Estilos CSS

Sintaxis CSS

La sintaxis para aplicar reglas CSS es la siguiente:

```
<selector> {
        /*property: value*/;
        color: blue;
}
```

Selectores CSS

Podemos separar los selectores CSS en varias categorías:

1- Elemento: Se aplican estilos a todos los componentes de un documento HTML que estan bajo un mismo elemento:

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

2- Identificador: Se aplican los estilos al elemento asociado a un identificador:

```
#parrafo {
    color: red;
}
```

3- Clase: Se aplican los estilos al elemento o elementos asociados a una clase:

```
.parrafo {
    color: red;
}
```

4- El selector universal: Se aplican los estilos a todos los elementos del documento HTML.

```
* {
    color: red;
}
```

5- Selector agrupado. Si tenemos varias directrices con la misma o mismas reglas para varios elementos. Se pueden agrupar de la forma:

```
#parrafo, span, .clase{
    color: red;
}
```

Como veis, para utilizar selectores agrupados, se pueden mezclar tanto elementos, como identificadores y clases.

Cómo añadir CSS a nuestra página

Podemos añadir estilos CSS a nuestra página de 3 maneras distintas:

1- Externo: Se carga un fichero externo de CSS

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

2- Interno: Se añaden los estilos dentro de la etiqueta **style** y todo el bloque se añade a la etiqueta **head:**

```
<!DOCTYPE html>
<html>
        <head>
                <style>
                        body {
                         background-color: linen;
                        }
                        h1 {
                         color: maroon;
                         margin-left: 40px;
                        }
                </style>
        </head>
<body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
</body>
</html>
```

3- En línea: Se añaden los estilos directamente a un elemento HTML:

<p style="color:red;">Hola mundo!<\p>

Prioridad de Estilos

La prioridad de estilos se realiza en cascada. Si tenemos estilos aplicados para un **h1** tanto en el documento externo, como de manera interna y en línea, el orden de prioridad es el siguiente:

1- CSS en línea.
2- Externo
3- Interno
4- CSS del navegador por defecto.

Comentarios

Tanto en un fichero .css como en un fichero HTML dentro de la etiqueta **style,** los comentarios se escriben con el siguiente formato:

/*Soy un comentario*/

Unidades

Unidades Absolutas

cm: centimetros
mm: milimetros
in: pulgadas (1in = 96px = 2.54cm)
px: pixels (Relativa a la pantalla)

pt: 1pt = 1/72 de 1in
pc: 1pc = 12pt

Unidades Relativas

em: relativo al tamaño de fuente
rem: relativo al tamaño de fuente del elemento raiz (html)
vw: relativo al ancho de la pantalla 1vw = 1% del ancho de la pantalla
vh: realtivo al alto de la pantalla 1vh = 1% del alto de la pantalla
%: Relativo al elemento padre

Colores

Cambiar color de texto

color: valor;

Cambiar color de fondo

background-color: valor;

Cambiar color del border

border-color: valor;

Posibles valores

CSS permite diferentes vías para declarar valores de colores:

- **Hexadecimal:** #FFFFFF
- **RGBA:** rgba(255,255,255,1)
- **Valores predefinidos por los navegadores:** red, blue, yellow, etc. Puedes ver la lista entera en el siguiente enlace:
    - https://www.w3schools.com/colors/colors_names.asp

Fondos

Cambiar fondo

background-color: valor;

Imagen fondo

background-image: url("ruta de la imagen");

Repeat

Por defecto cuando añadimos una imagen de fondo, html la repite hasta ocupar todo el ancho del contenedor al que se le aplica. Para configurar esta repetición podemos utilizar la regla **background-repeat: valor;**

- **background-repeat: repeat-x;** //Repite la imagen solo horizontalmente
- **background-repeat: repeat-y;** //Repite la imagen solo verticalmente
- **background-repeat: no-repeat;** //No repite la imagen

Posición

Define la posición de inicio de la imagen de fondo. Los posibles valores son: top, left, right, bottom.

- **background-position: left top;**

Attachment

Se le dice a la imagen de fondo si se queda fija o hace scroll con el resto de la página:

- **background-attachment: scroll;** //Cuando haces scroll la imagen desaparece junto al resto de contenido
- **background-attachment: fixed;** //Cuando hace scroll la imagen se mantiene al mismo sitio hasta que acaba el contenedor al que está haciendo

Origen

Especifica el origen de la posición de la imagen de fondo. Posibles valores:

- **padding-box (defecto):** El origen de la imagen es arriba a la izquierda del elemento que establece la imagen de fondo.
- **border-box:** La imagen empieza justo en el borde arriba a la izquierda.

- **content-box:** El fondo empieza arriba a la izquierda pero donde empieza el contenido.

Tamaño

Especifica el tamaño de la imagen de fondo. Posibles valores:

- **auto:** defecto. Tamaño de la imagen original.
- **lenght:** Especificar ancho y alto de la imagen en la unidad que queramos . El primer valor especifica el ancho y el segundo valor especifica el alto.
- **percentage:** exactamente igual pero en % para especificar alto y ancho.
- **cover:** Redimensiona la imagen para cubrir el alto y el ancho del contenedor. Mantiene el aspect ratio de la imagen. Lo que hará ocultar partes de la imagen.
- **contain:** Redimensiona la imagen para cubrir todo el contenedor pero no mantiene el aspect ratio, con lo que se vera la imagen entera pero deformada.
- **initial y inherit:** igual que para background-origin.

