**Curso desarrollo WEB**

**Plan del curso**

1. Primero aprendemos las etiquetas básicas de HTML5, como crear textos, imágenes, listas, menús, contenedores y formularios.
2. Luego en CSS3 creamos estilos para textos, colores, menús de navegación, animaciones, formularios, botones y más.
3. Con estas 2 tecnologías creamos algunos proyectos y comienza el proyecto final
4. Aprendemos el lenguaje de la web, JavaScript, donde aprendemos sintaxis, funciones, trabajar en el DOM, manipular el DOM, crear un cotizador online y más.
5. Luego nos movemos a jQuery, donde es más fácil escribir código, integrar plugins, crear galerías, reaccionar a eventos de la pag web y más.
6. Integramos las 4 tecnologías vistas en el proyecto final.
7. La parte final o casi es con sitios web dinámicos, con PHP y MySQL, donde aprendemos sintaxis, crear funciones y consultas, crear una aplicación, hacer modular nuestro sitio y más.
8. Luego unimos cada una de las tecnologías vistas y avanzamos a lo largo del curso hacia el proyecto final.

*Este curso incluye 7 tecnologías de desarrollo WEB*

* **HTML5**
* **CSS3**
* **JavaScript**
* **jQuery**
* **AJAX**
* **PHP**
* **Meucci**

*Proyectos del curso:*

* UpTask – **Administrador de proyectos** hecho en HTML5, CSS3, JavaScript, Ajax, PHP y MySQL.
* **Agenda de contactos** hecha con PHP, MySQL y Ajax, ES6, CSS Grid y Flexbox.
* Proyecto final – **Sitio web completo con pagos en PayPal y admin.**

*Editor de textos*

* **Microsoft Visual Studio**
* Cloud Atom
* Sublime Text
* Adobe Brackets

*Servidor Local*

* Wampserver (Windows)
* MAMP (Mac)
* XAMPP (Windows, Mac y Linux)
* Bitnami (Windows, Mac y Linux)

**Distintas áreas del diseño web:**

**Tecnologías front end:** incluyen HTML, CSS y JavaScript y dan forma a lo que es el **diseño** de la página web. (Desarrolladores Front End)

**Tecnologías back end:** incluyen bases de datos, servidores y lenguajes de programación como Python, PHP o Ruby (Desarrolladores Back End)

**(El desarrollo Full Stack consiste en el desarrollo front y back end)**

**¿Cómo es la estructura básica de un sitio web?**

Ejemplo, creamos un documento, normalmente se le nombra **index.html**, suponiendo que se trabaja en html.

Creamos un encabezado con su diseño y el nombre de la página/empresa, logo, imagen, etc.

Colocamos una barra de enlaces, 1 o 2 o más entradas y del lado derecho una barra lateral donde ponemos información relacionada, en la parte inferior un foro.

Creamos una copia con otro nombre.html con el fin de enlazar con el primero y así mas copias para ir enlazando (depende el tamaño de la web que quieras hacer)

**EL ROL DE HTML**

Sin html no existen las páginas web, tiene una sintaxis fácil, por ejemplo <p> = párrafos, <nav> = navegaciones, <header> = encabezado del sitio

La estructura consiste en crear una etiqueta de apertura “<html>” y otra de cierre “</html>”, luego se crean dos elementos, un “<head>” de entrada y un “</head>” de cierre, en el **HEAD** se colocan etiquetas como el título de la pág web. Luego un “<body>” de entrada y un “</body>” de cierre, en el **BODY** se coloca el contenido de la pag web

**EL ROL DE CSS**

Crea “hojas de estilo en cascada” que permite **dar diseño** a un código html.

Se encarga de definir tamaños, tipos de fuentes, dar colores, espacios, márgenes, adaptar diseños a distintos dispositivos, animaciones, etc.

La estructura consiste en la etiqueta <style> de apertura y </style> de cierre, dentro podemos encontrar selectores, como por ejemplo “p”, se pone con una “{“ de apertura y otra “}” de cierre, al contenido en su interior se lo llama “declaraciones”, ej color:blue; (siempre termina con “;”)

**EL ROL DE JAVASCRIPT**

Añade comportamiento y poder interactuar con la página web. Reacciona al usuario, permite validar formularios, procesar información, mostrar mapas o ubicaciones, etc. Posee variables, funciones y estructuras de control por lo que es un lenguaje de programación a diferencia de html y css que no lo son.

