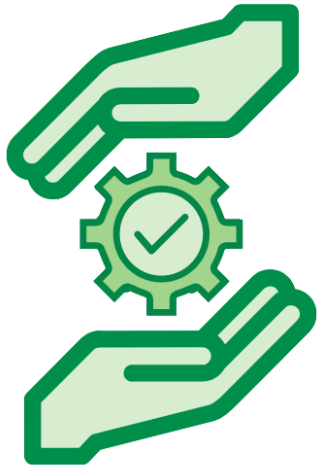


Bootcamp de Desarrollo Web

Clase 7

CSS

¿Qué es BEM?

Es una metodología que te ayuda a crear componentes reutilizables y compartir código en el desarrollo de **front-end**.

Es decir, es una manera de nombrar las clases de los elementos **HTML** para crear estilos **CSS** de una manera fácil, sencilla y clara.

El objetivo de BEM es dar mucha más **transparencia** y **claridad** en tu estructura HTML y CSS.

¿Significado de BEM?

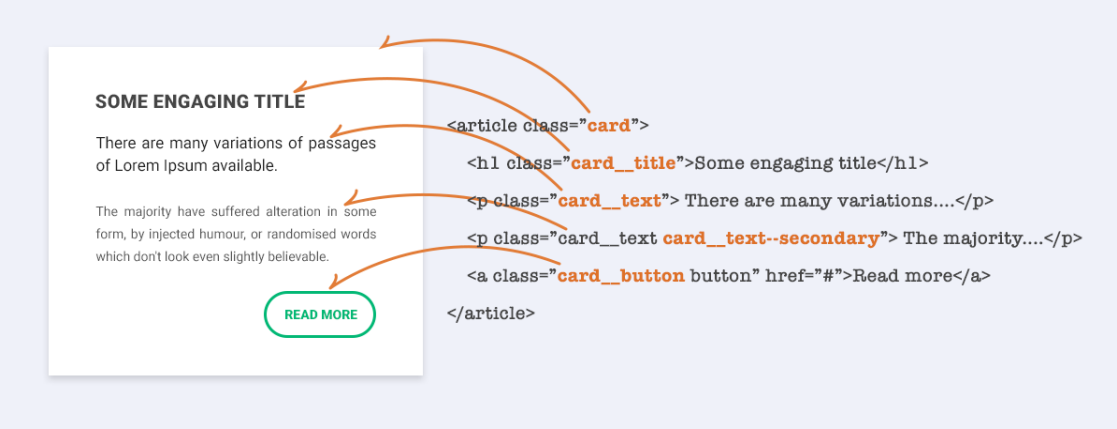
BEM dice cómo se relacionan las clases entre sí, lo que es útil en secciones complejas del documento.

BEM consta de tres siglas.

B de bloque.

E de elemento.

M de modificador.



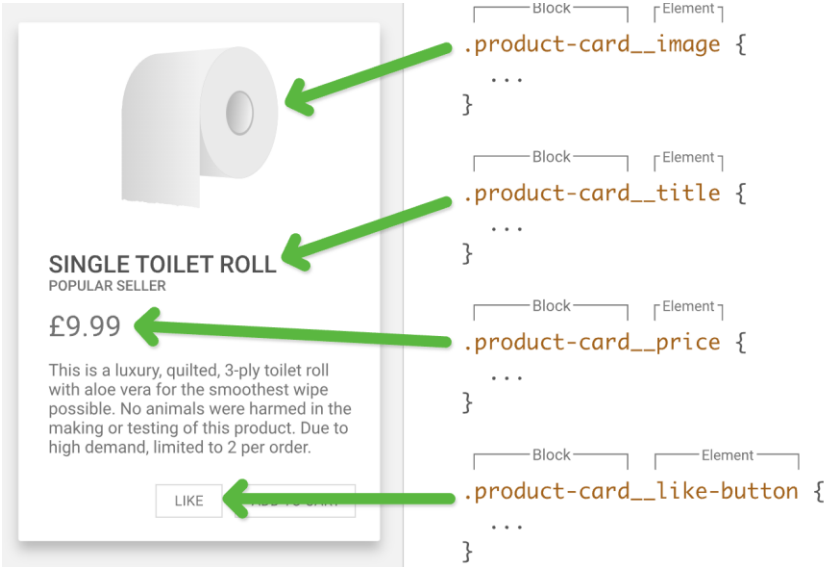
Block (bloque)

Un bloque es una parte componente de un sitio web que puede existir por sí mismo. Estamos utilizando una tarjeta de producto como ejemplo de bloque en este artículo.

