

Introducción a CSS

CSS es el segundo lenguaje más básico y esencial para crear páginas web.

1. El primero sería HTML, con el que se define el contenido de la página.
2. El segundo CSS, con el que se define la parte de la presentación (estilo), es decir, cómo deben mostrarse los elementos de la página, su posición, forma, espaciados, colores y en resumen, toda la parte estética.

CSS (siglas en inglés de *Cascading Style Sheets*) es un lenguaje que consiste en una serie de elementos mediante los cuales se declaran los estilos, básicamente éstos son los más importantes:

- Selectores, mediante los cuales podemos especificar a qué elementos de la página nos estamos refiriendo
- Atributos de estilo para definir qué cosas queremos estilizar sobre los selectores indicados
- Una serie de valores, que indican qué estilo se debe aplicar a cada atributo sobre cada selector.

Los valores se expresan con unidades CSS, que sirven para cuantificar los valores (píxeles, puntos, etc.).

Aprender CSS no es difícil, pero cuando se usa profesionalmente se deben tener en cuenta muchos detalles y buenas prácticas, como la organización del código, la reutilización, la optimización, etc.}

Separar el código CSS del código HTML

El enfoque de CSS es servir para definir la capa de presentación, es decir, la parte relacionada con el aspecto. Es algo que cualquier estudiante suele tener claro cuando está aprendiendo CSS, ya que al enseñar HTML probablemente se haya insistido, pero que siempre conviene reforzar.

En el código HTML colocamos el contenido, es decir, qué debe visualizarse, mientras que con CSS definimos la presentación, es decir, cómo debe visualizarse. Esto nos lleva a una serie de usos de CSS que debemos de respetar como buenas prácticas.

- Lo adecuado cuando trabajamos con CSS es escribir el código en ficheros independientes, que tendrán extensión .css.
- No conviene colocar código CSS por tanto dentro de archivos HTML. Debemos evitar colocar estilos en etiquetas <style> dentro del propio código HTML.
- Por supuesto, mucho menos aconsejable es colocar estilos en los atributos "style" de las etiquetas HTML.

La aparición de CSS 3 sólo se materializó en el año 2014 con el movimiento de HTML 5. Vino a aportar soluciones a la mayoría de las necesidades de los diseñadores y a permitir finalmente cubrir el objetivo principal del lenguaje, la separación del contenido de la presentación. No obstante cabe decir que CSS 3 se presentó por medio de un nutrido grupo de especificaciones, que han sido mejoradas con el transcurso del tiempo, por lo que no es tanto un lanzamiento puntual, sino una continua mejora del estándar a diversos niveles.

Usos de las CSS

Hemos denominado a este apartado los diferentes usos de las CSS y relata justamente eso, los distintos niveles a los que podemos usar las Hojas de Estilo, que van desde definir los estilos de manera específica, para un pequeño fragmento de una página, hasta los estilos para todo un sitio web completo.

Para definir estilos en secciones reducidas de una página se puede utilizar el atributo style en la etiqueta sobre la que queremos aplicar estilos.

Como valor de ese atributo indicamos en sintaxis CSS las características de estilos.

Lo vemos con un ejemplo, pondremos un párrafo en el que determinadas palabras las vamos a visualizar en color verde.

<p>

Esto es un párrafo en varias palabras en color verde. resulta muy fácil.

</p>

Nota: la etiqueta `` del HTML quizás no sea tan conocida como otras. Es una etiqueta que, por sí sola, no tiene ninguna representación en la página. Es muy habitual usarla justamente para lo que hemos hecho en el anterior ejemplo, separar partes del contenido de texto de una etiqueta, para aplicar estilos determinados a esa parte de dentro de la etiqueta.

Estilo definido para una etiqueta

De este modo podemos hacer que toda una etiqueta muestre un estilo determinado. Por ejemplo, podemos definir un párrafo entero en color rojo y otro en color azul. Para ello utilizamos el atributo `style`, que es admitido por todas las etiquetas del HTML.

Nota: este uso de las CSS podríamos decir que es en realidad el mismo que el anterior. Solo que la etiqueta es de bloque y no una etiqueta inline (en línea) como ``.

```
<p style="color:#990000">
```

Esto es un párrafo de color rojo.

```
</p>
```

```
<p style="color:#000099">
```

Esto es un párrafo de color azul.

```
</p>
```

Estilo definido en una parte de la página

Con la etiqueta <div> podemos definir secciones de una página y aplicarle estilos con el atributo style, es decir, podemos definir estilos de una vez a todo un bloque de la página.

El uso de la etiqueta div es englobar partes de un documento HTML para que podamos aplicar estilos a todo el grupo de etiquetas, como el posicionamiento, color, borde, tamaño de letra, etc.

```
<div style="color:#000099; font-weight:bold">
```

```
<h3>Estas etiquetas van en <i>azul y cursiva</i></h3>
```

```
<p>
```

Seguimos dentro del DIV, luego permanecen los estilos

```
</p>
```

```
</div>
```

Hasta aquí hemos visto los usos de las CSS más específicas, que se consiguen usando el atributo style en la etiqueta. Realmente todos los usos anteriores eran el mismo, pero el enfoque era distinto ya que las etiquetas del HTML que hemos usado tienen distintos alcances. Sin embargo, hay otras formas más avanzadas de usar las CSS, que deberías tener en cuenta porque son todavía más versátiles y recomendadas.

Estilo definido para toda una página

Podemos definir, en la cabecera del documento, estilos para que sean aplicados a toda la página. Es una manera muy cómoda de darle forma al documento y muy potente, ya que estos estilos serán seguidos en toda la página y nos ahorraremos así, generar mayor cantidad de etiquetas HTML colocando el atributo style.

Es común que los estilos declarados se quieran aplicar a distintas etiquetas dentro del mismo documento.

La aplicación de estilos para toda la página, se utiliza para escribir los estilos una vez y usarlos en un número indefinido de etiquetas. Por ejemplo podremos definir el estilo a todos los párrafos una vez y que se aplique igualmente, sea cual sea el número de párrafos del documento. Por último, también tendremos la ventaja que, si más adelante deseamos cambiar los estilos de todas las etiquetas, lo haremos de una sola vez, ya que el estilo fue definido una única vez de manera global.

Estilo definido para todo un sitio web

Una de las características más potentes del desarrollo con hojas de estilos es la posibilidad de definir los estilos de todo un sitio web en una única declaración.

Esto se consigue creando un archivo donde tan sólo colocamos las declaraciones de estilos de la página y enlazando todas las páginas del sitio con ese archivo. De este modo, todas las páginas comparten una misma declaración de estilos, reutilizando el código CSS de una manera mucho más potente.

Este es el modelo más ventajoso de aplicar estilos al documento HTML y por lo tanto el más recomendable. De hecho, cualquier otro modo de definir estilos no es considerado una buena práctica y lo tenemos que evitar siempre que se pueda.

Las ventajas de este modelo de definición de estilos son las siguientes:

- Se ahorra en líneas de código HTML, ya que no tenemos que escribir el CSS en la propia página (lo que reduce el peso del documento y mejora la velocidad de descarga).
- Se mantiene separado correctamente lo que es el contenido (HTML) de la presentación (CSS), que es uno de los objetivos de las hojas de estilo.
- Se evita la molestia de definir una y otra vez los estilos con el HTML y lo que es más importante, si cambiamos la declaración de estilos, cambiarán automáticamente todas las páginas del sitio web.

Esto es una característica muy deseable, porque aumenta considerablemente la facilidad de mantenimiento del sitio web. Si en cualquier momento se desea cambiar el contenido, no tenemos que preocuparnos por los estilos y viceversa, si queremos cambiar el aspecto del sitio web, no necesitamos preocuparnos ni andar editando el contenido.

