HTML

Atributo → href => Se utiliza para indicar el destino al que apunta nuestro enlace.

Atributo → alt => Este atributo nos permite especificar el texto alternativo

Atributo → src => Se utiliza para indicar el destino en donde está alojada nuestra imagen.

Atributo → width => Este atributo nos permite especificar el ancho de la imagen.

Atributo → height => Este atributo nos permite especificar el alto de la imagen.

**TODO LO QUE SE VEA DENTRO DE LA PAGINA SIEMPRE VA DENTRO DEL BODY**

# Clase 11:

HTML es hiper text maksh lenguage

Lo crearon para compartir papers entre cerebritos. Está compuesto de etiquetas que son porciones de texto que interpreta el navegador.

Hay muchas etiquetas y todas cumplen un rol fundamental.

**ETIQUETAS DE H:**

-Apertura:<h1> La etiqueta va rodeada por cocodrilos.

-Clausura: </h1> Lleva barra para cerrar

♦ Para modificar etiquetas usamos los **ATRIBUTOS** estos están compuestos por:

-El nombre del atributo

-El signo =

-El valor del atributo entre comillas.

**♦ Estructura básica del archivo html:**

1- <Doctype>: Define la versión del lenguage que se está utilizando.

2- <html>: Define que todo el contenido interno será de html

3- <head>: Guarda toda la información referente con el documento como por ej la codificación de caracteres y el título a mostrar en la pestaña del navegador entre muchas cosas más.

4- <meta>Etiqueta de uso múltiple

<meta charset = "UTF-8"> Se usa para definir a partir del atributo charset la codificación de caracteres del doc. Acá dice UTF-8 y este atributo define que el documento deberá soportar todos los caracteres del castellano y otros idiomas que no son el inglés ej. Ñ

5-<title>: Define que se mostrara en la pestaña del navegador

6-<body>: Contiene todas las demás etiquetas que generaran todo el contenido visual

Algunas etiquetas pueden estar dentro de otras, por ejemplo la etiqueta <head> anida (contiene) a las etiquetas <meta< y <title>

♦**Elemento de línea y de bloque:**

*-Elementos de bloque*: Ocupa todo el "renglón" de una hoja

*-Elemento de línea*: Ocupa solo su espacio, puede tener otros elementos a los costaditos.

♦**Etiquetas de HTML**

H = headings (Titulares) Hay 6 tipos de headings del H1 al H6

Se verán con distintos tamaños, pero todos se verán en negrita.

Se pueden usar cada vez que necesitemos generar un bloque diferencial de texto respecto al resto del contenido.

**Importante:** El H1 al ser el más power de los heading solo se puede usar una sola vez por documento HTML, sin embargo también está bueno saber que los heading pueden ir en cualquier orden

2- <p> Paragrahp (párrafo): Sirve para generar bloques de texto. La podemos usar ilimitadamente. Es una de las etiquetas que más se usa en una página web.

Tenemos dos etiquetas de contenido visual en HTML:

<strong>: Hace textos en negrita.

<em>: Hace textos en cursiva.

Estas dos etiquetas son elementos de línea. Las etiquetas <h> y <p> son de bloque.

<br> BreakLine: Si bien no es una etiqueta de texto, sirve para generar saltos de línea entre las distintas líneas de un párrafo.

**♦ Listas:** Las listas son elementos de bloques. Tenemos dos tipos:

**-Ordenadas:** Acá cada ítem tiene un número que lo ordena según "relevancia". <ol>

-**Desordenadas:** Es una palabra abajo de la otra sin más. <ul>

Una vez que definimos si la queremos ordenada o desordenada, debemos escribir entre las etiquetas de orden la etiqueta <li> que representa a los elementos lista propiamente dichos.

**Se escriben así:**

Ej: lista ordenada:

<ol>

<li>

comprar fruta

comer fruta

cagar fruta

</li>

</ol>

♦**Rutas e hipervínculos:**

**-Ruta**: En html una ruta es un trayecto que un doc html debe atravesar hasta llegar a una ubicación de un determinado recurso.

*-Hay dos tipos de rutas*:

**-Relativas:** Contienen solo la información necesaria para que el documento html llegue a la ubicación del recurso determinado. Ósea básicamente no le dice exactamente donde está, pero lo ubica, es como si dijera que en mi casa la sal en la cocina, se dónde ir a buscar, pero no del todo. Estas rutas contienen coordenadas parciales de una ubicación.

**-Absolutas**: Contienen toda la información de una ubicación de un recurso determinado ej: una url de un sitio web www.google.com

Es como una coordenada exacta para ubicar datos

**Pautas para las rutas:**

1- Nunca pueden hacer referencia a una carpeta dentro del disco duro de nuestra PC.

2-Deben ser fácilmente identificables por el documento dentro de cual las estamos gestionando.

Hay una etiqueta que nos permite referenciar imágenes dentro de un doc. Html esta etiqueta se llama <img> la cual no necesita cierra, pero sí de los siguientes atributos.

1- src = sourse (origen): Acá va la ruta del recurso de imagen que queremos mostrar. Acá podemos escribir una ruta relativa o absoluta, en esta última no es necesario que la imagen este en nuestra computadora, pues la misma está siendo llamada desde alguna ubicación en internet.

Si usamos una ruta relativa si es necesario que el archivo de imagen este en alguna carpeta dentro de nuestro proyecto.

2- alt : Acá va un texto alternativo a esta imagen que se muestra cuando no carga la imagen.

**♦Enlaces o hipervínculos:**

Estos son etiquetas de html que nos permiten vincular distintos archivos entre sí o distintas ubicaciones en internet.

Para generar enlaces vamos a usar la etiqueta <a> </a> la cual contendrá el texto que queremos mostrar al usuario como enlace.

La etiqueta <a> necesita una referencia "href" en donde se escribe la ruta del recurso que queremos vincular entre comillas

Se puede usar además el atributo "target" el cual define en donde se abrirá ese documento que estamos queriendo vincular.

Si queremos enlazar dos documento que tenemos en la carpeta de nuestro proyecto deberíamos usar una etiqueta <a> con su href apuntando a nuestro archivo.

Los enlaces se pueden catalogar como Externos (nos llevan fuera de nuestro sitio web) o Locales (nos permite enlazar archivos de nuestro proyecto o sitio)

**♦ Semántica de HTML**

Hace referencia a las buenas prácticas que tenemos para crear una página web y hacer nuestro código muy legible.

Antes se usaba un <div> para todo, pero ahora hay un montón de etiquetas para hacer que el código se vuelva más legible ya que cada etiqueta posee determinados elementos por consenso general.

-La etiqueta <header> servirá para generar la cabecera de un documento, y contiene logotipo, barra de navegación, barra de búsqueda y enlaces a redes sociales.

-La etiqueta <nav> sirve para generar una barra de navegación sea la principal o una alternativa. Dentro de esta etiqueta el árbol de navegación se suele implementar a través de listas desordenadas, elementos de listas y enlaces o de hipervínculos.

-La etiqueta **<footer>** sirve para generar el pie de página de nuestra web o de una sección de contenido. En general acá van los derechos reservados, algunos datos o enlaces adicionales de la web

-La etiqueta **<secction>** sirve para definir una sección de contenido monotemática, dentro de esta se puede usar nuevamente un header y un footer si fuera necesario.

Si quisiéramos crear un breve apartado dentro de la página de inicio que hable de un producto o servicio esta etiqueta va de perlas.

-La etiqueta **<article>** servirá para definir una porción de contenido dentro de una sección, si necesitáramos regenerar distintos apartados de texto o contenido de una misma temática esta sería la etiqueta ideal.

En general article va dentro de sección, pero no es obligatorio.

Básicamente la semántica hace que nuestro código sea tan legible como para que un robot buscados como el de google pueda entenderlo fácilmente.

**♦Estructura de un sitio web:**

Para esto debemos crear muchos contenedores para cada elemento de nuestro sitio web.

Para generar un doc html básico y sencillo podemos empezar por algo sencillo, que se vería algo así:

<body>

<header>

<div> Logotipo</div>

<nav>

<ul>

<li> cualquier lista </li>

</ul>

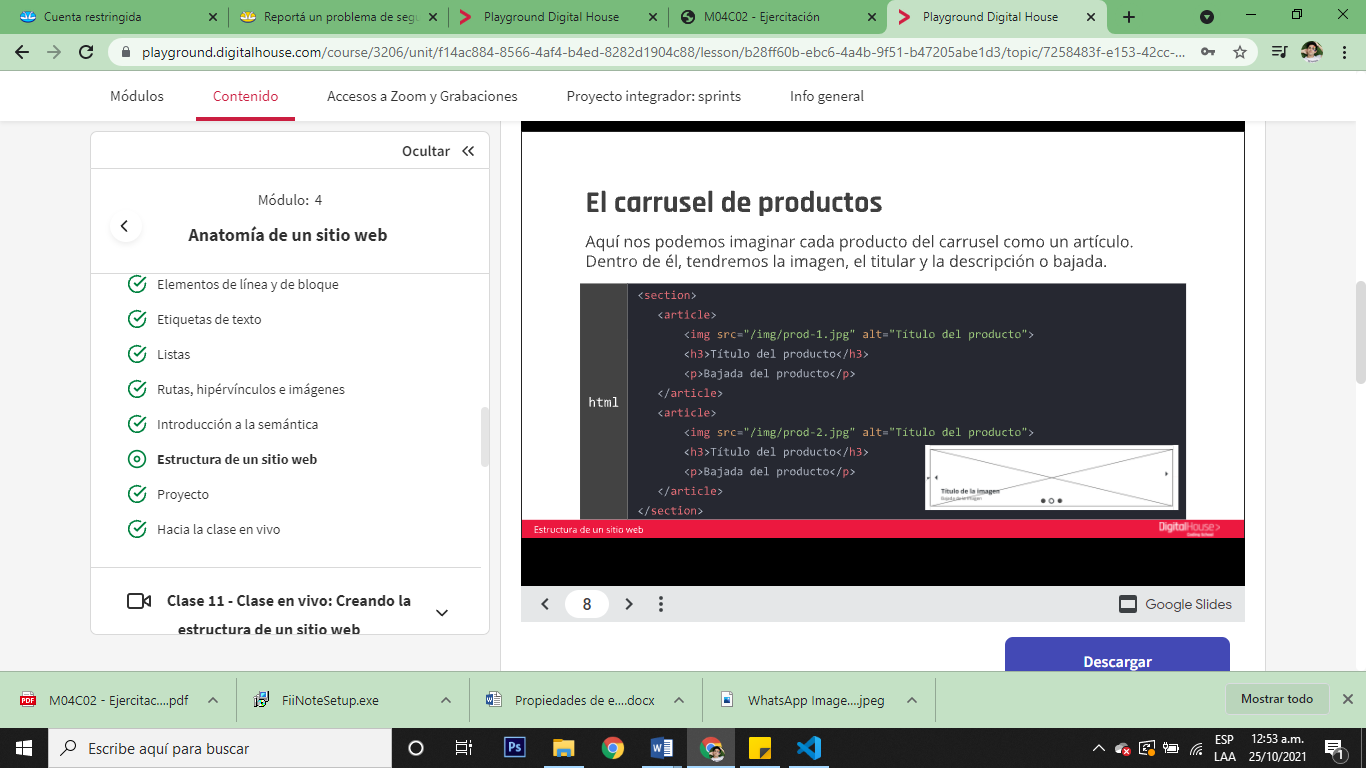
</nav>

</header>

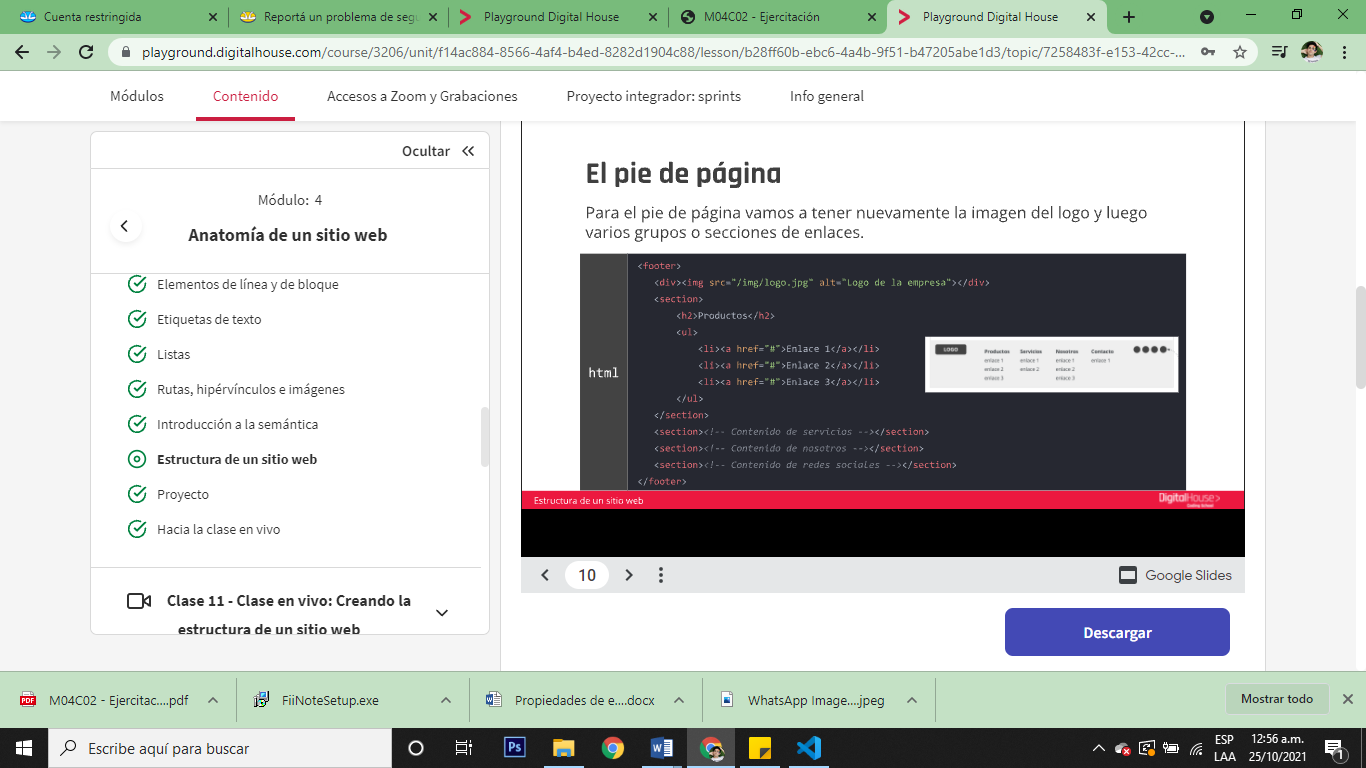
</body>

-Recordemos que todo lo que se ve en la página va siempre dentro del body

Por otro lado si tuviéramos que crear una porción de contenido para algún producto destacado podemos disponer un "contenedor" que guarde toda esta información acá usaríamos < seccion> y < article>.



Por ultimo para darle un cierre al documento necesitaríamos un contenedor para hacer un pie de página que empezara con un footer



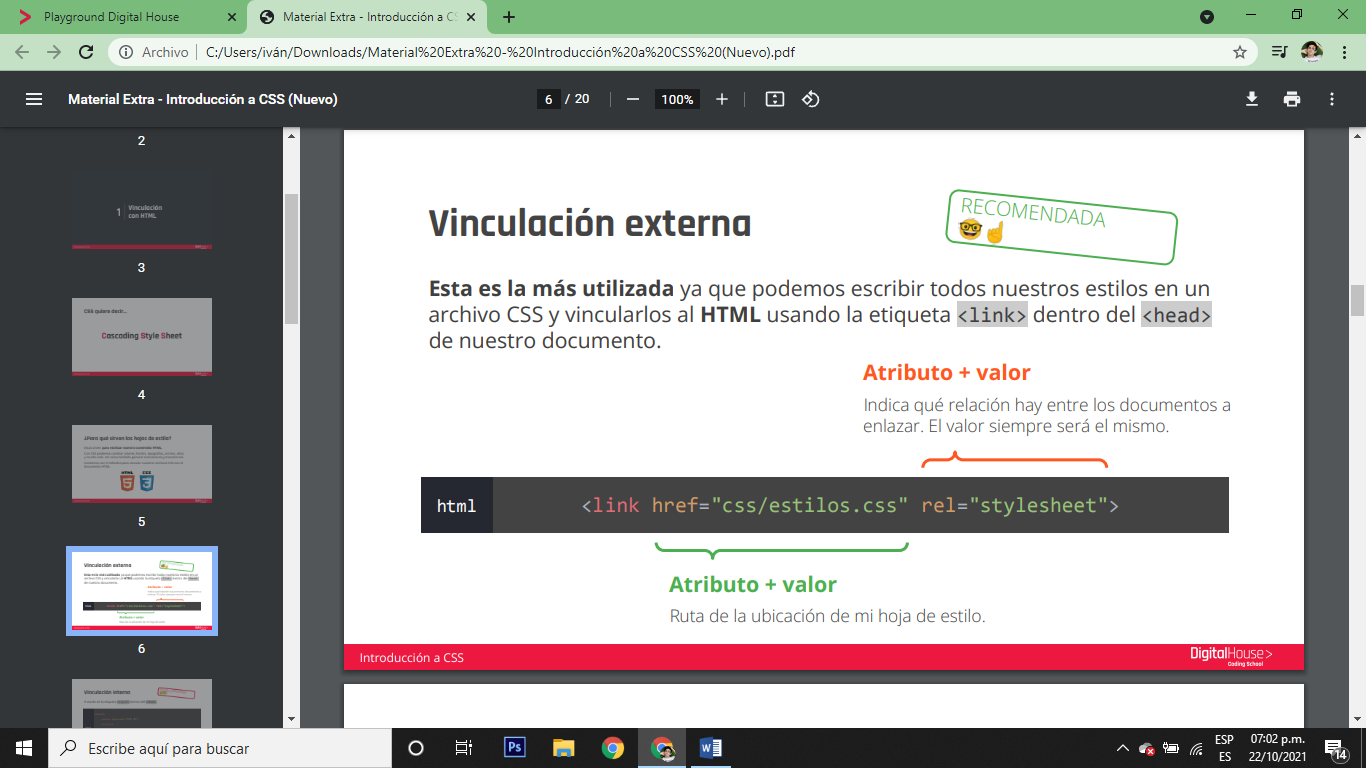
Como se llamaría este documento? Siempre el primer documento de un sitio web, lo que vendría a ser la homepage debería llamarse index.html, esto es un estándar que se usa para poder identificar al primer archivo de todo sitio web como la página principal o índice de contenido.

# CLASE 12

**Vinculación externa:**

Esta es la más utilizada ya que podemos escribir todos nuestros estilos en un archivo CSS y vincularlos al HTML usando la etiqueta dentro del de nuestro documento.

**ESTRUCTURA DE UNA VINCULACION EXTERNA:**



Cuando ponemos un **for** y un **if** no pongamos un else con un return porque el return corta la iteración del FOR si ponemos un else ponemos un ***console.log*** en lugar del return para que el for siga iterando

<link href=”ccs/estilos.css” rel=”stylesheet”>

**♦Propiedades de estilo tipográficas de CSS:**

1- **Color** ej: color: red;

2- **Font Family**: ej font-family: Verdana /Arial

2b - **Font Size** ej: font-size: 15px;

3- **Font Style** ej: font-style: normal o italic (italic es en cursiva)

4- **Font weight**: ej font-weight: bold; (bold me pone el texto en negrita) normal; (Lo deja normal)

5- **Text Align** ej: text-align: left

Esta propiedad de estilo tiene 4 opciones para alinear el texto left (izquierda) / right (derecha) / center (centrado) y justify (ajustar)

Por defecto se tiene a left.

6- **Text Decoration** ej: text-decoration: none;

-Esta propiedad de estilo tiene varias opciones para decorar un texto. None (elimina la decoración del texto) / underline (Que aplica subrayado a cualquier texto que queramos) / line-through (Tacha el texto)

7- **Text transform** ej text transform: none;

Esta propiedad de estilo tiene varias opciones para transformar un texto. None (elimina la transformación existente) / Uppercase (Hace que el texto pase a mayúscula/ lowercase (pasa a minúscula)

8- **Line-Height** Se ocupa del interlineado del texto. Va de la mano del font size porque depende de este para tener una legibilidad correcta

Ej: font-size: 15px;

line-height: 23px

Acá digo que quiero un interlineado de 8px entre las letras de mis renglones (ya que line-height determina el alto del renglón)

**♦Color en CSS:**

*-Colores por nombre*: color: red

Ofrece solo 140 colores

-*Colores por valor hexadecimal*: color; #f5f5dc;

Ofrece MUCHISIMOS COLORES

-*Colores por RGB*; color: RGB (x, x, x);

En cada x va un número entre 0 y 255.

