COMPENDIO DE TEMAS Y TIPS HTML Y CSS

- **Inserción de texto**: texto "tirado" directamente en el body es inválido. Siempre un texto es semántico: un título, un párrafo, un refuerzo semántico, una lista, etc. El párrafo es una idea dentro de un contexto, en textos extensos no debe incluirse un único párrafo sino varios (cada punto y aparte o finalización de una idea es un cierre de un párrafo y la apertura del siguiente). Anidar un párrafo dentro de otro es inválido. Pueden contener dentro refuerzos semánticos, vínculos, etc. pero nunca un encabezado (ni tampoco puede haber un párrafo dentro de un título).
- **Saltos de línea**: el enter en el código html no implica un salto de línea en la página ya que los espacios en blanco generados por esa tecla, el tab y la barra espaciadora por demás (más de un espacio lógico consecutivo) son eliminados por el navegador. El salto de línea se produce con la etiqueta br. Esta etiqueta solo puede usarse dentro de etiquetas contenedoras de texto (párrafos, títulos, etc.) NUNCA por fuera de ellas para separar elementos. Tampoco deben incluirse más de un br consecutivos, aunque estén dentro de un párrafo, por ejemplo. Si misión es producir un salto de línea interno en el contenedor de texto en el que se incluye.
- **Jerarquía de encabezados**: no elegir una h (h1 al h6) por el preformato que le aplica el navegador, si algo es subtítulo de un título previo es un nivel inferior (por ejemplo: subtítulo h3 de título h2 previo). NO se puede incluir una h5 si no es subtítulo de una h4 previa. El copyright (en el footer) no es título, el texto de los botones no son títulos.
- **Refuerzos semánticos**: se incluyen para destacar la parte más importante de un texto, no para poner en negrita o en itálica una parte de este. Esto implica que su uso es por cuestiones lógicas (destacar lo importante) y no físicas (por el formato que le aplica el browser). No se pueden anidar entre ellos, ni pueden ser hijos de un encabezado (el título es más jerárquico que el refuerzo semántico, no tiene sentido reforzar parte de un título).
- **Imágenes**: sin atributo alt son inválidas, es obligatoria la inclusión de este atributo. No redimensionar imágenes con atributos width y height, estos atributos sirven para reservar el espacio para la imagen al cargar la página, sus valores son en píxel, pero SIN unidad de medida (si se incluye se invalida la etiqueta). Las imágenes deben ser optimizadas en Photoshop en peso y tamaño para web (72dpi).
- **Vínculos absolutos**: se deben incluir todas las partes de la url, esto es, protocolo, dominio y ruta, por ejemplo, **http://davinci.edu.ar/**. Se usan para vínculos a páginas externas del sitio.
- **Vínculos relativos**: Se incluye solo la ruta (sin protocolo ni dominio ya que se usan para vínculos internos del sitio donde ambos valores son constantes). Se calcula la ruta desde el html donde estamos incluyendo la etiqueta a. Si es necesario entrar en una carpeta para encontrar el html a vincular, se nombra la carpeta y luego se incluye una barra para "entrar" (esto se repite por cada carpeta en la que sea

necesario entrar), por ejemplo: **secciones/productos.html**
Si es necesario "salir" de una carpeta para encontrar el html a vincular, NUNCA se la nombra, para salir se incluyen dos puntos seguidos y una barra: ../ (esto se repite por cada carpeta desde la que se deba salir), por ejemplo: **../index.html**
Si dos recursos o archivos están en la misma carpeta, simplemente alcanza con escribir el nombre y la extensión del recurso a linkear, por ejemplo: **contacto.html**
Más allá de los vínculos este tipo de ruteo se utiliza para linkear, vincular, abrir o descargar cualquier recurso del sitio (por ejemplo, para incrustar imágenes con la etiqueta img en su atributo src).