Incluir estilos con un fichero externo

1- Creamos el fichero con la declaración de estilos

Es un fichero de texto normal, que puede tener cualquier extensión, aunque solemos asignarle la extensión .css para aclararnos qué tipo de archivo es. El texto que debemos incluir debe ser escrito exclusivamente en sintaxis CSS, es decir, sería erróneo incluir código HTML en él. Podemos ver un ejemplo a continuación.

```
P {
```

```
font-size : 12pt;
```

```
font-family : arial, helvetica;
```

```
font-weight : normal;
```

```
}
```

```
H1 {
```

```
font-size : 36pt;
```




<codoa
codoo/>

font-family : verdana,arial; text-

decoration : underline; text-align

: center; background-color :

Teal;

}

TD {

font-size : 10pt;

font-family : verdana,arial; text-

align : center; background-color :

666666;

}

BODY {

background-color : #006600; font-

family : arial;

color : White;

}

Agencia de
Aprendizaje
a lo largo
de la vida

2- Enlazamos la página web con la hoja de estilos

Para ello, vamos a colocar la etiqueta <link> con los atributos siguientes:

- rel="STYLESheet" indicando que el enlace es con una hoja de estilos
- href="estilos.css" indica el nombre del fichero fuente de los estilos Veamos una

página web entera que enlaza con la declaración de estilos anterior:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<link rel="STYLESheet" href="estilos.css">
```

```
<title>Página con estilos.css incorporado</title>
```

```
</head>
```

```
<body>
```

```
<h1>página con estilos.css</h1>
```

Esta página tiene un enlace a estilos.css

```
<br>
```

```
<br>
```

```
</body>
```

```
</html>
```


Reglas de importancia en los estilos

Los estilos se heredan de una etiqueta a otra, como se indicó anteriormente. Por ejemplo, si tenemos declarado en el `<BODY>` unos estilos, en muchos casos, estas declaraciones también afectarán a etiquetas que estén dentro de esta etiqueta.

La herencia de estilos desde padres a hijos no ocurre siempre. Depende del estilo utilizado.

Por ejemplo, el color del texto se hereda en todos los componentes, aunque no ocurre con el borde de un elemento.

Pero las declaraciones de estilos se pueden realizar de múltiples modos y con varias etiquetas, también entre estos modos hay una jerarquía de importancia para resolver conflictos entre varias declaraciones de estilos distintas para una misma porción de página. Se puede ver a continuación esta jerarquía, primero ponemos las formas de declaración más generales, y por tanto menos respetadas en caso de conflicto:

- Declaración de estilos con fichero externo (para todo un sitio web)
- Declaración de estilos para toda la página. (con la etiqueta `<STYLE>` en la cabecera de la página)
- Definidos en una etiqueta en concreto (utilizando el atributo `style` en la etiqueta en cuestión)

Para definir un estilo se utilizan atributos como `font-size`, `text-decoration`, etc. seguidos de dos puntos y el valor que le deseemos asignar. Podemos definir un estilo a base de definir muchos atributos separados por punto y coma.

Ejemplo:

`font-size: 10pt; text-decoration: underline; color: black;`

(el último punto y coma de la lista de atributos es opcional)

Para definir el estilo de una etiqueta se escribe la etiqueta seguida de la lista de atributos encerrados entre llaves.

Ejemplo:

```
H1{text-align: center; color:black}
```

Selectores CSS

Los selectores en CSS nos permiten acceder a cualquier elemento o grupo de elementos, para aplicar estilos en una única declaración y como su nombre indica, permiten seleccionar aquellos elementos sobre los que se van a aplicar las reglas de estilo.

Dentro del código CSS podemos usar selectores y aplicarles un conjunto de estilos determinado.

Para ello escribimos el selector y colocamos los atributos de estilos encerrados entre llaves:

```
selector { color:  
  red; width:  
  auto;  
}
```

Existen selectores de lo más variados, que permiten ajustar de una manera muy precisa qué elementos se desea seleccionar.

Los más importantes son los siguientes:

Etiqueta: sirven para seleccionar todos los elementos de una misma etiqueta o tag HTML. h1 {

```
  font-size: 26px;
```

```
}
```


Clase: selecciona todos los elementos de una clase determinada. (Class de CSS).

```
.destacado {  
  font-weight: bold;  
  color: orange;  
}
```

Identificador: permiten seleccionar etiquetas individuales por el atributo Id de la etiqueta.

```
#mifomulario {  
  border: 1px solid #99c;  
}
```

Atributo: permiten seleccionar todas las etiquetas que tengan un atributo dado, o bien un atributo con un valor determinado.

```
[title] {  
  text-decoration: none;  
}
```

```
[align="center"] { border:  
  1px solid red;
```

}

Además, los selectores se pueden combinar entre sí para conseguir selectores mucho más precisos:

Estos selectores obtienen las imágenes que tengan el atributo alt y los párrafos que tengan la clase "desactivado".

```
img[alt] {  
    border: none;  
}
```

```
p.desactivado {  
    color: #ddd;  
}
```

También podemos relacionar los selectores con un espacio y entonces el significado cambia totalmente, ya que se estaría indicando que un elemento tiene que estar dentro de otro.

Este selector aplicaría estilos a todos los elementos <h2> que estén dentro de contenedores que tienen la clase "nota".


```
.nota h2 {
```

```
  font-weight: normal;
```

```
}
```

También podemos combinar los selectores de CSS usando una coma. Entonces estamos indicando que los atributos de estilo deben aplicarse a los dos selectores por separado.

Así estaríamos indicando que queremos aplicar estilos sobre todos los párrafos y todas las divisiones con la clase "bloque".

```
p, div.bloque {
```

```
  margin-bottom: 25px;
```

```
}
```

CSS sirve para definir el aspecto de las páginas web, eso ya debe haber quedado claro. No obstante, hay diferentes niveles a los que podemos aplicar los estilos y es algo que vamos a describir a continuación.

Atributos Globales

Los atributos globales son atributos comunes a todos los elementos HTML y pueden usarse en todos los elementos, aunque pueden no tener efecto en algunos de ellos.

Los atributos globales pueden especificarse en todos los elementos HTML, e incluso en aquellos no especificados en el estándar. Esto significa que cualquier elemento no estándar también debe permitir estos atributos, aún cuando el uso de tales elementos significa que el documento ya no cumple con HTML5.

Por ejemplo, los navegadores que cumplen HTML5 esconden contenido que se marque como `<foo hidden>...</foo>`, aún cuando `<foo>` no es un elemento HTML válido.

Descripción

A continuación enumeramos atributos que se utilizan en HTML y que reciben o interactúan con CSS para cumplir con la norma de maquetado actual.

accesskey

Proporciona y genera un acceso de teclado para el elemento actual. Este atributo consiste de una lista de caracteres, separadas por espacios. El navegador debe utilizar el primero que exista en la distribución del teclado del ordenador.

El valor del atributo debe constar de un solo carácter imprimible (que incluye caracteres acentuados y otros que pueden ser generados por el teclado).

class

Es una lista de clases del elemento, separadas por espacios. Las clases permiten a CSS y JavaScript seleccionar y acceder a elementos específicos a través de selectores de clase o funciones, como el método `Document.getElementsByClassName()`, que veremos en detalle y usaremos más adelante.

HTML

<p>Periodista: Gracias por la entrevista.</p>

<p class="nota editorial">Esta es una nota realizada en 2019.</p>

<p>Periodista: Como fueron sus comienzos?</p>

<p class="nota">[Podés continuar leyendo esta nota en la pag.20]</p>

CSS

```
.output {  
  font: 1rem 'Fira Sans', sans-serif;  
}
```

```
.nota {  
  font-style: italic;  
  font-weight: bold;  
}
```

```
.editorial {  
  background: rgb(255, 0, 0, .25);  
  padding: 10px;  
}
```

```
.editorial:before {  
  content: 'Editor: ';  
}
```

contenteditable

Es un atributo enumerado que indica si el elemento puede ser modificable por el usuario. Si es así, el navegador modifica el elemento para permitir la edición. El atributo debe tener uno de los siguientes valores:

true o un valor vacío, el cual indica que el elemento debe ser editable.
false, el cual indica que el elemento no debe ser editable.

HTML

```
<blockquote contenteditable="true">  
  <p>Edite el contenido de este texto</p>  
</blockquote>
```

