**CSS**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente** **Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

**Fuente**

* font-family
  + font-family: Arial, sans-serif
    - En caso de que la primera no se pueda aplicar, se aplica la segunda, tercera, etc
* Font-size
  + Font-size: 23px
    - Las unidades suelen ser px, em y rem
* Font-style
  + Font-style: italic, normal u oblique
  + Para algunos elementos como los <em> el valor por defecto será italic
* Font-weight
  + Font-weight: 500
  + Font-weight: bold
  + Define el peso de la tipografía. Recibe los valores bold, lighter, normal, entre otros. También puede recibir un valor numérico que se irá incrementando de 100 en 100. Para algunos tags el valor por defecto será bold.
* Text-align
  + Text-align: justify
    - Permite definir la alineación del texto. Los valores que recibe son center, left,
    - right, inherit y justify. El valor por defecto para todos los elementos es left.
* Text-decoration
  + Text-decoration: underline;
  + Permite elegir un tipo de decoración para el texto. Recibe los valores line-through, underline, overline y none. Para algunos elementos, como los enlaces, el valor por defecto será underline
* Line-height
  + Line-height: 20px
  + Permite definir el alto de cada línea de textos. Esto también suele llamarse interlineado.
  + Recibe un valor numérico acompañado de la unidad de medida y, por lo general, está relacionado con el tamaño de la letra o font-size.
  + Sirve para indicar cuánto espacio tiene que haber entre renglones
* Text-transform
  + Capitalize
    - Primera letra mayúscula
  + Uppercase
    - Todas las letras mayúsculas
  + Lowercase
    - Todas las letras minúsculas

Fondos

* Background-image
  + Backgroun-image url
  + 
* Background repeat
  + Nos permite controlar si se va a repetir, y de qué manera, la imagen dispuesta. Recibe los valores: repeat, no repeat, repeat-x, repeat-y, round y space
* Background-position
  + Nos permite mover la imagen dentro del elemento y decidir dónde colocarla. Recibe como valores tamaños en pixeles y porcentajes, así como también right, bottom, left, etcétera. Podemos asignarle uno o dos valores. El primero para especificar la posición en el eje x y el segundo, la posición en el eje y.
* Background-attachment
  + Nos permite establecer si la imagen de fondo se va a mover junto con la página al hacer scroll o si se va a quedar fija. Recibe como valor: fixed, scroll, inherit e initial.
* Background-size
  + Nos permite establecer el tamaño de la imagen de fondo. Recibe como valor: contain, cover, inherit; así como también tamaños en pixeles y porcentajes, indicando con el primer valor el ancho y con el segundo, el alto.
* Display
  + Sirve para modificar còmo se comportan los elementos. H2 es por defecto un elemento en bloque. Si queremos que se comporte como elemento en línea, en el css utilizamos: display: inline
    - Display: inline
    - Display: inblock

**Elementos en línea y elementos en bloque**

Los elementos en línea y lo elementos en bloque tienen propiedades diferentes.

* Elementos en línea
  + No pueden tener height. width

**Colores**

Formatos:

1. Nombre de color
   1. Font-color: green
2. Hexadecimal
   1. # f05331
3. RGB (255,100,42)
4. RGBA (244,9,33, 0.5)
   1. El último campo sirve para indicar la opacidad (intensidad) del color. Debe ser un valor entre 0 y 1

Color de texto:

El atributo color nos permite asignarle un color al texto de un elemento.

Recibe como valor cualquiera de los formatos de color permitidos

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Color de fondo:

El atributo *background-color* nos permite asignarle un color de fondo a un

elemento.

Recibe como valor cualquiera de los formatos de color permitidos.

