



Clase 8

CSS Parte 4

CSS





Medidas



Las medidas en CSS se emplean para definir la altura, el ancho, los márgenes de los elementos y para establecer el tamaño de letra del texto. Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).

CSS divide las unidades de medida en dos grupos: **absolutas** y **relativas**.



Medidas



Absolutas: Las unidades absolutas son medidas fijas, su valor real es directamente el valor indicado que se ve igual en todos los dispositivos.

Relativas: Las medidas relativas definen su valor en relación con otra medida, por lo que para obtener su valor real, se debe realizar alguna operación con el valor indicado.

Flexibles: dentro de las medidas relativas están las flexibles que son relativas al tamaño del viewport.



Medidas Absolutas



La principal ventaja de las unidades absolutas es que su valor es directamente el valor que se debe utilizar, sin necesidad de realizar cálculos intermedios. Pero la desventaja es que son muy poco flexibles y no se adaptan fácilmente a los diferentes medios y por esto no suelen ser utilizadas.

De todas las medidas absolutas, la más utilizada es el pixel (px).



Medidas Absolutas



Una medida indicada mediante unidades absolutas está completamente definida, ya que su valor no depende de otro valor de referencia.

cm, centímetros.

mm, milímetros.

px, pixeles. Un pixel equivale a unos 0.26 milímetros.

pt, puntos. Un punto equivale a unos 0.35 milímetros.

in, pulgadas: Una pulgada equivale a 2.54 centímetros.

pc, picas. Una pica equivale a unos 4.23 milímetros.



Medidas Relativas



Las unidades relativas, a diferencia de las absolutas, es que no están completamente definidas, ya que su valor siempre está referenciado respecto a otro valor.

Son las más utilizadas por la flexibilidad con la que se adaptan a los diferentes medios.



Medidas Relativas



em: relativa respecto del tamaño de letra del elemento. Por defecto el tamaño de letra debería ser de 16px que equivaldrían a 1em pero, por ejemplo, si le daríamos un font-size de 10px al body, 1em equivaldría a 10px. Siempre va a variar dependiendo cual es el tamaño del elemento padre. 1.2em sería 20% más que el tamaño de su elemento padre.

rem(root em): es muy similar a em con la diferencia de que no es escalable, no depende del elemento padre sino del elemento raíz del documento o sea `<html>` (:root representa a la etiqueta html). Si el font-size del `<html>` es 16px, 1rem sería igual a 16px en cualquier parte del documento.



Medidas Flexibles



Las unidades flexibles, son todas relativas a las dimensiones tanto del ancho o alto del viewport en el que se visualice nuestra página, ya sea un dispositivo móvil o de escritorio.

vw: viewport width, esta medida es relativa al 100% del viewport. Lo que quiere decir que si decimos que un div debe medir 50vw, es equivalente al 50% del ancho total del viewport.

vh: viewport height, va a ser un porcentaje relativo a la altura total del viewport. Entonces, si definimos que un div mide 50vh y el alto del viewport es 800px, nuestro div medirá 400px.



Posicionamiento



Podemos alterar la posición de las cajas con la propiedad `position` que acepta los siguientes valores:

- ❖ **static:** es el valor por defecto, un elemento con este valor no está posicionado.
- ❖ **relative:** se comporta igual que `static` a menos que le agreguemos las propiedades: `top` | `bottom` | `right` y/o `left` y así causando un reajuste en su posición, otro elemento no podrá ajustarse a cualquier hueco dejado por este elemento.



Posicionamiento



- ❖ **absolute:** la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.
- ❖ **fixed:** hace que la caja este posicionada con respecto a la ventana del navegador, lo que significa que se mantendrá en el mismo lugar incluso al hacer scroll en la página.
- ❖ **sticky:** se posiciona según el estado de desplazamiento del usuario. Se "pega" en su lugar, después de alcanzar una posición de desplazamiento determinada.



z-index



En los casos en que haya elementos que queden superpuestos, podemos determinar el orden en que se "apilarán". La propiedad z-index indica el orden de un elemento posicionado, los elementos con mayor valor z-index van a cubrir a aquellos con menor valor.

z-index: auto | number | initial | inherit;



Display



Display es la propiedad más importante para controlar estructuras. Cada elemento tiene un valor de display por defecto, ya vimos los valores por defecto block e inline que los navegadores le dan a los elementos.

- ❖ **block:** un elemento block empieza en una nueva línea ya lo vimos en elementos como div, h1-h6, header, etc.
- ❖ **inline:** Un elemento inline puede contener algo de texto dentro de un párrafo sin interrumpir el flujo del párrafo.
- ❖ **none:** es utilizado para ocultar elementos sin eliminarlos, no deja un espacio donde el elemento se encontraba.
- ❖ **inline-block:** Los elementos inline-block fluyen con el texto y demás elementos como si fueran elementos en-línea y además respetan el ancho, el alto y los márgenes verticales.



