# CSS

**¿Qué significa CSS?**

CSS es el acrónimo para Cascading Style Sheets, que traducido al español es Hojas de Estilo en Cascada.

Complementa el lenguaje HTML.

El término cascada es porque se aplican de arriba hacia abajo siguiendo un patrón al que se le denomina "herencias".

**¿Qué es el lenguaje CSS?**

En términos simples, el lenguaje CSS es el que describe cómo nuestros elementos en HTML serán mostrados en una pantalla de computadora, celular u otro dispositivo multimedia.

Las hojas de estilos CSS son de gran ayuda y ahorran mucho trabajo al momento de programar un sitio web, ya que pueden controlar el aspecto de múltiples páginas dentro del sitio a la vez con una sola hoja de estilos o documento de CSS.

Entonces, a través del CSS tendrás la capacidad de determinar el diseño, color, fuente, entre otras características de tus elementos.

**Diferencias entre HTML y CSS**

La diferencia crucial entre los dos es que el HTML se utiliza para la creación de las páginas web y el CSS se utiliza para controlar el estilo y el diseño de las páginas web.

En HTML, primero se escriben palabras y luego se añaden elementos o etiquetas, que luego aparecen en la página. De este modo, el navegador conoce el título de la página, el comienzo y el final del párrafo, etc.

En CSS, las reglas se usan utilizando las propiedades CSS, las cuales se clasifican generalmente en dos categorías principales.

- La primera es la presentación, que especifica el color del texto, el tipo de fuente, el tamaño de la fuente, los colores de fondo, las imágenes de fondo, etc.
- Y la segunda es la presentación, que define la posición de los distintos elementos en la pantalla.

**Diferencias clave**

- HTML es el lenguaje de marcado básico que describe el contenido y la estructura de las páginas web. Por otro lado, CSS es la extensión del HTML que modifica el diseño y visualización de las páginas web.

- El archivo HTML puede contener código CSS, mientras que las hojas de estilo CSS nunca pueden contener código HTML.
- HTML se compone de etiquetas que rodean el contenido, mientras que CSS se compone de selectores que van seguidos de un bloque de declaraciones.

**Ventajas y Desventajas del Lenguaje CSS**

**Ventajas**

- Unificar todo lo referente al diseño visual en un solo documento
- Se pueden hacer modificaciones en un solo lugar sin tener que recurrir a los archivos HTML por separado
- Es menor la probabilidad de que exista duplicación de estilos en diferentes lugares, debido a esto es más fácil de organizar y hacer cambios. A esto debemos añadir que la información a transmitir es considerablemente menor y por tanto las páginas también se descargan más rápido
- La creación de versiones para otros dispositivos: tablets o smartphones es muy simple.

**Desventajas**

Compatibilidad con algunos buscadores.

## ¿Cuál es la estructura y sintaxis de CSS?

- **Regla:** Cada regla está compuesta de una parte llamada "selectores", un símbolo de "llave de apertura" ({), otra parte denominada "declaración" y por último, un símbolo de "llave de cierre" (}).
- **Selector:** indica el elemento o elementos HTML a los que se aplica la regla CSS. El selector indica "a quién hay que hacérselo“
- **Declaración:** especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS. La declaración indica "qué hay que hacer"
- **Propiedad:** característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color de fondo, etc.
- **Valor:** establece el nuevo valor de la característica modificada en el elemento.


Para hacer la declaración de estilo correctamente es necesario que respetemos los signos utilizados y su orden. Por ejemplo, para asignarle color rojo al texto de un botón, la declaración de CSS se vería así:

```
button {
    color: red;
}
```

**Que son los selectores:**

Los selectores CSS son herramientas utilizadas para definir el estilo que quieres dar a tus elementos en CSS. Existen muchos tipos de selectores en este lenguaje, cada uno con su propia sintaxis y utilidad. El uso de las reglas de programación correctas ayuda al explorador a aplicar propiedades específicas a los elementos indicados.

