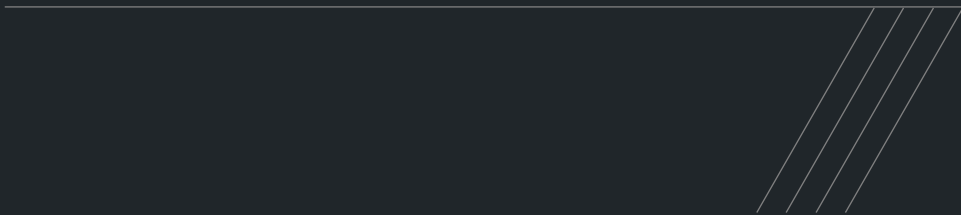


INNOVARE SUITE

WEB DEVELOPMENT

PARTE 2



HOJAS DE ESTILO EN CASCADA – CSS (Cascading Style Sheets)

Es un lenguaje creado para controlar el aspecto o presentación de los elementos de HTML. Como HTML, CSS no es realmente un lenguaje de programación, tampoco es un lenguaje de marcado, es un lenguaje de **hojas de estilo**, es decir, permite aplicar estilos de manera selectiva a elementos en documentos HTML.

Un archivo de tipo CSS no es más que un archivo simple de texto cuya extensión es **.css**. Se pueden crear los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.

Enlazando un archivo CSS

CSS externo: Para enlazar un archivo .css se utiliza la etiqueta <link> de la siguiente manera:

```
<link rel="stylesheet" type="text/css" href="/css/estilos.css"/>
```

rel: indica el tipo de relación que existe entre el recurso enlazado (el archivo CSS) y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet.

type: indica el tipo de recurso enlazado. Para los archivos CSS su valor siempre es text/css.

href: indica la URL del archivo CSS que contiene los estilos.

Esta etiqueta se debe colocar en head de nuestro html.

CSS en línea: dentro del atributo **style=""** incorporamos los estilos que se van a aplicar solo en esa misma etiqueta. Opción no recomendable.

```
<p style="color: White; background-color: violet">Esto es un párrafo</p>
```

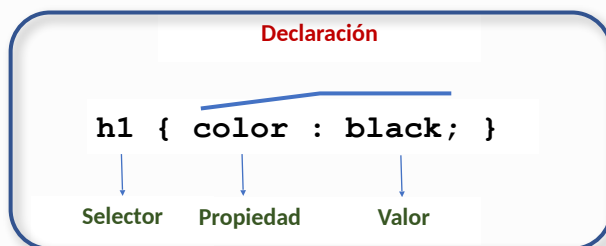
salida  Esto es un párrafo

CSS interno: Se incluye la etiqueta <style> dentro del <head> en el documento. Opción menos recomendable:

```
<style>
  h1 {
    color: white;
    background-color: red;
  }
</style>
```

En el ejemplo anterior todas las etiquetas **<h1>** tendrán color de fuente blanco y fondo de color rojo.

Anatomía de una regla CSS



La estructura completa es llamada **regla predeterminada** (o **regla**). Está compuesta por:

Selector: indica el elemento o elementos HTML a los que se aplica la regla CSS.

Declaración: especifica los estilos que se aplican a los elementos.

Propiedad: característica que se modifica en el elemento seleccionado, como por ejemplo su tamaño de letra, su color, etc.

Ejemplo:

```
p {
  color: red;
}
```

[?] Esto aplica al elemento párrafo del documento HTML el color rojo

TIPOS DE SELECTORES

Selector	Descripción	Sintaxis
Universal	Selecciona todos los elementos del documento	<code>* { propiedad: valor; }</code>
Etiqueta o tipo	El elemento a aplicar	<code>etiqueta { propiedad: valor; }</code>
Clase	Refiere a una clase	<code>.clase { propiedad: valor; }</code>
Identificación	Utiliza el atributo ID para seleccionar un elemento	<code>#id { propiedad: valor; }</code>
Combinados	Permite dar un estilo a todos los selectores indicados	<code>selector1, selector2, selector3 { propiedad: valor; }</code>

Hijos	Selecciona un elemento que es hijo de otro elemento y se indica mediante el signo >	selector1 > selector2 { propiedad: valor; }
Adyacente	Selecciona elementos que están seguidos en el código HTML. Se indica con el +	selector1 + selector2 { propiedad: valor; }
Atributos	<p>Permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos:</p> <p>[attr]: Seleccionará todos los elementos que contengan el atributo attr, sin importar el valor que tenga</p> <p>[attr=val]: Seleccionará los elementos con el atributo attr, pero solo aquellos cuyo valor coincida con val</p> <p>[attr^=val]: Seleccionará todos los elementos cuyo atributo attr comienza por el valor val</p> <p>[attr\$=val]: Elegirá todos los elementos cuyo atributo attr termina por el valor val</p>	<p>a[class] { color: blue; } <i>/*Se muestran de color azul todos los enlaces que tengan un atributo "class", independientemente de su valor*/</i></p> <p>a[href="http://www.ejemplo.com"] { color: blue; } <i>/*Se muestran de color azul todos los enlaces que apunten al sitio referenciado*/</i></p> <p>a[href^="mailto:"] { ... } <i>/*Selecciona todos los enlaces que empiezan con la palabra "mailto"*/</i></p> <p>a[href\$=".html"] { ... } <i>/*Selecciona todos los enlaces que terminan en .html*/</i></p>
Descendente	Selecciona los elementos que se encuentran dentro de otros elementos. Se forma por dos o más selectores separados por espacios.	<p>selector1 selector2... selectorN { propiedad: valor; }</p> <p><i>/*Siendo el selector N el elemento sobre el que se aplica el estilo, los anteriores indican el lugar en el que se encuentra el elemento*/</i></p>

Pseudo-clases

Consta de una clave precedida de : que se añaden al final del selector para indicar que daremos estilo a los elementos seleccionados solo cuando estos se encuentren en un estado determinado.

li:first-child { ... }

//Selecciona el primero elemento de una lista

li:last-child { ... }

//Selecciona el último elemento de una lista

li:nth-child(2n+1) { ... }

//Selecciona todos los elementos impares

li:nth-child(2n) { ... }

//Selecciona todos los elementos pares

a:hover { ... }

//Se activa cuando el Usuario pasa el mouse por encima de un elemento

a:active { ... }

//Se activa cuando el usuario activa un elemento, por ejemplo, cuando hace clic sobre un elemento

a:focus { ... }

//Se activa cuando el elemento tiene el foco del navegador, es decir, cuando está seleccionado

a:visited { ... }

//Selecciona cualquier <a> que ha sido visitado

Pseudo-elementos

Son parecidos a las pseudo-clases, con la diferencia de que estos están precedidos por :: que se añaden al final del selector para elegir cierta parte de un elemento

h1::before { content: "Capítulo –"; }

//añade contenido antes del elemento

p::after { content: "."; }

//añade contenido después del elemento

::first-letter { ... }


//para darle estilos a la primera letra

::first-line { ... }

//para darle estilos a la primera línea de texto

Herencia

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

body { color: blue; }  Todos los elementos de la página tendrían letra de color azul.

