**CCS**

**01-04-2025**

1. **Establece ver el curso en 3 partes:**

**BÁSICO, INTERMEDIO Y AVANZADO**

Explica que es CCS es un lenguaje de diseño el cual son hoja de estilo en cascada, donde las instrucciones de estilo se leen de arriba abajo

**Enlazar html con CSS**

**Explica como enlazar nuestro ccs con html a traves de la etiqueta**

**<link rel="stylesheet" href="index.css">**

**Donde el atributo href="" es la ruta de nuestro arhivo css, y rel="stylesheet" es el atributo de la relacion en este caso un archivo de estilo css**

**2. SINTAXIS EN CSS**

Utiliza selectores y propiedades

**selector**: es el elemento al que vamos aplicar estilos

**propiedad**: es lo que vamos a cambiar

**valor:** es el nuevo valor que le vamos a dar a la propiedad

sintaxis

**propiedad:valor;** se le denomina declaración

y todo el conjunto **selector + declaraciones** se le denomina regla ejemplo:

**selector{**

**propiedad:valor;**

**propiedad2:valor;**

**propiedad3:valor;}**

**3. CATEGORÍAS DE SELECTORES ENTRE ELLOS TENEMOS:**

**a) selectores simples:** están los selectores elementales y los selectores de atributos

**b) selectores compuestos:** selectores agrupados, selectores combinadores, y pseudoclases-pseudoelementos

**Selectores simples que son elementales como:**

El universal que se representa con un **"\*"** y esta llama a toda la página**, recomienda no abusar, solo usarlos en momentos concretos**

**su sintaxis**

**\*{ propiedad: valor; }**

**Selectores de tipo etiqueta, el cual llama a la etiqueta tal cual está en el codigo html**

**ejemplo:**

**h1 {**

**propiedad:valor;**

**}**

**4. SELECTORES DE ATRIBUTO ID="" Y CLASS""**

Donde el id del elemento [identificable, se](http://identificable.se/) llama desde css con un numeral **"#nombre-id"**

**seguido del nombre de id, cabe mencionar que no se recomienda para estilos porque es un identificador único.**

**#ejemplo{ }**

**por otro lado, está el selector de clases class"" el cual se llama desde ccs con un punto (".nombreclases")** seguido del nombre de la clase, este es el mejor para dar estilo porque puede tener varias clases y con un mismo estilo aplicar a esas clases

**Nota:** los nombre que se le colocan tanto al id como a la clases class, son sensible a la mayúscula (case sensitive) por eso se recomienda ponerlo todo en minúscula

**SINTAXIS:**

**.ejemplo{ }**

**5. OTROS TIPOS DE SELECTORES**

**A) Selectores de Atributos:** son aquellos elementos que se seleccionan según su atributo,llamándolos desde CSS encerrándolos entre corchetes.

**Ejemplos:**

**Nota: cada sintaxis varía dependiendo de cómo la vamos a usar en la página de html, SEGUN NUESTRO REQUERIMIENTO DE PROGRAMACION**

**OPCION 1.** llamando solo al atributo entre corchetes **[]**, es decir todas las etiquetas que tenga ese atributo se le aplicara el estilo, y este es la base de las demás opciones.

**sintaxis:**

**[atributo] { propiedad: valor;}**

**ejemplo:**

**[href] { background-color: red;}**

**OPCION 2.** llamando al atributo por su valor ="valor" entre corchetes. **Nota:** importante las comillas, porque si no, no funciona, al igual que el valor, tiene que ser el mismo porque si no, no lo encuentra.

**sintaxis**

**[atributo="valor"] { propiedad: valor;}**

**ejemplo**

**[href="**[**https://google.com**](https://google.com/)**"] { background-color: blue;}**

**OPCION 3.** Se aplica si el atributo empieza con ese valor, y se representa con circunflejo ^

**sintaxis**

**[atributo^="valor"] {propiedad: valor;}**

**ejemplo:**

**[href^="color"] {background-color: pink;}**

**En este caso, busca todo lo que empiece con la palabra color del atributo, se le aplicara el estilo seleccionado, porque funciona buscando la primera palabra que esta después de cada igualdad = del atributo**

**OPCION 4.** funciona parecido al anterior pero independientemente de la posición del valor, se representa con un asterisco **\*** en el atributo

**sintaxis**

**[atributo\*="valor"] {propiedad: valor;}**

**ejemplo:**

**[href\*="beisbol"] {background-color: yellow;}**

**nota: donde beisbol es el valor que va a buscar en el atributo**

**OPCION 5.** Se aplica si el atributo termina en la palabra del valor que colocamos. y se representa con el signo dólar **$**

**sintaxis**

**[atributo$="valor]" { propiedad: valor;}**

**ejemplo:**

**[href$="perro"] {background-color: purpul;}**

**En este caso, busca que todo lo que TERMINA con la palabra perro del atributo, se le aplicara el estilo seleccionado**

**OPCIÓN 5.** Los Estilo para los casos que tengan un guion por ejemplo en los casos especiales que sea necesario por tema de sintaxis en el lenguaje se le aplicara el estilo colocando una barra vertical **|**

**ejemplo**

**caso del atributo Lang (que es de selección del lenguaje)**

**[lang | = "en"] {background-color: purpul;}**

**6. SELECTORES AGRUPADOS**

permite agrupar clases en una misma regla de estilo, separándolos con una coma **","** esto permite una mejor optimización a la hora de procesar el código por los navegadores

**ejemplo:**

**sintaxis**

**.text-1, . text-2, .text-3, .text-4 {background-color. steelblue;}**

este caso se ve la agrupación de clases en una misma regla para dar estilo cuando se tienen estilos en común.

**7. SELECTORES COMPUESTOS COMBINADORES**

**COMBINADOR HIJO:** selecciona los elementos hijos del primer nivel que este asociado

**ejemplo:**

caso que tengamos varias clases con el mismo nombre, si queremos dar estilo a una clase directa si necesidad de cambiar el nombre podremos aplicar este tipo de selección, donde se llama primero al bloque que pertenece y luego se coloca la clase:

* Caso que contengan un **<div>** como contenedor

div .clase-dada {background-color: steelblue;}

**Nota:** si el elemento está dentro de otro elemento es Hijo, en cambio si el elemento esta al mismo nivel es hermano.

**COMBINADOR HERMANO SIGUIENTE:** son secciones de combinaciones al mismo nivel y se añade al seleccionador con un más **"+"**

**ejemplo:**

Sí se encuentra mismo el nivel, si tiene una clase se añade por clase, si no tienes una clase se añade por la etiqueta

en este caso tiene una clase:

.text-1 + .titulo-2 {background-color: yellow;}

con el signo de + solo se añade el hermano que este después--ojo

**OTRO CASO ES HERMANOS SIGUIENTES** **(PLURAL)**: se añade con una verbulilla o tilde **"~"** y toma todos los hermanos siguientes que estén al mismo nivel, es decir que no estén dentro de bloques.

**ejemplo:**

.text-1 ~ .titulo-2 {background-color: yellow; }

**nota: recomienda no bajar más de dos niveles**

**SELECTOR COMBINADOR HIJO DIRECTO:** en el caso que estén en un contenedor dentro de otro contenedor y queremos seleccionar solo los que estén dentro de la etiqueta contenedor, podremos hacerlo con el este seleccionador que se añade con el signo **">"**

**ejemplo:**

**estructura en html**

<div class="contenedor">

<p>hola com estas </p>

<div>

<p>me voy de viaje</p>

</div>

<p>regrese otra vez</p>

</dv>

**estructura en css**

**.contenedor > p** {background-color: tomate;}

en este caso se seleccionarán todos los que estén en la clase contenedor sin tomar el **<p>** interno que está dentro del **<div>,** si no se hiciera así se seleccionaría competo incluyendo el **<p>** que está en el **<div>** interno.

