Css

Códigos principales Css

\*Para enlazar Css con Html se utiliza un link entre archivos así los estilos influyen sobre el texto de la página web.

**Link+tab=** nos deja con **<link rel=” stylesheet” href=”css/(**nombre del archivo de la hoja de estilos**)” .**

**!important** No se toma en cuenta la cascada de código y se mantiene.

Comentario

**/\* “(comentario)” \*/**

Id o Atributo

\*Es recomendable utilizar las denominadas Utilidades, clases con líneas de código específicas que puedes reutilizar en diferentes etiquetas.

\*Se utiliza la id para poder dar un valor personalizado a cada etiqueta así generar su estilo.

**Id:”(nombre de la id)”** Este es mucho más específico y solo se puede usar una vez por pag. Con este podemos darle clase a un div, section, header, etc y con un link y #(nombre de id) nos lleve a tal punto.

En ccs: **#” (nombre de la id)”.**

**Class: “(nombre de la clase)”** Se puede reutilizar incluso en una misma etiqueta.

En ccs: **.”(nombre de la clase)”.**

Códigos de Estilos CSS

\*Nombra los hijos de la etiqueta padre en la clase de css asi utilizar menos codigo

\*Los atributos de CSS **margin y padding**, aunque en algunas ocasiones puedan comportarse de manera parecida en la práctica, son muy distintos. **Margin** es el margen que hay desde un elemento hasta los que tenga al lado. **Padding**es el espacio que hay en un contenedor entre el contenido y los bordes del contenedor.

\*Se pueden separar inputs con comas, como realizar un **:not(type=”[]”)** y que no se aplique a tal elemento.

\*Para los Botones se debe pasar de **inline** a **inline-block**  asi poder colocarle margin.

\*Crear clases sobre determinados hijos así no se afectan la tercera línea de artículos.

\*Utilizar rems en vez de **px** así todas las pantallas tienen las mismas medidas.

\*Utilizar Css Normalizer para obligar al navegador a usar ciertos valores.

\*Colocar tus estilos de Google Font u otros al final de la cascada en el Head.

\*Se puede usar RBA color para generar transparencia

\*Se puede poner dos clases con selectores específicos separándolos con coma.

\*El **padding** puede sumarse a la anchura de **width**.

\*Pseudoselector se utiliza poniendo dos puntos junto a una clase o etiqueta.

\*Para que fluya la página mientras se agranda o achica utilizar porcentajes en el **width**.

\*Con hover también se puede cambiar el color de fondo del botón cuando esta el mouse sobre él.

\*Los background images son definidos por otros objetos, para que aparezcan darles un min-height de algún valor.

\*Con un > y elemento: **p, li, ul,** etc se le aplica solo al primer nivel de hijos que se encuentre.

\*Se puede poner dos clases juntas con puntos ej. .**Clase1.Clase2** para que se aplique la primera clase y que la segunda tenga otro tipo de estilización pero se mantenga la primera.

\*Para ingresar una imagen a un texto que este sobre el o en su caja de elementos utilizar un **before** o **after** y un **position relative** y **absolute.**

**../:** Cambio de directorio para que no se tome la misma carpeta.

**:Hover**: (pseudo-clase) Se utiliza con dos puntos:y sirve para cuando el usuario utiliza el cursor sobre tal elemento suceda tal estilización. Utilizarlo con transition le da más naturalidad. Puede ser puesto sobre la clase y afectar a dos elementos a la vez, como un **img**  o un **p**, o solo en ese elemento y aplica individualmente.

**:Nth-child(** pseudo-clase, número del div con la clase**);** Pseudoselector con el cual elegimos el número del hijo con tal clase y le aplicamos tal css.

**:Root {}** Con esto se pueden crear variables que luego se utilizan con **var(**nombre de la variable**)**. Dentro del root se pone el nombre de la variable con dos **--ejemplo:** y el código solo de estilización para referirse con tal nombre, es decir solo el color o la fuente o el número, NO el código que es el general de la categoria.

**::After:** Pseudo-elemento, se coloca con DOS dos puntos, este lo que hace es crear un elemento que no se puede seleccionar luego de la clase o etiqueta elegida.

\*SE DEBE CREAR SI O SI UN **CONTENT** pero va vacio.

\*La hoja de estilos debe saber cuánto va a medir.

\*Se puede utilizar para imágenes para cambiar su transparencia.

\*Para que un hijo sea más importante que el after se debe poner en ese un **position:relative;** y luego **z-index:1;** sin el **::after**.

**::Before:** Pseudo-elemento, se coloca con DOS dos puntos, este lo que hace es crear un elemento que no se puede seleccionar luego de la clase o etiqueta elegida.

