APUNTES

HTML + CSS

Santiago Fonzo – HTML + CSS – 2022

HTML

HyperText Markup Language. Se usa específicamente para estructurar la página web.

Siempre hay que iniciar la página con <html>: agrega un formato estándar constituido por <head></head> y <body></body>. El primero incluye todo lo que la página web no muestra, por ejemplo, título en el navegador; el segundo es dónde se estructuran las cajas y títulos de la página.

Hay dos tipos de caja o elementos: display block o display line. En el primer tipo, el tamaño del elemento en cuestión ocupa el resto de la línea; en el segundo el elemento se ajusta al contenido. Normalmente las etiquetas de estructura tipo <h1>, , etc. son en bloque, por lo que siempre aparecen en forma de lista; las de formato como <b>, , etc. se adaptan al texto escrito. Dos elementos en bloque seguidos en el código (uno al lado del otro) aparecerán uno debajo del otro en la página puesto que cada uno ocupa el espacio restante de su línea, mientras que dos elementos del tipo línea aparecerían uno al lado de otro en la página web a pesar de estar escritos uno bajo el otro en el código.

Indice de etiquetas:

- Genéricamente:
    - <etiqueta atributo="valor">: abre objeto
    - </etiqueta>: cierra objeto
    - Ej.: <button atributo="valor"></button>
- 2 tipos de etiqueta(ej.):
    - Se cierran automáticamente (no es necesario):
        - <input>
        -
    - Es necesario cerrarlas (incluyen algo)
        - <hn></hn> (Títulos); n = 1-6; <h1> solo 1 vez.

- Header (Títulos):
    - **<hn></hn>**; n = 1-6; <h1> solo 1 vez.
- Paragraph (Párrafos):
    - ** **
- Black Font (negrita):
    - **<b><b/>**
- Itallics (cursiva):
    - ****
- Strike (tachado):
    - **<strike></strike>**
- Small (letra chica):
    - **<small></small>**
- Atribute (referencia de vínculo):
    - ****

        - Para ingresar una página exógena al proyecto es requisito agregar **https://** (Hypertext Transfer Protocol Secure). Ej.: ****
        - Para referir a una página endógena al proyecto (otro archivo html) simplemente alcanza con poner el nombre del archivo html. Ej.: .: ****
        - Si quiero moverme a una página endógena que se encuentra en otra carpeta simplemente hay que agregar el nombre de la carpeta/. Ej.: ****. Si además quiero desde esta nueva página volver a la anterior que se encuentra en otra carpeta, se agrega **../** antes de la página destino: Ej.: **** .
        - En caso de querer que el vínculo se abra en una nueva pestaña es necesario emplear la etiqueta **target** dentro de la etiqueta <a>. Ej.: **Visitanos!**
        - Si quiero moverme dentro de la misma página, lo que tengo que usar es el atributo **href="#xxx"**, donde xxx es un **id** que asigné a otro sector o título de la página utilizando el atributo **id=""** (este es un identificador). Ej.: ** ----------> <h2 id="xxx">asdasd</h2>.**
- Br (**completa línea con espacio** o una línea extra en caso de agregar 2)
    - **
**
- Span (para dar estilo a texto):
    -
- Unordenated list (lista no-ordenada, sin orden explicito):
    - **
**: se ingresa el título de la lista utilizando un <h1></h1>, luego de agrega la etiqueta **
** para definir cada item de la lista.
- Ordenated list (lista con orden explicito o numerada):
    - **
**: funciona de exacta igual manera a .
- Imagen (introducir una imagen):
    - **<img src=" ">** : puedo completar el atributo utilizando una fuente externa (link con https://) o con una imagen interna del dispositivo (imagen.jpg u otro formato teniéndola en la carpeta del proyecto)
    - **Width o height:** puedo **modificar el tamaño de la imagen** utilizando alguno de estos modificadores. Si defino ambos se puede desproporcionar la imagen, por lo tanto debería usar uno solo. Sin embargo, no es recomendable porque inhabilita la modificación vía CSS. Ej.**: <img src="imagen.jpg"** height="200px"**>**
    - **Alt**: es otro atributo que puede agregarse en este caso, vendría a ser como la "cosa" alternativa que muestra la página en caso de no encontrar la imagen. Ej.: **<img src="imagen.jpg" alt="Imagen">**. Indispensable.
    - **Title**: es un título o descripción de la imagen que deseamos que se muestre en una pequeño cuadro al suspender el cursor sobre la imagen. Ej. **<img src="imagen.jpg" alt="Imagen" title="descripción de la imagen">**.
- Video/audio (introducir video o audio de tal):
    - **<video src=" "></video>**: esta etiqueta requiere cierre, los videos necesariamente tienen que terminar con una extensión de video. Si nosotros ponemos el video sin

más no tendremos controles play/pause/barra, por tanto es necesario agregar el atributo **controls=" "** que no necesariamente tiene que tener un valor (en este caso se utilizan controladores por defecto del navegador.
    - **<audio src=" "></audio>**: funciona exactamente igual que el anterior sin mostrar la imagen del video.
- Div (permite sectorizar el contenido):
    - ** **: dentro de la etiqueta, si bien html acepta un texto sin más, lo correcto es poner un <hn></hn> y posteriormente otros como , <img> o lo que sea necesario.
- Forms (formularios):
    - **<form></form>**: esta etiqueta se utiliza para generar formularios y va acompañada de inputs. En caso de lenguaje de *Back-End*, se acompaña con un atributo de encriptación como method="post" para que se envíe al servidor.
    - **<input type=" " name=" " placeholder:"">**: esta toma y procesa la información ingresada para devolver un dato al usuario, por ej. una contraseña y el hecho de que sea o no correcta. Además puede estar acompañado de **value=""** que mostrará qué aparecerá en el cuadro previo al ingreso de datos. **Placeholder=""** se utiliza para mostrar un texto indicativo que desaparecerá al ingresar el primer carácter. El atributo **type=" "** implica el tipo de información que se introduce, pueden ser:
        - **number**: solo permite introducir números
        - **text**: permite ingresar texto sin restricciones
        - **password**: se ingresa texto sin restricciones que se "tapa" con *
        - **email**: solo permite introducir formatos de correos electrónicos
        - **color**: permite elegir un color desde una paleta
        - **range**: barra deslizable, además permite ingresar los modificadores min= "" y max="" para determinar extremos del rango. El indicador se mueve a escala según la amplitud del rango.
        - **date**: fecha en formato dd/mm/aaaa
        - **time**: formato hora
        - **button**: siempre acompañado del atributo **value=""**, esto define qué se ve en el recuadro de dicho botón.
        - **submit**: este es un botón específicamente para enviar los datos del formulario, si no defino el value="" para este aparece por defecto "enviar" en el idioma detectado por el dispositivo, por lo tanto no es necesario como sí lo es para button.
            - Además, existen otros atributos:
                - **required=""**: este determina si el campo del formulario es o no obligatorio. No se suele utilizar este tipo de validaciones en *Front-End* ya que pueden saltearse modificando el código asequible al público.
                - **name=" "** se utiliza para que el servidor interactúe con el input del usuario.

        - **<textarea></textarea>**: es una especie de input que permite modificar el tamaño del recuadro correspondiente al input anterior; en ocasiones particulares puede acompañarse de un atributo readonly="" que imposibilita modificar lo que esté por default escrito en ese campo (como si fuera una nota o recordatorio al usuario).
- Metadatos: información que describe datos o recursos.
    - <meta>: se coloca en el head.
        - **<meta charset="">** (normalmente su valor es **utf-8)** sirve para cambiar la codificación de la página, es decir, los caracteres que permite visualizar.
        - **<meta name="keywords" content="xxx,xxxx,xxxxx">**: sirve para "destacar" una página en un motor de búsqueda a través de palabras clave.
        - **<meta name="description" content="descripción">**: se utiliza para describir la empresa o la institución que esté detrás de la página. (~100 palabras).
        - **<meta name="autor" content="nombre del autor de la web">**: para registrar al autor de la página.
        - **<meta name="copyright" content="compañía x">**: se utiliza en caso de que la empresa posea copyright sobre lo publicado.
        - **<meta name="robots" content="index/noindex">**: se utiliza para indexar o no la página, es decir que aparezca entre los resultados del navegador o no.
        - **<meta name="robots" content="follow/nofollow">**: para dar o no importancia en el SEO.

HTML Semántico: mejor forma de escribir u optimizar el código.

