**14. Creando textos con HTML**

**Etiquetas Heading  
<h1>** resalta lo más importante**</h1>**

Rango: <h1> a <h6>

Se recomienda solo tener un h1 por documento HTML

Un generador de texto **Lorem Ipsum** para rellenar diseños o maquetas.

**15. Estructurando contenido en HTML.**

**Etiquetas:**

<header> Parte superior

<footer> Parte inferior

<nav> Menú de navegación

<main> Contenido Principal

<section> Sección. Si el primer elemento hijo es un heading se usa un section forzosamente. Es decir cuanto hay un nuevo hx. **Excepción**: Solo si es el contenido principal.

<article>

<aside>

<div> Se usa cuando no se puede usar ninguno de los otros.

**16. Creando Enlaces y Navegación.**

<nav> ***Aquí se colocan los enlaces*** </nav>

**Para menús de múltiples niveles se usa:**

<ul>

<li> </li>

</ul>

**Enlaces:**

<a href=”#”> Texto </a> Se usa un # para simular un enlace que aún no se tiene.

**17. Añadiendo imágenes en HTML**

Se recomienda usar Tabler Icons, Hero Icons, Font Awesome.

Los íconos se recomiendan en Scalable Vector Graphic (SVG)

Etiqueta <img src””> No tiene cierre

**18. Formularios en HTML**

Debemos usar etiqueta

<form> </form> para indicar que es un formulario

<fieldset>

<legend> Describe el formulario </legend>

<label>

</fieldset>

Tipos de input: checkbox, radio, text, tel, email.

<input type=” xx ” placeholder”aquí un texto en el campo antes de escribir”>

<textarea> </textarea> para mensajes donde el usuario escribirá libremente

**19. El rol de CSS.**

Colores, tamaños espacios o animaciones.

Son hojas de estilo en cascada que permitirán darle al código HTML un diseño especial y único en pocas líneas, se encarga de:

* Tamaños y tipos de fuentes
* Colores
* Espacios
* Márgenes
* Adaptar diseños a distintos dispositivos
* Animaciones

**Sintáxis:**

p{

color: blue;

}

Selector: **p** está seleccionando un párrafo

Propiedad: **color** es una propiedad.

Valor: **Blue** es un valor único que nosotros asignamos.

Son hojas de estilo en cascada, donde si tienes 2 códigos iguales con valores diferentes, toma el último para aplicarlo.

**20. Primeros pasos con CSS**

**Aplicándolo directo en el HTML**

<head>

<style> </style>

</head>

**En una hoja de estilo separada, en el html se liga:**

<link rel=”preload” href=”” as=”style”> Es para que el navegador pre-cargue nuestra hoja de estilos

<head>

<link href=”” rel=”sylesheet”>

</head>

Rel es la relación que hay de ese elemento con el documento html

**22. Estilos a nuestro proyecto.**

Para hacer un poco más pequeño una sola sección de David Martinez – Freelancer lo separamos con un <span>

**Unidades de medida en diseño:**

Pixeles Px

Em

Rem

Con este código se ajusta a modo de que 1 rem = 10 px

html{

*font-size*: 62.5%;

}

body{

*font-size*: 16px;

}

1 rem = 10 px

**23. Introducción a selectores CSS.**

h1 span {

}

Cuando se pone así, todos los span que estén dentro de h1 se le aplicará lo que defina el CSS

**Tipos de selectores:**

**Selector de elemento:** Selecciona un elemento en base a su etiqueta.

p{

color: blue;

}

**Selector de clase:** Son reutilizables e inician con un punto

**.**cliente{

color: blue;

}

**Selector de ID:** Puede tener múltiples ID por página pero no puede repetirse el nombre.

#cliente {

color: blue;

}

**Selector de atributo:** Selecciona elementos basados en un atributo que tenga.

[src=” ”] {

color: blue;

}

**Combinación de descendentes:** Selecciona los elementos hijo cuyo padre sea una clase (o ID) en específico.

**.**cliente **.**nombre {

color: blue;

} busca la clase cliente y después busca la clase de nombre y entonces aplica el código CSS.

**Todos los hijos:** Aplica la regla a todos los párrafos hijos

**.**cliente **>** p {

color: blue;

}

**24. Especificidad.**

Es como el navegador va a mostrar el CSS de acuerdo a que tan específico es el selector que hemos creado. CSS es en cascada, pero es más importante la especificidad del selector.