Bordes

Estilo

Le otorga un estilo al borde:

border-style: valor;

- **dotted:** borde a puntos.
- **dashed:** borde a rallas
- **solid:** borde de línea contínua
- **double:** doble línea
- **groove, ridge, inset, outset:** bordes con efecto 3D
- **none:** sin borde
- **hidden:** borde oculto

Ancho

Especifica el ancho del borde:

border-width: valor;

Color

Especifica el color del borde:

border-color: valor;

Forma abreviada

Los bordes se suelen aplicar con su forma abreviada:

border: [ancho] [estilo] [color];

border: 1px solid blue;

Bordes a lados concretos

También se puede asignar el borde a solo uno o varios lados del elemento al que se le asigna, o bién establecer diferentes reglas de bordes a cada lado:

- **border-top: 5px solid blue;**
- **border-bottom: 5px solid red;**
- **border-left: 5px solid yellow;**
- **border-right: 5px solid green;**

Borde redondeado

Se puede añadir un borde redondeado a los elementos HTML. Cuanto mayor sea el valor, mayor será la redondez del borde.

border-radius: valor; → **border-radius: 10px;**

Márgenes

Los márgenes se utilizan para crear espacio alrededor de los elementos a lo que se asigna, incluso del borde si tuviera.

- **margin-top, margin-bottom, margin-left y margin-right**

Se pueden añadir todos a la vez de dos maneras:

- Para añadir el mismo margen a todos los lados:

  margin: 30px;

- Para asignar margen distinto a cada lado:

  - margin: [arriba y abajo] [derecha e izquierda]
  margin: 10px 20px; //Esta linea añade 10px de margen arriba y abajo del elemento y 20px a los lados

  - **margin: [arriba][derecha][abajo][izquierda]**
  margin: 10px 20px 30px 40px; // Añade 10px arriba, 20px a la derecha, 30px abajo y 40px a la izquierda.

- También se puede añadir el valor auto. De esta manera añade el margen necesario para alinear horizontalmente al medio:

margin: 0 auto;

Esta regla especifica 0 pixeles de margin arriba y abajo y a los lados especifica un márgen automático, lo que hará que se alinee al centro del elemento padre.

Páddings (Márgenes internos)

Añade espacio alrededor del contenido del elemento pero dentro de los márgenes de éste.

Se añade de la misma manera que los márgenes.

Ancho y alto (width y height)

Especifican las dimensiones de un elemento. Estas propiedades establecen las dimensiones del elemento si tener en cuenta los márgenes, paddings y bordes.

Posibles Valores:

- **auto (default)**: El navegador lo calcula automáticamente.
- **length**: Define el tamaño en px, cm, etc.
- **%**: Porcentage en relacion al contenedor padre

Ademas se puede poner un tope al ancho con:
max-width: 100px;

Outline

Propiedad CSS parecida al borde. Se dibuja alrededor del elemento al que se establece. A diferencia de los bordes los contornos no ocupan espacio, son dibujados por encima del elemento.

- **outline-style: valor;** (Posibles valores: dotted, dashed, solid, double, groove, ridge, inset, outset, none, hidden)

- **outline-width: valor;** (Posibles valores: thin(1px), medium(3px), thick(5px) o especificar tamaño en (px, pt, cm, etc.))

- **outline-color: valor;** Especifica el color del outline

- **outline-offset: valor;** Añade espacio entre el outline y el borde del elemento. Digamos que es un padding entre el borde del elemento y el outline.

Texto

Color

Para aplicarle un color al texto se utiliza la siguiente regla CSS:

color: valor;

Los posibles valores, como ya hemos visto son colores en formato hexadecimal, rgb, rgba y colores predefinidos del navegador.

Alineación de Texto

Para ajustar el alineamiento horizontal de un elemento respecto a su elemento padre es:

text-align: valor;

Los posibles valores son: **center, left, right y justify**

Dirección del texto

Dependiendo del idioma, se escribe de izquierda a derecha o de derecha a izquierda. Para asignar estos valores existe la regla:

direction: valor;

Posibles valores: **rtl y ltr** (derecha a izquierda e izquierda a derecha respectivamente)

Añadir línea de decoración al texto

Se puede añadir una linea de decoración al texto con la regla:

text-decoration-line: valor;

Posibles valores:

- **overline:** linea por encima del texto
- **line-through:** línea atravesando el texto (texto tachado)
- **underline:** texto subrayado
- **overline underline:** linea decorativa por encima y debajo del texto

Aplicar color a la línea de decoración al texto

Para especificar el color de la linea decorativa que le hemos asignado a un elemento con la regla **text-decoration-line** es la siguiente regla:

text-decoration-color: valor;

Especificar estilo de la línea de decoración al texto

Se le puede especificar un estilo a la línea de decoración del texto con la siguiente regla:

text-decoration-style: valor;

Los posibles valores son: **solid, double, dotted, dashed, wavy**

Espesor de la línea de decoración al texto

Podemos asignar un espesor específico a la line de decoración del texto:

text-decoration-thickness: valor;

Modo abreviado

Al igual que con los márgenes, los bordes y los paddings se puede utilizar la manera abreviada:

text-decoration: [localización] [color] [estilo] [espesor];

text-decoration: underline red double 5px;

Transformación del Texto

Se pueden aplicar transformaciones al texto con la siguiente regla:

text-transform: valor;

Los posibles valores son:

- **uppercase:** Transforma el texto y lo convierte todo a mayúsculas.
- **lowercase:** Transforma el texto y lo convierte todo a minúsculas.
- **capitalize:** Transforma el texto de forma que convertirá la primera letra de cada palabra del texto a mayúscula.

Espaciado del texto

Indentación del Texto

Para añadir indentado al texto se consigue con la siguiente regla:

text-indent: valor;

Como valor tenemos que especificar una unidad absoluta o relativa. (mm, px, %, rem, …)

Espaciado entre letras

Se puede especificar el espacio que habrá entre las letras de un texto:

letter-spacing: valor;

Por ejemplo: **letter-spacing: 2px;** quiere decir que entre los caracteres del texto al que se le aplique habrá dos 2 píxels de distancia.

Espaciado entre líneas de texto

Además del espaciado entre caracteres en un texto, también se puede especificar el espaciado entre líneas de texto:

line-height: valor;

Donde el valor es una unidad absoluta.