**8. FUNDAMENTOS DE CCS  
VIDEO 14** <https://www.youtube.com/watch?v=6fh7ZTj2t2k&list=PLROIqh_5RZeDbvISffzihyxzqJBt_z3-Z&index=15>

**COMO FUNCIONA CSS**

**Especificidad CSS =** Establece como de especifico es un selector para saber que estilo aplicar.

El cálculo se realiza con las siguientes formulas

**formular y reglas**

El navegador sigue como reglas el valor de la suma de los elementos

ejemplo:

**1. etiquetas y pseudo-elementos (0,0,1)**

**2. clases atributos y pseudo-clases (0,1,0)**

**3. ID (1,0,0)**

**4. estilos en linea (1,0,0,0)--> estos son los que ponemos directamente en HTML**

**5. !important --🡪 LE GANA A TODO**

**NOTA:** CUANDO APLICAMOS UN **!important** en la relga nos SOBRE ESCRIBE TODO Y POR ENDE NOS cargados o nos vale madre la especificidad, por eso no se debe aplicar, Y SE DEBE CONOCER COMO FUNCIONA CSS

**/\* PARA EVITAR ESO SIMPRE USEMOS LAS CLASES \*/**

**/\* CSS FUNCIONA EN FORMA DE CASCADA (ES DECIR EL ULTIMO SOBRE ESCRIBE AL PRIMERO) POR LO TANTO, SIEMPRE QUE LA ESPECIFICIDAD SOBRE EL ELEMENTO SEA LA MISMA, SE APLICARA LA PROPIDAD ESTABLECIDA.** \*/

**EJEMPLO**

.titulo{background-color: brown; }

.titulo{background-color: blueviolet; }

en este caso en ultimo sobre escribe al primero a pesar de tener la misma clase.

/\* **Nota:** tener en cuenta que si la especificidad no es la misma, la cascada no funciona por eso es recomendable usar clases siempre para que la cascada no se vea afectada\*/

Ejemplos de especificidad

**/\* 1. etiquetas y pseudo-elementos (0,0,1)\*/**

h1 {background-color: yellow;}

**/\* 2. Clases, atributos y pseudoclases (0,1,0) \*/**

.titulo{background-color: brown;}

**/\* 3. ID (1,0,0) \*/**

#titulo{background-color: blueviolet;}

**/\* 4. etiquetas más clase (0,1,1) \*/**

h1.titulo{ background-color: green;}

**/\* 5. etiquetas más id (1,1,1) \*/**

h1.titulo#titulo{background-color: coral;}

Recomienda la página CSS specificity graph generator, para saber si estamos realizando correctamente el CSS según las especificaciones

[icon](https://jonassebastianohlsson.com/specificity-graph/) <https://jonassebastianohlsson.com/specificity-graph/>

**9. HERENCIA DE UN ELEMENTO**

capacidad que tienen algunos elementos de heredar algunas propiedades de su contenedor (padre, abuelos etc)

además, existen herencias de etiquetas con propiedades por defectos establecidas en el lenguaje html para distinguir su uso, por ejemplo, las cabeceras, enlaces, entre otros.

Existen propiedades en ccs que se heredan como color y tamaño de la fuente: color, font-size se heredan

**.titulo{** color:blueviolet;    font-size: 20px;**}**

**cabe destacar que hay situaciones en que algunas elementos o etiquetas que no heredan propiedades por su tipo de función o estilo propio en html ejemplos los enlaces, para poder modificarlos se debe asignar una clase propia**

**.link{**color:pink;}

**nota:** existen situaciones donde si queremos que herede la herencia de la propiedad donde está contenida en el elemento usamos como valor llamado **inherit**

ejemplo:

En un párrafo que contenga un link y que a su vez el párrafo tenga un color de fuente y q**ueremos que se le aplique igual al link , que por defecto, no hereda, por lo tanto, utilizamos inherit** para que pueda heredar dichas propiedades

.parrafo1{

    font-size: 50px ;

  color: silver;

}

.link{

    color:inherit;

}

**nota2:** en la situación contraria a la anterior, cuando queremos que un elemento no herede una propiedad usamos un valor llamado **initial**;

ejemplo

Donde hay una lista **ul** con una **class="lista"** y queremos que uno de los items no herede el color, por lo tanto, le asignamos una class diferente **class="lista-extra"** con el valor **initial**

**.lista{**color:blue;}

**.lista-extra{**color:initial;}

**Herramientas de desarrollo en la inspección de la página web**

Esta herramienta es útil con el fin de ver errores y depurar nuestra línea de código, de esta manera darnos cuenta del porque no se están aplicando algunos estilos.

**nota:** en la parte de desarrollo verificar la etiqueta **style** y **computer**, seleccionando previamente el elemento que vamos a verificar

**10. NORMALIZE**

**VIDEO 17**

Explica la normalización de estilos de los diferentes navegadores, **utilizando una hoja de estilo de CSS para su reseteo (LLAMADA NORMALIZE)**, con esta incorporación de estilo de RESET, ¡podremos aplicar nuestros estilos y que en todos navegadores se vean igual!

**nota: debemos colocarlo por encima de nuestra hoja de estilo**

ejemplo:

<link rel="stylesheet" href="/CSS/estilos-por-defecto/normalize.css">

<link rel="stylesheet" href="/practicas\_ccs/practica1ccs.css">

**11. PREFIJOS DE PROPIETARIOS**

Cuando una propiedad nueva en CSS se empieza a implementar, los navegadores tienen que ser capaces de interpretarla y por eso se usan los prefijos de propietarios.

para que esto, se nos escriba automáticamente **utilizamos un programa llamado PREPOS.**

instalamos PREPROS y linkiamos el archivo que nos devuelve compilado-procesado PREPROS, y **es importante colocarlo en nuestro head**

**nota:** cabe destacar, que donde debemos escribir el código es en hoja de estilo es en css

<https://www.youtube.com/watch?v=d9cBmsODumI&list=PLROIqh_5RZeDbvISffzihyxzqJBt_z3-Z&index=19>

**video 18 ----ojo**

**12. BOX MODEL**

Explica lo importante que es distinguir en el espacio lo que ocupa nuestro contenido, todo está contenido en cajas y podemos modificarlas según nuestro diseño:

**las principales son:**

**contenido:** básicamente es un escrito, una imagen, un logo, algo que queremos introducir

**borde:** define el borde que ocupa el contenido

**relleno (padding):** define el relleno interno entre el contenido y el borde

**margenes(margin):** define el margen que ocupa en el espacio de la pantalla o contenedor, es decir el espacio externo fuera de los bordes

**ancho(width):** es la distancia que ocupa el borde en sentido horizontal

**height(alto):** es la distancia que ocupa el borde en sentido vertical

**COMO FUNCIONA ANCHO(WIDTH) Y HEIGHT(ALTO), CON ELEMENTOS EN BLOQUE Y EN LINEA**

**en bloque:** si se aplican las propiedades de **ancho y alto**

**en línea:** no se aplica la propiedad de ancho y alto porque está determinada por su contenido (las medidas solo se aplican a los elementos en bloque)