En la parte superior de una página web deben encontrarse un **encabezado** y/o un **panel de navegación**, debajo se encuentra una área denominada **artículo** conformada por distintas **secciones** y al lado el **aside** (barra lateral). Por último se agrega un **footer**, o pie de página.

- **<nav></nav>**: Se utiliza para crear un panel de navegación dentro de la etiqueta **<header>** (que a su vez va dentro del **<body>**), tipo lista o menú. Se puede usar un ** asdasd
** n veces/renglones según cantidades de vínculos **
**.
- **<article></article>**: cuerpo de la página, útil y necesaria para estructurarla.
- **<section></section>**: incorporado dentro del artículo, para delimitar dos o más secciones.
- **<aside></aside>**: contenido extra en una pequeña sección que suele estar a un lado de la página.
- **<footer></footer>**: se utiliza para agregar copyright, redes, contacto, etc.

Tablas: estructuras de columnas y filas.

- **<table></table>**: abre la tabla. Se puede acompañar con atributos como **border="npx"** (aunque es mejor hacerlo con CSS).
    - **<tr></tr>:** define las filas
        - **<td></td>**: define cada campo de la fila

De esa manera: abro tabla, determino una fila e ingreso categorías o clasificación de los datos a ingresar debajo, nuevamente determino otra fila (la segunda de la tabla) y allí colocaría la primer fila de datos correspondiéndose en orden con las categorías definidas en la primera. Repetir x filas.

Posición de texto, imagen o video:

- ** **: Insertando la imagen o título dentro de esta etiqueta se la centra en el espacio de la línea. Sin embargo, se recomienda acomodar mediante CSS y no utilizando esta etiqueta.
    - En caso de querer centrar datos de una tabla, es necesario colocar un center dentro de cada <td></td>. De lo contrario, si se pone antes y después de la tabla como se haría como una imagen o título se centra la tabla "como si fuera una imagen" y no los datos ingresados.

Iconos: aplicables al título del navegador (dentro de head)

- **<link rel="icon" href="xxx.ico">**: indispensable que la extensión del archivo sea .ico.

Clase: a cualquier elemento puedo darle una denominación de clase mediante el atributo **class="**xx**"**. Es útil para hacer una gran selección y luego formatearlos mediante CSS.

Comentarios: para dejar un comentario o parte del código como borrador, lo que se hace es aplicar <!-- código -->/. Así queda "escondido" y no se ejecuta.

Errores comunes!

- Buscar que HTML sea visualmente atractivo.
- No usa etiquetas obsoletas, Google recompensa a las páginas con una sintaxis correctas dándoles prioridad en el orden de aparición. Por ej. al usar <hgroup>, o invertir algunos ordenes de etiquetas, agregar por demás.
- No hay que mezclar lenguaje.
- Modularizar: separar por módulos.

CSS

CSS: Cascading style sheets

Para insertar **CSS sobre un código html** puedo:

- Modo etiqueta: antes del cierre **</body>**, agregar **<style type="text/css">** y escribir directamente el código.
- Carga de archivo premade: de la misma manera puedo también cargar un archivo de código CSS utilizando **@import** antes del cierre **</body>**. No muy utilizada.
- Atributo: por ej. para insertarle color a un párrafo **<p color="red">**.
- Linkeado: dentro del **<head>**, empleo la etiqueta **<link rel="stylesheet" type="text/css" href="estilo.css"**. Esto implica crear y desarrollar el código CSS en un **archivo alterno .css** que luego se vincula al ya escrito html.
- Estilo en línea: insertando el atributo **style=""** al elemento y definiéndolo según la línea que se daría en CSS.

Estructura básica:

- **selector {**
  propiedad:valor;

  }

Donde:

- Selector: define qué elemento/s se quiere/n modificar para modificar la propiedad. Poniendo una "," entre selectores puedo aplicarles la misma propiedad. (Para ampliar sobre selectores ver: https://www.w3schools.com/cssref/css_selectors.asp)
  - Universal (*****): selecciona a todos los elementos.
  - De tipo (**nombre del elem.**): en este caso selecciono puntualmente qué elemento o elementos del mismo tipo del código html quiero modificar de manera indefinida (abarca a todos los del mismo tipo). Por ej. simplemente un h1.
  - De clases (**.clase**): dónde clase es la denominación dada a un elemento mediante el atributo **class="**clase**"** en el código html. Al tener la clase definida previamente solo aquellos que la tengan se verán modificados, independientemente del tipo que sean. Más que una clase, en realidad, sería una simple etiqueta que se le da a un elemento; podrías definir **class="mono"** y cualquier elemento con tal etiqueta se vería afectado.
  - Por ID (**#id**): actúa igual al de clases (como una etiqueta. Sin embargo, es IMPORTANTE tener en cuenta que **NO DEBE REPETIRSE** para más de un único elemento (incompatibilidad con JS). Es literalmente un identificador de un elemento puntual, y además tiene un mayor rango de especificidad.

- Por atributo (**[atributo=”valor”]**): Igual a clase pero toma **atributos definidos** de los elementos. Recordar que un atributo se puede inventar y en este caso no deja de ser diferente a un **.clase** o **#id**.
- Descendiente (**x y**): se utiliza para seleccionar **por tipo diferenciando** de aquellos que estén contenidos dentro de elementos determinados o en función de la clase definida de tales elementos. Por ej. los párrafos que estén dentro de un h2 se seleccionarían como **h2 p**; párrafos que estén dentro de elementos de clase z se seleccionarían como **.clase p**. De esta manera puedo crear “cadenas” con prioridad hacía el elemento más interno (predominio sobre el estilo del conjunto de elementos menos específicos).
- Por pseudoclases (**elemento o clase:pseudoclase**): las pseudoclases son como clases condicionadas y dinámicas, por ej. hover es una pseudoclase que se activa cuándo el cursor está sobre el elemento. Entonces, si quisiera que un párrafo se torne rojo al ubicar el cursor sobre el mismo podría definir **p:hover**.

- Propiedad: literalmente eso, qué característica el elemento seleccionado quiero modificar.
  - Ej. **color: #f00;** tener en cuenta que para c/color se utiliza un código hexadecimal específico.
- Valor: es la modificación que quiero hacer, por ej. en caso de querer cambiar el color de negro (dafault) a rojo, el valor correspondiente a la propiedad color sería red.

Especificidad: es la jerarquía que se les da a las distintas operaciones, para superponerse o para prevalecer ante una acción posterior.

!Important
Estilos en línea
Identificadores
Clases
Pseudoclases
Atributos
Elementos
Pseudoelementos

!Important: se puede asignar a cualquier elemento o pseudoelemento (o lo que sea) para fijar inmutablemente una característica. Por ej.:

```
h1 {
color: green !important
}
```

Solo recomendable en casos muy particulares, no es necesario si se hace un uso correcto de las especificidades.

El efecto “cascada” o superposición solo se lleva a cabo cuando el nivel jerárquico o cuando el posterior es superior al primero.

Metodología BEM (bloques, elementos, modificadores): Es un sistema de clases/etiquetas que sirve para evitar conflictos de superposición de propiedades/jerarquías y trabajar el código de una manera más organizada.

- Bloque: es un bloque o espacio contenedor (puede ser un div) definido que va a contener distintos elementos. Esta denominación en la etiqueta va a diferenciar a un bloque de otro dentro del código.
- Elementos: nada, eso, elementos como un <p> o <h2>. Esta denominación en la etiqueta va a diferenciar a los distintos elementos dentro del bloque.
- Modificador: será lo que destaque a un elemento en particular de otros elementos del mismo tipo dentro del bloque con el objetivo de aplicarle una modificación especial a ese elemento particular.

Se genera de la siguiente manera: **Bloques__Elementos—Modificador.**

Unidades:

- Relativas: dependen de la caja contenedora como un total, si la caja se achica el elemento definido con tales unidades se achica proporcionalmente de manera que se mantenga la relación tamaño.
    - Se puede utilizar ***em***, que es por default 16 px (según el navegador); sin embargo si defino el font-size del bloque sea 20 px, 1 **em** será 20 px y por lo tanto, será relativizado a lo que se defina anteriormente. Esto es aplicable a todas las propiedades que se puedan ser definidas en unidades de medidas/dimensiones.
    - Otra que se puede utilizar es **viewport height** o **viewport width** (**vh** o **vw**, alto y ancho relativamente). Estas unidades son más bien porcentajes, van desde 0 a 100, donde 100 es el 100% del espacio del dispositivo.
- Fijas: son aquellas que simplemente dependen del número que se les asigna y solo se modifican ante la modificación de tal número, incluyen a las magnitudes cm, px, mm, etc.

