CERTIFICACIÓN UNIVERSITARIA

UNC | FCEFyN

Desarrollo web

CSS

mundosE

Hojas de Estilos o CSS: ¿Qué son y cómo debemos usarlas?

CSS es el acrónimo de Cascade Style Sheets; en español: Hojas de estilos en cascada. En términos simples, los CSS son archivos que mediante una definición, permiten aplicar ajustes, cambios o mejoras visuales a los archivos HTML.

Los archivos CSS son los responsables de que los archivos HTML puedan tener esa apariencia tan vistosa, que podamos ajustar colores, posiciones y tamaños, de forma detallada.

¿Qué propiedades podemos ajustar?

En CSS podemos definir o ajustar múltiples propiedades de los elementos. Vamos a intentar clasificar estas propiedades en las siguientes categorías:

- **Textos** (para ampliar: https://developer.mozilla.org/es/docs/Web/CSS/CSS_Text)
- **Colores y fondos** (para ampliar: https://developer.mozilla.org/es/docs/Web/CSS/CSS_Backgrounds_and_Borders)
- **Modelo de cajas y posicionamiento** (para ampliar https://developer.mozilla.org/es/docs/Web/CSS/CSS_Box_Model)
- **Diseño (layouts)** (para ampliar: https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout)
- **Grid** (para ampliar: https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Grids)
- **Animaciones** (para ampliar: https://developer.mozilla.org/es/docs/Web/CSS/animation)
- **Misceláneas**

¿Cómo funcionan los estilos?

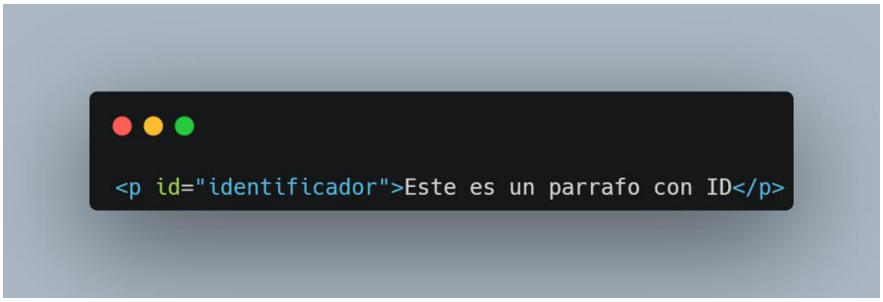
Los archivos CSS se definen como propiedades que son aplicadas a elementos del archivo HTML, por lo tanto, cuando usamos estilos, hablamos de “aplicar estilos” a elementos. Entonces, de alguna forma debemos decir que queremos aplicar tal estilo o propiedad al elemento X. ¿Cómo podemos realizar esta acción?

Para ello, debemos primero seleccionar el elemento, de esta forma, existen múltiples opciones para que podamos obtener o seleccionar el elemento que deseamos ajustar. En ese caso, vamos a usar los selectores, y, a continuación, se explica ¿cómo los utilizaremos?

Seleccionando los elementos

Para que apliquemos estilos o mejoras visuales a cualquier elemento HTML, debemos seleccionar y poder indicarle esos ajustes. Para ello, CSS define múltiples formas de seleccionarlo, pero las principales son:

- **Por ID:** Todo elemento de un archivo HTML puede tener un ID. La realidad es que deberíamos asignarle siempre un ID único. ¿Cómo podemos saber el ID?, cuando verificamos el elemento dentro del archivo HTML, podemos identificarlo de esta forma:

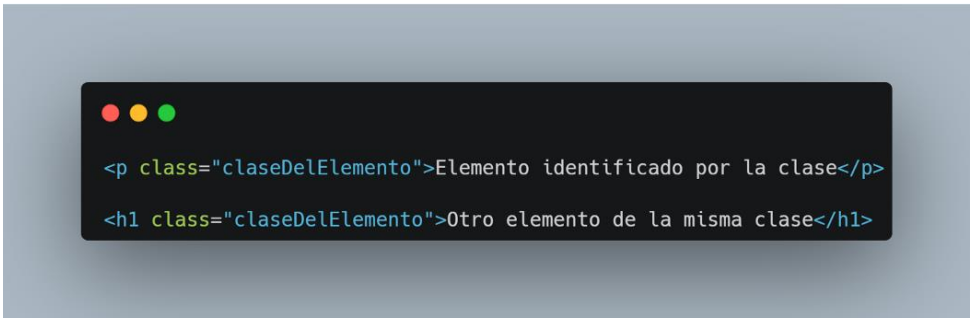


Seleccionando un elemento por su ID.

Luego, en un archivo CSS aplicamos propiedades sobre ese elemento seleccionando por el ID:

Aplicando propiedades CSS a un elemento por su ID.

- **Por clase:** En HTML tenemos un atributo que se denomina class y es posible asignarlo a cualquier elemento. De esta forma, podemos categorizar los elementos y agruparlos, para luego poder aplicar propiedades CSS a todos aquellos que sean de la clase X.



```html
<p class="claseDelElemento">Elemento identificado por la clase</p>

<h1 class="claseDelElemento">Otro elemento de la misma clase</h1>
```

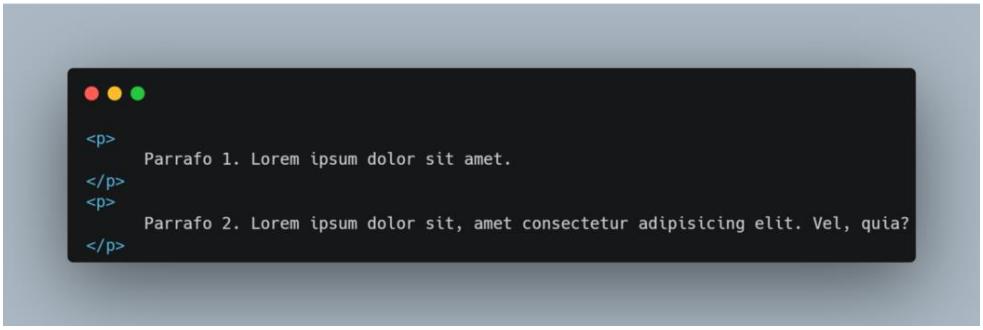
Seleccionando elementos por la clase.

Ahora veamos cómo indicamos las propiedades que se deben aplicar a todos los elementos de esa clase en un archivo CSS:

```css
.claseDelElemento {
  /* Este estilo será aplicado a todos los elementos de esta clase */
  font-size: 1em;
  color: #ff0000;
  margin-left: 1em;
  font-family: Arial;
}
```

Aplicando propiedades CSS a elementos por su clase.

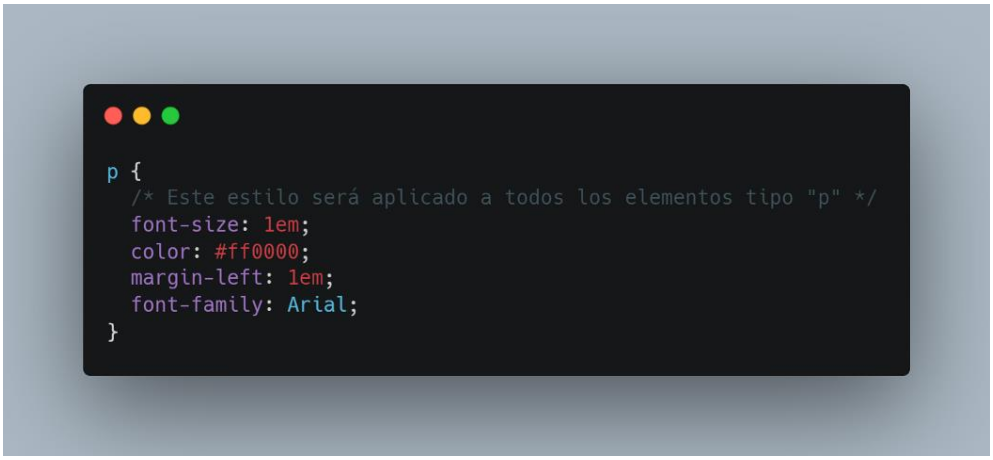
- **Por etiqueta:** En este caso, seleccionamos todos los elementos de un tipo, por ejemplo, todos los párrafos.



```html
<p>
    Parrafo 1. Lorem ipsum dolor sit amet.
</p>
<p>
    Parrafo 2. Lorem ipsum dolor sit, amet consectetur adipisicing elit. Vel, quia?
</p>
```