**nota:** ¡¡cabe destacar, que no todos los elementos se comportan igual!!

**Ojo**-🡪siempre adaptar PREPROS A NUESTRO FLUJO DE TRABAJO

**VIDEO 19**

[**https://www.youtube.com/watch?v=-ivUAB-aG2U&list=PLROIqh\_5RZeDbvISffzihyxzqJBt\_z3-Z&index=20**](https://www.youtube.com/watch?v=-ivUAB-aG2U&list=PLROIqh_5RZeDbvISffzihyxzqJBt_z3-Z&index=20)

**13. MARGENES**

**margin:** es la propiedad que nos permite general espacio entre los elementos

**es un shorthand (propiedad abreviada)** que controla los cuatro lados posibles a los que dar márgenes.

margin-top: margen superior

margin-right: margen derecho

margin-bottom: margen inferior

margin-left: margen izquierdo

admite hasta 4 valores que van en el orden de las agujas del reloj

4 valores  margin: top, right, bottom, left;

3 valores margin: top, left/right, bottom;

2 valores margin: top/bottom, left/right

1 valor margin: top/ right/ bottom/ left

**NOTA:** los elementos en línea solo tienes márgenes horizontales y los elementos en bloque tienen márgenes en los 4 lados, para que funcione tiene que tener un ancho declarado

**ejemplos:**

**.bloque{**

**background-color: rgb(38, 0, 255);**

**width:200px;**

**height:100px;**

**margin-left: auto;**

**margin-right: auto; }**

**.linea{**

**background-color: lightcoral;**

**color:inherit;**

**margin-left: 100px;}**

**LOS ERRORES MÁS COMUNES CON LOS MÁRGENES**

Donde menciona una mala práctica es usar el selector universal para quitar los márgenes y el relleno paddig:

**\*{margin:0; padding: 0;}**

recomienda no usar código que genere más código, y escribir con sentido lo que estamos haciendo, entendiendo cómo funcionan las propiedades y las reglas en CSS

**14. RELLENO (PADDING)**

**padding:** es la propiedad que nos permite generar espacios internos entre el borde y la caja

**es un shorthand (propiedad abreviada) que controla los 4 lados posibles:**

padding-top: relleno superior

padding-right: relleno derecho

padding-bottom: relleno inferior

padding-left: relleno izquierdo

**admite hasta 4 valores que van en el orden de las agujas del reloj**

**4 valores  padding: top, right, bottom, left;**

**3 valores padding: top, right/left, bottom;**

**2 valores padding: top/bottom, left/right**

**1 valor padding: top/ right/ bottom/ left**

**nota:** el padding no tenemos limitaciones cuando se aplican las medidas a los elementos en línea, funciona igual tanto como en bloque como en línea

**DIFERENCIA ENTRE MARGIN Y PADDING**

Con margin el elemento se desplaza y padding el elemento crece.

**utilización:** si quieres separar cajas de los elementos que lo rodean usas margin y si quieres aumentar el tamaño de la caja de los bordes de contenido usas padding.

**15. Borde**

Es la propiedad que nos permite modificar el borde de la caja

 también es un shorthand (propiedad abreviada) que agrupa 3 propiedades:

**border-width: ancho del borde**

        border-top-width

        border-right-width

        border-bottom-width

        border-left-width

**border-style: estilo del borde**

        border-top-style

        border-right-style

        border-bottom-style

        border-left-style

**border-color: color del borde**

        border-top-color

        border-right-color

        border-bottom-color

        border-left-color

**EJEMPLO:**

**HTML**

 <div class="contenedor">

        <p class="caja">SOY UNA CAJA</p>

    </div>

**CSS**

.contenedor{

    background-color: steelblue;

    width: 200px;

    height: 200px;

    margin-top: 20px;

    margin-left: 50px;

    padding: 50px 150px;

    border-style: outset;

    border-color:darkgreen;

    border-width:15px; }

.caja{

   background-color:red ;

   width: 200px;

   height: 100px;

   border: 10px groove springgreen}

**16. PROPIEDAD BOX-SIZING**

 Es la propiedad que nos permite controlar el cálculo que hace el navegador a la hora de modificar las propiedades content (contenido), padding (relleno) y border(borde).

  los 2 valores que podemos darle son:

**content-box =** valor por defecto, el cálculo que realiza va ser solo el contenido (content), también llamado tamaño del elemento

**border-box=** calcula el tamaño del elemento incluyendo el padding y el border

**nota:** content: es el contendo de la caja y suma a la hora de calcular el box-sizing

**nota:** para ver vi el cálculo usamos la consola de desarrollo en la etiqueta inspector

ejemplo de buenas prácticas es usar el selector universal para ajustar todas las cajas a una sola medida sumando border-box

**ejemplo:**

\*{**box-sizing= border-box**}

Con estocalcula el tamaño del elemento incluyendo el padding y el border, es decir, si el tamaño del elemento es width= 200px y height: 200px entonces con la propiedad border-box para llegar a los 200px desglosara esos 200px entre el número de padding asignado y el border asignado, y entre la suma del desglose da como resutado el total 200px

**17. PROPIEDAD BORDER RADIUS**

Permite redondear vértices de forma independiente.

**Es un shorthand que engloba 4 propiedades:**

border-top-left-radius -> Radio del borde superior izquierdo

        border-top-right-radius -> Radio del borde superior derecho

        border-bottom-right-radius -> Radio del borde inferior derecho

        border-bottom-left-radius -> Radio del borde inferior izquierdo

**Si sólo recibe un valor**

border-radius: 100px;

**APLICARA EL VALOR EN LOS 4 vértice DEL CUADRO**

**SI RECIBE 2**

VALORES APLICARA SOLO A LAS DOS CARAS OPUESTAS DEL VERTICE

**SI RECIBE 3 VALORES:**

border-radius: 50px 100px 200px;

**-> Círculo de 50px en el vértice superior izquierdo, de 100px en el superior derecho y el inferior izquierdo y de 200px en el inferior derecho.**

**CON 4 VALORES**

**SE LES APLICARA A LOS 4 VERTICES EN SENTIDO DE LAS AGUJAS DEL RELOJ**

**Ejemplo:**

**.box-radius{**

**background-color: darkorange;**

**width: 200px;**

**height: 200px;**

**margin: 30px 200px 30px;**

**border-top-left-radius: 10px;**

**border-top-right-radius: 10px;**

**border-bottom-right-radius: 15px;**

**border-bottom-left-radius: 15px;**

**/\* con border-radius se aplica a los 4 vertices  \*/**

**/\* border-radius: 30px; \*/**

**}**

**BORDER RADIUS ELIPTICO**

Funciona igual que la anterior con la diferencia que se separa los ejes de los radios dentro de los vértices, es decir se asignan dos valores para que reconozca que es una elipse y se separan los vértices con una barra**"/"**

ejemplo:

Al tener dos radios debemos agrupar los radios y separarlos por una barra donde los de la izquierda de la barra corresponden al eje Y, y los de la derecha de la barra corresponde al eje X **--OJO**