P {

Color: red

}

P {

Color: blue

}

Aquí tomaría el BLUE por ser el último selector, ambos tienenla misma especificdad

P .parrafo {

Color: red

}

P {

Color: blue

}

Aquí toma RED porque tiene el selector con un selector de clase.

P#parrafo{

Color: red

}

P {

Color: blue!important

}

La bandera ¡ importan le da prioridad

**25. Colores en CSS**

Hexadecimal. #000, #

Nombre: blue, black, cornflowerblue

Hsl o hsla para transparencias

rgb o rgba para transparencias

Definimos un selector con dos puntos : que son pseudoselectores

:root (almacenamos variables de css)

--blanco: #ffffff;

--oscuro: ;

Para usar alguna variable del pseudoselector

Color: var(--primario);

**26. Añadir fuentes externas.**

Un lugar para encontrar fuentes atractivas Google fonts.

El código de la fuente va dentro del HEAD en HTML y sus reglas CSS para especificar la familia de la fuente.

**27. Añadiendo Normalize a CSS.**

Cada navegador muestra la página con algunos estilos diferentes, algunos elementos más sombreados, cosas por el estilo, normalize sirve para hacer que la página tenga el mismo aspecto en cualquier navegador que visitemos el sitio.

Resetea y hace que todos los elementos sean consistentes, se agrega igual que las fuentes de Google, dentro del head, y debe ser la Liberia que se cargue primero y también agregarle un PRELOAD

Normalize se agrega a una nueva hoja o documento css.

**28. Estilos a la navegación – Primeros pasos**

Background-color

Width, para limitar la barra de la navegación

Margin lleva 4 valores, top, right, bottom, left. (margin-top, margin-right…)

Atajo: Margin: 0 auto 0 auto; el orden de los valores debe ser: top, right, bottom, left.

Si son valores similares como en nuestra página basta con margin: 0 auto; y duplicará los valores faltantes acorde a lo que tiene como atributo.

**29. Estilos CSS a la navegación y Displays en CSS**

Agregamos el fondo amarillo a los enlaces de la navegación seleccionando la clase y la etiqueta:

. .navegacion-principal a{

*background-color*: var(--primario);

}

**Display Block o Inline:** Algunos elementos se muestran de una forma y otros de otra. Todos los elementos en HTML ya tiene un display por default, que es el inline

**Display: Block;** Significa que el elemento se colocará uno debajo del otro sin importar su tamaño y contenido.

**Display: inline;** Significa que el elemento se posicionará a la derecha una vez que haya tomado todo el espacio que requiere.

**Nota:** Ningún elemento tiene display: flex; o display: grid; por default

No es necesario agregar un display a cada elemento, solo si queremos modificar el que tiene por default.

**30. Estilos a la navegación y tipo de efectos a textos.**

Hay confusión cuando se usa margin y padding.

Padding engorda el elemento, es decir, dentro del elemento, es decir la barra de navegación.

Margin es de la barra hacia afuera.

Se le da **color** al texto con color: var (--blanco);

Se **quita el subrayado** a los enlaces con: text-decoration: none;

**31. Estilos a la navegación y posicionando los enlaces con Flexbox.**

Para implementar flexbox tenemos que ir al elemento padre de lo que se desea alinear. Se activa con display: flex;

Para modificar el texto de la navegación, usamos **justify-content: space-between;**

**Flexbox:** Es un modelo unidimensional para crear diseños. Corre de izquierda a derecha o de arriba hacia abajo, solo en una dirección. Tiene **2 ejes:** fila (row) que es por default y columna (columna)

**32. Estilos para escribir código CSS – BEM, Utility First, Módulos.**

**BEM – Bloques, Elementos, Modificadores**

.card {} **BLOQUE**

.card\_ \_titulo {}

.card \_ \_ imagen{} **ELEMENTOS**

.card \_ \_ botón {}

.card \_ \_ botón--activo{} **MODIFICADOR**

**Utility First**

Se crean clases con una sola propiedad que describe que es lo que haría.

.text-center{}

.color-red-100 {}

.bg-blue-200{}

.p-2{}

**Módulos**

Se define el contenido principal y después se selecciona cada uno de los elentos HTML.

