiento (position), las coordenadas (top, right, bottom, left) determinan su ubicación con respecto a otro elemento de referencia.

- Ejemplo Visual de Coordenadas en CSS

```css
.elemento {
    position: absolute;
    top: 50px;    /* Mueve el elemento 50px hacia abajo */
    left: 100px;  /* Mueve el elemento 100px hacia la derecha */
}
```

En este caso, el elemento se posicionará 50px desde la parte superior y 100px desde la izquierda del contenedor más cercano con posición relativa (relative, absolute o fixed).

🎯 Explicación de cada propiedad

Propiedad	Descripción	Ejemplo
top	Mueve el elemento desde la parte superior del contenedor de referencia.	top: 20px; moverá el elemento 20px hacia abajo.
right	Mueve el elemento desde el borde derecho del contenedor.	right: 10px; acercará el elemento al lado derecho.
bottom	Mueve el elemento desde la parte inferior del contenedor.	bottom: 30px; empujará el elemento hacia arriba.
left	Mueve el elemento desde el borde izquierdo del contenedor.	left: 50px; empujará el elemento hacia la derecha.

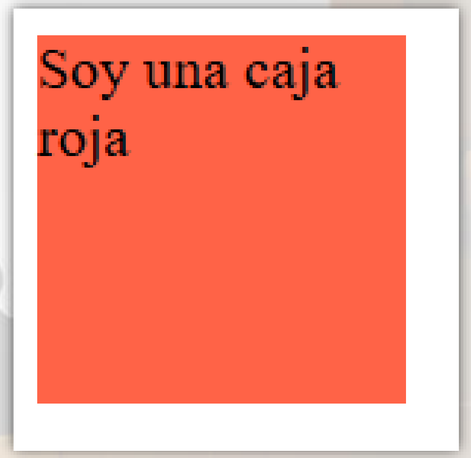
📌 Tipos de posicionamiento en CSS

🏠 1. Posicionamiento estático (static)

Este es el posicionamiento por defecto. Cuando no especificamos ningún tipo de position, el navegador coloca los elementos uno debajo del otro, como un texto en un documento.

Ejemplo:

```css
.caja {
  width: 100px;
  height: 100px;
  background: red;
}
```



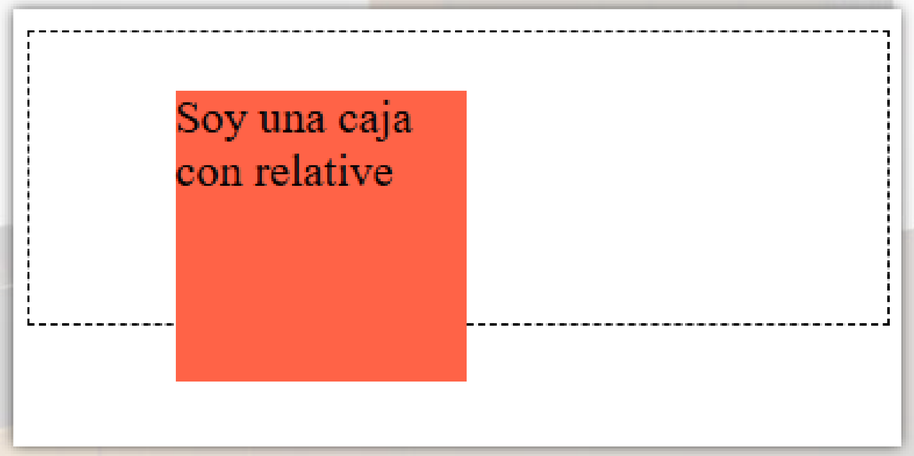

```html
<div class="caja">Soy una caja roja</div>
```

📌 Este div aparecerá en la página como cualquier otro texto, siguiendo el flujo normal del documento.

IT Academy
by KIBERNUM

🔍 Posicionamiento relativo (relative)

Un elemento con **position: relative; sigue ocupando su espacio normal**, pero podemos **moverlo** usando las propiedades **top**, **left**, **right** y **bottom**.