Espaciado entre palabras

También existe la regla CSS para aplicar un espaciado personalizado entre las palabras de un texto:

word-spacing: valor;

Donde el valor es una unidad absoluta.

Sombra del texto

Se puede aplicar una sobra en un elemento de texto. La regla es la siguiente:

text-shadow: [arriba-abajo] [derecha-izquierda] [desenfoque] [color];

Por ejemplo: **text-shadow: 2px 2px 5px red;**

Fuentes de texto

Para declarar una fuente de texto a un elemento HTML se realiza con la regla:

font-family: valor, valor2, valor3;

Donde cada valor hay que reemplazarlo por una tipografía. Si hay varias especificadas el navegador intenta cargar la primera, si no puede carga la segunda y si no puede cargar ninguna cargará la fuente por defecto del navegador.

Estilo de la fuente

Para aplicar el estilo a la fuente se utiliza la siguiente regla:

font-style: valor;

Los posibles valores son: **normal, italic, oblique.**

Grosor de la fuente

Para aplicarle un grosor distinto a la fuente se puede hacer con la regla:

font-weight: valor;

Los posibles valores son: **normal, bold.**

Tamaño de la fuente

Para especificar el tamaño de la fuente hay que aplicar la siguiente regla al elemento de texto:

font-size: valor;

Donde el valor es una de las unidades absolutas o relativas.

Iconos

La manera más simple de añadir iconos a nuestra página es utilizar una librería externa como por ejemplo: Font-Awesome. Hay que tener en cuenta que cargar una librería tiene su peso y puede afectar al rendimiento de nuestra página. Si solo tenemos que utilizar unos pocos es mejor descargarlos como imagen y utilizarlos así.

A continuación, se describen las librerías más famosas de iconos.

Font Awesome

Para poder utilizar los iconos de la librería FontAwesome, se requiere cargarla dentro de las etiquetas <head> de nuestro documento HTML:

<script src="https://kit.fontawesome.com/*yourcode*.js" crossorigin="anonymous"></script>

NOTA: Lo más recomendable es buscar la última versión estable.

Bootstrap

Otra librería famosa tanto para iconos como para el framework completo.

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

NOTA: Lo más recomendable es buscar la última versión estable.

Google Icons

Finalmente, podemos cargar también la librería de iconos de Google.

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

NOTA: Lo más recomendable es buscar la última versión estable.

Estilos en los enlaces

Con CSS también podemos capturar los diferentes estados de un enlace. Los diferentes estas capturables son los siguientes:

Link

Es la opción por defecto y sirve para darle estilos a un enlace que aun no ha sido visitado:

```
a:link{
   //Reglas a aplicar
}
```

Visitado

Es la opción sirve para darle estilos a un enlace que ya ha sido visitado:

```
a:visited{
   //Reglas a aplicar
}
```

Hover

Es la opción sirve para darle estilos a un enlace cuando el puntero esta encima de él:

```
a:hover{
   //Reglas a aplicar
}
```

Activo

Es la opción sirve para darle estilos a un enlace en el momento en que es clicado:

```
a:active{
   //Reglas a aplicar
}
```

Listas

Estilos para listas

CSS tiene por defecto distintos estilos para los distintos tipos de listas. Estos estilos se pueden especificar:

list-style-type: valor;

Posibles valores: **circle, square, upper-roman, lower-alpha**

Imagen como marcador del item

También se puede especificar una imagen para el marcador del item de la lista, es decir, para sustituir los puntos, numeros o letras de las listas:

list-style-image: url('imagen.png');

Posición de los marcadores del item

Para especificar la posición de los marcadores de lista se utiliza la siguiente regla:

list-style-position: valor;

Posibles valores:

- **outside:** los marcadores de los items de la lista se ubican fuera de los limites del elemento lista.
- **inside:** os marcadores de los items de la lista se ubican dentro de los limites del elemento lista.

Eliminar opciones por defecto

Se pueden eliminar las reglas por defecto asociadas a los items de lista con las siguientes reglas:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

Tablas

Bordes de tabla

Para definir borde en las tablas se utiliza la misma regla que para los demás elementos:

```
table, td, th{
   border: 1px solid;
}
```

Tabla ancho 100%

Para asignar el ancho de una tabla podemos utilizar la regla de ancho:

```
table{
   width: 100%;;
}
```

Colapsar los bordes de la tabla

Con el ejemplo anterior, donde le hemos aplicado borde tanto a la table como a los th y td, nos aparecerá un borde doble. Con la siguiente re

```
table {
  border-collapse: collapse;
}
```

Si solo le aplicamos el borde al elemento table obtendremos un borde perimetral de la tabla.

Tamaño de tabla

Podemos ajustar el tamaño de las celdas utilizando las reglas **width** y **height.**

Alineamiento de tabla

Para alinear el contenido de las celdas podemos hacerlo de manera horizontal y de manera vertical:

- **Horizontal.** Se realiza con la regla **text-align: valor;**
    - Posibles valores: left, right, center
- **Vertical.** Se realiza con la regla **vertical-align: valor;**
    - Posibles valores: top, bottom, middle.

Display

La regla **display** es una de las más importantes para controlar el diseño de la página y sirve para decirle a un elemento como se va a mostrar y si se va a mostrar.

Por defecto, cada elemento tiene asociado un **display** dependiendo de si es un elemento de **bloque** o **en línea**

Elementos de bloque

Los elementos de tipo bloque siempre empiezan en una nueva línea y ocupan el 100% del elemento que lo contiene (elemento padre).

Elementos en línea

Los elementos en línea no empiezan en una nueva línea y ocupan el ancho que tienen y no el 100%. Solo comenzaría en una línea nueva si no tiene espacio suficiente.