```
<cite contenteditable="true">-- Puede escribir algo aquí</cite>
```

CSS


```
.output {  
  font: 1rem 'Fira Sans', sans-serif;  
}  
  
blockquote {  
  background: #eee;  
  border-radius: 5px;  
  margin: 16px 0;  
}  
  
blockquote p {  
  padding: 15px;  
}  
  
cite {  
  margin: 16px 32px;  
}  
  
blockquote p::before {  
  content: '\201C';  
}  
  
blockquote p::after {  
  content: '\201D';  
}  
}
```

```
[contenteditable='true'] {  
  caret-color: red;  
}
```

Nota: content: '\201C'; y content: '\201D'; se refiere a las dobles comillas.

data-*

Son atributos globales que forman una clase denominados atributos de datos personalizados , y que se caracterizan por permitir intercambiar información entre el HTML en el momento de la carga del archivo HTML.

HTML

```
<h1>Socios del Club</h1>
```

```
<ul>
```

```
<li data-socio="10784">Juan Carlos, Vitalicio.</li>
```

```
<li data-socio="97865">Roberto Gomez, Juvenil.</li>
```

```
<li data-socio="45732">Juana Martinez, Juvenil.</li>
```

```
</ul>
```

CSS

```
.output {
```


font: 1rem 'Fira Sans', sans-serif;

}

h1 {

margin: 0;

}

ul {

margin: 10px 0 0;

}

li {

position: relative;

width: 200px;

padding-bottom: 10px;

}

li:after {

content: 'N° Socio: ' attr(data-socio);

position: absolute;

top: -22px;

left: 10px;

background: black;

color: white;

padding: 2px;

```
border: 1px solid #eee;  
opacity: 0;
```

```
transition: 0.5s opacity;
```

```
}  
cod/>
```

```
li:hover:after {  
  opacity: 1;
```

```
}  
cod/>
```

dir

Es un atributo enumerado que indica la direccionalidad del texto del elemento. Puede tener los siguientes valores:

ltr, significa left to right y se utiliza para idiomas que se escriben de izquierda a derecha (como el Español).

rtl, significa right to left y se utiliza para idiomas que se escriben de derecha a izquierda (como el Árabe);

auto, permite que el navegador decida utilizando un algoritmo básico, que analiza los caracteres dentro del elemento hasta que encuentra un caracter con una direccionalidad fuerte, a continuación, aplica la direccionalidad a todo el elemento.

HTML

```
<p dir="rtl">Esto está en español pero no está bien alineado.</p>
```


<p dir="ltr">Esto está en español y está bien alineado .</p>

<hr>

<p>هذه الفقرة باللغة العربية ولكن بشك خاطئ من اليسار إلى اليمين</p>.

dir="auto"> <p>هذه الفقرة باللغة العربية ، لذا يجب النقال من اليمين إلى اليسار</p>.

CSS

.output {

font: 1rem 'Fira Sans', sans-serif;

letter-spacing: 1px;

}

hidden

Es un atributo Booleano que indica si el elemento aún no es, o ya no es, relevante. Por ejemplo, puede usarse para ocultar elementos de la página que no pueden usarse hasta que el proceso de ingreso se complete. El navegador no mostrará ni renderizará dichos elementos.

HTML

<p hidden>Este texto está oculto</p>

<p >Este texto se puede ver </p>

CSS

```
.output {  
  font: 1rem 'Fira Sans', sans-serif;  
}  
  
p {  
  background: #ffe8d4; border:  
  1px solid #f69d3c; padding:  
  5px;  
  border-radius: 5px;  
}
```

id

Define un identificador único (ID) el cual debe ser único en todo el documento. Su propósito es identificar el elemento para distinguirlo en programación de JavaScript o definir un estilo CSS.

HTML

<p>Este es un párrafo normal.</p>

<p id="resaltar">Este párrafo queremos que se note mucho!</p>

CSS

```
.output {  
  font: 1rem 'Fira Sans', sans-serif;  
}
```



```
#resaltar {  
  background: linear-gradient(to bottom, #ffe8d4, #f69d3c);  
  border: 1px solid #696969;  
  padding: 10px; border-  
radius: 10px;  
  box-shadow: 2px 2px 1px black;  
}
```

```
#resaltar:before {  
  content: "i "; margin-  
right: 5px;  
}
```

lang

Participa en la definición del lenguaje del elemento, refiriéndonos exclusivamente al idioma en que son escritos estos elementos. El tag contiene un solo valor en el formato definido.

HTML

<p>Párrafos en diferentes idiomas.</p>

<p lang="en-GB">This paragraph is defined as British English.</p>

<p lang="fr">Ce paragraphe est défini en français.</p>

CSS

```
.output {
```



```
font: 1rem 'Fira Sans', sans-serif;
```

```
}
```

```
p::before {
```

```
padding-right: 5px;
```

```
}
```

```
[lang="en-GB"]::before {
```

```
content: url('imagen de una bandera british-flag.png');
```

```
}
```

```
[lang="fr"]::before {
```

```
content: url('imagen de una bandera french-flag.png');
```

```
}
```

Puede consultarse la lista internacional de idiomas definidos por región en:

<https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry>

style

Contiene declaraciones de estilo CSS para ser aplicadas al elemento.

Agencia de
Aprendizaje
a lo largo
de la vida

Siempre es recomendado que los estilos sean definidos en archivos separados. Este atributo y el elemento <style> tienen el objetivo principal de permitir un estilizado rápido, por ejemplo con fines de testeo o pruebas.

HTML

```
<div style="background: #ffe7e8; border: 2px solid #e66465;">
```

```
<p style="margin: 15px; line-height: 1.5; text-align: center;"> Esta  
es la primer línea del párrafo y ahora un salto de línea<br>
```

```
esta segunda línea sigue siendo aplicada por el style aunque <b>esto debe hacerse con  
CSS.</b></p>
```

```
</div>
```

CSS

```
.output {  
  font: 1rem 'Fira Sans', sans-serif;  
}
```

tabindex

Es un atributo que recibe un parámetro con un número entero y que indica si el elemento puede obtener el foco del cursor y si debe participar de la navegación secuencial con el teclado. Si se cumplen estas condiciones, también determina en qué posición estará ordenado según la tabulación.

Puede tomar diferentes valores:

- un valor negativo significa que el elemento debe ser “focusable”, pero no debe ser alcanzado vía la navegación secuencial del teclado.
- Un cero significa que el elemento puede obtener el foco y alcanzable vía la navegación secuencial del teclado, pero el orden relativo es definido por la convención de la plataforma.
- un valor positivo que significa que puede obtener el foco vía la navegación secuencial del teclado, el orden relativo es definido por el valor del atributo y sigue la secuencia en orden ascendente según el valor de tabindex.

Si varios elementos comparten el mismo tabindex, su orden relativo sigue su posición en el documento.

HTML

<p>Hacé click en cualquiera de estos componentes.</p>

<label>Primer elemento:<input type="text"></label>

<div tabindex="0">Segundo elemento</div>

<div>Elemento no indexado</div>

<label>Tercer elemento<input type="text"></label>

CSS

```
.output {  
  font: .9rem 'Fira Sans', sans-serif;  
}
```



```
p {  
  font-style: italic;  
}  
  
div, label  
{  
  display: block; letter-  
  spacing: .5px;  
  margin-bottom: 1rem;  
}  
  
div:focus {  
  font-weight: bold;  
}
```

title

Contiene un texto que representa información de asesoramiento en relación al elemento al que pertenece. Dicha información puede típicamente, pero no necesariamente, presentarse al usuario como un tooltip.

HTML

<p>Estamos viendo las siguientes tecnologías:

<abbr title="Hypertext Markup Language">HTML</abbr>,

<abbr title="Cascading Stylesheets">CSS</abbr>, and
JavaScript.</p>

CSS

```
.output {  
  font: 1rem 'Fira Sans', sans-serif;  
}
```

translate

Es un atributo enumerado que se utiliza para especificar si los valores de los atributos de un elemento son traducidos cuando su página es localizada, o si hay que dejarlos sin cambios. Puede tener los siguientes valores:

- ❑ cadena vacía y "yes", indica que el elemento será traducido.
- ❑ "no", indica que el elemento no será traducido.

Ejemplo:

En este ejemplo, el atributo se utiliza para pedir a las herramientas de traducción que no traduzcan la marca de la empresa en el pie de página.