Interfaz de usuario gráfica, Texto, Aplicación, Sitio web

Descripción generada automáticamente

Opacidad

Para darle mayor o menos transparencia a un elemento.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

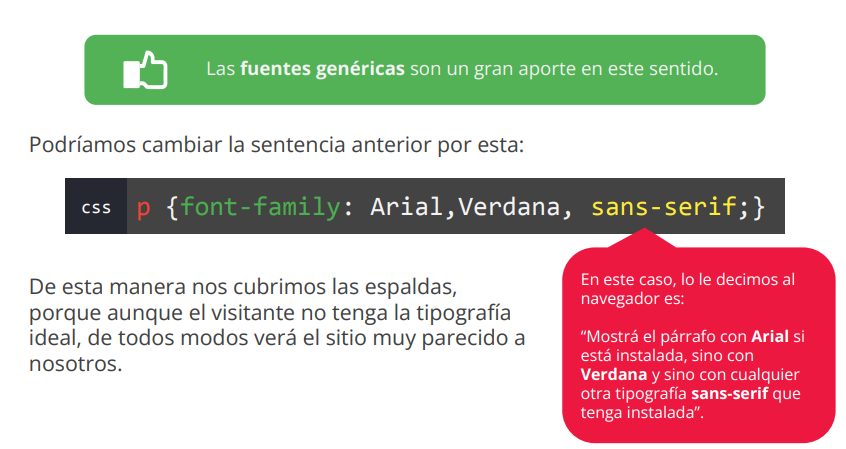
**Buenas prácticas:**

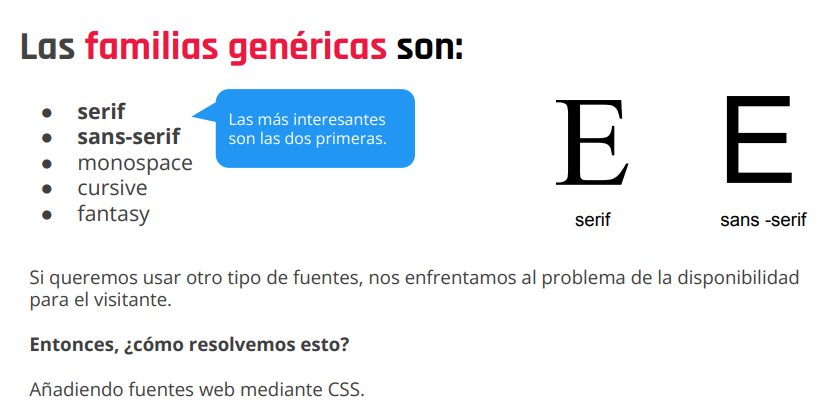
Orden de decoración:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Fuentes:**





Fuentes de terceros

Mientras que si se utilizan fuentes de terceros, como Google o Adobe Fonts, solo estamos incluyendo las fuentes mediante la vinculación a un activo externo.

Alojar fuentes localmente significa que tenemos los archivos de fuentes en nuestro propio servidor. Para ello vamos a hacer uso de la directiva css @font-face.

La regla @font-face permite cargar fuentes personalizadas en una página web. Una vez que se agrega a una hoja de estilo, la regla indica al navegador que descargue la fuente desde donde está alojada y luego la muestre como se especifica en el CSS.

**Iconos**

Hay muchas librerìas gratuitas y cada una de ellas tendrá sus propias instrucciones para ser instaladas

La forma más simple sería incluir la propia librería de ellos. Los pasos para hacerlo son:

Incluimos el enlace a nuestros iconos dentro del <head> del HTML:

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.15.2/css/all.css">

Insertar el icono seleccionado en el lugar del <body> que más nos guste:

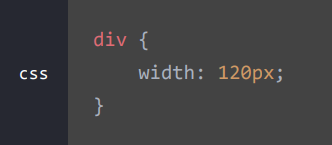
<i class="fas fa-exclamation-triangle"></i>

**Modelo de cajas**

****

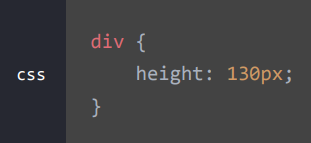
Propiedad width:

Si un elemento de bloque no tiene declarada la propiedad width, el ancho será igual al 100% de su padre contenedor. Podemos asignarle un valor a esta propiedad usando cualquier unidad de medida, como porcentajes (%) o píxeles (px)