📌 **Reglas importantes:**

1. Se mueve **desde su posición original**, pero el espacio que ocupaba sigue reservado.
2. No afecta a los demás elementos de la página.
3. No se sale del flujo normal del documento.

```css
.container {
  border: 1px dashed;
}
.caja {
  width: 100px;
  height: 100px;
  background: tomato;
  position: relative;
  top: 20px;
  left: 50px;
}
```

```html
<div class="container">
  <div class="caja">Soy una caja con relative</div>
</div>
```

IT Academy
by KIBERNUM

✅ Ejemplo visual:

```css
.container {
  border: 1px dashed;
}

.caja {
  width: 100px;
  height: 100px;
  background: blue;
  margin-bottom: 10px;
}

.caja-relativa {
  position: relative;
  top: 30px;
  left: 20px;
}
```

Si tienes 3 cajas:

```html
<div class="caja">Caja 1</div>
<div class="caja caja-relativa">Caja 2 (relative)</div>
<div class="caja">Caja 3</div>
```

🎯 **La segunda caja (caja-relativa) se mueve, pero Caja 1 y Caja 3 siguen en su sitio.**

IT Academy
by KIBERNUM

🎯 Posicionamiento absoluto (absolute)

Cuando usamos **position: absolute; , la caja desaparece de su posición normal y se coloca en una nueva posición basada en su ancestro posicionado**.

📌 **Reglas importantes:**

1. Si el elemento **no encuentra un padre con position: relative o absolute**, se posiciona en relación con el **<body>,** es decir, **la página entera**.

2. Se mueve usando **top, left, right** y **bottom**, pero **NO deja espacio en su posición original**.

Ejemplo (Sin padre posicionado):

styles.css > ...

```css
.container {
  border: 1px dashed;
  width: 320px;
  height: 140px;
}

.caja {
  width: 100px;
  height: 100px;
  background: tomato;
  position: absolute;
  top: 20px;
  left: 270px;
}
```

```html
<body>
  <div class="container">
    <div class="caja">Caja con Absolute</div>
  </div>
</body>
```

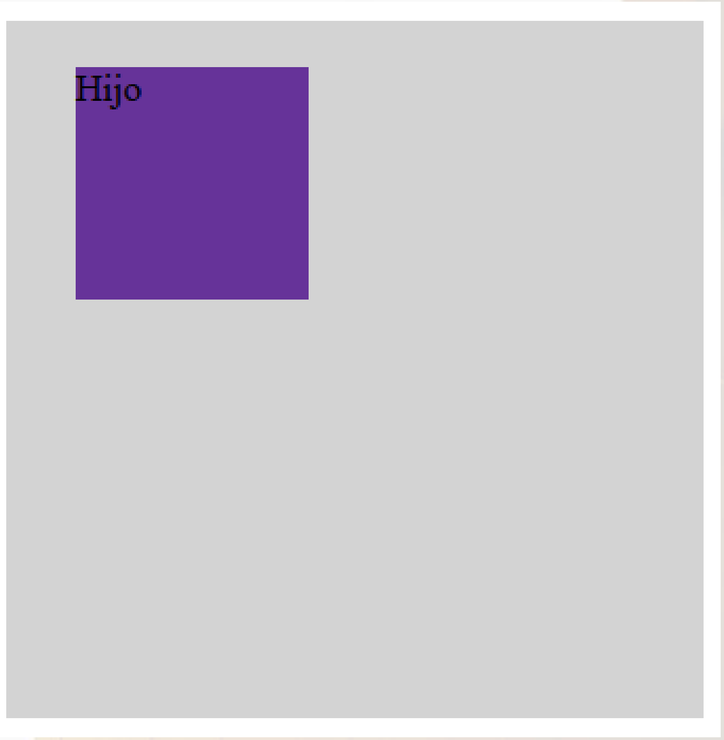
IT Academy
by KIBERNUM

Ejemplo (Con padre posicionado):

Si un **div** padre tiene **position: relative;**, el hijo **absolute** se posicionará en **relación con ese padre**.

```css
.padre {
  width: 300px;
  height: 300px;
  background: lightgray;
  position: relative;
}

.hijo {
  width: 100px;
  height: 100px;
  background: rebeccapurple;
  position: absolute;
  top: 20px;
  left: 30px;
}
```




```html
<div class="padre">
  <div class="hijo">Hijo</div>
</div>
```

🎯 **Ahora la caja roja (hijo) se mueve dentro de la caja gris (padre) y NO en la página entera.**

📌 4. Posicionamiento fijo (fixed)

Cuando un elemento tiene **position: fixed**; , **se queda en la misma posición sin importar si hacemos scroll**.

