# Básico

**Sobre sitios web**

Es un conjunto de archivos y directorios que están almacenados en una computadora de acceso público. Esa computadora se llama servidor web y está conectada a internet las 24hs del día. Puede ser un servidor compartido (hosting) o uno privado (servidor dedicado). A esa computadora se puede acceder de dos formas: mediante un nombre de dominio o un número IP.

**El modelo cliente-servidor**

Cuando abrimos un browser y pedimos una página web, lo hacemos por su nombre normalmente. A lo largo de toda la red de internet, hay una serie de máquinas que hacen de agenda y nos dan la IP.

Cuando nuestra petición llega al servidor, el mismo resuelve:

- Si el sitio está en ese servidor
- Qué directorio o carpeta corresponde a ese sitio
- Qué archivo estamos pidiendo (si no pedimos ninguno hay uno por defecto)
- Qué tecnologías conforman ese archivo

**Explorando un sitio**

Para ver un sitio, los usuarios utilizan navegadores que se encargan de dibujar la web en pantalla. Una página web está formada por varios distintos archivos que se vinculan entre sí por LINKS. Estos archivos son de tipo HTML.

# HTML

Es un lenguaje de marcado para crear documentos web que permite indicar dónde queremos cada elemento. Solo se encarga de lo estructural, no del diseño. Ha sido estandarizado por la W3C.

**Convenio de archivos**

Deben cumplir tres reglas el nombre:

1. No debe tener espacios, acentos, eñes, ni símbolos.
2. Deben estar escritos en minúsculas.
3. Deben tener la extensión .html.

**Comentarios en HTML**

Es parte del código que el navegador no mostrará. <!-- bloque comentado →

**Estructura básica**

El HTML tiene una estructura de elementos que debe insertarse siempre en cada documento. Tiene dos elementos principales:

- Head: toda la información para el navegador o los buscadores. Debe tener obligatoriamente una etiqueta <title> </title> adentro con contenido, que es el título de la pestaña.
- Body : todos los elementos a mostrar.

Ambos están dentro de un elemento principal llamado <html>.

**Doctype**

Antes de la etiqueta HTML debemos indicarle al navegador qué tipo de HTML estamos haciendo mediante la declaración de un DOCTYPE.

<!Doctype html>

**Warning**
Nos queda por atribuirle el atributo lang al html para que sepa qué lenguaje tiene.
<html lang=”es”>
PAra indicar el idioma de alguna palabra usando el atributo lang hay que aplicárselo a alguna etiqueta. Si semánticamente ninguna etiqueta cumple con ese objetivo, se utiliza la etiqueta  .

**Codificación**
El HTML permite definir la codificación del texto a mostrar. Esto se llama CHARSET. Le indica al browser qué mapa de caracteres usar. Lo define con una etiqueta <meta charset= “UTF-8”/>. UTF-8 es el mapa completo de todos los que existen en el teclado (A-Z, 0-9, tildes, eñes).

# Etiquetas

**Tipos de etiqueta**
Todas las etiquetas que van en el body se dividen en dos grupos:

- Elementos de bloque: son los que, sin ser modificados por CSS, ocupan el 100% del ancho de su contenedor y se mostrarán uno abajo del otro.
- Elementos de línea: solo ocupan el ancho que diga su contenido y se verán uno al lado del otro.

Es importante porque ciertos elementos no validan al insertar un elemento de bloque dentro de uno de línea.

**Atributos de las etiquetas**
Todas las etiquetas aceptan atributos, que es cualquier característica que puede ser diferente entre una etiqueta y la otra. Se escribe en minúscula y el valor que entrá va entre comillas. Una etiqueta puede tener más de un atributo, separados por espacios entre sí.

**Dispositivos alternativos**
Los usuarios no-videntes que utilizan programas que les leen la pantalla encontrarán un cambio sutil en la forma de escuchar palabras destacadas:

- Strong hace un texto reforzado que será leído remarcando la palabra pero sin afectar al tono de voz.
- Em hace énfasis, por lo cual hay un cambio significativo en el tono de voz.

**Posicionamiento SEO**
Sin contenido, los buscadores no nos posicionan. Por eso hay que tener texto. Los motores de búsqueda (principalmente google) entienden los elementos strong y em como el nivel más básico del texto relevante. En el océano de texto, todo elemento destacado con estas etiquetas será considerado como criterios fundamentales de lo que trate la página web.

# Encabezados

Encabezado es semánticamente el texto que encabeza el contenido que sigue. Hay seis niveles: de h1 al h6. Representan una jerarquía por nivel de importancia, no por orden de aparición.

# Texto

**Listas en web**

Se usan para agrupar elementos que tienen una relación entre sí. Cualquier enumeración de dos o más propiedades o características de una persona/producto, es una lista. La etiqueta li acepta cualquier otra etiqueta dentro (párrafo, imagen, strong, otras listas, etc.). Es un elemento de bloque. Hay tres tipos distintos:

- Ordenadas: establecen un orden de lectura de sus ítems y una jerarquía entre los elementos.

<ol>
    <li>
    </li>
</ol>

- Desordenadas

<ul>
    <li>
    </li>
</ul>

- De definición: para hacer glosarios o diccionarios. Etiqueta <dl> </dl>, definition list. Tendrá adentro siempre un juego de pares:
    1. Un término a definir <dt> Término </dt> (definition term).
    2. Las aceptaciones para ese término (definition data) —> <dd> Significado del término </dd>

**Acrónimos y abreviaturas**

Un acrónimo es la sigla que representa un conjunto de palabras, como YPF. Una abreviatura es una palabra resumida, como etc flia, ej. Tanto acrónimos como abreviaturas se insertan con la etiqueta <abbr> </abbr>. Dentro de la etiqueta va la sigla o abreviatura y con el atributo title se indica el significado. Cuando el usuario pasa el mouse por encima, aparece un tooltip con el texto del atributo title.

**Citas**

Hay tres etiquetas para citar texto

- <q> inserta una quote que es un texto de una sola línea. Algunos navegadores le agregan comillas automáticamente.
- <cite> se incerta para referenciar la fuente de un texto citado en un quote. Es decir, quién dijo o publicó este quote.
- <blockquote> está pensado para citar texto de gran volumen, como puede ser el extracto de un libro.

Tanto q como cite son elementos de línea, deben ir adentro de un párrafo. Por lo general, ambos van dentro del mismo párrafo. Blockquote es de bloque y puede tener adentro un párrafo con el texto. Tanto blockquote como q aceptan el atributo cite que indica la URL del texto citado.

# Navegación

**Arquitectura de un sitio**

Un sitio web está formado por archivos y carpetas. Estas carpetas se suelen usar para organizar las imágenes, videos, scripts, css, etc. No es recomendable tener los archivos HTML adentro de una subcarpeta. El archivo principal de nuestra página web se debe llamar index.html. Los archivos se navegarán usando hipervínculos.

**Navegación de un sitio**

Los vínculos se basan simplemente en usar un elemento del HTML y hacerlo clickeable para que acceda a otro archivo diferente, que puede ser HTML, JPG, EXE, etc.

El archivo que busco puede estar en nuestro sitio web (archivo local) o en un dominio distinto (recurso externo). Los archivos locales tienen que estar en carpetas y subcarpetas, excepto los archivos HTML.

**Hipervínculos**

Un vínculo o hipervínculo es cualquier elemento que esté dentro de una etiqueta <a> </a>. Tiene un atributo obligatorio que es el href, al que le decimos a qué documento apunta el link.

Es un elemento de línea y por eso no puede tener dentro de un encabezado, parrado o lista. Sí puede tener dentro un strong, em, abbr, imagen.