JavaScript corre sobre el DOM, que son los elementos que integran la pag web

**EL ROL DE PHP Y MYSQL**

*¿Qué es PHP?*

Es un lenguaje de **scripting** diseñado para ser utilizado con html, no se compila y es similar a C y JavaScript. Para utilizarlo necesitás instalar un servidor local.

*¿Qué es MySQL?*

Es un motor de base de datos open source, guarda y lee datos, en grandes cantidades, puede relacionar datos entre sí.

*¿Cómo funcionan juntos PHP y MySQL?*

Supongamos que un usuario entra a tu pag web, esto envía una petición a los archivos php creados y estos archivos envían una petición a la base de datos o al servidor, luego devuelven una información que los archivos procesarán y por ultimo envían el resultado de la pag web al navegador.

**Datos HTML**

1. **HTML siempre corre de arriba hacia abajo.**
2. En Visual Studio, poniendo un “!” sale una opción que dice “Emmet Abbreviation”, **esto sirve para ya abrir la estructura básica de código.**
3. **Las etiquetas “meta” no tienen cierre.**
4. <Title> es el título de la pag
5. **Lo que está adentro de <body> es lo visible**, ej ingresar <h1>Hola mundo</h1> (tipea en la página)
6. **En HTML no importa el orden**, no hace falta ingresar datos dentro de otros, sino que si están ya los va a correr sin importar donde estén ingresados.
7. La diferencia entre <h1> hasta el <h6> **es decir** **que contenido es más importante**, **además de cuanto mayor sea el H más chica es la letra**. Se pueden poner varios h1 pero no es recomendable, sino utilizar 1 solo y después los demás.
8. Los párrafos se crean con <p>, son más chicos que los h pero tienen la misma función.
9. La etiqueta <br> (break) provocan un salto de línea y se coloca en medio de la apertura y cierre del <p>. Igualmente no se usan tanto, se recomienda crear otro párrafo abajo.
10. HTML cierra automáticamente los párrafos por ejemplo, igualmente se recomienda cerrarlos.
11. Si tipeas ej <p>Lorem (es automático por Emmet Abbreviation) crea un parrafo aleatorio, esto sirve para cuando por ej el cliente no te da datos y queres ver cómo queda el diseño.
12. <blockquote> y </blockquote> es una etiqueta que solo sirve para cuando hay un testimonial, es similar a un párrafo (si verificamos en nuestra pag web, a diferencia del parrafo añade sangría al texto)
13. Seleccionando un comando, por ejemplo se selecciona solo el “em” de <em>*texto*</em>, apretando CTRL D varias veces selecciona todas las etiquetas iguales, esto permite cambiar todas a la vez y no una por una. (em o strong (negrita) no se utilizan ya que sirve mejor el CSS para esto)
14. Cuando al crear un enlace no tenemos aún el archivo, podemos poner entre las comillas un “#”, no afecta en nada y sirve de enlace solo de vista.
15. <!—aclaración - - > sirve para agregar una nota en HTML

**¿Cómo crear enlaces en HTML?**

* Los enlaces se crean presionando “a” y después “tab”, esto queda <a href=""></a>, entre la apertura y el cierre es donde se indica el nombre del enlace. Entre las comillas del href se indica hacia donde te dirige ese enlace.
* Otra forma de crear enlace, es nuevamente crear uno, pero entre las comillas indicamos un sitio web ya creado, por ej netflix.com, para esto hay que escribir “http://” primero.
* Target= ”\_blank” sirve para que al abrir el enlace se abra en otro link y no en el mismo.

**¿Cómo colocar imágenes?**

* Las imágenes se colocan con la etiqueta “img”, dándole tab autocompleta el código, mostrando <img src="" alt=""> (Solamente tiene apertura). “src” significa source, básicamente donde se encuentra el archivo de la imagen.
* “alt” significa texto alternativo, en el caso de que la imagen no cargue se mostraría este.
* (Ambos son atributos o propiedades, se encuentran varias veces dentro de la etiqueta)
* Si te dirigís a la carpeta, escribís “img/” ya te muestra el contenido, ahí seleccionas la imagen y listo.
* Para que cuando des click en el logo te lleve a la pag principal, se coloca un enlace rodeando la etiqueta de img (quedaría por ej)

