TIPOS DE POSICIONAMIENTO DE LAS CSS

Posicionamiento CSS es un módulo de CSS que define cómo posicionar elementos absoluta y relativamente en la página.

STATIC

Es el valor predeterminado del atributo y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML, es decir, según estén escritos en el propio código HTML. Por decirlo de otra manera, static no provoca ningún posicionamiento especial de los elementos y por tanto, los atributos top, left, right y bottom no se tendrán en cuenta.

```
<div style="position: static;
            background-color: #ff9;
            padding: 10px;
            width: 300px;">
    Esto es una capa con posicionamiento estático</div>

<div style="position: static;
            background-color: #f9f;
            padding: 10px;
            width: 500px;">
    posicionamiento static, predeterminado.</div>

<h1>CSS</h1>

<div style="background-color: #9ff;
            padding: 10px;
            width: 400px;">
    Posicionamiento static, aunque en este caso no se
    indicó el atributo position static, pues no hace falta.
</div>
```

Esto es una capa con posicionamiento estático

posicionamiento static, predeterminado.

CSS

Posicionamiento static, aunque en este caso no se indicó el atributo position static, pues no hace falta.

Posicionamiento CSS

Esta capa tiene posicionamiento absoluto.

Me permite especificar top y left para colocarla con respecto a la esquina superior izquierda.

or (la capa aparece por debajo de otras que se

```
<div style="position: absolute;
            width: 300px;
            height: 140px;
            top: 100px;
            left: 30px;
            background-color: #ff8800;
            color: #fff;
            padding: 15px;
            z-index: 2;">
    Esta capa tiene posicionamiento absoluto.

    <br>
    <br>

    Me permite especificar top y left para colocarla
    con respecto a la esquina superior izquierda.
</div>

<div style="position: absolute;
            width: 820px;
            height: 30px;
            padding: 10px;
            background-color: #ddf;
            top: 150px; left: 10px;
            z-index: 1;">
    Posicionamiento absoluto con z-index menor (la
    capa aparece por debajo de otras que se superponen
    con z-index mayor.
</div>

<div style="position: absolute;
            width: 100px;
            height: 20px;
            padding: 10px;
            background-color: #ddf;
            bottom: 10px;
            right: 10px;">
    Posicionamiento absoluto con atributos bottom y right
</div>

    <h1>Posicionamiento CSS</h1>
```

ABSOLUTE

El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el elemento en el esquema de la página. Es posicionado relativo a su ancestro posicionado más cercano, si lo hay; de lo contrario, se ubica relativo al bloque contenedor inicial. Su posición final está determinada por los valores de top, right, bottom, y left.
Este valor crea un nuevo contexto de apilamiento cuando el valor de z-index no es auto. Elementos absolutamente posicionados pueden tener margen, y no colapsan con ningún otro margen.

RELATIVE

El elemento es posicionado de acuerdo al flujo normal del documento, y luego es desplazado con relación a sí mismo, con base en los valores de top, right, bottom, and left. El desplazamiento no afecta la posición de ningún otro elemento; por lo que, el espacio que se le da al elemento en el esquema de la página es el mismo como si la posición fuera static. Este valor crea un nuevo contexto de apilamiento, donde el valor de z-index no es auto. El efecto que tiene relative sobre los elementos table-*-group, table-row, table-column, table-cell, y table-caption no está definido.

```
<h1>Hola</h1>

<div style="background-color: #606;
            color:#ffc;
            padding:10px;
            text-align: center;
            width: 300px;">
            Hola esto es una prueba
</div>
<div style="position: relative;
            width: 300px;
            padding: 10px;
            background-color: #066;
            color:#ffc;
            top:100px;
            left: 30px;">
            capa de posicionamiento relative
            <br>
            Se tiene en cuenta esta capa para
            posicionar las siguientes.
</div>
<h2>hola de nuevo!</h2>
```

Hola

Hola esto es una prueba

ho

capa de posicionamiento relative
Se tiene en cuenta esta capa para posicionar las siguientes.

```
<div style="position: fixed;
            width: 300px;
            height: 140px;
            top: 100px;
            left: 30px;
            background-color: #ff8800;
            color: #fff;
            padding: 15px;
            z-index: 1;">
    Esta capa tiene posicionamiento fixed.

    <br>
    <br>

    Me permite especificar top y left para
    colocarla con respecto a la esquina
    superior izquierda.
</div>

<div style="position: fixed;
            width: 700px;
            height: 30px;
            padding: 10px;
            background-color: #d0f;
            top: 150px;
            left: 10px;
            z-index: 2;">
    Posicionamiento fixed
</div>

    <h1>Hola</h1>

<div style="position: fixed;
            width: 100px;
            height: 30px;
            padding: 10px;
            background-color: #0df;
            bottom: 10px;
            right: 10px;
            z-index: 4;">
            Posicionamiento fixed
</div>
```

Hola

Esta capa tiene posicionamiento fixed.

Posicionamiento fixed

para colocarla con respecto a la esquina superior izquierda.

FIXED

El elemento es removido del flujo normal del documento, sin crearse espacio alguno para el elemento en el esquema de la página. Es posicionado con relación al bloque contenedor inicial establecido por el viewport, excepto cuando uno de sus ancestros tiene una propiedad transform, perspective, o filter establecida en algo que no sea none, en cuyo caso ese ancestro se comporta como el bloque contenedor. Su posición final es determinada por los valores de top, right, bottom, y left.