**Rutear archivos**

Es el listado de carpetas y subcarpetas que se debe seguir para llegar a un archivo. En los vínculos locales, esta ruta se realiza desde el archivo que quiere acceder al otro link.

- Si se quiere ir a un archivo que está en la misma carpeta, solo el nombre del archivo.
- Si se quiere ir a un archivo que está en una subcarpeta, se escribe: subcarpeta/archivo.html
- Si se quiere salir de una carpeta se debe poner dos puntos seguidos: ../archivo.html

**Título del link**

Los vínculos aceptan también el atributo title, que permite mostrarle al usuario cuando pose el mouse sobre el link. Se sugiere no darle el mismo valor que el texto adentro del <a>, a menos que sea un ícono.

**Anclas**

Sirven para hacer scroll a un lugar del mismo documento. Se usa también la etiqueta <a>. El href debe tener como valor una palabra con un numeral adelante: href=”algo”

Hará scroll hasta cualquier etiqueta a la que hayas puesto el atributo id igual a ese valor pero sin numeral: id=”algo”

**Target**

Por defecto cuando hacemos click en un vínculo se abre en la misma pestaña. El atributo target permite definir si se quiere abrir en una pestaña nueva o no.

- target=”_blank”: abre en una nueva
- target=”_self”: en la misma ventana

**Mandar un mail**

Los vínculos aceptan distintos protocolos a la hora de disparar el href. Con mailto, al hacer click al vínculo, se abre el gestor de mail que el usuario tiene instalado por defecto.

<a href=”mailto:camibarba@hotmail.com”>

Click y contactame!

</a>
Además, se puede asignar un asunto del mail. Después del correo ponemos un signo de pregunta y seguido subject=Tu Asunto
<a href="mailto:camibarba@hotmail.com?subject=Contacto web">
Click y contactame!
</a>

# Imágenes

**Enriquecer el HTML**

Se insertan con la etiqueta <img />. Para funcionar requiere indicarle en dónde está el archivo a mostrar, con el atributo src (source). Se comporta como elemento de línea pero no lo es, dado que no acepta contenido dentro. Para validar, tiene que tener el atributo alt que es un texto con una breve referencia sobre la foto insertada.
Párrafos y encabezados **no** son contenedores de imágenes. Sí puede ser un list-item o una etiqueta .
Otros atributos importantes:

- width: ancho de la imagen. No lleva unidad, se asume pixeles.
- height: alto de la imagen. Same.
- title: es un tooltip que aparece al pasar el mouse por encima.

**Optimización**

La imagen debe reducirse en Kbytes: es pesada si supera los 300kb y es óptima por debajo de los 80kb.
Una web de primero no debería superar los 2Mb entre todos sus archivos.

**Imágenes como links**

Existen tres formas de hacer clickeable una imagen.

- Con el atributo onclick="" que lleva javascript adentro
- Encerrando la etiqueta <img /> en un <a> </a>
- Con <map>, definiendo distintas áreas de la imagen para que sean clickeables. Requiere 4 elementos:
    - Una <img /> con el atributo usemap="#algo"
    - Una etiqueta <map> con id="algo"
    - Las etiquetas <area /> dentro del <map> </map>, que indican las zonas clickeables.

A su vez, cada área está formada por 4 atributos:

- alt
- shape: forma geométrica que se quiere insertar. Puede ser rect, circle, poly.
- href: archivo o ancla al clickear la figura
- coords: coordenadas de la figura dentro de la imagen, cada valor va separado por coma de los otros. Varían según la forma geométrica:
    - rect son 4: x,y de origen + x,y de destino
    - circle son 3: x,y de origen + diámetro del círculo
    - poly requiere tantos nodos como sean necesarios.

# CSS

**Premisas**

Es el lenguaje web para aplicar formato visual al HTML. Bien implementado permite cambiar todo el diseño de un sitio web sin modificar el HTML.

**Tipo de formato**

Hay 3 grupos de declaraciones CSS que podemos implementar:

1. Formato de texto: familia tipográfica, color y tamaño de fuente, interlineado, negritas, itálicas, subrayados. Estas son las heredadas.
2. Formato de cajas: ancho, alto, bordes, color e imágenes de fondo, márgenes sombras, rotación.
3. Ubicación de elementos: posicionamiento, flotaciones, mostrar/ ocultar cajas.

**Propiedades**

Color para cambiar el color del texto

Font-size para indicar el tamaño del texto de cualquier elemento

**Formas de aplicar CSS a HTML**

1. Todas las etiquetas del HTML tienen el atributo style="" y entre comillas se escribirá la regla CSS para formatear únicamente ese elemento.
2. Existe una etiqueta <style> </style> que va en el HEAD y contendrá las reglas CSS para formatear.
3. Existe una etiqueta <link /> que va en el HEAD y se usa para cargar un archivo externo con extensión .css que permite formatear múltiples archivos HTML. El link no funciona si no tiene el atributo rel (qué relación tiene el documento importado con este HTML). Debe tener el valor stylesheet.

   <link rel="stylesheet" href="archivo.css" />

**Formatear un elemento**

Por etiqueta: p {formato para todos los párrafos}

Por atributo ID: #ppal {solo formatea la etiqueta con id="ppal"

Por atributo class: .ppal {formatea todas las etiquetas con class="ppal"

**Reglas sintácticas**

1. Cada declaración de CSS está formada por un juego de pares —> propiedad: valor
2. Si a un elemento se le aplica más de una propiedad se deben superar con punto y coma.
3. No se ve afectado por el espacio en blanco.
4. Se pueden escribir de corrido o una debajo de la otra
5. Los comentarios se hacen como en Javascript /* hola */
6. Siempre que la propiedad represente un número, el valor debe indicar bajo en qué unidad se expresa. Unidades de tamaño exacto, es decir, con un tamaño fijo independientemente del dispositivo: cm, mm, in, pt. Unidades con tamaño relativo que se calculan según el tamaño que tiene el elemento del contenedor: px, em, ex, %.
7. Entre el número y la unidad no pueden existir espacios.
8. Si se necesita aplicar el mismo formato CSS a más de un elemento diferente, no hace falta escribir dos veces todas las propiedades. Se escriben los dos o más elementos separados por comas. #caja, p {}

**Cascada CSS**

Define una ruta que deben cumplir los elementos dentro del HTML. Se trata de una lista de elementos separados por espacios. Si algún elemento cumple esta ruta, le aplica el formato CSS al último elemento de la lista.

**Precedencia de declaraciones**

Cuando reglas distintas apuntan al mismo objeto, las propiedades se suman si son distintas y si tienen alguna propiedad repetida, solo queda una. Esto es lo que se llama **precedencia**.

1. ID
2. Class
3. Etiqueta

La declaración !important corta la precedencia. Si necesitamos más de 5 algo estamos haciendo mal.

# CSS texto

**Definir la familia tipográfica**

CSS tiene la propiedad font-family para indicar qué fuente utilizar en un texto. El valor de la tipografía debería estar entre comillas, principalmente si el nombre tiene espacios. Tiene una restricción: la tipografía seleccionada debe estar instalada en la PC del usuario que entra a la web. Si esto no sucede, el browser sustituirá la fuente por alguna de sistema. Se pueden indicar varias familias separadas por coma, la primera de la lista que esté instalada será usada.