**border-radius: 25px / 50px;**

**Elipse de 25px en Y y 50px en X a cada vértice.**

**border-radius: 25px 50px / 50px 100px;**

**Elipse de 25px en Y y 50px en X a los vértices superior izquierdo e inferior derecho**

**y una elipse de 50px en Y y 100px en X en los vértices superior derecho e inferior izquierdo.**

**border-radius: 25px 50px 75px / 50px 100px 150px;**

**Elipse de 25px en Y y 50px en X al vértice superior izquierdo,**

**elipse de 50px en Y y 100px en X en los vértices superior derecho e inferior izquierdo**

**elipse de 75px en Y y de 150px en X al vértice inferior derecho.**

**border-radius: 25px 50px 75px 100px / 50px 100px 150px 200px;**

**Elipse de 25px en Y y 50px en X al vértice superior izquierdo,**

**una elipse de 50px en Y y 100px en X en el vértice superior derecho,**

**una elipse de 75px en Y y de 150px en X al vértice inferior derecho**

**una elipse de 100px en X y 200px en Y en el vértice inferior izquierdo.**

**Me gusta este comentario**

**18. DESBORDAMIENTO (PROPIEDAD OVERFLOW)**

Decimos que un contenido se desborda cuando la caja es más pequeña que el contenedor

**Overflow:** Es la propiedad que controla qué debe hacer la caja cuando su contenido se desborda del contenedor.

**Es un shorthand que engloba overflow-x y overflow-y**

Tiene cuatro posibles valores:

visible -> Es el valor por defecto, es decir que siempre sea visible

**hidden** -> El contenido que se desborde no se verá, queda oculto. Si aplicamos esta propiedad en un solo eje, el otro se pondrá automáticamente en el valor scroll.

**scroll** -> Aparecerán barras de scroll en el eje asignado (x, y o ambos)

**auto** -> Aparecerán barras de scroll en el caso de que hicieran falta.

ejemplo:

/\* desbordamiento propiedad overflow \*/

.desbordamiento{

background-color: white;

color: black;

width: 250px;

height:350px;

margin: auto;

border-radius: 50px;

**overflow: hidden;**

}

/\* el valor hidden mantiene oculto el desborde \*/

.desborde1{

    width: 250px;

    padding: 20px;

    background-color: lightcoral;

}

**19. EL COLAPSO DE MARGENES EN CSS**

El error que existe en CSS, es con los márgenes verticales, el cual se anidan unos con otros, por eso recomienda solo colocar márgenes inferiores para que no se aniden y evitar colapsos y código de más.

Se practica el colapsado de márgenes

en CSS

body{

    background-color: rgb(80, 118, 255);

    margin-top: 0; }

.caja1{

    margin-bottom: 30px; }

.cabecera{

    background-color: lightcoral;

    height: 100px;

    margin-bottom: 20px;

   /\*  CON ESAS 3 OPCIONES SE PUEDE CORREGIR DE PADRES E HIJOS \*/

    /\* overflow: hidden; \*/

    /\* padding-top: 0.1px; \*/

    /\* border-top: 0.1px solid lightcoral; \*/}

.caja1, .caja2{

    background-color: white;

    width: 200px;

    height: 200px;

    margin-left: auto;

    margin-right: auto;

    border: 2px solid brown; }

**20. PROPIEDAD DISPLAY Y COMO USARLA**

sirve para cambiar el contexto de los él elementos dentro del navegador

admite varios valores:

**none:** que hace que el elemento no se muestre, pero siga cargándose

**block:** hace que el elemento sea de bloque y funcione como un elemento de bloque

por ejemplo, aplicarlo en un elemento de línea para dar medidas

**inline:** hace que el elemento sea de línea y se comporta como un elemento de linea

**inline-block:** hace que el elemento sea de línea, pero admite medidas y márgenes verticales, en el caso de las imágenes no funciona si queremos cambiar de posición con la propiedad **text-align.**

**21. PROPIEDAD OUTLINE**

propiedad que nos permite dibujar un borde por fuera del modelo de caja sin que nos ocupe espacio y genere desplazamiento

Es un **shorthand** que engloba

**outline-width:** Controla el ancho del outline

**outline-style:** Controla el estilo del outline

**outline-color:** Controla el color del outline

**Tiene las mísmas propiedades y sintaxis que border, pero con algunas diferencias.**

* No ocupa espacio, ya que no forma parte del box-model
* No se puede redondear
* No se pueden controlar los lados de forma independiente

También **cuenta con la propiedad outline-offset,** que nos permite aumentar o disminuir la distancia del **outline** respecto a la caja a la que pertenece

**ejemplo en css**

.border, .outline{

    background-color: rgb(0, 255, 149);

    width: 150px;

    height: 100px;

    margin: 50px 50px;

    padding: 25px; }

.border{

    border: 2px solid lightcyan; }

.outline{

    outline: 5px dashed rgb(180, 157, 184);

    /\* con outline-offset controlamos la distanciacon respecto a la caja  puede estar por fuera o por dentro sin alterar el espacio\*/

    outline-offset: -15px ;

    border: 15px solid lightslategray; }

**22. PROPIEDAD TEXT-ALIGN**

Es la propiedad que nos permite **alinear horizontalmente el contenido** de un elemento de bloque siempre que el contenido NO TENGA ANCHO DECLARADO----- **> OJO**

**Acepta 4 posibles valores**

**text-align:left;** -> Alinea el contenido a la izquierda

**text-align:right;** -> Alinea el contenido a la derecha

**text-align:center;** -> Alinea el contenido al centro

**text-align:justify;** -> Alinea el contenido de forma justificada. No es recomendable usarlo.

ejemplo

**HTML**

<hr>

    <span><b>OUTLINE</b></span>

    <div class="border">BORDER</div>

    <div class="outline">OUTLINE</div>

    <div class="contendor-imagen">

      <img src="/asset/iconos/linkedin.webp" alt="numeros" class="imagen">

    </div>

    <a href="#" class="enlacecenter">ENLACE CENTRAR TEXT ALING</a>

**CSS**

/\* ALINIACION DE CONTENIDO EN UNA CAJA CON TEXT-ALIGN \*/

.border{

    border: 2px solid lightcyan;

    text-align: center;

    padding-top: 35px;

}

/\* ALINEACION DE CONTENIDO EN UN ELEMENTO EN LINEA \*/

.enlacecenter{

color: white;

display: block;

text-align: center;

margin-top: 10px;

}

/\* en el caso de que la imagen este dentro de un bloque contenedor, si se centra, porque centra el contenido del bloque en este caso la imagen  \*/

.contendor-imagen{

    text-align: center;

    background-color: aliceblue; }

/\*  en caso que la imagen este fuera de un elemento de bloque, la imagen es el mismo contenido, es decir, ya ocupa todo el ancho del contenido y ese caso no funciona text-align, porque ya es un elemento independiente y hay que darle margenes para alinearlo \*/

/\*  
 .imagen{

 margin: 20px 350px 20px;

background-color: aliceblue;

 }

**23. La propiedad box-shadow**

se creó para añadir efectos de sombra a nuestra caja.

En esencia lo que hace es crear un clon de la caja respetando la forma de su box-model (ancho, alto, redondez)

La sintaxis de box-shadow se puede escribir de distintas formas según lo que queramos conseguir.

 Los valores que le podemos poner son:

**offset-x ->** Desplazamiento en x (obligatorio)

**offset-y ->** Desplazamiento en y (obligatorio)

**blur-radius ->** Desenfoque de la sombra

**spread-radius ->** Expansión de la sombra

**color ->** El color de la sombra, si no lo especificamos lo heredará del elemento al que pertenece

**inset ->** Determina si la sombra será interior o exterior

ejemplo:

.caja{

    width: 250px;

    height: 150px;

    margin: 100px 150px 30px;

    background-color: rgb(63, 149, 154);

    text-align: center;

    padding-top: 60px;

    font-size: xx-large;

    border-radius: 20px;

    box-shadow: inset **5px 5px** 15px rgb(129, 36, 129), 5px 5px 15px red}

/\* podemos añadir cuantas sombras queramos separándolos por una coma, es importante el orden en los comandos ejemplo:

**box-shadow:** eje **X, Y**, blueradius, spread radius, color, inset\*/

**24. POSITION QUE ES? ¿Y CÓMO USARLO? VIDEO 34**

Explica lo importante de saberse la teoría para aplicar esta propiedad **POSITION** que consiste en la posición de los elementos.