No mostrar elementos

Si lo que queremos es no mostrar un elemento podemos utilizar la siguiente regla con el siguiente valor:

display: none;

Sobreescribir display

Aunque los elementos tienen asociado un display por defecto, podemos sobreescribirlo utilizando la regla:

```
span{
   display: block;
}
```

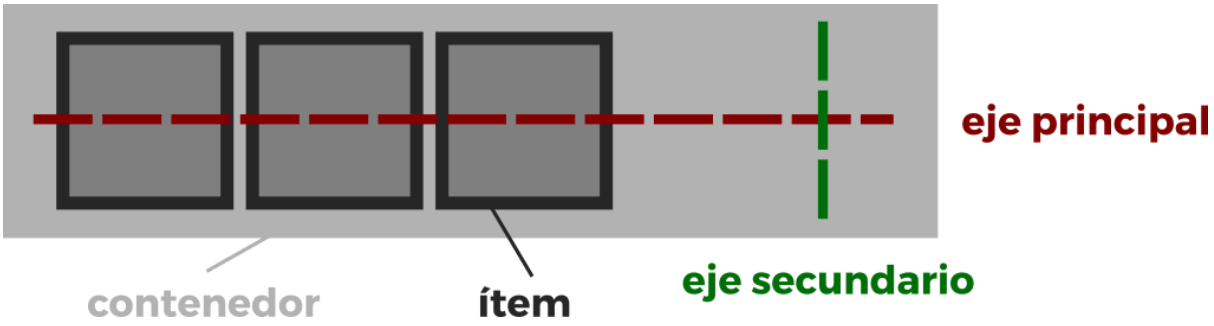
Display **flex**

Para empezar a utilizar **flex** lo primero que debemos hacer es conocer algunos de los elementos básicos de este nuevo esquema, que son los siguientes:

Contenedor: Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles. Observa que al contrario que muchas otras estructuras CSS, por norma general, en Flex establecemos las propiedades al elemento padre.

Eje principal: Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flex es en horizontal (*en fila*).

Eje secundario: De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (*y viceversa*).



Como hemos comentado el elemento al que se le asigna como **flex** es el contenedor padre.

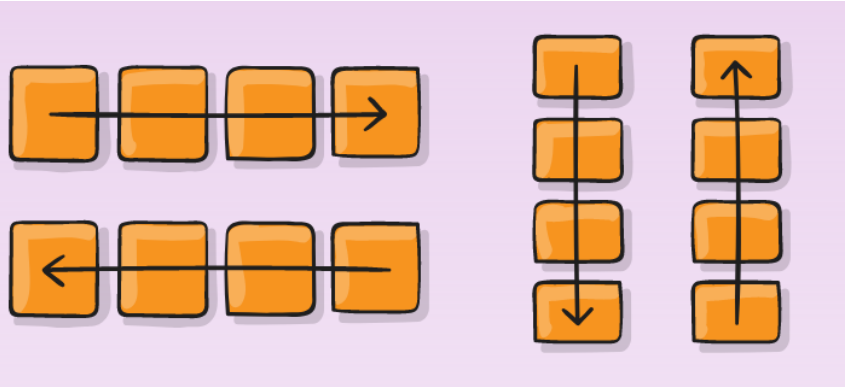
PROPIEDADES PARA EL CONTENEDOR PADRE

Display: Lógicamente la primera regla a aplicar a un contenedor que queremos que sea **flex** es asignarle la propiedad: ***display: flex;*** Establece un contexto flex para todos sus hijos (elementos que contiene)

A continuación ya podemos detallar las demás propiedades aplicables a un contenedor padre de tipo **display: flex.**

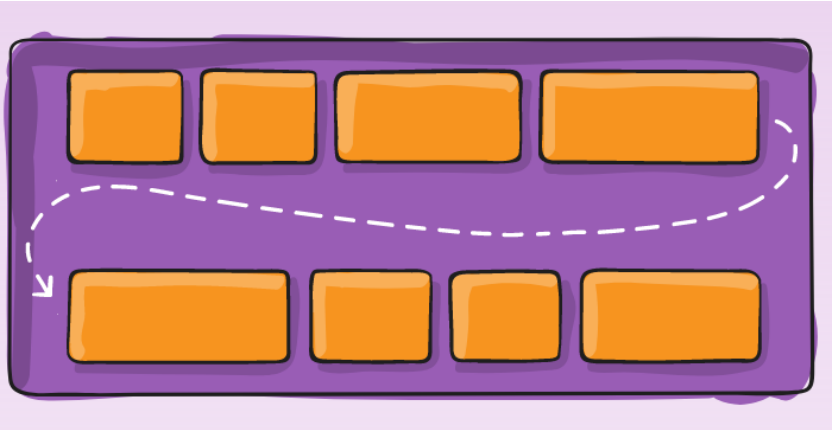
Flex-direction: Establece la dirección del eje principal. Los posibles valores son:

- **row (defecto):** left to right en ltr; right to left en rtl
- **row-reverse:** Al revés qué row.
- **column:** lo mismo que row pero de arriba a abajo
- **column-reverse:** al revés de column.



Flex-wrap: Por defecto, los elementos flex intentarán colocarse todos en una línea pero esto se puede cambiar. Posibles valores:

- **nowrap (defecto):** todos los elementos flex estarán en la misma fila.
- **wrap:** los elementos flex que no quepan pasarán a la línea de abajo. Se van colocando en múltiples líneas de arriba a abajo.
- **wrap-reverse:** igual que wrap pero de abajo a arriba.