Flex-box



Cuando trabajamos con flexbox primero hay que definir un contenedor flexible con la propiedad display y el valor flex, luego necesitamos pensar en términos de dos ejes: el eje principal y el eje cruzado. El principal está definido por la propiedad flex-direction y el eje cruzado es perpendicular a este.

flex-direction: define en que dirección el contenedor va a apilar los flex-ítems, posee cuatro valores: row | row-reverse | column | column-reverse





Flex-box



❖ **column:**



❖ **column-reverse:**





Flex-box

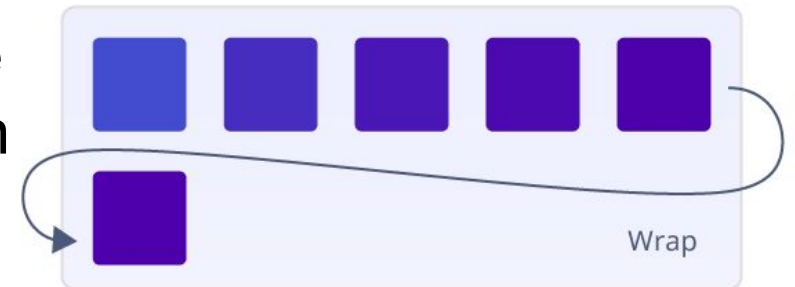


flex-wrap: especifica si los ítems son obligados a permanecer en la misma línea o si pueden fluir en varias.

❖ **nowrap:** los ítems van a ser distribuidos en una sola línea.



❖ **wrap:** los ítems van a ser colocados en varias líneas, y si llegan a sobrepasar el ancho de su contenedor se va a generar un quiebre para que los ítems se puedan seguir apilando en varias líneas.



❖ **wrap-reverse:** los ítems van a ser colocados en varias líneas pero en orden inverso.





Flex-box



flex-flow: establece las propiedades flex-direction y flex-wrap juntas.

flex-flow: <flex-direction> <flex-wrap>;



Flex-box



justify-content: se utiliza para alinear ítems.

- ❖ **flex-start:** es el valor por defecto, alinea los elementos al principio del contenedor
- ❖ **center:** alinea los elementos en el centro del contenedor.
- ❖ **flex-end:** alinea los elementos al final del contenedor.
- ❖ **space-between:** muestra a los elementos con espacio entre las líneas.
- ❖ **space-around:** muestra a los elementos con espacio antes, entre y después de las líneas.



Flex-box



align-items: se utiliza para alinear los ítems.

- ❖ **flex-start:** alinea los elementos en la parte superior del contenedor.
- ❖ **center:** alinea los elementos en el centro del contenedor.
- ❖ **flex-end:** alinea los elementos en la parte inferior del contenedor.
- ❖ **stretch:** estira los elementos en la parte inferior del contenedor.
- ❖ **baseline:** alinea los elementos como sus líneas de base se alinean.



Flex-start



Stretch



Flex-end



Center



baseline



Flex-box



align-content: ajusta las líneas dentro de un contenedor flex.

- ❖ **flex-start:** las líneas son ajustadas a partir del final del eje transversal. El borde transversal de inicio de la primera línea y el del contenedor flex quedan unidos.
- ❖ **flex-end:** las líneas son ajustadas a partir del final del eje transversal. El borde transversal final de la última línea y el del contenedor flex quedan unidos.
- ❖ **center:** las líneas son ajustadas hacia el centro del contenedor flex. Las líneas son unidas entre si, y centradas dentro del contenedor.



Flex-box



align-content:

- ❖ **space-between:** las líneas son distribuidas de manera uniforme en el contenedor flex. El espaciado entre cualquier par de elementos adyacentes es el mismo. Los bordes transversales de inicio y de fin del contenedor son unidos a los bordes de la primera y la última línea.
- ❖ **space-around:** las líneas son distribuidas uniformemente de modo que el espacio entre cualquier par de elementos adyacentes sea el mismo. El espacio vacío antes de la primera línea y el espacio después de la última es igual a la mitad del espacio entre los elementos.
- ❖ **stretch:** las líneas son estiradas para usar el espacio sobrante.



Juegos para aprender CSS



- ✓ <https://flukeout.github.io>
- ✓ <http://cssgridgarden.com>
- ✓ <http://www.flexboxdefense.com>
- ✓ <https://flexboxfroggy.com>
- ✓ <https://mastery.games/flexboxzombies>
- ✓ <https://cssbattle.dev>