Hay algunos conceptos que se deben conocer para entender **POSITION**

**Flujo de renderizado** **->** Por norma general los elementos se dibujan de izquierda a derecha y de arriba abajo. El punto 0,0 de los elementos, por norma general, es la esquina superior izquierda.

**Espacio reservado** **->** Es el espacio que tiene un elemento asignado en el navegador.

**Elemento posicionado ->** Esto significa que el elemento tiene la propiedad **POSITION** con un valor distinto de **"static"**, que es el valor que tiene esta propiedad por defecto.

**Stackin context ->** Contexto de apilamiento. Es el orden en el que se apilarán las cajas que se superponen, dentro del mismo contenedor.

**Al posicionar un elemento se habilitan 5 propiedades que podemos utilizar para mover los elementos en los 3 ejes.**

**top ->** El elemento se moverá desde la parte superior la distancia que le hayamos indicado.

**right ->** El elemento se moverá desde la parte derecha la distancia que le hayamos indicado.

**bottom ->** El elemento se moverá desde la parte inferior la distancia que le hayamos indicado.

**left ->** El elemento se moverá desde la parte izquierda la distancia que le hayamos indicado.

**z-index ->** Nos permite mover el elemento en el contexto de apilamiento (eje z)

**NOTA:** Si a un elemento le declaramos la propiedad top y/o left, las propiedades bottom y/o right no funcionarán, porque hay que asignarle un posicionamiento diferente al que tiene por defecto static.

Los posibles valores que le podemos dar a position son.

**Static ->** Es el valor que tiene por defecto un elemento, con este valor el elemento NO ESTÁ POSICIONADO y por lo cual no podremos moverlo

**Relative ->** El elemento mantendrá su posición y medidas en el flujo de renderizado y mantendrá su espacio reservado. Si lo movemos lo hará usando su posición en el html como punto de referencia.

**Absolute ->** El elemento perderá sus medidas y su espacio reservado. Si lo movemos usará el elemento posicionado contenedor como referencia. Si no tiene ninguno, usará el elemento html de referencia.

**Fixed ->** El elemento perderá sus medidas y su espacio reservado.

Si lo movemos usará el elemento html de referencia, y además se quedará fijo en esa posición, aunque hagamos scroll.

**Sticky ->** Es una mezcla de position relative y "fixed".

**Nota:** Con este tipo de posicionamiento **los valores top, left, bottom y right** no sirven para mover el elemento, si no para indicarle en qué punto pasará a tener un comportamiento de posicionamiento similar a fixed, hasta llegar a ese punto se comportará como si no tuviera posicionamiento, aunque sí contará como posicionado si necesitáramos colocar otro elemento respecto a él (absolute)

**z-index:** Es la propiedad que nos permite ordenar los elementos superpuestos para controlar cual se coloca por delante y cual por detrás.

Me gusta este comentario

**25. Cómo se mueve los elementos con la propiedad POSITION RELATIVE**

**En este caso usa la posición relativa (position: relative;)**

usa como referencia para moverse su posición en HTML a partir de su punto renderizado, conserva sus medidas y no perderá su espacio reservado en el navegador a pesar de desplazarse

ejemplo en css

    .caja-1, .caja-2{

        width: 200px;

        height: 200px;

      }

    .caja-1{

        background-color: lightcoral;

        position: relative;

        border-radius: 100px;

        top: 20px;

        left: 0px;

        /\* left: 200px;

        top: 200px;

     \*/

    }

    .caja-2{

        background-color: blueviolet;

        position: relative;

        left: 100px;

        bottom: 0px;

    }

**nota:** cabe destacar que si existe top no existe bottom y viceversa, porque se mueve en una sola posición vertical.

así mismo, si existe left no existe right y viceversa, hay que darle valor en una sola posición, y los elementos siempre se comportan como cajas en el navegador a pesar de tener otras formas--- **> OJO**

**26. EXPLICA POSICIÓN ABSOLUTA (ABSOLUTE)**

En este caso funciona contrario a la posición relativa (relative), pierde sus medidas, pierde su espacio reservado u original, si se encuentra dentro de un elemento de bloque lo usa como referencia para moverse a partir de su punto renderizado **(siempre y cuando el elemento contenedor este declarada su posición, si no, no tendrá ningún efecto--ojo)**, si no lo tiene, usa de referencia la posición en HTML

**nota:** cabe destacar, que una vez que hemos posicionado un elemento dentro de otro (caso un elemento dentro de un elemento de bloque), ¡si el bloque se mueve todo lo que este dentro de él se moverá!

**ejemplo:**

**ejercicio en CSS**

    .caja {

        width: 200px;

        height: 200px;

        text-align: center; }

    .contenedor{

        width:400px;

        height:400px;

        background-color: steelblue;

        position: relative;

        margin: 50px 250px ; }

    .caja-1{

        background-color: lightcoral;

        position: absolute;

        left:0px;

        top: 400px;

        /\* position: relative;

        border-radius: 100px;

        bottom: 0;

        left: 0; \*/

        /\* left: 200px;

        top: 200px;

     \*/

    }

    .caja-2{

        background-color: blueviolet;

        position: absolute;

        left: 0px;

        top: 0px;

        /\* bottom: 0px; \*/

        /\* bottom: 0;s

        right: 0; \*/

        /\* position: relative;

        left: 0;

        bottom: 0; \*/

    }

**NOTA:** existe un pequeño truco con los elementos en línea para darles medidas sin usar **display=block**, con la posición absoluta generamos que un elemento de línea actúe como bloque y podamos darle medidas --- >**ojo**

**27. EXPLICA POSITION FIXED**

funciona igual que position absolute, perdiendo su espacio reservado y sus medidas, con la diferencia es que siempre su punto de referencia para moverse, va ser HTML, independientemente si está en un contenedor, además, se quedara fijo en su posición aunque tenga desplazamiento (scroll up o scroll down) en la página web.

**Ejemplo**

ejercicio en css con un menú en la parte superior

\*{

    box-sizing: border-box;

    margin: 0;

}

main{

    online: 50px;

}

header, nav{

    background-color: crimson;

    color: white;

    text-align: center;

    font-size: x-large;

    height: 50px;

    position: fixed;

    width: 100%;

  }

[href]{

    color: white;

}

**28. POSITION STICKY**

¿Como se usa?