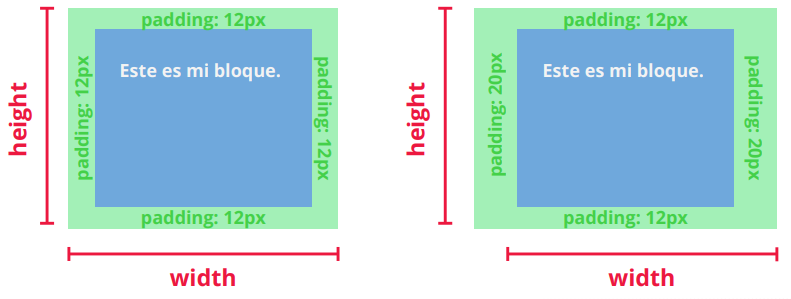
Propiedad height:

Si un elemento no tiene declarado la propiedad height, el alto será igual a la altura que le proporcione su contenido. Sea un elemento de bloque o de línea. Podemos asignarle un valor a esta propiedad usando cualquier unidad de medida, como pixeles (px). Para la altura de los elementos no se recomienda usar porcentaje



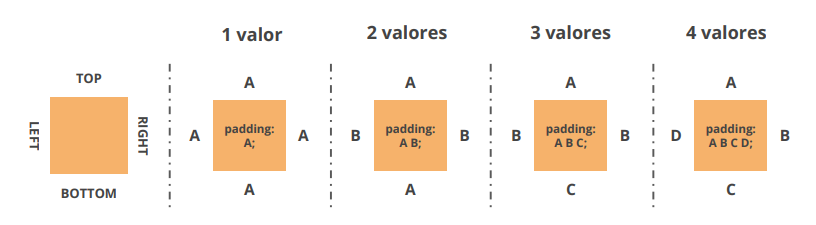
Propiead Padding:

Es el espacio de relleno que podemos agregar entre el contenido del elemento y su borde.



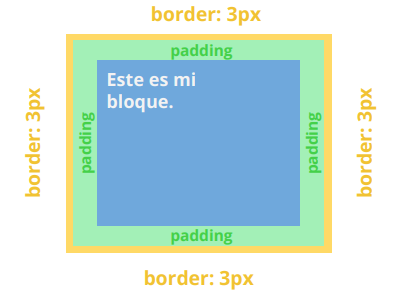
La propiedad padding puede tener 1, 2, 3 o 4 valores:

* Si tiene un valor: Es la misma cantidad de pixeles para todos los lados de la caja
* Si tiene dos valores: El primer valor es para los bordes de arriba y abajo y el segundo valor es para los bordes de izquierda y derecha
* Si tiene tres valores: El primer valor es para el borde de arriba, el segundo valor es para los bordes de izquierda y derecha y el tercer valor es para el borde de abajo
* Si tiene cuatro valores: El primer valor es para el borde de arriba, el segundo valor es para el borde de la derecha, el tercer valor para el borde de abajo y el cuarto valor para el borde de la izquierda



**Border**

Hace referencia al borde del elemento. Se ubica entre el contenido y el margen.

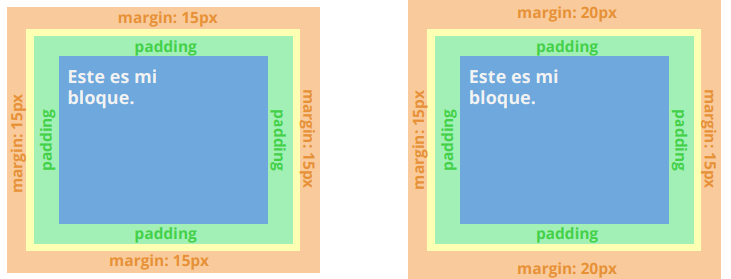
****

Podemos asignarle un valor a esta propiedad definiendo el estilo de línea, su espesor y su color. El estilo de línea puede ser **solid, dotted, dashed o double**. El espesor de línea puede ser cualquier unidad de medida de CSS. El color puede ser cualquier color válido de CSS.