📌 **Reglas importantes:**

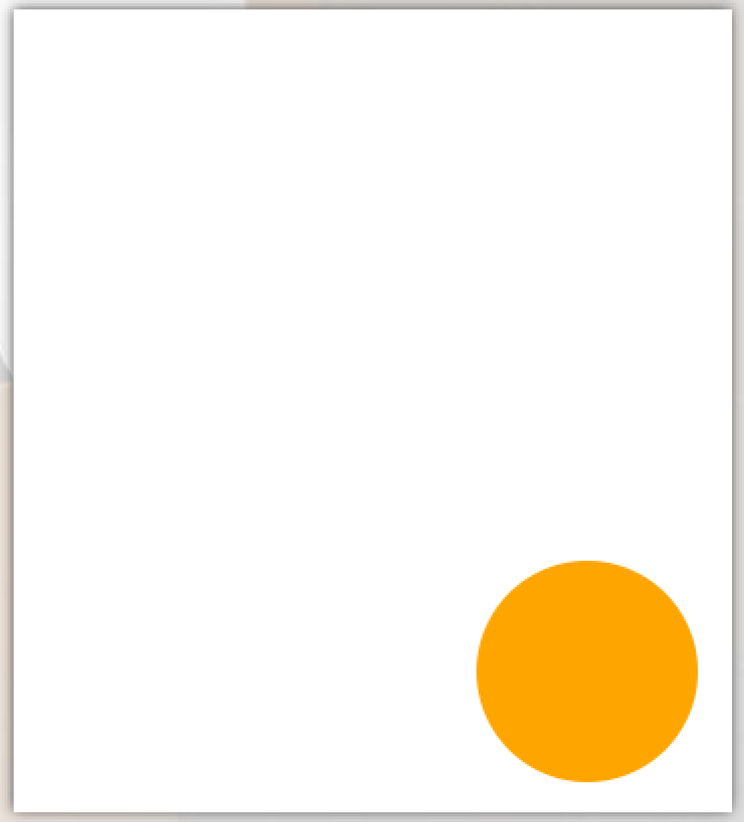
1. Se posiciona **en relación con la ventana del navegador**, NO con el **<body>** ni con ningún otro elemento.
2. No se mueve, aunque hagamos scroll.
3. Se puede usar para **barras de navegación fijas** o botones flotantes.

```css
.caja {
  width: 100px;
  height: 100px;
  background: orange;
  position: fixed;
  bottom: 20px;
  right: 20px;
}
```

```html
<div class="caja"></div>
```

Esta caja naranja siempre estará en la esquina inferior derecha de la pantalla, aunque hagamos scroll.

✅



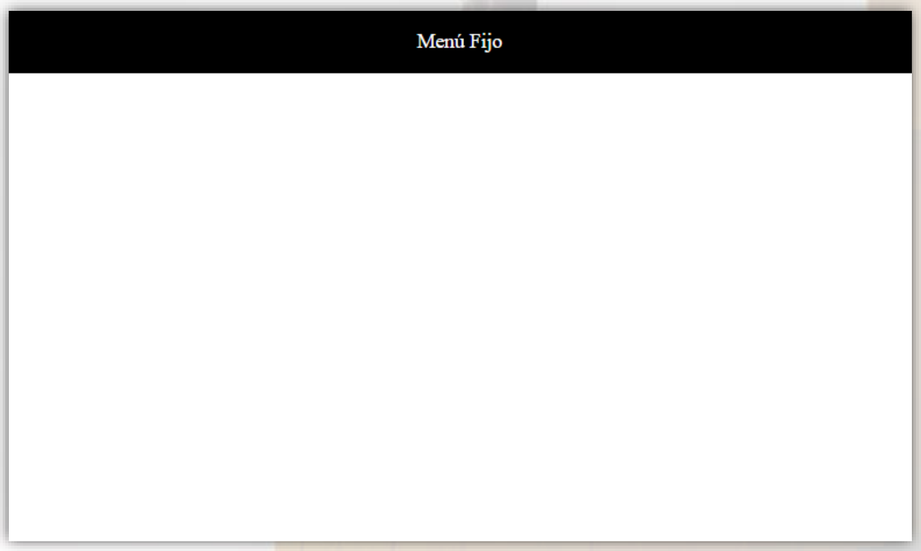
IT Academy
by KIBERNUM

📌 4. Posicionamiento fijo (fixed)

Cuando un elemento tiene **position: fixed**; , **se queda en la misma posición sin importar si hacemos scroll**.

```css
.nav {
  width: 100%;
  height: 50px;
  background: black;
  color: white;
  position: fixed;
  top: 0;
  left: 0;
  text-align: center;
  line-height: 50px;
}
```

```html
<div class="nav">Menú Fijo</div>
```