Otros ejemplos de un bloque son el encabezado y pie de página de un sitio web, etc.

Element (elemento)

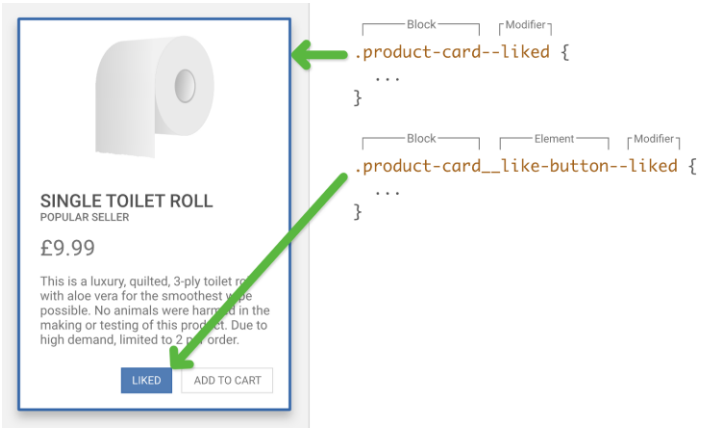
Un elemento es un aspecto secundario de un bloque que solo existe dentro de ese bloque. En nuestro ejemplo de la tarjeta de producto, la imagen, el título, el precio y el botón de "me gusta" son ejemplos de elementos dentro del bloque de la tarjeta de producto.



Modifier (modificador)

Un modificador puede aplicarse tanto a un bloque como a un elemento y se utiliza para ajustar el aspecto o comportamiento predeterminado del bloque o del elemento.

En nuestro ejemplo de la tarjeta de producto, el modificador liked existe y puede aplicarse tanto al bloque de la tarjeta de producto como al elemento del botón de me gusta cuando un producto ha sido marcado como gustado.



Beneficios de BEM

Modularidad: Los estilos de los bloques nunca dependen de otros elementos en una página, por lo que nunca experimentarás problemas de cascada.Además, tienes la capacidad de transferir bloques de tus proyectos terminados a nuevos.

Reutilización: Componer bloques independientes de diferentes maneras y reutilizarlos de manera inteligente reduce la cantidad de código CSS que tendrás que mantener.

Con un conjunto de pautas de estilo establecidas, puedes construir una biblioteca de bloques, haciendo tu CSS sumamente efectivo.

Beneficios de BEM

Estructura: La metodología BEM brinda a tu código CSS una estructura sólida que sigue siendo simple y fácil de entender.

Nombres consistentes para selectores CSS: Cada desarrollador debería poder entender rápidamente dónde se usa cada clase de CSS y comprender cómo deberían nombrar nuevas clases.

Menos estilos rotos inesperados: Como las clases de BEM siempre apuntan a un bloque, teóricamente, cualquier cambio de estilo relacionado con ese bloque no afectará a otros bloques.