Propiedades:

- De tipografía:
    - **font-family**: tipografía o fuente. Puedo utilizar algunas preexistentes como Georgia, sans-serif, etc. o puedo insertar un .css de fonts.google.com en el <head> del html y luego en mi .css aplicarla de la manera que sugiere Google Fonts (por ej. **font-family: 'Oswald', sans-serif**; donde la **,** del medio implica que de no encontrar la primera fuente (no ingresada en el <head>) aplique la siguiente en la regla.
    - **font-size**: tamaño de fuente.
    - **line-height**: altura de la línea, altura de letra + interlineado de la línea; relativo al tamaño de fuente: (1 = alto de la letra, si fuese 2 serían 2 alturas de letras donde 1 altura se convierte en interlineado dividiéndose en 0.5 altura de letra arriba y 0.5 altura de letra abajo.
    - **font-weight**: grosor de los caracteres, dependen de la tipografía.
    - **font-stretch**: estiramiento o compresión de la fuente.

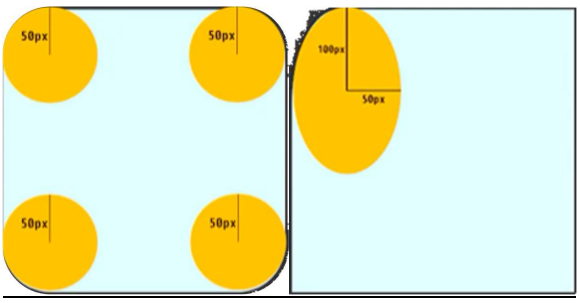
- **font-variant**: esta propiedad admite el valor **small-caps**, utilizado para convertir las letras minúsculas del texto en mayúsculas conservando la altura de las primeras.
- **text-shadow**: da sombra difuminada al texto y permite introducir 4 valores. El primero es la cantidad que debe desplazarse en el sentido x^+ (derecha)y el segundo es la cantidad que debe desplazarse en el sentido y^- (abajo), el tercero es el grado de difuminación que deseo darle con respecto al tamaño de la caja y el cuarto es el color de la sombra. Si deseo intensificar el efecto puedo agregar una **,** en lugar del **;** y reescribir la propiedad, luego finalizar con **;** (también valido para **box-shadow**).

- De cajas:
  - **display**: sirve para definir si un elemento se comporta como bloque o en línea (**inline)**. Por ej. en este caso el **h2** que normalmente es un elemento de bloque, se comportaría como uno de línea.
    - **h2 {**
      display: inline;
      }
    - Es importante tener en cuenta que al modificar bloque → en línea, dejaran de funcionar propiedades como **height** o **width**.
  - **background**: permite cambiar color de fondo.
    - Normalmente el valor asignado es un color sólido.
    - Un valor distinto es **linear-gradient**. Este permite que el background en cuestión sea un gradiente. Ej. **background: linear-gradient(to bottom, transparent, #xxx);** donde **to bottom** índica la dirección del vector gradiente en cuanto a solidez del color (hacia donde crece), **transparent** indica el color inicial y **#xxx** el final.
    - Otro valor es **radial-gradient**. Genera un background gradiente de color radial y su sintaxis es similar a la del anterior.
    - Esta propiedad es un shorthand y engloba a las propiedades:
      - **background-color**
      - **background-image: url()**: aplica una imagen como fondo del elemento.
      - **background-size**: esta propiedad admite porcentajes que representan una fracción sobre el tamaño original de la imagen, admite valores en unidades (px, etc.) y valores funcionales como **cover**, **contain** (similar **object-fit**).
      - **background-repeat**: determina el patrón de repetición de la imagen de fondo. Admite valores como **no-repeat** (más común). Es importante tener en cuenta que si la imagen sin repetir no cubre el elemento completo se rellena con el **background-color**.
      - **background-position**: posiciona la imagen de fondo admitiendo dos valores entre **top**, **bottom**, **left**, **right**, **center**. El primer valor posiciona en el eje horizontal y el segundo en el eje vertical.

- **background-attachment**: cambia la referencia de la posición de la imagen, es decir, si scrollea o no (por ej.).
    - **background-attachment: scroll;**: este valor mantiene fija la imagen usando como referencia el contenedor (por defecto).
    - **background-attachment: fixed;**: este valor mantiene la imagen fija en un segundo plano, literal como suele funcionar el valor **fixed** en la propiedad **position** pero funcionando como **background**.
- **background-clip**: esta propiedad limita qué parte de la imagen será visible.
    - **border-box**: muestra la imagen desde el border del contenedor.
    - **padding-box**: muestra la imagen desde el **padding.**
    - **content-box**: muestra la imagen desde el contenido.
- **background-origin**: esta propiedad genera la imagen desde el punto asignado, a diferencia de **background-clip** que la recorta. Admite los mismos valores que la anterior.
    -

- **padding**: distancia entre el texto y el borde de la caja (márgenes dentro de la caja). Si bien puedo aplicar uno general con la propiedad padding cada uno se puede definir como **padding-top**, **padding-bottom**, **padding-right** y **padding-top**. Es una propiedad acortada, es decir, se puede modificar parcialmente mediante las últimas propiedades.
    - También se pueden determinar en una única línea **padding** considerando el siguiente criterio:
        - Si inserto dos valores en pixeles (por ej.) serán en referencia al eje *y* (alto) y luego al *x* (ancho)
        - Si inserto cuatro valores serán en referencia a top-right-bottom-left (sentido horario).
        - En caso de insertar 3 valores, se obtiene la relación top-*eje x*-bottom.
- **height**: permite ajustar la altura de la caja. También sub-propiedades **min-height** y **max-height** que funcionan como límites.
- **width**: permite ajustar el ancho de la caja. También sub-propiedades **min-width** y **max-width** que funcionan como límites.
- **margin**: es el espacio que hay entre las cajas. Y es una propiedad acotada como el **padding** y cada margen se define de la misma manera (top, right, left, bottom) y en caso de usar una única línea funciona también de la misma manera. Si defino **margin auto;** se utiliza en el eje *y* los márgenes definidos y en *x* se centra la caja.
- **border**: puedo definir las características del borde, es decir el estilo de borde (ancho, si es doble, punteado, etc.). Se aplica por ej. de la manera: **{border: 4px solid blue}**.

- **border-radius**: Redondea las esquinas en la medida que lo defino (en unidades o %).
  - Es una propiedad shorthand/acotada que engloba las propiedades: **border-top-left-radius**, **border-top-right-radius**, **border-bottom-right-radius**¸**border-bottom-left-radius**, **top-left-and-bottom-right** y **top-right-and-bottom-left**.
  - Para **border-radius**:
    - Si defino un valor aplica a todas las esquinas.
    - Si defino dos valores: el primero aplica a las esquinas superior izq. e inferior der, el segundo para el par restante.
    - Si defino tres valores: el primero aplica a la esquina top left, el segundo aplica para la esquina top right y la esquina bottom left, el tercero aplica para la esquina bottom right.
    - Si defino cuatro valores: el primero será la esquina superior izq. y el luego las restantes en sentido horario.
    - Si ingreso dos valores y los separo mediante una **/** se aplicarán elipses en lugar de círculos.
    - Si ingreso dos valores a cada lado de la barra colocaría un corte elíptico desigual en las dos diagonales (considerar criterio dos valores sin **/**).
    - Si ingreso tres valores a cada lado de la barra se colocan tres elipses distintas (criterio sin barra) y se corresponden las medidas abc de la izq. de la barra con las a'b'c' de la der. de la barra.
    - Si ingreso 4 valores serán 4 elipses distintos, con los abcd (izq.) correspondientes a los a'b'c'd' (der.).
    - Se recomienda trabajar con las esquinas de manera independiente dada la complejidad de la sintaxis general.
  - Para las **shorthand**:
    - Si coloco un solo valor funciona como se menciona arriba.
    - Si coloco dos valores: el primero es el radio de una elipse en el eje *x* y el segundo es el radio de la elipse en el eje *y*.
  - Los valores pueden ser en px o en porcentajes (estos se calculan en referencia a la altura y ancho de la caja).