STICKY

El elemento es posicionado de acuerdo al flujo normal del documento, y luego es desplazado con relación a su ancestro que se desplace más cercano y su bloque contenedor (ancestro en nivel de bloque más cercano) incluyendo elementos relacionados a tablas, basados en los valores de top, right, bottom, y left. El desplazamiento no afecta la posición de ningún otro elmento.

```
    <style>
        .dt {
            background: #B8C1C8;
            border-bottom: 1px solid #989EA4;
            border-top: 1px solid #717D85;
            color: #FFF;
            font: bold 18px/21px Helvetica, Arial, sans-serif;
            margin: 0;
            padding: 2px 0 0 12px;
            position: -webkit-sticky;
            position: sticky;
            top: -1px;
        }
    </style>
</head>
<body>
    <p class="dt">hola</p>
    hola
    Lorem ipsum dolor
</body>
</html>
```

hola

hola Lorem ipsum dolor, sit amet consectetur adipisicing elit. Alias, officiis dolorem! Delectus, minima tenetur facilis placeat, eius vitae qui ipsa dolore sint porro dolor veniam, reprehenderit pariatur

CASTELAN CONTRERAS ANA YULENI

ELEMENTOS FLOTANTES

La propiedad CSS float ubica un elemento al lado izquierdo o derecho de su contenedor, permitiendo a los elementos de texto y en línea aparecer a su costado. El elemento es removido del normal flujo de la página, aunque aún sigue siendo parte del flujo

Un elemento flotante es un elemento en el que el valor calculado de float no es igual a none. Como float implica el uso del layout de bloques, este modifica el valor calculado de los valores display , en algunos casos:

VALOR ESPECIFICADO	VALOR COMPUTADO
inline	block
inline-block	block
inline-table	table
table-row	block
table-row-group	block
table-column	block
table-column-group	block
table-cell	block
table-caption	block
table-header-group	block
table-footer-group	block
inline-flex	flex
inline-grid	grid
otros	sin cambios

VALORES

LEFT
El elemento debe flotar a la izquierda de su bloque contenedor.

RIGHT
El elemento debe flotar a la derecha de su bloque contenedor.

NONE
El elemento no deberá flotar.

INLINE-START
El elemento debe flotar en el costado de inicio de su bloque contenedor. Esto es el lado izquierdo con scripts ltr y el lado derecho con scripts rtl.

INLINE-END
El elemento debe flotar en el costado de término de su bloque contenedor . Esto es el lado derecho con scripts ltr y el lado izquierdo con scripts rtl.

SINTAXIS

```
/* Valores clave || Keyword values */
float: left;
float: right;
float: none;
float: inline-start;
float: inline-end;

/* Valores globales || Global values */
float: inherit;
float: initial;
float: unset;
```

EJEMPLO

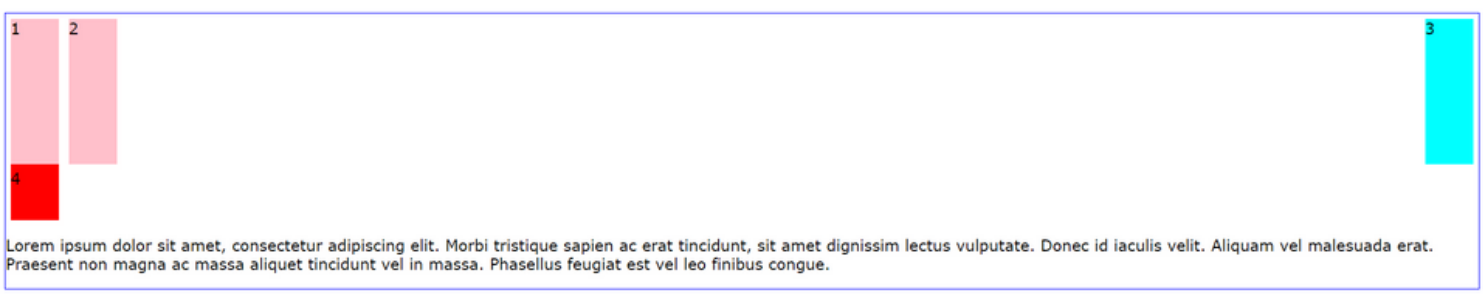
```
<style>
    section {
    border: 1px solid blue;
    width: 100%;
    float: left;
    }

    div {
    margin: 5px;
    width: 50px;
    height: 150px;
    }
```

```
    .left {
    float: left;
    background: pink;
    }

    .right {
    float: right;
    background: cyan;
    }
    .none{
    float: none;
    background: red;
    }
</style>
```

```
<body>
    <section>
        <div class="left">1</div>
        <div class="left">2</div>
        <div class="right">3</div>
        <br><br><br>
        <div class="none">4</div>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
           Morbi tristique sapien ac erat tincidunt, sit amet dignissim
           lectus vulputate. Donec id iaculis velit. Aliquam vel
           malesuada erat. Praesent non magna ac massa aliquet tincidunt
           vel in massa. Phasellus feugiat est vel leo finibus congue.</p>
      </section>
</body>
```



