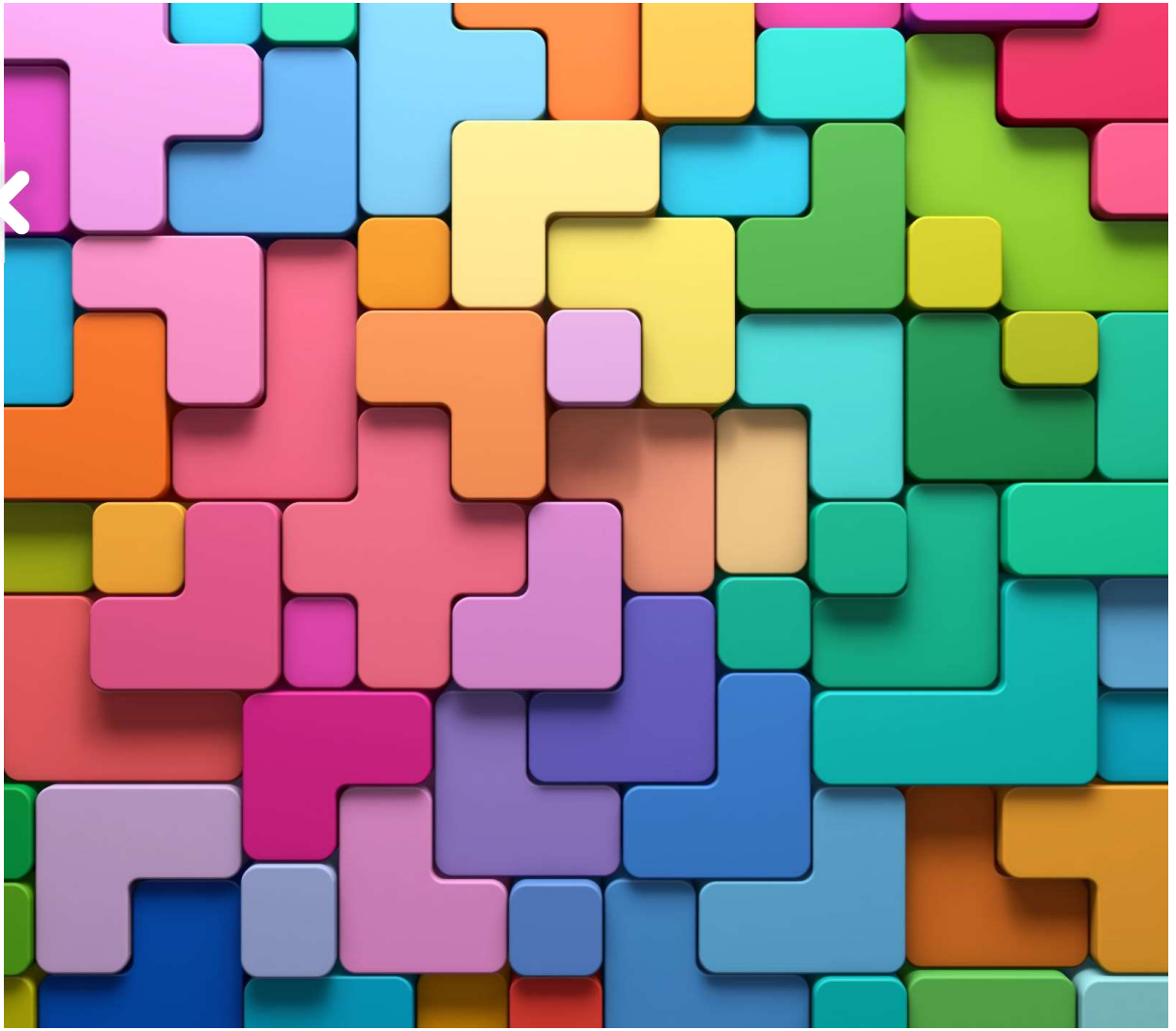
DISPLAY



Los elementos HTML se dividen en dos grandes tipos:

+ elementos **block** que tienden a ocupar el espacio disponible a todo lo ancho y en caso de existir varios se sitúan unos debajo de otros
+ elementos **inline** que ocupan el espacio necesario dentro de una línea y en caso de existir varios se sitúan uno junto a otro en la misma línea (siempre que haya espacio).

La propiedad **display** nos permite alterar el tipo de caja con que se muestra un elemento.