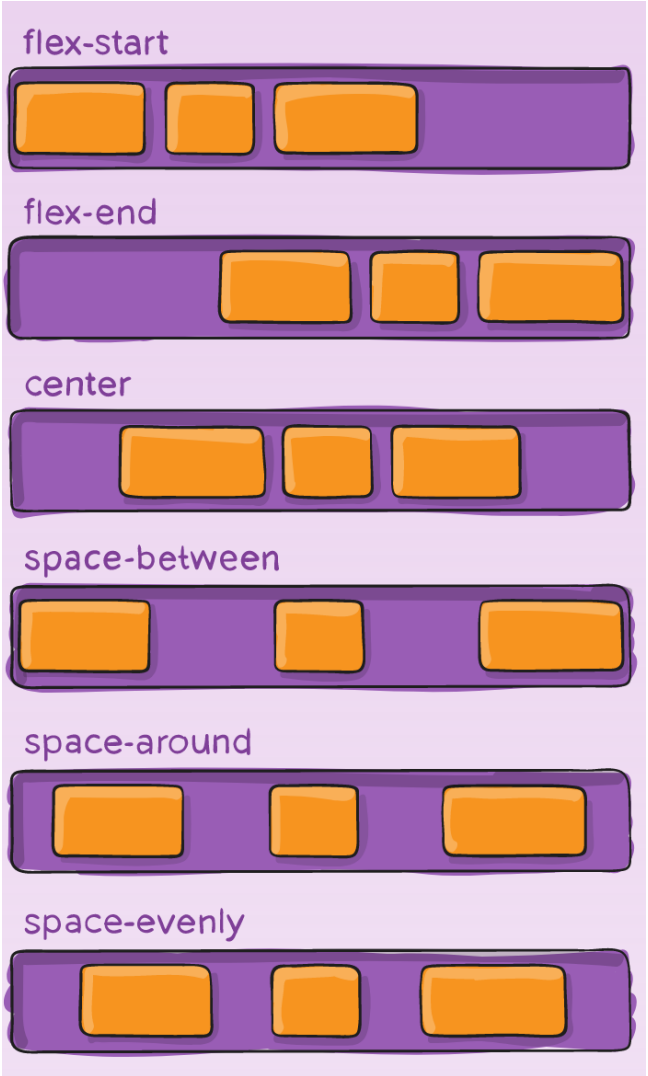
Flex-flow: Esta propiedad actúa como atajo de las dos anteriores, *flex-direction y flex-wrap* respectivamente. Por ejemplo, un contenedor padre con display: flex, flex-direction: row y flex-wrap: nowrap quedaría de esta forma.

flex-flow: row nowrap;

Justify-content: Determina cómo se alinean los elementos flex dentro del container padre. Posibles valores:

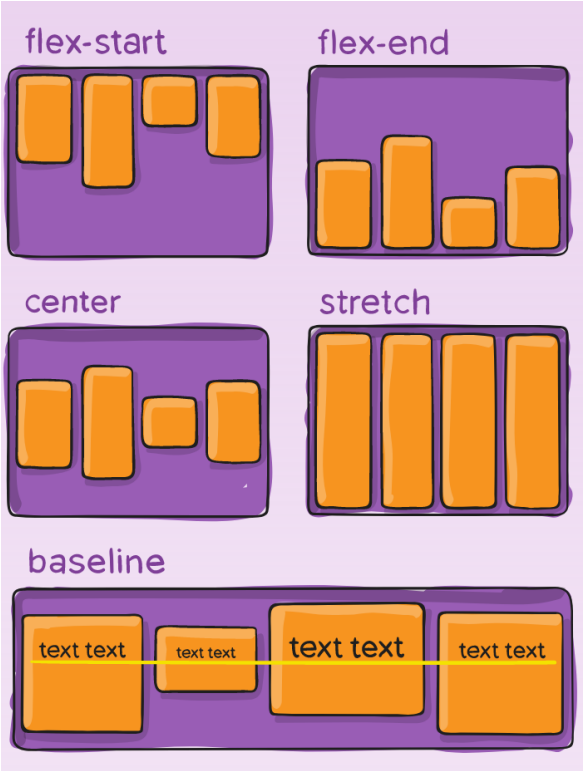
- **flex-start:** alineado al principio del contenedor.
- **flex-end:** alineado al final del contenedor.
- **center:** alineado al centro
- **start:** items alineados al principio pero respecto a la dirección de escritura establecidas en ese elemento (rtl, ltr).
- **end:** items alineados al final pero respecto a la dirección de escritura establecidas en ese elemento (rtl, ltr).

- **space-between:** los elementos ocupan todo el contenedor y se aplican espacios entre ellos para que estén repartidos.
- **space-around:** igual que el anterior pero además se establecen espacios alrededor de los elementos.

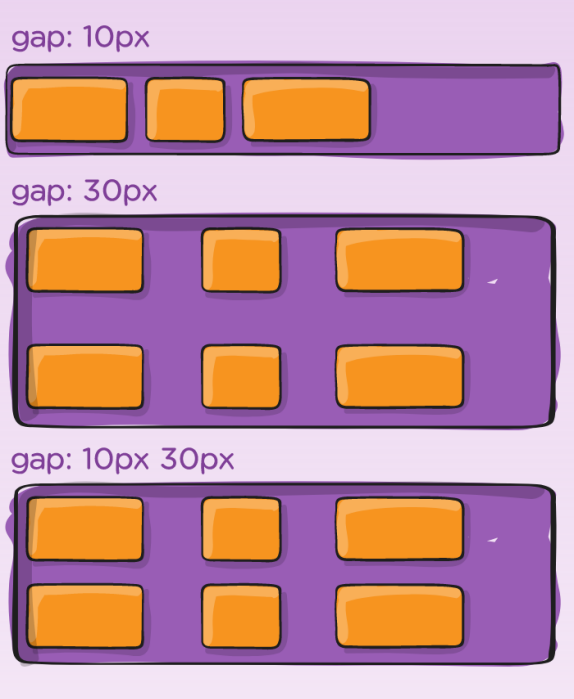


Align-items: alinea los elementos flex respecto al eje vertical. Posibles valores:

- **stretch (default):** hace que los elementos tengan la misma altura y ocupen todo el alto del contenedor padre.
- **flex-start:** alinea los elementos arriba del contenedor padre.
- **flex-end:** alinea los elementos debajo del contenedor padre.
- **center:** alinea los elementos al centro.



gap, row-gap, column-gap: establece el espacio que debe haber entre los items que contiene el elemento flex



PROPIEDADES PARA LOS ÍTEMS EN EL CONTEXTO FLEX

Ahora procederemos a describir las distintas reglas aplicables a los ítems que conviven en el contexto flex. Es decir, que están contenidos dentro del contenedor flex.