****

**Margin:**

Hace referencia al margen exterior del elemento. Sirve para separar una caja de la otra.

****

**Box-sizing**

Box-sizing: content-box

Por defecto su valor será content-box, y el comportamiento será el de aplicarle el ancho y alto que definamos al contenido del elemento.

box-sizing: border-box

Si le asignamos el valor border-box, el ancho y alto que indiquemos tomará en cuenta no solo el contenido del elemento, sino también el padding y el borde, dejando solo el margen por fuera.

**Posicionamiento sticky:**

El posicionamiento sticky puede considerarse un híbrido de los posicionamientos relativo y fijo. Un elemento con posicionamiento sticky es tratado como un elemento posicionado relativamente hasta que cruza un umbral especificado, en cuyo punto se trata como fijo hasta que alcanza el límite de su padre.

Sticky es para que algo quede fijo a pesar de que sea haga scroll.

****

**Position:**

Cuando desplazamos un elemento tomando un costado como referencia, empujaremos el elemento si el número es positivo o tiraremos de él si el número es negativo.

El posicionamiento relativo nos permite trasladar un elemento desde su posición original a una nueva posición.

Cuando movemos una caja, el punto de referencia serán sus propios costados. Al posicionar la caja 1 de manera relativa, el espacio que ocupaba originalmente seguirá ocupado. Eso quiere decir que los elementos que estén a su lado (caja 2) no van a ser afectados por esta modificación. Solemos utilizar posicionamiento relativo cuando queremos desplazar un elemento sin modificar el flujo original de los demás que están a su lado.

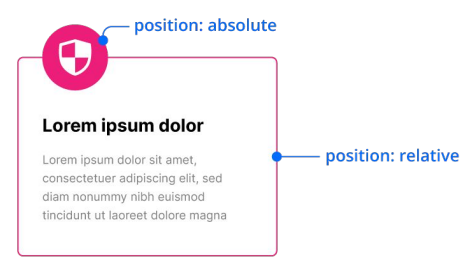
Posicionamiento absoluto

El posicionamiento absoluto nos permite trasladar un elemento tomando como referencia los costados del body.

Con el posicionamiento absoluto, los puntos de referencia serán los costados el body. Cuando movemos una caja de manera absoluta, **el espacio que ocupaba quedará vacío y otros elementos podrán ocuparlo.**

Si nuestras cajas (hijas) están dentro de otra caja (padre), el punto de referencia seguirá siendo el body a menos que hagamos relativa la posición de su padre.

Se suele utilizar cuando queremos sacar un elemento del flujo normal y posicionarlo en un punto fijo con respecto a su contenedor o el body.

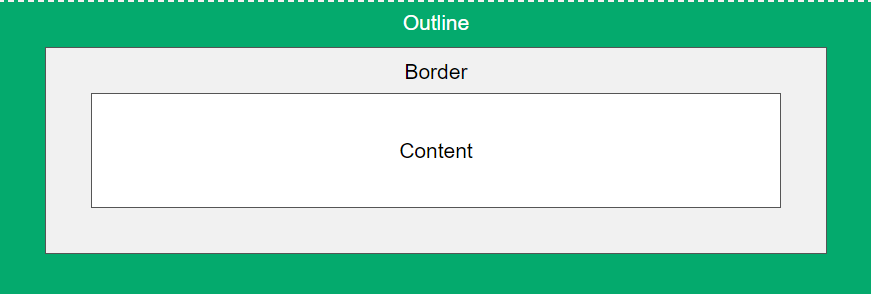
****

Posicionamiento fijo

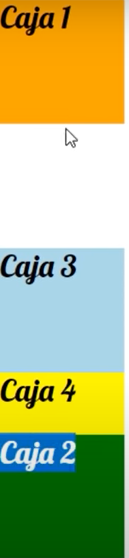
Con el posicionamiento fijo, los puntos de referencia serán los costados la ventana del navegador. Sin importar que hagamos scroll en la página, el elemento siempre se mantendrá fijo con respecto a la ventana del navegador.