.card{}

.card h2{}

.card img{}

.card a{}

**33- Finalizando la navegación.**

Para el color amarillo que resalta cuando colocamos el cursor sobre un enlace, usamos una pseudoclase de la navegación con la etiqueta de enlace

.navegacion-principal a:hover{

*background-color*: var(--primario);

*color*: var(--oscuro);

}

**34. Introducción a Responsive Web Design y Media Queries**

**Responsive web desing:** Los diseños deberán adaptarse a las interacciones del usuario y la resolución del dispositivo que utilizan. (Celular, Tablet, Pc, tv, etc) **¿Cómo?** Con media querys.

@media (condición){

Código css que se ejecuta cuando cumple la condición

}

Ejemplos:

@media (*min-width*: 480px){

    body{

*background-color*: rgb(241, 48, 13); /\* Cambia el color de fondo cuando la pantalla supera los 480px, es decir, mínimo debe haber 480px para ejecutar el código CSS

    }

}

@media (*min-width*: 768px){

    body{

*background-color*: cornflowerblue;

    }

}

**35. Añadiendo media queries**

Para hacer responsive la página, en este caso la barra de navegación se añade lo siguiente:

.navegacion-principal{

*display*: flex;

*flex-direction*: column; Se le añade esta configuración para que por default la navegación se muestre en columna, ideal para celulares, PERO CAMBIA CON EL MEDIA QUERIE

*justify-content*: space-between;

}

@media (*min-width*: 768px){ Aquí lo que hace es que de 768px en adelante la barra de navegación se restaura en forma de fila. Aquí la muestra idealmente para computadoras

    .navegacion-principal{

*flex-direction*: row;

    }

}

El scroll a los lados se corrige desde el contenedor:

.contenedor{

*max-width*: 120rem; Toma lo máximo cuando sea posible, si no se adapta

*margin*: 0 auto;

}

Media queries estandar:

480px cuando sea teléfono

768px para Tablet

1140px para laptop o desktop

1440px pantallas más grandes

**NOTA: En flexbox para alinear cuando tenemos en fila se usa JUSTIFY CONTENT, cuando es en columna se usa ALIGN ITEMS**

**36. Imágenes con CSS**

Le añadimos un class al section donde tenemos el rol, la ubicación con el SVG y el botón de contacto.

Para añadir imagen desde CSS usamos

.hero{

*background-image*: url(*../images/hero.jpg*);

}

Pero eso hace que la imagen se repite } haciendo más pequeña la pantalla, es decir no tiene límite. Usamos

*background-repeat*: no-repeat;

Pero ahora la imagen no cubre todo el espacio que requiere la página acorde al tamaño disponible, usamos

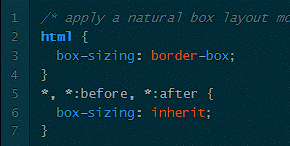
*background-size*: cover;

**37. CSS Box Model.**

Todo css es una caja, pero esa caja como se muestre depende de 4 cosas.

El tamaño de lo que se muestra en la pantalla está delimitado por 4 cosas:

* Tamaño de contenido.
* Tamaño de relleno (padding).
* Tamaño de borde.
* Margen.



Snippet para deshacernos o hacer más fácil el box model en CSS.

**38. Posicionando elementos con position absolute.**

Posicionar el contenido en un lugar preciso: Padre siempre debe tener posición relativa para que el hijo tenga posición abosoluta.

Para la sombra sobre la imagen, dentro del .contenido-hero usamos background color pero en formato rgba, donde los primeros 3 digitos son para color y el 4to para nivel de transparencia, con valor de 0 a 1, siendo 1 totalmente negro, y siendo 0 totalmente transparente.

.contenido-hero{

*position*: absolute;

*background-color*: rgba(0,0,0,.6); /\*Sintáxis Anterior\*/

*background-color*: rgb(0 0 0 / 70%); /\*Nueva sintáxis\*/

*width*: 100%;

*height*: 100%;

}

La nueva sintáxis para poner la transparencia sin el alfa Chanel, es una diagonal y se establece el valor del 0% al 100%

**39. CSS a los headings**

Se va a central el contenido del .hero

*display*: flex;

*flex-direction*: column;

*align-items*: center;

*justify-content*: center;

Ese código se usa para alinear vertical como horizontalmente.