**¿Cómo aplicar los selectores de CSS?**

Existen tres formas en las que se pueden incluir los selectores de CSS en una hoja de estilos en una página web:

1. **Inline Styles** o  Estilo de Línea: Los cuales se aplican directamente en un elemento HTML, en la que se adiciona el atributo style=”propiedad:valor;?” en la etiqueta de apertura de ese elemento. Por ejemplo:

```
<p style="color:green;">Texto en verde</p>
```

2. **Etiqueta** <style>: Dentro del documento HTML se pueden aplicar los estilos generales

haciendo uso de la etiqueta de estilo (<style>). Esta se agrega en el encabezado de la página (<head></head>) y aplica para todo el contenido dentro de esta o los elementos especificados. Por ejemplo, si queremos definir el estilo de los textos h1 haremos algo como esto:

```
<head>
    <style type="text/css">
        h1 {
        color: red;
        }
    </style>
</head>
```

3. **Hoja de estilo externa**: Este método es el más recomendado para sitios web complejos o con muchas secciones y elementos ya que de esta forma podrás manipular tus estilos de CSS sin volverte loco.

Esto se logra creando por separado un documento con extensión de archivo CSS (.css), es un código escrito en lenguaje CSS que luego se vincula con el código HTML. Puedes nombrar este archivo como quieras, pero muchos desarrolladores suelen llamarlo “style.css”.

La forma de vincularlo con el archivo HTML es haciendo uso de la etiqueta <link> y un par de atributos importantes que lucen así:

```
<link rel="stylesheet" href="nombre-de-tu-archivo.css">
```

Para un correcto enlace del documento CSS, es necesario hacer uso de los atributos rel=”stylesheet” que se encarga de decirle a HTML el tipo de documento que se está vinculando, el href=”” que hace referencia al documento enlazado.

¿Notaste que la etiqueta <link> no tiene una etiqueta de cierre? Bueno, esto es porque no todas las etiquetas de HTML se deben de cerrar, simplemente porque es necesario que cuenten con contenido extra como texto entre ambas etiquetas como sucede con la de >button>Comprar curso</button>.

Esta etiqueta se debe colocar en el head de nuestro html.

### Tipos de selectores CSS

Revisaremos seis de los principales tipos de selectores CSS y algunos ejemplos de selectores para cada uno. Veremos:

1. Selector universal
2. Selector de tipo
3. Selector de clase
4. Selector de ID
5. Selector de atributo
6. Selector de pseudo-clase

Cada uno de ellos te ayudará a seleccionar diferentes grupos de elementos en un sitio web. Comenzaremos con uno de los tipos que te permitirán dirigirte a grupos grandes de elementos y, posteriormente, veremos algunos que te servirán para seleccionar elementos de manera más específica.

**1. Selector universal**

El asterisco (*) es el selector universal en CSS. De forma automática, el asterisco selecciona todos los elementos en un documento.

Sintaxis del selector universal

Un selector universal puede tener cualquiera de las siguientes sintaxis:

- * o *|* { propiedades de estilo } - selecciona todos los elementos
- ns|* { propiedades de estilo } - selecciona todos los elementos en el espacio de nombre (ns)
- |* { propiedades de estilo } - selecciona todos los elementos sin algún espacio de nombre definido

Ejemplo de selector universal

Supongamos que quieres hacer que todos los elementos de tu página sean de color naranja. En este caso lo mejor es utilizar un selector universal.

Así es como luce un documento en HTML:

```
   <h1>All elements on the page, from the heading 1</h1>
<h2 class="pastoral">to the heading 2 with class=pastoral</h1>
<p>to the paragraph will be orange.</p>
```

Este es el código en CSS con un selector universal para todos lo elementos:

```
    * {
color: orange;
}
```

Este es el resultado que obtendremos al combinar los archivos:


**2. Selector de tipo**

Un selector de tipo permite seleccionar todos los elementos en HTML que tienen un nombre de nodo común.

Por ejemplo, al usar «a» el selector elegirá todos los elementos <a> y les aplicará el valor de la propiedad en CSS. «Input» seleccionará todos los elementos <input>, «span» todos los elementos  y así sucesivamente.

Además, puedes usar espacios de nombre definidos para restringir las selecciones de los selectores de tipo a elementos que están dentro de este espacio.

Sintaxis del selector de tipo

La sintaxis del selector de tipo es muy sencilla. Únicamente deberás ingresar en tu código:

- elemento { propiedades de estilo }

Ejemplo de selector de tipo

Supongamos que tu documento contiene párrafos (<p>) y líneas (). Sin embargo, únicamente quieres darle una tonalidad anaranjada a las líneas.

Así es como luciría este documento en HTML:

```
    <span>One span. </span>
<p>No span. </p>
<span>Two span.</span>
<p>No span. </p>
```

Este es el código en CSS con un selector que define todas las líneas:

```
    span {
background-color: orange;
}
```

Este es el resultado que obtendremos al combinar los archivos:

One span.