- **box-sizing**: esta propiedad define como se ajustaran los bordes de la caja contenedora. Si la defino como **box-sizing: content-box;** la caja mantiene las dimensiones (**width** y **height** definidos) de manera adicional al **padding** definido; si la defino como **box-sizing: border-box;** la caja ajusta las dimensiones a las definidas descontando el **padding** (definidas = padding + diferencia).
- Las propiedades **line-height**, **padding**, **border** y **margin** son aditivas, es decir el tamaño total de la caja (incluyendo margin) será la suma de todas ellas. Dicho de otra manera, no se superponen.
- **opacity**: es la opacidad o transparencia que deseo darle al fondo de un elemento. Varía entre 0 y 1.
- **box-shadow**: es la propiedad de sombre para la caja. Se puede definir con cinco valores: **box-shadow: 10px 20px 15px 0 #xxx;** donde el primero es la cantidad que debe desplazarse en el sentido x^+ y el segundo es la cantidad que debe desplazarse en el sentido y^-, el tercero es el extra de tamaño que deseo darle con respecto al tamaño de la caja , el 4to es un extra de tamaño que deseo darle pero no de sombra sino de color sólido y el 5to es el color.
- **transform**: es una función que sirve para manipular la caja en el espacio, por ej. para rotar a través del valor **rotate**: **transform: rotate (90deg);**.
- **outline**: Es un shorthand. Genera un borde que no ocupe un espacio real en el DOM (document object model). Sirve para resaltar únicamente. A diferencia de los **border** no modifica la ubicación de la caja, cosa que es sería necesaria para ver un borde superior en una caja que se encuentre al inicio de la página.
- **position**: que un elemento esté posicionado implica que afecta al flujo en el que se acomodan los elementos del html. Es un shorthand compuesto por los desplazamientos **top**, **left**, **right** y **bottom,** siendo **top** y **left** abosolutas frente a **right** y **bottom** (esto implica que ante la presencia de las primeras definidas las segundas son ignoradas y solo tienen efecto ante la ausencia de las primeras). Al permitirse usar valores negativos es posible utilizar solo las primeras para todos los desplazamiento.
    - **static**: se mantiene "clavado" en un lugar de la página.
    - **relative**: hace que la caja conserve su espacio y a partir de este que se desplace según lo indicado desde dicha posición. De esta manera puede superponerse a otras cajas.
    - **absolute**: a diferencia del **relative** el espacio de la caja no está "reservado". En este caso, entonces, la referencia de desplazamiento será el contenedor si es que este está posicionado, de no estarlo la referencia será el viewport. Además, salvo que las dimensiones estén especificadas el espacio del contenedor se ajusta al contenido, incluso aunque sea un elemento en bloque. En caso de querer centrar en el contenedor se podría calcular matemáticamente, sin embargo, en caso de no conocer las dimensiones del mencionado puedo aplicar **top**, **left**, **right** y **bottom:** todos con el valor **0**, seguido de **margin: auto;**, esta forma es la más útil ya que en caso de adaptarse a otras resoluciones las dimensiones del contenedor varían.

        - **fixed**: funciona al igual que un **absolute** no ocupa espacio fijo, sin embargo está fijo en el viewport mas no en la página. Es útil para barras de navegación y menús fijos.
        - **sticky**: es una combinación de **relative** y **fixed**. Este conserva el espacio como el primero pero permite determinar desde qué momento se podrá mover con la página tal **relative** y desde qué momento estará fijo tal **fixed** (como si se lo llevara puesto, esto se puede definir con un top).
    - **z-index**: Al superponerse las cajas entra en juego el **z-index** que es un valor en la dirección del eje *z* (normal al display), los valores mayores traerán el elemento al frente; solo funciona en caso de que se encuentren posicionadas las cajas! Además es recomendable que se usen intervalos de 50 puntos para definirlos en lugar de una secuencia *n*; *n+1* dado que esta última implicaría modificar manualmente todos los valores en caso de necesitar agregar elementos en el medio. Esta propiedad tiene conflictos entre elementos contenedor/hijo, un elemento contenedor no podrá colocarse por encima de los hijos, para lograr esto solo se puede dándole la propiedad **z-index: -1;** al elemento hijo y dejar el contenedor sin **z-index** definido.
    - **display**:
        - **block**: elementos que ocupan al resto del display.
        - **inline**: se ajustan al contenido. Normalmente usado para texto.
        - **inline-block**: se ajustan al contenido pero permiten modificar sus dimensiones. Dicho de otra manera, que sea un bloque pero que el tamaño se ajuste.
        - **table**: comportamiento de tabla.
        - **inline-table**: comportamiento de tabla en línea.
        - **list-item**: comportamiento de <nav>.
        - **table-cell:** comportamiento de una celda de la tabla.
        - **table-row**: comportamiento como fila de tabla.
        - **table-column**: comportamiento como columna de tabla.
        - **flex**: los elementos con estas características mantienen la misma altura siempre, ajustando el ancho ante la diferencia de contenido.
        - **grid:**
        - **inline-flex**:
        - **inline-grid**:
    - **overflow**: modifica el contenido que sobra o sobresale de la caja. Es un shorthand que engloba a **overflow-x** y a **overflow-y**.
        - **inherit**: deja que se exceda hacia afuera.
        - **scroll**: agrega barras de scroll sea necesario o no en ambos ejes.
        - **auto**: detecta y agrega una barra de scroll en el eje que sea necesario.
        - **hidden**: esconde la barra de scroll del eje que se desee, se utiliza en una de las propiedades *x* o *y*.
        - En caso de utilizar una imagen dentro de la caja el tamaño se ajustará al contenedor y se centra, sin embargo en caso de querer desplazarla de tal posición definiendo un **position: relative;** junto a un desplazamiento **top** y

left de determinada cantidad de **px** la caja se mostrará por fuera de la misma salvo que se aplique **overflow: hidden;** al contenedor y en tal caso se oculta el exceso de la imagen.

- **float**: ubica a un elemento hijo al lado izquierdo (**float:left;**) o al lado derecho (**float: right;**) del elemento contenedor.
    - Float está actualmente considerado anticuado para lo que fue creado, sin embargo una utilidad que se le puede dar es que si dentro de un contenedor como un **div** hay texto (sin ser parte de un ** **) y previamente una imagen como en ej. ** <img src=""> texto ** al dar la propiedad **float: left/right;** a la imagen se da el efecto del texto ajustado a la misma como el típico que da office, aunque para ajustar el alineado y ordenar es necesario darle un margin a la **img** y un padding al **div**.
- **object-fit**
    - **contain**: ajusta la imagen al contenedor manteniendo la resolución real.
    - **cover**: la imagen se ajusta al contenedor manteniendo la resolución real, cubriendo todo el contenido y cortando aquello que sobre (como haciendo un **overflow: hidden;**).
    - **none**: usa las propiedades por defecto, es decir el tamaño real de la imagen.
    - **scale-down**: entre **none** y **contain**, selecciona el tamaño más chico.
- **object-position**: acomoda la imagen dentro de la caja. Se puede acompañar con **cover** para que en caso de que haya recorte se pueda seleccionar que parte. También admite medidas para moverla dentro de la caja.
- **cursor**: permite modificar el cursor al posicionarlo sobre un elemento.
    - Valores: se pueden buscar en w3s school.com
    - Se puede aplicar dentro de una pseudoclase :**active** para que al clickear el cursor cambie durante el período del click.
- **direction**: esta propiedad define como fluye el texto. Una especie de alineado de texto que suele utilizarse para idiomas particulares que implican escritura de derecha a izquierda (árabe o hebreo).
- **direction: ltr;** es el valor que la propiedad tiene por defecto y **ltr** significa *left to right*.
- **direction: rtl;** es un valor que puede darse a la imagen es un valor que puede darse a la imagen.
- **letter-spacing**: esta propiedad aplica un espaciado a las letras de un texto. Admite unidades absolutas y relativas.
- **scroll-behavior**: esta propiedad modifica la velocidad a la que un link local (**href="id"**) puede referir a otra sección (id="id") de la página misma página. Esta propiedad debe aplicarse al **body**, no al elemento destino ni al link ni al contenedor, puesto que quien sufre el scroll es el **body**.
    - **auto**: es la propiedad por defecto e implica un scroll hacia el elemento destino instantáno.

        - **smooth**: hace que el desplazamiento hacia el elemento destino sea suave. La función de tiempo y la duración del desplazamiento dependerán del navegador.
    - **user-selected**: es una propiedad que controla si el usuario puede seleccionar el texto.
        - **none**: este valor imposibilita a los usuarios a seleccionar texto y por ende imposibilita copiar, buscar la selección en google, etc.
        - **text**: el texto puede ser seleccionado por el usuario.
        - **contain**: permite la selección dentro de un elemento mas está limitada por los límites del elemento en cuestión. No funciona en IE, en ese caso se usaría **element**.