Como hay reglas específicas para los headings (h1, h2 y h3) se ponen las reglas específicas desde CSS para no repetirlas por separado, después del body.

h1{

*font-size*: 3.8rem;

}

h2{

*font-size*: 2.8rem;

}

h3{

*font-size*: 1.8rem;

}

h1, h2, h3{

*text-align*: center;

}

Para que el ícono quede al lado del texto, metemos el código del SVG dentro de la etiqueta <p> o se separa en un nuevo div con class ubicacion para modificarlo aparte con Flexbox

.ubicacion{

*display*: flex;

*align-items*: flex-end;

}

**40. CSS para crear botones.**

Se creó la clase botón con las propiedades en css

.boton{

*background-color*: var(--secundario); COLOR DE FONDO

*color*: var(--blanco); COLOR DE TEXTO

*padding*: 1rem 3rem; TAMAÑO DE BOTON, ES DECIR EXTENSION DE COLOR AZUL

*margin-top*: 3rem; SEPARACION DE ARRIBA CON RESPECTO A “TLAHUAC CDMX”

*font-size*: 2rem; TAMAÑO DE LA FUENTE

*text-decoration*: none; QUITA EL SUBRAYADO DEL ENLACE

*text-transform*: uppercase; TRANSFORMA EL TEXTO EN MAYUSCULAS

*font-weight*: bold; HACE LAS LETRAS EN NEGRITA

*border-radius*: .5rem; REDONDEA LOS BORDES DEL BOTON

*width*: 90%; HACE QUE TOME EL 90% DEL ESPACIO QUE HAY DISPONIBLE, APLICA Y TIENE MEJOR VISTA EN DISPOSITIVOS MOVILES, SE RESETEA CON UN MEDIA QUERIE A CONTINUACION

*text-align*: center; ALINEA EL TEXTO AL CENTRO

}

Media Querie para reseteo de botón y se vea bien en dispositivos de más resolución:

@media(*min-width*: 768px){

    .boton{

*width*: auto;

    }

}

**41. Comentar código con CSS**

En HTML se comenta **<!—Parte comentada -->**

En CSS se comenta **/\* Parte comentada \*/**

**42. Degradados en CSS**

Esto se aplica al fondo total de la página que tiene un color degradado que toma forma de abajo hacia arriba. Como es a toda la página se pone dentro del selector de body en CSS

*background-image*: linear-gradient(to top, var(--grisclaro) 0%, var(--blanco) 100%); DONDE IMAGE SE USA PARA USAR LA PROPIEDAD LINEAR GRADIENT, TO TOP NOS INDICA HACIA DONDE SE DIRIGE EN DEGRADADO, EL PRIMER COLOR Y SU PORCENTAJE ES PARA INDICAR EL COLOR DOMINANTE Y DESDE DONDE EMPIEZA EN LA PÁGNA, EL SEGUNDO COLOR ES AL QUE SE QUIERE LLEGAR DEL DEGRADADO Y EL PORCENTAJE ES PARA INDICAR QUE SE TIENE QUE APLICAR AL 100% DE LA PAGINA.

**43. Sombras con CSS**

Es para aplicar a la parte de mis servicios. La herramienta CSSmatic ayuda a hacer el degradado. En este caso usamos el apartado box shadow, se crea una nueva clase y se configura en CSS

.sombra{

*-webkit-box-shadow*: 0px 5px 15px 0px rgba(46,44,46,1); SOPORTE DE CHROME

*-moz-box-shadow*: 0px 5px 15px 0px rgba(46,44,46,1); SOPORTE DE FIREFOX

*box-shadow*: 0px 5px 15px 0px rgba(46,44,46,1);

*background-color*: var(--blanco); COLOR DE FONDO DE LA CAJA

*padding*: 2rem; ENSANCHA LA CAJA HACIA TODAS LAS DIRECCIONES

*border-radius*: 1rem; REDONDEA LOS BORDES DE LA CAJA

}

**44. Introducción a CSS grid.**

Permite definir la ubicación y el tamaño de los elementos en el sitio web.

Mientras que en flexbox crece y se ubica automáticamente en cierto lugar, en CSS grid el cometido se agrupa en un área definida.

En cierta forma es como una tabla de HTML pero con mayor control y flexibilidad en el diseño. En flexbox solo tenemos row y column, en Grid tienes acceso a ambas al mismo tiempo. Algunos diseños son más fáciles en grid y otros en flexbox.