Cascada

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

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

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

Especificidad

Consiste en un cálculo que hace el navegador para establecer su nivel de importancia. Lo que hace es otorgar un valor de puntos a los diferentes tipos de selectores y la suma de estos establece la importancia de ese selector en particular.

Por ejemplo:

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

Modelo de cajas

En CSS todo elemento está enmarcado en una caja.

Hay dos tipos de cajas: cajas de bloque y cajas en línea. Estas características se refieren al modo como se comporta la caja con otras.

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

Las cajas **en línea** sólo ocupan el espacio necesario para mostrar su contenido. Los párrafos por ejemplo son elementos de bloque y los links son de línea.

Tipo de visualizaciones: interna y externa

Las cajas tienen un tipo de visualización *externa*, que define si se trata de una caja en bloque o en línea.

Sin embargo, las cajas también tienen un tipo de visualización *interna*, que determina cómo se disponen los elementos dentro de esa caja.

Podemos cambiar el tipo de visualización interna utilizando valores de **display**, como **flex**. Si en un elemento establecemos **display: flex;**, el tipo de visualización externa es de tipo **block**, pero el tipo de visualización interna cambia a **flex**.

Hablemos un poco de Display:

Esta propiedad permite que formateemos texto de una manera determinada, más allá de cómo estos estén estructurados. La ventaja de ello es que podemos controlar la forma en la que un texto o párrafo se visualizará más allá de cómo esté declarada, desde su origen, la forma de mostrar dicho texto.

Ej.

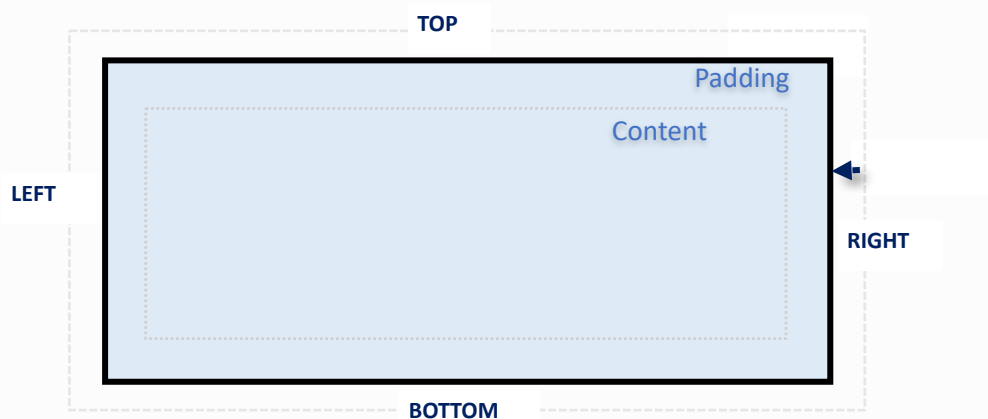
```
Elemento {  
  display: inline;  
}
```

En el desarrollo de webs dinámicas, donde los textos provienen de una base de datos o de un archivo externo, el texto heredado muchas veces contiene un formato HTML incrustado, que limita la forma de mostrar dicho texto.

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

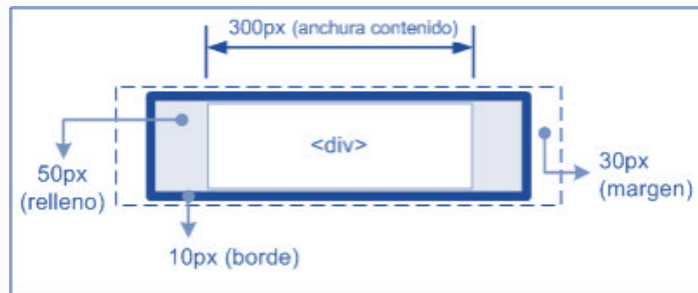
- **block**: hace que el comportamiento del elemento sea como un bloque.
- **inline**: los elementos se muestran en la misma línea, ajustándose al contenido y sin tener en cuenta el ancho, alto o márgenes de la caja.
- **inline-block**: los elementos se muestran en la misma línea respetando el ancho,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

Una caja está formada de la siguiente manera:



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


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



Ancho total de la caja: $30 + 10 + 50 + 300 + 50 + 10 + 30 = 480 \text{ px}$

Si deseamos activar el modelo de cajas alternativo para un elemento, podemos utilizar la propiedad **box-sizing** que acepta dos valores:

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

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

Propiedades margin y padding

La propiedad **margin** es utilizada para crear espacio alrededor de los diferentes elementos que integran una página. A través de **margin**, podemos especificar el tamaño del espacio en blanco en todo el borde que rodea un elemento (div, p, h, etc.). Lo podemos configurar por medio de **margin-top**, **margin-right**, **margin-bottom** y **margin-left**.

Se pueden aplicar los siguientes valores:

- **auto:** la página calcula el margen de forma automática.
- **length:** número que especifica el valor, con las unidades px, pt o cm.
- **%:** porcentaje determinado que permite especificar el ancho del elemento contenedor.
- **inherit:** el margen es especificado de acuerdo con la herencia del elemento superior.

También disponemos de la opción abreviada, que nos permite en una sola línea especificar dos o cuatro márgenes.

Ej. **margin: 20px, 40px, 60px, 80px;**

Padding

Es utilizada para generar espacio alrededor del contenido de un elemento. Al igual que **margin**, **padding** permite establecer valores mediante **auto**, **length**, **%** e **inherit**, de manera individual sobre cada lateral, o de manera conjunta utilizando en una única línea hasta cuatro parámetros.

Ej. **padding: 10px, 10px, 10px, 5px;**

Propiedad position:

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

Los posibles valores que puede adoptar son:

- **static:** valor por defecto, no tendrá en cuenta los valores para las propiedades top, left, right y bottom.
- **relative:** se comporta de la misma manera que static a menos que agreguemos las propiedades top, left, right y bottom. Esto causará que su posición normal se reajuste, pudiendo causar solapamiento entre los distintos elementos.

- **absolute:** se hace en base a su elemento contenedor, por lo general el body, o un elemento posicionado relativamente que lo contenga.
- **fixed:** sirve para posicionar un elemento como si tuviera posicionamiento absoluto, pero su posición final será siempre fija, es decir, que, aunque se desplace el documento con las barras de desplazamiento siempre aparecerá en la misma posición.
- **sticky:** su comportamiento va a ser relativo hasta que al hacer scroll llegamos al elemento con position sticky, y en ese momento pasa a tener un comportamiento fijo.

Caja flexible (flexbox):

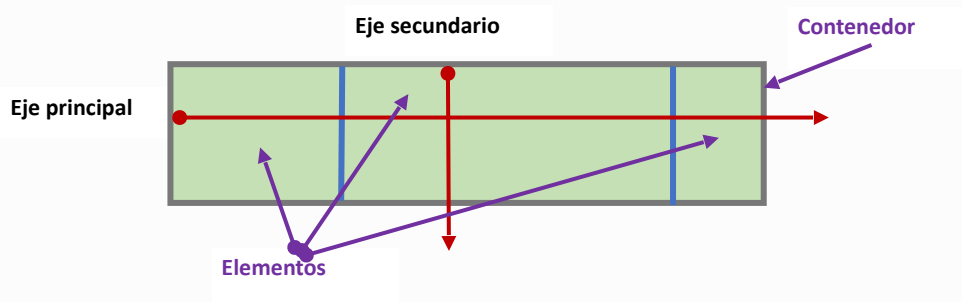
Es un método de diseño unidimensional para organizar elementos en filas y columnas. Los elementos se expanden para llenar espacio adicional o se encogen para caber en espacios más pequeños.