No span.

Two span.

No span.

**3. Selector de clase**

Los selectores de clase son herramientas que, como su nombre lo indica, permiten seleccionar todos los elementos que tienen un mismo nombre de clase.

Por ejemplo, .intro te permitirá elegir todos los elementos que pertenecen a la clase «intro», así como .index escogerá todo elemento que tenga una clase «index».

Sintaxis del selector de clase

La sintaxis del selector de clase es la siguiente:

- .nombre de clase { propiedades de estilo }

Ejemplo de selector de clase

Supongamos que en esta ocasión quieres cambiar todos los elementos de la clase "pastoral" a un color anaranjado.

Así es como luciría este documento en HTML:

```
    <h1>Not orange</h1>

<h1 class="pastoral">Very orange</h1>
```

Este es el código en CSS con un selector que define todos los elementos pertenecientes a la clase "pastoral":

```
.pastoral {
color: orange
}
```

De acuerdo con estas reglas, la línea en h1 no debe cambiar de color, pero la segunda sí. Este es el resultado:


**4. Selector de ID**

Un selector de ID está diseñado para seleccionar elementos con base en su atributo de ID. Por ejemplo, #toc seleccionará todos los elementos que tengas como ID «toc». Ten en cuenta que este selector solo funcionará cuando el valor dado en el selector es idéntico al del ID del atributo que quieres referir.

Sintaxis del selector de ID

La sintaxis del selector de ID funciona del siguiente modo:

- #nombre de ID { propiedades de estilo }

Ejemplo de selector de ID

Si en este ejemplo queremos cambiar el color y la alineación de un elemento nombrado con el ID "hubspot" tendremos que seguir las siguientes instrucciones.

Así es como luciría el código en HTML:

```
<h1 id = "hubspot"> #id selector</h1>
```

Este es el código en CSS con un selector de ID que hace referencia a todos los elementos con el nombre "hubspot":

```
#hubspot {
color:orange;
text-align:right;
}
```

Este es el resultado:


**5. Selector de atributo**

Los selectores de atributo están hechos para seleccionar todos los elementos que correspondan a un atributo específico o a un valor de atributo definido.

Por ejemplo, a[href] elegirá todos los enlaces, mientras que a[href*="hubspot"] únicamente escogerá todas las URL que contengan la palabra «hubspot».

Puedes utilizar este tipo de selector para aplicar reglas de CSS a elementos que tienen un valor de atributo, por lo que si quieres modificar el estilo de todos los elementos que contengan «hubspot» en su URL, deberás utilizar a[href*="hubspot"].

Por otro lado, también puedes utilizar un espacio de nombre en combinación con este selector para restringir la búsqueda a elementos dentro de ese espacio.

Sintaxis del selector de atributo

Algunas de las sintaxis disponibles para el selector de atributo son las siguientes:

- [attr] { propiedades de estilo }
- [attr=value] { propiedades de estilo }
- [attr~=value] { propiedades de estilo }
- [attr|=value] { propiedades de estilo }
- [attr^=value] { propiedades de estilo }
- [attr$=value] { propiedades de estilo }
- [attr*=value] { propiedades de estilo }

La elección de una de estas sintaxis dependerá de si los elementos que quieres seleccionar tienen atributos que han sido ajustados a un valor específico.

Ejemplo de selector de atributo

Supongamos que quieres hacer que todos los links que contienen «hubspot» cambien el color de la URL a anaranjado. En este caso puedes utilizar a[href*="hubspot"].

Así es como luciría el código en HTML:

```
    <ul>
<li><a href="http://blog.com">blog.com</a></li>
<li><a href="http://hubspot.com">hubspot.com</a></li>
<li><a href="http://google.com">google.com</a></li>
<li><a href="http://blog.hubspot.com">blog.hubspot.com</a></li>
</ul>
```

Este es el código en CSS con un selector de atributo que modificará todos los elementos que contienen «hubspot» en su URL:

```
    a[href*="hubspot"] {
color:orange;
}
```

Este es el resultado:

- blog.com
- hubspot.com
- google.com
- blog.hubspot.com

Ten en cuenta que si el ID del atributo del elemento está escrito en minúsculas el selector deberá incluir el nombre en minúsculas. De lo contrario, los elementos con una diferente escritura no serán seleccionados.

**6. Selector de pseudo-clase**

Un selector de pseudo-clase permite aplicar CSS a una selección de elementos o a elementos que se encuentran en un estado específico.