Elementos de una misma etiqueta.

Allí aplicamos el estilo haciendo referencia al nombre de la etiqueta:



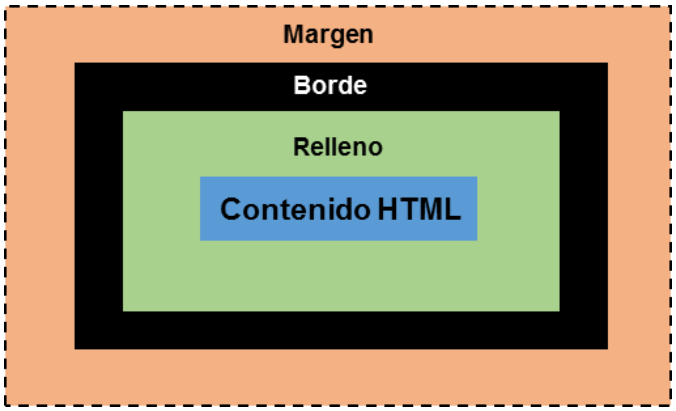
```
p {
  /* Este estilo será aplicado a todos los elementos tipo "p" */
  font-size: 1em;
  color: #ff0000;
  margin-left: 1em;
  font-family: Arial;
}
```

Aplicación del estilo a una etiqueta.

Posicionamiento — El modelo de cajas

En CSS, cuando hablamos de posicionamiento, nos referimos al modelo de caja, donde cada elemento de la página se comporta como una caja en la que se deben identificar los siguientes aspectos:

- **El contenido del elemento:** Se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.).
- **El borde del elemento:** Línea que encierra completamente el contenido y su relleno.
- **El espacio entre el contenido y el borde (el relleno):** Espacio libre opcional existente entre el contenido y el borde.
- **El espacio entre el elemento y otros elementos (el margen):** Separación opcional existente entre la caja y el resto de cajas adyacentes.



Modelo de caja.

Luego, para ajustar la posición de caja tenemos las siguientes herramientas disponibles:

Propiedades display y float

En CSS existen 2 propiedades que nos permiten ajustar la posición de los elementos. Una de estas propiedades es el display, que define la forma de presentación (o no) del elemento, y puede tener alguno de los siguientes valores:

- **none:** El elemento no es presentado en pantalla, es decir, no existe y no utiliza ningún espacio (ojo no confundir con ocultar, donde el espacio sigue ocupado, pero no se muestra nada).
- **block:** El elemento se comporta como un bloque, es decir que toma el espacio total disponible.
- **inline:** El elemento se inserta de forma tal que sólo ocupa el espacio necesario, alineándose con el resto de elementos de la línea.
- **inline-block:** Es una mezcla, el elemento ocupa sólo su espacio, pero respetando las propiedades de margen, ancho y alto.

Existen otros valores que vamos a comentar más adelante, pero para posicionamiento, usando estos valores se pueden realizar operaciones de alto impacto en la presentación.