-*Colores por RGBA* (Es igual que RGB, pero este además permite darle opacidad al color)

EJ: color: rgba (255, 0, 0, 0.5);

Acá el cuarto número es la transparencia del color. Este debe ser un numero decimal entre el 0 y el 1

*Para buscar colores en el navegador tengo que buscar "color picker"*

♦**Fondos en CSS:**

***-background-color***: (cualquier forma de representar un color)

***-background-image***: ruta donde está la imagen que quiero usar de fondo

Queda así: body {background-image: url ('https: //images3.alphacoders.com/621/621682.jpg')}

*-Las imágenes de fondo se sobreponen a los colores de fondo*

-***background-repeat***: repeat (hace que la imagen de fondo se repita por toda la imagen por el navegador) / no repeat (Hace que la imagen de fondo no se repita) / reapeat -x / repeat -y (hace que la imagen se repita en un eje) /

-***background-position***: center bottom (aca hay que chequear que no se esté repitiendo la imagen, la primer posición es el lugar en el eje X y la segunda es la posición en el eje Y)

*Podemos usar valores numéricos o verbales, para X left y right y para Y bottom y top.*

-***background-attachment*** (sirve para definir el comportamiento de la imagen de fondo cuando el usuario scrolea la pantalla)

Ej: background-attachment: scroll/fixed.

Scroll hace que la imagen se desplace en conjunto tipo como los tiktoks.

Fixed mantiene la imagen fija mientras se scrolea el contenido de nuestro navegador tipo el fondo de los chats de Whatsapp.

-**background-size:** Nos permite controlar el tamaño de la imagen de fondo. Puede tener dos valores CONTAIN y COVER

**CONTAIN**: Ocupa todo el ancho o largo (lo que suceda primero) de la pagina

**COVER**: Ajusta a la imagen para ocupar todo el fondo del navegador.

# CLASE 13

♦Modelo de caja

Todos los elemento de html tiene 4 costados como un cuadrado/ rectángulo

Propiedades del modelo de caja:

-Display: Permite cambiar el comportamiento original de los elementos de línea y de bloque. Sus posibles valores son:

Block: Permite hacer que un elemento de línea pase a ser uno de bloque.

Inline: Permite hacer que un elemento de bloque pase a ser uno de línea.(No se recomienda nunca)

Inline-block: Acá hace que el elemento de bloque pase a ser de línea, pero manteniendo algunas propiedades del elemento de bloque.

Ahora veremos unas propiedades que solo se aplicaran cuando al elemento al cual se las asignamos sea un elemento de bloque o semibloque.

**width** (ancho): Determina el ancho del elemento. Recibe un valor número expresado en pixeles o porcentajes.

**height** (alto): Determina el alto del elemento. Recibe un valor número expresado en pixeles o porcentajes. No es la mejor ya que al setear esto estamos limitando el alto del elemento y no siempre queremos que esto suceda

Padding (relleno): Permite asignar un espacio de separación entre el contenido del elemento y sus costados. Básicamente crea márgenes.

Se expresa generalmente en pixeles.

Como entender el padding:

-Con un solo valor es la misma medida para los 4 márgenes.

- Con dos valores el primero es el margen superior e inferior y el segundo es los márgenes izquierdo y derecho.

- Con tres valores el primero se aplicara al costado superior, el segundo a los costados derecho e izquierdo y el tercero al costado inferior.

- Con cuatro valores el primero se aplicara al costado superior, el segundo al costado derecho, el tercero al costado inferior y el cuarto va al costado izquierdo.

Si quiero aplicar un solo margen a mi elemento html puedo usar padding-top, padding-bottom, padding-right, padding-left respectivamente para el costado que quiera aplicar.

border (borde): Asigna una línea decorativa a los costados del elemento.

border-width: Define el ancho del borde y se expresa en pixeles.

border-style: Define el tipo de línea para el borde. Puede ser -solid (raya común) -dashed (línea punteada con cuadraditos) - dotted (línea punteada) -double (subrayado doble)

border-color= Define el color de la línea y acepta cualquier formato de color valido.

No importa el orden de estos tres valores siempre y cuando este separados por un espacio entre si.

Algo importante a tener en cuenta es que tanto los valores del padding y el ancho del borde se SUMAN a los valores de width y height. Con lo que seguramente el elemento va a ampliar su tamaño total.

Si queremos evitar que esto se sume deberemos usar la propiedad box.sizing.

-margin (márgenes): Nos permite gestionar los márgenes de un elemento, estos márgenes los podemos entender como el distanciamiento posible entre nuestro elemento y los de más que lo rodean

Margin no suma tamaño a la caja, pero si ocupa el espacio del contexto en el que se encuentra.

Además si queremos aplicar un solo margen podemos usar las siguientes propiedades respectivamente. margin-top, margin-bottom, margin-right, margin-left

Otra particularidad es que si escribimos "margin: auto;" el elemento que la recibe quedara alineado horizontalmente al centro, siempre y cuando el mismo sea un elemento de bloque.

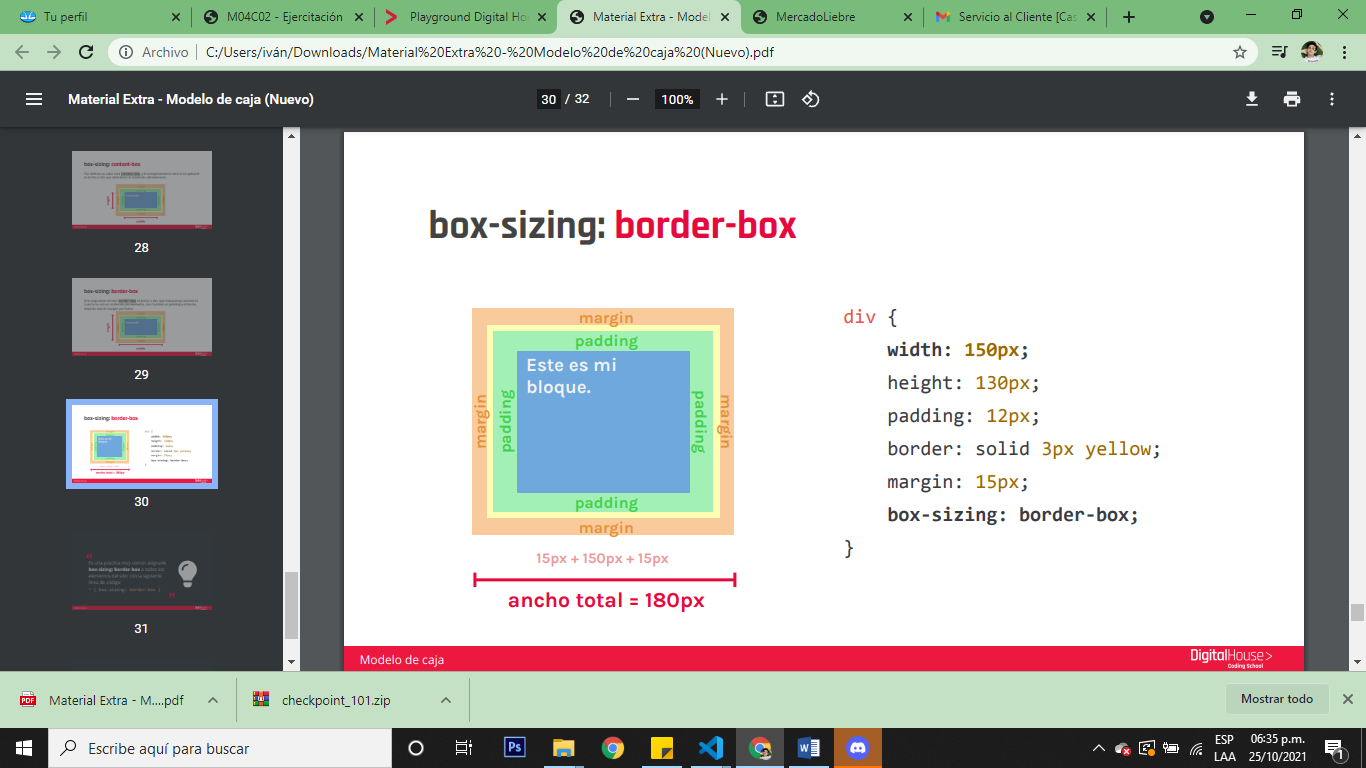
box-sizing: border-box; Esto evita que el elemento se amplié más allá de lo estipulado en el width y en el heidth

Generalmente aplicaremos la propiedad box-sizing muy seguido, para hacer esto css tiene un selector universal, el cual permite aplicar declaraciones a todos los elementos del html.

Se recomienda aplicar solo en el "box-sizing-border-box".

Se ve así:

\* { box-sizing: border-box} Acá el asterisco es para que lo pongamos una sola vez en css arriba de todo y se le apique a todos los elementos.



Viewports:

Son el área visible dentro de cualquier navegador web y dispositivo, ya que hace que nuestro sitio sea responsive o adaptativo a cualquier resolución de pantalla.

El viewport NO VA EN CSS, va en HTML dentro de la etiqueta <head> .Se ve así:

<meta name="viewport" content="width=device-width, initial-scale=1">

El atributo name, permite usar la etiqueta meta para configurar el viewport.

El atributo content, permite configurar dos cosa básicas width por un lado y device-wid, acá estamos definiendo que el ancho del contenido del documento html debe ser igual al ancho del dispositivo. Esto por sí solo no alcanza para que nuestro sitio sea 100% responsive, necesitaremos escribir cositas en css también.

El initial-scale=1 como valor del atributo content determina que la escala inicial o zoom in de nuestro documento será igual a 1.

Con esta etiqueta basta para configurar el viewport y dejar nuestro documento html preparado para generar documentos 100% responsive

Un batazo: La diferencia entre padding y margin es que el pading separa el contenido del borde y el margin separa la caja de otra.

**En html cuando un elemento esta un líneas distintas html agrega un espacio entre estos elementos. Para evitar esto hay que poner los elementos pegados uno al lado del otro.**

**Lugar para sacar las medidas de pantalla: getbootstrap.com**

MEDIDAS RELATIVAS

Medidas absolutas: Son radicales, no cambian según su contexto Ejemplo: px

Medidas Relativas: Cambian según su contexto (caja madre). Ejemplo: %

El contexto de cambio siempre es directo, se toma siempre a la caja madre más cercana como contexto

Unidades de medida de viewport:

VW: viewport-width

VH: viewport-heigth

Son medidas relativas como los porcentajes, pero con una sutil diferencia... que es que al momento de identificar a su contexto VW y VH no tendrán en cuenta a la caja madre, sino tomaran su valor en función de la medida que tenga asignado el viewport.

Resumiendo la Diferencia em vs. Rem

Trasladar unidades rem a valor de pixel se determina por el tamaño de la fuente del elemento html. Éste tamaño de fuente es influenciado por la herencia del ajuste del tamaño de fuente del navegador a menos de que explícitamente se sobrescriba con una unidad no sujeta a herencia.