Cuando queremos que un elemento siga al usuario a medida que navega nuestro sitio. Por ejemplo, un botón fijo para siempre tener disponible la opción de crear un nuevo producto.

**Propiedades de caja**

* Box-shadow:
  + Sirve para darle efecto de sombra a las cajas
  + A box shadow is described by X and Y offsets relative to the element, blur and spread radius, and color. Es decir, tiene 4 componentes.
* Text-shadow
  + Sirve para darle efecto de sombra al texto
  + Each shadow is described by some combination of X and Y offsets from the element, blur radius, and color. Es decir, tiene 3 componentes
* Outline
  + An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".
  + Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

**Position**

* Fixed
  + Hace que el element se encuentre siempre anlado a pesar de los scrolls. SI se scrollea para abajo, el element siempre sigue arriba de todo
  + Es importante ponerle el top:0 para que quede anclado arriba
* Sticky
  + El elemento inicialmente està por debajo y scrollea normalmente pero, al llegar al top, se queda anclado arriba de todo y ya no se mueve de esa posición.
  + Al sticky es importante ponerle top:0 para que quede anclado arriba
* Relative
  + An element with position: relative; is positioned relative to its normal position.
  + Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
  + Supongamos que se tienen 4 cajas, una encima de la otra. Si a la segunda caja, le aplicamos position relative y botom, esta caja va a moverse esa determinada cantidad de pixeles hacia abajo. Lo que es importante, es que el espacio que ocupaba previamente lo sigue “teniendo reservado”, es decir queda el hueco que no es ocupado por otra caja. Los espacios no se cuentan desde arriba de todo en el navegador sino que desde el espacio que ocuparìa la caja en su posición normal.
    - 
* Z index
  + Supongamos que se tienen dos cajas superpuestas. Para determinar, què caja va a estar en el frente, y cuàl va a estar en el fronto se utiliza el z index. Aquella caja con un valor de z index mayor será la que se encontrará en el frente.
  + La propiedad Z index se aplica únicamente cuando la caja está posicionada. Esto quiere decir que en las propiedades del CSS, tiene que tener el campo position: atributo. Si se tienen dos cajas y una de las dos tienen position relative y la otra no tiene, la que tiene position relative se pondrá por encima de la otra. Si ninguna tiene o si ambas tienen, “la que gana” será la que se encuentre por debajo en el HTML ya que este será el orden que se respetará. Supongamos ahora que ambas cajas tienen position relative. Para solucionar este tipo de escenarios es que se utiliza el z index. Aquella caja con un valor de Z index de mayor valor es quién se pondrá por delante
  + Conflicto de padres e hijos
    - Supongamos que tenemos un contendor padre y dentro una caja (hijo). El CSS siempre respeta el orden en el que todo fue dispuesto en el HTML para poder priorizar en caso de no tener position relative. Si al hijo se le pone position relative, el hijo quedará por encima del padre.
    - El conflicto se basa en que si ambos tienen position relative y se les quiere aplicar z index, se observará que no importa el valor que se le adopte al hijo y al padre, el padre NUNCA estará por encima del hijo.
    - Para resolver esto, hay que cambiar la lógica con la que se lo piensa. Esto quiere decir que lo que se debe hacer es poner al hijo POR DETRÁS del padre. Para hacer esto, al hijo se le pondrá z index con valor negativo y al padre no se le pondrá ningún tipo de Z index. Es MUY importante que no se le ponga ningún tipo de Z index al padre ya que, con el simple hecho de poner un valor (incluso 0) este conflicto no se resolverá y se seguirá viendo la caja del hijo por encima de la del padre.
* Absolute
  + El espacio ya no es más reservado como en position relative. Es decir que si se tienen 3 cajas, una debajo de la otra y la caja 2 tiene position absolute, la caja 3 ocupará el mismo espacio que la caja 2 ya que verá que ese espacio no tiene reserva y por lo tanto “está libre”
  + Otro punto importante a destacar es que la caja toma como nuevo punto de referencia (para los top, bottom, etc) el punto donde quedó o la caja contenedora.