```
<head>  
  <meta http-equiv="Content-Language" content="es"/>
```

```
</head>
```

```
<html>
```

```
<body>
```

esta página está en idioma español

<footer>

<small>© 2020 <code translate="no">Brand Name</code>

</footer>

</body>

Especificidad

La **especificidad** es la manera mediante la cual los navegadores deciden qué valores de una propiedad CSS son más relevantes para un elemento y, por lo tanto, serán aplicados. La especificidad está basada en las reglas de coincidencia que están compuestas por diferentes tipos de [selectores CSS](#).

¿Cómo se calcula?

La especificidad es un peso (importancia o valor) que se le asigna a una declaración CSS dada, determinada por el número correspondiente de

cada [tipo de selector](#). Cuando varias declaraciones tienen igual especificidad, se aplicará al elemento la última declaración encontrada en el CSS. La especificidad sólo se aplica cuando el mismo elemento es objetivo de múltiples declaraciones. Según las reglas de CSS, en caso de que un elemento sea objeto de una [declaración directa](#), esta siempre tendrá preferencia sobre las reglas heredadas de su ancestro.

La siguiente lista de tipos de selectores incrementa en función de la

Tipos de selectores

Nota: La [proximidad de elementos](#) en el árbol del documento no tiene efecto en la especificidad.

1. [Selectores de tipo](#) (p.e., h1) y pseudo-elementos (p.e., ::before).
2. [Selectores de clase](#) (p.e., .example), selectores de atributos (p.e., [type="radio"]) y pseudo-clases (p.e., :hover).

3. Selectores de ID (p.e., #example).

El selector universal (*), los combinadores (+, >, ~, ', ||([en-US](#))) y la pseudo-clase de negación (:not()) no tienen efecto sobre la especificidad. (Sin embargo, los selectores declarados *dentro de* :not() si lo tienen.)

Para más información, visita "[Especificidad](#)" en "[Cascada y herencia](#)", también puedes visitar: <https://specifishity.com>

Los estilos *inline* añadidos a un elemento (p.e., style="font-weight:bold") siempre sobrescriben a cualquier estilo escrito en hojas de estilo externas, por lo que se puede decir que tienen la mayor especificidad.

Cuando se emplea important en una declaración de estilo, esta declaración sobrescribe a cualquier otra. Aunque técnicamente !important no tiene nada que ver con especificidad, interactúa directamente con esta. Sin embargo, el uso de !important es una **mala práctica** y debería evitarse porque hace que el código sea más difícil de depurar al romper la [cascada \(artículo en inglés\)](#) natural de las hojas de estilo. Cuando dos declaraciones en conflicto con el !important son aplicadas al mismo elemento, se aplicará la declaración con mayor especificidad.

Algunas reglas de oro:

- Busca **siempre** una manera de emplear la especificidad antes de considerar el uso de !important.
- Usa !important **sólo** en declaraciones específicas de CSS que sobrescriban CSS externo (de librerías externas como Bootstrap o normalize.css).
- **Nunca** uses !important cuando estés intentando escribir un plugin/mashup.
- **Nunca** uses !important en todo el código CSS.

En lugar de usar !important, considera:

1. Hacer un mejor uso de las propiedades en cascada de CSS.

2. Usar reglas más específicas. Indicando uno o más elementos antes del elemento que estás seleccionando, la regla se vuelve más específica y gana mayor prioridad:

3. `<div id="test">`

4. `Text`

5. `</div>`

6. `div#test span { color: green; }`

7. `div span { color: blue; }`

```
span { color: red; }
```

8. Como un caso especial sin sentido para (2), duplicar selectores simples para aumentar la especificidad cuando no tiene nada más que especificar

9. `#myId#myId span { color: yellow; }`

```
.myClass.myClass span { color: orange; }
```

Cómo se debería usar !important:

A) Sobrescribiendo los estilos en línea

1. Tienes un archivo CSS que establece aspectos visuales de tu sitio de manera global.
2. Tú (u otros) usan estilos inline en los propios elementos. Esto es considerado como una muy mala práctica.

En este caso, puedes establecer ciertos estilos en tu archivo CSS global como importantes, superando así los estilos en línea configurados directamente en los elementos.

Ejemplo del mundo real: Algunos **plugins jQuery** muy mal escritos que usan estilos inline.

B) Otro escenario:


```
#someElement p { color: blue;
```

```
}
```

```
p.awesome { color: red;
```

```
}
```

¿Cómo haces que los párrafos awesome se vuelvan siempre rojos, incluso los que se encuentren dentro de #someElement? Sin !important, la primera regla tendrá más especificidad y ganará a la segunda.

Cómo sobrescribir !important

A) Simplemente añade otra regla CSS con !important y, o bien da al selector una especificidad mayor (añadiendo una etiqueta, id o clase al selector), o añadiendo una regla CSS con el mismo selector en un punto posterior al ya existente. Esto funciona porque en caso de empate en especificidad, la última regla prevalece.

Algunos ejemplos con una gran especificidad:

```
table td {height: 50px !important;}
```

```
.myTable td {height: 50px !important;} #myTable td
```

```
{height: 50px !important;}
```

B) O añade el mismo selector después de uno existente:

```
td {height: 50px !important;}
```

C) O reescribe la regla original para evitar el uso de !important.

Para más información, visita (en inglés):

<http://stackoverflow.com/questions/3706819/what-are-the-implications-of-using-important-in-css>

<http://stackoverflow.com/questions/9245353/what-does-important-in-css-mean>

<http://stackoverflow.com/questions/5701149/when-to-use-important-property-in-css>

<http://stackoverflow.com/questions/11178673/how-to-override-important>

<http://stackoverflow.com/questions/2042497/when-to-use-important-to-save-the-day-when-working-with-css>

La pseudo-clase negación :not *no* es considerada una pseudo-clase para el cálculo de la especificidad. Pero los selectores colocados dentro de ella, si cuentan como selectores normales a la hora de determinar el valor de los [tipos de selectores](#).

Aquí tienes un pedazo de CSS:

```
div.outer p { color:orange;
}

div:not(.outer) p { color: lime;
```

<codoa codo/>

}

cuando se usa con el siguiente HTML:

```
<div class="outer">
```

```
<p>Esto está en el outer div.</p>
```

```
<div class="inner">
```

```
<p>Este texto está en el inner div.</p>
```

```
</div>
```

```
</div>
```

Debería aparecer en pantalla como:

Esto está en el outer div

Este texto está en el inner div

Agencia de
Aprendizaje
a lo largo
de la vida

Fuente: <https://developer.mozilla.org/es/docs/Web/CSS/Specificity>

Span y Div

span - abarcar. Es un contenedor en línea. Sirve para aplicar estilo al texto o agrupar elementos en línea. Sus etiquetas son: `` y `` (ambas obligatorias). Está definido como: Elemento especial, y por lo tanto en línea. Crea una caja: En línea. Puede contener: Texto, y/o Elementos en línea.

div de "division" -división. Sirve para crear secciones o agrupar contenidos. Sus etiquetas son: `<div>` y `</div>` (ambas obligatorias). Está definido como: Elemento en bloque. Crea una caja: En bloque. Puede contener: Texto, y/o cero o más elementos en bloque o en línea.

Tipo de caja

Para comenzar, hay que saber que cada etiqueta HTML tiene un tipo de representación visual en un navegador, lo que habitualmente se suele denominar el tipo de caja. En principio, se parte de dos tipos básicos: **inline** y **block**.

Valor	Denominación	Significado	Ejemplo
inline	Elemento en línea	El elemento se coloca en horizontal (un elemento a continuación del otro).	<code></code>

block	Elemento en bloque	El elemento se coloca en vertical (un elemento encima de otro).	<div>
-------	--------------------	---	-------

Obsérvese que por defecto, todos los elementos `<div>` son elementos de bloque (block) y todos los elementos `` son elementos en línea (inline). Para entender esto fácilmente, vamos a crear un HTML con 3 etiquetas `<div>` como las siguientes:

```
<div>Elemento 1</div>
```

```
<div>Elemento 2</div>
```

```
<div>Elemento 3</div>
```

A estas etiquetas HTML le vamos a aplicar el siguiente código CSS: `div {`

```
background: blue; color: white; margin: 1px;  
}
```

Con esto observaremos que en nuestro navegador nos aparecen 3 cajas azules colocadas en vertical (una debajo de otra) que cubren todo el ancho disponible de la página. Esto ocurre porque la etiqueta `<div>` es un elemento en bloque, o lo que es lo mismo, que tiene un tipo de representación block por defecto. Cada etiqueta HTML tiene un tipo de representación concreta.

Sin embargo, este comportamiento de elementos puede cambiarse con la propiedad CSS `display`. Tan sencillo como añadir `display: inline` en el ejemplo anterior y veremos como pasan a ser 3 cajas azules colocadas en horizontal (una al lado de la otra) que cubren sólo el ancho del contenido de cada una. Ahora los `<div>` de esa página son elementos en línea (el tipo de representación visual que tienen los ``).

Otros tipos de elementos

A medida que vamos cambiando el tipo de representación de estos elementos, nos damos cuenta que es insuficiente para realizar tareas y vamos necesitando más tipos de caja.

Vamos a rellenar un poco más la tabla, con las características más importantes de las opciones que puede tomar la propiedad CSS [display](#):

Tipo de caja	Características
block	Se apila en vertical. Ocupa todo el ancho disponible de su etiqueta contenedora.
inline	Se coloca en horizontal. Se adapta al ancho de su contenido. Ignora width o height .

inline- block	Combinación de los dos anteriores. Se comporta como inline pero no ignora width o height .
flex	Utiliza el modelo de cajas flexibles Flexbox . Muy útil para diseños adaptables.
inline-flex	La versión en línea (ocupa sólo su contenido) del modelo de cajas flexibles flexbox.
grid	Utiliza cuadrículas o rejillas con el modelo de cajas Grid CSS .
inline-grid	La versión en línea (ocupa sólo su contenido) del modelo de cajas grid css.
list-item	Actúa como un ítem de una lista. Es el comportamiento de etiquetas como .

table	Actúa como una tabla. Es el comportamiento de etiquetas como <code><table></code> .
table-cell	Actúa como la celda de una tabla. Es el comportamiento de etiquetas como <code><th></code> o <code><td></code> .
table-row	Actúa como la fila de una tabla. Es el comportamiento de etiquetas como <code><tr></code> .

Ocultar elementos

En la lista anterior, falta un valor de la propiedad `display`. Mediante la mencionada propiedad, es posible aplicar un valor `none` y ocultar completamente elementos que no queramos que se muestren, los cuales desaparecen por completo. Es muy útil para hacer desaparecer información cuando el usuario realiza alguna acción, por ejemplo.

Tipo de caja	Características
--------------	-----------------

none	Hace desaparecer visualmente el elemento, como si no existiera.
------	---

No obstante, también existe una propiedad CSS llamada [visibility](#) que realiza la misma acción, con la ligera diferencia de que no sólo oculta el elemento, sino que además mantiene un vacío con el mismo tamaño de lo que antes estaba ahí.

Dicha propiedad [visibility](#) tiene los siguientes valores posibles:

Valor	Significado
visible	El elemento es visible. Valor por defecto.
hidden	El elemento no es visible pero sigue ocupando su espacio y posición.

collapse	Sólo para tablas. El elemento se contrae para no ocupar espacio.
----------	--

Utilizar [visibility:hidden](#) es muy interesante si queremos que un elemento y su contenido se vuelva invisible, pero siga ocupando su espacio y así evitar que los elementos adyacentes se desplacen, lo que suele ser un comportamiento no deseado en algunas ocasiones cuando se aplica [display: none](#).

Otra opción interesante es utilizar la propiedad [opacity](#) junto a transiciones o animaciones, desplazarse desde el valor 0 al 1 o viceversa. De esta forma conseguimos una animación de aparición o desvanecimiento.

Etiquetas semánticas

En versiones anteriores a HTML5, al crear la estructura de una página, normalmente utilizamos etiquetas `<div>` para ir agrupando secciones de la página. Unido a esto, íbamos añadiendo `id` o clases (atributos) dependiendo de nuestro interés, para que quedase más claro. Una estructura como la que se menciona podría ser la siguiente:

```
<div class="articulo">
```

```
<h1>Título del artículo</h1>
```

```
<p class="intro">Pequeña introducción. </p>
```

```
<p class="contenido">Aquí va el texto del artículo en cuestión con sus detalles.
```

</p>

<p class="pie">Escrito por Manz</p>

</div>

Vemos que en esta estructura tenemos una agrupación que contiene todos los elementos de un artículo, donde el primer elemento es un encabezado <h1> (titular), luego un párrafo de introducción, seguido de un párrafo de contenido y un último párrafo a pie de artículo. La estructura podría ser más sencilla o más complicada, pero nos viene bien como ejemplo de introducción a este tema.

Nótese que los elementos utilizados <div> y <p> no tienen una semántica específica, salvo que son etiquetas de agrupación y que la segunda contiene un párrafo. En HTML5 se introducen una serie de etiquetas de agrupación que funcionan exactamente como un <div>, pero que además tienen un significado semántico porque indican la naturaleza del contenido que agruparán.

Veamos el ejemplo anterior, utilizando etiquetas semánticas:

<article>

<header>

<h1>Título del artículo</h1>

<p class="intro">Pequeña introducción.</p>

</header>

<p class="contenido">Aquí va el texto del artículo en cuestión con sus detalles.</p>

<footer>

<p>Escrito por Manz.</p>

</footer>

</article>

De esta forma, preparamos nuestro documento HTML para que cualquier navegador, robot de buscador o aplicación o sistema informático sea capaz de leer el documento HTML y conocer perfectamente la naturaleza del contenido de dicha sección.

Etiquetas semánticas

Veamos un listado de las etiquetas semánticas que se introducen en HTML5 (los encabezados ya existían en versiones anteriores):

Etiqueta	Descripción
<article>	Artículo. Parte principal de un escrito (<u>posts, mensaje en foros, comentario...</u>)

<nav>	Apartado de navegación (<u>enlaces de secciones, categorías, etc...</u>)
<header>	Cabecera visual de la página (<u>logotipo, título, etc...</u>). No confundir con <head>.
<h1>	Encabezado de nivel 1. Equivalente al título del documento.
<h2>	Encabezado de nivel 2. Equivalente al tema del documento.
<h3>	Encabezado de nivel 3. Equivalente a la sección de un tema.
<h4>	Encabezado de nivel 4. Equivalente a un apartado de la sección.

<h5>	Encabezado de nivel 5. Equivalente a un ejemplo del apartado.
<h6>	Encabezado de nivel 6. Equivalente a un subapartado del ejemplo.
<footer>	Pie de página de una sección (<u>o del documento completo</u>).
<section>	Sección o grupo temático de contenido. No usar sólo para dar estilo.
<aside>	Agrupación de contenido no relacionado con el tema principal del documento.
<address>	Agrupación con la información de contacto del autor del artículo o documento.

Nótese que las etiquetas `<article>` pueden contener otras etiquetas `<article>`, como por ejemplo comentarios dentro de artículos:

```
<article class="post">
```

```
<p>Este sería el texto del artículo. </p>
```

```
<article class="comentario">
```

```
<p>Aquí iría el texto del comentario. </p>
```

```
</article>
```

```
</article>
```

Práctica de Selectores

Selector de Tipo

Selecciona todos los elementos que coinciden con el nombre del elemento especificado. Ejemplo: `input` se aplicará a cualquier elemento `<input>`.

El selector de tipo CSS hace coincidir elementos por nombre de nodo. En otras palabras, selecciona todos los elementos del tipo dado dentro de un documento.