Transladar unidades em a valor de pixel se determina por el tamaño fuente del elemento en el que son usadas. Éste tamaño de la fuente es influenciado por la herencia de elementos padre a menos que explícitamente se sobrescriban con una unidad no sujeta a herencia.

Por qué usar Unidades rem:

La mayor potencia que las unidades rem ofrecen no es solo que dan consistencia al tamaño sin importar la herencia del elemento. Más bien, es que nos dan una forma de que los ajustes del tamaño de la fuente del usuario tengan influencia en cada aspecto del maquetado de un sitio al usar unidades rem donde solíamos utilizar unidades px.

Por ésta razón el propósito primario de utilizar unidades rem debería ser asegurar que cualquier tamaño de fuente predeterminado que tenga un usuario establecido en su navegador, el maquetado se ajustará para acomodar el tamaño del texto dentro de él.

Un sitio puede ser diseñado inicialmente enfocándose en el tamaño fuente del navegador predeterminado de 16px.

Pero al usar unidades rem, si un usuario incrementa el tamaño de su fuente, la integridad del maquetado se preservará, y el texto no se encogerá en un espacio rígido preparado para un texto más pequeño.

Por Qué usar Unidades em

El valor clave que las unidades em ofrecen es que permiten que los valores de tamaños sean determinados por un tamaño de fuente diferente al del elemento html.

Por ésta razón, el propósito primario de las unidades em debería ser permitir escalabilidad dentro del contexto de un elemento de diseño específico.

Por ejemplo, podrías establecer el padding, margin y line-height alrededor de un elemento del menú de navegación para usar valores em.

De ésta forma si cambias el tamaño de la fuente del menú el espacio que rodea a los elementos que rodean a los elementos del menú escalarán proporcionalmente, independientemente del resto del maquetado.

Usa Unidades em Para:

Cualquier tamaño que debería escalar dependiendo del tamaño de la fuente de un elemento en lugar que en el de la raíz.

Hablando en términos generales, la única razón que necesitarás para utilizar unidades em es escalar un elemento que no tiene un tamaño de fuente predeterminado.

Por nuestra experiencia arriba, componentes de diseño como elementos de menú, botones y encabezados pueden tener sus tamaños de fuente explícitamente declarados. Si cambias éstos tamaños de fuente, quieres que todo el componente se escale proporcionalmente.

Las propiedades comunes a las que ésta pauta aplicará son ajustes de margin, padding, width, height y line-height, cuando se usan con elementos con tamaño de fuente no predeterminado.

Recomiendo que cuando uses unidades em, el tamaño de la fuente del elemento en el que son usadas debería ser establecido en unidades rem para preservar escalabilidad pero evitar confusión de herencia.

Usa unidades rem para:

Cualquier tamaño que no necesite unidades em por las razones descritas arriba, y que deberían escalar dependiendo de la configuración del tamaño fuente del navegador.

Esto cuenta para casi cualquier cosa en un diseño estándar incluyendo la mayoría de alturas, la mayoría de las anchuras, la mayoría de los padding, la mayoría de los márgenes, anchuras de bordes, la mayoría de los tamaños de fuente, sombras, básicamente casi cada parte de tu maquetado.

A grandes rasgos, todo lo que puede hacerse escalable con unidades rem, debería hacerse.

Consejo

Cuando se crean maquetados es frecuentemente más fácil pensar en pixeles pero sacar en pantalla en unidades rem.

Puedes tener pixel para hacer cálculos rem automáticamente vía un proprocesador como Stylus / Sass / Less, o un postprocesador como PostCss con el plugin PXtoRem.

Alternativamente, puedes usar PXtoEM para hacer manualmente tus conversiones.

Siempre utiliza rem en Media Queries

Importante, cuando se usan unidades rem para crear un diseño uniformemente escalable, tus media queries deberían estar también en unidades rem. Esto asegurará que cualquier tamaño de la fuente que tenga el navegador del usuario, tus media queries responderán a él y ajustarán tu maquetado.

MEDIA QUERIES: Cuando desarrollamos una aplicación web queremos que se vea bien en todos los dispositivos, para esto ccs nos da una poderosa herramienta: Los media queries.

Cuando hacemos un sitio web responsive no esperamos una resolución definida, sino que pensamos en que el sitio se vea perfectamente bien en cualquier contexto en el que se visualice.

Las media queries es una regla de estilo que van en nuestra hoja de estilos que contendrá un conjunto de otras reglas que se aplicaran si y solo si se cumple un condición determinada.

Las media queries no buscan reescribir todas las reglas de estilo, no que ajustan solo aquellas que lo necesiten e incluso crean algunas nuevas, siempre con el objetivo de que el sitio se vea bien.

Así se ve una media querie:

@media (condición) {

Mis reglas de estilo}

Para tener ordenado nuestro código se recomienda tener nuestras media querie al final de la hoja de estilo.

Las media queries nos permiten optar por dos caminos Mobile first (desarrolla de dispositivos más chico al más grande) y el Mobile last (al revés que Mobile first)

Cuando queremos aplicar media queries en el modo mobile first la media se verá así:

@media (min.width: 560px) {

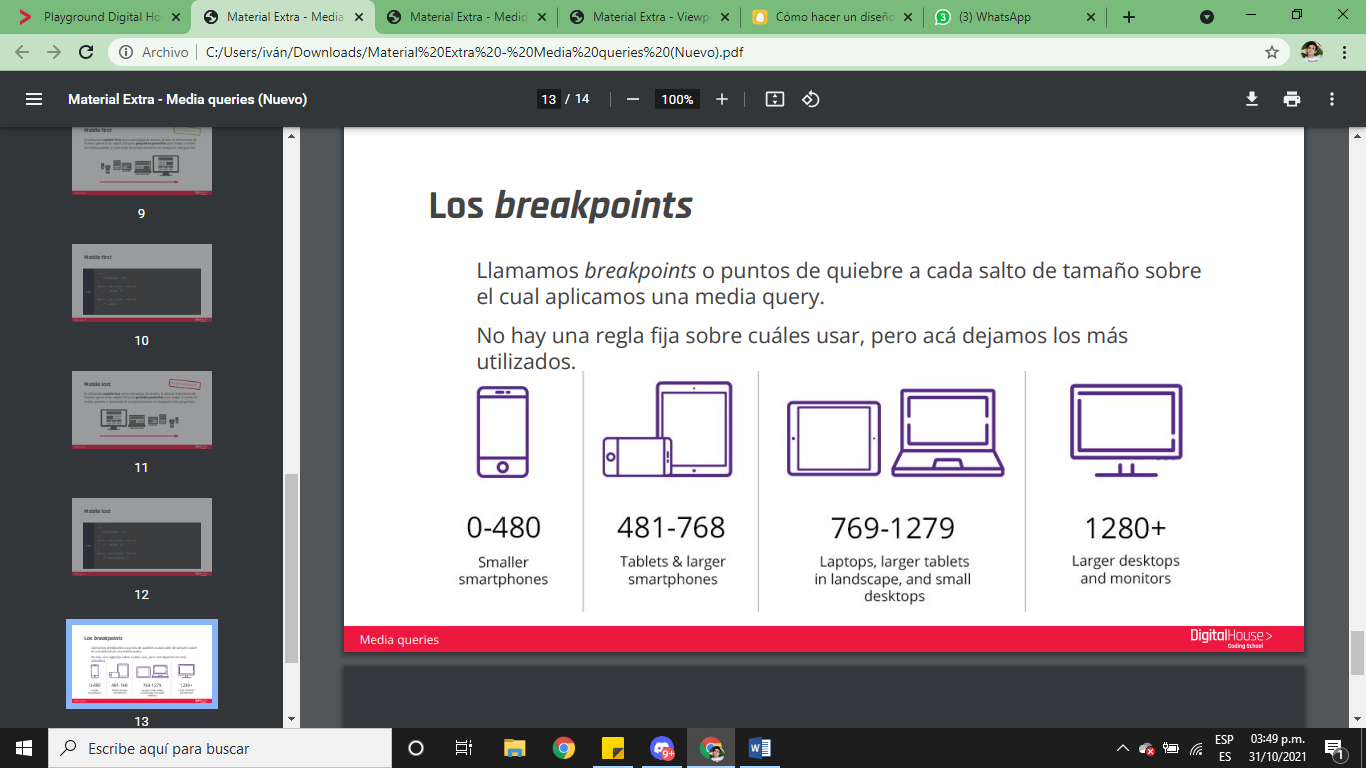
Mis reglas de estilo}

Esto le dice a la consola que aplique la media querie solo cuando el sitio tenga un width superior a 560 px



¿Cuándo es bueno definir una media querie?

Recomendación, si tu sitio en esa resolución de salida se ve mal, ahí pones un media querie. (Se le llama breakpoint a esta instancia)



Si tenemos más de 4 media queries no estamos haciendo muy bien las cosas negry.

# Clase 14

INTRODUCCION A FLEXBOX:

Cuando trabajamos con CSS algo común es querer ordenar nuestros elementos de manera particular y no una abajo de la otra. Para eso usamos flexbox.

Para ordenar de forma distinta nuestro elemento usaremos flexbox. Lo más importante de esto es que la mayoría de las propiedades las vamos a aplicar en el contenedor padre del elemento que queremos manipular.

Como se usa flexbox?

Aplicaremos la siguiente declaración en el elemento padre, se veria así:

.elementoPadre {

display: flex;

}

Esto le indica al elemento .elementoPadre, que cualquier elemento que contenga deberá ser lo que conocemos como flex item, además los elementos deberán estar siempre uno al lado del otro sin importar si son elemento de bloque, semibloque o línea.

Algo particular que pasa con display flex es que todos los elementos hijos se ponen uno al lado del otro en una fila y si es necesario se le ajusta el tamaño para que así sea.

Esto se debe a que el contenedor por defecto implementa la propiedad **flex-wrap con el valor nowrap**. Para evitar este comportamiento y hacer que las cajas hijas conserven el tamaño en width asignado a estas tendremos que cambiar el valor de **flex-wrap a wrap.**

Ósea así:

.contenedor { display:flex;

flex-wrap:nowrap;}

a

.contenedor { display:flex;

flex-wrap: wrap;}

**EJES:**

Cuando trabajamos con un contenedor flex es importante saber que en el mismo no tenemos costados, ya que no está limitado.

El contenedor flex no usa el concepto de costados sino el de ejes. Los ejes de flex son Main Axis (Horizontal) y Cross axis (Vertical). Estos ejes pueden ser intercambiados si así lo quisiéramos.

¿Qué podemos hacer para configurar este comportamiento desde CSS?