Repaso de tipos de selector

- Selector de tipo o etiqueta: h1, div, p
- Selector de clase : .card, .button
- Selector de id : #titulo, #menu
- Selector universal (*)
- Selector de atributo [~=, | , *, ^, $]

Combinadores

Hay cuatro tipos de Combinadores en CSS3:

- Selector Descendente (espacio)
- Selector de hijo directo (>)
- Selector de elemento adyacente (+)
- Selector basado en elemento precedente con el mismo padre(~)

Selector Descendente

Este selector afecta a todos los elementos que sean descendientes de otro elemento especificado, sin importar el nivel de profundidad al que se encuentren. Por ejemplo la siguiente regla afectaría a todos los elementos **<p>** dentro de el elemento ** ** con la clase "contenido" y se indica poniendo un espacio entre un selector y el otro:

```
1
2 <div class="contenido">
3    <p>Párrafo 1</p>
4    <div class="noticias">
5       <p>Párrafo 2</p>
6       <p>Párrafo 3</p>
7    </div>
8 </div>
9
```

```
1
2 div.contenido p{
3    color: red;
4 }
5
```

Selector de hijo directo

Este selector actúa sobre todos aquellos elementos que sean hijos de otro elemento especificado, pero que se encuentren en el primer nivel, es decir, si están dentro de otro elemento hijo de ese mismo padre, no serán tomados en cuenta.

```
<div class="contenido">
    <p>Párrafo 1</p>
    <div class="noticias">
        <p>Párrafo 2</p>
        <p>Párrafo 3</p>
    </div>
</div>
```

```
div.contenido > p{
    color: red;
}
```

Selector de elemento adyacente

Este selector afecta a los elementos que, teniendo el mismo elemento como padre, estén inmediatamente seguidos uno de otro, esta relación se representa con el símbolo + entre los selectores. Con el mismo ejemplo anterior, la siguiente regla afectaría solo al segundo elemento <p> que contiene el texto Párrafo 3 por estar inmediatamente después de otro <p> hijo del mismo padre que él.

```
<div class="contenido">
    <p>Párrafo 1</p>
    <div class="noticias">
        <p>Párrafo 2</p>
        <p>Párrafo 3</p>
    </div>
</div>
```

```
div.noticias p + p {
    color: red;
}
```

Selector basado en elemento precedente con mismo padre

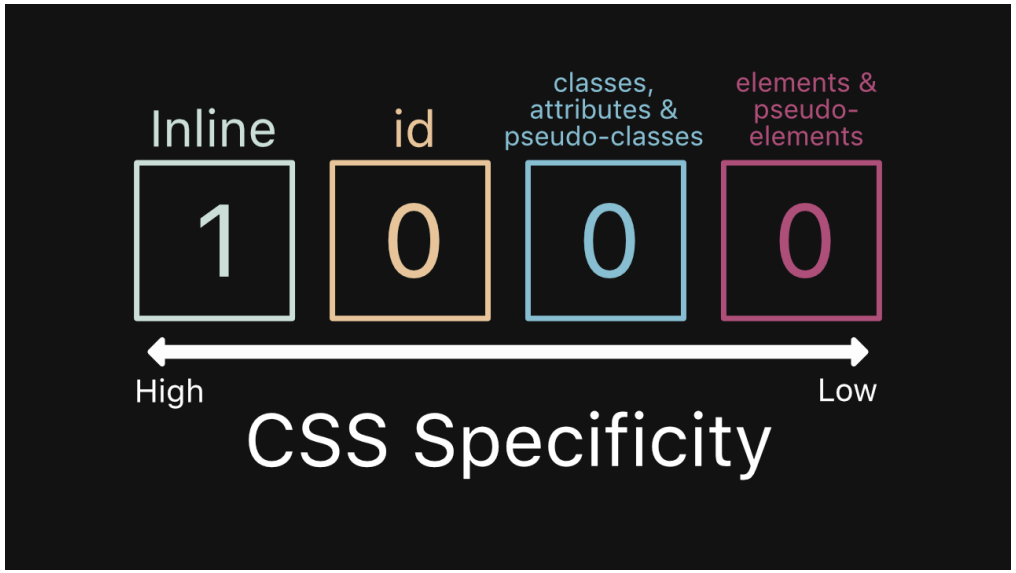
Este selector actúa sobre aquellos elementos que se encuentren precedidos por un elemento específico y que tengan como padre al mismo elemento, y se representa con el símbolo ~ entre los dos selectores.

```
<div class="contenido">
   <p>Párrafo 1</p>
   <div class="anuncios">
    <a href="publicidad.html" >Link a publicidad</a>
   </div>
   <div class="noticias">
      <p>Párrafo 2</p>
      <p>Párrafo 3</p>
   </div>
</div>
```

```
p ~ div {
  margin-bottom: 20px;
  border: 1px solid blue;
}
```

¿Qué es la especificidad?