.servicios{

*display*: grid;

*grid-template-columns*: 1fr 1fr 1fr; /\* 1 fracción igual cada columna, si queremos una más grande podemos ponerle 2fr y grid hace los cálculos\*/

    /\*grid-template-columns: 33.3% 33.3% 33.3%; Para dividir todo el espacio el partes iguales o se usa la que voy a marcar\*/

    /\*grid: 200px 800px / 300px 300px; Versión corta, primeros valores de filas (rows

     segundos valores para columnas)\*/

}

*grid-template-columns*: repeat(3, 1fr); /\*Versión abreviada\*/

Pero se tiene que hacer responsivo, entonces, se mostrará en 3 columnas con una resolución de pantalla mínima de 768px con media queries. Menos de esa resolución lo mostrará normal sin aplicar GRID, es decir, en 3 filas.

@media (*min-width*: 768px){

    .servicios{

*display*: grid;

*grid-template-columns*: repeat(3, 1fr);

    }

}

**45. Cuando usar GRID, Flexbox, floats.**

**Flexbox.** Alineación y distribución de elementos que estarán dentro de contenedores (por ejemplo una barra de navegación).

**Grid.** Definir el layout del sitio web, como columnas o contenedores de elementos (como las columnas de servidio).

Añadimos espacio entre cada columna con:

*column-gap*: 1rem;

**46. Finalizando la sección de servicios.**

Agregamos un class a cada servicio para separarlos entre sí, con una clase llamada servicio, aplicandolo dentro del section de cada uno, en total 3.

<section class="servicio">

                <h3>E-Commerce</h3>

De igual manera para poder manipular los íconos los separamos con un div a cada uno con una clase llamada íconos (en el servicio que tiene 2 iconos, los 2 quedan dentro de un solo div)

<div class="iconos">

Con esto en CSS pudimos manipular por separado cada elemento, en el caso de íconos, para posicionarlos y alinearlos en el centro.

Y usando la clase servicios apoyados de la selección de etiqueda dimos formato:

.servicio{

*display*: flex;

*flex-direction*: column;

*align-items*: center; Todo esto fue para centrar los círculos. Se añade flex para poder alinear correctamente.

}

.servicio p{

*line-height*: 2;

*text-align*: justify; Con la etiqueta p le dimos formato especifico a los parrafos

}

.servicio h3{

*color*: var(--secundario);

*font-weight*: bold; con la etiqueta h3 modificamos los headings 3

}

.iconos{

*height*: 15rem; Alto del espacio del ícono

*width*: 15rem; Ancho del espacio del ícono

*background-color*: rgba(250, 242, 10, .7); Color de fondo del ícono

*border-radius*: 50%; Se rodea el borde para darle apariencia circular

*display*: flex;

*justify-content*: space-evenly;

*align-items*: center; Se añade flexbox para posicionar los íconos dentro del círculo, más arriba en .servicios le dimos posicionamiento central a los círculos.

}

**47. Primeros pasos con el formulario.**

Usamos margin 0 auto para alinear centralmente siempre que tengamos un elemento como formulario y cuyo padre no tenga un display flex. El formulario tiene un display block

.formulario{

*background-color*: var(--gris);

*width*: min(60rem, 100%); /\* Lo que hace min es utilizar el valor

    más pequeño acorde a la pantalla (este, o este)\*/

*margin*: 0 auto;  /\*Centra el formulario\*/

*padding*: 2rem; espacio dentro del form

*border-radius*: 1rem; esquinas redondeadas

}

.formulario fieldset{

*border*: none; quita el borde que se veía del formulario

}

.formulario legend{

*text-align*: center;

*font-size*: 1.8rem;

*text-transform*: uppercase;

*font-weight*: bold;

*margin-bottom*: 2rem;

*color*: var(--primario);

}

**48. CSS a los inputs**

.campos{

*margin-bottom*: 2rem;

}

.campos label{

*color*: var(--blanco);

*font-weight*: bold;

*margin-bottom*: .5rem;

*display*: block;

}

.area-escritura{

*width*: 100%;

*border*: none;

*padding*: 1.5rem;

*border-radius*: .5rem;

}

.enviar .boton{

*width*: 100%;

}

**49. Posicionando los inputs.**

REVISAR DE NUEVO VIDEO.

**50. Añadiendo un efecto de Scroll**