- **Vínculos ancla**: Se utilizan para navegar internamente un documento html para evitar hacer scroll hasta la parte del contenido que se quiere visualizar. Para ello se incluye el atributo id en la etiqueta a la que quiero llegar en el contenido con un valor ÚNICO por página (el ancla) y luego se incluye un vínculo que recargue la página y apunte al valor de ese id. Por ejemplo, si en una h2 se agrega el **id="manzana"** (este es el ancla), desde un vínculo (etiqueta a) se navega a ella incluyendo el valor al **href="#manzana"**. Al hacer click en este vínculo la h2 "se pega" en el top de la ventana del browser sin necesidad de hacer scroll.
- **Recomendaciones generales**: los nombres de los archivos (html, imágenes, carpetas, etc.) como de los valores de los atributos html o cualquier recurso que se utilice en web, deben realizarse en minúsculas, sin acentos, eñes ni caracteres especiales (salvo el guion bajo _). El sitio DEBE mantener una coherencia estética y funcional, esto implica que el diseño de sus distintas páginas deben estar en sistema (no dar la impresión que al navegar a una sección estoy en otro sitio diferente), ciertos elementos constantes (logo, barra de navegación, footer, etc.) deben presentarse en el mismo lugar en todas las secciones y la barra de navegación debe presentar SIEMPRE todos los botones y EN EL MISMO ORDEN, incluso el botón que hace referencia al html que se está visualizando (por ejemplo, si estoy en productos.html, el botón "productos" debe estar presente en el mismo lugar que en cualquier otra sección y si se le hace click "recargar" la página con el #).
  Navegación jerárquica no es para web (esto es si, por ejemplo, desde el index navego a contacto y tengo un botón "volver" al index para poder volver a navegar a otra sección en vez de presentar la barra de navegación desde donde el usuario pueda navegar desde cualquier documento a cualquier otro).
- **Naturaleza de los elementos**: Las etiquetas se pueden dividir en dos grupos: **en línea**: ocupan solo el ancho del contenido, se ubican en línea con el texto u otras etiquetas en línea, es decir, una al lado de la otra, el alto corresponde al alto de la línea y tamaño de fuente y pueden contener texto u otra etiqueta en línea, meter una de bloque como contenida de una en línea no valida; **de bloque**: ocupan el 100% del ancho de su contenedor (si es el body es el ancho de la ventana del browser), se ubican uno debajo del otro ya que al ocupar todo el ancho disponible no dejan lugar para que ningún otro elemento se ponga a su lado, el alto se ajusta a la cantidad de contenido que tengan (si tienen mucho texto se generan la cantidad de líneas de texto necesarias para que lo contenga completamente y su

alto se ajusta automáticamente), pueden contener texto, etiquetas en línea y etiquetas de bloque.