```
/* Todos los <a> elements. */ a {  
  color: red;  
}
```

Ejercicio

CSS

```
span {  
  background-color: skyblue;  
} HTML  
<span>Acá hay un texto con color de fondo</span>  
<p>Este es su segundo renglón</p>  
<span>Acá la publicidad de Codo a Codo.</span>
```

Selector de clase

Selecciona todos los elementos que tienen el atributo de class especificado. Sintaxis:
.classname

Ejemplo: .index seleccionará cualquier elemento que tenga la clase "index".

El selector de clases de CSS hace coincidir los elementos en función del contenido de su atributo.class

```
/* Todos los elementos de class="espacios" */
```

```
.espacios {  
  margin: 2em;  
}
```

```
/* All <li> elements with class="espacios" */ li.
```

```
espacios {  
  margin: 2em;  
}
```

```
/* Todos los elementos <li> con una clase que incluya ambos "espacios" y "elegante" */
```

```
/* For example, class="elegant retro spacious" */  
li.espacios.elegante {
```

```
  margin: 2em;  
}
```

Ejercicio

CSS

```
.red {  
  color: #f33;  
}
```

```
.yellow-bg {  
  background: #ffa;  
}
```

```
.fancy {  
  font-weight: bold;  
  text-shadow: 4px 4px 3px #77f;  
} HTML
```

```
<p class="red">Este texto es rojo.</p>
```

```
<p class="red yellow-bg">Este texto es rojo con fondo amarillo.</p>
```

```
<p class="red fancy">Este renglón es rojo con sombra.</p>
```

```
<p>Este renglón no tiene ningún formato.</p>
```


Selector de ID

Selecciona un elemento basándose en el valor de su atributo id. Solo puede haber un elemento con un determinado ID dentro de un documento.

Sintaxis: #idname

Ejemplo: #toc se aplicará a cualquier elemento que tenga el ID "toc".

El selector de ID de CSS coincide con un elemento en función del valor del id atributo del elemento . Para que el elemento sea seleccionado, su id atributo debe coincidir exactamente con el valor dado en el selector.

```
/* El elemento con id="demo" */  
#demo {  
    border: red 2px solid;  
}
```

Ejercicio

CSS

```
#confondo {  
    background-color: skyblue;  
}
```

HTML

```
<div id="confondo">Este renglón tiene color de fondo azul</div>
```

```
<div>Este renglón no tiene fondo.</div>
```

Selector universal

Selecciona todos los elementos. Opcionalmente, puede estar restringido a un espacio de nombre específico o a todos los espacios de nombres.

Sintaxis: `* ns|* *|*`

Ejemplo: `*` se aplicará a todos los elementos del documento.

El selector universal de CSS (`*`) coincide con elementos de cualquier tipo.

`/* Seleccionar todos los elementos */`

```
* {  
  color: green;  
}
```

A partir de CSS3, el asterisco se puede utilizar en combinación con namespaces:

- `ns|*` - coincide con todos los elementos en el espacio de nombres ns
- `*|*` - coincide con todos los elementos
- `|*` - coincide con todos los elementos sin ningún espacio de nombres declarado

Sintaxis: `* { style properties }`

El asterisco es opcional con selectores simples. Por ejemplo, `*.warning` y `.warning` son equivalentes.

Ejercicio

CSS

```
* [lang^=en] {  
  color: green;  
}
```

```
*.warning {  
  color: red;  
}
```

```
*#maincontent { border: 1px  
  solid blue;  
}
```

HTML

```
<p class="warning">  
  <span lang="en-us">Esto es verde</span>en una linea roja.  
</p>  
<p id="maincontent" lang="en-gb">  
  <span class="warning">Esto es rojo</span> en una linea verde.  
</p>
```

Selector de atributo

Selecciona elementos basándose en el valor de un determinado atributo.

Sintaxis: [attr] [attr=value] [attr~=value] [attr|=value] [attr^=value] [attr\$=value] [attr*=value]

Ejemplo: [autoplay] seleccionará todos los elementos que tengan el atributo "autoplay" establecido (a cualquier valor).

El selector de atributos CSS hace coincidir los elementos en función de la presencia o el valor de un atributo determinado.

```
/* <a> elementos que tengan una a */ a[title] {  
  color: purple;  
}
```

```
/* <a> elementos que vayan a la url "https://ejemplo.com.ar" */  
a[href="https://ejemplo.com.ar"] {  
  color: green;  
}
```

```
/* <a> ejemplos que apunten a una url que contenga "ejemplo" */  
a[href*="ejemplo"] {  
  font-size: 2em;  
}
```

```
/* <a> elementos que apunten a una url que termine en ".org" */  
a[href$=".org"] {  
  font-style: italic;  
}
```

```
/* <a> elementos que su atributo clase contenga la palabra "logo" */ a[class~="logo"] {  
  padding: 2px;  
}
```

Sintaxis

1. [attr] Representa elementos con un nombre de atributo de attr .

2. [attr=value] Representa elementos con un nombre de atributo de attr cuyo valor es exactamente value .
3. [attr~=value] Representa elementos con un nombre de atributo de attr cuyo valor es una lista de palabras separadas por espacios en blanco, una de las cuales es exactamente valor .
4. [attr|=value] Representa elementos con un nombre de atributo de attr cuyo valor puede ser exactamente value o puede comenzar con value seguido inmediatamente por un guión, -(U + 002D). A menudo se utiliza para coincidencias de subcódigo de idioma.
5. [attr^=value] Representa elementos con un nombre de atributo de attr cuyo valor está prefijado (precedido) por valor .
6. [attr\$=value] Representa elementos con un nombre de atributo de attr cuyo valor tiene el sufijo (seguido) por valor .
7. [attr*=value] Representa elementos con un nombre de atributo de attr cuyo valor contiene al menos una aparición de valor dentro de la cadena.
8. [attr operator value i] Agregar un i(o I) antes del corchete de cierre hace que el valor se compare sin distinción entre mayúsculas y minúsculas (para caracteres dentro del rango ASCII).
9. [attr operator value s] Agregar un s(o S) antes del corchete de cierre hace que el valor se compare entre mayúsculas y minúsculas (para caracteres dentro del rango ASCII).

Ejercicio

Enlaces
CSS

```
a {  
  color: black;  
}
```

```
/* Links internos que comienzan con "#" */  
a[href^="#"] {  
  background-color: gold;  
}
```



```
/* Links con "ejemplo" en la URL */  
a[href*="ejemplo"] {
```

```
    background-color: silver;
```

```
}
```

```
/* link con la palabra "sinmayusculas" en la URL, sin  
chequear mayúsculas */
```

```
a[href*="insensitive" i] {
```

```
    color: blue;
```

```
}
```

```
/* link con la palabra "ConMAyusculas" en la URL, con  
chequeo de mayúsculas */
```

```
a[href*="ConMAyusculas"] {
```

```
    color: orange;
```

```
}
```

```
/* Links que terminen en ".org" */
```

```
a[href$=".org"] {
```

```
    color: red;
```

```
}
```

```
/* Links que comienzan con "https" y terminan en ".org" */ a[href^="https"][href$=".org"] {
```

```
    color: green;
```

```
} HTML
```

```
<ul>
```


- Link Interno
- Link a ejemplo.com
- Sin chequear Mayusculas link
- Ejemplo org link
- ConMAyusculas link
- ejemplo https org link

Tipografías

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.

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>f</u> uente	Indica el nombre de la fuente (tipografía) a utilizar.
font-size		Indica el tamaño de la fuente.

font-style	normal italic oblique	Indica el estilo de la fuente.
font-weight	<u>p</u> eso	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 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)
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	Oblicua. Idem al anterior, salvo que esta inclinación se realiza de forma artificial.

Con la propiedad font-style podemos aplicar 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 (más gruesa). Generalmente, se incrementan en valores de 100 en 100.

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:

```
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.

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:

- **Gratis:** 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.

Google Fonts

Browse fonts

Featured

Articles

About



Open Sans

Designed by Steve Matteson

Download family

Select styles

Glyphs

About

License

Styles

Type here to preview text:

Almost before we knew it, we had left the gro.