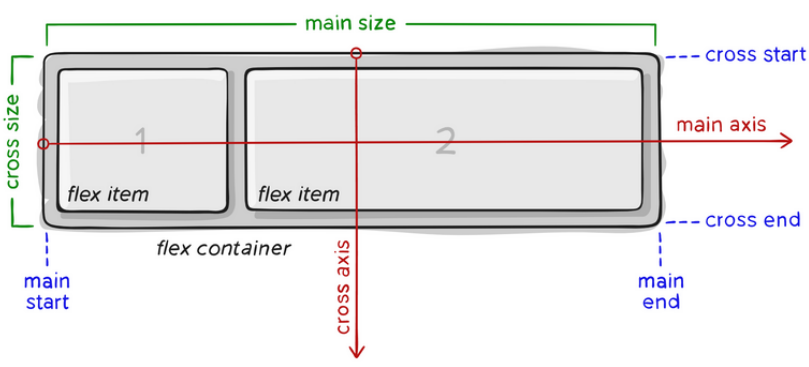
CASTELAN CONTRERAS ANA YULENI

FLEXIBLE BOX LAYOUT

El Módulo de Caja Flexible, comúnmente llamado flexbox, fue diseñado como un modelo unidimensional de layout, y como un método que pueda ayudar a distribuir el espacio entre los ítems de una interfaz y mejorar las capacidades de alineación.

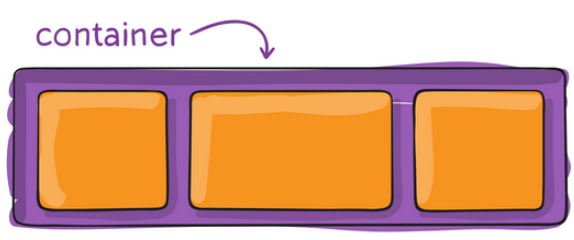
CONCEPTOS BASICOS Y TERMINOLOGIA

- MAIN AXIS: el eje principal de un contenedor flexible es el eje principal a lo largo del cual se colocan los elementos flexibles. Cuidado, no es necesariamente horizontal; depende de la flex-directionpropiedad (ver más abajo).
- MAIN-STRAT/END: los elementos flexibles se colocan dentro del contenedor comenzando desde el inicio principal hasta el final principal.
- MAIN SIZE : el ancho o alto de un elemento flexible, cualquiera que esté en la dimensión principal, es el tamaño principal del elemento. La propiedad de tamaño principal del elemento flexible es la propiedad ancho' o 'alto', cualquiera que esté en la dimensión principal.



- CROSS AXIS : el eje perpendicular al eje principal se denomina eje transversal. Su dirección depende de la dirección del eje principal.
- CROSS-START/END: las líneas flexibles se llenan con artículos y se colocan en el contenedor comenzando en el lado de inicio cruzado del contenedor flexible y yendo hacia el lado del extremo transversal.
- CROSS SIZE: el ancho o alto de un artículo flexible, cualquiera que esté en la dimensión cruzada, es el tamaño cruzado del artículo. La propiedad de tamaño cruzado es cualquiera de 'ancho' o 'alto' que esté en la dimensión cruzada.

PROPIEDADES PARA EL PADRE (FLEX CONTAINER)



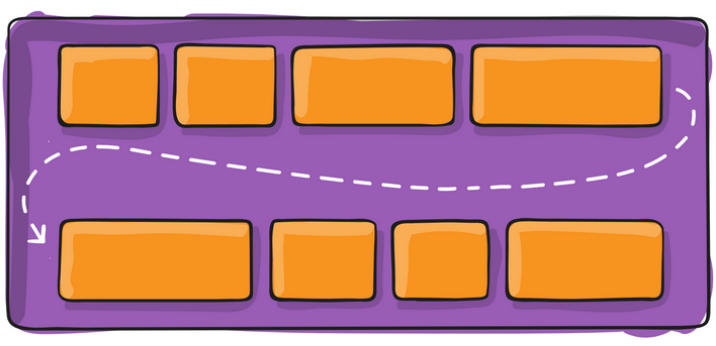
DISPLAY

Esto define un contenedor flexible; en línea o en bloque según el valor dado. Permite un contexto flexible para todos sus hijos directos.
.container {display: flex; /* or inline-flex */}
Tenga en cuenta que las columnas CSS no tienen ningún efecto en un contenedor flexible.

FLEX-WRAP

De forma predeterminada, todos los elementos flexibles intentarán encajar en una línea. Puede cambiar eso y permitir que los elementos se ajusten según sea necesario con esta propiedad.
.container {flex-wrap: nowrap | wrap | wrap-reverse;}

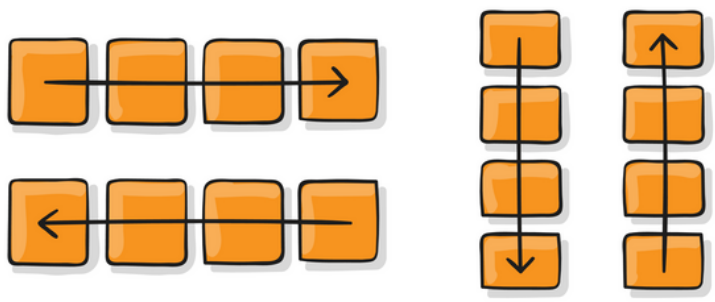
- nowrap (predeterminado): todos los elementos flexibles estarán en una línea
- wrap: los elementos flexibles se ajustarán a varias líneas, de arriba a abajo.
- wrap-reverse: los elementos flexibles se ajustarán a varias líneas de abajo hacia arriba.