- **Flujo normal o natural de la página**: Es la distribución de los elementos de la página, la relación entre ellos y el lugar o espacio que ocupan, lo que depende de la naturaleza de las etiquetas, por lo tanto, las etiquetas en línea se ubicarán una al lado de la otra y las de bloque una debajo de la otra en el orden en el que se escribió el código html (lo que escribo primero se dibuja primero, lo que escribo después se dibuja después) pero sin importar si se escribe todo el código en una sola línea o en varias (los párrafos que son de bloque se ubicarán siempre uno debajo del otro sin importar si se escribieron uno al lado del otro en el código). Todos los elementos se ubicarán en el lugar que le corresponde lo más arriba y a la izquierda posible.
- **Listas**: tanto la lista ordenada (ol) como la desordenada (ul) solo pueden tener list-items (li) como hijos, ninguna otra etiqueta y esos ítems (li) solo pueden ser hijos de una lista (no pueden estar, por ejemplo, dentro de un párrafo). La barra de navegación DEBE tener una estructura de lista desordenada donde cada botón es un li que contiene al vínculo que navega al documento a linkear. Por ejemplo: productos (se incluye un único botón como muestra pero, por supuesto, que la barra de navegación deberá presentar un botón por cada sección que tenga el sitio y a la que se deba navegar).
- **Etiquetas no semánticas**: tenemos una de bloque: **div** y otra en línea: **span**. No agregan valor semántico ni significado a su contenido, sirven para maquetar, contener elementos, contener texto para informar que están escritos en otro idioma (agregando el atributo lang), aplicar formato desde CSS, etc. En esta materia utilizaremos un **div contenedor** de todos los elementos que tenga cada página (abre luego de la apertura del body y cierra antes del cierre de este).
- **Estructura semántica**: Dan valor semántico a su contenido y se utilizan para estructurar los distintos elementos o partes principales de la página.
    - **header:** cabezal de la página o de una sección o artículo, se puede repetir, su función principal es contener el título de lo que encabeza (si es el principal, la h1), sin título no tiene razón de ser. Además, puede contener la barra de navegación (si es el principal).
    - **main**: destaca el contenido principal de la página. Puede encerrar todo ese contenido o solo una parte que se quiera jerarquizar como lo más importante, incluso puede no incluirse (no es obligatorio). Puede haber uno solo por documento. No puede ser hijo de: header, nav, article, aside ni footer. No debe contener elementos repetitivos del sitio como el header principal, el footer, logo, barras laterales, etc. Generalmente no lleva contenido directamente (títulos, párrafos, imágenes, etc.) sino secciones o artículos principales del contenido a los que destaca.
    - **footer**: pie de contenido o general de la página. Puede repetirse. Si es el principal generalmente contiene el copyright, datos legales, formulario de contacto, un duplicado de la barra de navegación, etc.

- **nav**: contenedor de la barra de navegación principal del sitio (no cualquier vínculo, ni tampoco la barra de redes sociales). Puede repetirse.
- **article**: contenido independiente del contexto de la página, la parte donde se desarrolla el contenido y la más indexable, puede tener su propia estructura interna, incluso su propia h1 (en casos muy específicos, no siempre), en cualquier caso, siempre debe contener un título. Representa UN TEMA en particular que se desarrolla completamente. Su contenido puede estar dividido en secciones (la section puede ser hija del article). Por supuesto que puede repetirse ya que, en un blog, por ejemplo, serían cada uno de los posteos.
- **section**: representa y encierra una unidad temática de contenido, donde se "habla" de un mismo tema y diferente del anterior o posterior a él. Puede encerrar varios artículos que tratan un mismo tema, pero también puede tener contenido directo (título, párrafos, imágenes, etc.). Se recomienda que tenga un encabezado. Son literalmente secciones o bloques de información interna de cada página.
- **aside**: contenido aparte del contexto donde se ubique, si no estuviera el contenido de la página no se resiente, su contenido tiene menor peso de indexación (se deja "para el final" ya que su contenido no es relevante). Generalmente contiene una publicidad, formularios, buscador, calendario, widget del clima, etc.
- **figure y figcaption**: el figure es el contenedor semántico de imágenes, gráficos, fotos, etc. Puede contener una o varias imágenes. También puede contener como hijo al figcaption que es el título o bajada de la imagen, pero solo puede contener un único figcaption, el cuál debe ser el primer o último hijo del figure. El figcaption solo puede ser hijo de un figure (no puede usarse dentro de un div, por ejemplo) y puede contener texto directamente o etiquetas semánticas para texto (título, párrafos, etc.).

- **Planificación del sitio**: cada proyecto web presenta una serie de pasos que hay que cumplimentar para obtener un resultado exitoso. Esos pasos son los siguientes:
  - **Relevamiento de la información**: Lo primero a realizar definir el mensaje a transmitir, el tono de la comunicación (cómo le hablaremos al usuario), el público objetivo (a qué usuarios queremos llegar principalmente, conocer sus gustos y necesidades, sus intereses, etc.), el target (franja etaria a la que queremos llegar con el mensaje), etc. Luego debemos recopilar los textos y estructurarlos de manera tal que se presenten de la mejor forma posible, esto se conoce como arquitectura de la información. También debemos recopilar las imágenes a mostrar y optimizarlas en peso y tamaño para web.
  - **Wireframe**: con toda esa información vamos a ir planteando la estructura o el plano de la forma en la que presentaremos los