Por su parte, la propiedad float, permite hacer que el elemento flote sobre el resto, de tal forma que, si es indicado un valor, el resto de los elementos se ajustan sobre los márgenes. Los posibles valores que puede tomar son:

- **none:** No es un elemento flotante.
- **left:** El elemento será alineado a la izquierda y el resto de elementos va a alinearse a su margen derecha.
- **right:** El elemento será alineado a la derecha y el resto de elementos va a alinearse a su margen izquierda.

Propiedades avanzadas para diseño en CSS

CSS Grid y flexbox son herramientas CSS creadas para facilitar el posicionamiento y manipulación de elementos en HTML.

La recomendación de uso de CSS Grid es aplicarla cuando vayamos a utilizar elementos en dos dimensiones (ej.: filas y columnas), como en el diseño de una página.

Flexbox, por otro lado, es más recomendable para ser utilizado cuando vamos a trabajar en una sola dimensión, ya sea vertical u horizontal.

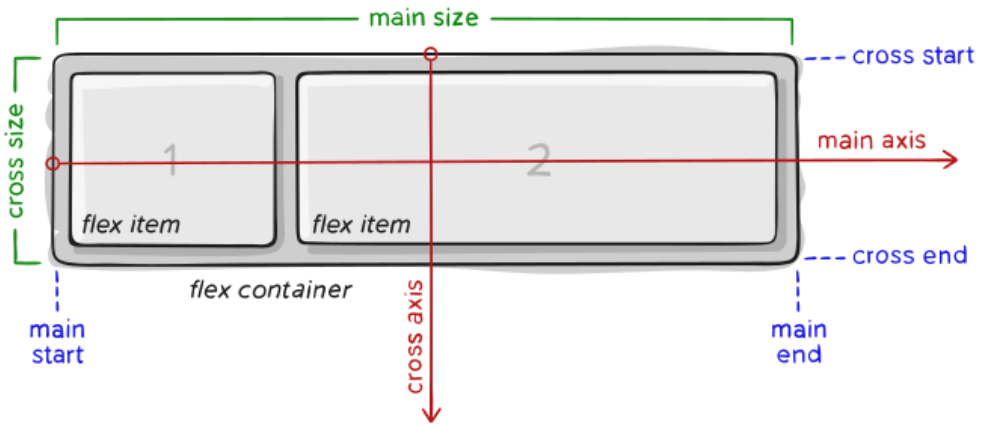
Flexbox

Es un sistema de elementos flexibles donde los elementos HTML se adaptan y colocan automáticamente, siendo mucho más fácil personalizar los diseños. Está pensado para diseños en una sola dimensión, es decir, filas o columnas.

Flexbox no es una propiedad, en realidad es un módulo completo, por lo tanto, aplicar flexbox es usar un conjunto de propiedades CSS que nos permitirán de forma simple gestionar layouts.

Algunos de ellos están destinados a establecerse en el contenedor (elemento principal, conocido como "contenedor flexible"), mientras que otros están destinados a establecerse en los elementos secundarios (llamados "elementos flexibles").

Si el diseño o layout "regular" se basa en direcciones de flujo en bloque y en línea, el diseño flexible se basa en "direcciones de flujo flexible".



Flexbox.

El tamaño a lo largo del eje principal de flexbox se denomina `main axis`, la dirección perpendicular es el `cross axis`. Los elementos se distribuirán siguiendo el eje principal (desde el `main start` hasta el `main end`) o el eje transversal (desde el `cross start` hasta el `cross end`).

- Eje principal o `main axis`: el eje principal de un contenedor flexible es el eje principal a lo largo del cual se disponen los elementos flexibles. Cuidado, no es necesariamente horizontal; depende de la propiedad `flex-direction`.
    - `main start` | `main end`: los elementos flexibles se colocan dentro del contenedor comenzando desde el inicio principal y yendo al extremo principal.

- Tamaño principal o `main size`: el ancho o la altura de un elemento flexible, lo 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.

- Eje transversal: es el eje perpendicular al eje principal. Su dirección depende de la dirección del eje principal.