\*SE DEBE CREAR SI O SI UN **CONTENT** pero va vacio.

\*DEBE TENER HEGIHT Y WIDTH.

\*Se puede utilizar para imágenes para cambiar su transparencia.

\*Suele necesitar que el contenido inicial tenga un **position:relative** y el ::**before** un **absolute**.

\*Para que un hijo sea más importante que el **before** se debe poner en ese un **position:relative;**

**Align-items:** Aliña ítems en el centro con flexbook. En flex-row los aliña en la línea y. en flex-column en línea x.

**Background-atachment: fixed**. Le da un efecto de movimiento a la imagen, es decir, cuando se mueve la página se ven otras partes que antes no se podían ver.

**Background-image: url (“ubicación de imagen”).**

**Background-repeat: no-repeat;** Para que no se repita la imagen.

**Background-size: cover;** toma las proporciones de la imagen y proporciona el tamaño de ella.

**Background-position: posición del fondo.**

Etiquetas

1. **Background-position: center center;** posición centrada del fondo con respecto a la página.

**Body:** Aplica la estilización a todos los contenidos. Contiene°

**Border**: Crea bordes. Puede tomar tres valores, width, style y color, en orden.

**Border-radius:** Le da un borde redondeado según el porcentaje dado.

**Cursor: pointer;** Obliga a la pag a utilizar el cursor de link sobre tala elemento.

**Cursor: pointer;** Para ponerle el pointer de link al hover de un submit.

**Display: Block;** el input toma todo el espacio disponible en su linea definido por las otras clases

**Display: inline-block;** Esto nos deja estilizar un link de diferentes formas ya que antes estaba automáticamente en css en línea. Si sería solo en block toma todo el lugar que tiene para formar el botón.

**Display: Table.** Los ponemos en tabla y con **margin-let: auto;** y **margin-right: auto;** se centrar

**Filter: gray;** Da un color grisáceo a una fotografía.

**Filter: brigthness ().** Le da brillo o le saca dependiedno del numero dentro de los ()

**Font-family**: fuentes.

**font-size:** Determina el grande de las letras. Usar **vw** para reescalar acordemente.

**Font-weight**: Anchura de la font (la tiene que soportar tu fuente, o tu google fonts)

**Height**: **100%** Esto se utiliza en el hijo así no sobrepasa al padre si se utiliza un flexbox

**Height:** Altura de un área por px (pixeles) o vh (view per height, es decir que se automatiza a cada computadora por igual, funciona para imágenes de background).

**Html {font-size: 62.5%}:** resetea los rems para que un rem sea 1º pixeles.

**Input [type=” (nombre del input)”;]** Esto nos deja elegir cada tipo de inputs que querramos estilizar.

**Justify-content**: Posicionar automáticamente horizontalmente de alguna forma el contenido.

Etiquetas

1. **Space-around:** Distribuye los espacios igualmente.
2. **Space-between**: distribuye igualmente los espacios pero cada elemento en la derecha y en la izquierda. Si esta en modo columna es arriba y abajo
3. **Flex**-**end:** Empuja el elemento a la derecha
4. **Center:** Para centrar los elementos. En flex-row los centra en la línea x. En flex-column los centra en la línea y.

**Last-of-type:** (pseudo-selector) Indica en la clase de css con solo el último elemento hijo de la etiqueta padre.

**Line-height**: interlineado. (Se pueden poner números solos y representa renglones). Con px se puede usar para centrar los elementos.

**List-style**: cambia el estilo de las listas.

Luego se debe conectar una clase con el pseudo-selector hover y la nueva medida.

**Max-width:** Altura máxima o anchura de un texto/imagen/lista.

**Min-heigth:** Altura mínima de un texto/imagen.

**Margin-left/right/top/bottom:** Genera espacio automáticamente entre elementos de forma proporcional en la posición que le des, se da valor con px o vh. Se pueden usar valores negativos.

Etiquetas

1. **Margin: “(arriba)” “(derecha)” “(abajo)” “(izquierda)”** Se puede utilizar de esta forma y darle valores como automático y 0 para simplificar (si se repite se puede acortar).

**Opacity:** opacidad del elemento con 0 desaparece.

**Overflow: hidden;** Esconde todo contenido que se pase sobre una medida que eh puesto, como **max-height: 0;.** Puede ser puesto sobre solo un eje, como el x o el y.