elementos en cada página (un wireframe por página del sitio), sin colores, textos ni imágenes, solo la estructura dibujada a mano o en pc.
    - **Mapa de navegación**: en esta herramienta planteamos en forma de árbol invertido, la distribución de las secciones que presente el sitio, la relación entre ellas, la navegabilidad del sitio y, en líneas generales, la información que contiene cada una.
    - **Mockup o wireframe de alta**: con el plano y la información lista, se hacen todos los planteamientos estéticos, paleta de colores, tipografías, misceláneas, etc., para producir la interfaz de cada página del sitio.
    - **Maquetación**: en este paso hacemos clickeable el mockup, es decir, maquetamos con html y CSS los elementos visuales que muestran la interfaz para llevarlos a un sitio web funcional.
    - **Testeo de usuarios**: antes de terminar debemos chequear si todas las predicciones y decisiones de organización de elementos y navegación que se tomaron son "entendibles" fácilmente por los usuarios, si la interfaz lograda es simple e intuitiva. Se le pide a distintas personas que naveguen el sitio y se va observando sus reacciones y anotando sus comentarios. Luego se le plantean distintas acciones a realizar en el sitio para ver si las puede lograr. Con esta información tendremos verificaremos si las decisiones tomadas a la hora de producir el sitio fueron correctas o hay que hacer modificaciones en algún o algunos elementos.
    - **Publicación del sitio**: una vez superado el paso anterior (haciendo las modificaciones necesarias) se publica el sitio en Internet para que pueda ser accedido desde cualquier computadora.
- **CSS**: La unidad mínima de aplicación de estilo se denomina regla CSS, que está formada por dos partes: el **selector**, el nombre de la regla, que es la parte que se encarga de seleccionar la/s etiqueta/s a las que se les va a aplicar el formato y la **declaración**, lo que incluimos entre las llaves, donde cada sentencia CSS está compuesta por **propiedad:valor;** y puede incluir la cantidad de sentencias que sea necesario. De acuerdo al tipo de nombre o selector se pueden diferenciar cuatro tipos de reglas CSS:
    - **De identificador**: el nombre de la regla comienza con # y luego, sin dejar espacios, el nombre que queramos para la regla, que puede ser en español, pero sin usar acentos, eñes ni caracteres especiales. Para aplicarlo al html, debemos agregar a la etiqueta el atributo id con el mismo nombre de la regla como valor, pero sin el # inicial. Como el atributo id debe tener valor único, este tipo de regla solo puede usarse una vez por página.
    - **De clase**: el nombre de la regla comienza con . (punto) y luego, sin dejar espacios, el nombre que queramos para la regla (con las mismas recomendaciones hechas anteriormente). Para aplicarlo al html, debemos agregar a la etiqueta el atributo class con el mismo nombre de la regla como valor pero sin el . inicial. El atributo class puede repetir su

valor las veces que queramos, por lo tanto, este tipo de regla se puede aplicar a la cantidad de etiquetas que queramos con el mismo valor (incluso las etiquetas pueden ser iguales o distintas, no es necesario que se aplique solo a muchos párrafos, por ejemplo, sino que la misma regla se le puede aplicar a un párrafo, un título, etc.).