Por ejemplo, los elementos tipo título como h1 son elementos block, y por tanto de forma “natural” tienden a ocupar todo el ancho de la página. Por el contrario, elementos como los links, a, o las imágenes son elementos inline.

Veamos cómo la propiedad display permite alterar estas cualidades.

PROPIEDAD DISPLAY

Esta propiedad nos permite establecer el tipo de caja que el navegador empleará para visualizar un elemento, siendo los tipos más comunes inline y block, aunque existen bastantes otros.

La sintaxis a emplear es del tipo:

selectorElemento {display: especificaciónDeVisualización; }

Función de la propiedad: Permite definir el tipo de posición de caja para visualizar un elemento.

Valor por defecto: el del elemento (inline para elementos inline y block para elementos tipo block).

Aplicable a: Todos los elementos.

Valores posibles para esta propiedad

inline : el elemento se muestra en una caja inline

block: el elemento se muestra en una caja block

none: el elemento no se muestra; el efecto es como si no existiera, por lo que su espacio será ocupado por otros elementos

list-item: el elemento se comporta como si fuera un elemento li **inline-block:** el elemento genera una caja block pero que se comporta como si fuera inline admitiendo otros elementos en la misma línea; el comportamiento se asemeja al de los elementos img

Otros: que llevan a que el elemento simule el comportamiento de otro (inline- table, table, table-caption, table-cell, table-column, table-column-group, table- footer-group, table-header-group, table-row, table-row-group)

Otros avanzados: flex, inline-flex, **grid**, inline-grid, run-in

inherit : se heredan las características del elemento padre

content: el contenedor al que se aplica desaparece visualmente, pero sus hijos directos aún se muestran como si estuvieran directamente dentro del elemento padre del contenedor

La propiedad display admite numerosos valores, pero los más usados son inline, block e inline-block.

Amplia conocimientos sobre **display**:

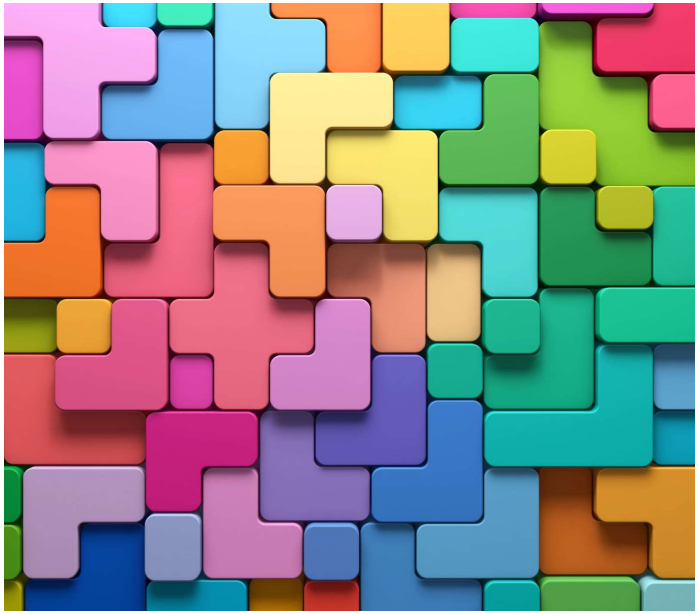
https://www.w3schools.com/cssref/pr_class_display.asp

https://www.w3schools.com/cssref/playit.asp?filename=playcss_display&preval=block

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_display_contents

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_display_inline2

https://www.w3schools.com/cssref/tryit.asp?filename=trycss3_flex-direction



POSITION

Se utiliza para controlar cómo se posiciona un elemento dentro del flujo normal del documento.

La sintaxis a emplear es del tipo:

selectorElemento {position: value; }

Valores posibles

static: los elementos se colocan según el flujo normal html (según estén escritos en el html) No se ven afectados por las propiedades top | right | botton | left

relative: El elemento se posiciona de forma relativa al flujo normal del documento, pero luego puede ajustarse mediante las propiedades top | right | botton | left

absolute: se posiciona de forma absoluta respecto a su primer ancestro posicionado (es decir, un elemento cuya posición no sea static. Si no hay un ancestro posicionado, se posicionará con respecto al elemento raíz del documento (el <html>) No siguen el flujo html, se colocan según las medidas que se indiquen a top | right | botton | left

fixed: igual que absolute, se posiciona de forma fija con respecto al viewport del navegador, pero al hacer scroll en la página no se mueve