Dentro nuestra hoja de estilo vamos a asignar la propiedad "flex-direction"

Esta nos permitirá definir la orientación del eje principal o "main axis" por defecto viene con el valor "row" que significa que el eje principal será el horizontal con punto de inicio en el lateral izquierdo y fin en el lateral derecho, en este escenario el cross axis es vertical.

Si quisiéramos que el main axis sea vertical tendríamos que cambiar el valor de la propiedad **"flex-direction: row"** a **"flex-direction: column"**

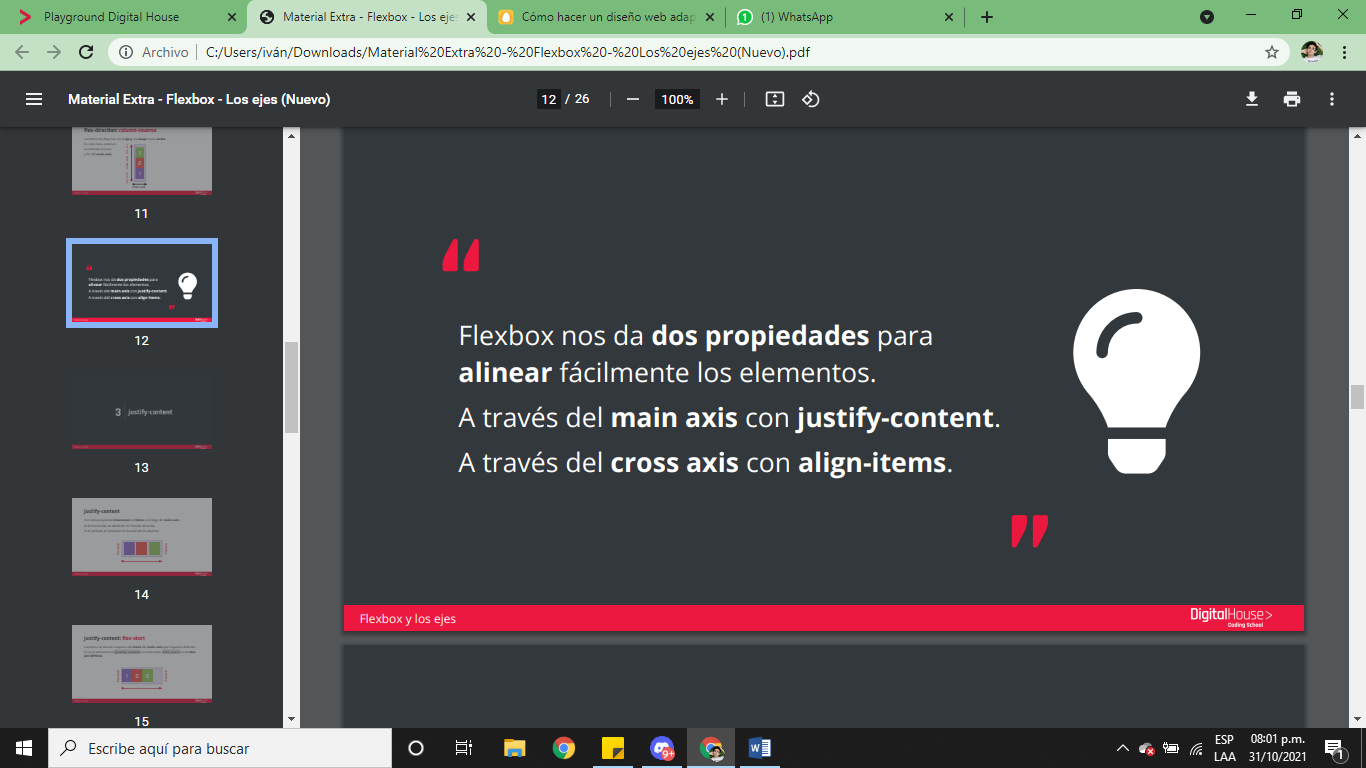
Acá el punto de inicio es en el punto superior izq y el fin en el punto inferior.

Sin embargo tenemos otros valores posibles como **"flex-direction: row-reverse"** Acá lo que pasa es igual a cuando es solo row, pero ahora el punto de inicio está en el punto inferior derecho y el punto de fin en el punto inferior izquierdo. El cross axis en este caso no se ve afectado.

Acá básicamente los elementos estarían en disposición horizontal, pero se leerían de derecha a izquierda.

**"flex-direction: column-revers**e" Acá lo que pasa es igual a cuando es solo column, pero ahora el punto de inicio está en el punto inferior y el punto de fin en el punto superior. El cross axis en este caso no se ve afectado.

Acá básicamente los elementos estarían en disposición vertical, pero se leerían de abajo para arriba.



**justify-content:** Con esta propiedad alineamos los ítems a lo largo del main axis. Si es horizontal, se alinearán en función de la fila. Si es vertical, se alinearán en función de la columna.

**justify-content:** **flex-start** Los ítems se alinean respecto del inicio del main axis que hayamos definido. Si no le aclaramos el justify-content al contenedor, flex-start es el valor por defecto.

**justify-content:** **flex-end** Los ítems se alinean respecto del final del main axis que hayamos definido.

**justify-content**: **center** Los ítems se alinean en el centro del main axis.

**justify-content: space-idently** Hace que los ítems pongan un espacio idéntico entre sí , es como si dispusiéramos que un cuadro tiene un marco de 10 cm y el de al lado también, en caso de usar este comando el costado donde se unen ambos costados el margen seria de 10cm totales, no de 20cm, como que se comparte un margen común.

**Justify-content: space-between** Los ítems se distribuyen de manera uniforme. El primer ítem será enviado al inicio del main axis, y el último ítem, al final. El espacio libre se repartirá para separar los ítems.

**justify-content: space-around** Los ítems se distribuyen de manera uniforme. El espacio libre disponible se repartirá entre todos los elementos. Del espacio que le toque a cada elemento, la mitad irá a la derecha y la otra a la izquierda (o arriba y abajo en caso de que sean columnas).

**align-items** Con esta propiedad alineamos los ítems a lo largo del cross axis. Si no aclaramos esta propiedad, el valor por defecto es stretch, en otras palabras, los ítems ocuparán todo el espacio disponible en el cross axis.

**align-items: stretch** Los ítems se ajustan para abarcar todo el contenedor. Si el cross axis es vertical, se ajustan en función de la columna. Si el cross axis es horizontal, se ajustan en función de la fila.

**align-items: flex-start** Los ítems se alinean al inicio del cross-axis.

**align-items: flex-end** Los ítems se alinean al final del eje transversal.

**align-items: center** Los ítems se alinean al centro del eje transversal.

Los ítems:

Position: absolut

Position: relative ver en clase grabada

Veremos un par de propiedades que pueden ser aplicadas directamente sobre los item para determinar su comportamiento.

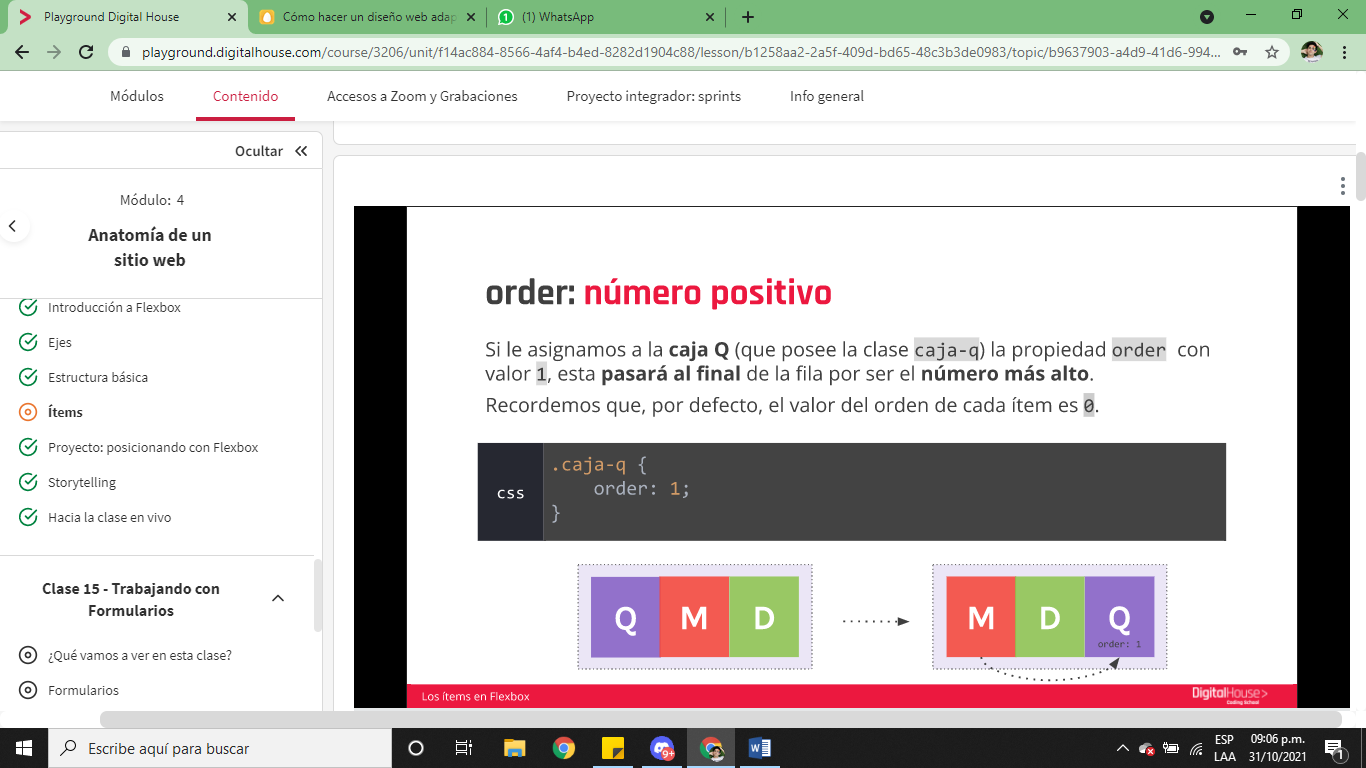
Order: Nos permite cambiar el orden de ubicación de un item determinado.

