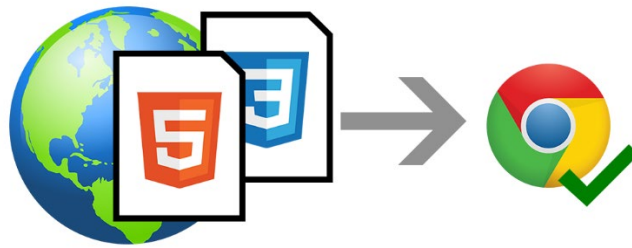


Fuente: lenguajehtml.com y lenguajecss.com

¿Qué es realmente CSS?

Antes de comenzar, debes tener claro un concepto clave: una página web es realmente un documento de texto. En dicho documento se escribe código HTML, con el que se crea el contenido de una web. Por otro lado, existe el código CSS, que unido al código HTML permite darle forma, color, posición (*y otras características visuales*) a una página.

En resumen, se trata de un idioma como podría ser el inglés o el alemán, que los navegadores web como Chrome o Firefox conocen y pueden entender. Nuestro objetivo como diseñadores y programadores web es precisamente ese: aprender el idioma.



Las siglas CSS (*Cascading Style Sheets*) significan «Hojas de estilo en cascada» y parten de un concepto simple pero muy potente: aplicar estilos (colores, formas, márgenes, etc...) a uno o varios documentos (*generalmente documentos HTML, páginas webs*) de forma masiva.

Se le denomina estilos en cascada porque se aplican de arriba a abajo y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

La idea de CSS es la de utilizar el concepto de separación de presentación y contenido, intentando que los documentos HTML incluyan sólo información y datos, relativos al significado de la información a transmitir (*el contenido*), y todos los aspectos relacionados con el estilo (diseño, colores, formas, etc...) se encuentren en un documento CSS independiente (*la presentación*).



De esta forma, se puede unificar todo lo relativo al diseño visual en un solo documento CSS, y con ello, varias ventajas:

- Si necesitamos hacer modificaciones visuales lo hacemos en un sólo lugar y no tenemos que editar todos los documentos HTML en cuestión por separado.
- Se reduce la duplicación de estilos en diferentes lugares, por lo que es más fácil de organizar y hacer cambios. Además, al final la información a transmitir es considerablemente menor (las páginas se descargan más rápido).
- Es más fácil crear versiones diferentes de presentación para otros tipos de dispositivos: tablets, smartphones o dispositivos móviles, etc...
- Antes de comenzar a trabajar con CSS hay que conocer las diferentes formas para incluir estilos en nuestros documentos HTML, ya que hay varias, cada una con sus particularidades y diferencias.
- En principio, tenemos tres formas diferentes de hacerlo, siendo la primera la más común y la última la menos habitual:

Nombre	Método	Descripción
CSS Externo	Etiqueta <link>	El código se escribe en un archivo .css aparte. Método más habitual.
CSS Interno	Etiqueta <style>	El código se escribe en una etiqueta <style> en el documento HTML.
Estilos en línea	Atributo style="..."	El código se escribe en un atributo HTML de una etiqueta.

Veamos cada una de ellas detalladamente:

Enlace a CSS externo (link)

En la cabecera de nuestro documento HTML, más concretamente en el bloque <head></head>, podemos incluir una etiqueta <link> con la que establecemos una relación entre el documento actual y el archivo CSS que indicamos en el atributo href:

```
<link rel="stylesheet" href="index.css" />
```

De esta forma, los navegadores sabrán que deben aplicar los estilos que se encuentren en el archivo index.css. Se aconseja escribir esta línea lo antes posible (*sobre todo, antes de los scripts*), obligando así al navegador a aplicar los estilos cuanto antes y eliminar la falsa

percepción visual de que la página está en blanco y no ha sido cargada por completo.

- Esta es la manera recomendada de utilizar estilos CSS en nuestros documentos.

Incluir CSS en el HTML (style)

Otra de las formas habituales que existen para incluir estilos CSS en nuestra página es la de añadirlos directamente en el documento HTML, a través de una etiqueta `<style>` que contendrá el código CSS:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Título de la página</title>
    <style>
      div {
        background: hotpink;
        color: white;
      }
    </style>
  </head>
  ...
</html>
```

Este sistema puede servirnos en ciertos casos particulares, pero hay que darle prioridad al método anterior (*CSS externo*), ya que incluyendo el código CSS en el interior del archivo HTML arruinamos la posibilidad de tener el código CSS en un documento aparte, pudiendo reutilizarlo y enlazarlo desde otros documentos HTML mediante la etiqueta `<link>`.

- Aunque no es obligatorio, es muy común que las etiquetas `<style>` se encuentren en la cabecera `<head>` del documento HTML, ya que antiguamente era la única forma de hacerlo.

Estilos en línea (atributo style)

- Por último, la tercera forma de aplicar estilos en un documento HTML es hacerlo directamente, a través del atributo style de la propia etiqueta donde queramos aplicar el estilo:

```
<p>iHola <span style="color:red">amigo lector</span>!</p>
```

- De la misma forma que en el método anterior, se recomienda no utilizarse salvo casos muy específicos, ya que los estilos se asocian a la etiqueta en cuestión y no pueden reutilizarse. Sin embargo, es una opción que puede venir bien en algunos casos.

Al igual que los documentos HTML, los documentos CSS son archivos de texto donde se escribe una serie de órdenes y el cliente (navegador) las interpreta y aplica a los documentos HTML asociados.

Sintaxis básica

La estructura CSS se basa en reglas que tienen el siguiente formato:



- **Selector:** El selector es el elemento HTML que vamos a seleccionar del documento para aplicarle un estilo concreto. *Por ejemplo, el elemento p.* Realmente, esto es mucho más complejo, pero ya dedicaremos una serie de capítulos exclusivamente a este tema.
- **Propiedad:** La propiedad es una de las diferentes características que brinda el lenguaje CSS e iremos aprendiendo.
- **Valor:** Cada propiedad CSS tiene una serie de valores concretos, con los que tendrá uno u otro comportamiento.

Con todo esto le iremos indicando al navegador que, para cada etiqueta (selector especificado) debe aplicar las reglas (propiedad y valor) indicadas.

Supongamos que este es el código HTML:

```
<!DOCTYPE html>

<html>

  <head>

    <title>Título de página</title>

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

  </head>

  <body>

    <div id="first">

      <p>Párrafo</p>

    </div>

    <div id="second">

      <span>Capa</span>

    </div>

  </body>

</html>
```

Y además, por otro lado, este sería el código CSS del archivo index.css:

```
p {
  color: red; /* Color de texto rojo */
}
```

De esta forma, a todas las etiquetas <p> se le aplicará el estilo especificado: el color rojo.

Se pueden incluir comentarios entre los caracteres /* y */, los cuales serán ignorados por el navegador y pueden ser utilizados por legibilidad y para documentar nuestros documentos CSS.

Sin embargo, esto es sólo un ejemplo muy sencillo. Se pueden aplicar muchas más reglas (*no sólo el color del ejemplo*), consiguiendo así un conjunto de estilos para la etiqueta indicada en el selector. Cada una de estas reglas se terminará con el carácter punto y coma (;).

En el siguiente esquema se puede ver las diferentes partes del código CSS con sus respectivos nombres:



El último ; de un selector (*en naranja*) no es obligatorio y se puede omitir.

Un buen consejo, para hacer más legible nuestro código CSS, es utilizar la siguiente estructura visual (*indentar el código mediante espacios, con una propiedad por línea*). Es una buena práctica, indispensable a la larga, que nos facilitará la lectura del código:

```
selector {  
    propiedad : valor ;  
    propiedad : valor  
}
```

Esto mejora sustancialmente la legibilidad del código y se considera un convenio a utilizar para evitar la complejidad de entender el código que no se encuentre correctamente indentado.

Identificador de etiqueta (id)

En HTML, podemos darle un identificador a una etiqueta HTML y de esta forma darle un nombre. Simplemente, añadimos el atributo `id` y colocamos el nombre como valor de ese atributo. Ese nombre de identificador no debe empezar nunca por un número y puede contener

letras mayúsculas, minúsculas, signos especiales (*guión, guión bajo...*) o números:

```
<div id="pagina">  
  <div>Aquí irá un anuncio</div>  
  <div id="articulo">Aquí irá el contenido de texto del artículo</div>  
  <div>Aquí irá un anuncio</div>  
</div>
```

Los **id** se suelen indicar cuando queremos localizar zonas específicas que **sabemos que no se van a repetir en esa misma página**.

Uno de los detalles importantes de los **id**, es que no pueden haber dos con el mismo nombre en una página. Ejemplos correctos de **id** serían «pagina» (*para contener toda la página completa*), «comentarios» (*para contener toda la zona de comentarios de la página*) o «header» (*para contener la parte con el logo y la cabecera de la página*). Lo que debe quedar claro de los **ids**, es que **no se puede tener el mismo id** en varias etiquetas HTML en una misma página.

En CSS hacemos referencia al componente con dicho **id** de la siguiente forma:

```
#articulo{  
}
```

Clases de etiquetas (class)

Las clases funcionan de una forma muy similar a los **id**, pero son mucho más flexibles. En primer lugar, no tienen la limitación de los **ids**, por lo que su nombre se puede repetir y no es necesario que aparezca sólo una vez por página.

De hecho, la idea de las clases es establecer géneros o tipos de etiquetas, a los que les asociemos características comunes. Sigamos con el ejemplo anterior:


```
<div id="pagina">

  <div class="anuncio">Aquí irá un anuncio</div>

  <div id="articulo">Aquí irá el contenido de texto del artículo</div>

  <div class="anuncio">Aquí irá un anuncio</div>

</div>
```

Obsérvese que, al tener la misma clase «anuncio», podemos realizar acciones para todas las etiquetas de ese tipo y no tener que hacerlo para cada una de ellas por separado. Un ejemplo clásico donde se ve bien su utilidad, es respecto a utilizar [ids y clases para dar estilo CSS](#), donde aplicaremos unos estilos concretos a todas las etiquetas HTML con clase «anuncio».

Además, una etiqueta puede tener múltiples clases diferentes, no una sola. Esto nos da más flexibilidad a la hora de crear clases específicas:

```
<div id="pagina">

  <div class="anuncio primero">Aquí irá un anuncio</div>

  <div id="articulo">Aquí irá el contenido de texto del artículo</div>

  <div class="anuncio ultimo">Aquí irá un anuncio</div>

</div>
```

Nótese que en la primera etiqueta del anuncio hemos aplicado las clases **anuncio** y **primero**, mientras que en el último anuncio hemos aplicado las clases **anuncio** y **ultimo**. Esto nos permitiría asignar atributos comunes al anuncio en la clase anuncio, y atributos que sólo dependan de la posición donde está colocado en **primero** y/o **ultimo**.

Y el código CSS que modifica al componente con las clases anuncio y primero:

```
.anuncio {

}

.primero {

}
```

Prefijos

Algunas de las propiedades que veremos no están definidas por completo, sólo son borradores o pueden variar en la especificación definitiva, por lo que los navegadores las implementan utilizando una serie de **vendor prefixes** (*prefijos por navegador*), que facilitan la segmentación de funcionalidades.

De esta forma, podemos utilizar varios prefijos para asegurarnos que, aunque dichas funcionalidades tengan un comportamiento o sintaxis diferente en cada navegador, podemos hacer referencia a cada una de ellas por separado:

```
div {  
    transform: ... /* Navegadores que implementan especificación oficial */  
    -webkit-transform: ... /* Versiones antiguas de Chrome (Motor WebKit) */  
    -moz-transform: ... /* Versiones antiguas de Firefox (Motor Gecko) */  
    -ms-transform: ... /* Versiones antiguas de IE (Motor Trident) */  
    -o-transform: ... /* Versiones antiguas de Opera (Motor Presto) */  
}
```

En el ejemplo anterior, la propiedad transform se refiere a los navegadores que tengan implementada la especificación definitiva por completo e ignorará el resto de propiedades. Por otro lado, otro navegador (*o el mismo en una versión más antigua*) puede tener implementada una versión anterior a la definitiva, por lo que hará caso a las propiedades con un prefijo concreto.

Orden

En CSS, es posible crear múltiples reglas CSS para definir un mismo concepto. En este caso, la que prevalece ante todas las demás depende de ciertos factores, como es la «*altura*» a la que está colocada la regla:

- El CSS embebido en un elemento HTML es el que tiene mayor precedencia, por lo que siempre será el que tenga prioridad sobre otras reglas CSS.

- En segundo lugar, el CSS interno definido a través de bloque <style> en el propio documento HTML será el siguiente a tener en cuenta en orden de prioridad.
- Por último, los documentos CSS externos son la tercera opción de prioridad a la hora de tomar en cuenta las reglas CSS.

Guía de CSS

<https://lenguajecss.com/css/introduccion/guia-css/#>

Propiedades relativas al color

Uno de los primeros cambios de estilo que podemos pensar realizar en un documento HTML es hacer variaciones en los colores de primer plano y de fondo. Esto es posible con las primeras dos propiedades que veremos a continuación:

Propiedad	Valor	Significado
color		Cambia el color del texto que está en el interior de un elemento.
background-color		Cambia el color de fondo de un elemento.

La propiedad color establece el color del texto, mientras que la propiedad background-color establece el color de fondo del elemento.

Todas las propiedades CSS donde existen valores, establecen la posibilidad de indicar 4 formas alternativas (*con algunos derivados*) para especificar el color deseado:

Nombre	Formato	Ejemplo
Palabra clave predefinida	<i>[palabra clave]</i>	red
Esquema RGB	rgb(<i>rojo, verde, azul</i>)	rgb(255, 0, 0)
Esquema RGB con canal alfa	rgba(<i>rojo, verde, azul, alfa</i>)	rgba(255, 0, 0, 0.25)
Esquema RGB hexadecimal	# <i>RRGGBB</i>	#ff0000
Esquema RGB hexadecimal con canal alfa	# <i>RRGGBBAA</i>	#ff000040
Esquema HSL	hsl(<i>color, saturación, brillo</i>)	hsl(0, 100%, 100%)

Nombre	Formato	Ejemplo
Esquema HSL con canal alfa	<code>hsla(<u>color</u>, <u>saturación</u>, <u>brillo</u>, <u>alfa</u>)</code>	<code>hsla(0, 100%, 100%, 0.25)</code>

A continuación, iremos explicando cada uno de estos formatos para entender cómo se especifican los colores en CSS y utilizar el método que más se adapte a nuestras necesidades.

Si lo que buscamos es un sistema para extraer colores (*eye dropper*) de una página web, podemos utilizar la extensión para Chrome de ColorZilla o el propio Chrome Developer Tools, que integra dicha funcionalidad.

Palabras clave de color

El primer caso (*y más limitado*) permite establecer el color utilizando palabras reservadas de colores, como red, blue, orange, white, navy, yellow u otras. Existen más de 140 palabras clave para indicar colores:

black
navy
darkblue
mediumblue
blue
darkgreen
green
teal
darkcyan
deepskyblue
darkturquoise
mediumspringgreen
lime
springgreen
aqua
cyan
midnightblue
dodgerblue
lightseagreen
forestgreen

seagreen
darkslategray
darkslategrey
limegreen
mediumseagreen
turquoise
royalblue
steelblue
darkslateblue
mediumturquoise
indigo
darkolivegreen
cadetblue
cornflowerblue
mediumaquamarine
dimgray
dimgrey
slateblue
olivedrab
slategray
slategrey
lightslategray
lightslategrey
mediumslateblue
lawngreen
chartreuse
aquamarine
maroon
purple
olive
gray
grey
skyblue
lightskyblue
blueviolet
darkred
darkmagenta
saddlebrown
darkseagreen
lightgreen
mediumpurple
darkviolet
rebeccapurple
palegreen
darkorchid
yellowgreen
sienna
brown
darkgray
darkgrey
lightblue
greenyellow
paleturquoise
lightsteelblue
powderblue
firebrick
darkgoldenrod
mediumorchid
rosybrown
darkkhaki
silver
mediumvioletred
indianred
peru

chocolate
tan
lightgray
lightgrey
thistle
orchid
goldenrod
palevioletred
crimson
gainsboro
plum
burlywood
lightcyan
lavender
darksalmon
violet
palegoldenrod
lightcoral
khaki
aliceblue
honeydew
azure
sandybrown
wheat
beige
whitesmoke
mintcream
ghostwhite
salmon
antiquewhite
linen
lightgoldenrodyellow
oldlace
red
fuchsia
magenta
deeppink
orangered
tomato
hotpink
coral
darkorange
lightsalmon
orange
lightpink
pink
gold
peachpuff
navajowhite
moccasin
bisque
mistyrose
blanchedalmond
papayawhip
lavenderblush
seashell
cornsilk
lemonchiffon
floralwhite
snow
yellow
lightyellow
ivory
white

Además, existen algunos valores especiales que puedes utilizar cuando quieras especificar un color, como colores transparentes o el color actual del texto, muy útil para SVG, por ejemplo:

Valor	Significado
transparent	Establece un color completamente transparente (valor por defecto de background-color)
currentColor	Establece el mismo color que se está utilizando para el texto (CSS3 y SVG)


Veamos algunos ejemplos de palabras clave de color:


```
div {  
  background-color: blue;  
  background-color: transparent;  
  background-color: lightpink;  
  background-color: rebeccapurple;  
}
```


Formato RGB


Uno de los métodos más conocidos por los diseñadores gráficos es utilizar el formato RGB. Las siglas RGB significan rojo, verde y azul, por lo que cada cifra (*de 0 a 255*) representa la intensidad de cada componente de color. Como se puede ver en la siguiente imagen, si utilizamos una cantidad (0, 0, 0) de cada canal, obtenemos el color negro. En cambio, si utilizamos una cantidad (255, 0, 0), obtendremos el color rojo.

Rojo	0	255	255	0
Verde	0	255	0	0
Azul	0	255	0	255

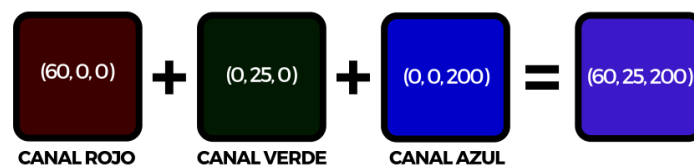
||


||


||


||


De esta forma, mezclando las cantidades de cada canal, se puede obtener prácticamente cualquier color. Existen muchos esquemas de colores, pero en diseño web nos interesa particularmente el esquema RGB (*junto al HSL*).



La mayoría de los editores tienen los denominados ColorPicker, que no son más que un sistema cómodo y rápido para elegir un color a base de clics por una paleta o círculo visual. También podemos hacerlo directamente en buscadores como Duck Duck Go o Google.

Veamos algunos ejemplos de colores en formato RGB:

```
div {
  background-color: rgb(125, 80, 10);
  background-color: rgb(0, 0, 34);
  color: rgb(255, 255, 0);
}
```

Formato hexadecimal

El formato hexadecimal es el más utilizado por los desarrolladores web, aunque en principio puede parecer algo extraño y complicado, sobre todo si no has oído hablar nunca del sistema hexadecimal (*sistema en base 16 en lugar del que utilizamos normalmente, en base 10*).

Cada par de letras simboliza el valor del RGB en el sistema de numeración hexadecimal, así pues, el color #FF0000, o sea HEX(FF,00,00), es equivalente al RGB(255,0,0), que es también equivalente al HSL(0, 100%, 100%). Veamos algunos ejemplos para clarificarlo:

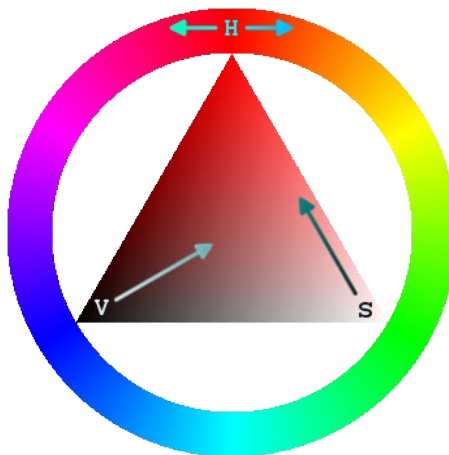
Hexadecimal	Hex. abreviado	Color RGB	Palabra clave
#FF0000	#F00	255,0,0	red (rojo)
#000000	#000	0,0,0	black (negro)
#00FFFF	#0FF	0, 255, 255	cyan (azul claro)
#9370DB	#97D	147,112,219	mediumpurple (lila)

Veamos algunos ejemplos del formato hexadecimal (RGB abreviado):

```
div {
  background-color: #512592;
  background-color: #000000;
  background-color: #451;    /* Equivalente a #445511; */
}
```

Formato HSL

Las siglas HSL significan color (*o matiz*), saturación y brillo. La primera cifra selecciona el matiz de color (*una cifra de 0 a 360 grados*), seleccionando el color del círculo exterior de la imagen. Por su parte, las dos siguientes, son el porcentaje de saturación y el brillo del color, respectivamente (*ambos, porcentajes de 0% a 100%*).



Veamos algunos ejemplos del formato HSL:

```
div {
  background-color: hsl(35deg, 0%, 100%);
  background-color: hsl(120deg, 25%, 75%);
  background-color: hsl(5deg, 20%, 20%);
}
```

Canales Alfa

Es posible que deseemos indicar un color que tenga cierto grado de transparencia, y de esta forma, refleje el contenido, color o imágenes que se encuentren detrás. Hasta ahora solo conocemos la palabra clave `transparent`, que es un color de transparencia total (*totalmente transparente*).

Sin embargo, existe la posibilidad de utilizar los denominados canales alfa, que permiten establecer una transparencia parcial en determinados colores. Estos se pueden establecer en cualquier formato, salvo en los colores con palabras clave. Vamos a ver cómo hacerlo en cada caso:

- **Formato RGB:** En lugar de `rgb()` indicamos `rgba()` para establecer que usaremos un canal alfa. Posteriormente, en lugar de establecer 3 parámetros (*rojo, verde, azul*), añadiremos uno más, que será el canal alfa. Dicho canal alfa será un valor (*del 0 al 1 con decimales*) o un porcentaje (*del 0% al 100%*).
- **Formato HSL:** Prácticamente idéntico al anterior. En lugar de `hsl()` indicamos `hsla()`. Añadimos un nuevo valor como canal alfa (*valor o porcentaje*).
- **Formato Hexadecimal:** Es posible indicar (*al final*) un par adicional que indique el grado de transparencia. Por ejemplo, el color `#FF0000` reescrito como `#FF000077` se trataría de dicho color, con un grado de transparencia casi del 50% (00 es 0%, 80 es 50%, FF es 100%).

Veamos algunos ejemplos de cada caso:

```
div {  
  background-color: rgba(0, 0, 0, 0.5);  
  background-color: rgba(0, 0, 0, 50%);  
  background-color: hsla(180deg, 50%, 25%, 0.75);  
  background-color: hsla(180deg, 50%, 25%, 75%);  
  background-color: #aa44ba80;  
}
```

El formato de transparencia en formato hexadecimal se encuentra actualmente bien soportado, pero puede no ser compatible en versiones más antiguas u otros navegadores.

Las tipografías (*también denominadas fuentes*) son una parte muy importante del mundo de CSS. De hecho, son uno de los pilares del diseño web. La elección de una tipografía adecuada, su tamaño, color, espacio entre letras, interlineado y otras características pueden variar mucho, de forma consciente o inconsciente, la percepción en la que una persona interpreta o accede a los contenidos de una página.

Detalles de una tipografía

Existen multitud de características en las tipografías que convendría conocer antes de continuar, por lo que vamos a ver algunas de ellas:

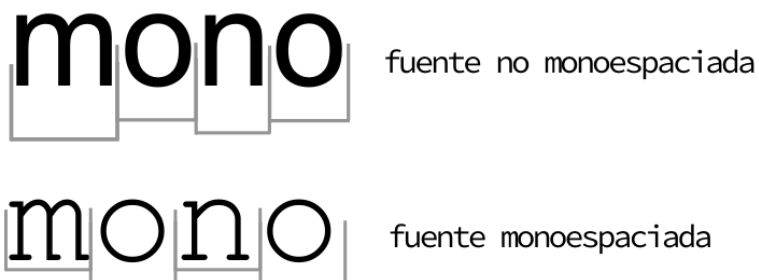


- **Serifa:** Las fuentes o tipografías que utilizan serifa o gracia, son aquellas que incorporan unos pequeños adornos o remates en los extremos de los bordes de las letras. Muchas de estas tipografías suelen terminar su nombre en «Serif» (*con serifa*).
- **Paloseco:** Las fuentes o tipografías de paloseco son las opuestas a la anterior: unas tipografías lisas, sin adornos o remates en los extremos de los bordes de las letras. Muchas de estas tipografías suelen terminar su nombre en «Sans Serif» (*sin serifa*).

Tradicionalmente, se han utilizado tipografías con serifa en medios impresos argumentando que dichos bordes ofrecen una mayor legibilidad que las tipografías de paloseco, ya que ayudan a reconocer más rápidamente las letras. En medios digitales, las tipografías de paloseco suelen ser más comunes puesto que dan un aspecto más limpio y ayudan a que se canse menos la vista del usuario. No obstante,

todo esto puede ser muy subjetivo y está sujeto a diferentes interpretaciones.

- Monoespaciada: Por otro lado, existe un tipo de tipografía denominada fuente monoespaciada, que se basa en que cada una de sus letras tienen exactamente el mismo ancho. Son muy útiles para tareas de programación o emuladores de terminal, donde se leen mejor líneas con estas características, ya que no queremos que una línea sea más corta dependiendo de su contenido.



Propiedades básicas

Existe un amplio abanico de propiedades CSS para modificar las características básicas de las tipografías a utilizar. Aunque existen muchas más, a continuación, veremos las propiedades CSS más básicas para aplicar a cualquier tipo de tipografía:

Propiedad	Valor	Significado
font-family	<u>fuente</u>	Indica el nombre de la fuente (tipografía) a utilizar.
font-size		Indica el tamaño de la fuente.

Propiedad	Valor	Significado
font-style	normal italic oblique	Indica el estilo de la fuente.
font-weight	<u>peso</u>	Indica el peso (grosor) de la fuente (100-800).

Con ellas podemos seleccionar tipografías concretas, especificar su tamaño, estilo o grosor.

Familia tipográfica

Empezaremos por la más lógica, la propiedad CSS para seleccionar una familia tipográfica concreta. Con esta propiedad, denominada font-family , podemos seleccionar seleccionar cualquier tipografía simplemente escribiendo su nombre.

Si dicho nombre está compuesto por varias palabras separadas por un espacio, se aconseja utilizar comillas simples para indicarla (*como se ve en el segundo ejemplo*):

```
body {
  font-family: Verdana;
  font-family: 'PT Sans'; /* Otro ejemplo */
}
```

Esta es la forma más básica de indicar una tipografía. Sin embargo, hay que tener en cuenta un detalle muy importante: estas fuentes sólo se visualizarán si el usuario las tiene instaladas en su sistema o

dispositivo. En caso contrario, se observarán los textos con otra tipografía «suplente» que esté disponible en el sistema, pero que puede ser visualmente muy diferente.

Esto convierte una tarea a priori simple, en algo muy complejo, puesto que los sistemas operativos (*Windows, Mac, GNU/Linux*) tienen diferentes tipografías instaladas. Si además entramos en temas de licencias y tipografías propietarias, la cosa se vuelve aún más compleja.

Un primer y sencillo paso para paliar (*en parte*) este problema, es añadir varias tipografías alternativas, separadas por comas, lo que además se considera una buena práctica de CSS:

```
div {  
    font-family: Vegur, 'PT Sans', Verdana, sans-serif;  
}
```

De esta forma, el navegador busca la fuente Vegur en nuestro sistema, y en el caso de no estar instalada, pasa a buscar la siguiente (*PT Sans*), y así sucesivamente. Se recomienda especificar al menos 2 ó 3 tipografías diferentes.

Tamaño de la tipografía

Otra de las propiedades más utilizadas con las tipografías es `font-size`, una tipografía que permite especificar el tamaño que tendrá la fuente que vamos a utilizar:

Propiedad	Valor	Tipo de medida
font-size	xx-small x-small small medium large x-large xx-large	Absoluta (tamaño predefinido)

Propiedad	Valor	Tipo de medida
font-size	smaller larger	Relativa (más pequeña/más grande)
font-size		Específica (tamaño exacto)

Se pueden indicar tres tipos de valores:

- **Medidas absolutas:** Palabras clave como medium que representan un tamaño medio (*por defecto*), small: tamaño pequeño, x-small: tamaño muy pequeño, etc...
- **Medidas relativas:** Palabras clave como smaller que representan un tamaño un poco más pequeño que el actual, o larger que representa un tamaño un poco más grande que el actual.
- **Medida específica:** Simplemente, indicar píxeles, porcentajes u otra unidad para especificar el tamaño concreto de la tipografía. Para tipografías se recomienda empezar por píxeles (*más fácil*) o utilizar estrategias con unidades rem (*mejor, pero más avanzado*).

Estilo de la tipografía

A las tipografías elegidas se les puede aplicar ciertos estilos, muy útil para maquetar los textos, como por ejemplo negrita o cursiva (*italic*). La propiedad que utilizamos es font-style y puede tomar los siguientes valores:

Valor	Significado
normal	Estilo normal, por defecto. Sin cambios aparentes.
italic	Cursiva. Estilo caracterizado por una ligera inclinación de las letras hacia la derecha.
oblique	Oblíqua. Idem al anterior, salvo que esta inclinación se realiza de forma artificial.

Con la propiedad font-style podemos aplicarle estos estilos. En la mayoría de los casos, se aprecia el mismo efecto con los valores italic y oblique , no obstante, italic muestra la versión cursiva de la fuente, específicamente creada por el diseñador de la tipografía, mientras que oblique es una representación forzosa artificial de una tipografía cursiva.

Peso de la tipografía

Por otro lado, tenemos el peso de la fuente, que no es más que el grosor de la misma. También depende de la fuente elegida, ya que no todas soportan todos los tipos de grosor. De forma similar a como hemos visto hasta ahora, se puede especificar el peso de una fuente mediante tres formas diferentes:

Propiedad	Valor	Significado
font-weight	normal bold	Medidas absolutas (predefinidas)
font-weight	bolder lighter	Medidas relativas (dependen de la actual)
font-weight	<u>peso</u>	Medida específica (número del peso concreto)



- Valores absolutos: Palabras claves para indicar el peso de la fuente: *normal* y *bold*. Normal es el valor por defecto.
- Valores relativos: *Bolder* (más gruesa) o *Lighter* (más delgada).
- Valor numérico: Un número del 100 (menos gruesa) al 900 (mas gruesa). Generalmente, se incrementan en valores de 100 en 100.

Ten en cuenta que los diferentes pesos de una tipografía son diseñados por el creador de la tipografía. Algunas tipografías carecen de diferentes pesos y sólo tienen uno específico. Esto es algo muy sencillo de ver en Google Fonts (*al seleccionar una tipografía*).

Antiguamente, utilizar tipografías en CSS tenía una gran limitación. Usando la propiedad font-family y especificando el nombre de la tipografía a utilizar, en principio deberían visualizarse. Pero fundamentalmente, existían dos problemas:

Las tipografías especificadas mediante font-family debían estar instaladas en el sistema donde se visualiza la página web.

```
p {  
    font-family: Vegur, Georgia, "Times New Roman", sans-serif;  
}
```

En el ejemplo superior, se han indicado las tipografías Vegur (*tipografía personalizada*), Georgia y Times New Roman (*tipografías de Microsoft Windows*) y la categoría segura sans-serif. Un usuario con la tipografía Vegur instalada, vería sin problema el diseño con dicha tipografía, mientras que un usuario de Windows la vería con Georgia (*o si no la tiene, con Times New Roman*), mientras que un usuario de Linux o Mac, la vería con otra tipografía diferente (*una tipografía sans-serif del sistema*).

Mientras que las tipografías que vienen en sistemas como Microsoft Windows de serie (*Times New Roman, Verdana, Tahoma, Trebuchet MS...*) se verían correctamente en navegadores con dicho sistema operativo, no ocurriría lo mismo en dispositivos con GNU/Linux o Mac. Y lo mismo con tablets o dispositivos móviles, o viceversa. Esto ocurre porque muchas tipografías son propietarias y tienen licencias que permiten usarse sólo en dispositivos de dicha compañía.

En definitiva, aunque teníamos los mecanismos, vivimos en un mundo complicado en el que no es tan sencillo establecer una fuente específica para obtener el mismo resultado de diseño en todos los navegadores y sistemas disponibles... al menos hasta que llegó @font-face.

La regla @font-face

La regla @font-face permite descargar una fuente o tipografía, cargarla en el navegador y utilizarla en nuestras páginas. Todo ello de forma transparente al usuario sin que deba instalar o realizar ninguna acción.

Veamos un ejemplo de cómo se puede utilizar:

```
@font-face {
  font-family: "gothic";
  font-style: normal;
  font-weight: normal;
  src: url("../fonts/gothic.otf");
}

h1{
  font-family: "gothic";
}
```

Sitio web para descargar fuentes: <https://www.fontsquirrel.com>

La regla `@font-face` suele colocarse al principio del fichero CSS para preparar el navegador para descargar la tipografía en el caso de no disponer de ella. En el ejemplo superior lo hemos hecho con la fuente Open Sans, una tipografía libre creada por Steve Matteson para Google y disponible en Google Fonts.

Basicamente, abrimos un bloque `@font-face`, establecemos su nombre mediante `font-family` y definimos sus características mediante propiedades como `font-style` o `font-weight`. El factor clave viene a la hora de indicar la tipografía, que se hace mediante la propiedad `src` (*source*) con los siguientes valores:

Valor	Significado	Soporte
<code>local('Nombre')</code>	¿Está la fuente 'Nombre' instalada? Si es así, no hace falta descargarla.	Todos
<code>url(file.woff2)</code>	Formato <u>Web Open Font Format 2</u> . Mejora de WOFF con <u>Brotli</u> .	<u>No IE</u>

Valor	Significado	Soporte
url(file.woff)	Formato <u>Web Open Font Format</u> . Es un TTF comprimido, ideal para web.	<u>Bueno</u>
url(file.ttf)	Formato <u>True Type</u> . Uno de los formatos más conocidos.	<u>Bueno</u>
url(file.otf)	Formato <u>Open Type</u> . Mejora del formato TTF .	<u>Bueno</u>
url(file.eot)	Formato <u>Embedded OpenType</u> . Mejora de OTF , propietaria de Microsoft.	<u>Sólo IE</u>
url(file.svg)	Tipografías creadas como formas SVG. No usar, considerada obsoleta.	<u>Malo</u>

Google Fonts

En la actualidad, es muy común utilizar Google Fonts como repositorio proveedor de tipografías para utilizar en nuestros sitios web por varias razones:

- Gratuitas: Disponen de un amplio catálogo de fuentes y tipografías libres y/o gratuitas.

- Cómodo: Resulta muy sencillo su uso: Google nos proporciona un código y el resto lo hace él.
- Rápido: El servicio está muy extendido y utiliza un CDN, que brinda ventajas de velocidad.

En la propia página de Google Fonts podemos seleccionar las fuentes con las características deseadas y generar un código HTML con la tipografía (*o colección de tipografías*) que vamos a utilizar.

The screenshot shows the Google Fonts interface for the 'Open Sans' font family. The main content area is titled 'Open Sans' and 'Designed by Steve Matteson'. It features a 'Download family' link and tabs for 'Select styles', 'Glyphs', 'About', and 'License'. The 'Select styles' tab is active, showing three font styles: 'Light 300', 'Light 300 italic', and 'Regular 400'. Each style has a preview of the text 'Almost before we knew it, we had left the gro...' and a 'Select this style' button. A 'Size: 30px' dropdown and a slider are also visible. On the right, a 'Selected family' sidebar shows the 'Embed' tab selected, displaying the HTML code to embed the font and the CSS rules to specify the font families.

Todo esto nos generará el siguiente código, que aparece en la zona derecha de la web (*en la zona «embed»*), y que será el fragmento de código que tendremos que insertar en nuestro documento HTML, concretamente, antes de finalizar la sección <head>:

```
<link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400&display=swap">
```

Cómo se puede ver el ejemplo anterior, al añadir este código estamos enlazando nuestro documento HTML con un documento CSS del repositorio de Google, que incluye los @font-face correspondientes. Esto hará que incluyamos automáticamente todo ese código CSS necesario para las tipografías escogidas, en este caso la tipografía Open Sans con los pesos 300 y 400.

Si además, añadimos también la familia de tipografías Roboto (*con grosor 400*) y Lato (*con grosor 300 y 400*), el código necesario sería el siguiente:

```
<link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Lato:wght@300;400&family=Open+Sans:wght@300;400;600&family=Roboto&display=swap">
```

De esta forma conseguimos cargar varias tipografías desde el repositorio de Google de una sola vez, sin la necesidad de varias líneas de código diferentes, que realizarían varias peticiones diferentes a Google Fonts.

Nota que, en este nuevo ejemplo, en caso de no tener instaladas ningunas de las tipografías anteriores, estaríamos realizando 6 descargas: (*el css de Google Fonts*), (*las 2 tipografías con los diferentes pesos de Open Sans*), (*la de Roboto*), (*y las 2 tipografías con los diferentes pesos de Lato*).

Por último y, para terminar, sólo necesitaremos añadir la propiedad font-family: "Open Sans", font-family: "Lato" o font-family: "Roboto" a los textos que queramos dar formato con dichas tipografías. No te olvides de añadir tipografías alternativas y fuente segura para mejorar la compatibilidad con navegadores antiguos.

Atajo para tipografías

Finalmente, algunas de las propiedades más utilizadas de tipografías y fuentes se pueden resumir en una propiedad de atajo, como viene siendo habitual. El esquema es el siguiente:

```
div {
    font: <style> <variant> <weight> <size/line-height> <family>;
}
```

Por ejemplo, utilizar la tipografía Arial , con la fuente alternativa Verdana o una fuente segura sin serifa, a un tamaño de 16 píxeles, con un interlineado de 22 píxeles, un peso de 400, sin utilizar versalitas y con estilo cursiva:


```
div {  
    font: italic normal 400 16px/22px Arial, Verdana, Sans-serif;  
}
```

<https://color.adobe.com/>

<https://htmlcolorcodes.com/es/>

<https://flukeout.github.io/>