Coordenadas de posicionamiento

Cuatro propiedades se encargan de mostrar un elemento en una posición exacta en la página:

left: Permite indicar la coordenada izquierda del elemento. Tomada desde el margen izquierdo

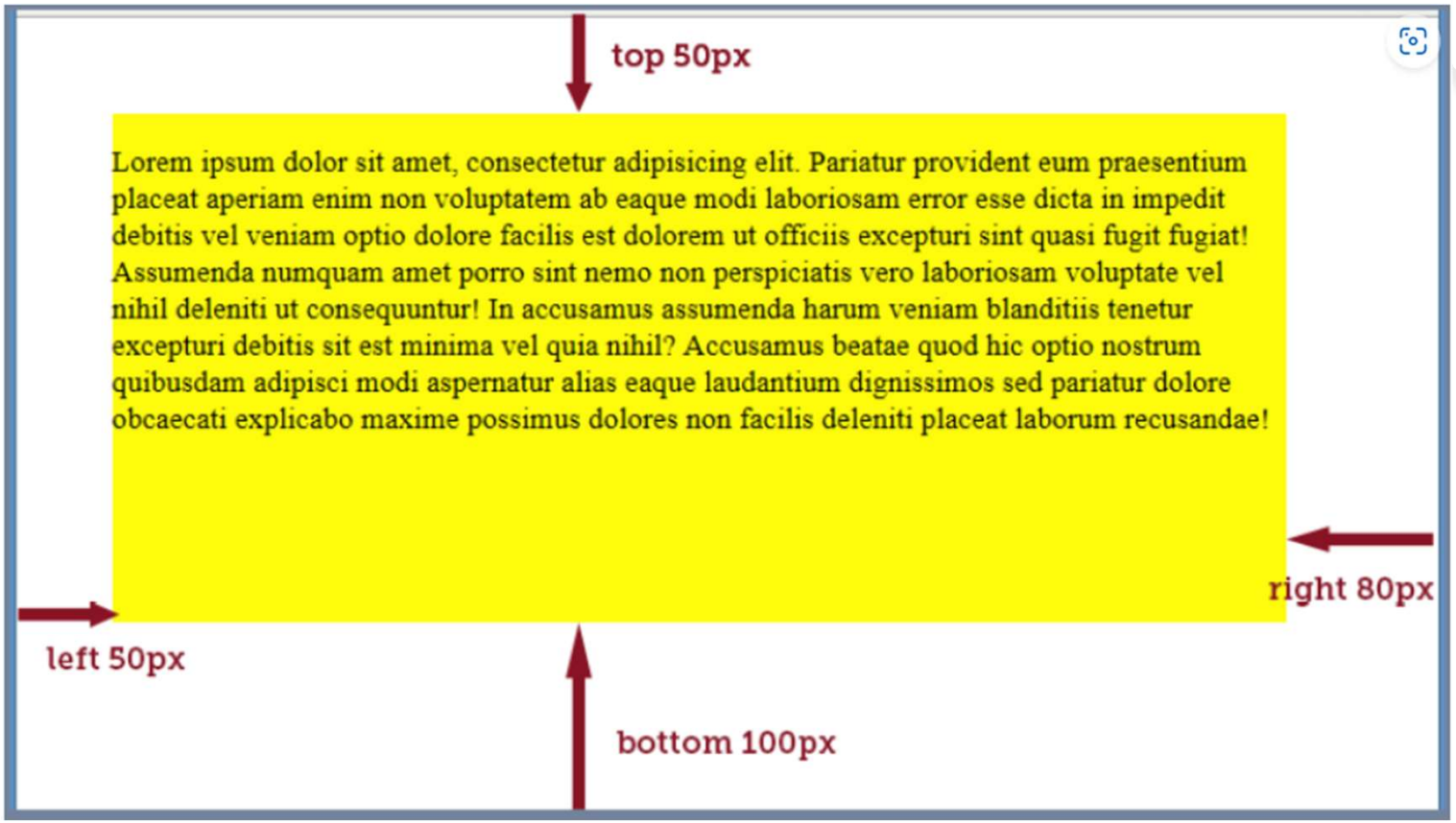
top: Coordenada superior. Tomada desde el borde superior.

bottom: Coordenada inferior. Distancia al borde inferior

right: Coordenada derecha. Distancia al borde derecho.