En la práctica, agrega un nuevo tipo de "display CSS", con una completa gama de nuevas propiedades aplicables a ese tipo de display, a partir de los que podemos conseguir cosas extraordinarias. Así como tenemos el display "block", "inline", "inline-block", etc., también disponemos de "flex" e "inline-flex", aceptando una gran cantidad de nuevas propiedades de gran utilidad.

Se pueden diferenciar dos elementos principales: la **caja contenedora** y los **elementos que situamos dentro**. El contenedor va a poder modificar las dimensiones y el orden de los ítems, para acomodarlos de distintas maneras, para distribuirlos a nuestro gusto, como hacer que los ítems se estiren para ocupar todo el contenedor, o se encojan para que quepan en él sin desbordar, distribuidos en filas o en columnas, etc.

Los ejes en flexbox:

Existen dos ejes, el **principal**, por defecto, es el eje horizontal y el eje **secundario** que es el eje vertical. Estos ejes implican el modo en el que los elementos se van a posicionar.



Propiedades flexbox:

Display: flex: al contenedor principal en un esquema flexbox es al que asignamos esta propiedad que hace que cambien las reglas con las cuales sus hijos van a ser representados en la página.

Ej.:

```
.contendor-flex {
  display: flex;
}
```

A partir de este momento, todos los elementos en la página con la clase “contenedor-flex” se comportarán según las reglas de **flexbox**. Sólo sus hijos quedan afectados, no el propio contenedor.

Display: inline-flex: es como un elemento inline-block, donde sus hijos se comportan con las reglas del **flexbox**.

Ej.

```
.contenedor-flex {  
    display: inline-flex;  
}
```

Una vez que el contenedor es **flex** o **inline-flex** podemos aplicarle toda una serie de propiedades adicionales para personalizar todavía más su comportamiento. Éstas son:

- **flex-direction:** esta propiedad sirve para definir la dirección del flujo de colocación de los elementos. Está relacionado con los ejes que vimos más arriba, pudiendo marcar si los elementos se van a colocar todos en la misma fila, o si se van a colocar en una columna, también permite indicar el orden de los ítems, normal o reverso. Los valores relacionados son:
 - **row:** es el valor por defecto, indica que los elementos se colocan en una fila, uno al lado del otro, de izquierda a derecha.
 - **row-reverse:** se colocan en una fila, pero con orden de derecha a izquierda.
 - **column:** se colocan uno debajo del otro, en orden los primeros arriba.
 - **column-reverse:** se colocan en una columna, pero los primeros aparecerán abajo.
- **flex-wrap:** sirve para indicar si queremos que haya saltos de línea en los elementos que se colocan en el contenedor, si es que éstos no caben en el espacio disponible. De manera predeterminada con **flexbox** los elementos se colocan en el eje de horizontal, en una fila. Si los elementos no caben en el contenedor, el comportamiento flex hará que se agrupen en la fila de manera que entren sin saltar de línea, pero también se puede configurar para que en el caso de que no quepan, se pasen a la línea siguiente. Sus valores son:
 - **nowrap:** es el valor predeterminado, hace que se produzcan saltos de línea.
 - **wrap:** si no caben, hace que se coloquen en la línea siguiente.
 - **wrap-reverse:** el salto de línea se producirá al contrario, o sea, hacia arriba.
- **flex-flow:** es sólo un atajo para escribir de una sola vez **flex-direction** y **flex-wrap**. El valor predeterminado es “row nowrap”.
- **justify-content:** es una propiedad muy útil para indicar cómo se van a colocar los justificados y márgenes de los ítems. Podemos indicar que justifique al inicio del eje o al final del eje, o, que a la hora de distribuirse se coloque un espacio entre ellos o un espacio entre ellos y los bordes. Sus posibles valores son:
 - **flex-start:** añade los elementos a partir del inicio del eje principal.
 - **flex-end:** añade los elementos a partir del final del eje principal.
 - **center:** los elementos se centran en el espacio del contenedor, siempre con respecto al eje principal.

- **space-between:** hace que los elementos se distribuyan con un espacio proporcional entre ellos, situando a los ítems de los extremos en el borde del contenedor.
- **space-around:** es parecido a space-between en el sentido de dejar un espaciado proporcional, sin embargo, en esta ocasión se deja también espacio entre el borde del contenedor y los ítems de los extremos.
- **space-evenly:** distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha.
- **align-items:** es similar a la propiedad anterior, sólo que ésta se alinea con respecto al eje secundario y no el principal. En el caso de un contenedor flex cuyo eje principal está en la horizontal, align-items servirá para obtener el alineamiento en el otro eje (vertical, de arriba abajo). Sus valores posibles son:
 - **flex-start:** indica que se posicionarán al comienzo del eje secundario.
 - **flex-end:** se posicionarán al final del eje secundario.
 - **center:** se posicionarán en el centro del eje secundario.
 - **stretch:** ocuparán el tamaño total del eje secundario.
 - **baseline:** para el posicionamiento de los elementos se tendrá en cuenta el texto que hay escrito dentro.
- **align-self:** actúa exactamente igual que align-items, sin embargo se utiliza sobre un ítem hijo específico y no sobre el elemento contenedor. Los valores que puede tomar son los mismos que align-items:
 - **flex-start:** alinea los ítems al principio del eje secundario.
 - **flex-end:** alinea los ítems al final del eje secundario.
 - **center:** alinea los ítems al centro del eje secundario.
 - **stretch:** alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
 - **baseline:** alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor.
 - **auto:** hereda el valor de align-items del padre (si no se ha definido, es stretch).
- **align-content:** esta propiedad sólo aplica cuando disponemos de varias líneas de elementos en el contenedor flexbox. El efecto conseguido será una alineación y separación de las filas en el eje secundario. Al final se conseguirá un efecto parecido al que conseguimos con align-items, en el sentido que aplicará al eje secundario su efecto de distribución, solo que aquí no estamos indicando la colocación de una única fila, sino de todas las filas. Sus valores son:
 - **flex-start:** indica que las filas se colocarán todas pegadas entre sí, desde el inicio del eje secundario.
 - **flex-end:** las filas se colocarán pegadas entre sí, pero esta vez hacia el final del eje secundario.
 - **center:** se posicionarán en el centro del eje secundario, pegadas entre sí.
 - **stretch:** sus dimensiones crecerán para ocupar todo el espacio disponible.
 - **space-between:** indica que las filas se separarán entre sí, dejando entre ellas un espacio proporcional.

- **space-around:** indica que las filas se separarán, dejando un espacio proporcional entre ellas, también en el borde.