La especificidad de CSS es un conjunto de reglas usadas por los navegadores para determinar cuál de los estilos definidos por el desarrollador se aplicará a un elemento específico.



Evaluación de especificidad

Para que un estilo se aplique a un elemento en particular, el desarrollador tiene que cumplir con las reglas, de modo que el navegador sepa cómo aplicar el estilo.

El navegador evalúa la especificidad en dos partes:

- Los estilos aplicados en línea en el HTML (Inline Styles).
- Los estilos aplicados mediante un selector (**Especificidad del selector**).

Jerarquía de la especificidad

Piense en la especificidad como un score/rank que determina qué declaraciones de estilo se aplican finalmente a un elemento.

La especificidad utiliza un sistema de ponderación o puntuación.

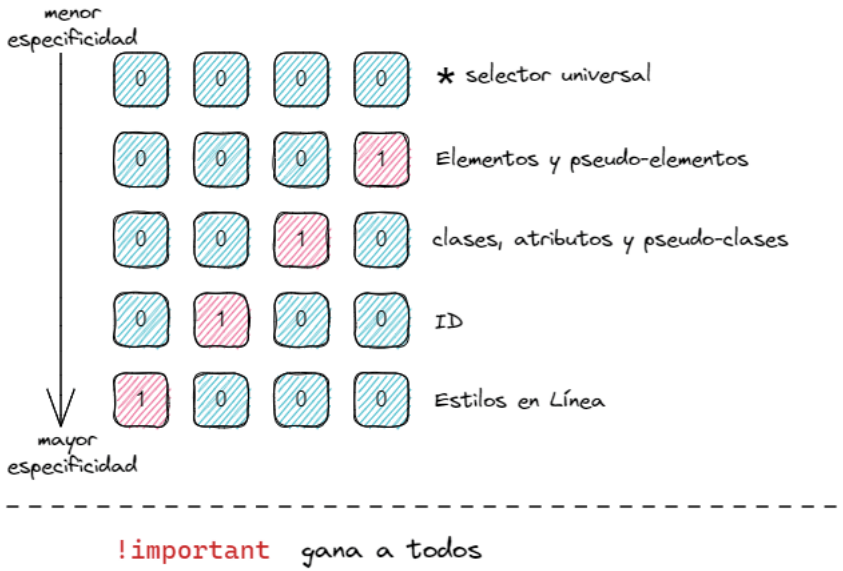
Cada tipo de selector recibe puntos que indican su especificidad, y se suman los puntos de todos los selectores que hayas utilizado. para calcular la especificidad total del selector.

Cada selector tiene su lugar en la jerarquía de especificidad. Hay cuatro categorías que definen el nivel de especificidad de un selector.

Jerarquía de la especificidad

Puntuación de cada tipo de selector

La cantidad de especificidad de un selector se mide usando cuatro valores diferentes separados por coma que tienen diferente peso y pueden describirse como millares, centenas, decenas y unidades (cuatro dígitos individuales dispuestos en cuatro columnas):



Selector Universal

El selector con menor peso (y menos valor) es el selector universal, que en CSS se define de la siguiente forma:

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

Este selector en CSS tiene un valor de **0-0-0-0**, eso quiere decir que cualquier otro selector cuyo valor sea **1-0-0-0**, **0-1-0-0** o incluso **0-0-0-1** tendrá mayor peso que el y lo reescribirá.

</> Elementos y ::pseudoelementos

Los selectores de tipo o de etiqueta: <p>, ,<main> etc.