FLEX-DIRECTION

Esto establece el eje principal, definiendo así la dirección en que se colocan los elementos flexibles en el contenedor flexible.
.container {flex-direction: row | row-reverse | column | column-reverse;}

- row(predeterminado): de izquierda a derecha en ltr; de derecha a izquierda enrtl
- row-reverse: de derecha a izquierda adentro ltr; de izquierda a derecha enrtl
- column: igual que rowpero de arriba a abajo
- column-reverse: igual que row-reversepero de abajo hacia arriba



FLEX-FLOW

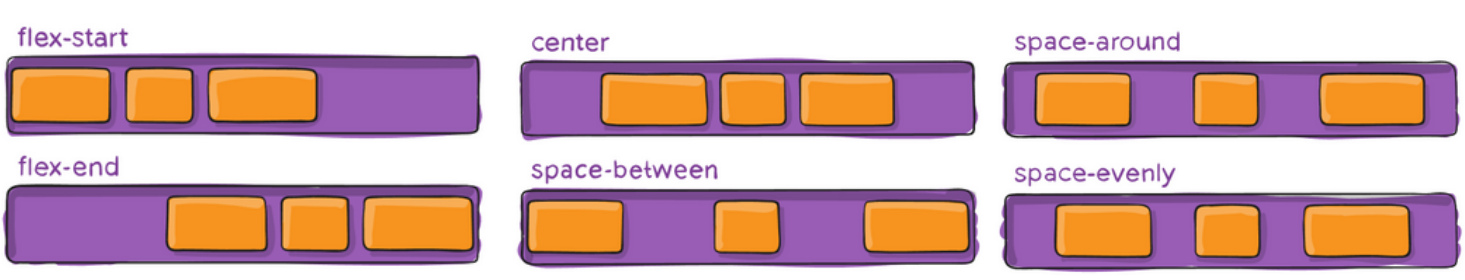
Esta es una abreviatura de las propiedades flex-directiony flex-wrap, que juntas definen los ejes principal y transversal del contenedor flexible. El valor predeterminado es row nowrap.
.container {flex-flow: column wrap;}

JUSTIFY-CONTENT

Esto define la alineación a lo largo del eje principal. Ayuda a distribuir el espacio libre adicional que queda cuando todos los elementos flexibles de una línea son inflexibles o son flexibles pero han alcanzado su tamaño máximo. También ejerce cierto control sobre la alineación de los elementos cuando desbordan la línea.
.container {justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;}

- flex-start (predeterminado): los elementos se empaquetan hacia el inicio de la dirección flexible.
- flex-end: los artículos se empaquetan hacia el final de la dirección de flexión.
- start: los artículos se empaquetan hacia el inicio de la writing-modedirección.
- end: los artículos se empaquetan hacia el final de la writing-modedirección.
- left: los artículos se empaquetan hacia el borde izquierdo del contenedor.
- right: los artículos se empaquetan hacia el borde derecho del contenedor.
- center: los elementos están centrados a lo largo de la línea.
- space-between: los artículos se distribuyen uniformemente en la línea.
- space-around: los elementos se distribuyen uniformemente en la línea con el mismo espacio alrededor.
- space-evenly: los elementos se distribuyen de modo que el espacio entre dos elementos cualesquiera sea igual.

También hay dos palabras clave adicionales que puede emparejar con estos valores: safe y unsafe. El uso safe garantiza que, independientemente de cómo realice este tipo de posicionamiento, no puede empujar un elemento de manera que se muestre fuera de la pantalla de tal manera que el contenido no se pueda desplazar.