<a href="/Proyecto BienesRaices/index.html">

        <img src="img/logo.svg" alt="">

    </a>

**Etiquetas para asignar espacios**

* **Header:** La etiqueta <header> </header> agrupa comandos del encabezado de la pag y se utiliza siempre para eso.
* **Nav:** La navegación se coloca en la etiqueta llamada <nav> </nav> e identifica navegaciones/links.
* **Section:** se utiliza cuando se cumpla la condición de que el primero sea un encabezado, ya sea h1, h2, h3, h4. Es decir, tiene la misma función que el header pero sirve para diferenciar la nueva sección de la del header. La pregunta es: ¿Ya hay un header? ¿Sí? Ok, creamos una sección. <section> </section>
* **Div:** Cuando tenes algo que no sabes dónde colocar, ya que no es una sección (o forma parte), ni un header ni un nav, lo asignas a un <div> </div>
* **Main:** El contenido principal de la página se identifica con la etiqueta <main> </main>, cumple la misma función que section.
* **Article:** La etiqueta <article></article>, sirve para cuando tenes algo de tipo blog, también en foros o clasificados por ejemplo, la diferencia con el section es que el section tiene la regla del H, el article se utiliza cuando un elemento habla por sí mismo, ya que tiene información independiente.
* **Footer:** El <footer></footer> es igual al header pero sirve para el final, es decir la parte de abajo en vez del encabezado o para la firma del que escribió el sitio web, por ejemplo.

**¿Cómo agregar código CSS?**

* La forma más fácil pero menos utilizada es ingresar la etiqueta <style></style> dentro de head **(NO se usa ya que si nuestra pag web tiene varios archivos, hay que copiar y pegar el código en cada una y no rinde)**
* Creamos una nueva carpeta, llamada CSS (para no confundir) o como quieras, creamos y nombramos un nuevo archivo, ej “style.css”, lo enlazamos con nuestro archivo html usando la etiqueta **“link”** (dentro del head), nos queda **<link rel="stylesheet" href="css/style.css">**
* Para darle color en este caso usa azul, en el archivo .css que creamos, escribimos **h1 { color: blue; }**

**Estructura de CSS** (se divide en 3 partes)

* La primera es el **selector**, es decir el elemento o cuerpo al que queres aplicarle lo que hace el css. En el proyecto se usa h1, ya que el h1 es el título de la pag y a ese le aplicamos color. Podes seleccionar cualquier elemento, sea img, header, section, div, parrafo, etc.
* Las **propiedades** permiten diseñar, ya sea por ejemplo “color:”, “font-size:”, etc. Lo siguiente son los **valores**, que es lo que continua a las propiedades, ya sea en color: “blue;”, “white;” y en font-size: “100px;” que sería el tamaño de las letras. Si se superponen propiedades y valores tira error en el código de la pagina
* Las **llaves { }** aplican todo lo que está dentro del elemento o cuerpo

**Datos CSS**

1. Se llama CSS y significa hojas de estilo en cascada ya que se leen los códigos de arriba hacia abajo en forma de cascada.
2. Si escribís por ejemplo 2 códigos iguales, sean h1, el ultimo va a ser el que predomine, ya que reemplaza al primero. Para que cuando quieras modificar color, background o font-size de un, por ej, parrafo solo no se modifiquen todos existen **clases y ID'S**. **\*1**
3. **Para crear un id**, lo que haces es agarrar un elemento, sea header, p, main, div, etc, le ingresas id=”” y entre las comillas le asignas un nombre, queda por ej “**<header id="site-header">”** 
   1. IMPORTANTE: para darle estilo a un id usas # seguido del nombre, ej **#site-header { }**
4. **Para crear una clase**, lo que haces es agarrar un elemento, sea header, p, main, div, etc, le ingresas class=”” y entre las comillas le asignas un nombre, queda por ej **“<h3 class="encabezado">Seguridad</h3>”**
   1. IMPORTANTE: para darle estilo a una class usas . seguido del nombre, ej **.encabezado { }**
5. Dentro de un class, agregando un espacio al nombre inicial de la clase, seguido de otro nombre, cuenta como si fueran 2 clases distintas, ahorrando código, por ejemplo “class= site-header inicio”, en este caso site-header es una clase, e inicio es otra. **Para darle estilo a ambas clases**, el código seria **.site-header .inicio { }**
6. Para agregar un **comentario** en CSS, escribís /\* (no cuenta como código, sirve para organizar) y cerras con \*/
7. **Cuando quieras buscar un archivo, pero no se encuentra a la misma “altura” de la carpeta donde está el doc**, ponemos **“..”** para que vaya para atrás, ej: “background: url(../img/header.jpg);”
8. Height sirve para dar tamaño a la img
9. En vez de px se puede usar vh (view x height), que hace que la imagen se vea depende la resolución de cada pantalla. Ej 100vh (100% de la pantalla)
10. min-height: sirve para indicar un mínimo de px, ej min-height: 600px;
11. **background-position:** sirve para posicionar la img, por ej **background-position: center center;** (center center muestra la img desde el centro)
12. **background-size:** hace que se tome todo el ancho y largo disponible de la img pero siempre guardando las proporciones indicadas, ej **background-size: cover;**
13. **max-widht:** (altura máxima) pone límite de tamaño en el valor indicado, ej max-width: 1200px;
14. **margin-left: / margin-right:** sirven para centrar, ej margin-left: auto;