- **De etiqueta**: el nombre de la regla es el de la etiqueta a la que queremos afectar, pero sin los signos < y > que se utilizan para generar la etiqueta html (solo el nombre). Para aplicarlo al html no hay que hacer nada, se aplica directamente a todas las etiquetas de ese nombre (por ejemplo, una regla podría llamarse p y afectar a todos los párrafos del documento directamente).
- **En cascada combinada o compuesta**: el nombre de la regla está formado por varios selectores separados por espacio, los selectores utilizados pueden ser de cualquiera de los tres tipos anteriores y se afecta solo al último selector que sea hijo del anterior y este del anterior si lo hubiera. En el html afectará a todas las etiquetas que cumplan esa relación de anidación o maquetación que se plantea en la regla. Por ejemplo: article p strong{…} en este caso son tres selectores de etiqueta separados por espacios y afecta SOLO a los strong que estén dentro de un párrafo y que el párrafo esté dentro de un article (se aplica directamente por ser tres selectores de etiqueta los que se utilizaron). Otro ejemplo: #cuerpo .derecha p{…} en este caso tenemos un primer selector de identificador, otro de clase y el último de etiqueta, afecta SOLO a los párrafos que estén dentro de alguna etiqueta (cualquiera) que tenga aplicado el atributo class con valor derecha y que, a su vez, esté dentro de otra etiqueta (cualquiera) que tenga aplicado el id cuerpo. Es decir que se refleja una situación de maquetación que debe presentar el documento, si no la encuentra, no se aplica. No es necesario que esa situación de maquetación se refleje completamente, en el ejemplo anterior si la regla es: #cuerpo p{…} afecta también a los párrafos que estén dentro de la etiqueta con id cuerpo, más allá que también estén contenidos por etiquetas con class derecha o no (al incluir más selectores se hace más específica la regla) es decir, que no es necesario que el párrafo sea hijo directamente de #cuerpo, sino con que finalmente esté contenido por él, ya alcanza para aplicarse.

- **Cascada CSS**: este concepto indica que si varias reglas afectan a una misma etiqueta se aplicarán todas, también que si varias hojas de estilo se linkean a un mismo html se aplicarán todas. Pero también que si varias reglas CSS (que provengan de una misma hoja de estilos o no) se aplican a una misma etiqueta y manejan la misma propiedad (una o más de una, por supuesto), pero con distinto valor (lógicamente un texto no se puede mostrar rojo, verde y azul al mismo tiempo, por ejemplo, debe elegir qué color aplicar) **y tienen IGUAL PRECEDENCIA**, se aplicará la última que entró. La última que entró no significa la última que se escribió sobre la etiqueta (por ejemplo si aplicamos varias

reglas de clase) sino la que se escribió más abajo en el documento CSS, si es que provienen de una misma hoja de estilos, o la que se escribió más abajo la etiqueta link que linkea distintas hojas de estilo a un mismo html.

- **Precedencia CSS**: la precedencia sirve para solucionar "conflictos" entre distintas reglas que afectan a una misma etiqueta con la misma propiedad CSS y distintos valores; para esa se aplicará el valor que provenga de la regla con mayor precedencia. La precedencia es la especificidad de una regla, a mayor especificidad (afecta a menor cantidad de etiquetas) mayor será su precedencia. El orden de especificidad o precedencia de mayor a menor es:
    - **Regla en cascada, combinada o compuesta que tenga un selector de identificación** (#contenedor article p).
    - **Regla de identificador** (#izquierda).
    - **Regla en cascada, combinada o compuesta que tenga un selector de clase** (.nota p).
    - **Regla de clase** (.destacado).
    - **Regla en cascada combinada o compuesta con todos sus selectores de etiqueta** (ul li a).
    - **Regla de etiqueta** (p).
    - **Herencia de formato de texto** (es la menor precedencia de todas que puede ser pisada por la regla de etiqueta).
- **Box Model**: todas las etiquetas html son consideradas como cajas por el CSS y presentan las siguientes propiedades (todas numéricas acompañadas por una unidad de medida): width y height (ancho y alto del contenido), padding (separa el contenido del borde), border (el borde de la caja) y margin (espacio transparente que separa la caja del resto de los elementos a su alrededor). Las propiedades son aditivas en cuanto al tamaño de la caja (se suman) y, por lo tanto, se desprenden dos conceptos importantes:
    - **Tamaño visible**: la suma del ancho o alto, el padding y el borde. Si consideramos el ancho visible de una caja resulta de sumar: width, padding-left, padding-right, border-left y border-right.
    - **Tamaño total**: el valor del tamaño visible más el margin. Si consideramos el ancho total de una caja resulta de sumar: ancho visible, margin-left y margin-right.
- **Propiedades shorthand**: las propiedades margin, border y padding aceptan 1, 2 o 4 valores. Si recibe 1 valor se aplica a los 4 costados de la caja. Si recibe 2 valores, el primero se aplica arriba y abajo y el segundo a la izquierda y derecha. Si recibe 4 valores se aplican en sentido horario arriba, derecha, abajo e izquierda. Las propiedades individuales reciben un único valor (por ejemplo, margin-top).
- **Centrado de cajas de bloque**: si una caja en bloque tiene un ancho menor al de su contenedor, la centramos aplicando margin-left y margin-right con valor auto (reparte el espacio sobrante a la derecha de la caja en partes iguales a sus lados). El valor auto arriba y abajo no agrega margen ni produce ningún efecto.