- `cross start` | `cross end`: las líneas flexibles se llenan con artículos y se colocan en el contenedor comenzando en el lado de inicio cruzado ocross `start` del contenedor flexible y yendo hacia el lado del extremo cruzado o `cross end`.

- Tamaño cruzado o `cross size`: el ancho o la altura de un elemento flexible, lo que esté en la dimensión cruzada, es el tamaño cruzado del elemento. La propiedad de tamaño cruzado es cualquiera de 'ancho' o 'alto' que esté en la dimensión cruzada.

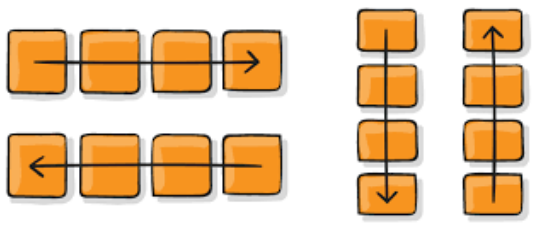
Para implementar flexbox, debemos por comenzar usando la propiedad `display` con valor `flex.`

Luego de ello podemos usar las siguientes propiedades:

flex-direction:

Definición del eje principal del contenedor.

Posibles valores: row, row-reverse, column, colum-reverse.

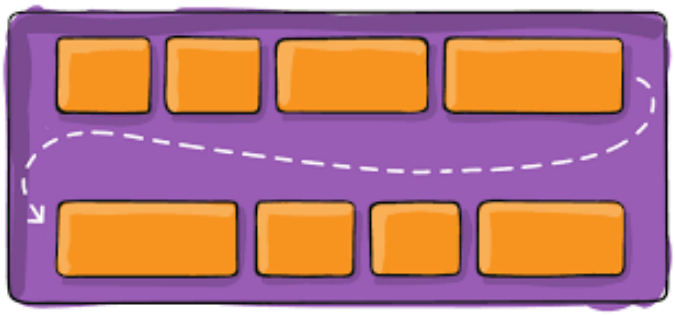


Representación gráfica de la propiedad. (Imagen tomada de css-tricks).

flex-wrap:

Define si los elementos deben agruparse en una línea o no.

Posibles valores: wrap, nowrap, wrap-reverse.

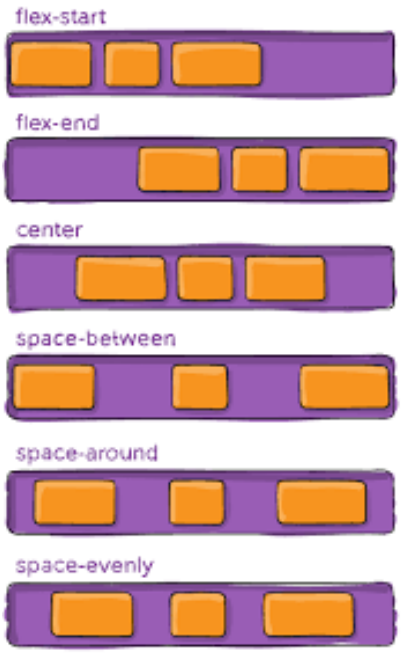


Representación gráfica de la propiedad. (Imagen tomada de css-tricks).

justify-content:

Define cómo se deben alinear los elementos flexibles a lo largo del eje principal.

Posibles valores: flex-start, flex-end, center, space-between, space-evenly, space-around.

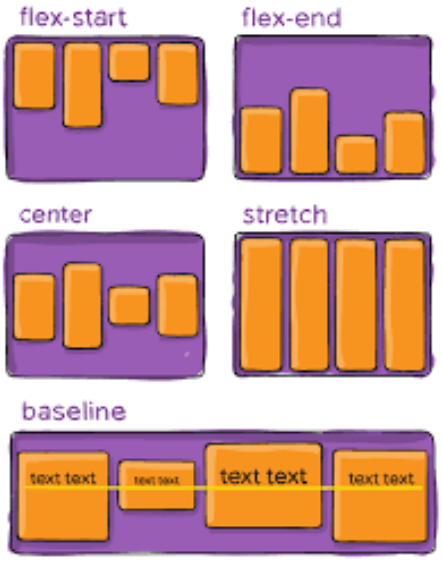


Representación gráfica de la propiedad. (Imagen tomada de css-tricks).

align-items:

Define la alineación de los elementos a lo largo del eje perpendicular o cruzado.