Propiedades de los elementos o ítems del contenedor flexbox:

- **flex-grow:** sirve para decir cómo deben crecer los elementos incluidos en el contenedor, es decir, cómo distribuir el espacio entre ellos, haciendo que ocupen más o menos espacio. El valor que acepta es numérico e indica la proporción de espacio que va a ocupar.
- **order:** Admite un valor numérico entero y sirve para aplicar un orden puramente arbitrario en la disposición de los elementos.
- **flex-shrink:** sirve para indicar que ciertos ítems deben encoger su tamaño. El valor predeterminado es de 1. Cualquier valor superior indica que ese elemento se encogerá con respecto a lo que ocuparía si no tuviera esa propiedad. A mayor valor, más reducido será el tamaño resultante del elemento.
- **flex-basis:** sirve para modificar las dimensiones de los elementos atendiendo a varias posibilidades. Los valores que utiliza son:
 - **Numérico, unidad CSS o porcentaje:** lo que indica las dimensiones iniciales del elemento, antes de otorgar espacio sobrante.
 - **auto:** es el valor predeterminado e indica que flex-basis no va a tener efecto, otorgando dimensionamiento en función de cualquier otro atributo que pueda haber en el elemento, o en función del contenido del propio elemento.
- **flex:** es sólo un atajo para escribir en una sola línea de código CSS las propiedades flex-grow, flex-shrink y flex-basis. El valor por defecto es **"0 1 auto"**.

[Más info y ejemplos](#)

Cuadrículas

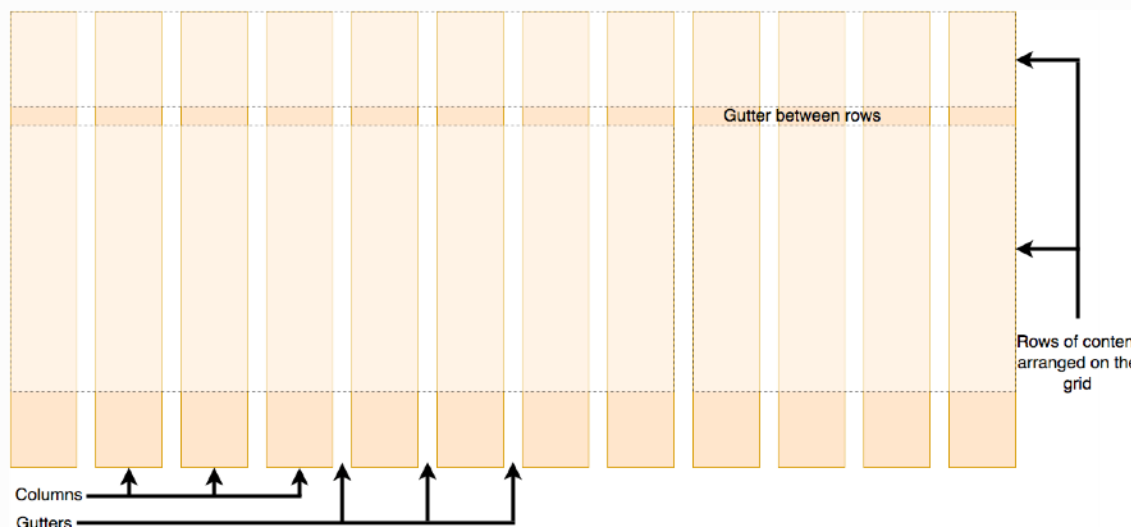
CSS Grid Layout es un sistema de diseño bidimensional para la web. Es un estándar de CSS que nos permite disponer elementos en la página por medio de una rejilla. Gracias a CSS Grid Layout podemos conseguir cualquier distribución de los elementos en filas y columnas totalmente configurables.

Básicamente nos permite conseguir todo tipo de distribuciones de los contenidos, con una rejilla adaptable a cualquier situación y tamaños de pantalla.

Propiedades principales:

- **Atributo display: grid:** el primer paso para trabajar con el sistema de rejilla es incorporar el atributo **"display: grid"** en un contenedor. Desde el momento en que colocamos ese tipo de display en un contenedor, todos sus hijos directos pasan a ser considerados celdas dentro del sistema de rejilla.
- **inline-grid:** aunque no se use tanto, tenemos la posibilidad de usar otra variante de declaración del contenedor grid, con el valor **"inline-grid"**, que hace que la rejilla como un todo se comporte como un elemento inline.

Una cuadrícula normalmente tendrá **columnas**, **filas** y luego espacios entre cada fila y columna denominados **canaletas (gutters)** o **calles**.



A diferencia de flexbox, los elementos no se verán inmediatamente diferentes. La declaración **display: grid** brinda una cuadrícula de una columna, por lo que sus elementos continuarán mostrándose uno debajo del otro como lo hacen en el flujo normal.

Para ver algo que se parece a una cuadrícula, debemos agregar algunas columnas. Por ejemplo, podemos agregar tres columnas de 200 píxeles. Se puede utilizar cualquier unidad de longitud o porcentaje:

```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

Además de longitudes y porcentajes, podemos usar **fr**. Esta unidad representa una fracción del espacio disponible en el contenedor de cuadrícula para dimensionar filas y columnas de cuadrícula de manera flexible. Por ejemplo, podemos cambiar la mitad utilizada en el ítem anterior por **fr**:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

Ahora tenemos pistas flexibles. La unidad **fr** distribuye el espacio proporcionalmente. Podemos especificar valores positivos para cada pista. Se puede mezclar unidades **fr** con unidades de longitud fija. En este caso, el espacio necesario para las vías fijas se utiliza primero antes de que el espacio restante se distribuya a las demás vías.

Espacios entre pistas

Para crear espacios entre pistas, usamos las propiedades:

- **column-gap**: para espacios entre columnas.
- **row-gap**: para espacios entre filas.
- **gap**: como abreviatura de ambos.

Ej.

```
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  gap: 20px;
}
```

Estos espacios pueden ser cualquier unidad de longitud o porcentaje, pero no una unidad **fr**.

Repetición de listas de pistas

Podemos repetir todo o solo una sección de nuestra lista de pistas usando la función **repeat()**.

Ej.

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
}
```

Esto generará tres pistas de 1fr como en el caso anterior. El primer argumento de la función especifica la cantidad de veces que deseamos que se repita la lista, mientras que el segundo argumento es una lista de pistas, que puede ser una o más pistas que deseamos repetir.

Grilla implícita y explícita

La diferencia entre una grilla explícita versus una implícita es:

- **Cuadrícula explícita**: creada con **grid-template-columns** o **grid-template-rows**.
- **Cuadrícula implícita**: extiende la cuadrícula explícita definida cuando el contenido se coloca fuera de esa cuadrícula, como en las filas dibujando líneas de cuadrículas adicionales.

Como forma predeterminada, las pistas creadas en la cuadrícula implícita tienen como tamaño **auto**, lo que significa que son lo suficientemente grandes para acomodar su contenido. Si deseamos dar un tamaño a las pistas de cuadrícula implícitas, podemos usar las propiedades **grid-auto-rows** y **grid-auto-columns**. Si agregamos a **grid-auto-rows** un valor de **100px**, veremos que esas filas creadas ahora tienen 100 píxeles de alto.

La función minmax()

Nos permite establecer un tamaño mínimo y máximo para una pista, por ejemplo, **minmax(100px, auto)**. El tamaño mínimo es de 100 píxeles, pero el máximo es **auto**, que permitirá expandirse para acomodar mejor el contenido.