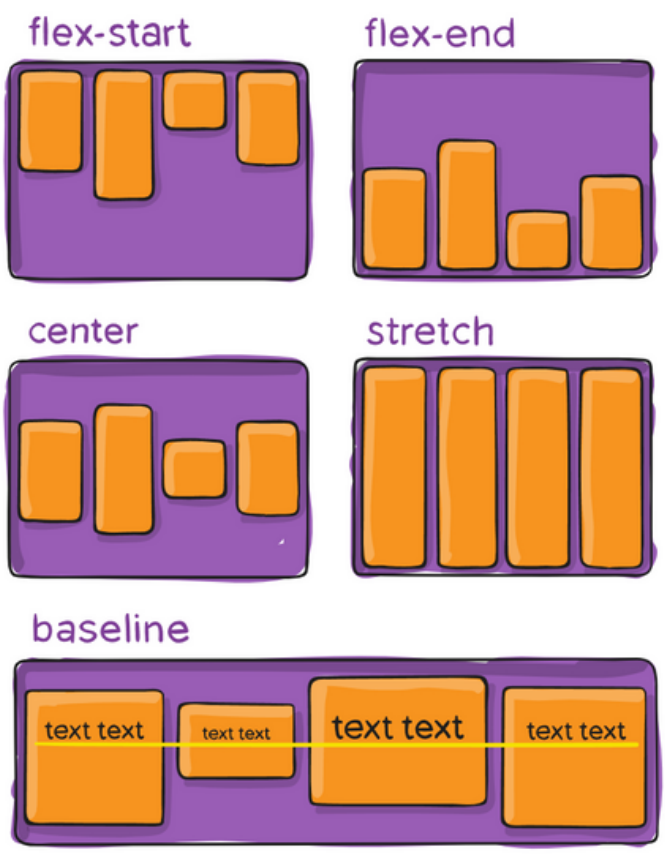
CASTELAN CONTRERAS ANA YULENI
DOMINGUEZ DURÁN ALAN AXEL

ALIGN-ITEMS

Esto define el comportamiento predeterminado de cómo se distribuyen los elementos flexibles a lo largo del eje transversal en la línea actual. Piense en ello como la justify-content versión del eje transversal.
.container {align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe;}

- stretch (predeterminado): estirar para llenar el contenedor (aún respetando el ancho mínimo / ancho máximo)
- flex-start/ start/ self-start: Elementos se colocan en el inicio del eje transversal.
- flex-end/ end/ self-end: Elementos se colocan en el fin del eje transversal.
- center: los elementos están centrados en el eje transversal
- baseline: los elementos están alineados, como sus líneas de base se alinean

Las palabras clave safe y unsafe modificadoras se pueden usar junto con el resto de estas palabras clave, y tratan de ayudarlo a evitar la alineación de elementos de modo que el contenido se vuelva inaccesible.

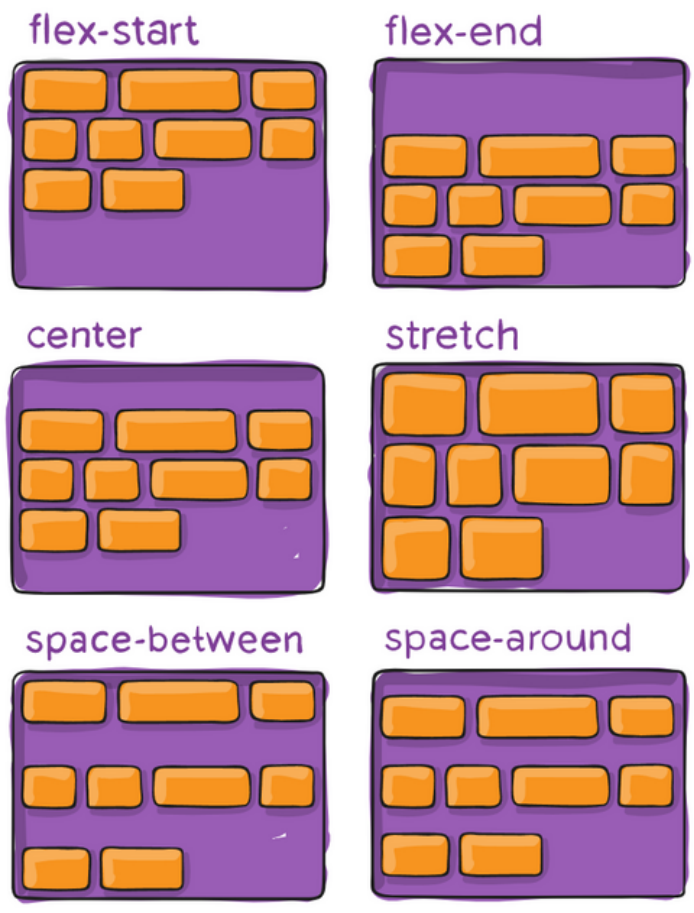


ALIGN-CONTENT

Esto alinea las líneas de un contenedor flexible dentro cuando hay espacio adicional en el eje transversal, similar a cómo justify-content alinea elementos individuales dentro del eje principal.
Nota: esta propiedad solo tiene efecto en contenedores flexibles de varias líneas, donde flex-wrap se establece en wrap o wrap-reverse). Un contenedor flexible de una sola línea (es decir, donde flex-wrap se establece en su valor predeterminado no-wrap) no reflejará align-content.
.container {align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;}

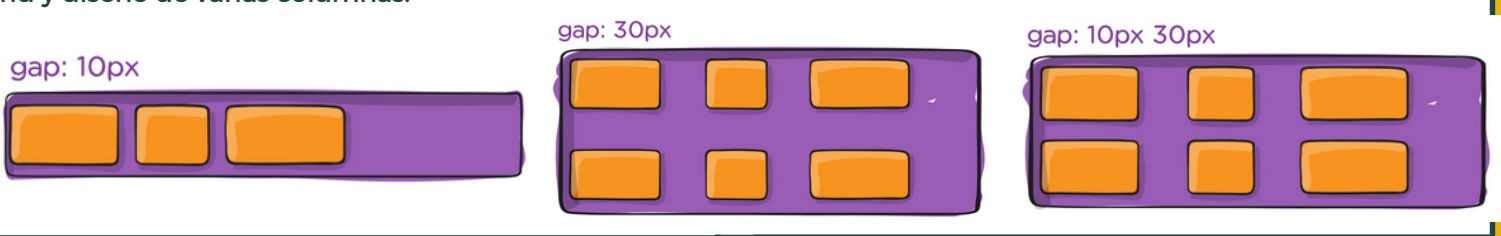
- normal (predeterminado): los elementos se empaquetan en su posición predeterminada como si no se hubiera establecido ningún valor.
- flex-start/ start: artículos empaquetados al inicio del contenedor.
- flex-end/ end: artículos empaquetados hasta el final del contenedor.
- center: elementos centrados en el contenedor
- space-between: artículos distribuidos uniformemente del inicio al final.
- space-around: elementos distribuidos uniformemente con el mismo espacio alrededor de cada línea
- space-evenly: los elementos se distribuyen uniformemente con el mismo espacio a su alrededor
- stretch: las líneas se estiran para ocupar el espacio restante