Con este tipo de posicionamiento los valores top, left, bottom y right no sirven para mover el elemento, si no para indicarle en qué punto pasará a tener un comportamiento de posicionamiento fijado (fixed), hasta llegar a ese punto se comportará como si no tuviera posicionamiento, aunque sí contará como posicionado si necesitáramos colocar otro elemento respecto a él (absolute)

**¿porque no funciona?**

hay que tener en cuenta, que elemento se quedara fijado, siempre y cuando tenga un alto su contenedor, porque lo toma como referencia, si no lo tiene no se quedara fijado-🡪ojo

**ejemplo:**

ejercicio CSS

header, nav{

    background-color: crimson;

    text-align: center;

    font-size: x-large;

    /\* height: 50px; \*/

    /\* position: sticky; \*/

    /\* width: 100%; \*/

    /\* top: 0;

    \*/

}

[href]{

    color: white;

    /\* position: sticky;

    top: 0; \*/

  }

h2{

    position: sticky;

    top: 0;

    background-color: cadetblue;

}

**nota: si hay un overflow oculto declarada no funcionara position sticky  
  
29.** **Propiedad Z-INDEX**

es la propiedad que sirve para modificar el orden de apilamiento unos sobre otro, dando prioridad por encima del que queremos apilar!

**nota:** no usar números consecutivos como recomendación, para dar más holgura en momentos que necesitemos apilar más elementos.

**NORMA GENERAL**

por norma general el orden que prevale en el apilamiento es el que está puesto en HTML (es decir, de arriba hacia abajo) si no está declarada su posición.

En cambio, si los elementos tienen declarada su posición, ¡¡el orden que prevalece es de abajo hacia arriba, se superponen!!

otro caso es, que, si un elemento tiene declarado su posición y otro no, el que se superpone es el que está declarado.

cabe destacar que cuando se trata de elementos padre e hijos, el padre **JAMAS** se colocara por encima del hijo, lo que sí se puede colocar, es el elemento hijo por detrás del padre con la propiedad **Z-INDEX** y un valor negativo, pero nunca se debe asignar un valor Z-INDEX al padre porque si no, no funciona

**ejemplo**

**ejercicio en CSS**

    .caja {

        width: 200px;

        height: 200px;

        text-align: center;

      }

    .contenedor{

        width:400px;

        height:400px;

        background-color: steelblue;

        position: relative;

        margin: 50px 20px 20px 250px;

    }

    .caja-1{

        background-color: lightcoral;

        position: relative;

        border-radius: 100px;

        margin: 20px 30px 20px 350px;

     }

    .caja-2{

        background-color: blueviolet;

        position: relative;

        top: 200px;

        border-radius: 100px;

        margin-left: 350px;

        z-index: -20;

      }

    .caja-3{

        background-color: rgb(57, 167, 61);

        position: relative;

        z-index: -10;

        right: 150px;

        top: -200px;

    }

**30. EXPLICA EL CONTEXTO DE APILAMIENTO**

¿cómo funciona?

**El stacking context** o el contexto de apilamiento es el espacio donde nuestros elementos se van a ir apilando para que unos queden por detrás y otros por delante.

**El orden del stacking context es:** (de delante a atrás):

**1.** Elementos posicionados con un z-index de 1 ó más

**2.** Elementos posicionados sin z-index declarado (o z-index:auto)

**3.** Elementos no posicionados

**4.** Elementos con z-index negativo

**31. COMO ODENAR LAS PROPIEDADES**

explica que ordenar las propiedades es importante para efectos de eficiencia en cuanto a modificaciones, no existe ninguna norma ni especificación sobre cómo hacerlo, pero la mayoría de expertos coinciden en un mismo punto a ordenar.

EJEMPLO

**1 - Propiedades de posicionamiento**

**2 - Propiedades del box model**

**3 - Propiedades de texto**

**4 - Propiedades visuales (colores, bordes, background...)**

**5 - El resto**

.contenedor {

**/\*1. Position \*/**

position: relative;

top: 0;

left: 0;

**/\*2. Box model \*/**

display: block;

width: 300px;

height: 600px;

padding: 10px;

margin: 10px;

overflow: hidden;

**/\* 3. Text \*/**

font-size: 16px;

text-align: center;

**/\*4. Visual \*/**

color: blue;

border: 2px solid red;

border-radius: 10px;

**/\* 5. otros \*/**

opacity: 1;

}

siguiendo este orden recomendando podemos orientarnos mejor

**32. MEDIDAS ABSOLUTAS Y RELATIVAS**

Introducción

Las medidas en CSS se pueden agrupar en 2 grupos, absolutas y relativas.

**Las medidas absolutas** **NO CAMBIAN** y siempre mantendrán su tamaño independientemente del dispositivo o la interacción del usuario.

La más usada es px (píxeles) cuya medida es 1/96 de 1 in, pero existen otras como:

cm(centímetros) -> 1cm = 96px

mm(milímetros) -> 1mm = 1/10 de 1cm

Q(cuarto de de milímetro) -> 1Q = 1/40 de 1cm

in(inches, pulgadas) -> 1in = 2,54cm = 96px

pc(picas) -> 1pc = 1/16 de 1in

pt(puntos) -> 1pt = 1/72 de 1in

**Las medidas relativas** **SIEMPRE** dependen de un contexto son las que se recomienda utilizar en la gran mayoría de ocasiones porque nos van a permitir que todos los elementos escalen proporcionalmente.

Las más usadas son **em, rem, %, vw, vh, vmin y vmax** aunque tenemos otras como:

ex -> Altura x de la fuente del elemento.

ch -> La medida del ancho del glifo "0" de la letra del elemento.

lh -> Altura de la línea del elemento.

**Unidades de MEDIDA EM y REM**

**rem** **->** representa a la medida de la letra "m" de la raiz del documento (html).

ejemplo: m= 16px = 1rem =1em

nota: em siempre es variable en función de contexto a utilizar

**em** **->** representa a la medida de la letra "m" del contexto donde nos encontremos. es decir, si la medida que utilicemos tiene que crecer proporcionalmente al tamaño del contenido, utilizamos **em** y si la medida tiene que ser siempre la misma relativa al tamaño de fuente usamos **rem**

**EJEMPLO:**

.link {

background-color: lightcoral;

color: silver;

/\* en este caso queremos que el elemento crezca en función del contenido que sería el tamaño de fuente para ello utilizamos **em** \*/

padding: 1em 2em;

border-radius: 0.5em;

/\* es este caso usamos **rem** porque queremos que la caja se ajuste a este tamaño de fuente determinado\*/

font-size: 2rem;}

El estándar del tamaño de fuente por defecto en la raíz del documento es 16px, y NO ES RECOMENDABLE MODIFICARLO**-- > OJO**

**NOTA:** Tanto "em" como "rem" se calcula en base a la propiedad font-size (tamaño de fuente)

**33. PORCENTAJES EN WIDTH Y AUTO**

WIDTH:

**% ->** Cuando usamos porcentaje usamos de referencia el tamaño del contenedor y el navegador calcula ese porcentaje.

**auto ->** Cuando usamos auto le pedimos al navegador que calcule el ancho en función del espacio disponible.

**nota:** al dar márgenes tomar en cuenta los desplazamientos en automático y los porcentajes dados para que no se descuadren.

**34. Porcentajes en HEIGHT Y AUTO**

**% ->** Cuando estamos dentro de un contenedor, este tiene que tener un alto declarado, si no no podrá calcular el porcentaje.

**auto ->** Cuando utilizamos auto el alto lo calculará el navegador en base al contenido del elemento. Si se le aplica una medida con porcentaje se sustituirá automáticamente por auto.

**NOTA:** Height es una propiedad que hay que usar con mucho cuidado si no es necesario establecer un alto. como recomendación dejar que el contenido sea el que decida el alto del elemento.

**NOTA2:** CUANDO USAMOS POSITION ABSOLUTE EL HEIGHT EN PORCENTAJE SE VA CALCULAR AUTOMATICAMENTE EN BASE AL VIEWPORT

**35. VIEWPORT**

El viewport es el área útil donde se mostrará la página web.