Las coordenadas se calculan desde la referencia de coordenadas establecidas por la propiedad **position**. Si es **fixed** se toma los bordes de la ventana.

Ejemplo:


top 50px
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Pariatur provident eum praesentium placeat aperiam enim non voluptatem ab eaque modi laboriosam error esse dicta in impedit debitis vel veniam optio dolore facilis est dolorem ut officiis excepturi sint quasi fugit fugiat! Assumenda numquam amet porro sint nemo non perspiciatis vero laboriosam voluptate vel nihil deleniti ut consequuntur! In accusamus assumenda harum veniam blanditiis tenetur excepturi debitis sit est minima vel quia nihil? Accusamus beatae quod hic optio nostrum quibusdam adipisci modi aspernatur alias eaque laudantium dignissimos sed pariatur dolore obcaecati explicabo maxime possimus dolores non facilis deleniti placeat laborum recusandae!
right 80px
left 50px
bottom 100px


Z-INDEX

Se utiliza para controlar el orden de apilamiento de elementos posicionados en un diseño.

Es especialmente útil cuando hay elementos superpuestos y se necesita especificar qué elemento debe mostrarse encima de los demás.

La sintaxis a emplear es del tipo:

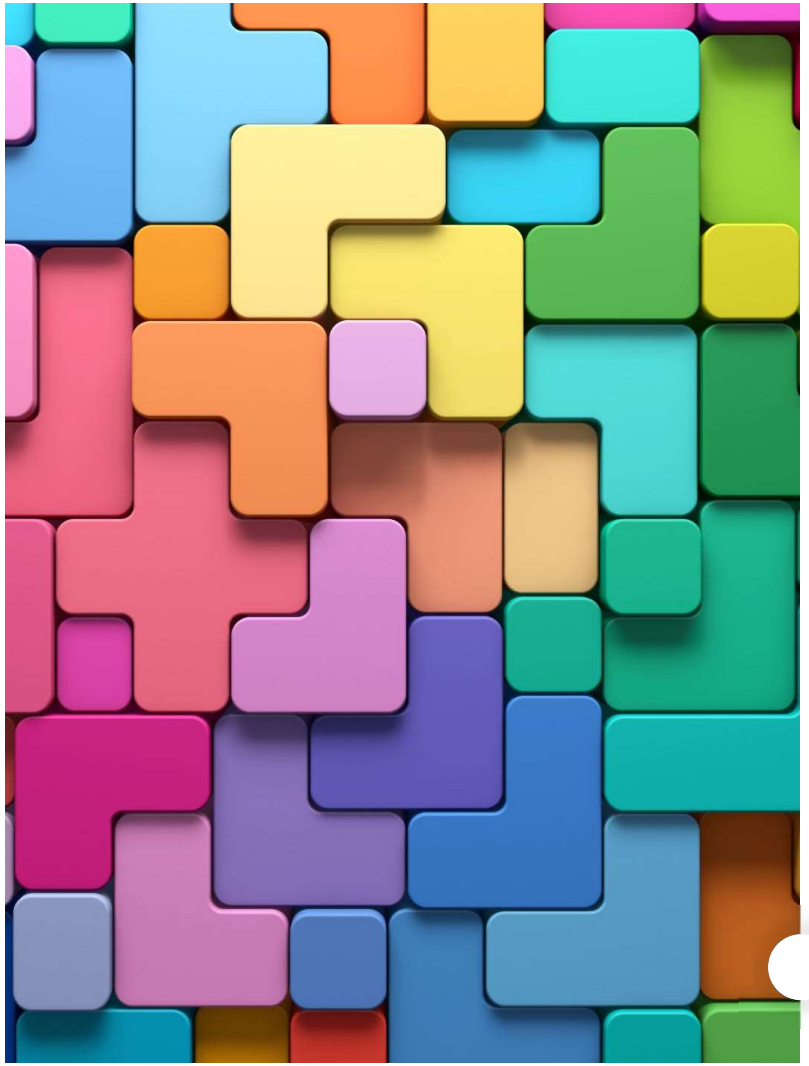
selectorElemento {z-index: value; }

Donde value es un número entero que representa la posición de apilamiento del elemento. Los valores más altos están más arriba en la pila de z-index (más cerca del usuario) y se muestran encima de los elementos con valores de z-index más bajos.

Se puede usar en elementos con posicionamiento absoluto, fijo o relativo para evitar superposiciones. Si no se indica z-index se hace de forma automática y el superpuesto más arriba es el último.