Algunas de estas funciones y propiedades pueden ser combinadas para crear el patrón que necesitemos o se ajuste a nuestra necesidad. Por ejemplo, podemos hacer que grid cree tantas columnas como quepan en el contenedor, lo hacemos usando la función **repeat()** en **grid-template-columns**, pero en lugar de establecer un número, usamos la palabra clave **auto-fill**

como primer parámetro y como segundo parámetro usamos la función **minmax()** con un valor mínimo igual al tamaño mínimo de pista que nos gustaría tener y un máximo de **1fr**.

Ej.

```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  grid-auto-rows: minmax(100px, auto);
  gap: 20px;
}
```

Esto crea tantas columnas de 200 píxeles como quepan en el contenedor y luego comparte el espacio sobrante entre todas las columnas. El máximo es **1fr** el que reparte el espacio de manera uniforme entre pistas.

grid-template-areas: Es una forma de organizar los elementos en la cuadrícula y dar un nombre a los diversos elementos de nuestro diseño.

Las reglas para **grid-template-areas** son:

- Es necesario que todas las celdas de la grilla estén llenas.
- Para abarcar dos celdas, repetir el nombre.
- Para dejar una celda vacía, usamos un **.** (**punto**).
- Las áreas deben ser rectangulares; por ejemplo, no puede tener un área en forma de L.
- Las áreas no se pueden repetir en diferentes ubicaciones.

[Más info y ejemplos](#)

Fondos y bordes

La propiedad **background** de CSS posee una serie de propiedades de fondo, de las cuales mencionaremos algunas:

- **background-color:** establece el color de fondo
 - **valores:** **color** | **transparent** (valor por defecto: transparent)
- **background-image:** establece la imagen de fondo
 - **valores:** **url** | **none** (por defecto: none)
 - **gradient:** este tipo de datos indica un tipo de imagen que consiste en una transición progresiva entre dos o más colores.
 - **valores:**
 - **linear-gradient:** donde el color se desvanece suavemente a lo largo de una línea imaginaria.
 - **radial-gradient:** Cuanto más lejos de un origen sea un punto, más lejos del color original será.
 - **repeating-linear-gradient** | **repeating-radius-gradient:** donde se repiten gradients lineales o radiales tanto como sea necesario para llenar toda la caja.

- **background-repeat:** controla el comportamiento de tipo mosaico de las imágenes.
 - **valores:** **no-repeat** | **repeat-x** (horizontalmente) | **repeat-y** (verticalmente) | **repeat** (valor por defecto; repite en ambas direcciones)
- **background-size:** ajusta el tamaño de la imagen para que quepa dentro del fondo.
 - **valores:** **cover** | **contain**
- **background-position:** permite elegir la posición en la que aparece la imagen de fondo.
 - **valores:** **valor** | **top** | **center** | **bottom** | **left** | **center** | **right** (por defecto: 0% 0%)
- **background-attachment:** Establece la fijación del fondo. Solo tiene efecto cuando hay contenido por el que se pueda desplazar.
 - **valores:** **scroll** | **fixed** (por defecto scroll)
- **background-clip:** especifica el área de dibujo del fondo. También aplica a imágenes de fondo. Acepta tres valores:
 - **border-box:** el fondo es dibujado hacia el extremo exterior del borde (predeterminado).
 - **padding-box:** el fondo es dibujado hacia el extremo exterior del relleno.
 - **content-box:** el fondo es dibujado dentro de la caja contenedora.

Bordes

Al aprender sobre el modelo de cajas vimos como los bordes pueden afectar el tamaño de nuestra caja. Por lo general, cuando a un elemento añadimos bordes con CSS, usamos una propiedad abreviada que establece el color, el ancho y el estilo del borde en una línea de CSS.

Podemos establecer un borde para los cuatro lados de una caja con **border:**

```
.box {  
  border: 1px solid black;  
}
```

O establecer solo un borde de la caja:

```
.box {  
  border-top: 1px solid black;  
}
```

Las propiedades que se han abreviado serían:

```
.box {  
  border-width: 1px;  
  border-style: solid;  
  border-color: black;  
}
```

Y, en el caso de `border-top`, la forma no abreviada sería:

```
.box {
  border-top-width: 1px;
  border-top-style: solid;
  border-top-color: black;
}
```

Las propiedades de estilo de borde

Hay varios estilos compatibles con la propiedad **border-style**:

- **none**: valor predeterminado, no tiene borde.
- **dotted**: borde punteado.
- **dashed**: borde discontinuo.
- **double**: borde doble.
- **groove**: borde ranurado.
- **ridge**: borde rugoso.
- **inset**: borde insertado.
- **outset**: borde saliente.
- **hidden**: borde oculto.

En CSS, es posible especificar diferentes bordes para diferentes lados, utilizando las propiedades: `border-top-style`, `border-right-style`, `border-bottom-style` y `border-left-style`.

La propiedad **border-width**:

Según la necesidad, otra propiedad de los bordes que puede modificarse es su ancho. Los valores que acepta esta propiedad son **inherit**, **thin**, **medium** y **thick**. También podemos modificarla especificando en píxeles el grosor que deseamos darle.

La propiedad **border-radius**:

CSS nos permite salir del estilo rectangular predeterminado de los bordes, y se los redondea con facilidad. Para ello, utilizamos **border-radius**, al que le especificamos la curvatura de las esquinas en píxeles. 0px corresponde a un rectángulo normal y, a partir de 1px, los extremos comienzan a redondearse de forma más pronunciada.

Por ejemplo:

Para hacer que las cuatro esquinas de una caja tengan un radio de 10 píxeles:

```
.box {
  border-radius: 10px;
}
```

Para hacer que la esquina superior derecha tenga un radio horizontal de 1 em y un radio vertical del 10%:

```
.box {
  border-top-right-radius: 1em 10%;
}
```

Propiedad **box-shadow**

Aplica sombra a los elementos. Los componentes de la propiedad **box-shadow** son decodificados por el navegador de la siguiente manera:

- La primera longitud para el **desplazamiento horizontal** desplegará la sombra hacia debajo de la caja (requerido).
- La segunda longitud es para el **desplazamiento vertical** que desplazará la sombra hacia debajo de la caja (requerido).
- El **color** de la sombra (opcional).
- El valor **blur** que realiza un desenfoque (opcional).
- El valor **spread** que realiza una propagación de la sombra (opcional).

También se puede utilizar valores negativos en donde:

- **Desplazamiento horizontal:** la sombra estará hacia la izquierda de la caja.
- **Desplazamiento vertical:** la sombra estará hacia arriba de la caja.
- **Radio de desenfoque:** los valores negativos no están permitidos.
- **Radio de propagación:** los valores negativos harán que la sombra se encoja.

Desbordamiento de contenido

El Desbordamiento sucede cuando hay demasiado contenido en una caja, y no cabe cómodamente en ella.

CSS proporciona varias herramientas para administrar este desbordamiento.

Propiedad overflow

Es el modo como controlamos el desbordamiento de un elemento. El valor predeterminado de esta propiedad es **visible**, por lo que, podremos ver cuando se desborda nuestro contenido.