Pseudoelementos: elementos que modifican elementos. Se aplican de la siguiente manera: s**elector::pseudoelemento {propiedades}**.

- **first-line**: no funciona en elementos inline. Lo que hace este es que al elemento seleccionado, como un **div** que contiene texto, aplica que la primera línea del texto tenga características (que hay que definir) distintas del resto y es dinámica, es decir, si el viewport se achica y el texto se ajusta solo la primer línea tendrá la propiedad y no específicamente una oración o parte del texto. Forma parte del DOM.
- **first-letter**: no funciona con elementos inline. Aplica propiedades específicas a la primera letra. Forma parte del DOM.
- **selection**: permite modificar las características del texto al seleccionar con el cursor. Se comporta como un elemento en línea. No forma parte del DOM.
- **placeholder**: permite modificar el placeholder asignado a un input. No forma parte del DOM.
- **before**: crea un pseudoelemento hijo del elemento seleccionado, se utiliza para añadir contenido estético *antes* al agregarle la propiedad **content: “”**. No forma parte del DOM.
- After: crea un pseudoelemento hijo del elemento seleccionado, se utiliza para añadir contenido estético *después* al agregarle la propiedad **content: “”**. No forma parte del DOM.

Pseudoclases: aplican propiedades a elementos como reacciones a eventos. Se aplican de la siguiente manera: .**selector::pseudoclase {propiedades}**. Funcionan como elementos en línea.

- **first-child**: modifica al primer elemento de cierto tipo contenido dentro de otro, útil para no asignar ninguna clase a este aunque no sería óptimo en caso de que sea necesario agregar algo antes. Si no se especifica cuál es el elemento (por ej. **div p:first-child**, especifica que es un **p)** se tomará el primer hijo independientemente del tipo.
- **nth-child(**n**)**: funciona de manera similar al anterior pero sirve para seleccionar otros elementos del grupo de hermanos.
    - Valores:
        - Los valores ingresados pueden ser series numéricas.
        - **n** = **número entero específico**: aplica a un elemento que se encuentra en n posición; por ej. para el segundo elemento de un grupo n = 2.

    - **even**: selecciona a los hijos que se encuentren en posiciones pares (n = par, o n = 2n con $n_0 = 0$).
    - **odd**: selecciona a los hijos que se encuentren en posiciones impares (n = impar, o n = 2n+1 con $n_0 = 0$).
    - Ejemplos de series:
        - Si quisiera todos los elementos que estén en posiciones múltiplos de 5 (0, 5, 10 ,15, etc.) el valor que debo asignar es 5n.
        - El valor 3n + 2 implicaría los valores 2; 5; 8; 11; etc.
        - El valor -n + 3 incluye a los tres primeros valores.

Comentado [SF1]: Considerando que para n = 0 el valor correspondiente es 3 y que para n = 1 el correspondiente es 2. Esto incluiría a los anteriores al término independiente de la función de la serie. Considerando que a medida que n > 2 se tomarán valores iguales a 0, o negativos, que no se corresponden con elementos del grupo de hijos solo se limitarían a tales. Simple math.

- **hover**: modifica al elemento de la clase definida, según las propiedades definidas, al posicionar el cursor sobre él.
- **link**: el enlace adopta ciertas propiedades solo hasta el momento en el que es visitado.
- **visited**: es el contrario al **link**. La propiedad se adopta una vez que en enlace es visitado. Tanto **link** como **visited** tienen más jerarquía que un simple **color aplicado** al elemento.
- **active**: el elemento modificará sus propiedades mientras se lo esté clickeando.
- **focus**: se aplica a los **inputs**. Modificará sus propiedades cuando el elemento esté seleccionado, en este caso cuándo se clickee para escribir.
- **lang**: es una función (requiere un parámetro, en este caso idioma) que modifica las propiedades de un lenguaje determinado (indicado previamente como un atributo **lang=""** de un elemento en html). Es como una selección por atributo.

Comentarios: para dejar un comentario o parte del código como borrador, lo que se hace es aplicar /* código */. Así queda "escondido" y no se ejecuta.

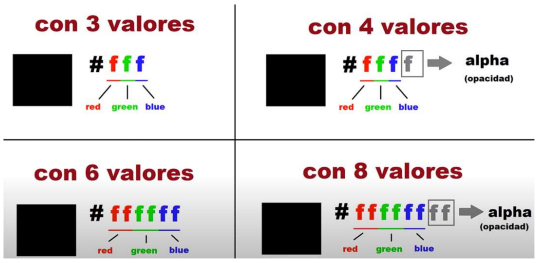
Normalize: es un archivo .css que reinicia los estilos del navegador. Buscar en google y descargar.

- Conviene modificar la regla de imágenes agregando un **max-width: 100%;** para que la imagen se ajuste al 100% del display móvil y no se exceda de los márgenes.
- También es bueno agregar al inicio un **box-sizing: border-box;** esto fija el tamaño de la caja y ajusta automáticamente el tamaño libre dentro de la misma teniendo en cuenta el **paddle** y el **line-height** definidos (ambos márgenes).

Coloración: formas de definir los colores.

- Se pueden elegir los nombres predefinidos que se pueden googlear. Sin embargo, el color puede variar según el navegador.
- La forma hexadecimal conservará exactamente el color definido independiente del navegador.
- La forma RGB es la determinación según la mezcla de colores rojo, verde y azul. Hay 256 matices de cada uno, por lo que los valores van de 0 a 255 (siendo 0x3 = negro y 255x3 =blanco). Se indica como **background: rgb(x,y,z);** donde **x**, **y z** son los valores de cada color. Hay que considerar que a diferencia de la práctica real el verde vendría a suplir al amarillo (rojo + verde = amarillo).
- La forma RGBA se utiliza de la misma manera pero la "a" es la opacidad del color, donde 1 es opaco o no transparente y 0 es totalmente transparente. Se indica como **background: rgba(x,y,z,w);**.

- La forma hexadecimal incluye a los números del 0 al 9 y las letras de la “a” a la “f”. La serie utilizada para cada color va como {00; 01; …; 09; 0a; ab;…)



Mobile first: Implica la adaptación de la web desde un dispositivo de display pequeño a uno de mayor, por ej. de un celular a una pc. Este concepto es importante porque Google recompensa a aquellas páginas que tengan mobile first sobre aquellas que no.

Responsive design: Implica la adaptación de la web desde un dispositivo de display mayor a uno de menor.

- En casos donde las cajas se compacten de acuerdo al display deformando y volviendo ilegible al contenido, se puede aplicar **@media only screen and (max-width: 800px) {…}**. Donde entre los {} puedo proponer las propiedades a modificar (pro ej. que se convierta en un elemento en bloque para ajustarse mejor a un viewport más angosto) y esto tomará lugar una cuándo el tamaño esté por debajo de los 800ox indicados.
- Es necesario aplicar la etiqueta **meta viewport** ya que de lo contrario se tomará a la pantalla siempre como si fuera una pc de escritorio y no funcionará correctamente. Al utilizar la etiqueta se utilizan las resoluciones originales del dispositivo:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

Flex-box:

- Son cajas flexibles. La caja contenedora se comporta como un elementos en bloque, sin embargo los elementos de adentro modificarán su comportamiento. Considerar que los items flex son elementos hijos, y solo hijos directos (no nietos), que están dentro de un contenedor que posea la propiedad **display: flex;**.
- Entre elementos flex, como pueden ser dos **divs**, por defecto se mantienen las alturas ajustando el ancho.
- Una caja normal el eje *x* es el *main axis* mientras que el *y* es el *cross axis*. A cada extremo del primero se lo denominan *cross-start* (arriba) y *cross-end* (abajo), mientras que a los del segundo se los denominan *main-start* (izquierda) y *main-end* (derecha).
- Propiedades aplicables al contenedor:
    - **flex-direction**: sirve para cambiar la dirección del *main-axis*.
        - **row**: mantiene el sentido del *main-axis* (izquierda a derecha). Por defecto.
        - **row-reverse**: invierte el sentido del *main-axis* (derecha a izquierda). 180° rotando sobre eje *y*.
        - **column**: gira el *main-axis* 90° en sentido horario (arriba hacia abajo).