Prueba en: https://www.w3schools.com/cssref/tryit.asp?filename=trycss_zindex cambiando en el ejemplo a un número positivo

OVERFLOW



Se utiliza para controlar el comportamiento del contenido que desborda el área de su contenedor. Es decir, determina cómo se maneja el contenido que es más grande que el tamaño establecido para el contenedor.

Es útil cuando hay que controlar el contenido que es demasiado grande para caber dentro de su contenedor asignado.

La sintaxis a emplear es del tipo:

selectorElemento {overflow: value; }

visible: es el valor predeterminado. El contenido que desborda el contenedor se muestra fuera de él, superponiéndose a otros elementos si es necesario. Este valor no oculta ni recorta el contenido que desborda.
hidden: El contenido que desborda el contenedor se oculta y no se muestra en absoluto. No hay desplazamiento para ver el contenido oculto.
scroll: Se agrega una barra de desplazamiento al contenedor, permitiendo que el usuario desplace el contenido que desborda para verlo.
auto: Similar a **scroll**, pero las barras de desplazamiento solo se mostrarán si hay contenido que desborde. Si no hay contenido que desborde, no se mostrarán barras de desplazamiento.
inherit: La propiedad **overflow** hereda el valor de su elemento padre

https://www.w3schools.com/cssref/tryit.asp?filename=trycss_overflow


FLEXBOX

Se refiere a una serie de propiedades CSS relacionadas con el modelo de diseño flexible (Flexbox). Estas propiedades se utilizan para controlar el diseño y la disposición de los elementos en un contenedor flexible.

La clave principal para habilitar Flexbox: Una vez que un contenedor se ha convertido en un contenedor flexible utilizando **display:flex;** se pueden usar otras propiedades de Flexbox para controlar cómo se distribuyen y alinean los elementos dentro de ese contenedor.

Propiedades principales:

flex-direction: Especifica la dirección principal de los elementos flexibles dentro del contenedor

row: Los elementos se colocan en la misma dirección que el texto

row-reverse: Los elementos se colocan en la dirección opuesta a **row**

column: Los elementos se colocan de arriba hacia abajo.

column-reverse: Los elementos se colocan de abajo hacia arriba.

justify-content: Alinea los elementos flexibles a lo largo del eje principal del contenedor. Valores:

flex-start: Los elementos se alinean al principio del contenedor flexible en el eje principal (se alinearán en el extremo izquierdo si la dirección principal es horizontal o en la parte superior si la dirección principal es vertical).

flex-end: Los elementos se alinean al final del contenedor flexible en el eje principal (se alinearán en el extremo derecho si la dirección principal es horizontal o en la parte inferior si la dirección principal es vertical).

center: Los elementos se alinean en el centro del contenedor flexible a lo largo del eje principal.

space-between: Los elementos se distribuyen de manera uniforme a lo largo del eje principal del contenedor. El primer elemento se coloca al principio del contenedor y el último elemento al final, y el espacio restante se distribuye equitativamente entre los elementos.

space-around: Similar a **space-between**, pero con espacio adicional distribuido antes del primer elemento y después del último elemento, así como entre los elementos.

space-evenly: Los elementos se distribuyen de manera uniforme a lo largo del eje principal del contenedor, con igual espacio entre los elementos y en los extremos del contenedor

align-items: Alinea los elementos flexibles a lo largo del eje secundario del contenedor. Valores:

stretch: Los elementos se estiran para llenar el contenedor a lo largo del eje secundario. (para que los elementos tengan la misma altura).

flex-start: Los elementos se alinean al principio del contenedor a lo largo del eje secundario.(los elementos se alinearán en la parte superior del contenedor si el eje secundario es vertical, o a la izquierda si el eje secundario es horizontal).

flex-end: Los elementos se alinean al final del contenedor a lo largo del eje secundario. (los elementos se alinearán en la parte inferior del contenedor si el eje secundario es vertical, o a la derecha si el eje secundario es horizontal).

center: Los elementos se alinean en el centro del contenedor a lo largo del eje secundario.

baseline: Los elementos se alinean en la línea de base del contenido. (es la línea imaginaria en la que se apoyan los caracteres de texto)

flex-grow: Especifica la capacidad de un elemento flexible para crecer si es necesario en relación con los otros elementos flexibles dentro del mismo contenedor flexible. Valores:

0: El elemento no crecerá y mantendrá su tamaño original.