**Padding**: Agrega espacios hacia los contenedores, es decir las cajas, para un texto o imagen. Si esta solo se aplica en las cuatro direcciones, sino especificar. Puede remover las líneas de listas con 0

**position-relative /** **Position-absolute:** Estos dos códigos, el primero en el padre y el segundo en el hijo, nos deja mover un elemento teniendo en cuenta al padre dependiendo de a cuantos pixeles pongamos su dirección (top, right, bottom, left). Con 0 en todos podemos hacer que la imagen tome todo el espacio disponible, o se posicione justo donde inicia el padre si es solo top u otro. Con los elementos de oposición puedo crear efectos con **hover** y **transition** para que desaparezca un elemento sobre otro. Por ejemplo **bottom: -80;**

**Target:** (pseudoselector) Nos deja activar una acción cuando se targetea la clase. Se le debe poner una id a la clase y utilizar un link con **a** con **#** y el mismo nombre de la iden el href. Se selecciona en css poniendo el id acompañado en su inicio con un #, ejem: **#target**.

**Text-align**: center; centra texto e imagen

**Text-decoration: none;** quita las líneas bajo los links de navegación.

**Text-transform**: uppercase/capitalize/lowercase.

**Transform: scale(1.1):** Se utiliza para escalar y crecer un objeto dentro de su propio bloque. En las fotos le da cierto zoom.

**Transform: rotate(**número **deg)** Gira el objeto un cierto ratio.

Tiene 3 valores que van en el mismo código, (el elemento que se quiere cambiar) (el tiempo en **s** seg) y (como se debe hacer, **ease in out** es una de ellas).

**Transition:** Nos deja cambiar un valor mientras transporta el hover por ese elemento.

**Video:** Con esto agregamos video al html. Dentro de tal se utiliza un **source** con el nombre del video con **src=video/”.mp4”.** Se le ingresa también un **type=”video/mp4** (si ese es el caso)**”**. Se le debe poner otros tipos de video, **webm** u **ogv** así cubren mas dispositivos.

Etiquetas

1. **Autoplay:** Se prende solo.
2. **Loop:** Se queda en un loop infinito.
3. **Poster:** Pone una imagen en algunos dispositivos. Se pone entre comillas dobles el nombre de la imagen.

Códigos Grid

\*Los display Grid se heredan, es decir cuando se utilizan en el padre no hay que utilizarlo de vuelta.

**Display: grid;** Nos deja utilizar comandos grid. Se utiliza sobre el padre y solo afecta al primer nivel de hijos.

**Grid: Unset;** Se desactiva el grid si se pone en el hijo.

**Grid-template-columns: repeast (2, 1fr[[1]](#footnote-1));** Pone en columnas los hijos del padre, se puede utilizar porcentajes para ver sus tamaños. Con **auto** toma el tamaño que requiere el elemento.

**Grid-template-rows:** Lo mismo que en columnas, pero los coloca en el eje x.

**Grid-auto-row:** Lo mismo que el superior pero coloca automáticamente las columnas. Con **minmax (min-content, max-content)** se arregla el error de chrome que nos soperpone las columnas dándole el valor minimo y máximo automatico.

**grid-column: 1 / 3;** Le indica a la columna 3 que se expanda desde la 1 hasta la 3 visualmente, se puede usar también para mover la columna a un lado de otra. Se utiliza en el hijo.

**Grid-row:** Es lo mismo que el column pero este maneja las columnas x no las y.

**Gap:** Espacio entre columnas

Etiquetas

1. **Row-gap:** Espacio hacia abajo entre columnas.
2. **Column-gap:** Agrega espacios horizontalmente entre columnas.

Codigos Flexbox

**Display: flex;** Se utiliza en una clase padre para que la primer clase de hijos puedan ser acomodados dentro de ella.

**Flex-wrap: wrap;** evita que los elementos se pongan en línea y los mete en columnas si se sobrepasa el tamaño normal de la pag web. Se utiliza en el padre.

**Flex:** Modificar el tamaño o cómo se comporta la caja de flexbox. Sus etiquetas se pueden juntar en una sola (grow, shrink, basis).

Etiquetas

1. **Flex-grow:** aumenta el tamaño base por una cantidad determinada. Con 1 crece todo lo que podría crecer.
2. **Flex-shrink: 0;** esto evita que se encoge si no tiene espacio y solo se tome el flex-basis. Se la pone en el hijo porque es quien escoge su tamaño.
3. **Flex-basis**: Anchura. Se recomienda usas calc(calculo) así se puede establecer un porcentaje de cada artículo y restarle una medida así se separan. Tamaño base principal (main size base) significa que no necesariamente ese será su tamaño al dibujarse por el navegador, pero que será un punto de partida para calcular el tamaño final

**Order:** Posiciona el objeto al número que le indicas. Flexbox pone un número a cada hijo en orden creciente.

**Flex-direction:** Nos deja modificar el curso de los objetos. Con reverse sobre la etiqueta (**row-reverse)** lo ponemos en el sentido contrario. Con **column** se colocan uno sobre otro.

