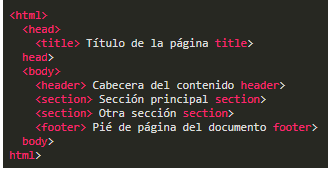
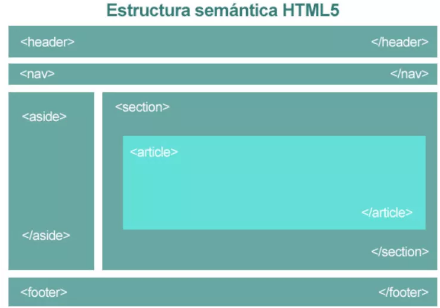
**Estructura de nuestro Sitio Web**





**Atributos HTML**

Los atributos son valores agregados a las etiquetas (tags) html que extienden su habilidad o funcionalidad con información específica.

A continuación, un ejemplo de los atributos más comunes y usados en algunas etiquetas:

Para **img**:

* **src**: específica la *ruta* de la imagen que será mostrada a través de esta etiqueta. La ruta puede ser *absoluta*(cunado especifica una dirección exacta, incluyendo el prefijo *http(s)* ) o *relativa* (cuando la referencia a la ubicación de la imagen parte de la ubicación del archivo actual).
* **alt**: indica un texto alternativo que será mostrado en lugar de la imagen cuando ésta no pueda ser mostrada.
* **width**: ancho de la imagen en pixeles.
* **height**: alto de la imagen en pixeles.

Para **link**, en la cabecera *head* del documento:

* **rel**: indica la relación del recurso con el contenido.
* **type**: indica el tipo de recurso / formato.
* **href**: indica la ubicación (url) del recurso enlazado.

Para **meta**, ambién en la cabecera *head* del documento:

* **charset**: indica la tabla de caracteres (utf-8 para caracteres latinos) usada en el documento.

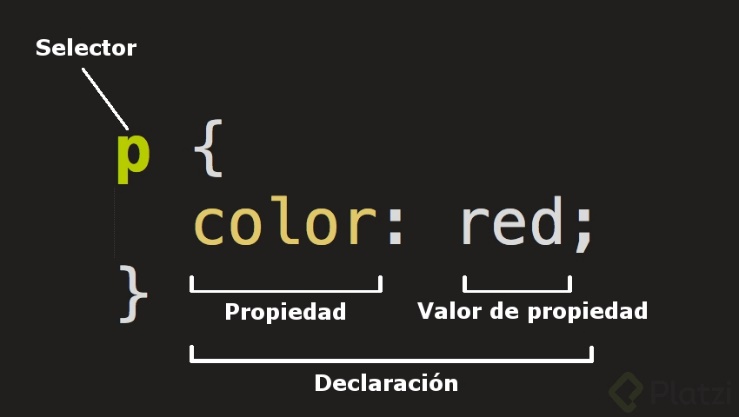
Para **a**:

* **href**: la ubicación o ruta a la que enlaza esta etiqueta de ancla. En el caso de querer enlazar a elementos que se encuentran dentro del mismo documento, este atributo debe indicar el valor del atributo ““id”” de ese elemento destino del enlace.

**GENERADOR DE LOGOS**

<https://realfavicongenerator.net/>

**Reglas, selectores, declaraciones, propiedades y valores de CSS**



**También recordar que existen varios tipos de selectores:**  
**1. Selector de etiqueta:** refiere a etiquetas HTML, como el ejemplo de la imagen.  
**2. Selector descendiente:** refiere a etiquetas HTML de forma hereditaria.  
**3. Selector de id:** refiere a atributos id de etiquetas HTML.  
**4. Selector de class (clase):** refiere a atributos class de etiquetas HTML.