**Formato básico de la fuente**

- El color de fuente se define con **color**, acepta nombre, hexadecimal o notación rgb.
- Para definir el tamaño de la fuente se usa **font-size**. Es numérica, acepta px, porcentajes, pot, em, ex… etc.
- El tamaño de la fuente va acompañado de un interlineado. El **line-height** indica cuánto mide cada línea de texto si se llega al final y hay una segunda línea. Debería ser igual o mayor al tamaño de fuente.
- Para manipular la negrita se usa **font-weight**. Se quita con el valor normal y aplica con bold.
- La itálica se indica con **font-style**, que puede ser normal o italic.

**Mayúsculas y minúsculas**

CSS tiene dos propiedades que manejan las mayúsculas.

Text transform acepta:

- uppercase: pasa todo el texto a mayúsculas.
- lowercase: pasa todo el texto a minúsculas.
- capitalize: primera letra de cada palabra en mayúsculas
- none: elimina esta propiedad (se ve como fue escrito).

Font-variant manipula las versátilas:

- small-caps: pone todo en mayúsculas y las letras que ya eran mayúsculas se ven más altas.
- normal: quita la propiedad.

**Espaciado de texto**

El interlineado maneja el espacio vertical del texto, pero hay tres propiedades que manejan el espaciado horizontal que aceptan valores numéricos con su unidad y pueden ser tanto positivos como negativos:

- letter-spacing: define el espacio que habrá entre cada carácter tipeado
- word-spacing: el espacio de cada palabra.
- text-indent: tabulación de la primera línea de texto.

**Formatear vínculos**

Los vínculos son los únicos que no heredan el formato del texto global. Son de color azul y subrayado. Se debe especificar cómo se formatearan. Además, tienen estados:

a {} —> el vínculo común y corriente

a:hover {} —> cuando se pasa el mouse por encima

a:active {} —> mientras el usuario mantiene el click

a:visited {} —> vínculos visitados

Generalmente a los vínculos se les da formato en su estado normal y el :hover. Si en nuestro diseño no queremos vínculos subrayados, hay que aplicarle text-decoration: none (acepta underline, overline, line-through o none).

En el :hover no se recomienda cambiar ni el tamaño del texto ni la negrita/itálica porque genera que el texto “baile” y sea poco práctico de usar.

Se recomienda solo cambiar color y subrayado.

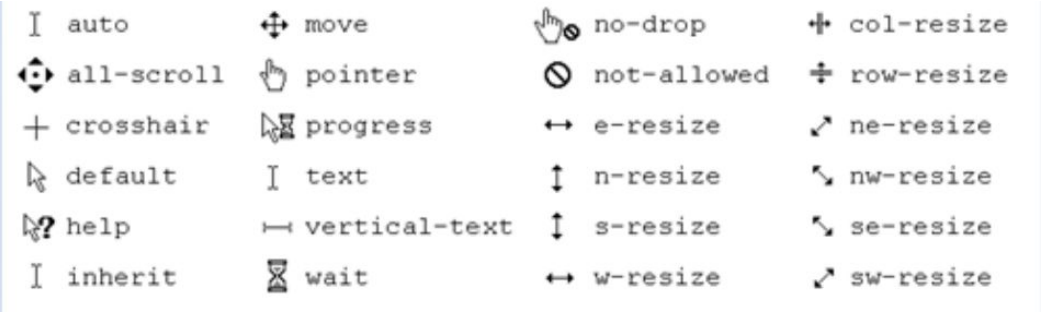
En realidad **todos** los elementos aceptan :hover

**Una etiqueta genérica**

Ya sabemos que el  es una caja genérica para meter adentro lo que queramos. Su paralelo a nivel texto es  . No tiene formato por defecto y solo se usa cuando adentro de un texto se necesita cambiar el CSS.

**Manipular el cursor**

Se puede definir qué cursor se mostrará al pasar el mouse sobre alguna etiqueta con la propiedad cursor. Los valores más comunes son:




También se puede usar una imagen en PNG con

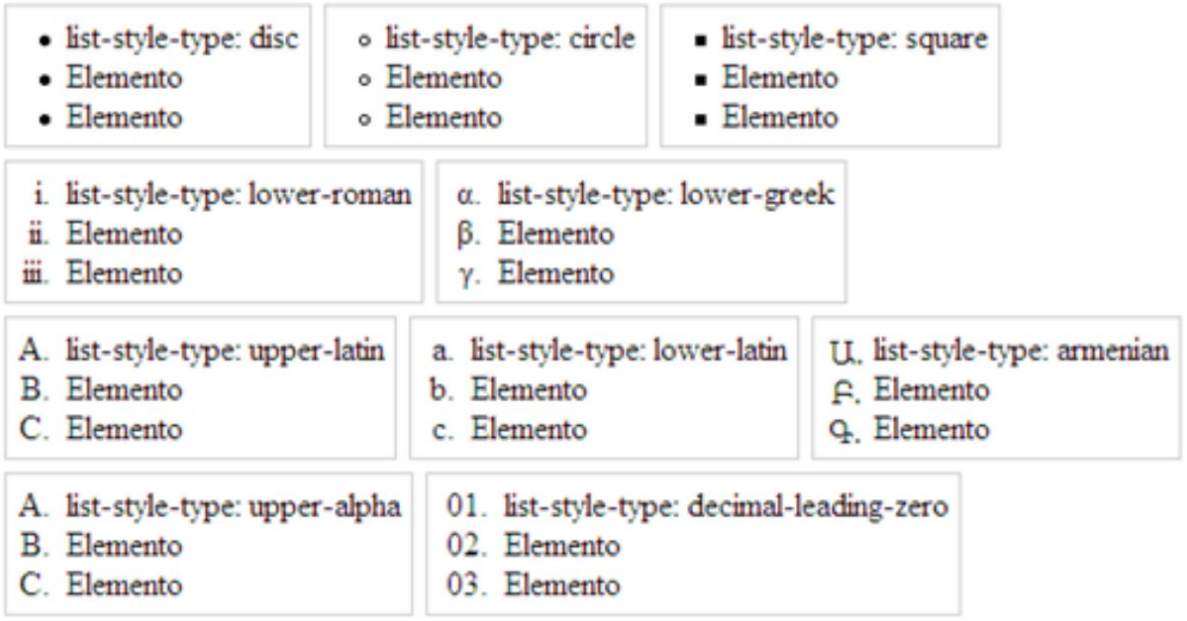
cursor: url (‘cursor.png’), default;

**CSS para listas (ul/ol)**

Las listas vienen por defecto con un margen y padding. Tienen un tipo de viñeta.

- list-style-type: TIPO_DE_VIÑETA;
- line-style-position: POSICION_VIÑETA
- list-style-image: url (ICONO_A_MOSTRAR);

```
list-style-type: disc; /*pelotita negra*/
list-style-type: circle; /*pelotita blanca*/
list-style-type: square; /*pelotita cuadrada*/
list-style-type: decimal; /* números */
list-style-type: decimal-leading-zero; /*0 adelante*/
list-style-type: lower-roman; /*núm. romanos minúscula*/
list-style-type: upper-roman; /*núm. romanos mayúscula*/
list-style-type: lower-greek; /*letras griegas minúsculas*/
list-style-type: lower-latin; /* letras a,b,c minúsculas */
list-style-type: upper-latin; /* letras a,b,c mayúsculas */
list-style-type: armenian; /* letras armenias */
list-style-type: none; /* sin viñetas */
```




**Esquema de colores**

- adobe color: http://color.adobe.com —> herramienta para crear paletas de colores armónicas. Partiendo de un determinado color base se pueden obtener colores afines del círculo cromático.

# Cajas

Como el  no produce una salida visual y acepta cualquier cosa adentro, se usa para delimitar las áreas del sitio web. Dentro de cada div irá todo el contenido que se va a mostrar en la misma área.