Hacer Responsiva la Pagina Web

El diseño debe adaptarse a las interacciones del usuario y las resoluciones que utilizan. Se debe adaptar a cualquier plataforma electrónica.

Se utiliza Media Queries para esta tarea.

Una forma de hacer responsive tu pag web, y sin queries, es con este código:

**.contenedor {**

**max-width: 1200px; margin: 0 auto**

**}**

**img {**

**max-width: 100%;**

**}**

Media Querie

Se utiliza en la hoja de estilos css.

Syntaxis: **@media (min-width: 500px;[[2]](#footnote-2)/ max-width:500px;[[3]](#footnote-3)) {**

**Body[[4]](#footnote-4) {**

**Backgroun-color: blue;[[5]](#footnote-5)**

**}**

**}**

Snippet

Se utilizan los snippets para crear códigos más cortos que puedan ser usados en css y no ocupen tanto tamaño.

\*Se debe siempre usar doble comillas para crear el código en un archivo json del snippet.

Su sintaxis es:

**“media queri”[[6]](#footnote-6): {**

**“prefix”[[7]](#footnote-7) : “mq”;**

**“body”[[8]](#footnote-8) : [**

**“@media (min-width: $1[[9]](#footnote-9)) {\n\t$0\n}[[10]](#footnote-10)”**

**]**

**}**

Floats

Con este tipo de comando se pueden ubicar todo tipo de elementos de forma mas fácil en usa sola línea, posicionándoles en la derecha o la izquierda.

**Float: left;** Se utiliza para posicionar el elemento a la izquierda de otro, respondiendo ante su **width** o **padding** y **margin**.

**Float: right;** Se utiliza para posicionar el elemento a la derecha, respondiendo ante su **width** o **padding** y **margin**.

. Se usa **Text-align: right;** para que se posicione totalmente a la derecha.

\*SE DEBE USAR CLEARFIX PARA QUE FUNCIONE DADO A BUG:

.clearfix:before,

    .clearfix:after {

        content: " ";

        /\* 1 \*/

        display: table;

        /\* 2 \*/

    }

    .clearfix:after {

        clear: both;

    }

**Webkits**

**-webkit-:** Es un prefijo de los códigos que nos permite crear ciertos diseños a travez de las herramientas de cada navegador. **-moz-** Es el de mozilla, **-o-** Opera, etc.

TODOS ESTOS CODIGOS PUEDEN SER USADOS SIN EL WEBKIT PERO TAL VEZ NO SEAN APLICADOS EN TODAS LAS PLATAFORMAS.

Codigos WebKit

**-webkit-appearance: none;** Esto se utiliza en los select de los formularios para evitar los prefijos y que se apliquen los estilos.

**–webkit-filter-: grayscale(1);** este se aplica en la mayoría de navegadores web un color grisaceo.

**-webkit-transform: scale(1.1):** Se utiliza para escalar un objeto dentro de su propio bloque. En las fotos le da cierto zoom.

**-webkit-transform: rotate(**numero**deg)** Gira el objeto un cierto ratio.

Crear una clase que se active solo en una pagina

Esto nos puede servir para activar algo a través de una clase cuando solo estamos en una página específica. Esto se hace dándole una clase al body de tal pag. y luego indicando otra clase llamada activo en la que diga que debe hacerse.

Luego en css llamamos primero a la clase del body, luego a la clase del elemento y por ultimo a su clase activo.

1. Repite dos veces 1fr: fracción que divide a la pantalla en columnas, ej: (2, 1fr= ½ + ½) [↑](#footnote-ref-1)
2. Movile first: Medida que hará que se active mq al momento que la pantalla llega o sobrepasa, de teléfono a escritorio. [↑](#footnote-ref-2)
3. Desktop first: Medida que hará que se sactive mq al momento que la pantalla llega o se rebaja, de escritorio a teléfono. [↑](#footnote-ref-3)
4. Selector. [↑](#footnote-ref-4)
5. Estilo para el selector. [↑](#footnote-ref-5)
6. Nombre del snippet. [↑](#footnote-ref-6)
7. Como lo voy a referenciar en el css. [↑](#footnote-ref-7)
8. La línea de código que se va a utilizar [↑](#footnote-ref-8)
9. Indica que debe dejarse un espacio ahí. [↑](#footnote-ref-9)
10. Indica que deje un espacio en el codigo [↑](#footnote-ref-10)