Los valores que podemos manejar para la propiedad **overflow** son:

- **hidden:** oculta el desbordamiento, solo lo debemos usar si no causa problema la ocultación del contenido.
- **scroll:** añade barras de desplazamiento cuando el contenido se desborde. Al usar **scroll** el navegador siempre mostrará barras de desplazamiento, incluso cuando no haya suficiente contenido para desbordarse. Esto hace que aparezcan barras de desplazamiento tanto de forma horizontal como vertical.
- **overflow-y:** añade la barra sobre el eje y para poder desplazarse en forma vertical solamente.
- **overflow-x:** añade la barra solo para desplazarse de forma horizontal, aunque no es una forma recomendada.
- **overflow-auto:** esto hace que aparezcan las barras de desplazamiento solo si hay contenido del que cabe en la caja.

TEXTO

Propiedad color: Establece el color del texto.

Tipografías: Son una parte muy importante del mundo de CSS. De hecho, son uno de los pilares del diseño web. La elección de una tipografía adecuada, su tamaño, color, espacio entre letras, interlineado y otras características pueden variar mucho la percepción en la que una persona interpreta o accede a los contenidos de una página.

Existe un amplio abanico de propiedades CSS para modificar las características básicas de las tipografías a utilizar. Veremos algunas propiedades básicas:

- **font-family:** indica el nombre de la fuente a utilizar. Con esta propiedad podemos seleccionar cualquier tipografía simplemente escribiendo su nombre.

Ejemplo:

```
body {
  font-family: Verdana;
}
```

Esta es la forma más básica de indicar una tipografía. Pero se debe tener en cuenta que estas fuentes sólo se visualizarán si el usuario las tiene instaladas en su sistema o dispositivo. Para poder brindar otras alternativas, se añaden varias tipografías alternativas, separadas por comas.

- **font-size:** Indica el tamaño de la fuente:
 - **Medida absoluta (predefinido):** `xx-small` | `x-small` | `small` | `medium` | `large` | `x-large` | `xx-large`
 - **Medida relativa:** `smaller` | `larger` (más pequeña/ más grande)
 - **Específica:** tamaño exacto, simplemente, se indica píxeles, porcentajes u otra unidad que especifique el tamaño concreto de la tipografía.
- **font-style:** Indica el estilo de la fuente (**normal** | **italic** | **oblique**).
- **font-weight:** Indica el grosor de la fuente (100-900):
 - **Medidas absolutas (predefinidas):** **normal** | **bold**.
 - **Medias relativas:** **bolder** | **lighter** (más gruesa/ más delgada).
 - **Medida específica:** valores concretos. Un número del 100 (menos gruesa) al 900 (más gruesa).

En la actualidad, es muy común utilizar [Google Fonts](#) como repositorio proveedor de tipografías para utilizar en nuestros sitios web por varias razones:

- Gratuitas: Disponen de un amplio catálogo de fuentes y tipografías libres y/o gratuitas.
- Cómodo: Resulta muy sencillo su uso: Google proporciona el código y el resto lo hace él.
- Rápido: El servicio está muy extendido y utiliza un CDN, que brinda ventajas de velocidad.

En la página de Google Fonts podemos seleccionar las fuentes con las características deseadas y generar un código HTML con la tipografía (o colección) que vamos a utilizar.

- **text-align:** Permite controlar la alineación por defecto de un texto.
 - **valores:** `left` | `center` | `right` | `justify`
- **text-decoration:** Establece la decoración del texto (subrayado, tachado).
 - **valores:** `none` | `underline` | `line-through` | `overline` (por defecto `none`)
- **text-transform:** Transforma el texto original, a mayúsculas, minúsculas, etc.
 - **valores:** `none` | `capitalize` | `uppercase` | `lowercase` (por defecto `none`)
- **letter-spacing:** Permite establecer el espacio entre las letras que forman las palabras del texto (interletrado).
 - **valores:** `normal` | `unidad de medida` | `inherit` (por defecto `normal`)
- **text-shadow:** Permite manipular textos o títulos, aplicándoles un efecto de sombra.
 - **valores:** `h-shadow` (posición horizontal de la sombra) | `v-shadow` (posición vertical) | `blur-radius` (genera un efecto borroso sobre la sombra) | `color` (aplica a la sombra)

Unidades de medida

Se emplean para definir dimensiones y márgenes de los elementos, también para establecer el tamaño de letra del texto. Se indican con un valor numérico entero o decimal seguido de la unidad de medida.

CSS divide las unidades de medida en:

- **Absolutas:** Son medidas fijas e indican cantidades exactas en alguna unidad. Su valor real es el valor indicado y se ve igual en todos los dispositivos, no dependen de otro valor de referencia. La desventaja que presentan, es que son muy poco flexibles.
- **Relativas:** Definen su valor en relación con otra medida y para obtener su valor real se debe realizar alguna operación con el valor indicado. Dentro de las medidas relativas están las **flexibles**, que son relativas al tamaño del viewport.

Veamos las unidades de medida absolutas:

- **px:** un pixel es el elemento más pequeño de imagen que puede mostrar una pantalla y su medida real depende del dispositivo.
- **cm:** centímetros (10 mm).
- **mm:** milímetros.
- **pt:** puntos. Un punto equivale a 0.35 mm.
- **in:** pulgadas. Una pulgada equivale a 2.54 cm.
- **pc:** picas. Una pica equivale a uno 4.23 mm.

Unidades de medidas relativas:

Su valor siempre está referenciado respecto a otro valor (resolución, densidad de pantalla, etc.). Son las más utilizadas por la flexibilidad con la que se adaptan a los diferentes medios y su potencia:

- **em:** 1em = tamaño de fuente establecida en navegador. 1em equivaldría a 16px, salvo que se modifique por el usuario, mientras que 2em serían justo el doble: 32px. Así también 0.5em equivale a la mitad, es decir, 8px.
 - **em** es relativa al tamaño de la letra, como vemos. Si empleamos un font-size de 10px en el body, 1em equivale a 10px. Su tamaño varía en función del tamaño del elemento padre.
- **ex:** 1ex = 0,5 em aprox. Es aproximadamente la mitad del tamaño de la fuente establecida por el navegador del usuario, por lo que 1ex es igual a 0.5em. La unidad **ex** se basa en la **altura de la x minúscula**. Por lo que su tamaño depende de la tipografía utilizada.
- **ch:** 1ch = tamaño de ancho del cero (0). Equivale al tamaño de ancho del **0** de la fuente actual.
- **rem:** 1rem = tamaño de la fuente del elemento raíz. Toma la idea de la unidad em, pero permite establecer un **tamaño base** personalizado (normalmente para el documento en general, utilizando **html** o la pseudoclase **:root**). De esta forma, podemos trabajar con múltiplos del tamaño base:
 - **html { font-size: 22px; }** //tamaño base
 - **h1 { font-size: 2rem; }** //el doble del tamaño base: 44px
 - **h2 { font-size: 1rem; }** //el mismo tamaño base: 22px

Si queremos cambiar el tamaño del texto en general, sólo tenemos que cambiar el **font-size** de **html** y el resto se modificará en consecuencia.