también existe la propiedad **margin:**, que toma los 4 valores, derecha, izquierda, arriba y abajo. El mejor ejemplo es imaginar un reloj, empieza a las 12 (arriba), 3 (derecha), 6 (abajo) y 9 (izquierda), se escribe por ej “margin: 0 auto 0 auto;” **(otro dato es que cuando se repiten como en este caso, 0 arriba y abajo, auto derecha izquierda, se puede simplificar y que quede “margin: 0 auto;”)**

1. display: flex; coloca en este caso, el menú o los links del lado derecho del logo.
2. **justify-content:** space-(en este caso usamos between, pero hay varias opciones), lo que hace es separar el logo de los links, cada opción los aleja o acerca más. Ej “justify-content: space-between;” (sirve solo horizontalmente)

A diferencia de justify-content, se puede usar **align-items:** y por ej **align-items: center;** que lo que hace es mover verticalmente.

1. padding: lo que hace es establecer un espacio sin tirar todo para abajo, puede ser en todas las direcciones, ej “padding-top: 30px;” **(lo que hace a diferencia del margin es establecer un espacio entre el mismo elemento, el margin separa directamente al elemento)**
2. **text-decoration:** subraya o no los enlaces o texto, ej text-decoration: none;
3. **pseudoselectores** como por ej .navegacion a:last-of-type { } agarra solo el último elemento, o **.navegacion a:hover { }** hace que cuando te pares arriba de un link pase algo, por ejemplo se ponga de color rojo etc
4. **flex-basis:** sirve para separar elementos, ej “flex-basis: 33.3%;”, hay un método de prueba, que es “flex-basis: calc(33.3% - 1rem);” lo que hace es establecer la misma función que la primera, pero podes probar restándole variables, es más útil.
5. **Text-transform:** cambia las opciones de texto, a esto nos referimos como por ej mayúsculas (“text-transform: uppercase;”), mayúsculas en la primer letra (“text-transform: capitalize;”), todas minúsculas (“text-transform: lowercase;”), etc. Conviene usar text-transform: y no escribir en mayusc cada H pq si tenes que cambiar todos así es más fácil.

**\*1 ¿Cuál es la diferencia entre id y class?**

* En CSS se le puede aplicar lo mismo a ambos, pero las class están hechas para reutilizarse, podes copiar y pegar el mismo nombre y no pasa nada, en cambio las id solo puede haber 1 con un nombre.

**¿Cómo cambiar la fuente?**

Para aplicar un cambio de fuente a todos los elementos, ingresamos en el doc.css **body { font-family: “la fuente seleccionada, ej Calibri) }**

Igualmente si queremos cambiar la fuente de un sector específico, podemos ir a por ej .barra y cambiar la fuente ahí, no afecta al resto, solo a ese sector.

Normalize ¿Para qué sirve?

Lo que hace normalize es adaptar todos los dispositivos y navegadores a la pag web, se descarga desde el sitio, creas un archivo normalize.css y pegas el contenido de la pag, añadís un link al normalize.css en tu doc.html y listo

Tener en cuenta que siempre es recomendable tener tu doc.css propio último, ya que es el último que html va a leer, esto sirve para que no haya errores ni lo reemplace.

**REMS ¿Para qué sirven?**

Con la cantidad de dispositivos y tamaños los PX no rinden demasiado, ya que justamente dependen del tamaño de la pantalla. Para esto existen los rems que hacen que el tamaño se ajuste a la pantalla.

Antes de utilizarlos hay que establecer un rango para que no sea muy grande,

Utilizamos en el doc.css el elemento de html, establecemos font-size: al 62.5%,

Queda “html { font-size: 62.5%; }” lo que hace que por ejemplo 100px sean 10rems, 1200px 120rems etc.

**IMPORTANTE: siempre usar rems y no px**