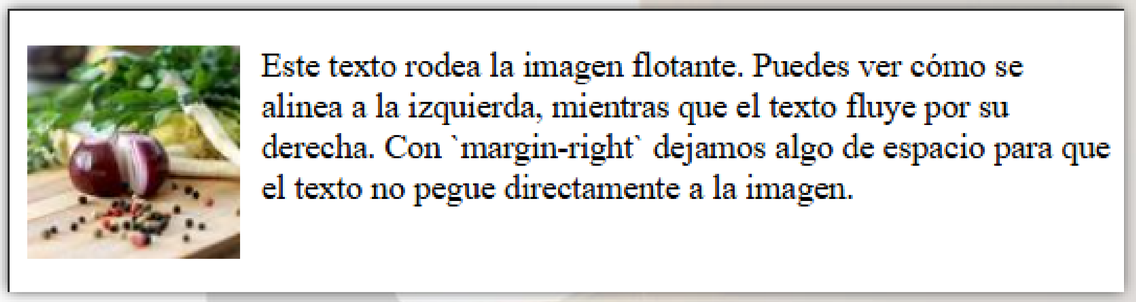
✅ **Este menú siempre estará en la parte superior, incluso si bajamos la página.**

5. Posicionamiento flotante (float)

Este es un tipo de posicionamiento antiguo que permitía que un elemento se alineara a la izquierda o derecha y que otros elementos se acomodaran a su alrededor. Hoy en día, **flexbox** y **grid** son más recomendados.

📌 **Reglas importantes:**

1. float: left; coloca el elemento a la izquierda.
2. float: right; coloca el elemento a la derecha.
3. El contenido siguiente fluirá alrededor del elemento flotante.




```html
<div class="img">
  <img src="https://picsum.photos/100" alt="Imagen aleatoria">
</div>

<p>
  Este texto rodea la imagen flotante. Puedes ver cómo se alinea a la izquierda,
  mientras que el texto fluye por su derecha. Con `margin-right` dejamos algo
  de espacio para que el texto no pegue directamente a la imagen.
</p>
```

```css
.img {
  float: left;
  margin-right: 10px;
}

.img img {
  width: 100px;
  height: 100px;
  object-fit: cover; /* Ajusta la imagen para que se vea completa */
  display: block;   /* Asegura que la imagen se comporte como bloque */
}
```

Tabla Comparativa

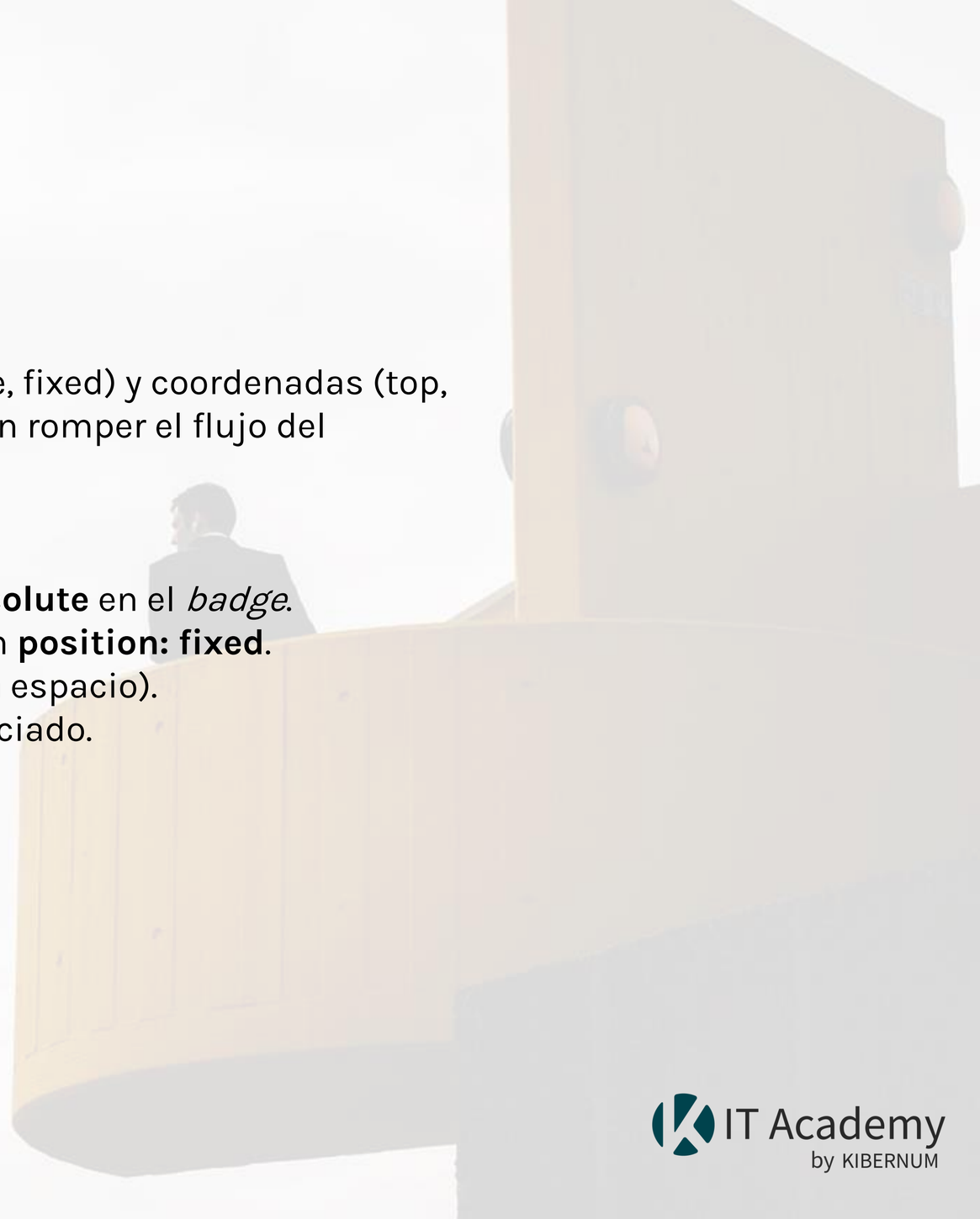
Tipo de Posicionamiento	Ocupa su espacio normal	Se mueve con top, left, etc.	Se posiciona con respecto a
static (default)	✅ Sí	❌ No	Flujo normal
relative	✅ Sí	✅ Sí	Su posición original
absolute	❌ No	✅ Sí	Su padre posicionado o <body>
fixed	❌ No	✅ Sí	La ventana del navegador
float	❌ No	❌ No	Se alinea a la izquierda o derecha