**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

1. Palabra clave: red, blue, pink, etc.
2. Hexadecimales: 0123456789ABCDEF
3. hls() hlsa()

* #000000 = Negro
* #FFFFFF = Blanco

**Se representan por grupos de 2 ->** Red, Green, Blue

* #FF0000 = Rojo
* #00FF00 = Verde
* #0000FF = Azul

**Para obtener un color más exacto se utiliza rgb o rgba:**

* rgb(0, 200, 145)
* rgba(0, 50, 70, 0.5) para obtener transparencia

**¿Qué es un pixel?**  
La menor unidad homogenea en color que forma parte de una imagen di

**Tipos de textos personalizados**

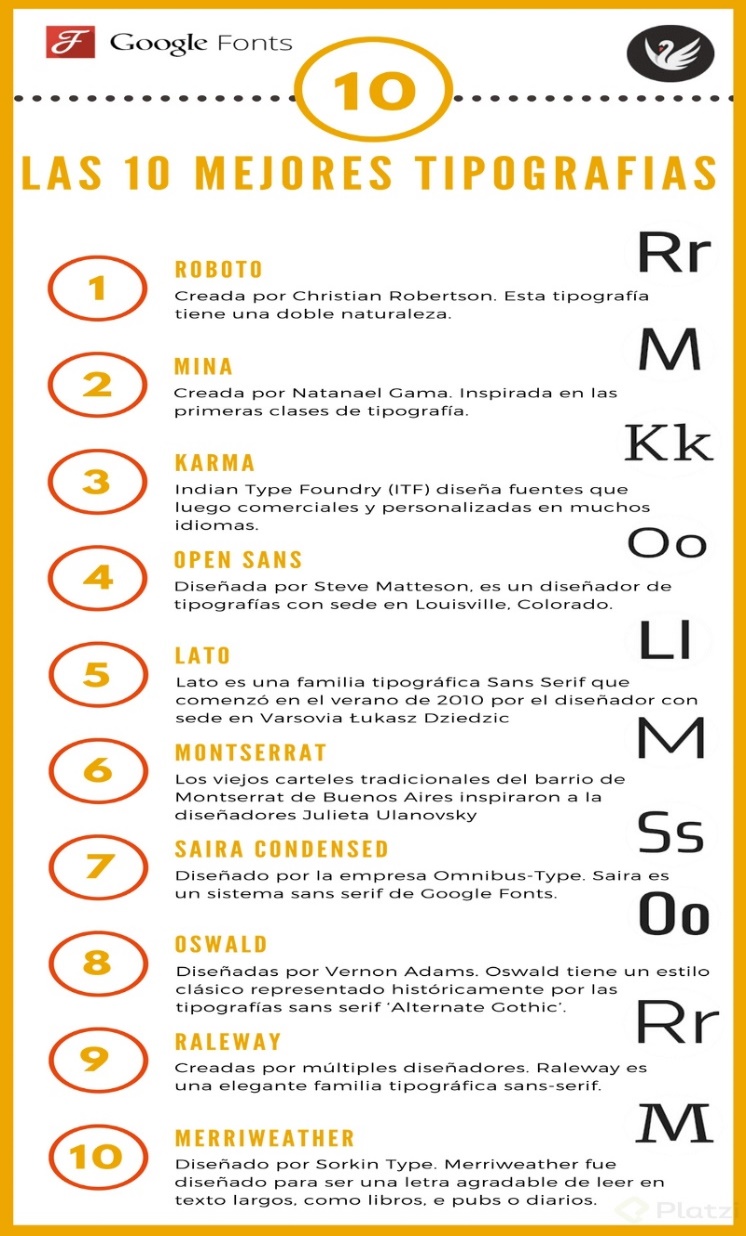
Los tipos de texto, también conocidos como **tipos de letras** o **fuentes**, son el conjunto de diseños tipográficos que representan a cada una de las letras y los caracteres gráficos en el documento. Su nombre correcto es **tipografía**. Los diferentes tipos de fuente están basados en archivos que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