Todos los elementos del HTML son cajas: un elemento de línea se ve uno al lado del otro y los elementos de bloque, uno debajo del otro. Ese concepto de que “todo es una caja” da lugar al box model.

Sin importar si son de línea o de bloque, todas las etiquetas tienen las siguientes propiedades en común:

1. Width: espacio declarado para el ancho del contenido de la caja

2. Height: espacio declarado para el alto del contenido de la caja. El alto en la web no existe. Todo contenido que exceda el alto declarado se va a superponer con lo de abajo. Los elementos de línea no tienen ancho ni alto.
3. Border: tipo de línea que envolverá la caja
    - border-color: el color del borde
    - border-width: el grosor del borde. Número + unidad.
    - border-style: tipo de borde a aplicar. Hay varios: solid, dashed (rayado), dotted (punteado), double (doble línea), inset, outset, ridge, groover. También existe el border-top-color, border-left-width, border-bottom-style y sus combinaciones.
4. Padding: separación entre el borde y el contenido de la caja
5. Margin: separación entre el borde y el AFUERA de la caja. Se usa para separar un elemento del otro. El body tiene margen por defecto, por eso el contenido está unos píxeles hacia adentro. Se quita con margin:0. Todos los encabezados y párrafos tienen margin tanto hacia arriba como abajo.

**Color de fondo**

Todos los elementos aceptan color de fondo mediante la propiedad **background-color.**

**Imagen de fondo**

Todos los elementos del HTML aceptan tener, además del color, una imagen de fondo. Llegado el caso que la imagen sea transparente o no cubra el ancho/alto total del elemento se competará con el background-color.

background-image: url(fondo.jpg)

Por defecto cubrirá toda la superficie del objeto. El **background-repeat** indica si la imagen de fondo se repite o no; sus valores son:

- no-repeat
- repeat-x
- repeat-y
- repeat

El background-position sirve cuando la imagen no se repite o lo hace en un solo eje. Recibe siempre dos valores, eje x / eje y. Los valores para el eje x son: left, center, right. Para y: top, center, bottom. También acepta números en píxeles o porcentajes. El valor será aplicado desde la esquina superior-izquierda.

El **background-attachment** define qué pasa con la imagen de fondo si el contenedor tiene una barra de scroll. Acepta dos valores:

- background-attachment: fixed; por más que se haga scroll, la imagen queda fija
- background-attachment: scroll; por defecto. La imagen se va junto al scroll.

¿Cuándo se usa el background-image y no <img />?

- Logo de la página web
- Íconos de cualquier vínculo
- Los íconos de un texto
- Fondos del body basados en imágenes

Cuando la foto debe existir para que el texto valga la pena, va <img />

**Espacio (interno y externo)**

Las propiedades margin y padding aceptan 1, 2 o 4 valores (separados por espacios)

- 1 solo valor: aplica el mismo valor a los 4 lados

- 2 valores: el primero afecta al tob y bottom. El segundo al left y right.
- 4 valores: empezando desde el top es cada lado en sentido de las agujas del reloj (top, right, bottom, left).

También estas propiedades aceptan con un guión del medio el lado a afectar.

**La interfaz del sitio**

No es solo el diseño sino el todo con lo que el usuario se comunicará con la web. Hay tres reglas fundamentales:

1. Cerrar toda la web en un  que defina la resolución mínim de la pantalla.
2. Pensar qué sucede si el usuario tiene más resolución que ese  principal.
3. Diseño consistente entre las distintas secciones:
    1) La botonera y el logo del sitio siempre presentes.
    2) Ningún link debe apuntar a un documento pelado.

**Menú**

Cuando hacen una botonera, siempre destaquen visualmente la sección activa.

**Aclaraciones**

- Para que el sitio quede centrado el  principal debe tener margin left y margin right en auto.
- Para que el texto se vea más angosto le das al contenedor el width deseado
- Para que el objeto esté más abajo, le das margin-bottom o top.

# CSS columnas

**Alto explícito**

Cuando un elemento tiene un alto fijo, cualquier contenido que se exceda la caja será visible. Esto genera un problema si después viene otro contenido porque se superpone. Tenemos la propiedad **overflow** que maneja todo el contenido que sobresalga del border y tiene cuatro valores:

- visible: valor por defecto. El excedente es visible.
- hidden: el excedente no se muestra (lo corta).
- scroll: genera una barra de scroll en los dos ejes del elemento aunque no se necesite.
- auto: genera el scroll solo en el eje necesario.

**Hacer columnas**

Los elementos de bloque se muestran por defecto uno por debajo del otro. Si queremos mostrar dos elementos de bloques seguidos o generar columnas con DIVs, tenemos dos técnicas:

1. Convertir el elemento de bloque en uno de línea con display

Se encarga de definir cómo se ve un elemento HTML. Los dos comportamientos más importantes son:

- Convertir un elemento de bloque a uno de línea —> display: inline
- Convertir un elemento de línea a uno de bloque —> display: block

Los elementos de línea se muestran uno al lado del otro pero tiene un problema: no entienden el margin ni el padding de arriba y abajo. Si bien agrandan el elemento, este se superpone a lo que esté arriba y abajo del mismo. Para que entienda el padding y margen de arriba y abajo, se usa **display: inline-block.** Se trata de un elemento de línea (uno al

lado del otro) respetando todas las reglas físicas de uno de bloque. El tema es que si los elementos tienen un alto distinto, se van a alinear con el borde inferior. Esto se modifica con el vertical-align que puede ser top, bottom o middle.
Quitar un elemento: el display tiene también un valor para quitar un elemento del layout —> display: none;
Lo oculta y lo quita, es decir, no ocupa su lugar. Además, CSS tiene una propiedad llamada visibility que indica si un elemento se visualiza o no. Tiene dos valores: hidden o visible. Pero un elemento con visibility: hidden sigue ocupando lugar en el layout aunque no se vea.

2. Correr el objeto para la derecha o izquierda con float

La flotación es mover un elemento hacia la derecha o izquierda de su línea base y todo lo que viene después se acomodará en el hueco que queda vacío. Se usa la propiedad **float** que acepta dos valores:

- left: corre caja a la izquierda
- right: corre caja a la derecha

Cuando un elemento flota, deja de pertenecer al flujo normal de HTML.
Todo elemento flotado deja de empujar en alto a su contenedor. Si todos los elementos flotan, el contenedor colapsa su altura. Al elemento que se colapsa hay que darle un overflow con cualquier valor menos scroll.

**Diferencias entre display: inline-block y float**

- El display siempre asume que el próximo elemento esté a la derecha mientras que el float permite ponerlo a la izquierda.
- El display permite alinearlos verticalmente con vertical align mientras que float siempre acomoda los elementos respecto al borde superior del elemento más alto.

# Position

Es una propiedad pensada para ubicar un elemento en una coordenada puntual. Se suele usar para:

- Superponer elementos
- Crear barras de herramientas fijas
- Crear publicidades que te sigan con el scroll
- Hacer un menú con submenú adentro

**Cómo ubicar un elemento**

En primer lugar hay que definir qué tipo de posición se quiere usar. Luego hay que indicarle las coordenadas en las que se posiciona ese objeto. Esto se hace indicándole al objeto un valor numérico para las propiedades **top, bottom, left, right**. No se puede indicar dos propiedades opuestas a la vez.
Hay cinco valores posibles para el position:

1. Fixed: los bordes serán calculados en base al browser. Si el usuario hace scroll, el objeto sigue fijo.
2. Absolute: los bordes serán calculados en base al browser pero se irá con el scroll, salvo que se use en conjunto con la propiedad siguiente.
3. Relative: sirve para delimitar los limites de cualquier elemento hijo con la position absolute, así no usará el body como referencia.