Tiene un valor de **0-0-0-1**

```
<header>
  <h1><span>Welcome</span> to my youtube channel</h1>
</header>
```

```
header h1 span {
    text-transform: lowercase;
}
span {
  text-transform:uppercase;
}
```

El elemento solamente tiene un peso de **0-0-0-1** pero también tenemos en la primera regla 3 etiquetas HTML que tienen un peso total de **0-0-0-3**, ¿Por que 3? La respuesta es que el **3** es la cantidad de elementos HTML declarados, por lo que todo el texto estará en minúsculas a pesar de que el tenga mayúscula.

.clases, atributos[...] y :pseudoclases

El selector de **clase** tiene mayor peso (o especificidad) que los selectores de tipo o elementos HTML y que el selector universal, seguiremos con el ejemplo anterior pero ahora con una clase llamada title en la etiqueta :

```
<header>
  <h1><span class="title">Welcome</span> to my youtube channel</h1>
</header>
```

```
header h1 span {
    text-transform: lowercase;
}
.title {
  text-transform:uppercase;
}
```

Ahora el texto del elemento si estará en mayúscula y todo lo demás estará en minúscula, esto sucede porque las clases tienen un peso de **0-0-1-0** mientras que el selector de 3 elementos será de **0-0-0-3**.

Selector de id

Este es de los selectores más específicos en este caso el ID tendrá un peso de 0-1-0-0.

Esto significa que un selector de ID tiene una mayor especificidad que un selector de clase.

```
<header id="header" class="header"> Lupita Code </header>
```

```
#header {
  background-color: rebeccapurple;

.header {
   background-color: khaki;
}
```

Estilos en línea

Los estilos inline añadidos a un elemento HTML siempre sobrescriben a cualquier estilo escrito en hojas de estilo externas o una etiqueta <style>, por lo que se puede decir que tienen la mayor especificidad. Los estilos en línea no tienen selector porque se aplican directamente al elemento al que se dirigen.

Los estilos en línea tienen un valor de **1-0-0-0**.

```
<li>
  <a href="/specials" class="featured"
style="background-color: orange;">
     Specials
 </a>
</li>
```

!important con la especificidad?

Para anular los estilos inline, tendrás que añadir un **!important** a la declaración de la hoja de estilos externa. Si los estilos en línea están marcados como **!important**, entonces nada puede anularlos.

!important es solo para casos desesperados y NO se recomienda usarlo ya que si comienzas a agregar **!important** en varios lugares de tus hojas de estilos te será más complicado debuggear o reescribir algún elemento.

Excepciones

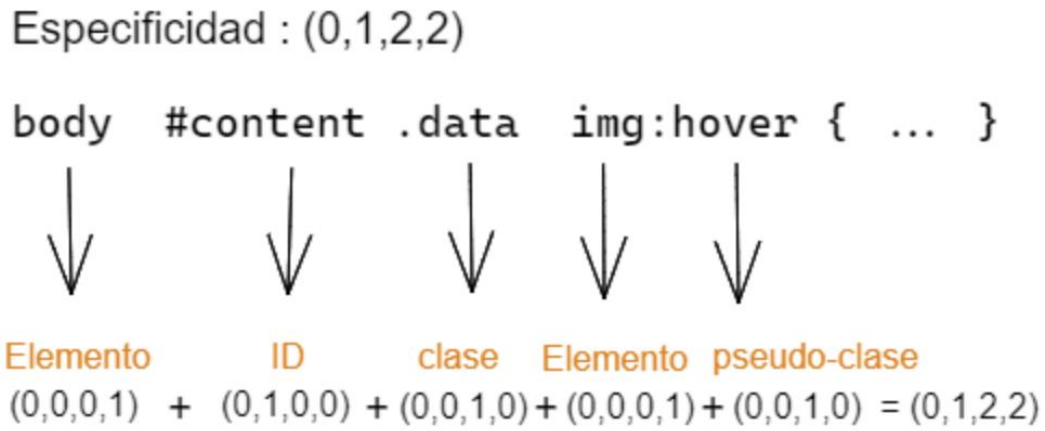
El selector universal **(*)**, los combinadores **(+, >, ~, etc)** y la pseudo-clase de negación **:not()** no tienen efecto sobre la especificidad. (Sin embargo, los selectores declarados dentro de **:not()** si lo tienen). Por ejemplo:

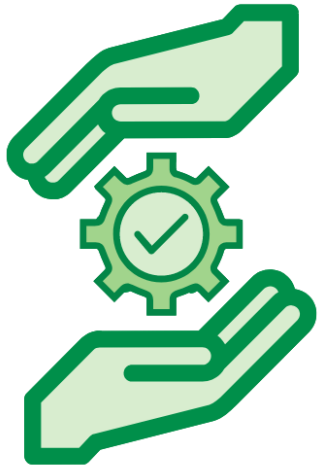
```
div.outer p {
  color:orange;
}
div:not(.outer) p {
  color: lime;
}
```

```
<div class="outer">
  <p>Esto está en el outer div.</p>
  <div class="inner">
    <p>Este texto está en el inner div.</p>
  </div>
</div>
```

Diferentes tipos de selectores, diferente especificidad

Cuando usamos diferentes tipos de selectores combinados (por ejemplo,body #content .data img:hover{}) el navegador contará el número de ids, pseudo-clases y pseudo-elementos y asignará un valor de especificidad a la regla, para compararla con las demás y decidir cuál usar. Por ejemplo:





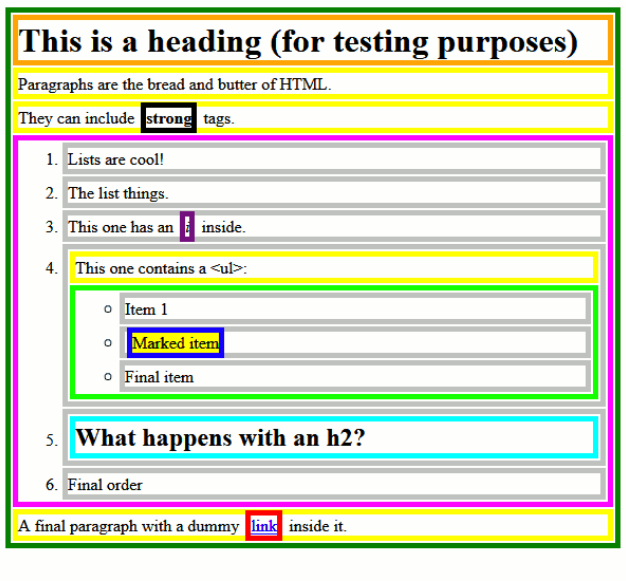
Bootcamp de Desarrollo Web

Clase 6

Posicionamiento CSS

¿Qué es el flujo normal del HTML?

El flujo normal (normal flow) en HTML es el orden natural en el que los elementos aparecen en pantalla, es decir, los elementos aparecerán colocados tal como estén ordenados en el código HTML solo si no se aplica ningún CSS que cambie la forma en la que se comportan.



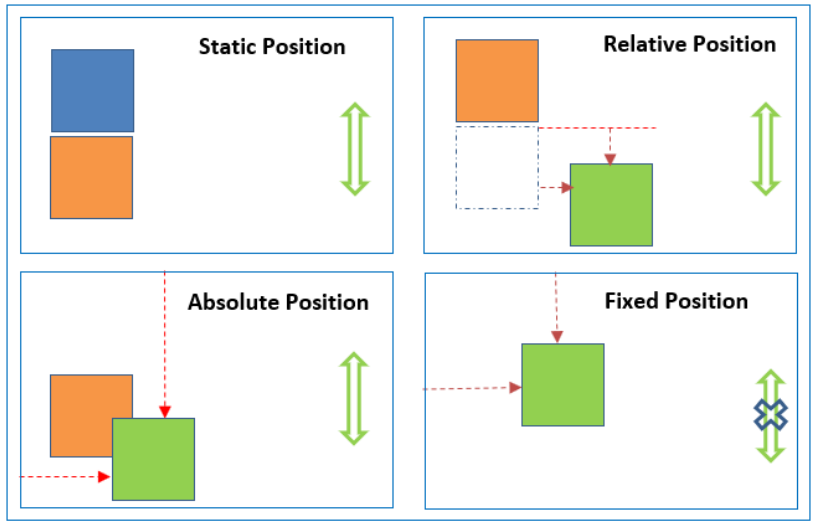
¿Qué es un elemento posicionado?