Las palabras clave safe y unsafe modificadoras se pueden usar junto con el resto de estas palabras clave, y tratan de ayudarlo a evitar la alineación de elementos de modo que el contenido se vuelva inaccesible.

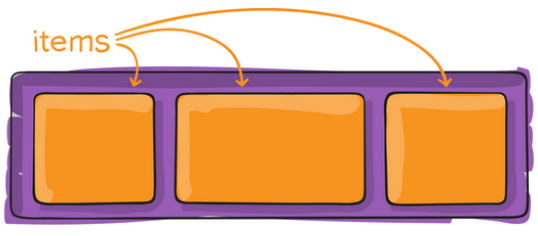


GAP/RAW-GAP/COLUMN-GAP

La propiedad gap controla explícitamente el espacio entre elementos flexibles. Se aplica ese espaciado solo entre elementos, no en los bordes exteriores.
.container {display: flex;...gap: 10px;gap: 10px 20px; /* row-gap column gap */row-gap: 10px;column-gap: 20px;}
El comportamiento podría considerarse como un canalón mínimo , como si el canalón fuera más grande de alguna manera, entonces el espacio solo tendrá efecto si ese espacio terminara siendo más pequeño.
No es exclusivo para flexbox, también gap funciona en grid y diseño de varias columnas.

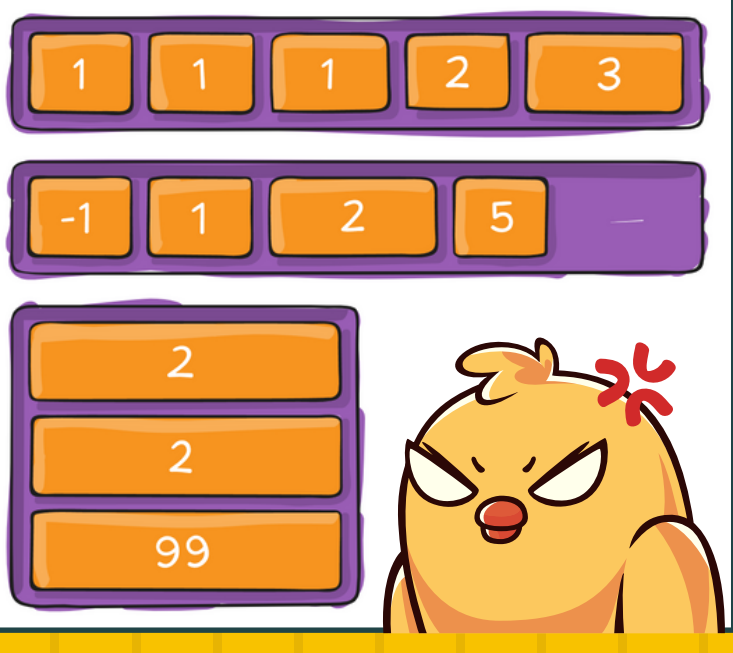


PROPIEDADES PARA EL HIJO (FLEX ITEM)



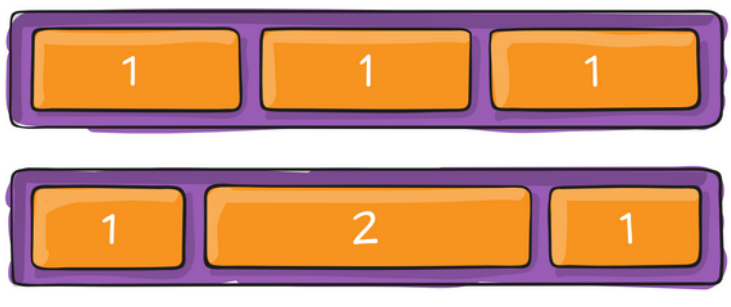
ORDER

De forma predeterminada, los elementos flexibles se distribuyen en el orden de origen. Sin embargo, la propiedad order controla el orden en que aparecen en el contenedor flexible.
.item {order: 5; /* default is 0 */}
Los artículos con el mismo order vuelven al orden de origen.



FLEX-GROW

Esto define la capacidad de un artículo flexible para crecer si es necesario. Acepta un valor sin unidades que sirve como proporción. Dicta la cantidad de espacio disponible dentro del contenedor flexible que debe ocupar el artículo.
Si todos los elementos se han establecido en 1, el espacio restante en el contenedor se distribuirá por igual a todos los hijos. Si uno de los hijos tiene un valor de 2, el espacio restante ocuparía el doble de espacio que los demás (o lo intentará, al menos).
.item {flex-grow: 4; /* default 0 */}
Los números negativos no son válidos.