Order recibe como valor un número entero, positivo o negativo. Cada item tiene el valor orden 0 por defecto y se ordenan según caigan dentro de la estructura en HTML.

order:-1 Lleva al valor al primer lugar



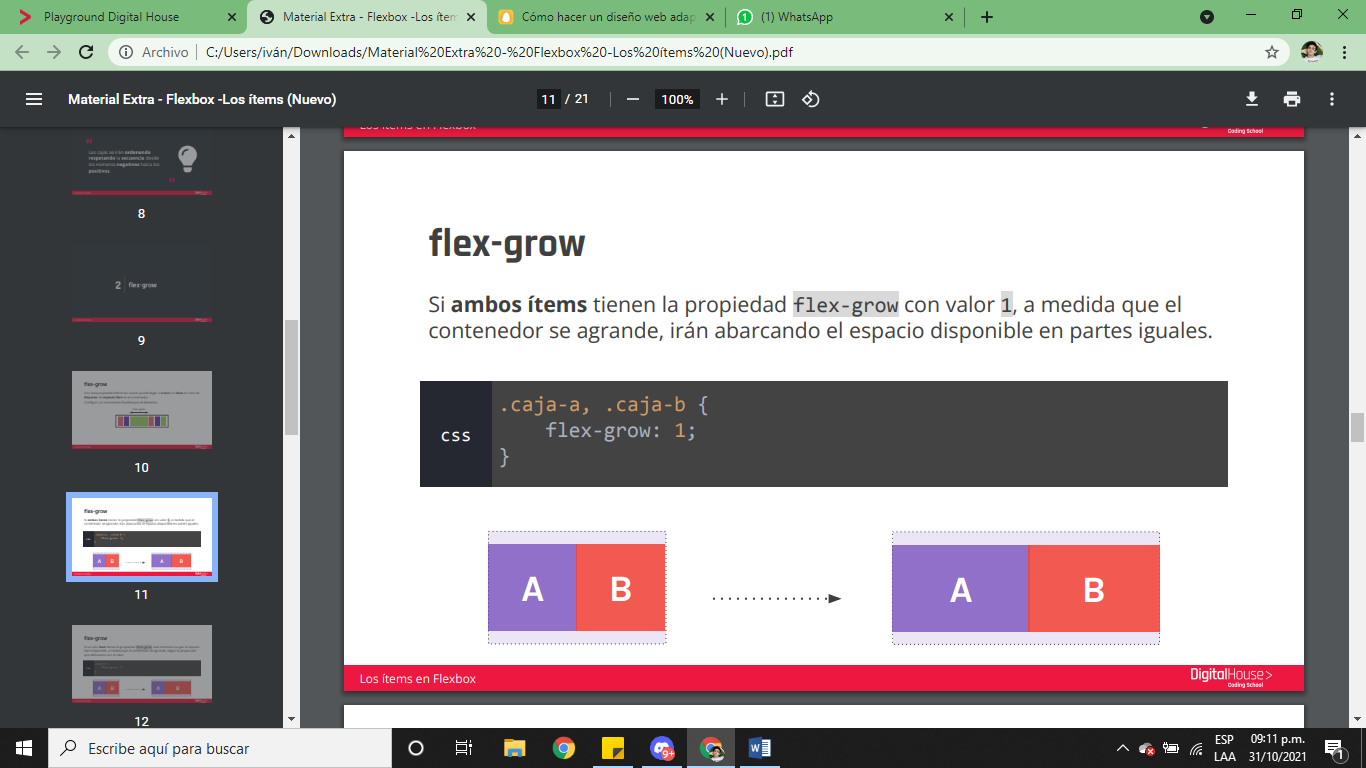
order: 1 Lleva al valor al último lugar (en el caso de que todos los demás valores sean 0)



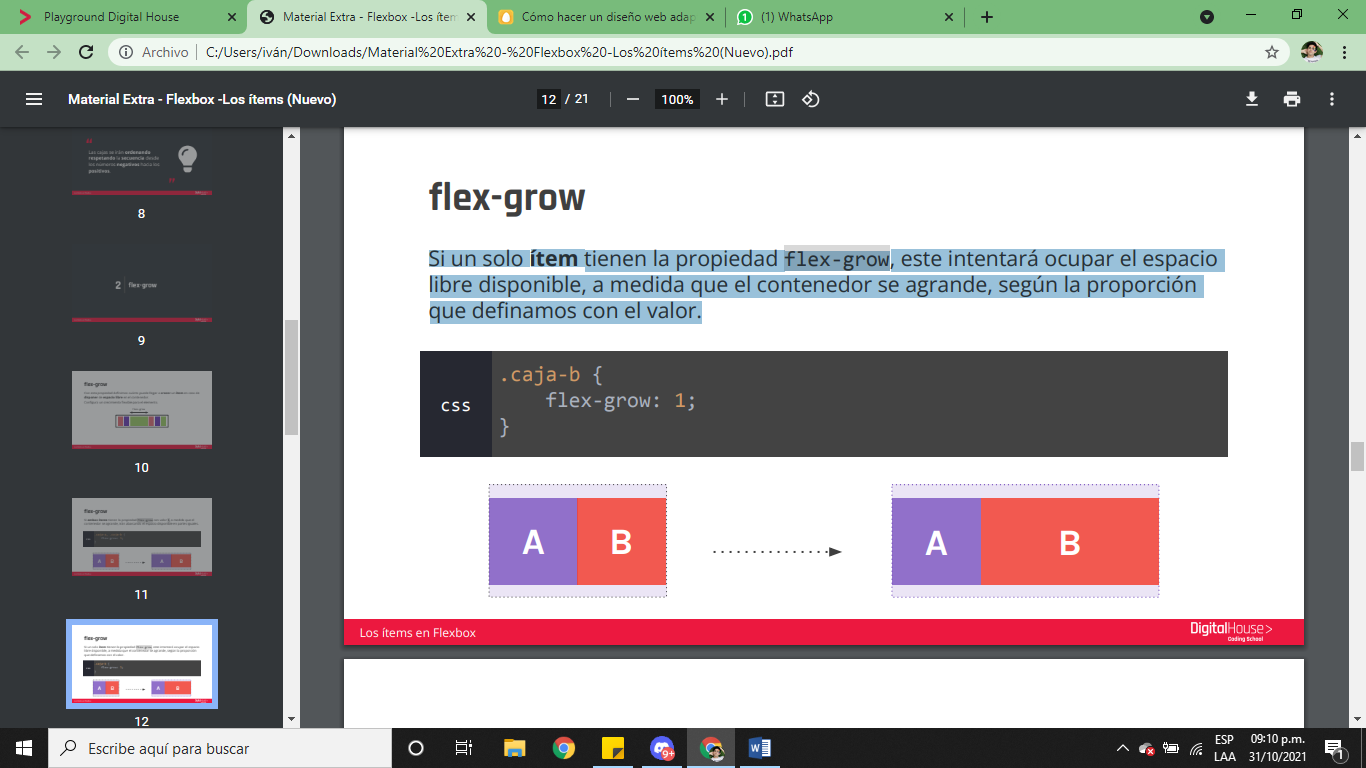
Align-self = Tiene los mismos valores posibles que la propiedad align-items, pero align-self afecta solo al item al cual se le aplico, ninguno más.

**flex-grow** Con esta propiedad definimos cuánto puede llegar a crecer un ítem en caso de disponer de espacio libre en el contenedor. Configura un crecimiento flexible para el elemento.

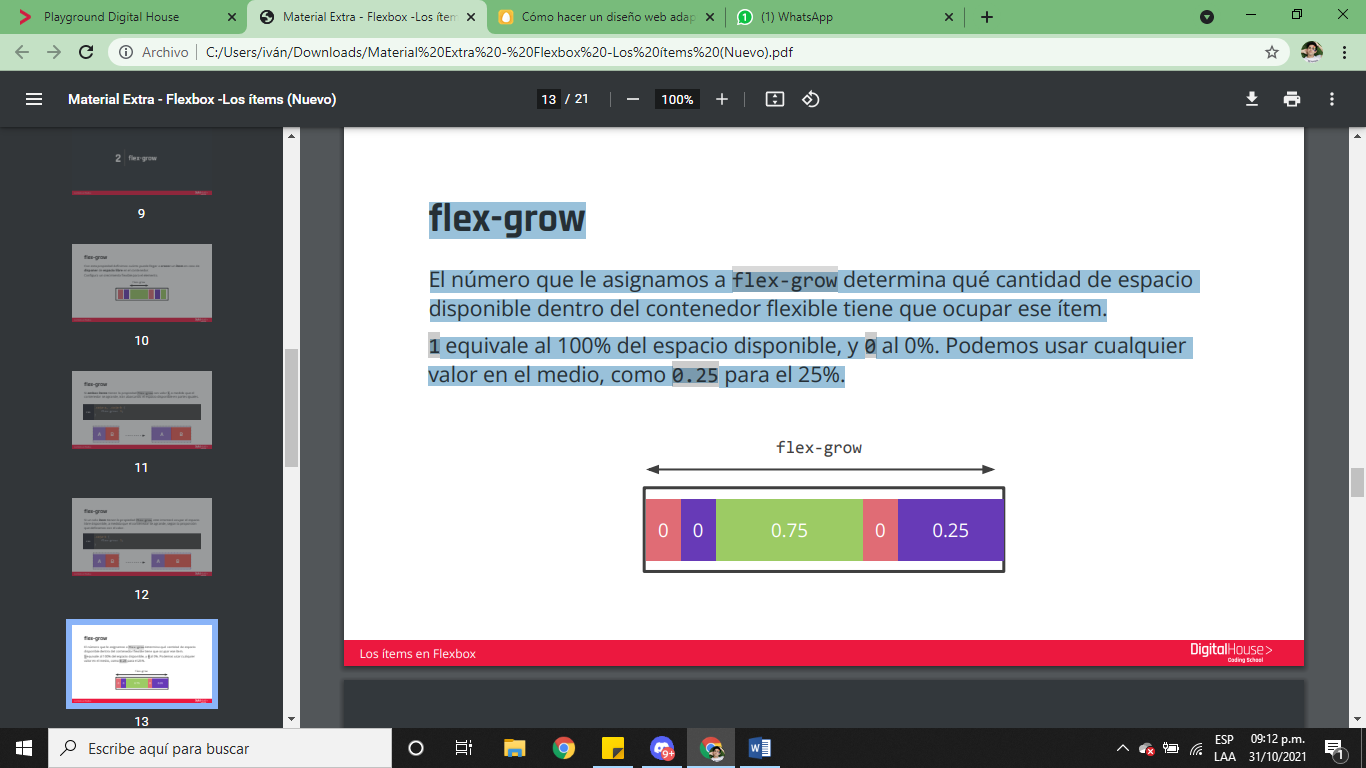
Si ambos ítems tienen la propiedad flex-grow con valor 1, a medida que el contenedor se agrande, irán abarcando el espacio disponible en partes iguales.