Por ejemplo, :hover hará que únicamente se modifique el estilo de un elemento cuando el usuario se desplace sobre él.

Otros ejemplos comunes son :active, :visited o :invalid.

Sintaxis del selector de pseudo-clase

La sintaxis del selector de pseudo clase es:

- selector:puedo-clase { propiedades de estilo }

Ejemplo de selector de pseudo-clase

En este ejemplo queremos cambiar el color de los enlaces a verde cuando el usuario haya visitado los sitios haciendo clic sobre el hipervínculo. Los enlaces con los que no haya interactuado el usuario deberán permanecer en su color original, azul. Además, queremos que los enlaces cambien a un color rosado cuando el usuario se desplace sobre ellos.

Así es como luciría el código en HTML:

*So you've already visited blog.hubspot.com. Why not check out our home site at hubspot.com?*

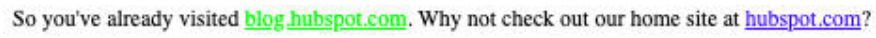
Este es el código en CSS con tres diferentes pseudo-clases para los enlaces que no han sido visitados, para aquellos a los que se ha accedido y para aquellos sobre los que se está deslizando el usuario:

```
    a:link {
color: #0000FF;
}

a:visited {
color: #00FF00;
}

a:hover {
color: #FF00FF;
}
```

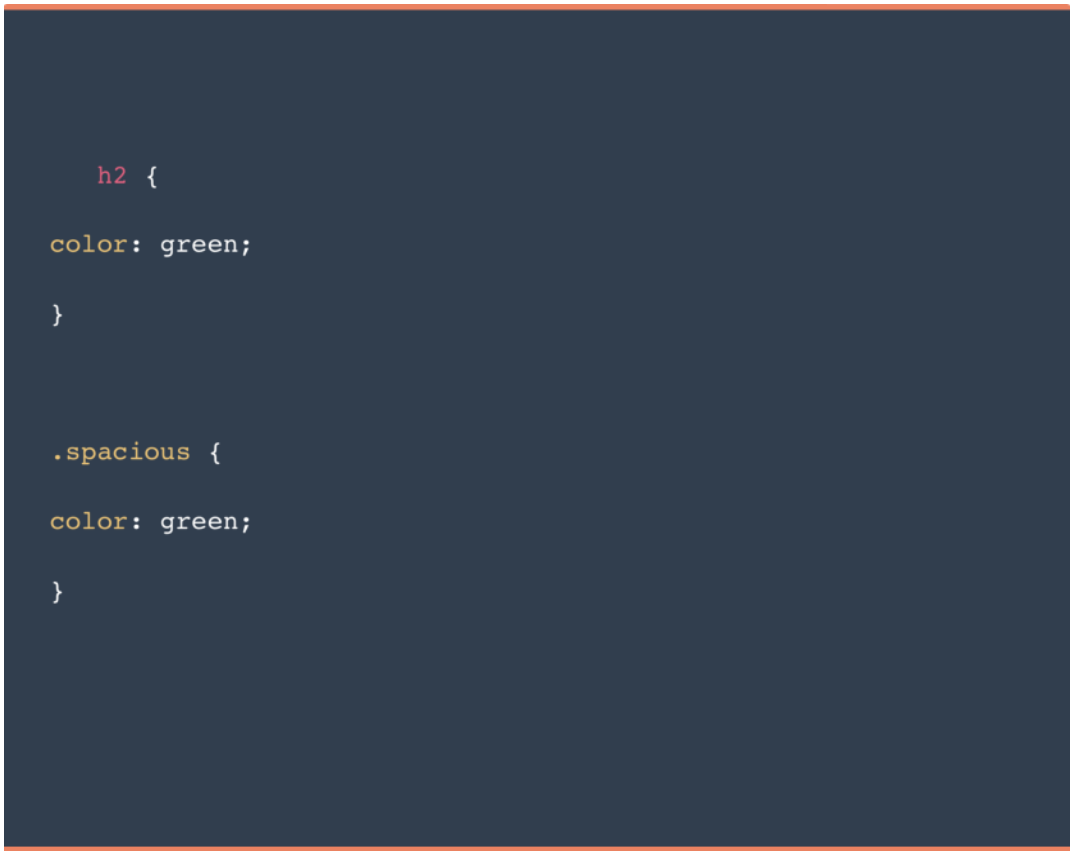
Este es el resultado:




Ahora que te has familiarizado con los principales tipos de selectores CSS hablemos de cómo puedes combinarlos en tu sitio web.