- **Propiedades del box model en cajas de bloque**: aplican todas las propiedades de caja desde CSS. Fusionan los márgenes verticales entre cajas de bloque consecutivas (el margin-bottom de la primera y el margin-top de la segunda) expresando solo el que tenga mayor valor de los dos. También fusionan márgenes verticales entre cajas de bloque anidadas (ambos margin-top o margin-bottom) expresando solo el que tenga mayor valor de los dos que se aplica al contenedor (aunque se aplique margin-top al hijo solo, ese margen se le aplica al padre ya que su margen es cero).
- **Propiedades de box model en cajas en línea**: no aplican valores ni de width ni de height, aplican padding, border y margin del eje horizontal (izquierda y derecha), el padding y border del eje vertical se aplica, pero no ocupa espacio (no empuja, es decir no hace crecer a la caja obligándola o ocupar mayor espacio) y se solapan con los elementos o texto de la línea superior e inferior; el margen del eje vertical no se aplica directamente.
- **Contenido de cajas de bloque**: el contenido se adapta al ancho de la caja y cuando ya no entra, salta de línea y la hace crecer automáticamente en alto. Si limitamos el alto de la caja y su contenido no entra, la excede por su borde inferior, nunca por el borde derecho.
- **Contenido excedente de una caja de bloque**: el contenido excedente se maneja con la propiedad overflow que recibe los valores:
    - **visible**: el contenido excedente es visible y desborda a la caja por la parte inferior (comportamiento por defecto).
    - **hidden**: el contenido excedente se oculta y no se visualiza todo aquel contenido que supere los límites de la caja.
    - **scroll:** agrega siempre las barras de scroll horizontal y vertical internas de la caja aunque no necesite ambas (la que no se necesita, la horizontal, aparece deshabilitada y si el contenido no excede la caja, se muestran ambas deshabilitadas).
    - **auto:** agrega solo la barra de scroll que necesita (la que permita visualizar el contenido excedente, en general la vertical) y solo si la necesita (si la caja tiene aplicado este valor y el contenido no la excede no se muestra la barra).
- **Contenido de cajas en línea:** tendrán siempre el alto del contenido (alto de la línea de texto) y también el ancho de su contenido (con más contenido crecen en ancho y si no entran horizontalmente en su contenedor, saltan a la línea de abajo la parte del texto que no entre). Por defecto tienen el ancho de su contenedor y el alto de su contenido.
- **Propiedad display:** determina como se renderiza una caja independientemente de su naturaleza. Sus valores son:
    - **block:** trabaja en bloque con todas las características de las cajas de bloque mencionadas.
    - **Inline:** trabaja en línea con todas las características de las cajas en línea mencionadas.
    - **none:** no se dibuja en pantalla, no ocupa lugar.

