**12- Formas de agregar estilos a HTML**

Hay tres opciones para incluir estilos que definan la apariencia de tu html:

Estilos en línea: se definen directamente en el elemento html que quieres estilizar, se agregan con el atributo style.

Estilos con el tag Style: regularmente este tag se incluye dentro de la etiqueta head del html.

Estilos enlazados desde un archivo css externo: utilizando la etiqueta link que nos permite enlazar recursos externos.

A CSS, se le llama hojas de estilos en cascada porque los estilos que se definen para una página, se van aplicando de arriba hacia abajo, y de lo más general a lo más particular, teniendo prioridad lo más particular. Esto es, los estilos que prevalecen son los que han sido definidos en línea, luego los que fueron definidos mediante la etiqueta style en la cabeza o cuerpo del html, y por último los estilos definidos en archivos externos enlazados con la etiqueta link. Esta prioridad se puede alterar al usar el modificador \*\*!important"" en la definición de algún estilo en particular, aunque esto no es recomendado.

**15- Agregando clases a los componentes escritos en HTML**

Para aplicar estilos a los componentes html, lo más común y recomendable es hacerlo a través de clases que se asignan al elemento html mediante el atributo class.

Un elemento html puede tener varias clases, se deben indicar en el mismo atributo class pero separadas por un espacio en blanco.

Al escoger los nombres de clases, debemos tener en cuenta que se puedieran aplicar a muchos elementos, o a elementos particulares, así que la claridad y precisión en su identificación facilitará la contextualización y mantenibilidad en el futuro.

Algunos de los estándares más usados para la identificación de clases son:

OOCSS

BEM

Component CSS

**16- Unidades de medida y colores**

Hay varias unidades de medida con las que se puede trabajar en CSS: %, em, rem, px, pt, fr, vw, vh

Las medidas más comunes y utilizadas son los pixeles. Un pixel es la menor unidad homogénea en color que forma parte de una imagen digital. Es la unidad más práctica y fácil de utilizar y manipular, y es la que utilizaremos mayormente en este curso.

Los colores en CSS pueden ser representados de al menos tres formas diferentes:

Representados con palabras claves para cada color, como: red, green, blue, pink, yellow, black, etc.

Usando la composición de tres colores (rojo, verde y azul): para esto podemos usar notación exadecimal o las funciones rgb() y rgba().

Usando la composición mediante valores de Matiz, Saturación y Luminosidad con: hls() y hlsa().

Con respecto a los valores hexadecimales, cada color está representado por 6 digitos, que representan 3 pares de hexadecimales: FF - FF - FF (rojo, verde y azul), en el que cada par puede tomar valores hexadecimales entre 00 y FF. Cada uno equivale a valores decimales entre 0 y 255, donde 0 es la ausencia de ese color y 255 la mayor cantidad disponible. De esta manera cada color se forma por la combinación de diferentes proporciones de rojo, verde y azul.

#000000 es equivalente a Negro

#FF0000 es equivalente a Rojo

#00FF00 es equivalente a Verde

#0000FF es equivalente a Azul

#FFFFFF es equivalente a Blanco

**17- Inspector de elementos**

Para ver y depurar el código de una página html, el navegador incluye una herramienta llamada **Inspector de elementos**, o simplemente **inspector**, que abre, en una sección de la ventana, una serie de espacios con información técnica muy detallada sobre todo lo que sucede en el DOM, incluídos los estilos que tienen aplicados cada uno de los elementos del html.La mayoría de los navegadores incluye algún tipo de **Inspector**, en el curso usamos Google Chrome, pero la misma herramienta (o similar) la encuentras en Firefox, Opera, Edge, etc.Utilizando el Inspector podemos hacer modificaciones (temporales) manualmente en el html de cualquier sitio web, consultar sus estilos y recursos enlazados, hacer pruebas en tiempo real con JavaScript, monitorear variables o eventos entre muchas otras tareas útiles para cualquier desarrollador.

**18-Tipos de textos personalizados**

Los tipos de texto, también conocidos como tipos de letras o fuentes, son el conjunto de diseños tipográficos que representan a cada una de las letras y los caracteres gráficos en el documento. Su nombre correcto es tipografía. Los diferentes tipos de fuente están basados en archivos que existen en cada sistema operativo.

Algunos ejemplos de tipos de texto o fuentes, son:

Arial

Times New Roman

Verdana

DeJaVu

Lato

OpenSans

Roboto

CSS permite utilizar fuentes diferentes a las disponibles en el sistema operativo del cliente, mediante la importación o el enlace a archivos de fuentes externas. Las más usadas son las que están disponibles a través del sitio web de Google Fonts.

Al definir el tipo de texto asociado a una clase css con la propioedad font-family indicamos al navegador, que debe intentar usar esa fuente en particular para darle la apariencia tipográfica a los textos de ese elemento html.