**Cómo agrupar múltiples selectores en CSS**

Supongamos que tienes una gran diversidad de elementos a los que quieres aplicar el mismo estilo, como un h2 y una clase .spacious, y quieres hacer verdes ambos elementos. El código podría ser escrito en dos líneas diferentes, como puedes ver aquí:




Otra alternativa es combinar los selectores en una lista de selección. Para crearla únicamente deberás listar todos los selectores que quieres emplear y separarlos con comas antes del corchete que contiene la propiedad de estilo. Ya que el espacio en blanco es válido antes y después de una coma, puedes agregar un espacio después de cada coma para hacer que el código sea más fácil de leer.

De acuerdo con lo anterior la sintaxis sería: elemento, elemento, elemento { propiedad de estilo }. Así luciría nuestro ejemplo

```
h2, .spacious {
color: green;
}
```

También puedes ubicar los selectores en su propia línea, si eso te ayuda a hacer más legible tu código. En este caso la sintaxis se verá de este modo:

```
h2,
.spacious {
color: green;
}
```

Combinar selectores CSS es una estrategia que te ayudará a reducir el tamaño de tus documentos y hará que tu página web cargue mucho más rápido.

Los selectores CSS te permiten mantener un buen control respecto al modo en que diseñas y codificas al momento de construir un sitio web desde cero. A pesar de que dominar estas herramientas puede llevarte algo de tiempo, es una inversión que rendirá sus frutos. Aprende a usarlos y prueba con los diferentes tipos que te hemos compartido para que puedas darle el estilo indicado a tu marca con un código elegante, simple y de carga rápida.

## Herencia:

Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos descendientes heredan de forma automática el valor de esa propiedad.

```
body { color: blue; }
```

Todos los elementos de la página tendrían color de letra azul.

## Cascada:

El orden de las reglas de CSS importa: cuando dos reglas tienen la misma especificidad, se aplica la que aparece en último lugar en el CSS.

```
h1 {
   color: red;
}
h1 {
   color: blue;
}
```

El h1 va a ser de color azul, ya que es la última regla que se aplicó.

## Especificidad:

La especificidad consiste en un cálculo que hace el navegador para establecer su nivel de importancia.

Lo que hace es otorgar un valor de puntos a los diferentes tipos de selectores y la suma de estos establece la importancia de ese selector en particular.

Por ejemplo:

- Un selector de clase es más específico que un selector de etiqueta.
- Un selector de ID es más específico que un selector de clase.

## Modelos de Cajas

En CSS todo elemento está enmarcado en una caja.

Hay dos tipos de cajas:

- **cajas en bloque**
- **cajas en línea.**

Estas características se refieren al modo como se comporta la caja con otras.

Las cajas en **bloque** siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea, aunque sus contenidos no lleguen hasta el final de la línea.

Las cajas en **línea** sólo ocupan el espacio necesario para mostrar su contenido.

Los párrafos (p) por ejemplo son elementos de bloque y lo links (a) son de línea.

Este comportamiento se puede modificar con la propiedad display de css.

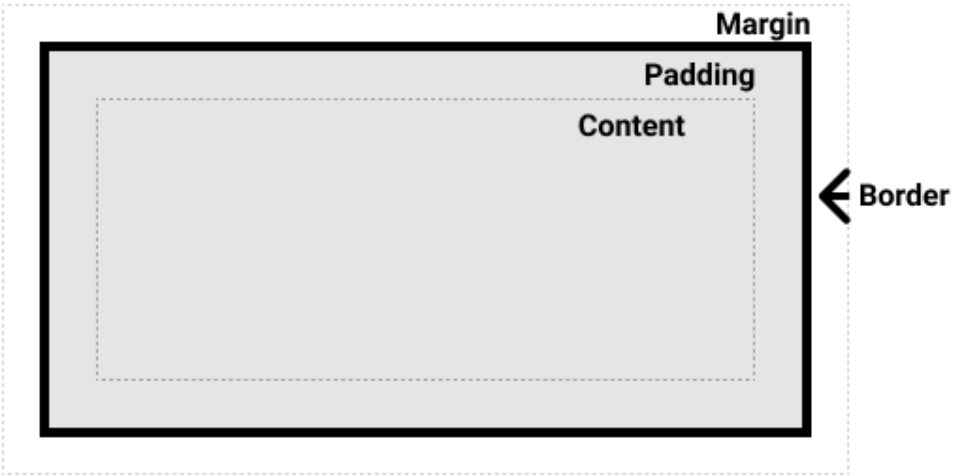
**Valores de la propiedad Display:**

Los valores más comunes que puede tomar la propiedad display son:

- **block:** hace que el comportamiento del elemento sea como un bloque.
- **inline:** hace que los elementos se muestren en la misma línea, ajustándose al contenido y sin tener en cuenta el ancho, alto o márgenes de la caja.
- **inline-block**: hace que los elementos se muestren en la misma línea respetando el ancho, el alto y los márgenes.
- **flex:** hace que los elementos hijos se comporten de manera flexible.
- **none:** oculta un elemento.

**Como está conformada una caja:**

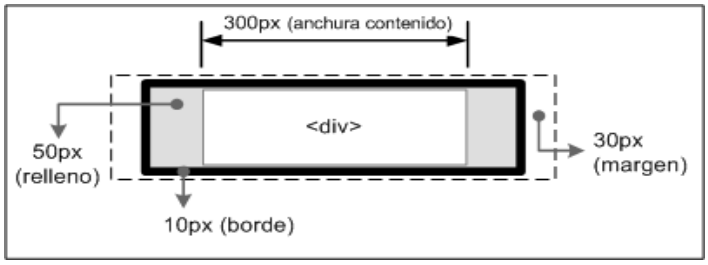
Una caja esta conformada de la siguiente manera:




**Cálculo del tamaño de la caja:**

El ancho y el alto de un elemento no solamente se calculan teniendo en cuenta sus propiedades width y height sino que el margin, el padding y border también suman al cálculo final.

```
div {
  width: 300px;
  padding-left:  50px;
  padding-right: 50px;
  margin-left:   30px;
  margin-right:  30px;
  border: 10px solid black;
}
```




**Ancho total de la caja: 30px + 10px + 50px + 300px + 50px + 10px + 30px = 480 píxel**

**box-sizing**

Esto se puede modificar a través de la propiedad **box-sizing** que acepta los siguientes 2 valores:

**content-box**: comportamiento por defecto para el tamaño de la caja.

**border-box**: el padding y border de ese elemento no incrementan su ancho. Si se define un elemento con un ancho de 100 pixeles, estos incluirán cualquier border o padding que se añadan, y la caja de contenido se encogerá para absorber ese ancho extra.

## Propiedades de Textos

## Propiedades de color y background

## Propiedades Position

Existe una propiedad llamada position que sirve para posicionar un elemento dentro de la página.

Dependiendo de cual sea la propiedad que usemos, el elemento tomará una referencia u otra para posicionarse respecto a ella.

Los posibles valores que puede adoptar la propiedad position son:

- Static
- Relative
- Absolute
- fixed

**position: static** Es el valor que toma un elemento por defecto para posicionarse. No tendrá en cuenta los valores para las propiedades top, left, right y bottom.

**position: relative** Se comporta de la misma manera que static a menos que le agreguemos las propiedades top, left, right y bottom. Esto causará que su posición normal se reajuste, pudiendo causar solapamiento entre los distintos elementos.

**position: absolute** Cuando se posiciona un elemento de manera absoluta, se hace en base a su elemento contenedor (por lo general el cuerpo del documento, o un elemento posicionado relativamente que lo contenga).

**position: fixed** Este atributo sirve para posicionar un elemento como si tuviera posicionamiento absoluto, pero su posición final será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición.

**position: sticky** Es una posición nueva que todavía no está soportada por todos los navegadores. Su comportamiento va a ser relativo hasta que al hacer scroll llegamos al elemento con position sticky y en ese momento pasa a tener un comportamiento fijo.

## Propiedades z-index

Cuando varios elementos se superponen, **z-index** determina cual está por encima del otro.

Un elemento con mayor z-index se ubica por encima de uno con z-index menor.

Esta propiedad se puede aplicar solamente en los elementos que poseen algún tipo de posición que no sea static.

## Flexbox

Es un sistema de elementos flexibles que se caracterizan por su habilidad de alterar su **ancho y alto** para ajustarse lo mejor posible al espacio disponible en cualquier dispositivo.

Este tipo de modelo flexbox, nos permite controlar parámetros tales como **la alineación**, **dirección de los elementos** (horizontal/vertical), entre otros.