    - **columna**-reverse**: gira el** *main-axis* 90° en sentido antihorario (abajo hacia arriba). O invierte el sentido del column.
- **flex-wrap**: fija el tamaño de las cajas y, frente a una disminución del *viewport*, al excedente de las cajas lo pasa a una segunda línea.
    - **no-wrap**: es el valor default, no hace nada.
    - **wrap**: mueve hacia abajo la última caja.
    - **wrap-reverse**: mueve hacia arriba la última caja.
- **flex-flow**: es una propiedad compuesta de las dos anteriores. Funciona como: **flex-flow: flex-direction flex-wrap;**

- **justify-content**:
    - **flex-start**: el margen del primer item es alineado al ras con el borde del comienzo principal de la línea y cada item siguiente es alineado al ras con el precedente.
    - **flex-end**: el margen del último item es alineado al ras con el borde del final principal de la línea y cada item precedente es alineado al ras con el siguiente.
    - **center**: los items flex se alinean al ras entre sí y en torno al centro de la línea. Los espacios vacíos primeros y últimos son equivalentes.
    - **space-around**: se alinean de manera que el espacio entre ítems es el mismo. El espacio vacío primero y último equivale a la mitad del espacio entre dos ítems ya que el espacio intermedio entre dos items es el doble del margin asignado al item (funciona como **margin:auto;**).
    - **space-between**: los items flex se distribuyen uniformemente sobre la línea y el espacio adyacente entre dos items es el mismo. Los bordes externos del *main-axis* se alinean al ras con los bordes de los items extremos.
    - **space-evenly**: alinea los items flex de manera que los espacios entre items y los extremos vacíos sea el mismo.
- **align-items**: se utiliza para cuándo hay una única línea de items y aplica en el eje *y* (o cross). Esta propiedad, al generar una segunda línea la centra en la página.
    - **stretch**: es la propiedad por defecto.
    - **flex-start**: alinea los flex items al inicio de la flex box de manera similar a **stretch**, con la diferencia de que ajusta el tamaño del flex item al contenido (**stretch** lo estira a través del *cross-axis*). Más exactamente, el límite del margen transversal inicial del elemento flexible es unido al borde transversal final de la línea
    - **center**: centra verticalmente en el centro de la flex box. Más exactamente, los márgenes del elemento flexible son centrados dentro de la línea sobre su eje transversal; si el tamaño transversal del elemento es mayor al del contenedor, se excederá por igual en ambas direcciones.
    - **flex-end**: alinea los flex items al final de la flex box pero al combinarla con un **flex-wrap: wrap-reverse;** las líneas se elevan. Más exactamente, el límite del margen transversal final del elemento flexible es unido al borde transversal final de la línea.
    - **baseline**: alinea los flex items al final de la flex box, con la diferencia de que no tiene el problema de **flex-end** frente al **wrap-reverse** por el hecho de alinear las bases al fondo de la flex box (siempre va a acompañar a un **wrap-reverse**. Más exactamente, todos los elementos flexibles son ajustados de modo que sus bases queden alineadas. El elemento con la distancia mayor entre su límite transversal inicial y su base es combinado con el borde transversal de la línea.
- **align-content**: se utiliza para cuándo hay dos o más líneas de items y aplica en el eje *y* (o cross). Esta propiedad, al generar una segunda línea automáticamente la

pone por debajo de la primera sin buscar espaciar el contenido de manera centrada verticalmente.

- Los valores son exactamente los mismos que para **align-items**.

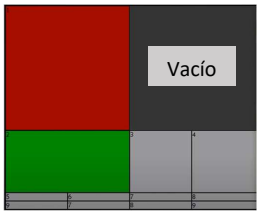
Flex-items: a estos elementos pueden aplicarse todas las propiedades básicas de cajas, sin embargo se adquieren nuevas posibilidades al ser items flex. Es importante considerar que al aplicar **margin: auto** a un item flex automáticamente se extenderá a la capacidad máxima para ocupar el espacio extra en la dirección (left, top, …) aplicada, salvo que se especifique con la medida.

- **align-self**: se puede aplicar a un item flex a través de su clase particular y funciona con los valores de **align-item/content**.
- **flex-grow**: hace crecer los items flex de acuerdo a cuánto espacio vacío quede en algún extremo de la línea de items, luego lo divide por el número de cajas y determina cuánto crecerán en tamaño en la dirección del *main-axis* (según el **flex-direction**). De esta manera el espacio siempre será repartido y no quedarán tales vacíos. Si está definido el **min-width**, por ej., y el viewport alcanzase tal medida simplemente se recorta el overflow a menos que se defina un **flex-wrap: wrap;** al contenedor. Si aplico esta propiedad a distintos items a través de una clase en común se distribuye equitativamente, sin embargo existe la posibilidad de asignarlo de manera particular a items puntuales con clases específicas de cada uno a modo de seleccionar cuáles sí y cuáles no.
- **flex-shrink**: sirve para definir, proporcionalmente, cuánto espacio cederá un elemento al contraerse el viewport. Funciona de manera similar con el efecto opuesto al **flex-grow**.
- **flex-basis:** es similar al **width**, siendo además de mayor jerarquía y específico para elementos flex. Coincide con la componente *x* o del *main axis* del elemento flex, es decir: con **width** cuando **flex-direction: row;** (por defecto) y con **height** cuando **flex-direction: columna;**. Además, define a partir de qué "**width**" o "**height**" empiezan a verse los efectos del **flex-shrink** y **flex-growth**.
- **flex**: es la agrupación de las 3 anteriores. Shorthand:
  - Si le asignamos un único valor numérico, se aplica la propiedad **flex-grow**.
  - Si se asigna un único valor numérico acompañado de unidades, se aplica la propiedad **flex-basis**.
  - Si se asignan dos numéricos, se aplican **flex-grow** y **flex-shrink**.
  - Si asignásemos dos valores, uno numérico y uno acompañado de unidades, aplica las propiedades **flex-grow** y **flex-basis**.
  - Si se asignan tres valores, dos numéricos y uno acompañado de unidades, funcionaran las propiedades **flex-grow**, **flex-shrink** y **flex-basis** (en ese orden).
- **order:** similar al **z-index** pero funciona en el *main-axis*. Especifica el orden utilizado para disponer los elementos en su contenedor flexible en la dirección mencionada; en caso de tener el mismo valor se ordenaran de acuerdo al DOM.

GRID:

Conceptos:

- **grid**: es un valor de la propiedad display. Es un estilo de layout. Otorga un comportamiento de bloque pero su contenido se seccionará como una grilla.
- **Grid container**: es el contenedor que tiene la propiedad **display: grid;**
- **Grid item**: es como se denomina a cada elemento que esté incluido en el contenedor. Estos necesariamente deben ser primogénitos del grid container, caso contrario no lo son (herencia similar a flex).
- **Grid cell**: es cada una de las subdivisiones de la grilla.
- **Grid tracks**: cada fila (row) o columna (columna).
- **Grid área**: es una sección del **grid** mayor a una celda definida por el desarrollador, necesariamente tienen que ser celdas consecutivas que formen un área rectangular o cuadrada.
- **Grid line**: son las líneas que delimitan las celdas. Horizontales son **row line** y verticales son **columna line**.
- Los **grid items** conformaran las celdas de acuerdo al siguiente criterio (por defecto!):
    - Primero se completará en dirección horizontal hasta completar la línea a través de las columnas que existan.
    - Al completarse una línea se continúa con la segunda en el mismo orden.
    - En caso de no completarse una línea, existiendo espacio definido para celdas (no ocupadas) esa sección estará vacía.
    - En caso de definir que una **celda** ocupe un **área grid** entre dos filas o columnas ya ocupadas para la columna o fila en cuestión, respectivamente, este empujará a los otros dejando un espacio vacío:



- Se pueden manejar unidades específicas:
    - **auto**:
    - **fr** (fraction): es el equivalente al **flex-grow** pero en grid. Luego de implementarse los tamaños asignados, en medidas definidas (px, rem, em, etc.) a las filas/columnas, el espacio restante se distribuye según los valores fr asignados (actúan como proporciones).
- **Grid explícito**: es aquella parte del grid que tiene características definidas (asignadas por código)
- **Grid implícito**: es aquella parte del grid que no tiene características definidas además de las default (puede ser por omisión de código o por creación de filas/columnas a modo de excedente, ver criterios anteriores).

Comentado [SF2]: Asumo esto en base a observación experimental. Chequear.