Los valores posibles son: flex-start, flex-end, center, stretch, baseline.

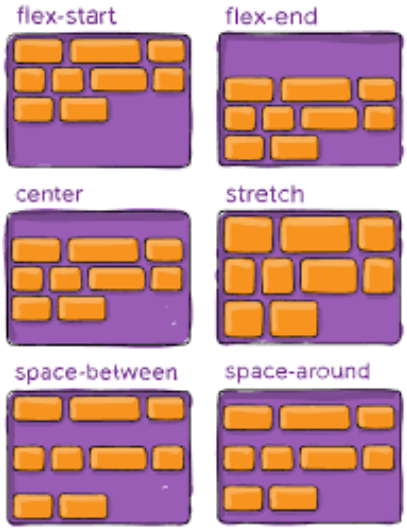


Representación gráfica de la propiedad. (Imagen tomada de css-tricks).

align-content:

Permite alinear los elementos en el eje perpendicular, al igual que lo hace align-items, siempre y cuando estos sean multilínea, para elementos flexibles de una sola línea, esta propiedad no realizará ningún ajuste.

Posibles valores: flex-start, flex-end, center, stretch, space-between, space-around.



Representación gráfica de la propiedad. (Imagen tomada de css-tricks).

Grid CSS

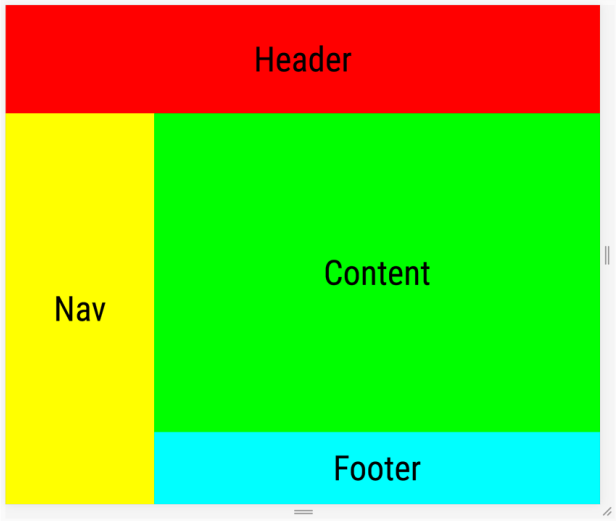
CSS Grid Layout (también conocido como “Grid” o “CSS Grid”), es un sistema de diseño bidimensional basado en cuadrículas que rompe con la forma tradicional de diseño web previamente conocida.

Con CSS siempre hemos diseñado la presentación en la web. Primero, usamos tablas, luego flotadores, posicionamiento y bloques en línea, pero todos estos métodos eran esencialmente trucos y dejaban fuera muchas funciones importantes (centrado vertical, por ejemplo). Luego Flexbox llegó y ha sido una excelente herramienta de diseño, pero con flujo unidireccional que en algunos casos puede que no sea lo suficiente simple para implementar. Posteriormente ha llegado Grid que es el primer módulo CSS creado específicamente para resolver los problemas de diseño que todos hemos estado solucionando desde que creamos sitios web.

¿Cómo funciona Grid CSS?

El conjunto de propiedades que componen el módulo de Grid CSS nos va a permitir dividir y manipular la pantalla en cajas, de forma tal, que nosotros podemos definir: filas, columnas y áreas. Ojo con este último término, las filas y columnas son tratadas con cierta similitud de lo que hacemos en flexbox, pero las áreas son un concepto nuevo que es extremadamente poderoso.