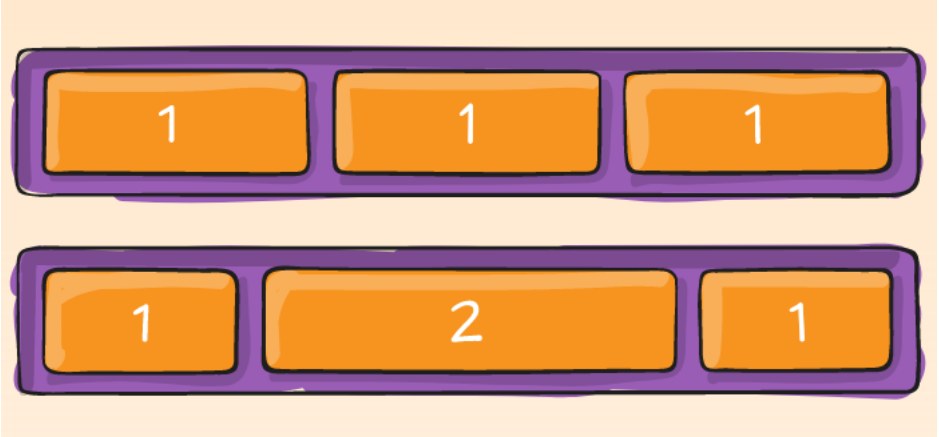
order: Por defecto, los ítems flex tienen el orden establecido, el orden en el que aparecen en el documento HTML pero este orden se puede cambiar. Por ejemplo:

```
.item {
    order: 5; /* Default: 0 */
}
```

En el ejemplo anterior se le asigna a los elementos de la clase .item el orden 5 dentro del contenedor flex. Si varios elementos comparten el mismo order, el orden vuelve a ser el que aparece en el documento HTML.

flex-grow: Esto define la capacidad de crecimiento de un elemento flexible si es necesario. Acepta un valor sin unidades que sirve como proporción. Determina qué cantidad de espacio disponible dentro del contenedor flexible debe ocupar el artículo.

Si todos los elementos tienen el crecimiento flexible establecido en 1, el espacio restante en el contenedor se distribuirá por igual a todos los niños. Si uno de los hijos tiene un valor de 2, ese hijo ocuparía el doble de espacio que cualquiera de los otros (o lo intentará, al menos).



flex-shrink: Define cómo se va a encoger el elemento cuando se vaya ampliando o reduciendo el contenedor padre. Si establecemos el valor 0 no se encogerá nunca, el valor 1 es el valor por defecto y cuanto más mayor sea este valor más rápido se va a encoger el elemento. (Ver ejemplo de clase)

flex-basis: define el ancho base que va a tener el elemento hijo antes de que el navegador distribuya el espacio restante.

align-self: permite sobreescribir el alineamiento establecido con la regla ***align-items*** del contenedor padre.

Visibilidad de un elemento

Al igual que con la regla **display** tenemos otro modo de ocultar un elemento pero con una pequeña diferencia:

```
span{
   visibility: hidden;
}
```

Diferencia entre visibilidad y display

La diferencia entre estas dos reglas reside en que utilizando **display,** el navegador no mostrará el elemento y éste tampoco ocupara su espacio, es decir, el navegador actuará como si el elemento no existiera. En cambio, la regla **visibilidad** tampoco muestra el elemento pero si que ocupa su espacio.

Regla de posición

Especifica el tipo de posicionamiento que va a tener un elemento dentro del documento HTML. A continuación, podremos establecer la posición con las reglas **top, bottom, left, right,** aunque para poder usarlos se tiene que especificar la regla **position**.
Los posibles valores son: **static, relative, fixed, absolute, sticky**

Static

Es la manera por defecto y no le afectan las reglas **top, bottom, left y right.** Básicamente se posiciona de acuerdo con el flujo normal de la página.

Relative

El elemento es posicionado respecto a su posición normal. Establecer las propiedades **top, bottom, left, right** de un elemento relativamente posicionado hará que se ajuste fuera de su posición normal. El resto del contenido no se ajustará para encajar en ningún espacio dejado por el elemento.

Fixed

Un elemento con **position:fixed**; se coloca en relación con la ventana gráfica (viewport), lo que significa que siempre permanece en el mismo lugar, incluso si se desplaza la página. Las propiedades **top, bottom, left, right** se utilizan para colocar el elemento.

Un elemento fijo no deja un espacio en la página donde normalmente habría estado ubicado.

Absolute

Un elemento con **position: absolute;** se posiciona relativo a su antecesor más cercano.

Los elementos de posición absoluta se eliminan del flujo normal y pueden superponerse a los elementos.

Sticky

Un elemento con **position: sticky;** se posiciona en función de la posición de desplazamiento del usuario.

Un elemento fijo alterna entre **relative y fixed**, dependiendo de la posición de desplazamiento. Se coloca en relación hasta que se alcanza una posición de desplazamiento determinada en la ventana gráfica; luego, se "pega" en su lugar (como posición: fija).

NOTA: Para que funcione el **position: sticky;** se debe utilizar al menos una de las reglas: **top, bottom, left, right.**

La Propiedad Z-Index

Cuando los elementos están posicionados, es decir, se les ha aplicado una regla ***position***

La propiedad **z-index** especifica el orden de pila de un elemento (qué elemento debe colocarse delante o detrás de los demás).

A un elemento se le puede asignar un z-index tanto positivo como negativo.

El **z-index** solo funciona en elementos posicionados (position: absolute, position: relative, position: fixed o position: sticky) y elementos flex (elementos que son hijos directos de visualización: elementos flex).