Un elemento posicionado es aquel elemento que ha salido de su **flujo normal** a través de la propiedad position, además adquiere nuevas propiedades.

La propiedad **position** establece en que punto de la pagina comenzara a posicionarse, mostrarse o dibujarse el elemento que se haya establecido en el código HTML.

Posición default

Es importante saber desde que punto se va a comenzar a pintar porque eso es lo que dirá que espacio ocupara cada elemento, por defecto si no se especifica, los elementos se crean con la propiedad **position** y el valor **static**, los elementos comenzaran a pintarse desde la esquina superior izquierda del elemento padre (0 x 0).



Valores pe la propiedad position

- **static:** La forma por defecto, obedece al flujo normal de la página
- **relative:** Establece que la posición de un elemento depende de otro
- **absolute :** Indica que la posición de un elemento no depende de otro
- **fixed:** Permite fijar un elemento en una posición determinada
- **sticky:** Es una combinación entre **relative** y **fixed**

Ejes de posicionamiento

Al tener un elemento posicionado podemos moverlo en los 3 ejes y corresponden a cinco propiedades:

Eje X:

right: mover el elemento desde la parte derecha hacia la izquierda

left: mover el elemento desde la parte izquierda hacia la derecha

Ejes de posicionamiento

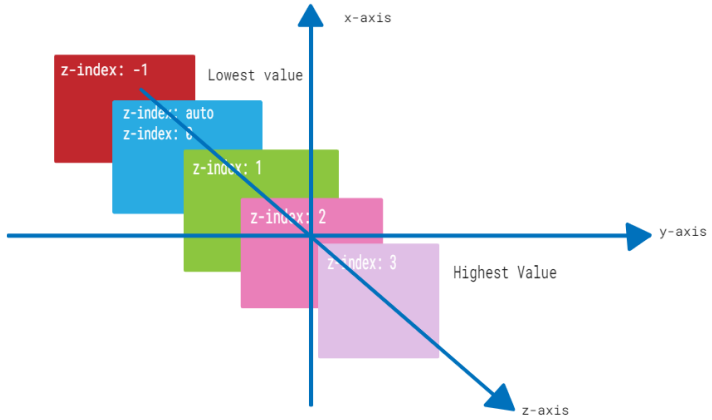
Eje Y:

- **top :** mover el elemento desde la parte superior hacia la inferior
- **bottom:** mover el elemento desde la parte inferior hacia la superior

Ejes de posicionamiento

Eje Z:

- **z-index:** cuando dos o mas elementos se solapan, podemos decidir cual aparece primero y cual por detrás de el)



Información adicional

⚠️ Las propiedades **top**, **right**, **bottom**, **left** y **z-index** no funcionaran y no serán habilitadas para los elementos con posicionamiento estático por lo tanto los elementos no se podrán mover o desplazar.

En la propiedad **z-index** solo se especifica un numero entero positivo/negativo, no se usa unidades tales como pixeles o porcentajes. La propiedad **z-index** toma un valor numérico entre 0 y ±2147483647 en la mayoría de los navegadores comunes.

Es recomendable no usar valores consecutivos como: 1,2,3,4...

Contexto

Un contexto o también llamado punto de referencia es un área, podemos verlo como un rectángulo imaginario a través del cual los elementos van a poder posicionarse, moverse, alinearse o distribuirse.

Cuando hablamos de posicionamiento, el contexto es la posición inicial del elemento (donde esta originalmente) y desde el cual se calcula hacia donde se va a mover el elemento posicionado cuando colocamos ya sea **bottom**,**right**,**left**,**top**.