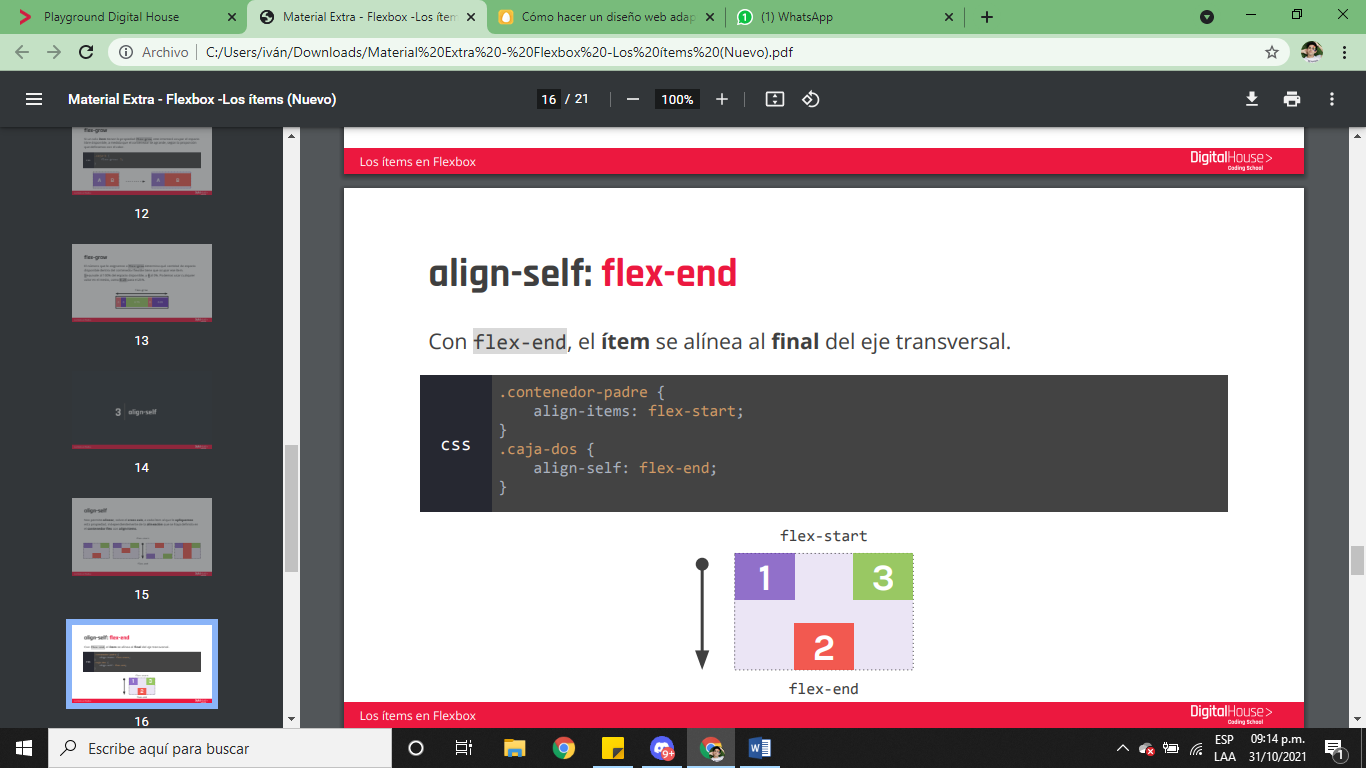
Si un solo ítem tiene la propiedad flex-grow, este intentará ocupar el espacio libre disponible, a medida que el contenedor se agrande, según la proporción que definamos con el valor.

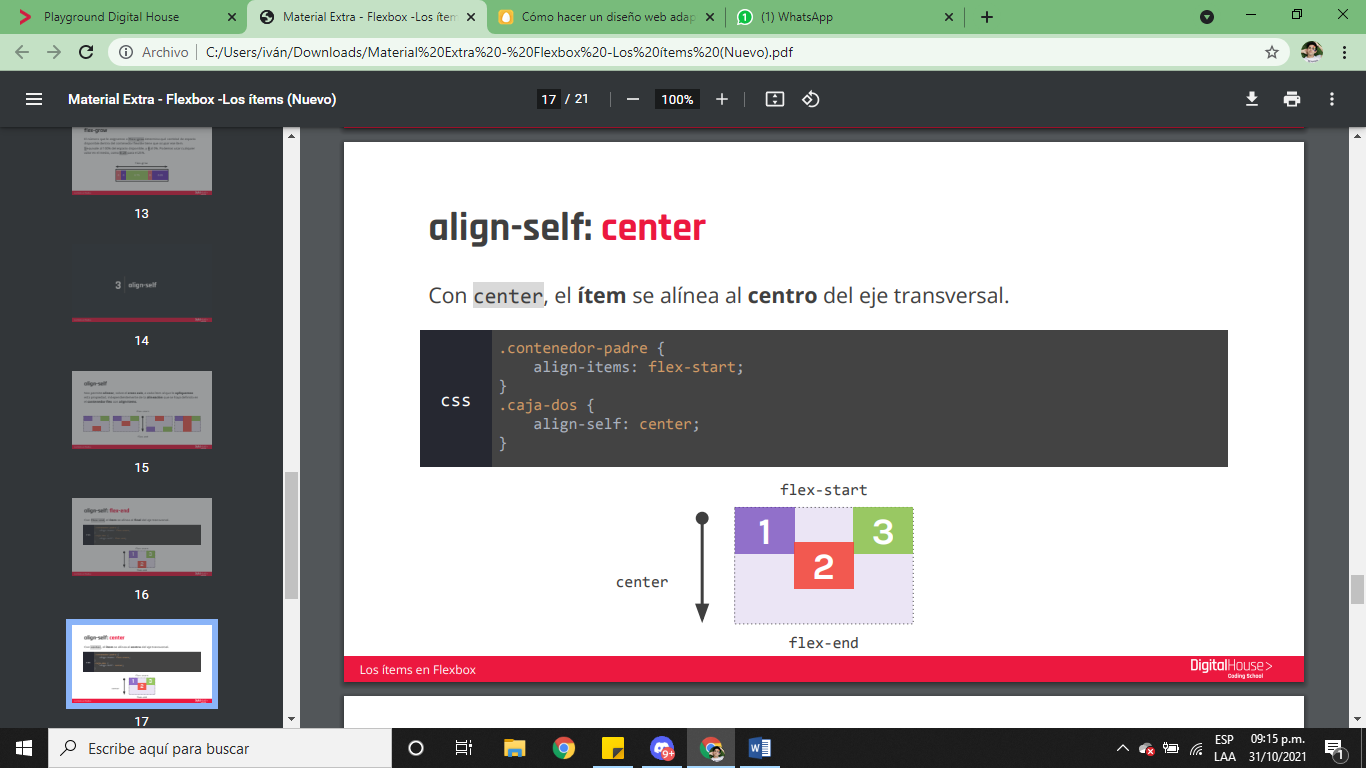


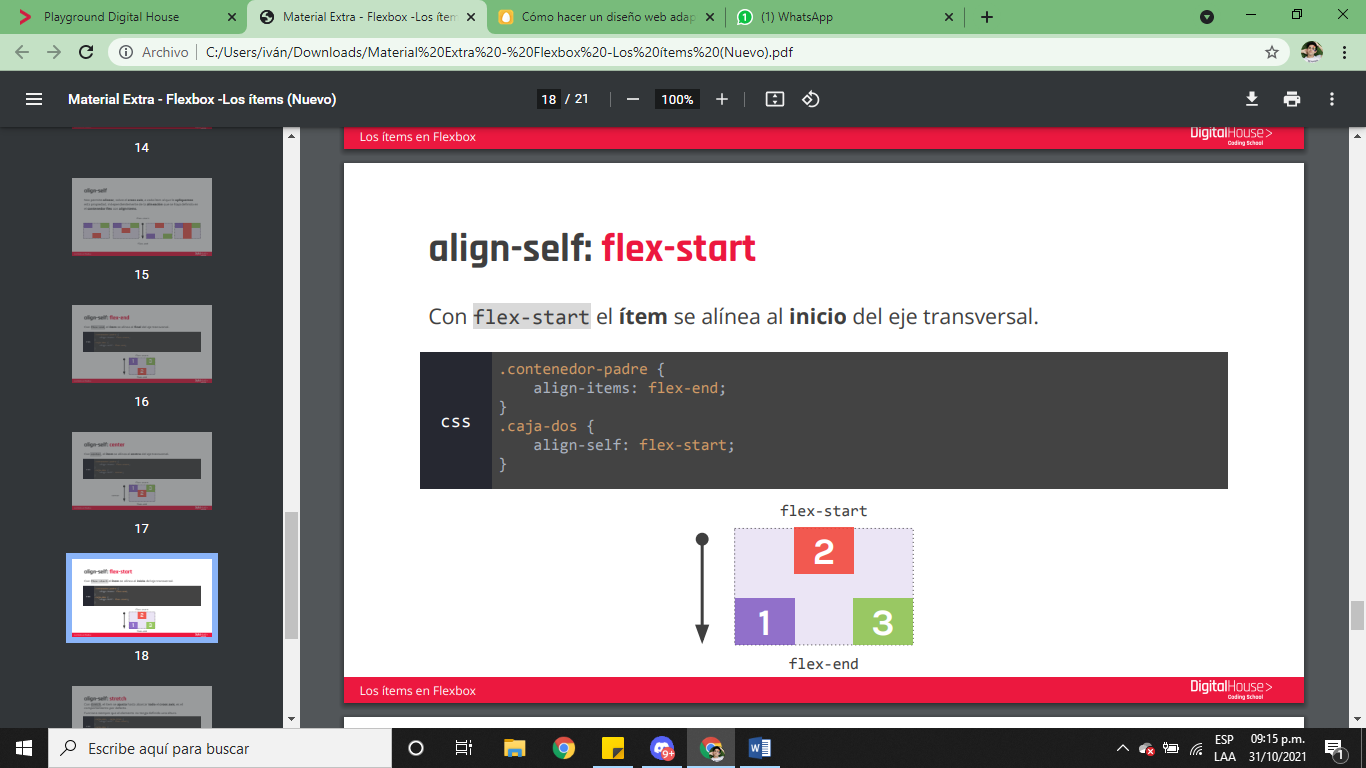
El número que le asignamos a flex-grow determina qué cantidad de espacio disponible dentro del contenedor flexible tiene que ocupar ese ítem. 1 equivale al 100% del espacio disponible, y 0 al 0%. Podemos usar cualquier valor en el medio, como 0.25 para el 25%.