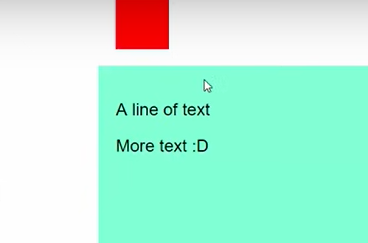


**Relacion entre absolute y relative**

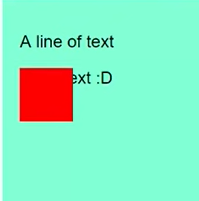
El absolute ignora todo el orden establecido en el html. No importa si en el html se encuentra dentro de un contendor o no. En el caso que lo que se quiera es que la caja con absolute no se vaya de los límites del contenedor y que el 0 0 para la caja sea el contenedor, el contenedor deberá contar con position: relative. El elemento con absolute buscará y respetará el elemento más cercano anterior con relative. En caso de que ningún elemento previo tenga position relative, tomará el index como elemento anterior.

Ejemplos:

La caja contenedora no tiene position relative por lo que la caja roja toma el index como relative y se va por fuera de los límites:



La caja conteneora **sí** tiene relative por lo que ahora todas las posiciones de la caja roja se encontrarán dentro del contenedor.



**Display**

El display define si un elemento va a ser tratado como **bloque** o como **línea.** Los objetos en lìnea (por ej h1, h2) tienen la característica de que no se les puede dar ni width ni height. Los elementos en bloque sì pueden hacerlo. Es por esto, que si quiseramos darle altura a un elemento en bloque, tendríamos que pedirle que sea tratado como en línea. Los comandos a utilizar serán

* Display: block
* Display: inline
* Display: inline-block
  + Se utiliza cuando se quiere que el elemento pueda ser un objeto en bloque (ocupar todo el espacio de un bloque) pero a la vez poder darle altura y ancho

The **display** [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) property sets whether an element is treated as a [block or inline element](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flow_Layout) and the layout used for its children, such as [flow layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flow_Layout), [grid](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout) or [flex](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout).

**Flex**

Propiedades:

**Flex-direction**

Se le debe indicar si lo queremos horizontal o vertical. Por defecto va a ser horizontal.

Column

Column-reverse

Row

Row-reverse

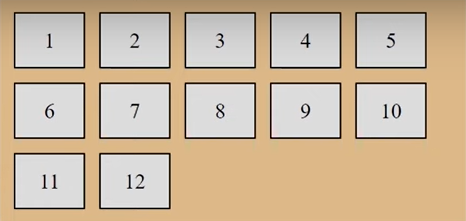
**Flex-wrap (o wrap reverse)**

Sirve para indicar si queremos que todas las cajas que se encuentran dentro del contenedor seencuentren sí o sí en una fila o, en caso de no entrar, pasen a una fila siguiente.

Sin flex wrap sería así:



Con flex wrap:



**Flex-flow:**

Es una union entre flex direction y flex wrap. Puede ser column-wrap, row-wrap

**Justifiy content:**

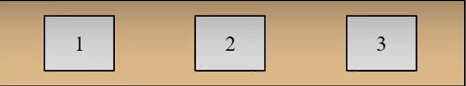
Sirve para poder posicionar los elementos de determinadas maneras. Por ejemplo justifiy-content: center me permitirá que todos los hijos se encuentren centrados.



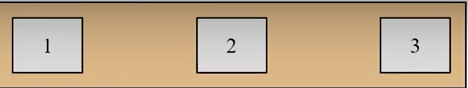
Si se quisiera alineados a la izquierda, el comando sería justify-content: flext start

Si se quisiera alinearlos a la derecha, el comando sería justify-content: flex end

Justify-content: space around:



Justify-content: space between:

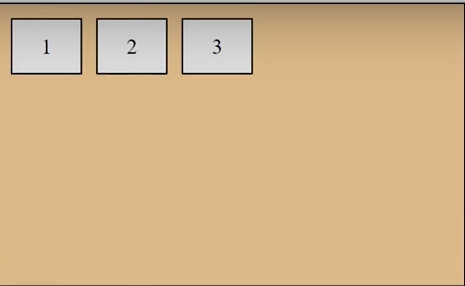


**Align-items**

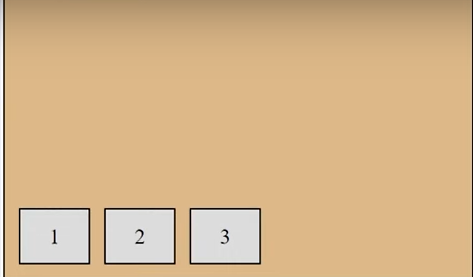
Sirve para alinear los hijos de los contenedores.