**IMPORTANTE, hay que tener en cuenta que es el area visible sin hacer scroll.**

**Podemos usar el viewport como medida de varias formas:**

vw-> Viewport Width, hace referencia al ancho del viewport.

vh -> Viewport Height, hace referencia al alto del viewport.

vmax -> Utiliza el valor más grande entre el ancho y el alto del viewport.

vmin -> Utiliza el valor más pequeño entre el ancho y el alto del viewport.

**1vw = 1% del ancho del viewport**

**1vh = 1% del alto del viewport**

**1vmin = 1% del valor que sea más pequeño del viewport**

**1vmax = 1% del valor que sea más grande del viewport**

**Nota:** con **vmin** siempre va calcular la medida minima del viewport(ancho-ato)

y con vmax siempre va calcular la medida maxima del viewport(ancho-ato), en funcion de eso se moveran!

**ejemplo:**

es una medida relativa que va a respetar la medida en función del tamaño del viewport

.caja-1{

    background-color: rgb(117, 117, 179);

    width: 50vw;

    height: 200px;

    text-align: center;}

.caja-2{

    background-color: chocolate;

    width: 60vmax;

    height: 100px;

    text-align: center;}

**Limitadores al ancho y alto**

Cuando establecemos un width y/o un height con una medida que es relativa, es posible que no queramos que se respete esa medida en todos los casos.

Para ello tenemos propiedades que limitan esas medidas:

**min-width:** Indica el ancho minimo que puede alcanzar el elemento.

**max-width:** Indica el ancho máximo que puede alcanzar el elemento.

**min-height:** Indica el alto minimo que puede alcanzar el elemento.

**max-height:** Indica el alto máximo que puede alcanzar el elemento.

Nos permitirá a poner limite en los elementos, para adaptarlos a las diferentes pantallas

a la hora deponer un elemento como leyenda en la parte inferior nos ayuda bastante delimitar esas medidas un ejemplo de ello es el footer

ejemplo:

body{

    background-color: cadetblue;

    color: white;

    position: relative;

**min-height: 100vh;**

    margin: 0;

    border: 2px solid red;}

footer{

    position: absolute;

    bottom: 0;

    width: 100%;

    padding: 0.5rem 0;

    background-color: chartreuse;

    color:initial;

    text-align: center;}

**35. Colores**

La mayoría las pantallas funcionan con un modo de color aditivo, es decir que se crea de colores mezclados. La suma de RGB  (red, green, blue) da como resultado un blanco

Tenemos varias formas de dar color en CSS

en el teclado: Keywords -> las Palabras clave que representan colores son el siguiente link:

<https://www.w3.org/wiki/CSS/Properties/color/keywords>

**IMPORTANTE:** Recordad que existe el color **"transparent"**

**CurrentColor**

Es una palabra clave que se usa para tomar un valor de color adyacente en lugar de colocar el mismo color siempe, es decir usará el valor ya sea establecido o heredado **en la propiedad color**.

ejemplo:

body {

background-color: mediumorchid;

color: aquamarine;

}

.main {background-color: currentColor;}

en este caso toma el color heredado del anterior y a si sucesivamente hasta que encuentre uno declarado en la propiedad color.

**RGB/RGBA**

En el modo RGB tenemos tres canales **(Red, Green, Blue)** y tenemos 8 bits de color por canal, (cada bit tiene dos posibles valores 0 ó 1) lo que significa que tenemos 2^8 posibles valores, que van desde 0 hasta 255.

Esto nos da un total de 255 \* 255 \* 255 = 16.581.375 colores.

Su sintaxis es rgb(R,G,B)

**Si ponemos todos los canales a 0 tendremos un negro puro y si los ponemos a 255 tendremos un blanco.**

**Para movernos por la gama de grises debemos poner los 3 canales con el mismo valor, si ponemos los 3 canales a 128 tendremos el gris puro.**

El modo RGB nos da la opción de utilizar un cuarto canal que correspondería al canal alpha, es decir, la transparencia, y su valor va desde 0 a 1. Tenemos dos sintaxis para utilizar el canal aplha

**rgba(R,G,B,A)**

**rgb(R G B / A)**

estas dos sintaxis son iguales y validas separarlo por comas o por espacios y barra

**ejemplo:**

background-color: rgba(0, 0, 0, 0.1);

background-color: rgb(0 0 0 / 0.3);

**Hexadecimal**

La anotación hexadecimal es la más común en desarrollo web, se basa en los mismos principios que el **RGB** pero escrito en notación hexadecimal.

El sistema hexadecimal es un sistema basado en 16 valores del 0 al 15

Utiliza los dígitos del 0 al 9 y las letras de la A a la F

En este tipo de notacíon también necesitamos 256 valores.

Se compone de 16 caracteres en parejas ya que 16\*16 = 256

**0 1 2 3 4 5 6 7 8 9 A B C D E F**

**0 1 2 3 4 5 6 7 8 9 A B C D E F**

Para indicar que vamos a usar anotación hexadecimal debemos usar el símbolo de hash/almohadilla # y a continuación usaremos esta notación por parejas.

#RRGGBB

Cuando una pareja usa el mismo valor se puede omitir el segundo valor, siempre que se haga en las 3 parejas.

**rgb(255,255,255) -> #FFFFFF -> #FFF**

**rgb(0,0,0) -> #000000 -> #000**

También tenemos la opción de usar transparencias en hexadecimal añadiendo un cuarto canal que sigue las mismas normas.

**rgb(0 0 0 / .5) -> #00000080 ~> #0007**

**HSL / HSLA**

El modo de color HSL es el más intuitivo para los humanos. Su nombre viene de las siglas Hue (tono) Saturation (saturación) y Lightness (luminancia)

El primer valor es el ángulo en el círculo cromático donde 0 y 360 será el color rojo. Hay ciertos valores que os pueden servir como referencia para relacionar mejor los colores con sus ángulos.

**0 -> rojo**

**60 -> amarillo**

**120 -> verde**

**180 -> cyan**

**240 -> azul**

**300 -> magenta**

El segundo valor es la saturación o intensidad del color.

0% -> gris

100% -> color puro

El tercer valor es la luminosidad del color

0% -> negro, nada de luz

100% -> blanco, luz máxima

**Nota:** Es importante que aunque el valor de saturación o luminancia sea 0 hay que poner 0% si no, no funcionará

El modo HSL también admite transparencia y su sintaxis es la misma que la del modo RGB. Tenemos dos sintaxis disponible.

**hsla(h,s,l,a)**

**hsl(h s l / a)**

En ambos casos el valor alpha va de 0 a 1

Alguna paginas de colores hsl son:

<https://brandcolors.net/>

<https://color.adobe.com/es/create/color-wheel>

**36. Fuentes y tipografía**

es la ciencia que estudia el diseño y uso de los tipos de letra

Las fuentes son tipos específicos de letra

Los glifos son cada uno de los dibujos que representan cada símbolo de la fuente, incluidos los espacios.

Los estilos de fuente son las modificaciones que se hacen a las fuentes, como el grosor o la inclinación.

Familia tipográfica, son las diferentes fuentes que pertenecen a una misma fuente base.

Cuando usamos una propiedad text-\* estamos afectando a un bloque completo

Cuando utilizamos una propiedad font-\* afectamos a la fuente del texto.