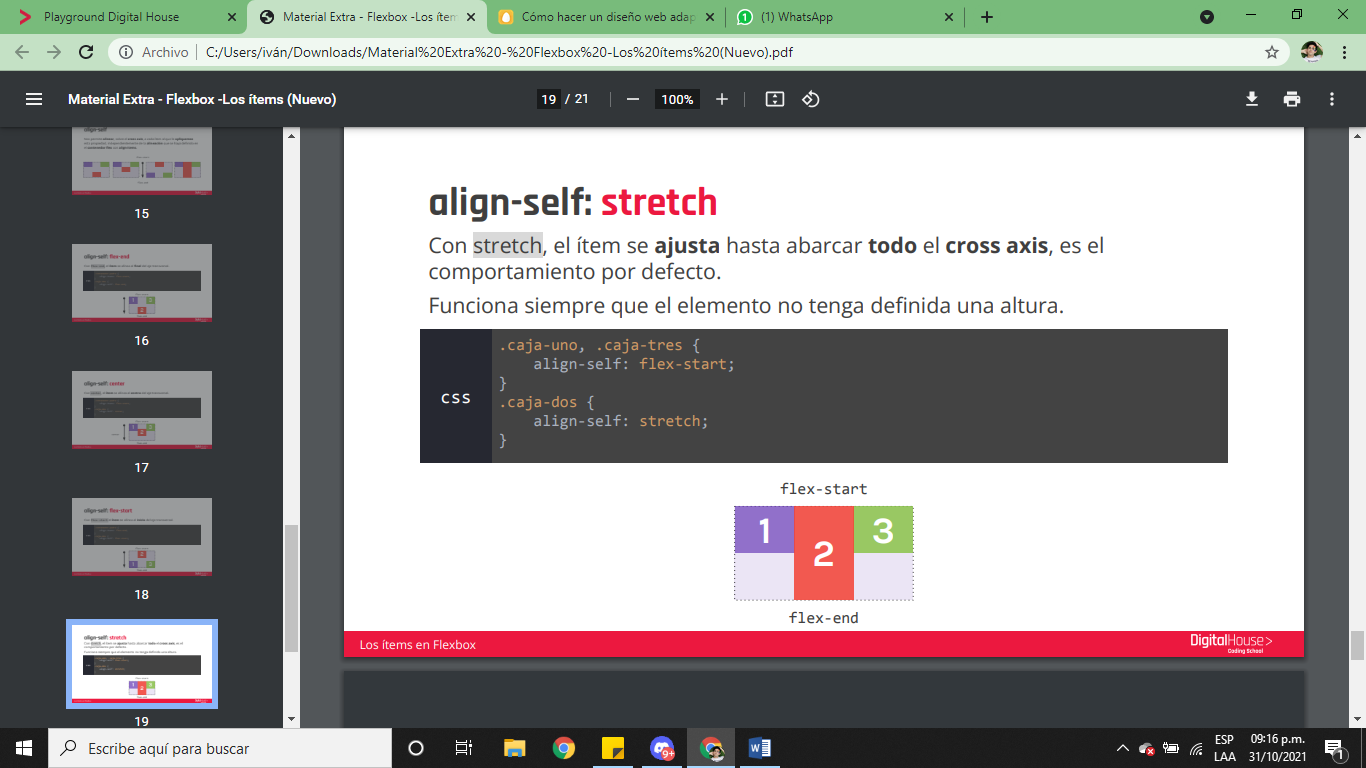












# Clase 15

FORMULARIOS en HTML:

HTML provee etiquetas para generar los formularios que nos encontramos eventualmente en el formulario.

la primera y mas importante es "form" esta nos permite definir algunas cosas basica para procesar la informacion recibida en un formulario. es importante decir que todos los campos que querramos implementar en formulario iran dentro de esta etiqueta.

form tiene dos atributos principales, "action" y "method"

Action: Sirve para definir la ruta en donde se va a procesar la informacion capturada en los campos del formulario, mientras que "method" define como se enviara la informacion. En este contexto hay solo dos valores posibles, "get" y "post"

Otra etiqueta importante es "label" define lo que esperamos que el usuario haga en el campo EJ poner su nombre.. Ayuda a definir de manera clara y precisa la informacion que esperamos recibir.

Los campos formalmente dichos, muchos de ellos los podemos llevar acabo con la etiqueta "input" la cual es una etiqueta multifuncion, esto es que con la misma etiqueta podemos crear distintos tipos de campos en el formulario.

Se implementa de la siguiente manera:

input type=""

En el type puede ir lo siguiente:

text : Genera un campo basico de una sola linea que recibe caracteres alfanumericos. Es el mas utilizado.

password: Genera un campo de una sola linea, pero aca todo lo que escribamos se vera con \*\*\*\*\*\*

email: Es muy similar al text, pero el mismo solicitara que el contenido del mismo sea compatible con un formato de correo electronico valido.

tel: Es el que cuando el visitante se encuentra desde un dispositivo mobil, activara un teclado numerico.

¿Que pasaria si necesitamos crear un campo de multiples lineas ?

Para esto HTML nos provee de la etiquetra "textarea" Esta nos permite crear el tradicional campo de comentarios, es bastante utilizado porque se estila poner esto en nustros sitios para que la gente deje sus comentarios.

Tambien hay una etiqueta que crea un campo de formulario que despliega una lista de opciones para el usuario, como por ejemplo caundo selecionamos el año en fecha de nacimiento.

Esta etiqueta es "select" generara una cajita que contendra las opciones que queremos mostrar.

¿Como generamos esas opciones?

Con la etiqueta "option" que contendra una sola opcion por etiqueta, Se puede poner muchas veces.

Ahora veremos algo importante y es crear el boton que enviara el formulario completado. Para esto usaremos la etiqueta "button" que recibe el atributo "type" con dos posibles valores reset y submit.

Reset: Resetea lo que pusimos en el formulario.

Submit: envia el formulario a la ruta especificada en el atributo action de la etiqueta form.

Algunos atributos que pueden ir en caulquier campo del formulario:

name: Nos permite darle un nombre al campo del formulario. En el ejemplo "Nacionalidad: \_\_\_\_\_\_\_\_\_\_\_\_" el atributo name representa a nacionalidad.

value: Se pone en los campos que no permiten insercion de texto por parte de los usuarios. Por ejemplo en una opcion el usuario no puede escribir, solo debe seleccionar lo que corresponda.

required: Se puede usar en cualquier campo que permita la insercion de texto por parte del usuario y hara que el campo sea obligatorio para poder enviar el formulario.

placeholder: Se puede usar en cualquier campo que permita la insercion de texto por parte del usuario y sirve para mostrar un texto de ayuda para el usuario que le explique que es lo que se espera que ponga en el campo.

Radio button y checkboxes:( Botones de opcion)  
<>  
  
Radio buttons: Son botones redondos que podemos marcar.  Para usarlo debemos poner dentro de la etiqueta input el atributo type con el valor  "radio", pero haciendo esto solo nos estaria haciendo falta el texto dentro del circulo marcable, se usar asi:  
<label>  
<input type="radio">Paises  
</label>  
  
Tanto el input como el texto estan dentro del label esto se da asi ya que de esta manera le estamos dando usabilidad a nuestra pagina porque asi el usuario puede hacer click en cualquier parte del texto y se seleccionara de manera automatica

FORMULARIOS en HTML:

HTML provee etiquetas para generar los formularios que nos encontramos eventualmente en el formulario.

la primera y mas importante es "form" esta nos permite definir algunas cosas basica para procesar la informacion recibida en un formulario. es importante decir que todos los campos que querramos implementar en formulario iran dentro de esta etiqueta.

form tiene dos atributos principales, "action" y "method"

Action: Sirve para definir la ruta en donde se va a procesar la informacion capturada en los campos del formulario, mientras que "method" define como se enviara la informacion. En este contexto hay solo dos valores posibles, "get" y "post"

Otra etiqueta importante es "label" define lo que esperamos que el usuario haga en el campo EJ poner su nombre.. Ayuda a definir de manera clara y precisa la informacion que esperamos recibir.

Los campos formalmente dichos, muchos de ellos los podemos llevar acabo con la etiqueta "input" la cual es una etiqueta multifuncion, esto es que con la misma etiqueta podemos crear distintos tipos de campos en el formulario.

Se implementa de la siguiente manera:

input type=""

En el type puede ir lo siguiente:

text : Genera un campo basico de una sola linea que recibe caracteres alfanumericos. Es el mas utilizado.

password: Genera un campo de una sola linea, pero aca todo lo que escribamos se vera con \*\*\*\*\*\*

email: Es muy similar al text, pero el mismo solicitara que el contenido del mismo sea compatible con un formato de correo electronico valido.

tel: Es el que cuando el visitante se encuentra desde un dispositivo mobil, activara un teclado numerico.

¿Que pasaria si necesitamos crear un campo de multiples lineas ?

Para esto HTML nos provee de la etiquetra "textarea" Esta nos permite crear el tradicional campo de comentarios, es bastante utilizado porque se estila poner esto en nustros sitios para que la gente deje sus comentarios.

Tambien hay una etiqueta que crea un campo de formulario que despliega una lista de opciones para el usuario, como por ejemplo caundo selecionamos el año en fecha de nacimiento.

Esta etiqueta es "select" generara una cajita que contendra las opciones que queremos mostrar.

¿Como generamos esas opciones?

Con la etiqueta "option" que contendra una sola opcion por etiqueta, Se puede poner muchas veces.

Ahora veremos algo importante y es crear el boton que enviara el formulario completado. Para esto usaremos la etiqueta "button" que recibe el atributo "type" con dos posibles valores reset y submit.

Reset: Resetea lo que pusimos en el formulario.

Submit: envia el formulario a la ruta especificada en el atributo action de la etiqueta form.

Algunos atributos que pueden ir en caulquier campo del formulario:

name: Nos permite darle un nombre al campo del formulario. En el ejemplo "Nacionalidad: \_\_\_\_\_\_\_\_\_\_\_\_" el atributo name representa a nacionalidad.

value: Se pone en los campos que no permiten insercion de texto por parte de los usuarios. Por ejemplo en una opcion el usuario no puede escribir, solo debe seleccionar lo que corresponda.

required: Se puede usar en cualquier campo que permita la insercion de texto por parte del usuario y hara que el campo sea obligatorio para poder enviar el formulario.

placeholder: Se puede usar en cualquier campo que permita la insercion de texto por parte del usuario y sirve para mostrar un texto de ayuda para el usuario que le explique que es lo que se espera que ponga en el campo.

Radio button y checkboxes:( Botones de opcion)  
<>  
  
Radio buttons: Son botones redondos que podemos marcar.  Para usarlo debemos poner dentro de la etiqueta input el atributo type con el valor  "radio", pero haciendo esto solo nos estaria haciendo falta el texto dentro del circulo marcable, se usar asi:  
<label>  
<input type="radio">Paises  
</label>  
  
Tanto el input como el texto estan dentro del label esto se da asi ya que de esta manera le estamos dando usabilidad a nuestra pagina porque asi el usuario puede hacer click en cualquier parte del texto y se seleccionara de manera automatica