- **lh:** Altura de la línea del elemento.
- **vw** (viewport width): 1% del ancho de la ventana gráfica. Por ejemplo, un div que mide 50vw, es equivalente al 50% del ancho total del viewport.

- **vh** (viewport height): 1% de la altura de la ventana gráfica. Por ejemplo, un div que mide 50vh y el alto del viewport es de 800px, el div medirá 400px.
- **vmin**: 1% de la dimensión más pequeña de la ventana gráfica.
- **vmax**: 1% de la dimensión más grande de la ventana gráfica.
 - Las unidades **vw**, **vh**, **vmin** y **vmax** son relativas al tamaño del ancho o alto del viewport (ventana gráfica, región visible de la página Web).
- **%**: Relativa a herencia (contenedor padre). Define una unidad en función de otra. Por ejemplo, si estamos trabajando en 12px y definimos una unidad como 150% obtendremos 18px.

Números

Algunos valores aceptan números sin ninguna unidad asociada. Una propiedad que podemos usar de ejemplo es la propiedad **opacity**, que controla la opacidad de un elemento (transparencia). Esta propiedad admite un número entre **0** (totalmente transparente) y **1** (totalmente opaco). Así que para determinar una opacidad media, usamos valores decimales, por ejemplo: **opacity: 0.6**.

Colores

La propiedad **color** se puede usar en cualquier elemento, aunque principalmente se usa para modificar el color del texto y el del background de un elemento.

Hay diferentes formas de especificar el color:

- **Por palabra clave**: red, blue, black, etc.
- **Valor hexadecimal**: **#31078C** (blue) o **#FF0000** (red). Cada par de letras simboliza el valor del RGB.
- **Valor RGB (Red, Green, Blue)**: rgb(250, 0, 250) es rosa, rgb(0, 0, 0) es el color negro y por el contrario rgb(255, 255, 255) es el blanco.
- **Valor RGBA (RGB + Alpha)**: rgba(5, 173, 213, 1), rgba(100%, 62.5%, 100%, 1). El valor Alpha tiene que estar comprendido en [0-1] y hace referencia a la transparencia del elemento, siendo 1 = opaco y 0 = transparente.
- **Valor HSL**: Estos valores, en lugar del rojo, verde y azul, admiten valores de matiz, saturación y luminosidad.
 - **Matiz**: el tono base del color. Toma un valor entre 0 y 360, que representa un ángulo en torno a una rueda de colores.
 - **Saturación**: toma un valor entre 0 y 100%, en el que 0 no es un color (tono gris) y 100% es el nivel de saturación total del color.
 - **Luminosidad**: toma un valor entre 0 y 100%, en donde 0 es negro y 100% es completamente claro, o totalmente blanco.
- **Valor HSLA**: es el equivalente a HSL con el agregado del canal Alpha.

Propiedades Height y width

Estas propiedades (alto y ancho) de todo elemento por defecto están configuradas en un valor automático, lo que hace que ocupen todo el ancho y el alto predeterminado por su contenido. A su vez, CSS permite, no solo especificar un alto y un ancho sobre los elementos, sino también limitarlos a un mínimo y un máximo.

Algunos ejemplos de uso:

- **height: 30px;**
- **width: 150px;**
- **max-height: 55px;** //determina el alto máximo de un elemento

- **max-width: 300px;** //determina el ancho máximo
- **min-height: 50px;** //determina el alto mínimo
- **max-width: 270px;** //determina el ancho mínimo

Hipervínculos

Los links o hipervínculos vienen por defecto establecidos en un estilo subrayado, con un color de fuente azul, que se torna a violeta una vez accedido. A través de CSS podemos personalizar el estilo de cada hipervínculo, manejados a través del tag **<a>**; es posible quitar el subrayado predeterminado y alternar el color de visualización, el color del mismo cuando el cursor se posiciona sobre este, y hasta el color que se muestra para los links activos.

Valores a utilizar:

- **a:link:** hipervínculo normal, no visitado.
- **a:visited:** hipervínculo que ha sido visitado.
- **a:hover:** hipervínculo donde el cursor se encuentra encima o señalándolo.
- **a:active:** hipervínculo en el momento en el cual es clickeado o activado.

Dando estilos a las listas

Las listas cobreados estilos más dinámicos y coloridos gracias a CSS; incluso, es posible salir de la clásica viñetas (para las listas desordenadas) e incorporar en su lugar diseños basados en archivos de imágenes.

Valores que se pueden utilizar:

- **list-style-type:** Permite especificar el estilo de viñeta o marcadores para las listas. Puede ser establecida como **circle**, **square**, **decima**, **disc**, **lower-alpha**, etc.
- **list-style-image:** Permite especificar una imagen que reemplaza a la viñeta. Sin embargo esta propiedad es un poco limitada por lo que respecta al control de la posición, el tamaño, etc., de las viñetas. Es más conveniente usar la familia de propiedades **background**.
- **list-style-position:** Especifica si los marcadores de elementos de lista deben aparecer dentro o fuera del contenido de la lista. Los valores que utiliza son **inside** u **outside**. El valor por defecto es **outside**.
- **list-style:** Esta propiedad simplifica el uso de las otras tres propiedades. En su declaración, agregamos los valores en forma de parámetros con el siguiente orden:
 - **list-style-type list-style-position list-style-image**

TABLAS con estilo

El aspecto de una tabla HTML se puede mejorar mucho con CSS. Algunas de las propiedades que podemos encontrar son:

- **border-collapse:** especifica si los bordes de la tabla se contraen en un solo borde o se separan de forma predeterminada.
- **border-spacing:** Si los bordes están separados, se puede usar para cambiar el espaciado.
- **caption-side:** Especifica la posición del título en la tabla. Los valores se pueden establecer como **top** (superior) o **bottom** (inferior).
- **empty-cells:** Especifica si muestra o no los bordes y el fondo de las celdas vacías de una tabla. Los valores posibles son **show** (mostrar) o **hide** (ocultar).
- **table-layout:** Especifica cómo se calcula el ancho de las columnas de la tabla. Los valores posibles son:

- **auto:** cuando el ancho de las columnas o celdas no está establecido explícitamente, el ancho de la columna será en proporción a la cantidad de contenido en las celdas que conforman la columna. Valor por defecto.
- **fixed:** cuando el ancho de las columnas o celdas no está establecido explícitamente, el ancho de la columna no será afectado por la cantidad de contenido en las celdas que conforman la columna.

Limitar el ancho de una tabla:

```
table {
  width: 80%;
}
```

El elemento table posee la capacidad de controlar el ancho que tendrá una tabla dentro de una página web. El ancho mínimo expresado en porcentaje es 20%, y el ancho máximo está limitado al ancho del documento HTML. En este último caso, se podría representar a través de la propiedad width: 100%;

Resaltar filas

En determinadas ocasiones, podemos encontrarnos con que generar tablas extensas que requieren visualizar muchos datos terminan generando una confusión al usuario que necesita consultar dicho contenido. Podemos optar por el **selector nth-child** o el **selector hover**. En ambos casos, debemos aplicar cualquiera de estas soluciones a la fila completa de una tabla, para resaltar su contenido de manera fija (**nth-child**) o de forma dinámica, cuando el usuario posicione el cursor del mouse sobre una fila (**tr:hover**).