4. Static: se usa para quitar cualquier position asignado.

**Propiedad z-index**
Se usa cuando dos elementos que tienen position se superponen. Acepta como valor un número sin unidad. A valor más alto, se mostrará por encima de los demás elementos. Por defecto, todos los objetos tienen z-index:1. Si dos objetos tienen el mismo valor de z-index y se superponen, el que fue creado después en el HTML se verá encima del otro.

**Menú con submenú**
El position se usa para hacer un menu que tenga submenú emergente. Los items del primer menú son relativos (sirven como borde de cualquier hijo). La lista de adentro de un list-item es absoluta. Por defecto, la sublista tiene display:none. Recién cuando un list-item detecta el :hover, si adentro tiene una lista, darle display:block.

**Position sticky (pegajoso)**
Es un valor alternativo del position que mezcla un elemento sin position con uno fixed. Mientras un objeto se encuentre en una ubicación con valor superior al indicado, se moverá junto con el scroll. Cuando el objeto esté por debajo de ese valor, por fuera de los bordes, quedará fijo como si estuviese fixed. Ideal para un menú con anclas si se tiene mucho scroll vertical.

**Cuando no se usa position**
No se usa position cuando podemos lograr lo mismo con float, display o margin.
¿Queres que todo este centrado horizontalmente con respecto al ancho de pantalla? Ancho fijo y margen: auto;
¿Queres hacer columnas? Display: inline-block o float: left
¿Queres una botonera con cada botón al lado del otro? Los botones tienen que tener display: inline-block

# Sintaxis y cajas

**Qué es el HTML5**
HTML5 es un mix de tecnologías basadas en:
- Nuevas etiquetas de HTML
- Más funcionalidades en Javascript
- Propiedades CSS más enriquecidas

**Elementos semánticos**
HTML5 evolucionó en nuevos elementos más semánticos la etiqueta - Son cajas que permiten contenedor otros elementos pero sectorizan la web.
- <nav>

Representa el área de la navegación primaria del sitio. No reemplaza la lista con vínculos adentro, sino que la envuelve.
- <header>

Delimita un encabezado. Puede ser un título (de una página, de un texto, de un área de la web). También puede contener elementos de navegación, como el nav. Y puede haber más de uno por documento ya que no se limita al encabezado de la página.

- <footer>

Representa el área donde se encuentran los contenidos propios de un pie (de página, de artículo, de lo que sea).
Generalmente tiene datos de lo que estemos viendo: si es una web, puede tener copyright y accesos a redes sociales o mapa del sitio; si es el caso de un texto de un artículo, el autor, fechas, vistas.

- <main>

Define el contenido principal del documento que se está visualizando. Debe ser único al documento y no debe encerrar ningún contenido que se repita a lo largo de los demás archivos. Solo puede haber uno por documento.

**Soporte de etiquetas**

Internet explorer tiene algunos issues dependiendo la versión. Por ejemplo, IE11 sabe que existe el main pero no que es un elemento de bloque por lo que se indica con CSS. Y hasta la versión 10 de Iexplorer no hay soporte para header, footer ni nada de eso. Por eso hay que cargar en el head un javascript llamado html5shiv.js que los habilita y hacerlos de bloque con CSS.

**<article>**

Representa un contenido que puede ser leído independientemente de la web. Un archivo HTML puede estar formado por muchos article, por ejemplo cada una de las entradas de un blog es un artículo. Un artículo puede tener su propio header, footer y también su propio h1. Para detectar un article se debe separar su contenido y confirmar si aún tiene sentido.

**<section>**

Contiene cada grupo de contenidos que representan un área del documento. En general reemplaza la mayoría de los div estructurales. Un section puede estar formado por un conjunto de article (las últimas publicaciones de un blog).

**<aside>**

Todo el contenido extra que no aporta nada al contenido principal de la web, es decir, que sin su presencia la web seguiría funcionando bien. Por ejemplo, las barras al costado de toda la web con el buscador, login, sitios recomendados, etc. En este caso, el aside estaría por fuera del main.
Pero también un artículo puede tener un aside, con los comentarios de un posteo o artículos relacionados. Entonces estaría dentro del main.

**Título del section**

Según la W3C, cada section debe tener un encabezado anunciando su sitio. Le podemos dar display:none; para no visualizarlo y que no nos rompa el diseño si es que no lo queremos.

**Imágenes con descripción**

En HTML se puede asignar un título visible o descripción a una <img/> insertando un <p> junto a la foto. Sin embargo, desde lo semántico, no existe relación entre párrafo e imagen.

HTML5 ofrece el elemento <figcaption> que es la descripción visible de la etiqueta <figure>. El <figure> será el contenedor de la imagen y su respectivo *caption*.

**Details y summary**
La etiqueta <details> permite mostrar, expandir o colapsar un bloque de contenidos extra. Es ese texto que no a todos los usuarios puede interesarle y solo te consume espacio. Al hacer click se abre/cierra de contenido. Acepta un hijo <summary> que contiene un título o resumen que se verá aún si está colapsado. Acepta un atributo open (sin valor) que muestra el bloque de contenido expandido por defecto.
Tanto details como summary no son soportadas por Internet Explorer ni Edge. Para hacerlo compatible hace falta un SHIM desarrollado por Tyler Uebele. Está formado por un Javascript (para habilitar la funcionalidad) y un CSS para emular el aspecto. Ojo, no funciona si no tiene el summary adentro.

**Listas numeradas**
Hay dos atributos funcionales de las listas que son útiles que se ponen en el ol o ul:

- start=”numero”: la lista iniciará la cuenta en ese número.
- reversed: este atributo no requiere valor. Hace que la numeración de los items sea descendente.

# Multimedia

Son todos los recursos externos al documento HTML que enriquecen la experiencia audiovisual. Nosotros ya trabajamos con un grupo: las imágenes. También pueden ser archivos flash, sonido, video, mapas.

**Flash**
Formato propietario de adobe. Hay un archivo editable (.FLA) y eso se exporta a un archivo que usa en la web (.SWF). Genera animaciones 2D basado en una línea de tiempo.
Incursionó en Web como aplicación para crear banners animados de alto impacto visual. Un banner es un anuncio que se suele insertar en una página web promocionando otro sitio o un producto.
En 2000 se hizo común ver sitios hechos en Flash. Estas webs son más dinámicas y cada click dispara toneladas de efectitos.
Flash simplificó mucho la inserción de sonido y video. El elemento multimedial se convierte a un formato propio de adobe. Luego se inserta en la web usando un archivo de flash que pueda entender y reproducir el FLV. De esta manera, para verlo, el browser solo necesita el plugin de Flash que muestre el SWF.
HTML5 y CSS3 creció lo suficiente para no depender de flash a la hora de efectos o videos.

**<embed />**
Significa embeber, es decir, incrustar un archivo. Tiene los mismos atributos que una imagen (width, height y src).

```
<embed width=”1000! height=”600” src=”miarchivo.swf” />
```

**<iframe> <frame>**
Significa inner frame (marco interno). Es un contenedor como el div que permite cargar archivos externos que también posee width, height y src. Es como insertar un minibrowser adentro del HTML. Por ende sirve para cargar cualquier tipo que el navegador pueda entender.

<iframe width=”800” height=”700” src=”http://davinci.edu.ar”> </iframe>

**<object> </object>**
Ua el atributo data para indicar la ruta al archivo y el atributo type para indicarle al navegador qué tipo de elemento se está insertando. Este atributo es optativo. Para los browsers viejos, lleva adentro un <param /> que indica nuevamente la ruta al archivo.