- **Inline-block**: es una mezcla de las mejores características de las dos naturalezas (no existe ninguna etiqueta inline-block por default, se debe forzar a trabajar de esta manera desde CSS): aplican width, height, padding, border y margin en los 4 costados de la caja ocupando lugar (características de las cajas de bloque), se ubican una al lado de la otra o en línea con etiquetas inline o texto (características de las etiquetas de línea), no fusionan márgenes entre ellas, por defecto tienen el ancho de su contenido y si no entran en el ancho de su contenedor bajan a la línea siguiente. Si tienen mayor alto que el alto de línea lo estiran y lo obligan a crecer. Dejan por defecto un espacio entre cajas inline-block consecutivas que corresponde al espacio que naturalmente también separan a las palabras (el texto que trabaja en línea también).

- **Corrección del espacio entre cajas inline-block:** para eliminar el espacio que queda entre cajas inline-block se debe "pegar" la etiqueta html de cierre de la primera caja con la de apertura de la segunda directamente (sin dejar ni un solo espacio entre ellas) o hacerlo mediante un comentario html (la apertura del comentario pegada a la etiqueta de cierre de la primera caja y la de cierre del comentario pegada a la de apertura de la segunda caja).
- **Centrado y otras particularidades de las cajas inline-block**: las cajas inline-block funcionan en la línea de texto por lo que se centran aplicando text-align center al contenedor de dichas cajas. Como por defecto tienen el ancho de su contenido, si queremos colocar dos cajas inline-block una al lado de la otra no alcanza solo con ajustarles este display, sino que, además, hay que darles un ancho a cada una con tal valor que asegure que "entren" ambas en una línea sin caer a la de abajo. Si queremos alinear verticalmente cajas inline-block con distinto alto, debemos aplicarles la propiedad vertical-align con los valores: top (se alinean en la parte superior), bottom (se alinean abajo), middle (al centro) o baseline (coinciden en la línea los últimos elementos o líneas de texto internas de ambas cajas).
- **Centrado vertical**: si una caja tiene una sola línea de texto (solo una) y la queremos centrar verticalmente (horizontalmente se logra aplicando text-align center al contenedor) dentro de su contenedor, debemos aplicarle el mismo valor de line-height que de height al contenedor. De esta manera el alto de línea ocupa todo el alto de la caja (por eso debe tener solo una línea de texto) y la línea de texto se ubica justo al medio del alto de línea.
- **Logo del sitio**: debe ser la h1 a la que se le agrega texto (el nombre de la empresa o producto) en el html y, desde el CSS, se aplica el ancho y alto de la h1 con los valores de ancho y alto de la imagen del logo; la imagen como fondo y se oculta el texto utilizando una de dos opciones: font-size cero (evita que el texto ocupe lugar y se dibuje) o text-indent negativo y suficiente para sacar todo el texto por el lado izquierdo de la caja y overflow hidden para ocultarlo.
- **Barras de navegación**: Requieren 4 propiedades que no deben faltar en barras horizontales y 5 en barras verticales. Horizontales, 2 propiedades al li: list-style none (saca los puntos de los li) y un ancho al botón (para contener al vínculo en

bloque) y 2 propiedades al vínculo: text-decoration none (los textos de los botones no deben subrayarse en reposo) y display block (para que ocupen todo el ancho del botón y reaccione al click del mouse en toda su superficie y no solo en el texto del botón ya que el vínculo es una etiqueta en línea por naturaleza). Verticales, se agrega 1 propiedad al li para ponerlos en línea: display inline-block o flotación.