NOTA: Si dos elementos posicionados no tienen asignada la propiedad z-index, el elemento de los dos que esté definido el último en el HTML se mostrará encima del primero.

La Propiedad Overflow

La propiedad CSS **overflow** define qué le pasa al contenido de un elemento cuando es demasiado grande para caber en el área asignada (desbordamiento).

Especifica si recortar el contenido o agregar barras de desplazamiento cuando el contenido de un elemento es demasiado grande para caber en el área especificada.

Los posibles valores son los siguientes:

- **visible:** El desbordamiento no se recorta. El contenido se muestra fuera de la caja del elemento.
- **hidden:** el desbordamiento se recorta. Todo el contenido que desborde de su elemento, se ocultará.
- **scroll:** el desbordamiento se recorta pero en este caso no se oculta. El navegador añade barras de *scroll*.
- **auto:** similar al anterior (scroll) pero en este caso las barras de scroll solo se añaden si son necesarias.

Propiedades **overflow-x - overflow-y**

Podemos únicamente capturar el desbordamiento en uno solo de los ejes de coordenadas. Con **overflow-x** capturaremos el desbordamiento en el eje horizontal, es decir a los lados izquierdo y derecho. Con **overflow-y** capturaremos el desbordamiento en el eje vertical, es decir, por arriba y por debajo.

La Propiedad Float

La propiedad float de CSS especifica cómo debe flotar un elemento. Se usa para posicionar y formatear contenido.

Puede tener estos valores:

- **left:** el elemento flota a la izquierda de su contenedor.
- **right:** el elemento flota a la derecha de su contenedor.
- **none:** el elemento no será flotante. Por defecto.
- **inherit:** toma el valor de float de su padre.

El ejemplo más usado es ajustar el **wrap** de un texto alrededor de una imagen.

Alineamiento Horizontal y Vertical

En esta sección veremos las distintas maneras de alinear contenido horizontal y verticalmente.

HORIZONTAL

Elementos alineados al centro

Para alinear elementos de tipo bloque utilizaremos **margin: auto;** de esta manera estamos aplicando el margen automático tanto arriba y abajo como a los lados. Para alinear horizontalmente bastaría con asignar el auto a los márgenes laterales. De tal forma podríamos establecer margenes arriba y abajo y seguiría alineado al centro de manera horizontal. Por ejemplo: **margin: 10px auto;**

Este tipo de alineamiento no funciona si no se aplica la regla **width** o si el elemento ocupa el 100% del contenedor.

Text alineado al centro

Como ya hemos visto, para alinear un elemento de texto al centro bastaría con aplicar la regla **text-align: center;**

Centrar imagen

Para centrar una imagen al centro bastaría con aplicar márgenes laterales a **auto** y hacer la imagen de tipo bloque (**display: block;**);

Alinear a la derecha o la izquierda con **position**

Una manera de alinear a la derecha o izquierda es utilizando el posicionamiento de tipo **absolute** (**position: absolute;**)

Una vez el elemento tiene posicionamiento absoluto podremos pegarlo a la izquierda usando **left: 0px;** o bien alinearlo a la derecha con **right: 0px;**

Alinear a la derecha o la izquierda con **float**

Como ya hemos visto en la sección anterior.

VERTICAL

Elementos alineados al centro usando **padding**

Es la solución más sencilla para alinear verticalmente un elemento al centro de su contenedor. Si tenemos un **div** que contiene un elemento **p** elemento queremos que este alineado verticalmente al centro, le sobreescribiremos los márgenes que tiene el elemento por defecto y posteriormente le asignaremos padding arriba y abajo lo que hará que se alinee en el centro.

```
<div>
        <p>Texto centrado verticalmente</p>
</div>

p{
    margin: 0;
    padding: 100px 0;
}
```

Si además le asignamos al **div** o al **p** la regla **text-align: center;** tendremos el texto alineado tanto en vertical como en horizontal.

Elementos alineados al centro usando **line-height**

Otra forma de alinear verticalmente es hacer uso de la propiedad line-height con un valor igual a la altura del contenedor donde se encuentra.

```
<div>
        <p>Texto centrado verticalmente</p>
</div>

div {
      height: 200px;
      line-height: 200px;
}
```

De esta manera, como la línea de texto ocupa 200px o lo que es lo mismo, la altura del contenedor, el texto se quedará alineado al centro verticalmente.

Elementos alineados al centro usando **position y transform**

Con estas dos reglas también se puede centrar de manera vertical y horizontal a la vez. Tenemos que establecer el **position** del contenedor a **relative,** y el **position** del texto a **absolute**. Una vez hecho esto, se desplaza el 50% y luego lo transformamos a -50%. El código para realizar esta alineación es el siguiente:

```
<div>
        <p>Texto centrado verticalmente</p>
</div>

.div {
    height: 200px;
    position: relative;
}

.center p {
    margin: 0;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
```

Elementos alineados al centro usando **display: flex**

Por último con las reglas de FlexBox podemos alinear elementos dentro de un elemento **flex** de la siguiente manera:

```
<div>
        <p>Texto centrado verticalmente</p>
</div>
```

```
div {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 200px;
}
```

CSS Combinators

Un selector CSS puede contener más de un selector simple. Entre los selectores simples, podemos incluir un combinador.

Hay cuatro tipos de combinadores en CSS:

- Selector descendente (espacio)
- Selector de hijo (>)
- Selector de hermanos adyacentes
- Selector general de hermanos

Selector descendente

Mediante este selector podemos aplicar estilos a todos los elementos que son hijos de un elemento.

Por ejemplo, si tenemos un **div** que contiene un **p**, podemos asignarle estilos al **p** de la siguiente manera:

```
div p {
 /* Reglas a aplicar al elemento p*/
}
```

Selector de hijo (>)

Seleccionaremos todos los elementos que sean hijos de otro. Por ejemplo, si tenemos un **div** que contiene un **p**, podemos asignarle estilos al **p** de la siguiente manera:

```
div > p {
  /* Reglas a aplicar al elemento p*/
}
```

Selector de hermanos adyacentes (+)

El selector de hermanos adyacente se usa para seleccionar un elemento que está directamente después de otro elemento específico.