Revisa el siguiente layout y piensa cómo hacerlo con lo que conocemos hasta ahora:



Seguramente pensaste en posicionamiento, flexbox o hasta tablas (es broma), pero si analizamos las propiedades de Grid, este layout se hace sólo definiendo los div y algunas propiedades CSS. Comencemos a revisarlas:

Propiedad display y GridCSS

Para convertir un contenedor en un Grid, necesitamos definir display:grid , de esta forma le decimos al navegador: convierte este contenedor en una grilla.

¿Cómo definimos las filas y las columnas?

Usando las siguientes propiedades: `grid-template-columns` y grid-template-rows. Veamos un ejemplo de ello:

De esta forma definimos que nuestro contenedor va a tener 3 filas y 2 columnas.

Con lo anterior, finalizamos una primera aproximación, pero la realidad es que aún no es suficiente. En ese caso, aún debemos avanzar en el siguiente concepto: *las áreas*. Y es que si analizamos el layout planteado, lo que queremos es tener áreas: el header, el nav, el content y el footer son áreas, como las definimos. Grid CSS tiene las propiedades necesarias para ello, en ese caso vamos a usar: `grid-area` para nombrarlas y podemos luego usarlas y vamos a usar `grid-template-areas` para poder definir cómo las vamos a representar.

Con los siguientes cambios podemos definir ese layout (verifica las nuevas propiedades):

```css
.header {
  grid-area: header;
}
.nav {
  grid-area: nav;
}
.content {
  grid-area: content;
}
.footer {
  grid-area: footer;
}

.grid-container {
  grid-template-columns: 20% 80%;
  grid-template-rows: 10% 80% 10%;
  grid-template-areas:
    'header header'
    'nav content'
    'footer footer'
}
```

En Grid además, tenemos otra serie de propiedades que nos permiten ajustar aún más el layout, ejemplo:

- `row-gap, column-gap, gap`: Estas propiedades nos permiten definir el espacio que debe existir entre filas, columnas o en ambas.
- `justify-items`: Nos permite definir la alineación de los elementos del grid a lo largo del eje horizontal.
- `align-items`: Nos permite definir la alineación de los elementos del grid a lo largo del eje vertical.
- `justify-content`: Sigue el mismo comportamiento de flexbox, en este caso alínea el grid en el eje horizontal, siempre que el grid no esté siendo ocupado totalmente (al igual que como sucede en flex con el contenedor flexible).
- `align-content`: Sigue el mismo comportamiento de flexbox, en este caso, alínea el grid en el eje vertical, siempre que el grid no esté siendo ocupado totalmente (al igual como sucede en flex con el contenedor flexible).

Además de estas propiedades, existen otras más que complementan el funcionamiento, les invito a que revisen la documentación de la MDN y puedan aprovechar al máximo está potente característica que CSS3 nos ha puesto a disposición.

Bibliografía

- MDN Web Docs. (13 de agosto de 2022). *CSS Background and Borders*. https://developer.mozilla.org/es/docs/Web/CSS/CSS_Backgrounds_and_Borders
- MDN Web Docs. (13 de agosto de 2022). *Modelo de Caja de CSS*. https://developer.mozilla.org/es/docs/Web/CSS/CSS_Box_Model
- MDN Web Docs. (13 de agosto de 2022). *Texto CSS*. https://developer.mozilla.org/es/docs/Web/CSS/CSS_Text
- MDN Web Docs. (17 de agosto de 2022). *Cuadrículas*. https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout/Grids
- MDN Web Docs. (19 de septiembre de 2022). *Animation*. https://developer.mozilla.org/es/docs/Web/CSS/animation
- MDN Web Docs. (27 de septiembre de 2022). *Diseño CSS*. https://developer.mozilla.org/es/docs/Learn/CSS/CSS_layout