- **Flotación:** las cajas pueden flotar a la izquierda o a la derecha y lo hacen desde el lugar que ocupaban en el flujo normal o natural del documento (al que dejan de pertenecer) dentro de su contenedor. Los elementos que quedan en el flujo ocupan el lugar que deja el elemento flotado y ni siquiera su contenedor lo reconoce como hijo (ya que no pertenece más al flujo). Si el elemento flotado tapa a un elemento que quedó en el flujo, éste desplaza su contenido para que no quede tapado, hacia la derecha si le da el ancho del contenedor o su propio ancho, o hacia abajo, lo suficiente como para que ya el elemento flotado no lo tape. Los elementos flotados consecutivos se ubican uno al lado del otro en el plano de flotación (a la izquierda o a la derecha). Y tienen las características similares al elemento inline-block: aplican width, height, padding, border, margin en los 4 costados de la caja, no fusionan márgenes, por defecto tienen el ancho de su contenido y si no entran en ancho dentro del contenedor bajan a una línea inferior de flotación.
- **Correcciones de la flotación:** Un contenedor que tenga todos sus hijos flotados no crece, para lograr que lo haga y contenga a sus hijos (para evitar que estos se solapen con los elementos que siguen) le aplicamos al contenedor overflow con valor hidden o auto. Si queremos que 2 elementos flotados se ubiquen uno al lado del otro, debemos ajustarles el ancho suficiente para que entren en el ancho de su contenedor. Si queremos que el elemento que continúa en el código html a los elementos flotados y que está en el flujo normal no se ubique por detrás del flotado sino por debajo (a continuación) le aplicamos la propiedad clear (limpieza de flotación) con los valores left (se ubica a continuación del último elemento flotado a la izquierda), right (idem pero del último flotado a la derecha) o both (por debajo del último elemento flotado en cualquier sentido de flotación).
- **Posicionamiento static:** es el posicionamiento por defecto de todas las cajas, las que se ubican de acuerdo a su naturaleza: las de línea una al lado de la otra y las de bloque una debajo de la otra ocupando todo el ancho de su contenedor. Es lo que definimos como flujo normal o natural del documento.
- **Posicionamiento relative:** sigue perteneciendo al flujo normal o natural, pero puede desplazarse de su posición con las propiedades de desplazamiento: top, left, right o bottom. Todas estas propiedades reciben valores positivos o negativos que significan sentido de desplazamiento, positivos: hacia la derecha o abajo, negativos: hacia la izquierda o abajo. También pueden tapar a otra caja al desplazarse y la caja tapada no desplaza su contenido. Si dos o más cajas se solapan la que queda arriba dependerá del valor de z-index (numérico sin

unidad de medida, el que tenga el valor más pequeño va abajo y el que tenga el valor más alto, por encima de las demás).

- **Posicionamiento absoluto:** deja de pertenecer al flujo normal o natural del documento. Si la página tiene scroll, el elemento absoluto scrollea. Se ubica y desplaza con las propiedades de desplazamiento que, en este caso, simbolizan el desplazamiento desde el borde del contenedor que indique la propiedad (top, left, right o bottom). El contendor a partir del cual se ubican y desplazan será el primero que encuentre con posicionamiento distinto de estático, si no encuentra ninguno, utiliza la ventana del browser (si su padre, o el padre del padre o el padre del padre del padre… tiene posicionamiento relativo, absoluto o fijo, se desplaza y ubica a partir de esa caja y no de la ventana del browser).
- **Posicionamiento fijo:** deja de pertenecer al flujo normal o natural del documento. Si la página tiene scroll, esta caja no scrollea, queda fija en su lugar. Siempre se desplaza desde la ventana del browser (no reconoce a ningún elemento padre para desplazarse) con las mismas propiedades de desplazamiento (top, left, right o bottom) que representan el desplazamiento desde ese lado de la ventana del navegador.
- **Flotación y posicionamiento:** una caja flotada y con posicionamiento relativo, aplica ambas propiedades y si se desplaza, lo hace desde el lugar que le corresponde por flotación. Si al desplazarse tapa a otra caja, esa caja no desplaza su contenido (aunque esté flotada) porque el solapamiento se produjo por el desplazamiento por su tipo de posicionamiento (si desplaza el contenido cuando queda tapada por el lugar que la caja ocupa por flotación). Una caja flotada y posicionada absoluta o fija, no aplica la flotación (ni tampoco el display inline-block ya que estos posicionamientos sacan del flujo a la caja).