*Align-items: center*

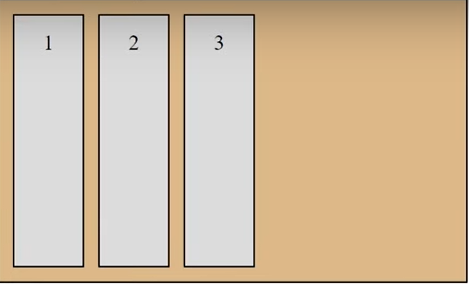
*Align-items: flex start*

**

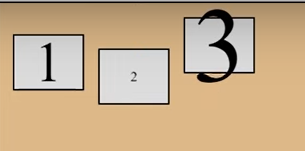
*Alignt-items: Flex end*

**

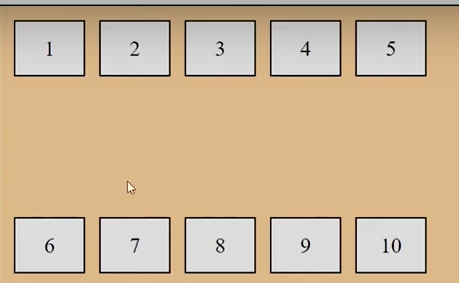
*Alignt-items: stretch*. Es el que viene por defecto, intentará ocupar toda el área:



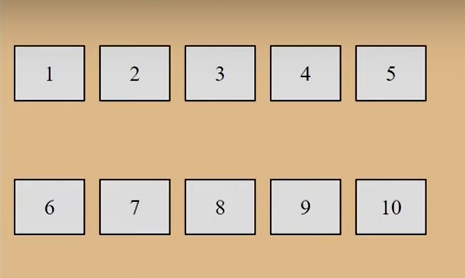
*Align-items: baseline* sirve para que todos lo elementos, sin importar el tamaño de la letra, se encuentren apoyados sobre una misma recta. Si se traza una recta debajo de 1, 2 y 3 se verá que es paralela a los bordes del contenedor.



*Align-content:* *space between*. Sirve si se quiere priorizar el espacio entre las cajas por sobre el espacio entre las cajas y los bordes:



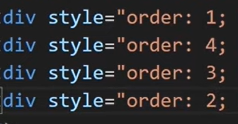
*Align-content: space around*. Sirve si se quiere priorzar el espacio entre las cajas y los bordes por sobre el espacio entre las cajas:

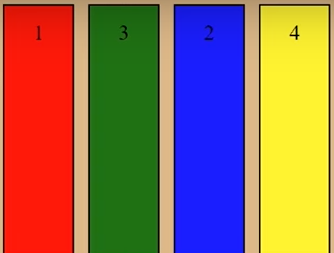


Luego se tienen para *align-content: flex start, flex end, center*

**Propiedades para los hijos:**

*Order:* Sirve para indicarle en qué orden queremos mostrar las cajas. Se irán acomodadno en números de order de menor a mayor:



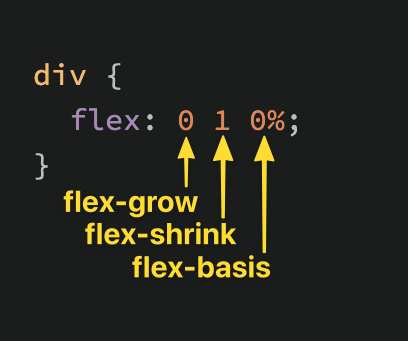


*Flex grow:* Sirve para indicar qué tanto “Derecho de piso” tiene cada una de las cajas. Cuanto mayor sea su flex grow, mayor tamaño ocupará esa caja. Por defecto todas las cajas tienen el mismo flex grow, por eso se ven todas del mismo tamaño. El análogo a *flex grow* es *flex shrink*

*Flex basis:* Es similar al width. Sirve para indicar qué tanto ancho se quiere para la caja.