<object width=”” height=”” data=””>
<param name=”movie” value=”nombrearchivo.mp4” />
</object>

El param que carga el archivo **siempre** es movie, ya sea para cargar flash, sonido o video.

**Codecs y compatibilidades**
A la hora de insertar música o video, intervienen los CODECs (que significa codificador-decodificador). Traduce un archivo de tipo audio/video a una señal de stream (reproducción en tiempo real). Un archivo de audio solo tiene un codec (el audio) y un video tiene un codec para el audio y otro para la imagen.
No todos los navegadores lo soportan.

**Etiquetas nativas**
HTML5 agregó etiquetas nativas para insertar sonido y video: <audio> <video>. A diferencia de object y embed, usan un player nativo propio del browser en vez de plugins externos. Por ende, lógicamente, no se puede tunear.
**audio** y **video** llevan adentro la etiqueta <source />. Por cada source se debe indicar el atributo src con la ruta al archivo que se quiere cargar y, optativo, el type.

<video width=”” height=””>
<source src=”peli.mp4” type=”video/mp4”/>
</video>

Se pueden insertar varios source, de distintos formatos. El primero que entienda el browser será reproducido. Si tenés un solo archivo, al igual que las imágenes, también podés no usar los source y darle el atributo src.

**Otros atributos**
Audio y video tienen una serie de atributos funcionales:

- controls: no lleva valor e indica que se debe mostrar la barra de control (play, volumen)
- autoplay: no lleva valor, especifica si el archivo debe reproducirse automáticamente
- loop: no lleva valor, especifica si al finalizar la reproducción debe repetirse
- poster, que indica una foto que se verá mientras se reproduzca el video —> le tengo que indicar en dónde está guardada la imagen con el atributo src.

**Sonido y video en tu web**
Siempre que insertemos sonido o video en web, que no arranque automáticamente. La música de fondo es una mala idea. Siempre que el navegador haga algo que el usuario no está esperando, lo estás invadiendo.

# Recursos de terceros

Un video puede ser pesado para web. Y no todos los navegadores entienden cualquier formato. Usando <video> y <audio> podemos indicar varios archivos distintos para cargar el que entienda. Pero el ancho de banda sale de nuestra web y seguimos sin garantías de ver el archivo. Usar un servicio de tercero es la respuesta.
Es simplemente abrir una cuenta en una red social para subir audio o video. Ese archivo se insertará en nuestra página web mediante el código para compartir el archivo. Ofrece dos ventajas:

- El ancho de banda para ver el archivo es de la red social.
- La promoción viral de estar en esa red social.

**Youtube**
Ofrece como etiqueta un iframe que pegaremos donde queramos que aparezca el video. Tenemos que tener cuidado con los atributos que nos da el vínculo, como por ejemplo frameborder que al validar nos da error.

**Vimeo**
Tiene oferta más limitada pero con menos traumas con el copyright. Ofrece un iframe para insertar el código. Podemos sacar el párrafo.

**DailyMotion**
Ofrece un iframe para incrustar el video.

**Soundcloud**
Es para audios. De la ventana para compartir, elegir la segunda pestaña llamada EMBED y usar el iframe. Antes eliminar todos los atributos inválidos.

**Mapita de google**
El mapa dinámico permite hacer zoom, moverse y hasta mostrarlo como streetview. Se puede usar:

- Poniendo una dirección o nombre del lugar
- Por medio de las coordenadas geográficas (latitud/longitud)

El mapa que se puede interactuar está ubicado en https://www.google.com.ar/maps
En la caja de texto se pone el lugar, dirección o coordenada y te muestra los resultados desde la URL: https://www.google.com.ar/maps/place/
Desde las opciones del menú lateral se accede a compartir y de las dos solapas vamos a incorporar mapa y al igual que youtube, nos da un iframe.

# Formularios

**HTML** no es un lenguaje de programación sino de estructura. Si se necesita recopilar información ingresada por un usuario, HTML ofrece controles de formulario. Son etiquetas donde el usuario ingresará o seleccionará valores y serán enviados a un archivo encargado de procesar la información.
Siempre que el usuario deba ingresar un dato, que no sea con Javascript, se usa un formulario.

**<form> </form>**
Para insertar un formulario se usa la etiqueta esta, que adentro lleva todos los controles que vayan al mismo destino. Requiere 3 atributos para funcionar:

- action: documento que se encarga de recibir datos y procesarlos. Debe ser un lenguaje de los llamado “del lado del servidor”: PHP, ASP, JSP. Si no se indica un valor, por defecto el action es el mismo archivo donde está el formulario. Como HTML no es un lenguaje de programación, perdiste toda la información si no definís el action.
- method: la forma en que será enviada la información. Existen dos métodos: GET y POST.
    - ➔ GET: la información viajará por la barra de direcciones a continuación del nombre del archivo. Si no se indica el method, aparece esta por defecto.
    - ➔ POST: la información viajará junto a los encabezados del HTML (será “invisible”).
      El problema con GET es que está limitado a 2048 caracteres, la información es visible por lo cual no se debe usar el traspaso de datos sensibles y no sirve para adjuntar archivos. Lo que tiene POST es que no se puede guardar los datos pasados en favoritos y es más lento por un millonésima de segundos.
- enctype: cómo se codificaran los contenidos del formulario a enviarse. Hay varias, de las cuales las dos más comunes son:
    - ➔ application/x-www-form-urlencoded: eñes, tildes, y caracteres raros se codificarán como las URLs. Es el valor por defecto.
    - ➔ multipart/form-data: es el que se debe usar para adjuntar archivos o uploads.

Como no programamos con PHP, el action de nuestros formularios puede ser directamente el archivo que dice “gracias”.

**Controles de formulario**
Son los campos que completará el usuario. Se dividen en tres grupos:

- Elementos de ingreso libre de texto
    1. Cajas de texto de una sola línea: no acepta el uso de la tecla enter
       <input type=”text” />
    2. Cajas para el ingreso de contraseñas: el contenido no será visible
       <input type=”password” />
    3. Cajas para contenido multilínea: puede ser una o muchas líneas de texto.
       <textarea rows=”4” cols=”15”> </textarea>
       Los atributos rows y cols para HTML5 no son obligatorios.

El texto que acompaña los elementos de entrada se formatean a mano, ya sea con un <p> o 

- Elementos de selección de opciones predefinidas.
- Botones de acción:
  - ➔ Input de tipo submit, envía el formulario
  - ➔ Input de tipo reset, resetea el formulario
  - ➔ Input de tipo button, no tiene acciones por defecto. Usa el atributo onclick que que lleva código JS al clickearlo.
  - ➔ Input de tipo imagen envía el formulario, pero en lugar de mostrar un botón permite usar una imagen con atributos src y alt.

Todos los controles de formulario deben tener un nombre que se indica con el atributo name. Ese es el nombre con el cual después se accederá a su valor en el archivo indicado en el action. Un control sin name no se envía. El name no se puede repetir.

**Validación del formulario**

El **form** es un elemento de bloque y acepta cualquier propiedad CSS que acepte un div, o p. Se pueden usar div para encerrar los elementos por cuestiones de CSS. Puede ser un div por cada control, o un div que cierre todo, depende de lo que queramos hacer.