1: El elemento crecerá en relación con los otros elementos flexibles dentro del contenedor. Si todos los elementos tienen **flex-grow: 1**, se distribuirá el espacio disponible de manera equitativa entre ellos.

Números mayores que 1: El elemento crecerá más rápido que los otros elementos flexibles con un valor de flex-grow más bajo. Por ejemplo, si un elemento tiene flex-grow: 2 y los otros elementos tienen flex-grow: 1, el primero crecerá el doble que los segundos.

Fracciones decimales: También puedes usar fracciones decimales para especificar el factor de crecimiento. Por ejemplo, 0.5 indicaría que el elemento debería crecer a la mitad que los elementos con flex-grow: 1

flex-shrink: Especifica la capacidad de un elemento flexible para reducirse si es necesario en relación con los otros elementos flexibles dentro del mismo contenedor flexible cuando no hay suficiente espacio disponible. Valores:

0: El elemento no se reducirá y mantendrá su tamaño original incluso si el espacio disponible es insuficiente.

1: El elemento se reducirá en relación con los otros elementos flexibles dentro del contenedor si no hay suficiente espacio disponible. Si todos los elementos tienen flex-shrink: 1, se reducirán en la misma proporción.

Números mayores que 1: El elemento se reducirá más rápido que los otros elementos flexibles con un valor de flex-shrink más bajo.

Fracciones decimales: También se pueden usar fracciones decimales para especificar el factor de reducción.

flex-basis: Especifica el tamaño base de un elemento flexible antes de que se distribuya el espacio disponible o se apliquen los factores de crecimiento y reducción. Valores:

Valor de longitud (por ejemplo, px, em, %): Especifica el tamaño base del elemento flexible en una unidad de longitud específica. Por ejemplo, flex-basis: 200px; establece el tamaño base del elemento en 200 píxeles.

auto: El tamaño base del elemento flexible se calcula automáticamente según su contenido y/o el tamaño de su contenedor.

content: Especifica que el tamaño base del elemento flexible debería ser el tamaño de su contenido. Esto puede ser útil para elementos que contienen texto o elementos con dimensiones intrínsecas conocida

align-self: Se utiliza para controlar la alineación de un único elemento flexible dentro de un contenedor flexible, anulando la alineación predeterminada establecida por align-items en el contenedor. Valores:

auto: Utiliza el valor de alineación predeterminado definido por align-items en el contenedor.

flex-start: El elemento se alinea al principio del contenedor a lo largo del eje secundario.

flex-end: El elemento se alinea al final del contenedor a lo largo del eje secundario.

center: El elemento se alinea en el centro del contenedor a lo largo del eje secundario.

baseline: El elemento se alinea en la línea de base del contenido del contenedor.

stretch: El elemento se estira para llenar el contenedor a lo largo del eje secundario

order: se utiliza para establecer el orden de visualización de los elementos flexibles dentro de su contenedor flexible. Por defecto, los elementos flexibles se muestran en el orden en el que aparecen en el código HTML, pero esta propiedad permite cambiar ese orden. Valores:

0: Este es el valor por defecto y significa que el elemento se mostrará en el orden en que aparece en el código HTML.

1, **2**, **3**, etc.: Estos valores positivos indican que el elemento se mostrará después de los elementos con valores de order menores.

-1, **-2**, **-3**, etc.: Estos valores negativos indican que el elemento se mostrará antes de los elementos con valores de order mayores

Jugamos con flexbox: Flexbox Froggy - Un juego para aprender CSS flexbox

GRID

Se utiliza para para establecer todas las propiedades de las líneas de la cuadrícula en un solo lugar. Es útil cuando deseas definir rápidamente el diseño de tu cuadrícula en CSS. Una vez que un contenedor se define **display:grid;** se pueden usar sus propiedades para definir la cuadricula. Propiedades:

grid-template-rows: Define el tamaño de las filas de la cuadrícula
- **Valores de longitud**: Por ejemplo, 100px, 50%, etc.
- **Valores relativos**: Por ejemplo, 1fr, 2fr, etc. (fracciones del espacio disponible).
- **auto**: Tamaño automático basado en el contenido del elemento.
- **min-content**: Tamaño mínimo basado en el contenido.
- **max-content**: Tamaño máximo basado en el contenido.

grid-template-columns: Define el tamaño de las columnas de la cuadrícula
los mismos valores mencionados anteriormente son aplicables

grid: abreviatura para las dos anteriores **grid: grid-template-rows / grid-template-columns;**

grid: auto / 1fr 2fr; / Una fila automática y dos columnas, la primera con 1 fracción y la segunda con 2 fracciones del espacio disponible */*

grid-template-areas:

Define el diseño de la cuadrícula utilizando áreas nombradas. Son los nombres de las áreas definidas dentro de la cuadrícula.