FLEX-SHRINK

Esto define la capacidad de un artículo flexible de encogerse si es necesario.
.item {flex-shrink: 3; /* default 1 */}
Los números negativos no son válidos.

CASTELAN CONTRERAS ANA YULENI

FLEX-BASIS

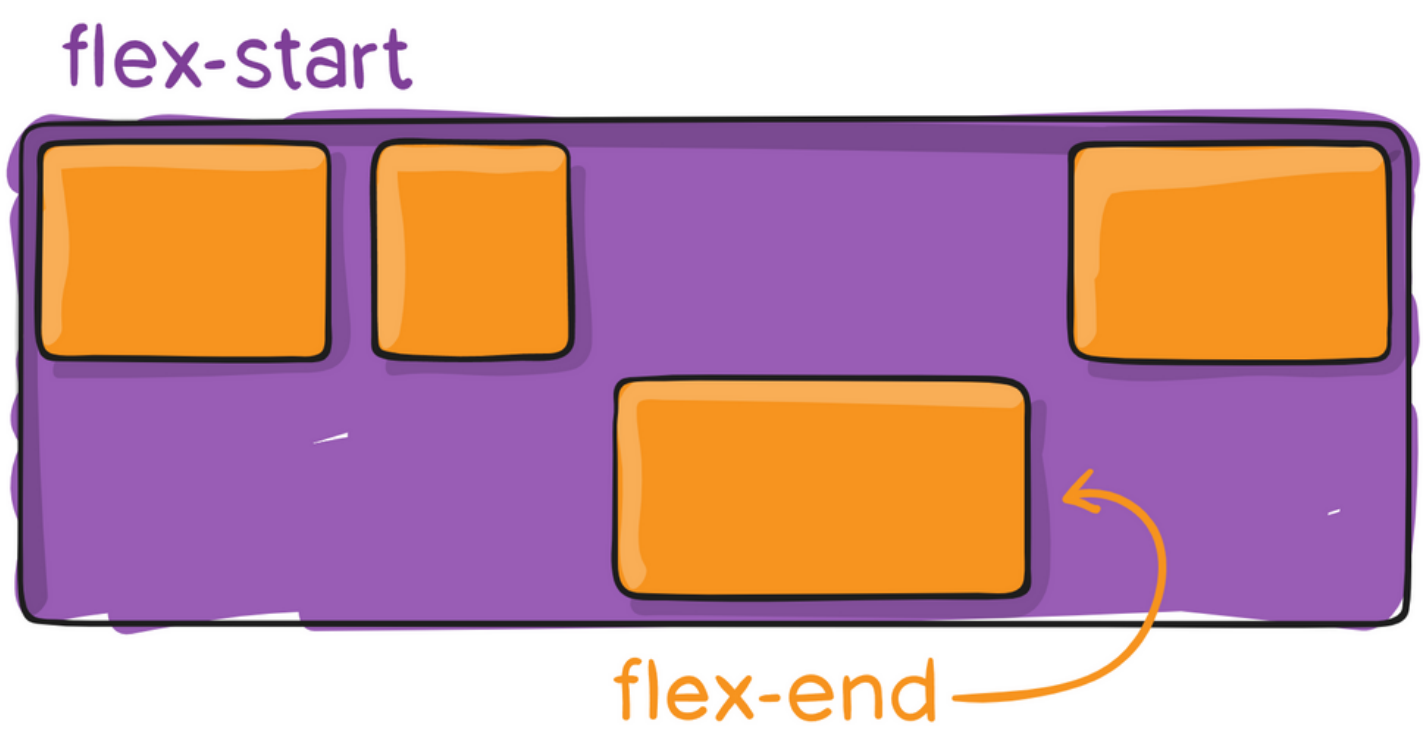
Esto define el tamaño predeterminado de un elemento antes de que se distribuya el espacio restante. Puede ser una longitud (por ejemplo, 20%, 5rem, etc.) o una palabra clave. La palabra clave significa "mirar mi propiedad de ancho o alto" (que fue hecho temporalmente por la main-size palabra clave hasta que quedó obsoleto). Los content medios de la palabra clave “tamaño se basa en el contenido del elemento” - esta palabra clave no está debidamente apoyado, sin embargo, por lo que es difícil de probar y más difícil saber lo que sus hermanos max-content, min-content y fit-content lo hacen.
.item {flex-basis: | auto; /* default auto */}
Si se establece en 0, el espacio adicional alrededor del contenido no se tiene en cuenta. Si se establece en auto, el espacio adicional se distribuye en función de su flex-growvalor.

FLEX

Esta es la abreviatura de flex-grow, flex-shrinky flex-basiscombinado. El segundo y tercer parámetro (flex-shrinky flex-basis) son opcionales. El valor predeterminado es 0 1 auto, pero si lo configura con un solo valor numérico, es como 1 0.
.item {flex: none | [<'flex-grow'> <'flex-shrink'>? || <'flex-basis'>]
}
Se recomienda que utilice esta propiedad abreviada en lugar de establecer las propiedades individuales. La taquigrafía establece los otros valores de forma inteligente.