**Atributo VALUE**

Es el valor por defecto que tiene un control. Hace algo distinto según la etiqueta que lo tenga:

- En los input de tipo text es lo que aparece al cargarse el formulario o apretar el botón reset
- En los botones, es el texto que aparece dentro del botón
- El textarea no tiene value, su valor por defecto se escribe entre la apertura y el cierre de la etiqueta
- En el input de tipo password, no tiene sentido indicar un value porque el contenido no se ve.

**CSS sólo de formularios**

Hay dos propiedades CSS solo de formulario:

- Cuando se hace foco en un control input/textarea algunos navegadores le ponen un borde alrededor. Este se quita con input, textarea {outline: none;}
- El textarea se puede redimensionar arrastrando las flechitas de abajo a la derecha. Esto se quita con textarea {resize: none;}

**Encolumnar los controles**

Si queremos que los controles y su texto aparezcan en columnas, encerramos los textos en una etiqueta cualquiera y la hacemos inline-block y le damos a todos el mismo ancho.

**Nuevos inputs**

HTML5 agregó una serie de inputs que ofrecen componentes enriquecidos que facilitan la solicitud de ciertos valores complejos. Otros vienen pre validados, y solo esperan un determinado valor y de no cumplirse frenan el envío del form. Adicionalmente le notifican al usuario del error cometido.

- Input de tipo color: muestra un picker para elegir un color.
- Input de tipo range: muestra una barra horizontal de tipo slider para que el usuario seleccione dentro de determinado rango.
- Input de tipo date.
- Input de tipo time: ingresar una hora.
- Input de tipo datetime-local
- Input de tipo week: para seleccionar toda una semana
- Input de tipo month: para seleccionar todo un mes

**Controles pre-validados**

Estos controles no permiten ingresar cualquier valor. Frenan el envío del formulario si el contenido no corresponde a lo esperado.

- Input de tipo mail
- Input de tipo url
- Input de tipo number
- Input de tipo tel

**Atributos fundamentales**

- required: hace obligatorio, frena el envío del form si está vacío
- placeholder: mostrará su valor dentro del input hasta que el usuario tipee algo
- Para que un campo no se autocomplete con valores anteriores, darle el atributo autocomplete=”off”

**Atributos funcionales**

En los input derivados del date, range y number se pueden establecer los valores mínimos y máximos a aceptar con min=”” y max=”” respectivamente.

En los tipos number y range, se puede indicar el valor de incremento/decremento con el atributo step=””

El input number solo aceptará valores decimales si tiene un step decimal

Un campo puede tomar el foco al cargar la página con el atributo autofocus

**Atributos adicionales**

Todos los controles tienen tres atributos comunes:

- readonly=”readonly”: solo lectura. No se puede cambiar su valor, pero puede ser copiado. Aunque no se pueda modificar, se manda al documento de destino (action)
- disabled=”disabled”: control deshabilitado. No se puede copiar ni cambiar su valor. Tampoco es enviado al documento de destino.
- tabindex=”4”: orden de tabulación. Cuando el usuario navega el formulario con la tecla TAB, se recorrerá del número más bajo al más alto.

**Conjunto de campos**

Cuando un formulario está formado por varios items, se pueden agrupar los campos en grupos. Para eso existe el <fieldset> </fieldset>. Genera una caja con un borde conteniendo los elementos que queden adentro.

Adentro se usa la etiqueta <legend> </legend> para crear una pestaña que se verá como título.

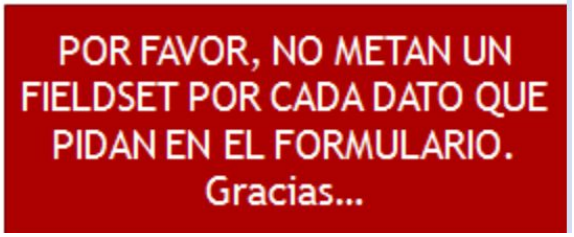
El fieldset solo se usa para agrupar visualmente campos dentro de un formulario que tienen objetivo en común.

```
<form method="post" action="recibir.php">

        <fieldset>
                <legend>Datos personales</legend>
                <!-- campos -->
        </fieldset>

        <fieldset>
                <legend>Estudios previos</legend>
                <!-- campos -->
        </fieldset>

</form>
```




**Adjuntar archivos**

La etiqueta input tiene el tipo file para insertar un control de “examinar archivos”. Desde CSS es un elemento intuneable porque cada navegador lo muestra como quiere. Tiene un atributo, que define el mime-type que acepta el input: accept=”image/jpeg”. El formulario obligatoriamente debe estar codificado como multipart/form-data y ser enviado por el método post, sino no se envía el archivo.

**Controles de selección**

El usuario no puede ingresar libremente un texto, sino que el programador le da una lista predefinida. En todos los casos, el dato que nos llega al elegir una opción se define desde el atributo value.

Existen tres grupos de controles de selección:

- Botones de radio: solo se puede elegir una opción. Son las opciones basadas en “pelotitas”. Se usa para datos de estado civil, genero, etc. Es la etiqueta input con el tipo radio. No tiene ningún atributo para definir su texto, se escribe a mano. Es la ÚNICA etiqueta que debe tener el mismo name para todas las opciones del mismo conjunto.
  <form method=”post” action=”recibir.php”>
   Seleccione su sexo 
   <input type=”radio” name=”sexo” value=”m” /> Masculino 
   <input type=”radio” name=”sexo” value=”f” /> Femenino 
  </form>
- Casillas de chequeo: de toda una lista de opciones, el usuario puede elegir una, todas o ninguna opción. Son las opciones basadas en “cajitas”. De toda la lista el usuario puede elegir todas, una o ninguna. También es un input, con el tipo checkbox. No tiene ningún atributo para definir su texto tampoco. NEcesita un name

distinto para cada opción. HTML no posee ninguna forma de indicarle una cantidad mínima a seleccionar, eso se hace con JS.

La etiqueta label se usa para estirar la zona clickeable de un radio o checkbox, y es un elemento de línea. <label> </label>. Hay dos formas para usarlo:

1. Encerrar el input y el texto en una etiqueta label.
2. Encerrar solo el texto en la etiqueta label y ponerle atributo for, que debe tener el mismo valor que el id del input a vincular.

- Menu desplegable: solo se puede elegir una opción. Es el llamado combobox, selector o menú. De toda la lista se puede elegir una opción. Para insertar el menú se usa la etiqueta select, que abre y cierra. Adentro va una etiqueta option, que abre y cierra, por cada opción a mostrar. Adentro del option va el texto a mostrar.

El select lleva el atributo name, los option no.
Los option llevan el value, el select no.

**Manipular el select**

Existe una etiqueta llamada optgroup que va dentro del select y agrupa visualmente los option. Con el atributo label en optgroup se le pone título al grupo. El select permitirá elegir varias opciones con CTRL presionado si tiene multiple=”multiple”. El select se puede mostrar como una lista desplegada, con el atributo size mayor a 1.

**Opciones preseleccionadas**

Para marcar una opción como seleccionada al cargar el form existen dos atributos distintos (según la etiqueta).

- checked: es solo para los input de tipo checkbox y radio. En los radio solo uno puede ser este atributo, en los checkbox más.
- selected: solo para los option del select. Si el select es múltiple más de una opción puede tenerlo.