- **Propiedades**:
    - **display: grid;** : es el atributo de la propiedad display que otorga características de grid a una caja.
    - **grid-template-rows**: define la cantidad de filas, y su respectiva altura. que tendrá la grid de acuerdo a la cantidad de parámetros (medidas altura) que se asignen. Se aplica al contenedor grid.
    - **grid-template-column**: define la cantidad de columnas, y su respectiva anchura. que tendrá la grid de acuerdo a la cantidad de parámetros (medidas ancho) que se asignen. Se aplica al contenedor grid.
        - Ejemplo para ambos **template**s:
            - **grid-template-row/column: 120px 30px;** : esto indicaría que la grid tendrá 2 filas/columnas de 120px y 30px de ancho respectivamente.
            - **grid-template-row/column: repeat(n, medida ancho/alto);** : esto facilita el dibujo, literalmente repite **n** veces lo que está después de la **,**, esto puede ser tanto una única medida como varios valores de distintas medidas. Por ej. **repeat(3, 160px 1fr)** implicaría escribir **160px 1fr 160px 1fr 160px 1fr**, de manera que obtenemos una especie de **grid** con características periódicas.
        - Si quisiéramos que todas tengan el mismo ancho podríamos darles el parámetro **auto** a la cantidad de filas/columnas que deseásemos.
        - También existe la posibilidad de nombrar las líneas tanto de rows como de columns al utilizar esta propiedad. De manera que funcionen como "variables". Ej. **grid-template-rows:**
          [first-line]
          1fr
          [second-line]
          1fr
          ;
          De esta manera luego al aplicar grid-row a un grid item puedo utilizar estos nombres para asignar el grid área en lugar de la numeración por defecto.
    - **grid-template**: es un shorthand para las propiedades anteriores. Funciona así: **grid-template: rows / columns;**.
    - **grid-row-gap**: define la distancia que habrá entre cada celda en dirección vertical (entre filas).
    - **grid-column-gap**: define la distancia que habrá entre cada celda en dirección horizontal (entre columnas).
        - **grid-gap**: es un shorthand de las anteriores. Si asigno una única medida aplica para ambas propiedades anteriores; si aplico dos, la primera corresponde a **grid-row-gap** y la segunda a **grid-column-gap**.
    - **grid-column**: defino el **grid-area** que ocupará el **grid item** en función de las **column lines** es decir le digo extiéndete desde esta línea a esta línea. Es importante recordar que **column line != column** (una **column** está delimitada por dos **column line**). Esto no implica que ocupará el lugar de dos **celdas** sino que

implica una extensión de la **celda**, es decir, la celda se extenderá en la dirección asignada "empujando" a las otras celdas hacia nuevas líneas (ver criterios de llenado de grid un poco más arriba) que carecen de formato definido (medidas estándar).

- **grid-row**: defino el **grid-area** que ocupará el **grid item** en función de las **row lines** es decir le digo extiéndete desde esta línea a esta línea. Es importante recordar que **row line != row** (una **row** está delimitada por dos **row line**). Tener en cuenta misma consideración que para **grid-column**.
    - Ambos son shorthands de las propiedades **grid-column-start/end** y **grid-row-start/end** respectivamente.
    - Sintaxis de los anteriores:
        - **grid-row/column: n_0 / n_f;**
        - **grid-row/column: n_0 / span m_f;** en este caso el **span** indica la cantidad de filas/columnas (y no de líneas) que debe abarcar iniciando desde la fila/columna **n_0**.
- **grid-auto-rows**: se utiliza para cambiar las características (altura) por defecto de las filas que se generen como **grid implícito**.
- **grid-auto-columns**: se utiliza para cambiar las características (ancho) por defecto de las columnas que se generen como **grid implícito**.
- **grid-auto-flow**: esta es una especie de **flex-direction**, en el sentido de que cambia la manera en la que se ordenaran los elementos hijos (invierte criterio mencionado arriba!) y el algoritmo de la grid de localizar automáticamente el exceso de celdas desplazado como nuevas filas (default).
    - Valores:
        - **row**: las celdas desplazadas se ubicarán en una nueva fila. Asignado por defecto.
        - **column**: las celdas desplazadas se ubicarán en una nueva columna.
        - **sparse**: es el algoritmo por default de orden ante la presencia de espacios vacíos y simplemente desplaza/empuja las celdas dejando los espacios. Mantiene el orden de los elementos.
        - **dense**: este aplica un algoritmo que ante la presencia de espacios vacíos los rellena con la siguiente **celda** de **área** no definida. Puede desordenar los elementos.
        - **row-dense**: es similar al dense pero ordena dando prioridad a completar primero una fila antes de pasar a la siguiente.
        - **column-dense**: idem anterior pero da prioridad a completar una columna antes de pasar a la siguiente.
- **grid-template-areas**: es muy útil para definir un layout. Permite definir áreas que tomará cada elemento a través de palabras claves o identificadores. Es importante notar que estas palabras claves pueden ser cualquiera y no necesariamente tener el nombre del elemento, aunque es aconsejable que de un indicio claro. Por ej.:

- **grid**-template-areas:

"header header header header"
"section section section aside"
"section section section aside"
"section section section aside"
"footer footer footer footer";

Esto es un grid de 4x5 dónde al distribuir las áreas definidas a los distintos elementos (header, section, aside y footer en este caso) nos devolvería un layout del tipo:

HEADER	
SECTION	ASIDE
FOOTER	

RESPONSIVE DESIGN: La idea detrás de este concepto es que la página responda de manera distinta, o se adapte, a distintas condiciones de medios para que el funcionamiento se vea optimizado en los distintos dispositivos.

@media se utiliza para aplicar media querys, que en esencia son "consultas" para determinar distintas condiciones del dispositivo:

- Ancho y alto de la ventana gráfica
- Ancho y alto del dispositivo
- Orientación
- Resolución

Para ello se pueden utilizar distintos comandos:

- **all**: apto para todos los dispositivos
- **print**: apunta al material impreso y visualizaciones de documentos en una pantalla en el modo vista previa de impresión
- **screen**: destinado principalmente a pantallas
- **speech**: destinado a sintetizadores de voz.

Operadores:

- **and**: se utiliza para definir condiciones necesarias y simultáneas (intersección de conjunto) para aplicar las ciertas propiedades.
- **or**: se utiliza para definir condiciones necesarias pero no necesariamente simultáneas (unión de conjunto), es decir, puede cumplirse solo una y es suficiente para que se apliquen las propiedades deseadas.

Orientaciones:

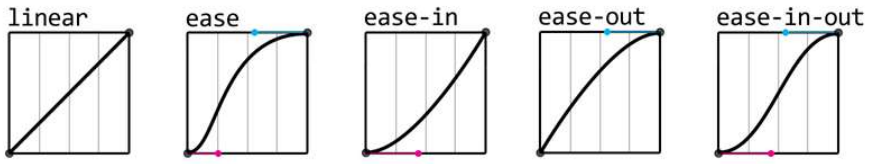
- Landscape: el ancho supera la altura, normalmente desktop.
- Portrait: la altura supera al ancho, normalmente mobile.

Enfoques:

- Mobile first: se basa en diseñar primeramente la página web para dispositivos móviles y posteriormente complejizarla hacia resoluciones mayores.
- Content first: esta idea se basa en el contenido, no en el diseño per sé sino en analizar qué tendrá mayor repercusión en términos de marketing. Se maquetará y ordenará sin seguir un orden particular asegurando que aquellos "must" del marketing estén presentes.

TRANSITIONS: la **transition** es una propiedad que muestra el proceso mediante el cual ocurre un cambio en la propiedad de un elemento ante la ocurrencia de un evento (hover, focus, etc.), sea un cambio de posición o el color, o de lo que se modifique en cuestión.

- Propiedades:
  - **transition-property**: se utiliza para indicar qué propiedad es aquella cuyo proceso de transición se quiere mostrar. Por ej. **transition-property: background;** se utilizaría para indicar que se quiere mostrar la transición en el cambio del background de un elemento. En caso de querer que todas las propiedades modificadas tengan una transición se puede emplear el valor **all**.
  - **transition-duration**: se aplica para definir el tiempo que se desea que dure tal transición.
  - **transition-delay**: se utiliza para retrasar la animación un tiempo determinado luego de que ocurra el evento en cuestión.
  - **transition-timing-function**: define la curva de velocidad de la transición.
    - **linear**
    - **ease**
    - **ease-in**
    - **ease-out**
    - **ease-in-out**



- **transition**: es el shorthand para las anteriores. La sintaxis consiste en lo siguiente: **transition: property duration timing-function delay;**. En caso de ser necesario aplicar a dos propiedades estas son separadas con una ",".