*Flex:* A secas. Sirve para dar en una sola línea las propiedades de grow, shrink y basis acotadas en una sola línea.

**Flex**



Si se escribe flex: 1 = flex: 1 1 0%.

So far everything we’ve touched with flexbox has used the rule flex: 1 on all flex items, which makes the items grow or shrink equally to fill all of the available space.

**Flex-grow**

La propiedad **flex-grow** de [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) especifica el factor de crecimiento de un elemento flexible (que tiene asignado display:flex), en su dirección principal. El factor de crecimiento especifica qué cantidad del espacio restante dentro del contenedor flexible, debería ocupar el item en cuestión.

La dirección principal puede ser la altura o el ancho del elemento, dependiendo del valor de [flex-direction](https://developer.mozilla.org/es/docs/Web/CSS/flex-direction).

El espacio restante es el tamaño del contenedor flexible menos el tamaño de todos los elementos flexibles juntos. Si todos los ítems dentro del contenedor tienen el mismo factor de crecimiento, todos los elementos reciben la misma cantidad del espacio restante. De lo contrario, el espacio restante se distribuye en función de los diferentes factores de crecimientos de cada item.

SI estamos con flex-direction: row, entonces si se aplica un flex-grow:1, todos los flex ítems crecerán de tamaño en un 100% hacia los costados, no hacia arriba y abajo

Imagen de la pantalla de un celular

Descripción generada automáticamente con confianza baja

**Flex-shrink**

La propiedad [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) **flex-shrink** especifica el factor de contracción de un flex item. Los flex items se encogerán para llenar el contenedor de acuerdo a su número flex-shrink , cuando el tamaño por defecto de los flex items sea mayor al de su contenedor flex container.

Texto

Descripción generada automáticamente

**Flex basis**

La propiedad de [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) **flex-basis** especifíca la base flexible, la cual es el **tamaño inicial** de un elemento flexible. Ésta propiedad determina el tamaño de una caja de contenidos a no ser que se haya especificado de otra forma usando [box-sizing](https://developer.mozilla.org/es/docs/Web/CSS/box-sizing).

**Alinear contenido en un flexbox**

1. Por un lado se tiene el método **flex**. Con este método vamos a poder agregar y achicar los contenidos hasta que ocupen el tamaño máximo disponible del contenedor. Al utilizar flex, los objetos perderán sus dimensiones original y se estirarán a lo largo o ancho (según la flex direction)
2. **justify-content**: Alinea los elementos a lo largo del **eje principal** sin modificar dimensión
   1. space-between
   2. flex-start
   3. flex-end
   4. center
   5. space-around
3. **align-items:** Alinea los elementos a lo largo del **eje secundario** sin modificar dimensión
   1. stretch
   2. center
   3. start (flex-start)
   4. end (flex-end)
   5. normal
   6. baseline
4. **align-self**: Alinea los elementos de la línea flexible actual, reemplazando el valor de *align-items*.. Por defecto es *auto*. Es decir, sirve para sobreescribir la propiedad de *align-items* del contenedor.
5. **align-content:** Funciona de manera similar al *align-items* pero en lugar de alinear ítems, alínea flexlines. Por lo general se usa con wrap

**Flex Wrap:**

**Flex-wrap:** Determina si todos los ítems deben ser forzados a estar en única línea o pueden estar distribuidos entre varias

* 1. *no wrap:* Todos los elementos en única línea aunque pueden salirse del contenedor
  2. *wrap:* Pueden ir en distintas líneas
  3. *wrap-reverse*

**Flex-flow:**

Es la combinación de Flow-direction y Flow-wrap por lo muy utilizados que son.

Flex-flow: row wrap