**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.
* line-height para modificar el alto de linea  
  font-size para modificar el tamaño de la fuente  
  font-weight para modificar el tipo de fuente  
  font-style para modificar el estilo de la fuente  
  letter-spacing para modificar el espacio entre letras  
  text-transform para transformar la fuente (mayusculas, minuculas, etc)  
  text-decoration para moodificar la decoración de la fuente

font-weight me gusta mas decirlo que modifica el peso de la fuente y negrita

**Backgrounds de color e imagen**

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

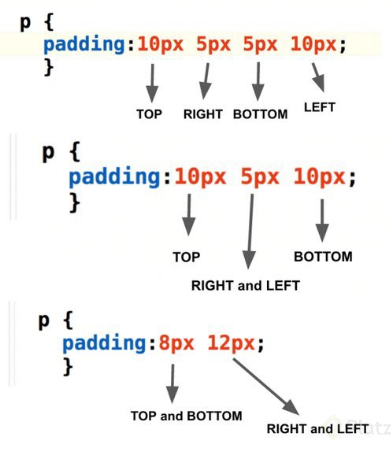
* **background**: con la que se puede indicar un color, o usada de manera extendida, puede incluir color de fondo, url de la imagen, posición y modo de repetición de la imagen.
* **background-image**: contiene la url que se usará como fondo del elemento.
* **background-color**: indica el color de fondo, 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.

**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.



**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*.

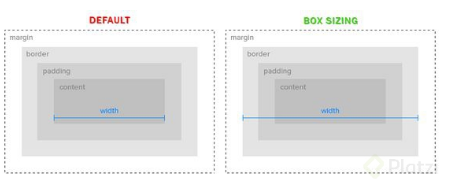


**Modelo de caja**

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 recomendad para trabajar.

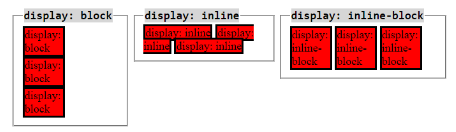


**Tipos de display**

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 segurmanete 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.



**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.

**justify-content: xxxx;**

**flex-start**: Alinea elementos al lado izquierdo del contenedor.

**flex-end**: Alinea elementos al lado derecho del contenedor.

**center**: Alinea elementos en el centro del contenedor.

**space-between**: Muestra elementos con la misma distancia entre ellos.

**space-around**: Muestra elementos con la misma separación alrededor de ellos.

**align-items: xxxx;**

**flex-start**: Alinea elementos a la parte superior del contenedor.

**flex-end**: Alinea elementos a la parte inferior del contenedor.

**center**: Alinea elementos en el centro (verticalmente hablando) del contenedor.

**baseline**: Muestra elementos en la línea base del contenedor

**stretch**: Elementos se estiran para ajustarse al contenedor.

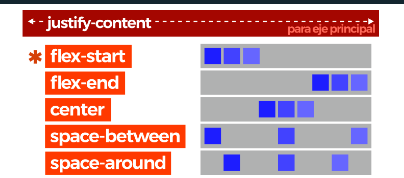
**flex-direction: xxxx;**

**row**: Elementos son colocados en la misma dirección del texto.

**row-reverse**: Elementos son colocados en la dirección opuesta al texto.

**column**: Elementos se colocan de arriba hacia abajo.

**column-reverse**: Elementos se colocan de abajo hacia arriba.



[https://darekkay.com/dev/flexbox-cheatsheet.html](https://darekkay.com/dev/flexbox-cheatsheet.html" \t "_blank)

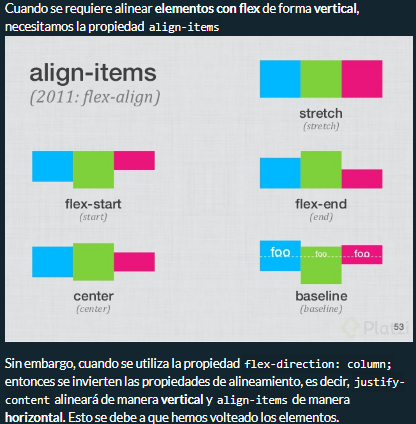
**Alineando elementos de forma vertical**

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.



**Estilos sección de eventos**

En esta clase definimos los estilos de la sección de eventos de nuestro portafolio. Vemos también algunas propiedades nuevas como **overflow**, **object-fit**, **position** y **z-index**.

<http://es.learnlayout.com/position.html>

<http://es.learnlayout.com/display.html>