**Escribir con ayuda**

Existe la posibilidad de que el usuario escriba libremente pero si pone una opción predefinida que le aparezca como sugerencia usando la etiqueta **datalist**. No genera ninguna salida visual y le permite que le indiquemos un listado de opciones. Cada opción será un option dentro del datalist. El datalist se vinculará a cualquier input que tenga el atributo list con el mismo ID del datalist.

```
<input type="text" list="lista" name="profesor" />
<datalist id="lista">
  <option>Mabel García</option>
   <option>Germán Rodríguez</option>
</datalist>
```

# Vendor prefix

Las empresas que desarrollan browser también generan propiedades que son útiles en la web, no solo el organismo que valida CSS. Muchas no son reconocidas por el validator y cuando se las reconoce algunas no se comportan como lo pensó el creador en un principio. Para eso existen los **vendors prefix.**
Si el browser quiere testear una propiedad experimental, le antepone un prefijo que solo lo interprete su motor. Esto significa que otro navegador puede no entenderlo o puede aplicarlo de otra manera con su propio prefijo.
Los **vendors prefix** son los prefijos que empiezan con un guión del medio antes de la propiedad de CSS. Hay uno distinto para cada navegador:

- Chrome: -webkit-PROPIEDAD: valor;
- Safari -webkit-PROPIEDAD: valor;
- Firefox -moz-PROPIEDAD: valor;
- Opera -o-PROPIEDAD: valor;
- IExplorer -ms-PROPIEDAD: valor;

**Propiedades**
¿Cómo saber qué propiedades llevan prefijo? Tenemos que pasar el código por algún sistema que nos indique si esa propiedad lleva o no prefijo.

1. Con recursos de terceros: http://pleeease.io/play/ es un formulario para tiperar tu código original y en tiempo real te muestra la versión prefijada. Lo más importante es cuántas versiones hacia atrás del browser debe darle soporte. Hay que darle entre 10 y 20 versiones.
2. Hacerlo con JS: http://leaverou.github.io/prefixfree/ es un script que busca toda declaración CSS que lleve prefijo y se lo agrega. Debe insertarse luego de llamar a la hoja de estilos.

**PrefixFree.min.js**
Esto es un script de Javascript. Al insertarlo en la web, te scanea todo el CSS y busca las propiedades que requieren prefijo y solo agrega si el usuario la necesita. En Chrome solo funciona cuando está online. El archivo vos te lo bajas hoy pero CSS sigue avanzando, por lo cual deberías actualizar el JS cada un tiempo.

# Un poco sobre tipografías

Cuando redactamos un texto extenso hay que respetar dos reglas:

- Mantener un margen entre los párrafos
- Usar columnas angostas de texto, hasta de 400px de ancho en textos extensos

**Texto en columnas**
Existe una propiedad de CSS para hacer columnas de texto angosto llamada **column-count** seguida por el número de columnas sin ninguna unidad. Acepta también un **column-rule** que hace de borde entre columna y columna, misma sintaxis que un border.

También tiene un **column-gap** para definir el espacio entre columnas (acá sí, con su unidad). Algunos browsers requieren prefijo propietario.
**column-span** nos permite trabajar con los encabezados que tengamos en el contenedor de las columnas. Por defecto si no establecemos esta propiedad el título se mantiene dentro de la primera columna. Acepta dos valores: none y all. Si lo establecemos con all, se extiende sobre todas las columnas para cuando el texto dividido corresponde al mismo encabezado.

# Clases

**Múltiples clases**
Ninguna etiqueta acepta dos veces el mismo atributo:

- Una imagen no puede tener dos alt
- Ninguna etiqueta puede tener dos ID
- Ninguna etiqueta puede tener dos class

Pero todas aceptan varios valores para el class, dentro de la misma comilla y separados por espacio:

- Si son distintas propiedades, se suman.
- Si son la misma propiedad, se define por precedencia.
- Si son igual de importantes, gana la última escrita en el CSS.

Si algún elemento solo debe aplicar formato cuando coincida que tiene ciertas clases a la vez, también se puede hacer una regla específica. En ese caso, el CSS deberá tener una regla que indique un class seguido de otro con su respectivo punto al principio sin espacios entre cada uno.

```
.rojo{ color: red } /* solo al class="rojo" */
.azul{ color: blue } /* solo al class="azul" */
.rojo.azul{ color: violet }  /* aplica al que tenga class="rojo azul"
o class="azul rojo" */
```

# Selectores específicos

Así como se puede formatear un elemento por medio de la cascada CSS, acotando la lista de etiquetas a afectar, se puede hacer todavía más específica la búsqueda basándola en cómo está organizado el árbol de elementos del HTML. CSS nos ofrece dos conjuntos de selectores que comienzan con dos puntos:

- pseudo-clases: son reglas que permiten formatear un elemento del HTML según su estado actual. El más claro ejemplo son los vínculos que tienen el estado :hover cuando se le pasa el mouse por encima. La sintaxis es objeto :pseudoclase{/*CSS*/}
- pseudo-elementos: son reglas que permiten formatear una parte específica de una etiqueta HTML. Por ejemplo, una oración formatear solo la primera letra o la primera línea. O en el caso de elementos, se puede aplicar formato antes o después del elemento. La sintaxis es objeto ::pseudoElemento{}

**Selector de nodos**

Busco elementos en base a otro usado como referencia.

- elementoA > elementoB: elemento B debe ser hijo directo de elementoA (no pudiendo existir etiquetas intermedias)
- elementoA + elementoB: elemento B debe ser hermado adyacente(pegado) de elementoA. Si después de elemento A vienen varios elementoB, solo aplica al primero
- elementoA ~ elementoB: elementoB debe ser hermano de elementoA pero no necesariamente adyacente

En todos los ejemplos se aplica el formato a elementoB

**Selector por atributos**

Aplica CSS a etiquetas que tengan los atributos buscados

- elemento [ atributo ] : no importa el valor, la etiqueta debe tener ese atributo
- elemento [ atributo = valor ] : la etiqueta debe tener exactamente ese valor
- elemento [atributo^= valor ] : el atributo comienza con ese valor (sin importar el final)
- elemento [atributo $ = valor ]: el atributo termina con ese valor (sin importar el inicio)
- elemento [ atributo *= valor]: el atributo contiene ese valor (inicio, medio, fin, da igual)

**Selector de hijos**

En todos casos A representa un contenedor. Ojo, este selector no busca la aparición número X de elemento adentro de A. Primero cuenta todos los hijos de A (cuántas etiquetas hay). Después se fija si el número X pedido es del tipo elemento. Y solo si coincide, le da formato.

- A elemento :first-child —> primer elemento adentro de A
- A elemento :last-child —> último elemento dentro de A
- A elemento :nth-child(X) —> elemento número x adentro de A
- A elemento :nth-child(odd) —> elementos impares adentro de A
- A elemento :nth-child(even) —> pares

**Selector de ocurrencias**

- elemento :first-of-type

Formatea al primer elemento de todos los de su tipo

- elemento :last-of-type

Formatea al último elemento de todos los de su tipo

- elemento :nth-of-type(numero)

Formatea al elemento si es la número vez que aparece

- elemento :empty

Formatea al elemento si no tiene nada adentro <elemento> </elemento>

- elemento :target

Formatea al elemento si la URL apunta a su ID (un ancla) página,html#ID

**Selectores de contenido**

- elemento::before

Aplica formato CSS justo antes de que inicie la etiqueta

- elemento::after

Aplica formato CSS justo después de cerrar la etiqueta.

- elemento::selection

Define el formato que tendrá el texto al ser seleccionado por el mouse, una buena práctica sería cambiar el fondo. Firefox no lo soporta.

**Selectores de texto + yapa**

- elemento::first-letter

aplica formato a la primera letra de ese elemento

- elemento::first-line

aplica formato CSS a la primera línea de texto, hasta que aparezca un <br /> o llegue al ancho de la etiqueta y baje

- elemento:not(selector)

aplica formato si el elemento no coincide con el selector del paréntesis (puede ser ID, class, etiqueta.