ANIMACIONES:

@keyframes es el comando que se utiliza para crear animaciones, es necesario nombrar la animación seguido del mencionado (por ej. **@keyframes nombre-de-la-animación**. Para asignarla a un elemento se utilizan distintas propiedades:

- **animation-name**: esto asigna al elemento la propiedad.
- **animation-duration**: define el lapso de tiempo que durará la animación.
- **animation-timing-function**: valores y funcionamiento igual a transition.
- **animation-iteration-count**: define cuantas veces se repetirá la animación, es posible aplicarle un número muy grande o el valor **infinite** en caso de querer loopearla.
- **animation-direction**: invierte el sentido de la animación.
    - **normal**: este valor es el que está definido por defecto.
    - **reverse**: invierte el sentido de la animación.
    - **alternate**: alterna el sentido de la animación.
    - **alternate-reverse**: es la combinación de los dos anteriores.
- **animation-fill-mode**: define como finalizará la animación.
    - **none**: al finalizar vuelve a la posición inicial, en las condiciones iniciales.
    - **backwards**:
    - **forwards:** al finalizar se mantiene en la posición final, en las condiciones finales de la animación.
    - **both**: inicia la animación desde las características iniciales asignadas al principio de la misma (no necesariamente es la misma que la del elemento antes de iniciarse).

Para crear la animación se parte desde el **@keyframes**. Su traducción sería algo así como "cuadros claves", o sea, puntos especiales de la animación donde ocurrían cambios en propiedades.

Para ello se pueden definir de dos formas:

- from – to (0 % y 100%, o inicio y final):

```
@keyframes nombre-animacion {
        from {
                propiedad: valor-inicial;
                propiedad2: valor-inicial;
        }

        to {
                propiedad: valor-final;
                propiedad2: valor-final;
        }

}
```

- Con porcentajes que representan una fracción de la duración de la animación asignada, por ej. 50% genera un **keyframe** a la mitad de la animación. De esta manera se logra más versatilidad y más fluidez respecto de la anterior.
    - **@keyframes nombre-animación {**
        0% {
            propiedad: valor-inicial;
            propiedad2: valor-inicial;
        }

        10% {
            propiedad: valor-10%;
            propiedad2: valor-10%;
        }

        … % {…}

        100% {
            propiedad: valor-final;
            propiedad2: valor-final;
        }

    }

TRANSFORMACIONES: **transform** es una propiedad que aplica una *función* a un elemento para transformarlo, pueden ser movimientos en el espacio o distorsiones, entre otras cosas. Esta propiedad es muchísimo más optima que otras como **left**, **top**, etc. utilizadas para posicionar, dado que la transformación ocurre "en una capa superior" a diferencia de las anteriores que lo que hacen es modificar e interferir con el layout, por esta razón requiere de menos recursos para producirse. Todas estas transformaciones tienen un origen definido por defecto que suele ser el centro del elemento, pero este es modificable a través de la propiedad **transform-origin**.

- **translate()**: es una función/valor shorthand de la propiedad **transform** que desplaza al elemento e incluye a las mencionadas a continuación (los valores de la función se introducen separados mediante una "**,**" y la introducción de un solo valor aplica en dirección *x*. **translate()** admite tanto valores numéricos de unidades como porcentajes, siendo estos últimos en relación al tamaño del elemento transformado.
    - **translateX()**: es una función/valor de la propiedad **transform** que desplaza al elemento horizontalmente.
    - **translateY()**: similar pero desplaza verticalmente.
- **scale()**: es una función/valor shorthand de la propiedad **transform** que escala (expande o contrae) al elemento.
    - **scaleX()**: aplica el escalado solo en la dimensión horizontal del elemento.
    - **scaleY()**: aplica el escalado solo en la dimensión vertical del elemento.
    - **scaleZ():** aplica el escalado solo en la dirección normal al elemento.
- **skew()**: es una función/valor de la propiedad **transform** que mueve los vértices del elemento en un ángulo determinado, dejando fijo el centro del elemento. A fin de cuentas,

tuerce y distorsiona al elemento. A diferencia de las anteriores, los valores admitidos para esta función son grados (degrees) y se especifican como: **90deg**, por ejemplo, significando 90°. También admite radianes y gradianes.

- **Rotate()**: es una función/valor shorthand de la propiedad **transform**, que rota al elemento sobre un plano (eje designado perpendicular al mismo) en el ángulo asignado a la función. También admite valores en grados, radianes y gradianes. Incluye a las funciones:
    - **rotateX()**: rota sobre el eje horizontal.
    - **rotateY()**: rota sobre el eje vertical.
    - **rotateZ()**: rota sobre el eje normal al elemento.

VARIABLES: CSS permite almacenar variables. Estas son sencillamente contenido que se almacena en una memoria. Existen dos tipos:

- Globales: pueden utilizarse desde cualquier selector/elemento. Para definirlas es necesario aplicarlas de la siguiente manera dentro del .css:
    - **:root {**

      --nombre-variable: valor-variable;

      }

Para utilizarlas simplemente basta con utilizar la variable como valor de una propiedad a la que se desee aplicar:

- **.classElement {**
  background: var(--nombre-variable);

  }

Esto equivale a aplicar la propiedad **background: valor-variable;**.

- Locales: las variables locales son aquellas que se definen dentro de un selector/elemento particular y solo funcionan dentro de dicho selector. Además en caso de haber definido la variable con el mismo nombre que una global, dentro del elemento en el cual se haya definido tendrá más jerarquía la segunda; y no será efectiva por fuera de dicho selector/elemento.

FILTROS: Se aplican mediante la propiedad **filter**. La propiedad, de la misma manera que **transform**, aplica funciones y por lo tanto admite combinaciones lineales que potenciarían o mezclarían los efectos (ej. **filter: drop-shadow(0px 1px 1px #000) brightness(1.5) blur(1px);**. Esta propiedad admite distintos valores:

- **none**: es el valor por defecto de la propiedad.
- **blur(unit)**: este valor desenfoca/blurrea la imagen en la medida que se asigne (desenfoque gaussiano). No admite %.
- **brightness(0-n)**: es una escala de brillo donde el 0 implica que la imagen se vea prácticamente negra (brillo 0%), el 1 equivale al brillo original de la imagen y valores superiores añadirán brillo. Admite % ya que trabaja con relaciones, no con medidas.

- **contrast(0-n)**: es una escala de contraste, los valores que admite funcionan de la misma manera que el anterior. Admite %. El contraste es la diferencia entre los puntos más y los menos brillantes o iluminados.
- **drop-shadow**: es similar al box shadow, con la diferencia de que mientras que el box-shadow aplica una sombra a una imagen sobre sus bordes, drop-shadow aplica sombra a la imagen sin tener en cuenta sus border (se aplica a png para dar sombra al dibujo). Su sintaxis es **drop-shadow(offset-x offset-y blur-radius color).** Si se aplicase varias veces se "potenciaría".
- **grayscale(0-1/100%)**: aplica una escala de grises al elemento, 0 implica mantener el color original y 1 convierte el elemento completamente a b&w. Admite %.
- **hue-rotate(0-360º)**: este filtro cambia la gama de colores (hue = tono?). Admite grados, radianes y gradianes.

- **invert**(): no necesita un valor dentro definido (por defecto será 1 o 100%), este filtro invierte los colores del elemento. Si se selecciona un valor intermedio se moverá entre los dos extremos (pensar en los extremos como números opuestos, el 50% de ese rango siempre valdrá 0 y por tanto obtendríamos un elemento en b&w).
- **opacity()**: este filtro afecta la transparencia del elemento. El valor 0 implica que será completamente transparente, mientras que el valor 1 mantendrá su visual estándar.
- **saturate()**: este filtro concentra los colores o reduce su intensidad. El valor 1 o 100% deja el elemento sin cambios, valores inferiores decoloran y valores superiores concentran los colores.
- **sepia()**: esta función lleva los colores a tonos de sepia. El valor 0 mantiene al elemento inmutado mientras que 1 representa el sepia total.

Se pueden combinar distintos filtros con animaciones para crear efectos realmente interesantes.

El material es recopilado de distintas fuentes siguiendo el orden de conceptos que se exponen en los cursos de **html** y **css** de "Soy Dalto" (https://www.youtube.com/c/soydalto/videos), MDN Web Docs, curso de desarrollo web de CODERHOUSE y otras fuentes al azar que resultaron útiles para profundizar sobre el funcionamiento de los recursos.