**19- Propiedades para los textos**

Además de todas las propiedades comunes que comparten los elementos estándar de html, como: display, position, margin, padding, top, left, right, bottom, border, etc., los elementos que admiten contenidos textuales aceptan una serie particular de propiedades entre las que se encuentran las siguientes:

* **font-family**: define el tipo de fuente aplicado al texto.
* **color**: define el colore del texto.
* **line-height**: define la altura desde la base del texto hasta la base de la siguiente línea de texto.
* **font-size**: define el tamaño del texto, admite cualquiera de las unidades de medida disponibles.
* **letter-spacing**: define el espaciado entre las letras del texto.
* **font-weight**: define el ““peso”” de la letra, negrita, normal, light y normalmente se indica en múltiplos de 100 o usando keywords.
* **text-decoration**: define el decorado del texto como subrayado, tachado, con subrayado superior, etc.
* **text-transform**: permite transformar el estado de mayúsculas / minúsculas en el texto, usando uppercase para mayúsculas sostenidas, lowercase para minúsculas sostenidas, etc.

**20- Dimensiones fijas para elementos**

Todos los elementos html comparten algunas propiedades de estilo, entre éstas se encuentran las propiedades relacionadas con sus dimensiones: **width** (ancho) y **height** (alto).

Al manipular las propiedades de dimensiones hay que tener en cuenta que si los contenidos de los elementos que estamos estilizando, son más grandes que las dimensiones que hemos indicado, se pudieran generar resultados inesperados en la apariencia, como solapamiento o desbordamiento.

**21- Backgrounds de color e imagen**

Algunas de las propiedades de css relacionadas con la apariencia del fondo de los elementos son:

* **background**: con la que se puede indicar un color, o usada de manera extendida, puede incluir color de fondo, url de la imagen, posición y modo de repetición de la imagen.
* **background-image**: contiene la url que se usará como fondo del elemento.
* **background-color**: indica el color de fondo, se puede usar en combinación con la imagen.
* **background-size**: se puede indicar en valores de alto y ancho o en alguna de las palabras claves permitidas: cover o contain.
* **background-position**: indica la posición de la imagen dentro del elemento, puede indicarse en unidades o en palabras claves como center, left, top y right.
* **background-repeat**: indica el método de repetición de la imagen de fondo, puede ser: repeat, repeat-x, repeat-y o no-repeat.

**22- Bordes**

Todos los elementos html admiten la propiedad de css **border**, que define la apariencia que tendrá el contorno del componente.  
El borde puede ser de muchos estilos, y al igual que las propiedades margin y padding que aprenderás más adelante, a los bordes se les puede colocar estilos tanto de forma general con la propiedad **border**, como de acuerdo al lado del elemento que se indique: border-top, border-right, border-bottom y border-left.

Con la propiedad **boder-radius** se define el redondeado de las esquinas de los bordes.

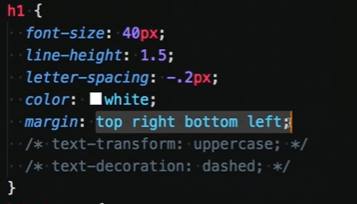
**23- Márgenes**

Los márgenes en CSS son el espacio que separa a los elementos html entre sí. Hay elementos de html que traen márgenes predefinidos (poe defecto) en los estilos propios del navegador como el caso de: body, h1, h2, h3, h4, h5, h6, ol, ul, li, p, y muchos otros.

Cuando hay dos márgenes de elementos diferentes que colindan entre sí, se presenta una situación llamada ““margin collapsing”” en la que el mayor margen de los dos se superpone al otro.

Se puede asignar una medida de margin para los cuatro lados del elemento, o márgenes individuales para cada uno de los lados con: margin-top, margin-right, margin-bottom y margin-left.

Se puede centrar un elemento html colocándole el valor de **margin: 0 auto**, cuando dicho elemento tiene display *block*.



**24- Rellenos**

Así como el margen separa a los elementos html entre sí, la propiedad **padding** de relleno, permite definir una separación entre el contenido interno y el *borde* de un elemento.

Al inspeccionar los elementos html en el navegador, se puede apreciar el margin con color naranja y el padding con color verde.

Una forma de identificar cuándo es mejor usar margin o padding en un elemento, es evaluando la necesidad de usar borde o background, ya que son éstos: el borde y el background, los que realmente diferencian el uso de uno u otro.

--el margen del h2 no empuja a su contenedor, solo sobre sale--

**25- Modelo de caja**

El modelo de caja es un concepto teórico de css que representa a cada elemento html en base sus propiedades de: **margin**, **border**, **padding** y **dimensiones** (alto y ancho).  
Para visualizar un elemento html en su representación como modelo de caja debemos irnos a la parte baja de la sección *styles* del inspector de elementos, o en la sección llamada **Computed**.