INTERCALAR FONDO EN FILAS: Con la propiedad nth-child, podemos alternar el color de fondo de las filas de una tabla. Si trabajamos con datos dinámicos, provenientes de una base de datos, y no sabemos cuántos registros mostraremos, podemos calcular el total de estos, y sobre el resultado aplicar **nth-child(odd)** o **nth-child(even)**, asegurándonos el contraste del fondo de las filas contra el color de fondo del encabezado de estas. Eso da a las filas pares y a las impares, colores diferentes.

Transiciones

Las transiciones en CSS nos permiten cambiar de un valor de propiedad a otro sobre una duración dada.

- **transition-property:** especifica la propiedad a ser transicionada.
- **transition-duration:** especifica la duración sobre la cual la transición debe ocurrir.
- **transition-timing-function:** especifica cómo el ritmo de la transición cambia sobre su duración. Puede tener los siguientes valores:
 - **ease:** la animación comenzará lentamente, luego acelerará rápidamente (predeterminado).
 - **ease-in:** comienza lentamente, luego acelera, y se detiene abruptamente.
 - **ease-out:** comienza rápidamente, pero se desacelera hasta detenerse.
 - **ease-in-out:** similar que **ease**, pero con una aceleración y desaceleración más sutil.
 - **linear:** velocidad constante a través de la animación; usualmente mejor para cambios de color u opacidad.

- **cubic-bezier()**: permite definir nuestros propios valores. Los posibles valores son desde 0 hasta 1.
 - Ej.: **transition-timing-function: cubic-bezier(0, 0, 1, 1);**
- **transition-delay**: especifica un retraso (en segundos) para el efecto de transición.

*Los efectos de transición pueden ser aplicados a una amplia variedad de propiedades CSS, incluyendo **background-color**, **width**, **height**, **opacity** y muchas más....*

Transformaciones

Las transformaciones en CSS permiten trasladar, rotar, escalar y sesgar elementos.

Una transformación es un efecto que permite a un elemento cambiar su forma, tamaño y posición. CSS soporta transformaciones 2D y 3D.

- **rotate()**: rota un elemento en sentido de las agujas del reloj o en sentido contrario (valores negativos), de acuerdo a un ángulo dado.
- **transform-origin**: permite cambiar la posición de los elementos transformados. El valor por defecto para la propiedad es 50% 50%, que corresponde al centro del elemento. Esta propiedad debe ser utilizada junto con la propiedad transform.
 - **0 0** es el mismo valor que **top left**, y **100% 100%** es el mismo valor que **bottom right**.
- **translate()**: mueve un elemento desde su posición actual. Los valores positivos empujarán un elemento hacia abajo y hacia la derecha desde su posición por defecto, mientras que los valores negativos empujarán un elemento hacia arriba y hacia la izquierda de su posición actual.
- **skew()**: sesga un elemento a lo largo del eje-x y el eje-y según los ángulos dados.
- **scale()**: aumenta o reduce el tamaño de un elemento, de acuerdo a los parámetros dados para el ancho y el alto. 1 equivale al tamaño original, 2 para el doble del tamaño original, y así sucesivamente. Si pasamos sólo un parámetro al método scale(), se aplicará ese factor tanto a la altura como al ancho.

Se pueden usar varias transformaciones al mismo tiempo.

FILTROS

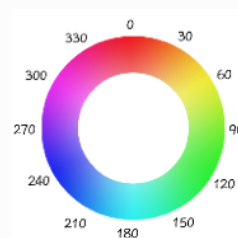
La propiedad **filter** permite aplicar efectos gráficos como **desenfoque (blurring)** o **cambio de color (color shifting)** a un elemento.

Los filtros se usan comúnmente para ajustar la representación de imágenes, fondos y bordes. El filtrado de imágenes es útil cuando desea tener un estilo diferente para la misma imagen. En lugar de cargar varias imágenes en el sitio web, puede cargar solo una imagen y luego definir los efectos visuales usando la propiedad **filter**.

Las funciones de filtro incluyen:

- **blur()**: aplica un efecto de **desenfoque** a una imagen. Sólo tiene como parámetro, **el radio**, que define cuántos píxeles en la pantalla se mezclan entre sí. (un mayor valor crea más desenfoque. El parámetro se especifica como una longitud CSS, pero no acepta valores porcentuales.

- **brightness(amount):** ajusta el brillo de una imagen. El parámetro amount (cantidad) determina el nivel de brillo de la imagen. El parámetro puede tomar un valor porcentual o un número. Un valor **0%** es una imagen completamente **negra**. Un valor 100% da como resultado una imagen que no **ha cambiado**. Cualquier cantidad superior al 100% produce una imagen más brillante, mientras que un valor inferior, oscurece la imagen.
- **contrast():** ajusta el contraste de la imagen. El parámetro puede tomar un valor porcentual o un número. A medida que el valor llega al 100%, el contraste aumenta. Un valor **0%** deja la imagen totalmente gris, y un contraste de **100%** no produce ningún cambio en la imagen.
- **drop-shadow(w h b c):** crea un efecto de sombra que se extiende más allá de una imagen para el ancho **w** y el alto **h** con desenfoque **b** y color **c**. Éstos son valores en píxeles.
- **grayscale(%):** convierte una imagen a **escala de grises**. El único parámetro define la proporción de la conversión. **0%** es la imagen original, mientras que **100%** hace que la imagen sea completamente en escala de grises.
- **hue-rotate():** aplica una rotación de tono (basada en el círculo de color) a una imagen. Esta función toma como parámetro un ángulo de rotación. El valor del ángulo define el número de **grados** que se ajustará alrededor del círculo. Por ejemplo, si la imagen contiene color rojo, que está a 0 grados en el círculo de color, rotar el tono 240 grados hará que el color rojo sea azul. Las rotaciones de 0° y 360° dejan la imagen sin cambio.
- **invert():** invierte los colores de una imagen para hacer que **las áreas oscuras se iluminen** y **las áreas brillantes se oscurezcan**. El parámetro que toma es la proporción de la conversión. Éste puede ser un valor **porcentual** o un **número**. 0% deja la imagen sin cambios, mientras que **100%** crea una imagen completamente invertida, similar a un negativo fotográfico.
- **opacity():** establece la opacidad de una imagen para cambiar su transparencia. **0%** crea una imagen totalmente **transparente**, mientras que **100%** es la imagen original.
- **saturate():** controla la **saturación de color** de una imagen. El único parámetro determina la **proporción de la saturación** que se aplica a la imagen. Éste puede ser un valor porcentual o un número. **0%** crea una imagen saturada (escala de grises), mientras que **100%** es la imagen original.
- **sepia(%):** convierte una imagen a sepia. Es decir, la convierte en tonos marrón-rojizo. Trabaja como en el caso de escala de grises con porcentaje donde **0%** es la imagen original y **100%** completamente **sepia**.



Uso de múltiples filtros: se pueden usar varios filtros CSS juntos separándolos con espacios.

Recursos consultados:

www.developer.mozilla.org

www.desarrolloweb.com

Colección "Programador web Full-Stack" - Red Users