Fuentes predeterminadas, son las fuentes que se consideran seguras ya que son genéricas y no establecen una familia tipográfica, si no un estilo de fuente.

serif: Fuentes que tienen serifas (adornos en los caracteres)

**sans-serif: Fuentes sin serifas**

**monospace:** Fuentes que tienen el mismo espaciado entre todos sus caracteres.

**cursive:** Fuentes que pretenden emular escritura a mano, con ligaduras y trazos fluidos

**fantasy:** Fuentes que suelen ser decorativas

**37. Fuentes Externas**

**GOOGLE FONTS CON CSS**

Fuentes desde un servidor remoto, son fuentes que se encuentran en un servidor y que nosotros descargamos para su uso, la más famosa es Google Fonts

**Siempre que utilicemos fuentes no genéricas, debemos poner fuentes genéricas de fallback (respaldo)**

**NOTA:** PARA TRAERNOS UNA FUENTE SIMPLEMENTE LA SELECCIONAMOS, COPIAMOS SU LINK Y LA COLOCAMOS EN EL HEAD HTML POR ENCIMA DEL CSS PARA QUE CARGE PRIMERO.

**ejemplo:**

en html head:

**<link rel="preconnect" href="**[**https://fonts.googleapis.com**](https://fonts.googleapis.com/)**">**

**<link rel="preconnect" href="**[**https://fonts.gstatic.com**](https://fonts.gstatic.com/)**" crossorigin>**

**<link href="**[**https://fonts.googleapis.com/css2?family=Roboto+Slab&family=Roboto:ital,wght@0,100..900;1,100..900&display=swap**](https://fonts.googleapis.com/css2?family=Roboto+Slab&family=Roboto:ital,wght@0,100..900;1,100..900&display=swap)**" rel="stylesheet">**

**<link rel="stylesheet" href="/CSS/box-model/viewport.css">**

**en css**

body{ font-family: "Roboto Slab", serif;}

**Nota: puedes poner después de la coma, fuentes alternas en caso de que la primera fuente no cargue (no hay límites para poner fuentes alternas)**

**38. Cómo Usar Librerías De ICONOS**

LOS ICONOS DE SERVIDORES REMOTOS PODEMOS USARLOS IMPORTANDO SU LINK Y COPIANDO EN EL HEAD AL IGUAL QUE LAS FUENTES, IMPORTANTE SABER QUE CUANDO SE TRABAJA CON SERVIDORES REMOTOS PODEMOS MODIFICAR EL COLOR Y EL TAMAÑO

Google Icons

Material Icons

[Googleapisfonts.googleapis.com/css?family=Material+Icons](https://fonts.googleapis.com/css?family=Material+Icons)

Material Icons Outlined

[Googleapisfonts.googleapis.com/css?family=Material+Icons+Outlined](https://fonts.googleapis.com/css?family=Material+Icons+Outlined)

Material Icons Two Tone

[Googleapisfonts.googleapis.com/css?family=Material+Icons+Two+Tone](https://fonts.googleapis.com/css?family=Material+Icons+Two+Tone)

Material Icons Rounded

[Googleapisfonts.googleapis.com/css?family=Material+Icons+Round](https://fonts.googleapis.com/css?family=Material+Icons+Round)

Material Icons Sharp

[Googleapisfonts.googleapis.com/css?family=Material+Icons+Sharp](https://fonts.googleapis.com/css?family=Material+Icons+Sharp)

**Nota:** Si vas a usar una gran cantidad de iconos en tu web compensa usar la librería remota

Si sólo usarás unos pocos es mejor descargarlos en formato svg y utilizarlos como imágenes

**38. Formatos para fuentes**

**True Type Font (TTF) ->** Fue el primer formato estándar.

**Open True Type Font (OTF) ->** Facilita o implementa la creación de las curvas.

**Web Open Font Format (WOFF) (WOFF2) ->** Es un formato mucho más ligero y mejora el rendimiento. Funciona a través de metadatos. WOFF tiene soporte total y WOFF2 no funciona en internet explorer.

**Scalable Vector Graphics (SVG) ->** Para efectos y animaciones sobre el texto, no es recomendable usarlo en el texto genérico de la web, tiene muchos problemas.

**Embedded Open Type (EOT) ->** Primer formato digital propuesto para sitios WEB y es comprimido. Se quedó como una propuesta pero no pasó a ser un estándar.

**para descargar fuentes en necesario conocer la siguientes reglas:**

**Regla @font-face**

Esta es la forma de utilizar fuentes que tenemos descargadas y que no traemos de un servidor remoto, cuando usamos google fonts, esto es lo que está pasando por detrás.

**La sintaxis es:**

@font-face {

**font-family:** Nombre de la familia, se lo asignamos nosotros;

src: aquí pondremos dos tipos de valores.

**local() ->** pondremos el nombre de la fuente para buscarla en el ordenador del usuario, de esa forma podremos evitar descargarla

**url() ->** Si no encontramos esa fuente procederemos a descargarla.

Existe otro valor, menos usado que es format

**format() ->** Este valor lo pasaremos de forma opcional si queremos ser más específicos con la fuente que vamos a buscar o descargar, si no lo ponemos cogerá el primer formato soportado, si lo ponemos sólo usará el que hemos establecido, y si no lo encuentra no hará nada.

}

Podemos añadirle más valores como **font-style, font-weight y font-display(experimental)**, pero esos son opcionales

paginas donde podremos descargar fuentes:

<https://www.1001fonts.com/>

<https://www.dafont.com/es/>

tomar encuenta las gratuitas

tamben existe covertidores de fuentes

<https://convertio.co/es/ttf-woff/>

**ejercicio:**

@font-face { font-family: 'poppins';

    src: local('Poppins-Medium'), url(/asset/fuentes/Poppins-Medium.ttf);}

**39. Transformaciones de texto**

**Text-transform:** Esta propiedad nos permite hacer transformaciones ESTÉTICAS en nuestro texto.

**Admite los valores:**

**none:** No se aplicará ninguna transformación.

**uppercase:** El texto se pondrá en mayúsculas

**lowercase:** El texto se pondrá en minúsculas

**capitalize:** Transforma la primera letra de cada palabra en mayúscula.

**Adornos en el texto**

**Text-decoration:** Esta propiedad nos permite controlar la línea que podemos poner en los textos.

**Admite los valores:**

**none:** Elimina cualquier línea.

**underline:** Coloca una linea bajo el texto.

**overline:** Coloca una linea sobre el texto.

**line-through:** Coloca una linea tachando el texto.

**Text-shadow:** Funciona exáctamente igual que box-shadow, con la única diferencia de que no admite el valor inset ni la expansión de la sombra.

**Los valores son:**

**offsetX:** desplazamiento horizontal.

**offsetY:** desplazamiento vertical

**blur:** desenfoque.

**color:** color, por defecto coge el del texto.

**ejercicio:**

article{

    text-transform: capitalize;

    /\* text-transform: uppercase;

    text-transform: lowercase; \*/}

a{

    color: inherit;

    text-decoration: overline;

    text-shadow: 1px -2px rgb(123, 72, 72), 1px -3px #255;

}

**40. Espaciado en fuentes y textos**

**letter-spacing:** Establece la separación entre caracteres, se puede establecer en cualquier medida válida para CSS

**word-spacing:** Establece la separación entre palabras,

se puede establecer en cualquier medida válida para CSS

**line-height:** Establece el alto de cada línea, se puede dar una medida concreta o se puede poner un número que se multipliará por el font size del texto al que se lo estemos aplicando.

**Orientación del texto.**

writing-mode: Establece la dirección del texto en vertical