En el modelo de caja, el **ancho total** de un *elemento html* equivale a la sumatoria de los valores de: **width**, **padding-left**, **padding-right**, **border-left-width**, **border-right-width**. De manera similar aplica para el **alto total** de cada *elemento*. Aunque **margin-left** y **margin-right**, forman parte del modelo de caja, no se incluyen para el calculo del ancho total.

Con la propiedad **box-sizing**, y en particular con el valor **border-box** de esta propiedad, podemos modificar el comportamiento del modelo de caja para que nuestro elemento nunca supere el tamaño máximo que le hayamos definido en **width** y **height**. Esta es la opción recomendada para trabajar.

**26- Tipos de display**

Display es la propiedad de css que indica cómo debe ser mostrado un elemento html. Todos los elementos tienen algún tipo de display. Si un elemento no se ve en pantalla es porque seguramente su display es none.

Los valores más comunes que puede recibir la propiedad **display** son:

* **block**: el elemento intenta abarcar todo el ancho posible.
* **inline**: reduce su tamaño exclusivamente hasta lo que abarca su contenido, descartando las propiedades width y height.
* **inline-block**: combina lo mejor de block e inline, ya que respeta las dimensiones indicadas en las propiedades width y height, pero coloca el elemento en línea (al costado) de elementos hermanos que también tengan display: inline o inline-block.
* **flex**: asume algunas propiedades por defecto que favorecen la alineación de los elementos internos.
* **grid**: similar a flex, asume algunas propiedades por defecto organizando los contenidos en filas y columnas.
* **none**: oculta el elemento.

**27- Propiedades de flexbox**

Flexbox se refiere al tipo de display en css que permite un manejo *flexible* de la alineación, dimensionamiento y distribución de elementos html.

Esta propiedad se aplica a un elemento padre, pero va a afectar principalmente a sus elementos hijos directos. Por defecto, los elementos internos quedan alineados unos seguidos de los otros. El comportamiento del modelo de caja de estos elementos hijos también se ha modificado, ya que pierden el efecto de su propiedad margin.

Los elementos hijos de un padre con propiedad **display: flex** tienen a su disposición algunas nuevas propiedades que aportan mayor flexibilidad a su comportamiento. Una de estas propiedades es **flex-shrink** que, junto a la propiedad **flex-wrap** del padre, permite adaptar y distribuir los elementos de manera dinámica en el espacio horizontal disponible hasta ocupar todo el espacio, y luego pasar a ocupar dinamicamente las siguiente filas hacia abajo.

**28- Alineando elementos de forma horizontal**

La propiedad de css que nos permite definir la forma en que se alinearán o distribuirán los hijos de un elemento al que se le ha asignado un *display flex* es: **justify-content**. Y puede tomar entre otros valores, los siguientes:

* **flex-start**: para alinear todos los elementos hacia el inicio del espacio disponible.
* **flex-end**: para alinear todos los elementos hacia el final, a la derecha.
* **center**: para alinear todos los elementos al centro del espacio disponible.
* **space-between**: para distribuir los elementos con un espacio proporcional e igual entre ellos.
* **space-evenly**: para distribuir los elementos con un espacio proporcional e igual entre ellos (incluyendo el primer y último elementos con respecto a los extremos del espacio disponible).
* **space-around**: similar a *space-evenly* pero tanto en el primero como en el último elemento, el espacio hacia los extremos es la mitad del espacio usado entre los elementos.

**29- Alineando elementos de forma vertical**

Similar a como sucede con *justify-content*, es posible alinear y distribuir los elementos internos en el espacio vertical disponible usando la propiedad **align-items**, que puede tomar también los valores de: **flex-start**, **flex-end** y **center**.

Algo que es muy importante y se debe tener en cuenta a la hora de usar **align-items** y **justify-content** es que dependiendo de la propiedad **flex-direction** que se haya definido, el efecto de ambos se invierte, no en cuanto a sus elementos internos, sino en cuanto a si se debe usar uno u otro de manera vertical u horizontal.

*IMPORTANTE*: Cuando la propiedad **flex-direction** se ha definido como **column**, la propiedad **justify-content** ya no va a aplicar sobre la alineación horizontal, sino sobre la vertical. Y **align-items** ya no aplicaría sobre la alineación vertical sino la horizontal. Se intercambian sus efectos.

Para centrar completamente los elementos internos de manera vertical y horizontal en su elemento padre, debemos usar el valor **center** en ambas propiedades.

**31- Estilos de la sección principal y de footer**

En esta clase aplicaremos a nuestro proyecto los estilos necesarios para que la sección principal o **hero** de nuestro portafolio se vea correctamente. También completaremos la información de nuestro footer y le aplicaremos los respectivos estilos.

En esta clase vemos que además de las propiedades relacionadas con *display:flex*, en lo que se refiere a la alineación de elementos internos, existen también propiedades equivalentes que nos permite alinear textos dentro de un contenedor, estas propiedades son: **text-align** y **vertical-align**.