ALIGN-SELF

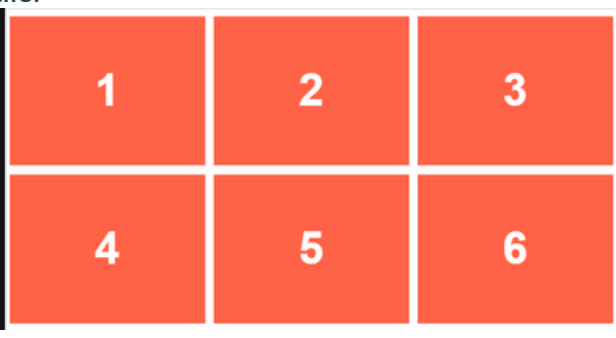
Esto permite que se anule la alineación predeterminada (o la especificada por align-items) para elementos flexibles individuales.
Consulte la align-itemsexplicación para comprender los valores disponibles.
.item {align-self: auto | flex-start | flex-end | center | baseline | stretch;}
Tenga en cuenta que float, cleary vertical-alignno tienen ningún efecto sobre un elemento flexible.



EJEMPLO

Empecemos por un ejemplo muy muy sencillo, resolviendo un problema casi diario: un centrado perfecto. No podría ser más simple si usa flexbox.
.parent {display: flex;height: 300px; /* Or whatever */}.child {width: 100px; /* Or whatever */height: 100px; /* Or whatever */margin: auto; /* Magic! */}
Esto se basa en el hecho de que un margen establecido auto en un contenedor flexible absorbe espacio adicional. Por lo tanto, establecer un margen de auto hará que el elemento esté perfectamente centrado en ambos ejes.
Ahora usemos algunas propiedades más. Considere una lista de 6 elementos, todos con dimensiones fijas, pero que se pueden ajustar automáticamente. Queremos que estén distribuidos uniformemente en el eje horizontal para que cuando cambiemos el tamaño del navegador, todo se adapte bien y sin consultas de medios.
.flex-container {/* We first create a flex layout context */display: flex;/* Then we define the flow direction and if we allow the items to wrap
*** Remember this is the same as:**
*** flex-direction: row;**
*** flex-wrap: wrap;**
***/flex-flow: row wrap;/* Then we define how is distributed the remaining space */justify-content: space-around;}**
Hecho. Todo lo demás es solo una preocupación de estilo.

```
<ul class="flex-container">
  <li class="flex-item">1</li>
  <li class="flex-item">2</li>
  <li class="flex-item">3</li>
  <li class="flex-item">4</li>
  <li class="flex-item">5</li>
  <li class="flex-item">6</li>
</ul>
```

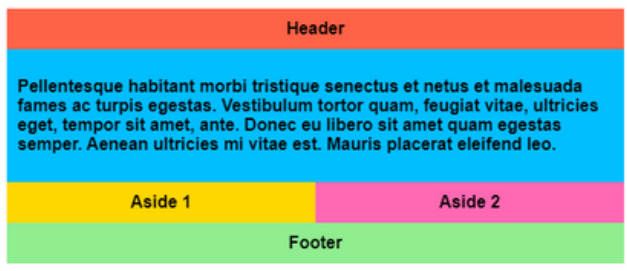


Probemos algo más. Imagine que tenemos un elemento de navegación alineado a la derecha en la parte superior de nuestro sitio web, pero queremos que esté centrado en pantallas de tamaño mediano y de una sola columna en dispositivos pequeños. Suficientemente fácil.
/* Large */.navigation {display: flex;flex-flow: row wrap;/* This aligns items to the end line on main-axis */justify-content: flex-end;}/* Medium screens */@media all and (max-width: 800px) {.navigation {/* When on medium sized screens, we center it by evenly distributing empty space around items */justify-content: space-around;}}/* Small screens */@media all and (max-width: 500px) {.navigation {/* On small screens, we are no longer using row direction but column */flex-direction: column;}}

```
<ul class="navigation">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Products</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```

¡Intentemos algo aún mejor jugando con la flexibilidad de los elementos flexibles! ¿Qué tal un diseño de 3 columnas para dispositivos móviles con encabezado y pie de página de ancho completo? E independiente del orden de origen.
.wrapper {display: flex;flex-flow: row wrap;}/* We tell all items to be 100% width, via flex-basis */.wrapper > * {flex: 1 100%;}/* We rely on source order for mobile-first approach
*** in this case:**
*** 1. header**
*** 2. article**
*** 3. aside 1**
*** 4. aside 2**
*** 5. footer**
***//* Medium screens */@media all and (min-width: 600px) {/* We tell both sidebars to share a row */.aside { flex: 1 auto; }}/* Large screens */@media all and (min-width: 800px) {/* We invert order of first sidebar and main**
*** And tell the main element to take twice as much width as the other two sidebars**
***/.main { flex: 2 0px; }.aside-1 { order: 1; }.main { order: 2; }.aside-2 { order: 3; }.footer { order: 4; }}**

```
<div class="wrapper">
  <header class="header">Header</header>
  <article class="main">
    <p>Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac
turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet,
ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est.
Mauris placerat eleifend leo.</p>
  </article>
  <aside class="aside aside-1">Aside 1</aside>
  <aside class="aside aside-2">Aside 2</aside>
  <footer class="footer">Footer</footer>
</div>
```



CASTELAN CONTRERAS ANA YULENI