Ej:

```
display: grid;
grid-template-areas: "header header" "nav main" "footer footer";
```

grid-auto-rows:

Especifica el tamaño predeterminado de las filas que no están explícitamente definidas. Los valores son los mismo que para grid-template-rows

grid-auto-columns:

Especifica el tamaño predeterminado de las columnas que no están explícitamente definidas. Los valores son los mismo que para grid-template-rows

grid-auto-flow:

Controla cómo se colocan los elementos que no tienen un área específica definida. Sus valores:

row: coloca los elementos en nuevas filas automáticamente. Si la cuadrícula tiene un número fijo de columnas y se agregan más elementos que pueden caber en una fila, se colocan en una nueva fila.

column: coloca los elementos en nuevas columnas automáticamente. Si la cuadrícula tiene un número fijo de filas y se agregan más elementos que pueden caber en una columna, se colocan en una nueva columna.

dense: intenta llenar los espacios vacíos dentro de la cuadrícula antes de agregar nuevas filas o columnas. Si un elemento puede caber en un espacio vacío entre otros elementos, se colocará allí en lugar de crear una nueva fila o columna

CSS Grid vs Flexbox - Cuando usar uno u otro

CSS Grid vs Flexbox - Cuando usar uno u otro. (youtube.com)

Jugamos con grid: Grid Garden - Un juego para aprender CSS grid (cssgridgarden.com)

UNIDADES DE MEDIDA

Unidades de medida que se pueden utilizar para especificar longitudes, tamaños y otros valores. Algunas de las unidades más comunes son:

Píxeles (px): se basa en el tamaño de píxel de la pantalla del dispositivo. Es una unidad absoluta y proporciona un tamaño fijo.

Porcentaje (%): se refiere a un porcentaje del tamaño del elemento padre. Por ejemplo, width: 50% significa que el elemento tendrá la mitad del ancho de su elemento contenedor.

Em (em): se refiere al tamaño de la fuente del elemento. Por ejemplo, font-size: 1.5em significa que el tamaño de la fuente será 1.5 veces el tamaño de la fuente del elemento padre.

Rem (rem): Similar a em, pero en lugar de basarse en el tamaño de la fuente del elemento, se basa en el tamaño de la fuente del elemento raíz (html), lo que lo hace más predecible y fácil de manejar.

Viewport Units (vw, vh, vmin, vmax): Estas unidades de medida están relacionadas con el tamaño de la ventana gráfica (viewport) del navegador. Por ejemplo, 1vw es igual al 1% del ancho de la ventana gráfica, mientras que 1vh es igual al 1% de la altura de la ventana gráfica.

Centímetros (cm), milímetros (mm), pulgadas (in): Estas unidades de medida se utilizan para dimensiones físicas y son útiles en casos donde se requiere precisión física, como impresión.

Otras unidades menos comunes son:

Puntos (pt): se utiliza comúnmente en la impresión y es igual a 1/72 de pulgada. Se utiliza a menudo para especificar tamaños de fuente en documentos impresos.

Picas (pc): Similar a los puntos, la pica es una unidad de medida utilizada principalmente en la tipografía y es igual a 12 puntos.

Porcentaje del ancho del contenedor (% width): Además del porcentaje (%), también puedes especificar anchos relativos utilizando % width. Por ejemplo, width: 50% hace que el elemento tenga la mitad del ancho de su contenedor.

Grados (deg): Esta unidad de medida se utiliza para ángulos en propiedades como rotate en transformaciones CSS. Por ejemplo, transform: rotate(45deg) rota un elemento 45 grados en sentido antihorario.

Segundos (s) y milisegundos (ms): Estas unidades de medida se utilizan para especificar duraciones en animaciones CSS y transiciones. Por ejemplo, transition-duration: 0.5s especifica una duración de transición de medio segundo.

Coordenadas (coord): Se utilizan en SVG (gráficos vecotoriales) y tienen valores de longitud que pueden ser números o porcentaje