Size: 30px

Light 300

Almost before we knew it, we had left the

Select this style

Light 300 italic

Almost before we knew it, we had left the gro

Select this style

Regular 400

Almost before we knew it, we had left the

Remove this style

Selected family

X

Review

Embed

To embed a font, copy the code into the <head> of your html

<link> @import

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

CSS rules to specify families

```
font-family: 'Open Sans', sans-serif;
```

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:wgh
t@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 fuentes seguras 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;  
}
```

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	Ejempl o
Palabra clave predefinida	<u>[palabra clave]</u>	red
Esquema RGB	rgb(<u>rojo</u> , <u>verde</u> , <u>azul</u>)	rgb(255, 0, 0)
Esquema RGB con canal alfa	rgba(<u>rojo</u> , <u>verde</u> , <u>azul</u> , <u>alfa</u>)	rgba(255, 0, 0, 0.25)
Esquema RGB hexadecimal	# <u>RRGGBB</u>	#ff0000
Esquema RGB hexadecimal con canal alfa	# <u>RGBBAA</u>	#ff000040

Esquema HSL	<code>hsl(<u>color</u>, <u>saturación</u>, <u>brillo</u>)</code>	<code>hsl(0, 100%, 100%)</code>
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>

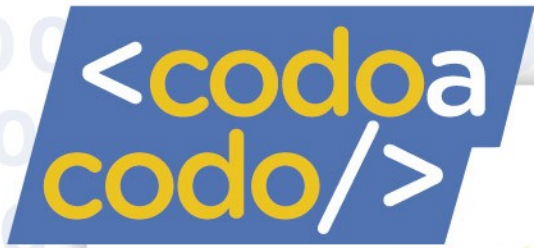
A continuación, iremos explicando cada uno de estos formatos para entender cómo se especifican los colores en CSS y utilizar el método que más se adapte a nuestras necesidades.

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
a
q
u
a
c
y
a
n
midnightblue
dodgerblue
lightseagreen
forestgreen
darkslategrey
limegreen
mediumseagreen
turquoise
royalblue
steelblue
darkslateblue
mediumturquoise
indigo
darkolivegreen
cadetblue
cornflowerblue
mediumaquamarine



dimgray
dimgrey

slategrey

lightslategray

mediumslateblue

lawngreen

chartreuse

aquamarine

maroon

purple

skyblue

lightskyblue

blueviolet

darkred

darkmagenta

saddlebrown

darkseagreen

lightgreen

mediumpurple

darkviolet

olive

gray

grey

rebeccapurple

palegreen

darkorchid

yellowgreen

sienna

brown

darkgray

darkgrey

lightblue

greenyellow

paleturquoise

lightsteelblue

powderblue

firebrick

darkgoldenrod

mediumorchid

rosybrown



<codoa
codoo/>

darkkhaki
silver
mediumvioletred
indianred
peru

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
lightp

Agencia de
Aprendizaje
a lo largo
de la vida

ink
pink
gold
peachpuff
navajowhite
moccasin
bisque
mistyrose
blanchedalm
ond
papayawhip
lavenderbl
ush
seashell
cornsilk
lemonchiff
on
floralw
hite
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 (*del 0 al 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*).

$$\begin{array}{c} \text{(60, 0, 0)} \\ \text{CANAL ROJO} \end{array} + \begin{array}{c} \text{(0, 25, 0)} \\ \text{CANAL VERDE} \end{array} + \begin{array}{c} \text{(0, 0, 200)} \\ \text{CANAL AZUL} \end{array} = \begin{array}{c} \text{(60, 25, 200)} \end{array}$$

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 clarificar:

Hexadecimal	Hex. abreviado	Color RGB	Palabra clave

Agencia de
Aprendizaje
a lo largo
de la vida

#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)

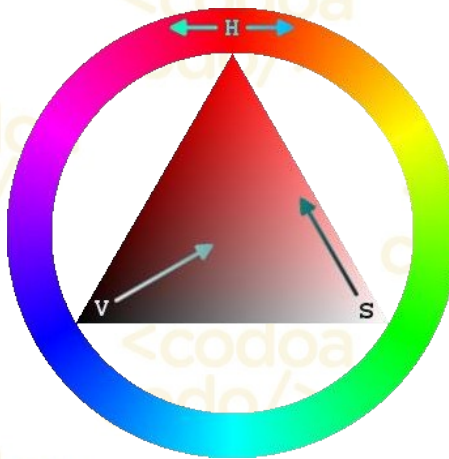
Veamos algunos ejemplos del formato hexadecimal (RGB abreviado):

```
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:
```

```
background-color: hsl(120deg, 25%, 75%);
```

```
background-color: hsl(5deg, 20%, 20%);
```

```
}
```

```
hsl(35deg, 0%, 100%);
```

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

```
}
```

Herramientas para hacer mezcla de colores:

- <https://color.adobe.com>
- <https://htmlcolorcodes.com/es/>

Medición y Unidades en CSS

Muchos de estos valores son unidades de medida (unidades CSS), por ejemplo, el valor del tamaño de un margen o el tamaño de la fuente. Las unidades de medida CSS se pueden clasificar en dos grupos, las relativas y las absolutas. Más la posibilidad de expresar valores en porcentaje.

Relativas: se llaman así porque son unidades relativas al medio o soporte sobre el que se está viendo la página web, que dependiendo de cada usuario puede ser distinto, puesto que existen muchos dispositivos que pueden acceder a la web, como ordenadores o teléfonos móviles. En principio las unidades relativas son más aconsejables, porque se ajustarán mejor al medio con el que el usuario está accediendo a nuestra web. Son las siguientes:

- Fuente actual: em es la fuente que se está trabajando por defecto, si la fuente es de 10 puntos y se coloca 2em se estaría trabajando con una fuente de 20 puntos.
- Altura de la letra: ex 1ex sería igual a la altura de la letra de la fuente actual del usuario.
- Píxeles: px varían su tamaño real en base a la resolución de la pantalla del usuario.

Absolutas: las unidades absolutas son medidas fijas, que deberían verse igual en todos los dispositivos. Como los centímetros, que son una convención de medida internacional. Pese a que en

principio pueden parecer mejores, puesto que se verían en todos los sistemas igual, tienen el problema de adaptarse menos a las distintas particularidades de los dispositivos que pueden acceder a una web y restan accesibilidad a nuestra web.

Puede que en tu ordenador 1 centímetro sea una medida razonable, pero en un móvil puede ser un espacio exageradamente grande, puesto que la pantalla es mucho menor. Se aconseja utilizar, por tanto, medidas relativas.

- Puntos pt Un punto equivale a 1/72 pulgadas
- Pulgadas in
- Centímetros cm
- Milímetros mm
- Picas pc (equivale a 12 puntos)

Porcentaje: el porcentaje se utiliza para definir una unidad en función de la que esté definida en un momento dado. Imaginemos que estamos trabajando en 12pt y definimos una unidad como 150%. Esto sería igual al 150% de los 12pt actuales, que equivale a 18pt.

Flexibles (vw y vh o vmin y vmax): relativas al tamaño del viewport (viewport is the user's visible area of a web page).

Márgenes de página

La propiedad CSS margin establece el margen para los cuatro lados. Es una abreviación para evitar tener que establecer cada lado por separado con las otras propiedades de margen: margin-top, margin-right, margin-bottom y margin-left.

También se permiten valores negativos.

/* Aplica a todos los cuatro lados */

margin: 1em;

/* Vertical | Horizontal */

margin: 5% auto;

/* Arriba | Horizontal | Abajo */

margin: 1em auto 2em;

/* Arriba | Derecha | Abajo | Izquierda */

margin: 2px 1em 0 auto;

/* Valores globales */

margin: inherit; margin:

initial; margin: unset;

Valores

Acepta uno, dos , tres o cuatro valores de los siguientes:

<length>

Especifica un ancho fijo. Valores negativos son permitidos.

<percentage>

Un <percentage> relativo al ancho del bloque contenedor. Se permiten valores negativos. auto

auto es reemplazado por algún valor apropiado. Por ejemplo, puede usarse para centrar horizontalmente un elemento bloque.

div { width:50%; margin:0 auto; } centrará el div horizontalmente. A

considerar:

- Un único valor aplicará para todos los cuatro lados.
- Dos valores aplicarán: El primer valor para arriba y abajo, el segundo valor para izquierda y derecha.
- Tres valores aplicarán: El primero para arriba, el segundo para izquierda y derecha, el tercero para abajo.
- Cuatro valores aplicarán en sentido de las manecillas del reloj empezando desde arriba. (Arriba, derecha, abajo, izquierda)

Sintaxis formal

[<length> | <percentage> | auto]{1,4}

HTML