Reto: Badge absoluto + botón fijo

Objetivo
Aplicar **posicionamiento en CSS** (static, relative, absolute, fixed) y coordenadas (top, right, bottom, left) para ubicar elementos con precisión sin romper el flujo del documento.

Requerimientos técnicos
- Usar **position: relative** en el contenedor y **position: absolute** en el *badge*.
- Agregar un **botón fijo** en la esquina inferior derecha con **position: fixed**.
- Controlar el **flujo del documento** (qué deja/qué no deja espacio).
- Usar **Box Model** (márgenes, rellenos, bordes) para espaciado.
- Solo **HTML + CSS** (sin librerías ni JS).

Tiempo estimado
30–40 min (individual o parejas).



IT Academy
by KIBERNUM

Reto: Badge absoluto + botón fijo

HTML base

```html
<main class="page">
  <article class="card">
    <img class="card__img" src="https://picsum.photos/640/360" alt="Imagen de ejemplo">
    <span class="card__badge">-20%</span>
    <div class="card__body">
      <h2 class="card__title">Producto Demo</h2>
      <p class="card__text">Descripción breve del producto. Incluye detalles esenciales.</p>
      <a class="card__cta" href="#">Agregar al carro</a>
    </div>
  </article>

  <!-- Botón de ayuda fijo (debe quedar siempre visible al hacer scroll) -->
  <button class="help-btn" type="button" aria-label="Abrir ayuda">¿Ayuda?</button>
</main>
```

Instrucciones:

1. Posicionar el **badge** en la **esquina superior izquierda** de la imagen, dentro de la tarjeta.
2. Dejar la **tarjeta** en el flujo normal (que ocupe su espacio).
3. Hacer que el **botón de ayuda** quede **fijo** abajo a la derecha de la ventana.

Preguntas para discutir

1. ¿Qué diferencia práctica hay entre **relative** y **absolute** respecto al **flujo del documento**?
2. Si un elemento absoluto **no** encuentra ancestro posicionado, ¿respecto a qué se ubica? ¿Por qué?
3. ¿Cómo interactúan las coordenadas top/left con el **Box Model** (márgenes y padding del contenedor)?
4. ¿Cuándo conviene usar **fixed** en lugar de **absolute**? Da un ejemplo real.
5. ¿Qué problemas comunes aparecen al usar z-index? ¿Cómo los evitarías?

IT Academy

by KIBERNUM