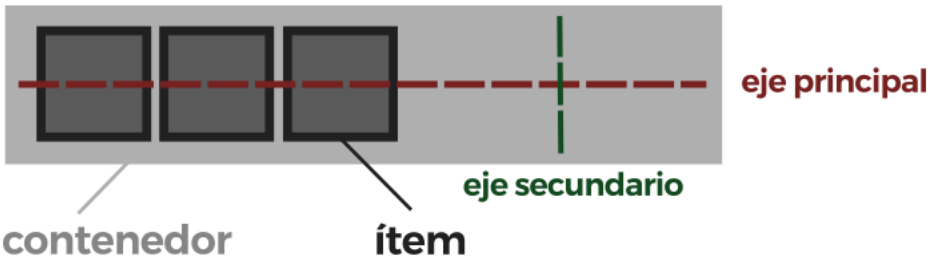
**Conceptos básicos:**

Para empezar a utilizar flexbox lo primero que debemos hacer es conocer algunos de los elementos básicos:

**Contenedor flexible (Flex container)**

El elemento "padre" que contiene cada uno de los ítems flexibles. Un contenedor flexible se define usando los **valores flex o inline-flex en la propiedad display.**

- **Eje principal:** Los contenedores flexibles tendrán una orientación principal específica. Por defecto, es en horizontal (en fila).
- **Eje secundario:** De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical, y viceversa.



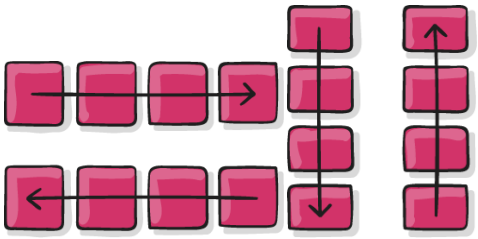

**Dirección de los ejes**

**Propiedad flex-direction**

Permite modificar la dirección del eje principal del contenedor para que se oriente en horizontal (por defecto) o en vertical.

Además, también podemos incluir el sufijo **-reverse** para indicar que coloque los ítems en orden inverso.

- **flex-direction: row;** Los elementos se visualizan de izquierda a derecha (valor por defecto)
- **flex-direction: row-reverse;** Los elementos se visualizan de derecha a izquierda.
- **flex-direction: column;** Los elementos se visualizan de arriba hacia abajo.
- **flex-direction: column-reverse;** Los elementos se visualizan de abajo hacia arriba.

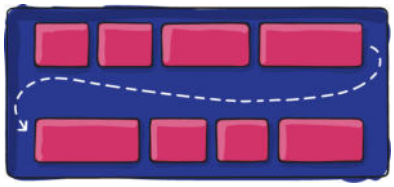


**Propiedad flex-wrap**

Permite especificar el comportamiento del contenedor respecto a evitar que se desborde (nowrap, valor por defecto) o permitir que lo haga, en cuyo caso, estaríamos hablando de un contenedor flexbox multilínea, es decir los elementos se distribuyen en varias filas.

- **flex-wrap: nowrap;** Establece los elementos en una sola línea (no permite que se desborde el contenedor).
- **flex-wrap: wrap;** Los elementos se muestran en una sola línea, pero si su ancho supera la del contenedor, se distribuyen en varias filas.

- **flex-wrap: wrap-reverse;** Su comportamiento es igual al flex-wrap: wrap pero en dirección inversa.



**Dirección de los ejes**

**Atajo / short hand**

Existe una propiedad llamada **flex-flow** que permite resumir los valores de las propiedades flex-direction y flex-wrap en una sola propiedad:

```
.container {
  /* flex-flow: <flex-direction> <flex-wrap>; */
  flex-flow: row wrap;
}
```

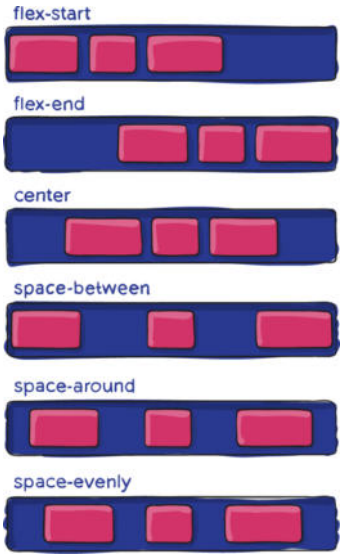
**Propiedades de alineación**

**Sobre el eje principal**

Existe distintas propiedades dentro de flexbox para disponer los ítems dependiendo de nuestro objetivo.