```
<div class="ex1">
```

```
margin: auto;
```

```
background: gold;
```

```
width: 66%;
```

```
</div>
```

```
<div class="ex2">
```

```
margin: 20px 0 0 -20px;
```

```
background: gold;
```

```
width: 66%;
```


</div>

CSS

.ex1 {

margin: auto;

background: gold;

width: 66%;

}

.ex2 {

margin: 20px 0px 0 -20px;

background: gold;

width: 66%;

}

Otro ejemplo

margin: 5%; /* 5% para todos los lados */

margin: 10px; /* 10px para todos los lados */

margin: 1.6em 20px; /* 1.6em arriba y abajo, 20px izquierda y derecha */

margin: 10px 3% 1em; /* 10px arriba, 3% izquierda y derecha, 1em abajo */

margin: 10px 3px 30px 5px; /* 10px arriba, 3px derecha, 30px abajo, 5px izquierda */

margin: 1em auto; /* 1em arriba y abajo, centrado horizontalmente */

margin: auto; /* 0px de margen vertical, centrado horizontalmente */

Centrado horizontal con margin: 0 auto;

Para centrar algo horizontalmente en navegadores modernos, se utiliza flex, una tecnología más moderna con la sentencia:

display: flex; justify-content: center; Aunque esto lo veremos más adelante.

La propiedad position de CSS especifica cómo un elemento es posicionado en el documento. Las propiedades top, right, bottom, y left determinan la ubicación final de los elementos posicionados.

Fuentes de iconos

Qué son las fuentes de iconos

Las fuentes de iconos son fuentes (tipos de letra) en los que cada carácter es un icono que se han hecho populares en los últimos años.

- Recopilación de sitios web con iconos: <https://css-tricks.com/flat-icons-icon-fonts/>
- Font Awesome: <https://fontawesome.com/> - [Font Awesome releases](#)
- Weather icons: <http://erikflowers.github.io/weather-icons/> - [Descargar weather icons](#)
- Explicación sobre el uso de fuentes de iconos: <https://gomakethings.com/icon-fonts/>

Pese a su popularidad, presentan aspectos criticables, como se comenta en [Seriously, Don't Use Icon Fonts](#) (noviembre 2015).

Usar fuentes de iconos

Si a una etiqueta se le asocia la fuente que contiene los iconos, se muestran los iconos correspondientes. Como los iconos suelen estar colocados en caracteres más allá de los habituales (letras, números, etc.), para referirse a ellos se debe utilizar su código Unicode.

El ejemplo siguiente muestra los primeros iconos de una de las fuentes de iconos más populares: [Font Awesome](#)

`<p>a b c d e f g ...</p>`

```
<p>&#xf000; &#xf001; &#xf002; &#xf003; &#xf004;  
&#xf005; &#xf006; &#xf007; &#xf008; &#xf009;<br>  
&#xf010; &#xf011; &#xf012; &#xf013; &#xf014;  
&#xf015; &#xf016; &#xf017; &#xf018; &#xf019;<br>  
...</p>
```



```
@font-face {  
  font-family: "FontAwesome";  
  src: url("fontawesome-webfont.woff");  
}  
  
p {  
  font-family: "FontAwesome", sans-serif; font-  
  size: 200%;  
}
```

a b c d e f g ...



Para facilitar la inserción de iconos, se recurre a insertar los iconos mediante la hoja de estilo. La idea consiste en crear una clase con el nombre del icono y hacer que la hoja de estilo inserte el carácter mediante el pseudo

elemento `::before` (más adelante entraremos más en detalle en el tema pseudo- elementos). De esta manera no hace falta conocer el código Unicode del carácter, sino únicamente el nombre de la clase. Lógicamente, la clase se puede asignar a cualquier etiqueta.

```
<p class="fa-check-circle"> Me gusta</p>
```

<p> No me gusta</p> @font-face {

font-family: "FontAwesome";

src: url("fontawesome-webfont.woff");

}

p {

font-size: 200%;

}

.fa-times-circle::before {

font-family: "FontAwesome", sans-serif;

content: "\f057";

}

.fa-check-circle::before {

font-family: "FontAwesome", sans-serif;

content: "\f058";

}

- ☒ Me gusta
- ☐ No me gusta

Las fuentes de iconos se distribuyen con hojas de estilo que contienen todas las clases correspondientes a todos los caracteres para que sólo haya que enlazar la hoja de estilo para poder utilizar los caracteres.

Además se ha extendido la convención de insertar los iconos utilizando la etiqueta `<i>`, que en HTML 4 era una etiqueta desaconsejada. Aprovechando que en HTML 5 la etiqueta `<i>` se ha recuperado (aunque sea para etiquetar texto "especial", como el nombre de una especie animal, el nombre de un barco, etc.), los diseñadores web la están utilizando para las fuentes de iconos (supongo que porque "icono" empieza por la letra "i").

Nota: En FontAwesome se asignan dos clases, la clase general `fa` y la clase específica con el nombre del icono.

...

```
<link rel="stylesheet" href="css/font-awesome.css">
```

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

```
</head>
```



```
<body>
```

```
<p><i class="fa fa-check-circle"></i> Me gusta</p>
```

```
<p><i class="fa fa-times-circle"></i> No me gusta</p>
```

```
</body> p {  
  font-size: 200%;  
}
```

☑ Me gusta

⊗ No me gusta

Los iconos de una fuente de iconos se tratan como un carácter más, por lo que se le pueden aplicar las propiedades de hojas de estilo.

```
...
```

```
<link rel="stylesheet" href="css/font-awesome.css">
```

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

```
</head>
```

```
<body>
```

```
<p><i class="fa fa-check-circle verde"></i> Me gusta</p>
```

<codoa codo/>

<p><i class="fa fa-times-circle rojo"></i> No me gusta</p>

```
</body> p {  
  font-size: 200%;  
}
```

```
.rojo { color:  
  red;  
}
```

```
.verde { color:  
  green;  
}
```

✓ Me gusta

✗ No me gusta

Agencia de
Aprendizaje
a lo largo
de la vida

Al descargar una fuente de iconos, hay que fijarse en la estructura de carpetas que presuponen las fuentes de iconos. A menudo la hoja de estilo se encuentra en una carpeta **css** y la fuente en una carpeta **font** o **fonts**. Se puede cambiar la ubicación de ambas, pero hay que corregir la hoja de estilo para que tenga en cuenta la nueva ubicación. Si conservamos las carpetas **css** y **fonts**, lo lógico es guardar todas las fuentes utilizadas en la carpeta **fonts** y todas las hojas de estilo en la carpeta **css**.

Por otro lado, se puede:

- mantener la hoja de estilo de la fuente de iconos como una hoja de estilo separada (en ese caso nuestra página contendrá al menos dos enlaces a hojas de estilo, uno a la hoja de estilo de la fuente y otro a nuestra hoja de estilo)
Nota: Cuando se enlazan varias hojas de estilo simultáneamente no se debe incluir el atributo **title** en el enlace.
- se puede copiar y pegar en nuestra hoja de estilo la parte de la hoja de estilo de la fuente que necesitamos (las clases generales y las clases de los iconos utilizados en la página)

Para que las fuentes de iconos funcionen en la mayor cantidad de navegadores posible, las fuentes de iconos se distribuyen en varios formatos (woff, svg, ttf, eot, etc.) y la regla **@font-face** de la hoja de estilo hace referencia a todas ellas. En los ejercicios propuestos en este curso se utiliza únicamente la fuente en formato woff.

Fuente: <https://www.mclibre.org/consultar/amaya/css/css3-fuentes-iconos.html>