El siguiente ejemplo selecciona el primer elemento <p> que se coloca inmediatamente después de los elementos

```
<div>
        <p>Texto centrado verticalmente</p>
        <p>Texto centrado verticalmente</p>
        <p>Texto centrado verticalmente</p>
</div>

div + p{
   /* Reglas a aplicar al elemento p*/
}
```

Transform

La propiedad de transformación le permite manipular visualmente un elemento torciendo, rotando, moviendo o escalando.

Scale

Esta opción se utiliza para redimensionar el elemento respecto a su tamaño original.

Si tiene un parámetro:

```
<div>
        <p>Hola Mundo!</p>
</div>

div{
    transform: scale(20);
}
```

Esta regla que se está aplicando hace que el elemento div se escale a 20 veces su tamaño inicial. Esto también afecta a los márgenes.

Si tiene dos parámetros:

```
<div>
        <p>Hola Mundo!</p>
</div>

div{
    transform: scale(2, 5);
}
```

Al usar dos parámetros lo que hace es escalar horizontalmente según el primer parámetro y verticalmente según el segundo. Por tanto, el div del ejemplo se estará escalando a el doble de manera vertical y se estará escalando cinco veces más grande horizontalmente.

También se puede escalar cada eje por separado utilizando:

```
transform: scaleX(2);
transform: scaleY(5);
```

Estas dos líneas dejarían el escalado igual que en el ejemplo con dos parámetros.

Skew

Las funciones de transformación skewX y skewY inclinan un elemento hacia un lado o hacia el otro.

.element {
transform: skewX(25deg); Inclina el eje X del elemento 25º
}

.element {
transform: skewY(25deg); Inclina el eje Y del elemento 25º
}

.element {
transform: skew(25deg, 25deg); Inclina el eje X y el eje Y 25º
}

Rotate

La función rotate sobre un elemento lo hace rotar en sentido horario el número de grados especificados.

.element {
transform: rotate(25deg); Rotará el elemento 25º
}

También se puede aplicar la rotación a un eje concreto con las siguientes funciones:
rotateX, rotateY, rotateZ

Translate

La función translate mueve los elementos arriba y abajo y a los lados, es decir, lo mueven en los ejes de coordenadas X e Y. Por ejemplo:

```
.element {
  transform: translate(25%, 25%);
}
```

En el primer parámetro, si especificamos un valor positivo lo movemos hacia a la derecha y si es negativo lo movemos a la izquierda.

En el segundo parámetro, si especificamos un valor positivo lo movemos hacia abajo y si es negativo lo movemos hacia arriba.

También podemos mover el elemento únicamente en un eje de coordenadas con las funciones: **translateX y translateY**

Animation

La regla animatino nos va a servir para otras muchas reglas CSS como **color, width, height, etc.**

Lo primero que debemos hacer es definir la animación, especificando los diferentes estados por los que atravesará. Para ello utilizaremos ***@keyframes:***

```
@keyframes <nombre> {
  0% {
    background-color: #001F3F;
  }
  100% {
    background-color: #FF4136;
  }
}
```

En esta definición estamos creando la animación **nombre** y definimos que el principio de la animación será tener el color de fondo de color #001F3F y el final de la animación será tener el color de fondo de color #FF4136.

NOTA: Este es el ejemplo más sencillo pero también podemos hacer que tenga más estados intermedios. Si por ejemplo queremos que la animación cambie cinco veces de color de fondo lo podríamos de la siguiente manera:

```
@keyframes animacion_color_fondo {
  0% {
    background-color: red;
  }
  0% {
    background-color: blue;
  }
  0% {
    background-color: yellow;
  }
  0% {
    background-color: green;
  }
  100% {
    background-color: pink;
  }
}
```

Una vez definida la animación se la aplicaríamos a un elemento de la siguiente manera:

```
div{
    animation: <nombre_animacion> <duracion> <iteracion de la animacion>;
}
```

Los 3 parámetros que le damos como valor son subpropiedades que podemos establecer una a una. Las subpropiedades de animation son:

- **animation-name: <valor>.** Se declara que el elemento va a utilizar la animación definida con el nombre <valor>
- **animation-duration: <valor>.** El tiempo que tarda la animación en hacer un ciclo completo.
- **animation-timing-function: <valor>.** Establece curvas de aceleración preestablecidas. Posibles valores: ease, ease-in, ease-out, ease-in-out, linear, step-start, step-end.
- **animation-delay: <valor>.** Establece el tiempo que transcurrirá entre que se carga el elemento en el navegador hasta que empieza la animación.
- **animation-direction: <valor>.** Establece la dirección en la que se reproducirá la animación, de adelante a atrás o al revés. Los posibles valores son: normal, reverse, alternate y alternate-reverse.
- **animation-iteration-count: <valor>.** El número de veces que se realizará la animación. Puede ser un valor numérico (1, 2, 3, 4, N) o el valor **infinite** que hará que la animación se reproduzca en bucle.
- **animation-play-state: <valor>.** Pausa o reproduce la animación.

SHORTHAND
Podemos aplicar todas las reglas de animación solo en una línea con la regla **animation** y el orden en el que se especifican las subpropiedades es el siguiente:

animation: <animation-name> <animation-duration> <animation-timing-function> <animation-delay> <animation-iteration-count> <animation-direction>

Como por ejemplo:

animation: nombre 5s ease-in 2s normal infinite.

MÚLTIPLES ANIMACIONES
En una misma línea de ***animation*** podemos declarar más de una animación a un elemento con el siguiente formato:

```
.element {
    animation:
    pulse 3s ease infinite alternate,
    nudge 5s linear infinite alternate;
}
```