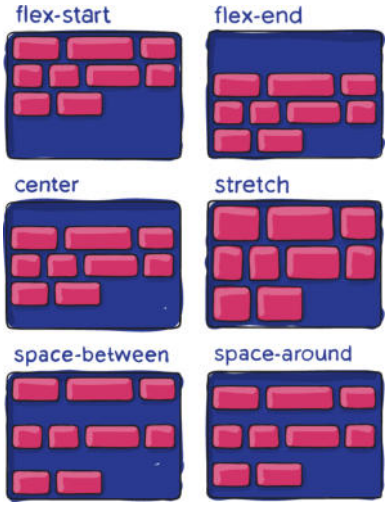
**justify-content:** Se utiliza para alinear los ítems del eje principal (por defecto, el horizontal). Y puede tomar los siguiente valores:

- **flex-start:** Agrupa los ítems al principio del eje principal.
- **flex-end:** Agrupa los ítems al final del eje principal.
- **center:** Agrupa los ítems al centro del eje principal.
- **space-between:** Distribuye los ítems dejando el máximo espacio para separarlos.
- **space-around:** Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha).
- **space-evenly:** Distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha.




La propiedad **align-content** actúa sobre cada una de las líneas de un contenedor multilínea (no tiene efecto sobre contenedores de una sola línea). Los valores que puede tomar son los siguientes:

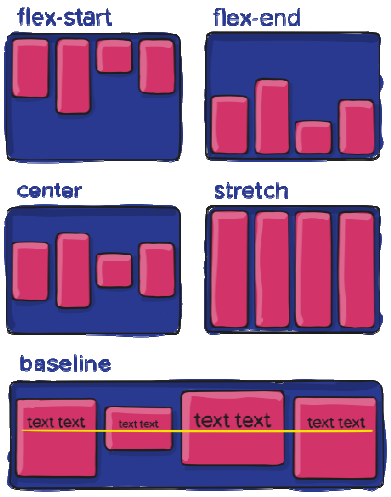
- **flex-start:** Agrupa los ítems al principio del eje principal.
- **flex-end:** Agrupa los ítems al final del eje principal.
- **center:** Agrupa los ítems al centro del eje principal.
- **stretch:** Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
- **space-between:** Distribuye los ítems desde el inicio hasta el final.
- **space-around:** Distribuye los ítems dejando el mismo espacio a los lados de cada uno.




**Sobre el eje secundario**

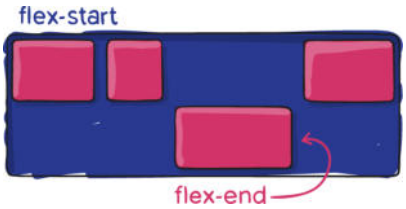
La propiedad **align-items**, se encarga de alinear los ítems en el eje secundario del contenedor. Los valores que puede tomar son los siguientes:

- **flex-start:** Alinea los ítems al principio del eje secundario.
- **flex-end:** Alinea los ítems al final del eje secundario.
- **center:** Alinea los ítems al centro del eje secundario.
- **stretch:** Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
- **baseline:** Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.




La propiedad **align-self** actúa exactamente igual que align-items, sin embargo se utiliza sobre un ítem hijo específico y no sobre el elemento contenedor. Los valores que puede tomar son los mismos que align-items:

- **flex-start:** Alinea los ítems al principio del eje secundario.
- **flex-end:** Alinea los ítems al final del eje secundario.
- **center:** Alinea los ítems al centro del eje secundario.
- **stretch:** Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
- **baseline:** Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor.
- **auto:** Hereda el valor de align-items del padre (si no se ha definido, es stretch).




**Propiedades de hijos**

Todas las propiedades vistas se aplican sobre el elemento contenedor. Las siguientes propiedades, se aplican sobre los ítems hijos.

- **flex-grow:** 0 | Número que indica el factor de crecimiento del ítem respecto al resto.
- **flex-shrink:** 1 | Número que indica el factor de decrecimiento del ítem respecto al resto.
- **flex-basis:** size - content | tamaño por defecto que tendrán los ítems antes de aplicarle la distribución de espacio. Generalmente, se aplica un tamaño (unidades, porcentajes, etc...), pero también se puede aplicar la palabra clave content que ajusta automáticamente el tamaño al contenido del ítem, que es su valor por defecto.
- **order:** 0 | Número que indica el orden de aparición de los ítems.

  **Flex-grow**